

**ASIA PACIFIC INSTITUTE OF INFORMATION
TECHNOLOGY**

**IN COLLABORATION WITH
STAFFORDSHIRE UNIVERSITY UK**

BEng (Hons) in Computing Science



FINAL SUBMISSION

Predictive Analysis of the U.S. Elections using Machine Learning

Prepared By:

Harindu Kodituwakku - CB005709

HF1631COMBEng

Date of Submission

2016/11/21

Supervisor

Mr. Nipunu Wijesinghe

Acknowledgements

I would like to thank my final year project supervisor Mr. Nipunu Wijesinghe for all the advice and support he has given me. Without his constant guidance this research and review would not have been a success.

I would also like to thank Mr. Javed Ahsan and all the other lecturers and academic staff at APIIT Sri Lanka who helped and encouraged me throughout this project. And also the APIIT Library staff for providing necessary assistance for this research.

Table of Contents

Acknowledgements	2
Table of Contents	3
List of Figures	8
List of Tables	10
INTRODUCTION	12
1. PROBLEM ANALYSIS	13
1.1 Problem Background.....	13
1.2 Problem Statement	13
2. PROJECT OUTLINE	15
2.1 The Solution	15
2.2 Aims and Objectives	15
2.3 Scope of the Project.....	15
2.4 Assumptions	16
2.5 Constraints.....	16
2.6 Deliverables.....	17
3. METHODOLOGY	18
3.1 Requirement Analysis	18
3.2 Research.....	20
3.3 Software Architecture and Design	21
3.4 Implementation	22
3.5 Software Testing	22
3.6 Performance Evaluation.....	23

4. REQUIREMENT ANALYSIS	24
4.1 Requirement Capture	24
4.2 Functional Requirements	24
4.3 Non-Functional Requirements	26
4.4 Hardware Requirements.....	27
4.5 Software Requirements	27
4.6 Use Case Diagram.....	28
5. LITERATURE REVIEW	29
5.1. Sentiment Analysis	29
5.1.1. Document Level sentimental analysis	30
5.1.2. Sentence level sentimental analysis	30
5.1.3. Aspect / Feature level sentimental analysis	31
5.2 Data Cleaning and pre-processing	32
5.3 Feature Extraction in sentiment classification	34
5.3.1 Term presence and frequency	34
5.3.2 Parts of speech Tagging (POS)	36
5.3.3 Negation	36
5.3.4 Topic-oriented features	37
5.4 Sentiment Classification techniques	38
5.4.1 Lexicon Based Approach	39
5.4.2 Machine Learning Approach	40
5.4.3 Supervised machine learning techniques	40
5.4.4 Probabilistic Classifiers	41
5.4.5 Naïve Bayes Classifier (NB).....	41
5.4.6 Maximum Entropy Classifier (ME)	44

5.4.7 Linear Classifiers	45
5.4.8 Support Vector Machines Classifiers (SVM)	46
5.4.9 Neural Network Classifiers.....	47
5.4.10 Decision Tree Classifiers	48
5.4.11 Rule based classifiers	48
5.5 Similar Approaches.....	49
5.5.1 Political sentimental analysis classification.....	49
5.5.2 General sentimental analysis classification	55
6. SOLUTION CONCEPT	57
6.1 Selection of Social Media	57
6.2 Selection of Sentiment Classification Technique	58
6.3 Data cleaning	61
6.4 Feature Extraction.....	63
6.5 Gathering Training data	64
6.6 Selection of the Programming Language.....	65
6.7 Selection of the Natural Language Processing Tool.....	66
6.8 Selection of the Database.....	67
7. DESIGN.....	69
7.1 Activity Diagrams.....	69
7.1.1 Overall Activity Diagram	69
7.1.2 Extract Features Activity Diagram	70
7.1.3 Train Classifier Activity Diagram	71
7.1.4 Generate Word Cloud Activity Diagram	71
7.1.5 Live Sentiment Analysis and Visualization activity diagram.....	72
7.2 Overall Class Diagram.....	73

7.2 Sequence Diagrams.....	74
7.2.1 Feature Extraction.....	74
7.2.2 Train Classifier.....	75
7.2.3 Generate Word Cloud	76
7.2.4 Live Sentiment Analysis and Visualization.....	77
8. IMPLEMENTATION.....	78
8.1 Implementation of the NoSQL Database	79
8.2 Implementation of the Twitter streamer	80
8.3 Pre-processing of tweets.....	83
8.3.1 Cleaning Tweets.....	83
8.3.2 Generating Feature List.....	84
8.4 Classification Algorithm.....	
8.4.1 Live Tweet Analyser Module.....	91
8.4.2 Word Cloud Map to Discover Underlying Topics of Tweets.....	93
8.5 Final Implementation of the Application.....	95
9 TESTING AND PERFORMANCE EVALUATION.....	96
9.1 Testing Environment.....	97
9.2 Unit Testing	97
9.2.1 Clean Tweets – Unit Tests	98
9.2.2 Extract Features – Unit Test	104
9.2.3 Generating Word Cloud – Unit Test.....	108
9.5 Integration Testing	110
9.5.1 Training Classifier – Integration Test.....	110
8.5.2 Word Cloud Generator – Integration Testing	112
9.4 Accuracy Testing	113

9.4.1 Accuracy Testing – Stage 1	113
9.4.2 Conclusion of the Accuracy Testing – Stage 1	115
9.4.3 Accuracy Testing – Stage 2	116
9.4.4 Conclusion of the Accuracy Testing – Stage 2	118
9.4.5 Accuracy Testing – Stage 3	119
9.4.6 Conclusion of the Accuracy Testing – Stage 3	120
9.4.7 Testing Live Sentiment Classifier.....	121
9.4.8 Testing the Word Cloud.....	127
9.5 Performance Evaluation.....	129
10 PREDICTIVE ANALYSIS	131
CRITICAL EVALUATION	138
CONTRIBUTION.....	141
SUGGESTED FUTURE ENHANCEMENT	143
CONCLUSION.....	144
REFERENCES	145
Appendix – A.....	149

List of Figures

Figure 1 - Use Case Diagram.....	28
Figure 2 - Sentimental Analysis Techniques	38
Figure 3 - Lexicon based Sentimental Architecture.....	38
Figure 4 - Sentimental Analysis Techniques.....	38
Figure 5 - Lexicon based Sentimental Architecture	39
Figure 6 - SVM Illustration.....	39
Figure 7 - Lexicon based Sentimental Architecture	39
Figure 8 - SVM Illustration.....	46
Figure 9 - Neural Network Architecture.....	46
Figure 10 - SVM Illustration	46
Figure 11 - Neural Network Architecture.....	47
Figure 12 - Figure 13 - Neural Network Architecture	47
Figure 14 - Dimensions with the polarity of the Sentiments	50
Figure 15.....	62
Figure 16 - Figure 9 - Effect of Pre-processing on Performance	62
Figure 17 - Popular software in Data Science	65
Figure 18 - Overall Activity Diagram.....	69
Figure 19 - Extract Features Activity Diagram.....	70
Figure 20 - Train Classifier Activity Diagram.....	71
Figure 21 - Generate Word Cloud Activity Diagram	71
Figure 22 - Live Sentiment Analysis Activity Diagram	72
Figure 23 - Overall Class Diagram	73
Figure 24 - Overall Class Diagram	73
Figure 25 - Extract Features Sequence Diagram	74
Figure 26 - Train Classifier Sequence Diagram	75
Figure 27 - Generate Word Cloud Sequence Diagram	76
Figure 28 - Live Sentiment Analysis - Sequence Diagram.....	77
Figure 29 - Initializing MongoDB	79
Figure 30 - Tweet as a JSON object in MongoDB	82
Figure 31 - Clean Tweets Class	83
Figure 32 - Replace Two or More method.....	84
Figure 33 - GetStopWordList method	85
Figure 34 - getFeatureVector Method.....	86

Figure 35 - Generate Bigrams method.....	87
Figure 36 - Train Naive Bayes Classifier	88
Figure 37 - Extract Features Algorithm	89
Figure 38 - Save Classifier as Pickle	90
Figure 39 - Classification of tweets	90
Figure 40- Live Twitter streamer.....	91
Figure 41 - Live Tweet Analyser.....	92
Figure 42 - Constructor of the WordCloud class	93
Figure 43 - Generating frequency of words.....	93
Figure 44 - Draw the Word Cloud using Matplotlib.....	94
Figure 45 - Predictive Analysis Dashboard	95
Figure 46- Main GUI of the solution	95
Figure 47 - Test Plan.....	96
Figure 48 - PyUnit Test Results.....	103
Figure 49 - PyUnit Test Results - 2	108
Figure 50 - Test Case No. 2	111
Figure 51 - CNN Tweet 2	115
Figure 52 - CNN Tweet 1	115
Figure 53 - Live Sentiment Analysis - Hillary Clinton.....	121
Figure 54 - Live Sentiment Analysis – Donald Trump.....	122
Figure 55 - Final Live Sentiment Analysis - Hillary Clinton	124
Figure 56 - Final Live Sentiment Analysis - Donald Trump	125
Figure 57 - Word Cloud.....	127
Figure 58 - Evaluating Accuracy of the Classifier.....	130
Figure 59 - Frequency of Tweets	131
Figure 60 - Official Twitter Statistics	132
Figure 61 - Final Presidential Debate Word Cloud	133
Figure 62 - Predictive Analysis - Hillary Clinton.....	135
Figure 63- Predictive Analysis - Donald Trump.....	136
Figure 64 - Clinton VS Trump Analysis.....	137

List of Tables

Table 1 - Functional Requirements.....	25
Table 2 - Classifier Accuracy	59
Table 3 - Evaluation parameters for Naive Bayes Classifier.....	60
Table 4 - Evaluation parameters for Support Vector Machine classifier.....	60
Table 5 - Feature to be reduced.....	61
Table 6 Feature Extraction Summary	63
Table 7- Comparison NLP Tools.....	66
Table 8- Remove Usernames Test Case	98
Table 9 - Remove URL Test Case	99
Table 10- Convert to lower case Test Case	100
Table 11 - Remove hashtags Test Case	101
Table 12 - Remove Additional spaces Test Case.....	101
Table 13 - Clean tweets final Test Case.....	102
Table 14 - Replace Repeating letters Test Case.....	104
Table 15 - Second implementation of remove repeating letters test case.....	105
Table 16 - Generate stop words list Test Case.....	106
Table 17 - Remove punctuations test case.....	107
Table 18 - Generate feature vector Test Case.....	108
Table 19 - Generate word Cloud Test Case.....	109
Table 20 - Training Classifier - Integration Test	111
Table 21 - Word Cloud Generator – Integration Testing.....	112
Table 22 - Accuracy Test 1	114
Table 23 - Accuracy Test 2.....	114
Table 24- Accuracy Test 3.....	116
Table 25 - Accuracy Test 4.....	117
Table 26 - Accuracy Test 5.....	119
Table 27 - Accuracy Test 6.....	119

List of Abbreviations

ML – Machine learning

NLP - Natural Language Processing

NB - Naïve Bayes

MaxEnt – Maximum Entropy

SVM – Support Vector Machines

SDM- Software Development Methodology

TDM- Traditional Development Methodologies

LDA - Latent Dirichlet Allocation

RDB – Relational Database

INTRODUCTION

Social media is becoming a huge social factor that can even impact elections. Monthly active U.S twitter users amounted to 65 million, as a percentage it's 23+% from the total population of the US (330 Million). A proper system isn't yet implemented to monitor, analyse and predict the outcomes of the elections from social media.

The proposed system will be very useful for many sectors of the society such as politicians, media, researchers and anyone who is interested in politics or election results. Processed data will be visually represented in order to understand it by the users even who doesn't have technical background. Vital decisions can be taken by the relevant users by considering predicted and analysed data.

Key chapters of this thesis are; ***Project outline*** where the solution, scope aims and objectives of the proposed systems is discussed, ***Methodology*** chapter discuss about the methods, standards which have been selected to this thesis, fourth chapter will be the ***Requirement analysis*** where it discuss the analysed requirements of the users and what is expected from the proposed system, Chapter five will be the ***Literature Review***. In the literature review key components are explored to build the proposed solution and already explored approaches to build the solution. The sixth and the last chapter is ***Solution Concept***. In this chapter, explored key components will be analysed and placed together to build the solution proposed with clear arguments.

1. PROBLEM ANALYSIS

1.1 Problem Background

Since the rapid development of the internet in the early 2000, by 2016 millions of people use internet for their daily needs. Currently 2.5 Exabyte of data is produced daily in the internet. Due to this phenomenon the famous buzz word '*Big Data*' is used by many developers to address this trend. Using this big data researchers have proved that it is possible to discover knowledge by performing certain actions also known as '*Data Mining*'. The usage of social media has impacted the society to share or publish their interests, opinions or even their private life. One of the biggest problems in the current society is the way of using these massive amounts of data for the betterment of the society.

The use of social media to explore the opinion of the users have been quite a popular topic for several years. But data mining has not been properly used to explore the predictability of social events. Though sentimental analysis has been used to discover the sentiments of statements, these techniques have not been majorly used to predict events such as Elections.

1.2 Problem Statement

Currently several manual techniques are used in predicting elections, such as conducting one to one interviews, conduct telephone interviews via landlines and cell phones. This proposed system will demonstrate a new dimension of the election predictions using Social media. Several researches have been published by analysing the correlation between social media, especially twitter and the results of the elections. But a proper system to analyse the sentiments of the public, trending keywords, discovery of underlying topics hasn't been developed yet.

Though there are systems that analyze the sentiments of the general topics, there are no system that is implemented purely to analyze elections. To achieve a higher accuracy in predicting the elections, the system should be modelled accordingly depending on the situation. Using this proposed system the public, politicians, media or any other party that is interested in elections could use to analyze the elections, trends or even the behavior of the general public.

The major events in the timeline of the U.S. Presidential Election 2016 are mentioned below.

2016

September 26 – First presidential general election debate.

October 9 – Second presidential general election debate.

October 19 – Third and final presidential general election debate.

These dates would be concerned specially during the collection of tweets.

2. PROJECT OUTLINE

2.1 The Solution

The proposed solution will be display the predictive results, trends, live tweet analysis of the frequently using hashtags and keyword where the backend of the proposed system is developed using machine learning algorithm with unigram+bigram feature selection. More than 75 % percent of accuracy could be gained using this approach. Proposed system analyzes data using twitter users' tweet, location, age and other information. The complete analyzation criteria is mentioned in the solution concept. Other than that outcome of this proposed project can provide live and long term sentiment trend analytics. On the other hand, this proposed system is focused on providing better visualizations to increase the understandability of sentiments for both technical and non-technical users.

2.2 Aims and Objectives

The system's main aim is to deliver a system that will visualize the predictive analysis of the U.S. Presidential elections. Which will be including the winning probability of the candidates and keyword analytic. This proposed project will also will depict the trending topics, key words, popularity visualization throughout the election period. The objectives of this proposed system includes to suggest that social media can be used to predict U.S. elections and can be considered as a mechanism to determine the political favourability towards presidential candidates.

2.3 Scope of the Project

This project is quite challenging as this involves advanced research areas prior to the implementation. The advanced research areas include Natural language processing, Sentimental Analysis, Data mining, Big Data Analysis and Predictive analysis. Complex data visualization techniques will also be required in the process. Thorough research on these areas have to be done to obtain high accuracy in the final analytics.

In order to perform a predictive analysis on elections, the tweets need to be classified to the relevant polarity which are generally positive, negative and neutral. Since the system should be trained to perform this task automatically, the use of Machine learning techniques are highly important. Proper testing phases should be conducted iteratively, in order to keep the machine learning classifier accurate. Machine learning testing concepts need to be researched and implemented. This proposed system will be designed only to classify the tweets which are published in English Language.

2.4 Assumptions

- The predictability of the final two candidates from Democratic and Republican are considered.
- Data required for this project will be collected in a certain time frame, using those data this application will be trained using machine learning techniques.
- A positive sentiment obtained from a tweet will be considered as a vote to the relevant candidate.

2.5 Constraints

- The system would only extract tweets for English language text.
- Opinion about political view may vary according to geographical location of the person. Geographic location of every tweet might not be enabled. Due to that categorizing according to location might contain less data.
- Sarcastic tweets might not be detected by the system. This is considered to be a drawback in classification of the tweets.

2.6 Deliverables

A desktop application that will be able to access by the users to view the predictive analytics, trending topics and keywords related to the elections. The proposed system is capable of analyzing the historical data as well as the real time data. This system can be used as a measurement in elections to determine the opinion rather than the traditional manual ways of forecasting elections.

This system will be able to use in any future U.S. Presidential election to predict the final sentiment of the people. Modern data visualization techniques will be used in representing the final analysis of the system.

3. METHODOLOGY

This chapter includes the methods, standards, tools and techniques which were used to perform the tasks such as Requirement Analysis, Research including primary and secondary research, Software architecture and design, Implementation, Software testing and Performance evaluation. Each selected standard and technique is justified in order to perform a thorough and accurate research on this methodology chapter.

3.1 Requirement Analysis

In the requirement analysis the author has thoroughly observed the current trends and ways of obtaining the requirements. The key requirements of the system is identified using various techniques in order to provide a satisfactory results for the system users. The core of the requirement gathering is to collect information about how users being used the existing systems and what improvements that the users are expected from the system. Requirement gathering can be differ from each system depending on the category of users. The proposed system is generally focused on the public, politicians, media, economic experts and any other personal who is interested in elections.

There are several ways of requirement analysis according to Hoffer et al. (1999);

- **Interviewing and Listening** – This is considered to be one of the primary ways of gathering requirements. Generally a large amount of people have to be interviewed to clearly understand the expectations out from the system. In interviewing there are several types of interviewing questions like, open-ended question and closed-ended questions. But close ended questions are quite easier since it contains true or false questions, multiple choice questions, rating response, ranking the order etc. There are guild lines in preparing interview questionnaires' in order to obtain the maximum information from the system users.

- **Observing users** - Though it is considered that interviewing people to be a primary way of gathering requirements, people are not always reliable in providing accurate information. Hoffer et al. (1999) claims that this is due to people cannot always be trusted to reliably interpret. This phenomenon can be exempted by observing the users. In the requirement analysis of this system this method has been used to discover the requirements. Users were mainly observed with the use of social media and internet to detect the requirements for such system.
- **Analyzing procedures and other documents** – Using this method, it is possible to get valuable insights of the system. Documents such as organization charts, mission statements, reports etc. can be used to determine information about problems with existing systems, different capabilities and priorities, rules for processing data. This methodology is also used in gathering requirements in this project.
- **Joint Applications Design (JAD)** - This method has a particular structure and similar to group interviews. There is a JAD session leader where the session is conducted. The leader is a trained in group management and he doesn't contribute any ideas or opinions during the session. This method was not used in this project.

After going through study on the similar approaches and how those systems collected information the author decided to use observing users and analyzing procedures and other documents as the methods for requirement analysis in this project. Most approaches have used analyzing procedure and the documents as the main source as this project enables slightly more theoretical aspects.

3.2 Research

After concluding a proper requirement analysis the author was able to identify the prime requirements of the system. A thorough research was performed to identify the technical concepts which are associated with this system. The main concepts need to build this system like; Machine learning techniques, data mining techniques, classification techniques etc. were discovered in the research. To obtain accurate information, sources like **IEEE Xplore Digital Library**, **Google Scholar**, **Staffordshire University Library**, **Association for Computing Machinery Library (ACM)**, **Science Direct** and many more renowned sources were used.

Books, journal articles, research papers and conference papers were thoroughly observed and referenced in order to conduct a quality research in this project. Research also includes fine similar approaches or related work which have been conducted by previous researchers. Exploring previous related work provides deep understanding about the concepts that they have used and the accuracy levels of each technique.

During the initial research, technical concepts such as natural language processing, sentiment analysis and data mining will be thoroughly studied using the above mentioned sources.

Guidance in the web articles, blogs and tutorials were followed in the implantation of visualization of the data. Twitter.com, Twitter API and the documentations were referred in order to stream tweets from the Twitter stream.

3.3 Software Architecture and Design

According to Wazlawick (2013), Unified Modeling Language (UML) is used to represent the vast literature and requirements of the systems in graphically. To illustrate the software architecture of this project UML is widely used. UML diagrams can be divided in to two categories.

- **Structured Diagrams** – Structure diagrams include class, component, objects, profile and deployment diagrams. These diagrams are used to illustrate what must be implemented in the system in terms of components. Structure diagrams are important to specify the part of the system architecture.
- **Behavior Diagrams** – Behavior diagrams include activity, use cases, sequence and state machine diagrams. These diagrams emphasize the process which will be taken inside the system. Using behavior diagrams it is reliable to demonstrate the functionalities of the system.

A Use case diagram will be included in the requirement analysis chapter, this diagram will elaborate the main functionalities of the proposed system. Afterwards the dataflow of the whole system will be illustrated by an overall Activity Diagram. This diagram will be derived under behavioral UML diagrams. Activity diagram will depict the flow of methods and passing of objects within the implemented classes.

Later the overall Class diagram and sequence diagrams will be included in the Design Chapter. The class diagram will demonstrate the classes that are implemented in the system. Relevant multiplicities will also be included in the class diagram. Sequence diagrams will depict the flow of methods and passing of objects within the implemented classes.

In the process of developing the system architecture and design all the diagrams may not be used but only the diagrams that are need to the project is recommended Wazlawick (2013). However use case, activity, class, sequence and component diagrams will be included in the Design chapter. **Object oriented concepts** were used in aid of the software design diagrams. UML diagrams are vital important to design before implementing the actual system as the diagrams give a wider view of the project.

3.4 Implementation

After the software architecture and design is completed the logic of the system is needed to be modeled. Before the actual implementation there were several test implementations to obtain experimental results from the scenarios. This would be a major plus point in the real implementation as it helps to obtain most important parameters. In this system several programming languages and APIs will be used. As for the backend or the core component, Python is decided to use with **JetBrains PyCharm 5.0 IDE**. PyCharm is considered to be one of the most efficient executor for Python language. Along with python, **Matplotlib** will be used to create the graphical user interfaces of the system.

As the first item of the implementation using Twitter API, the tweet streamer should be get connected to obtain tweets to be classified. Tweets will be stored as CSV files, most commonly used format in Big Data. In parallel large amount of training data should be collected to train the core algorithm. The core algorithms and methods will be implemented using Python as it supports more APIs and widely used in data mining. The implementation will be done iteratively, this is due to machine learning algorithms need to be fine-tuned to obtain accurate results.

3.5 Software Testing

As mentioned by Hoffer et al. (1999), regardless of the methodology the system is being made, once the implementation is begun software testing process can be started and proceed parallel. He also mentions whenever each component is coded, it can be tested individually and later test again as part of a larger program. The main purpose of testing is to confirm that the system delivers the requirements that are supposed to. Though testing will be started parallel to the implementation, planning for testing should be done earlier where it involves what needs to be tested and how to test. The training plan is to be finalized during the early stages of the implementation.

Though there are seven different types of test mentioned by the Hoffer et al. (1999), But the proposed system will be tested by using three main test types. **Unit testing** will be conducted to test each module individually. **PyUnit** testing framework which is the official unit testing

framework of python will be used in unit testing. This method is used as some parts of the system will be implemented way before the other components; e.g. Connecting with Twitter API. Since text cleaning and pre-processing functions implemented before polarity classification, these functions have to be integrated and tested later. To check the performance, **Integration testing** is proposed to use. Both unit testing and integration testing will be followed under '**White Box Testing**'. After bringing together all the components of the program, **Accuracy testing** will be used to critically test the whole system. Accuracy testing will be conducted in 3 stages, in order to achieve stable accuracy for the system. System testing will followed under the directions of '**Black Box Testing**'. In testing phrase of this proposed project, criteria such as response time, response to high volume of data (tweets), response to multiple inputs and so on will be tested. Under the black box testing, the User interfaces of the web application will be tested.

To test the non-functional features of the system **Stress Testing** will be carried under **Non-functional testing**. Automated testing tool for web applications such as '**Selenium**' is proposed to use in the testing phrase. After the proposed system is tested by the author, the completed system will be tested by the actual users. This testing phrase is known as "**Acceptance Testing**". In this phrase the system will be forced to fail in order to verify that recovery is properly executed; this will be known as '**Recovery testing**'.

3.6 Performance Evaluation

To check the performance of the classification machine learning algorithms which will be used in the proposed system, certain measurements are defined. **Precision** is used to measure the return of substantially more relevant results than irrelevant results returned by the algorithm. Precision provides the exactness of the classified results. High **Recall** means that the algorithm has returned most of the relevant results. Recall generally means the completeness of the results provided. **F1 measure** or the f measure is the weighted average of both precision and recall measurements.

The accuracy of the proposed system will be evaluated by testing with raw tweets. The tweets that are collected by the twitter stream will be divided in to categories; training data set and test data set to a ratio of 70:30. After the system is trained using the training data, system is tested using the remaining test data. More training data will increase the accuracy of the proposed system.

4. REQUIREMENT ANALYSIS

This chapter elaborates the functional requirements and the non-functional requirements of this system from the data which were collected by conducting observations of the users and analyzing related documents. Along with that this chapter includes the hardware, software and additional requirements to build this system.

4.1 Requirement Capture

Mainly the requirements were captured by analyzing documents which are related to this project. Initially it was found out that currently the predictions for the elections are calculated using the data gathered from telephone interviews, one to one interviews and etc. It was also noticed that there are websites that calculate the predictions using the bets placed by the public.

Most of the research papers that were written on Sentimental Analysis were useful finding the facts and requirements regarding this project. There were only few sentiment analysis research papers that were purely based on elections. These documents were useful in understanding the functional requirements of the proposed system.

4.2 Functional Requirements

Functional requirements can be considered as the core deliverable functions of the proposed system. These functions are consisted of what the user of this proposed system can perform in order to solve a problem. Generally a function means the outcome which is obtained after performing some actions using the provided inputs. Core functions in a system such as performing mathematical calculations, data processing, data visualizations and other crucial functionalities can be considered as '*Functional Requirements*'.

Functional Requirements	Response of the Proposed System
1.) User needs to be able to view the winning probabilities of final two candidates with the accuracy of over 75%.	Trend analysis graph will be generated using the historical data in the desktop application. Where the graph is sensitive to the sentiments of the tweets and change according to that.
2.) User should be able to detect the live sentiment of English tweets related to the election.	System should be able to clean and pre-process the tweet and classify the polarity using training data set. After that displaying the sentiments using live sentiment graph.
3.) User should be able to view the most frequent hashtags used related to the election.	System should be able to identify the most prominent hashtags and illustrate in a word cloud according to the frequency.
4.) User should be able to discover the most popular keywords used in the tweets.	The system should be able to generate a ‘Word Cloud’ using matplotlib. All the tweets need to be pre-processed and classify the sentiments before the word cloud generates.
5.) User should be able to view a timeline graph of the popularity of the final candidates.	The system needs a keep the trend record of the popularity percentage (positive and negative) with the timeline (month and date). System should update the trend graph according to the new sentiments and visualize the real time results.

Table 1 - Functional Requirements

4.3 Non-Functional Requirements

Non-functional requirements represent the qualitative features of the proposed system rather than quantitative features or the core functions. Non-functional requirements also play a major role in a system as it determines the usability to technical as well as non-technical users.

Reliability

- The accuracy of the probabilities should be over 75% to provide reliable information for the users.
- The application should be working properly without any crashes specially when generating data charts through the browser.

Usability

- In order to perform effectively, the desktop application should analyze and generate the required graphs and information efficiently. This is due to sometimes analyzation of tweets consume much time in the polarity classification process.
- English has been used as the language to perform the activities.
- Interactive graphical user interfaces have to be implemented to understand the data representations clearly. The users don't have to be technical persons to understand the outputs of the system.

Portability

- Since this system visualize the data in a desktop application, it should be possible to view the results in any desktop operating system.

The live updates should be classified and updates without refreshing the page.

4.4 Hardware Requirements

Hardware requirements provides the hardware resources that are needed for the development of this proposed system. Due to the process of machine learning algorithms significant amount of processing power is required to generate the final classifications. And also to store the streamed tweets and to perform visualizations on the classified data high amount of computational power is required.

- Processor: Intel Core i5-6300HQ CPU @ 2.30GHz (4 CPUs)
- RAM : 8192 MB
- HDD: 20 GB
- VGA: 4160 MB - NVIDIA GeForce GTX 960M

4.5 Software Requirements

This section includes the software that is required in the implementation and testing of the proposed system with help of above mentioned hardware components.

- Operating System: Windows 10
- Programming Language: Python 3.2, JavaScript, PHP
- Programming IDE: JetBrains PyCharm
- Software in designing: Visual Paradigm 12, Microsoft Project 2016
- Internet Browser: Google Chrome, Firefox, Microsoft Edge
- Software in documentation: Microsoft Word 2013

4.6 Use Case Diagram

The following use case diagram illustrates the functional and non-functional requirements which was mentioned above.

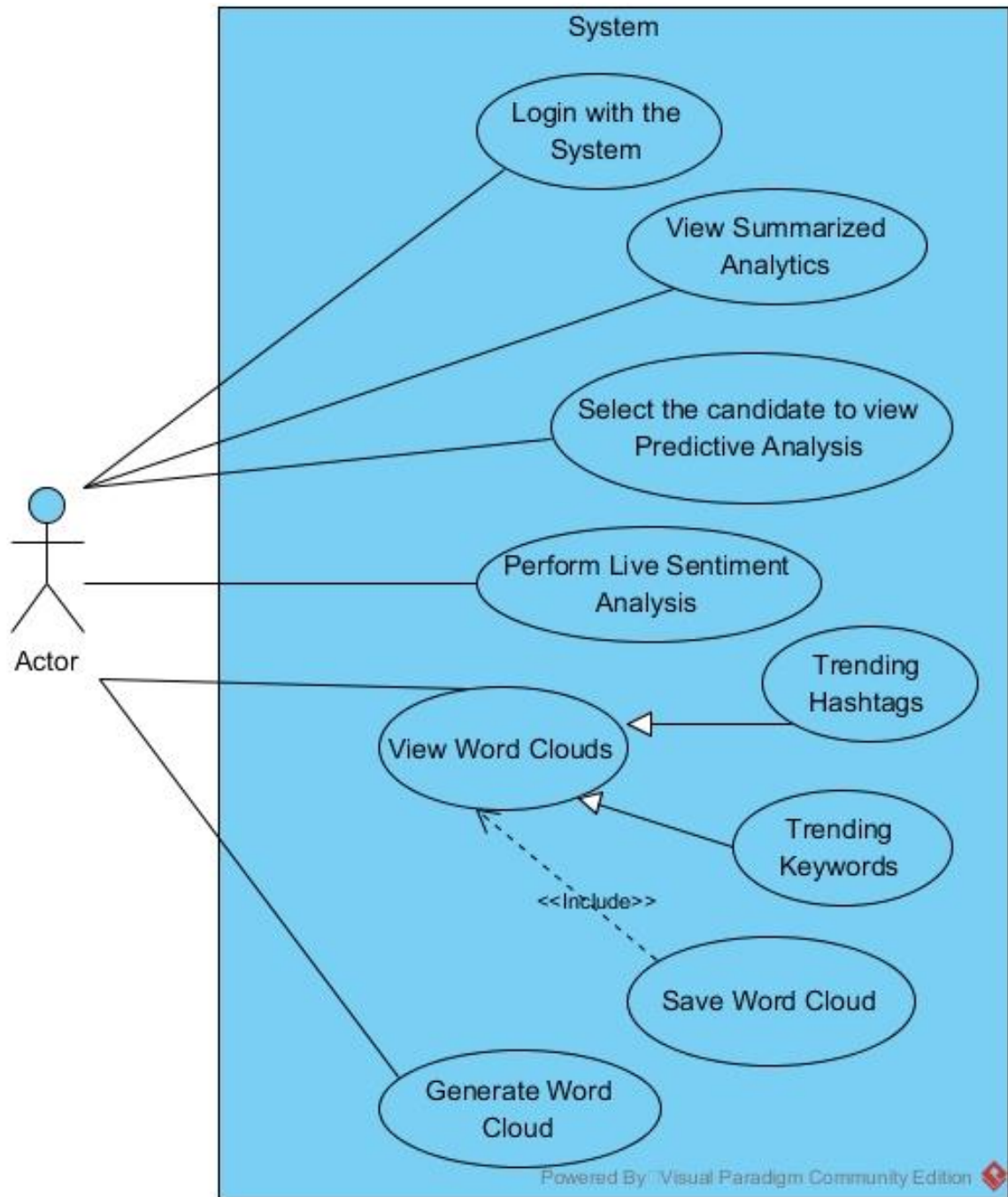


Figure 1 - Use Case Diagram

5. LITERATURE REVIEW

5.1. Sentiment Analysis

Sentimental analysis (SA) or Opinion mining (OM) is considered as a major ongoing field of research in natural language processing. Sentimental analysis is mainly focused on the study of people's opinion, attitudes and emotions towards a certain entity. It is considered that sentimental analysis and opinion mining has slightly contrasting meanings, where sentimental analysis identifies the sentiment that is mentioned in a text and then analyze while Opinion mining extracts and analyze user's opinion towards a certain entity. (Medhat et al., 2014)

According to Vohra & Teraiya (2013), sentimental analysis primarily classifies the opinions in the given text to polarity categories like "Positive" or "Negative" or "Neutral" depending upon the requirements of the problem. Due to the rise of Big Data evolution many research fields like Subjectivity detection, Sentiment Prediction, Aspect based sentiment summarization, Contrastive viewpoint summarization use SA to identify the opinions and emotions.

Sentimental analysis can be categorized into three main classification levels;

- Document Level sentimental analysis
- Sentence Level sentimental analysis
- Aspect / Feature Level sentimental analysis

5.1.1. Document Level sentimental analysis

According to Pang & Lee (2006), there are two aspects in document level SA namely, Single and Multi-document opinion-oriented summarization. In this approach the whole document will be analyzed first and then a sentiment will be generated to the whole document depending upon the polarities analyzed. Though this approach is mostly suitable for analyze the sentiment (positive, negative or neutral) of a whole documents or reviews about different products, people or services, it won't be able to identify the accurate sentiment if there are several contrasting opinions in the same document itself.

Due to this phenomenon Document level sentimental analysis is widely used in obtaining the sentiment of a whole document (mostly the sentiment about the topic of the document).

5.1.2. Sentence level sentimental analysis

Medhat et al. (2014) states that the sentence level sentimental analysis is used to classify sentiment expressed in each sentence in a document. The first step is to identify whether the given sentence is subjective or objective. If the sentence is considered to be subjective, sentence level sentimental analysis will provide positive, negative or neutral opinion. However it is stated that there is less difference between document and sentence level sentimental analysis as a sentence means a short document.

Though the sentence level sentiment accuracy levels are slightly higher than the document level sentimental analysis, this approach is unable to determine the correct sentiment of a sentence due to emoticons and may have contained several opinions in the same sentence.

5.1.3. Aspect / Feature level sentimental analysis

According to Medhat et al. (2014) aspect level sentimental analysis is targeted to classify the sentiment with respect to specific aspects of the entities. The first step would be to identify the entities in the given aspects. This approach is considered as the most accurate approach for determine sentiment with comparing to sentimental analysis levels discussed previously in this section. As an example if the following tweet is considered,

“I like Hilary’s personality but I don’t support Democrats”

When the above tweet is analyzed two sentiments or opinions can be identified, positive opinion and negative opinion respectively. Though the particular person likes Hilary Clinton as a person he or she might not vote her since she’s a Democrat. In aspect level analysis, *Hilary* and *Democrat* are considered as the entities with different aspects.

According to (Hu et al. 2004) research, aspect level analysis has two major components.

- **Regular opinions**

Opinions that are straightforward and express sentiment about a single entity are considered as Regular Opinions.

E.g.: *“Donald Trump will make America great again”*

The above mentioned tweet has a positive sentiment towards Donald Trump and the tweet only describes about him as an entity.

- **Comparative opinions**

Ganapathibhotla & Liu (2008) have stated that opinions which describe two or more entities in a single statement as Comparative opinions since the opinion might compare the entities.

E.g.: *“Berny Sanders has better foreign policies than Donald Trump”*

A positive sentiment towards Berny Sanders and a negative sentiment towards Donald Trump is expressed from the above tweet; which compares the two main entities of the statement.

5.2 Data Cleaning and pre-processing

Due to the 140 limitation of characters per tweet, it has varying and unpredictable nature of language used. Most of the tweets might be included with spelling mistakes or grammatical mistakes, acronyms or sometimes the usage of slangs. Because of these problematic issues the tweets that are collected to use should be cleaned and standardized. Data cleaning is considered as a vital stage in Data mining as the quality of the features extracted from the collected data or training data set directly affects the performance of the machine learning classifiers.

According to Gokulakrishnan et al. (2012) there are several components that are needed to be cleaned in a micro-blogging sites like Twitter before they are being used to extract features.

➤ **Emoticons or Emoji's**

Due to the introduction of new types of emoticons by various companies like Google, Apple and Microsoft to their users, there is a major trend in using emoticons in micro blog posts as an easy way of expressing emotions and feelings. As stated by Gokulakrishnan et al. (2012) there are over 30 emoticons such as :) :(:D =] :] =) =(=[to describe the positive and negative opinion or emotions.

Since emoticons are considered as noisy labels in training process, it might affect negative impact on the accuracies of the machine-learning classifiers like SVM and MaxEnt but according to the researches a little effect on the Naïve Bayes classifier due to the mathematical model and feature weight selections. (Go et al., 2009)

➤ **URL Extraction**

Around 35% of the tweets that are shared contain URLs in order to share more content than the given limited characters. According to Gokulakrishnan et al. (2012) though those external URLs might contain elaborated information regarding the emotions it will not be possible to crawl the particular URL for the content. Due to this phenomenon URLs are removed from the training data sets.

➤ **Lower case and Upper case Identifications**

In order to express powerful emotions in tweets people tend to use capital letter (E.g. DISASTER). This expression is called e-shouting, and considered to be a good indicator to determine the sentiment of the statement. Before removing the upper casing from the keyword it is identified as ALL_CAPS and extracts before text cleaning.

In some instances the tweets are consisted of both lower and upper casings. But to extract features correctly and to map all tokens corresponding to the features the keywords need to be in consistent case. (E.g. *SuPpOrtDoNaldTruMP*)

➤ **Detection of Usernames, Retweets and Hashtags**

In tweets there can be the symbols like @, RT, # to demonstrate the username or the users tagged, retweets and hashtags respectively. These feature could be identified as noisy labels in the statements. To avoid this, above mentioned symbols are replaced with <USER>, <RETWEET> and <HASHTAG>. (Gokulakrishnan et al., 2012)

➤ **Removal of stop words**

According to Gokulakrishnan et al.(2012) when deciding polarity of sentiments, it is standard to remove words that are extremely common (which has a high IDF value) since those types of words have no value to the classification process. Words such as *a*, *an*, and *the* are collectively called as stop words. Due to this reason such words are extracted from the statements.

➤ **Compression of words**

There can be users that uses informal language to elongate words to express strong expressions through the tweets such as, the term “Victoryyyyyyy”. That particular word phrase might provide higher degree of expression than “Victory”. Due to this phenomenon such words containing subsequent occurrence of the same letter or character, we have to reduce the sequence of the occurrence but keeping a record that the particular phrase contained higher expression than the average usage of the word.

5.3 Feature Extraction in sentiment classification

It is considered that the second phase of sentiment classification is extract and select the features from the text. The features can express the how the document, sentence is represented. Feature extraction can be used to reduce the complexity and provide simple representation of data representing each variable in feature space as a linear combination of original input variable as stated by Khalid et al. (2014). In text classification using machine learning approaches in general, there are extensive amount of features that need to be considered and feature extraction depends on the specific problem.

According to Pang & Lee (2008), there are several techniques that can be considered in feature extraction;

5.3.1 Term presence and frequency

In traditional information retrieval (IR) it is the standard to express a piece of text as a feature vector where the entries correspond to each individual terms. Term presence can be expressed as a binary valued feature vector where the entries indicate a term occurrence (value 0 if the word appears and value 1 if otherwise) form more efficient basis for review polarity classification than real valued feature vector in which entry value increase with the occurrence frequency of the related term. (Pang & Lee, 2008). Where the term frequency is determined by calculating the number of term the specific term has been mentioned.

Features like Individual words or word, uni-grams or n-grams and their frequency of presence have been widely used in sentiment classification. In sentimental analysis more researches have used N-gram feature rather than selecting the corpus. However there are several types of N-gram but it will differ with the number of words used.

N-gram

According to Cavnar et al. (1994), N-gram means an N-character slice of a long string. N-grams provides a simple and reliable way to categorize documents in wide range of classification tasks. *Unigrams*, *Bigrams* and *Trigrams* are the usual N-grams that are used in modern natural language processing.

Unigram

In unigrams each individual is word in the sentence is separated and considered as a Unigram.

E.g. *“Donald Trump is changing politics”*

Unigrams – [*“Donald”*, *“Trump”*, *“is”*, *“changing”*, *“politics”*]

Bigram

Bigrams are consisted of two words blocks of a sentence. From the beginning of the sentence two words are bring parried together to form a single bigram block and moves to the right side of the sentences performing the pairing.

E.g. *“I will vote Hilary Clinton”*

Bigrams – [{*“I”*, *“will”*}, {*“will”*, *“vote”*}, {*“vote”*, *“Hilary”*}, {*“Hilary”*, *“Clinton”*}]

Trigram

Likewise in paring words in Bigram, here three words are paired together in order for feature extractions. In general a string of length k , padded with blanks will have $k+1$ tri-grams. (Cavnar et al., 1994)

E.g. *“I will vote Hilary Clinton”*

Trigram – [{*“I”*, *“will”*, *“vote”*}, {*“will”*, *“vote”*, *“Hilary”*}, {*“vote”*, *“Hilary”*, *“Clinton”*}]

5.3.2 Parts of speech Tagging (POS)

In English language, main parts of speech can be considered as noun, pronoun, adjective, verb, adverb, preposition and conjunction. POS tagging is a commonly used feature extracting technique in data mining. According to Pang & Lee (2008) there is a high correlation between the adjectives and the subjectivity of a sentence in a text corpus. Due to that it is clear that adjectives are good indicators of sentiments and guide to perform the correct sentiment classification. Pang & Lee (2008) also stated that depending on the problem, the usage of POS tagging or N-gram selection would generate different accuracies.

In most of the cases adverbs, adjectives, verbs play a major role in feature extraction as they are important indicators of opinions. To identify whether it's an adverb or adjective a label is placed to identify the role in the grammatical context. Pang et al. (2002) has showed the usage of POS tagging clearly in his research;

“I love this movie” and “This is a love story”

Though the word love is a verb in the first sentence and an adjective in the second sentence, the polarities of the sentences are completely different. First sentence provides a Positive sentiment while the second provides a neutral sentiment. If these words were considered as unigrams, the polarities might be different compared to POS tagging.

5.3.3 Negation

Negation handling is a vital function in natural language processing as most of the tweets contain negation related words. When the users express something that is not true or is not the case, negative words, phrases or clauses can be used. Negation happens most commonly when the users use negative words such as *no*, *not*, *never*, *none*, *nobody* or *don't*, *doesn't*. According to Pang & Lee (2008) phrases like *“I like Donald Trump”* and *“I don't like Donald Trump”* are considered to be similar by commonly used similarity measures, and the only differing key component is the negation term *“don't”* which makes the two phrases opposite to each other.

Due to techniques in natural language processing, the negation might not be able to identify. Pang & Lee (2008) shows that due to this phenomenon attaching “**NOT**” to the words occurring closest to the negation term so that in the sentences, “*I don’t like Donal Trump*”, the token “*like*” will be converted to a new token “*like-Not*”. This approach will not be the only solution to address this given problem.

Another difficulty in modelling negations is that to detect sarcasm with often expressed with negation terms. To detect such instances the help of N-grams will be suitable. Due to the feature that negation terms can reverse the polarity of a corpus, it is an important factor to be considered in feature extraction process.

5.3.4 Topic-oriented features

According to Pang & Lee (2008) the interaction between the related topic and the sentiment play an important role in data mining. Opinion words and phrases are commonly used to express the opinion including good, bad, hate or like. Vohra & Teraiya (2013) state that statistical based or lexicon based approaches need to identify the semantic orientation of an opinion word.

Gokulakrishnan et al. (2012) suggest that the query term itself should not be used to decide the sentiment of the related topic or the post. If the query term is present, it should be replaced with a <QUERY> keyword.

5.4 Sentiment Classification techniques

In natural language processing, sentimental classification can be divided in to three main categories namely; Machine learning (ML) approach, Lexicon based approach and Hybrid approach. Depending on the research or the project of the sentiment analysis the classification approaches can be different. According to Medhat et al. (2014), Machine learning approaches are highly used in this area due to the higher accuracy rate of the outcomes. Machine learning algorithms use linguistic features to classify the sentiment. The lexicon based approach depends on the sentiment lexicons (preprocessed sentiment terms) while the Hybrid approach combines both approaches but more biased towards the lexicons.

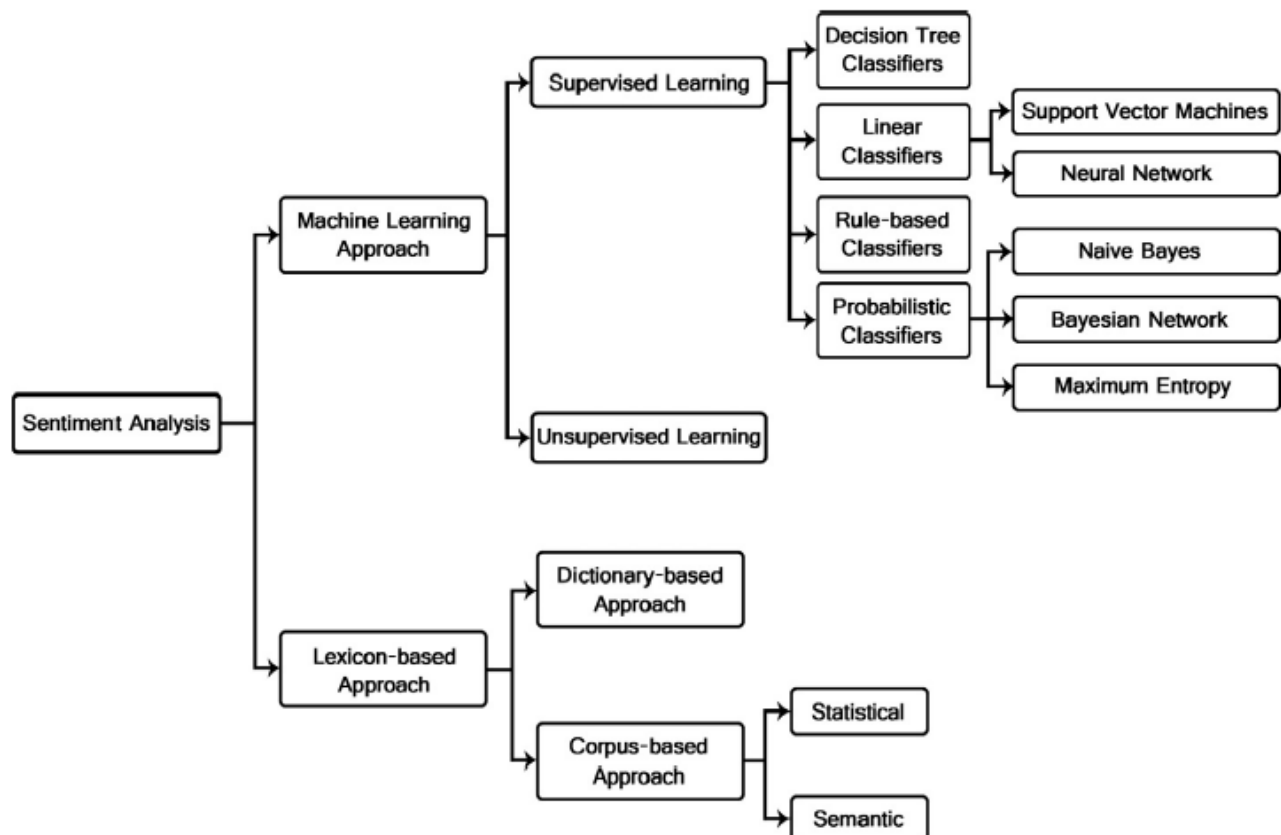


Figure 2 - Sentimental Analysis Techniques

Source: Medhat et al. (2014)

5.4.1 Lexicon Based Approach

Sentiment classification using Lexicon based approach can be categorized in to dictionary based approach and corpus-based approach and in corpus based approach two major classification techniques are Statistical and Semantic approaches. According to Medhat et al. (2014), due to the usage of sentiment lexicons opinions and idioms should be divided in to positive and negative opinions. There are two approaches to collect these opinion word list manually and automatically but the manual method is time consuming.

In dictionary based approach the opinion words are collected manually with the use of word libraries, thesaurus and check for the synonyms and antonyms. Then the newly found words are added to the training list, likewise this iteration is take place till the process is completed. Again manual inspection is needed to check the accuracy of the training set.

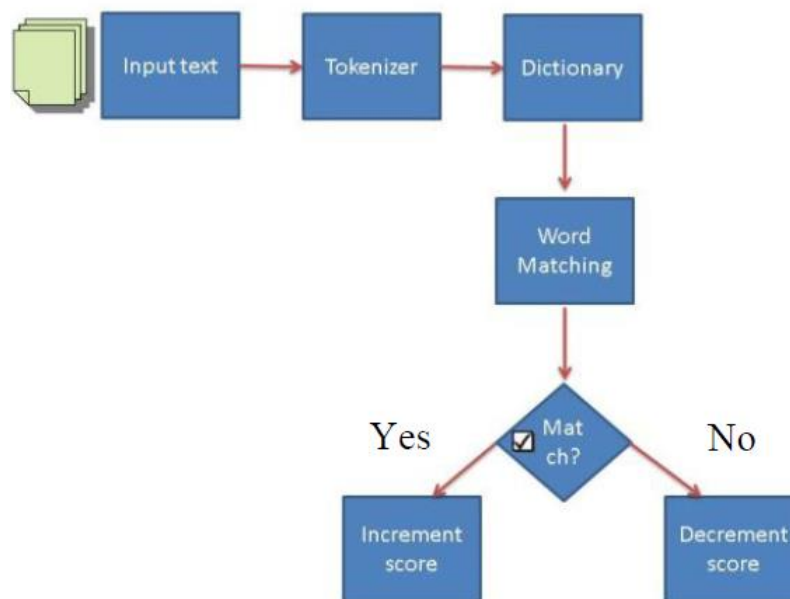


Figure 5 - Lexicon based Sentimental Architecture

5.4.2 Machine Learning Approach

Machine learning is considered to be the base of Artificial Intelligence (AI), where the system or the program itself learn and perform the given set of functions. Machine learning techniques are used in image recognition, optical character and speech recognition, search engine recommendations and many more. Text classification in machine learning approach can be further divided into Supervised and Unsupervised machine learning techniques. Vohra & Teraiya (2013) state that in machine learning techniques there are two types of data required; training data set and test data set. The polarity features and characters of corpus are learned by the automatic classifier using the training data set while the test data is used to monitor the accuracy of the machine learning classifier.

Generally in supervised learning approach, large amount of training dataset is required while unsupervised learning approach can be used if it is unable to find labeled training data set with polarity. In machine learning approaches there are four main stages according to Thakkar & Patel, (2015); Data collection, Data cleaning and preprocessing, Training data and classification.

5.4.3 Supervised machine learning techniques

Supervised machine learning technique completely depends on the properly labeled training data or documents by manually. By using these labeled training data set predictions or forecasts can be done. Supervised machine learning techniques can be considered as the most commonly used approach in many researches due to the better accuracy. Medhat et al. (2014) claimed that there are several subcategories in supervised machine learning;

- **Probabilistic Classifier**
- **Linear Classifier**
- **Decision Tree Classifier**
- **Rule Based Classifier**

5.4.4 Probabilistic Classifiers

According to Medhat et al. (2014), Mixture models are used by the Probabilistic classifiers in text classification. “The mixture models assumes that each class is a component of the mixture. Each mixture component is a generative model that provides the probability of sampling a particular term for that component.” These types of classifiers are also known as “*Generative classifiers*”.

There are three major probabilistic classifiers mentioned by Medhat et al. (2014);

- **Naïve Bayes Classifier (NB)**
- **Maximum Entropy Classifier (ME)**
- **Bayesian Network (BN)**

5.4.5 Naïve Bayes Classifier (NB)

Medhat et al. (2014) has claimed that the Naïve Bayes classifier as the simplest and the most commonly used machine learning algorithm. Though this algorithm is considered to be one of the oldest classification algorithm due to the robustness and the simplicity due to that it is surprisingly effective and efficient. Wikarsa & Thahir (2015) state that the Naïve Bayes algorithm is derived from Bayes theorem producing statistical classification based on probabilistic opportunities. In classification of tweets, it separates the tweets according to polarity; Positive, negative and neutral assuming all the features are conditionally independent.

According to Pang et al. (2002), one approach in text classification is that for a document d out of all classes c (where $c \in C$) and the classifier returns the class \hat{C} (^ represents the estimated correct class; Positive, Negative or Neutral) which derive the maximum posterior probability given the document.

$$\hat{C} = \operatorname{argmax}_c P(c / d)$$

$P(c / d)$ – The conditional probability of class c given the document d .

Using Bayes' Theorem;

$$\hat{C} = \operatorname{argmax}_c P(c / d) = \operatorname{argmax}_c P(d / c) P(c)$$

According to Jurafsky DanMartin (2000) the above mentioned equation can be represented using the likelihood and prior probability.

$$\hat{c} = \operatorname{argmax}_{c \in C} \overbrace{P(d|c)}^{\text{likelihood}} \overbrace{P(c)}^{\text{prior}}$$

Where the Posterior probability can be derived from the following relationship.

$$\textit{Posterior} \propto \textit{Prior} \times \textit{Likelihood}$$

According to Jahanbakhsh & Moon (2014), the above relationship can be formulated in to the following equation;

$$P(\textit{label} = l_j | \textit{tweet}) \propto P(\textit{label} = l_j) \times \prod_i P(w_i | \textit{label} = l_j)$$

$P(\textit{label} = l_i / \textit{tweet})$ means the posterior probability of the tweet's label (or word) when the tweet is given.

The label or word with maximum likelihood is calculated from the \hat{C} as mentioned above,

$$\textit{label}_{NB} = \operatorname{argmax}_{l_j \in L} P(\textit{label} = l_j | w_i),$$

The above equation can be summarized as;

$$c_{NB} = \operatorname{argmax}_{c \in C} P(c) \prod_{f \in F} P(f|c) \quad \text{Where } f \text{ is the feature set of } d.$$

As to increase the speed and the performance of the Naïve Bayes algorithm and due to calculations in language processing done in log space (Jurafsky DanMartin, 2000) , it can be generally expressed as;

$$c_{NB} = \operatorname{argmax}_{c \in C} \log P(c) + \sum_{i \in \text{positions}} \log P(w_i|c)$$

There are several variations of the Naïve Bayes classifier such as, Multinomial Naïve Bayes (MNB), Gaussian Naïve Bayes (GNB), Bernoulli Naïve Bayes (BNB). These variations are developed by different researchers depending on the requirements of the solutions.

5.4.5.1 Multinomial Naïve Bayes (MNB)

Go et al. (2009) has used MNB in his research on “Twitter sentiment classification using distant supervision”. As he has stated the MNB algorithm is as follows;

$$P_{NB}(c|d) := \frac{(P(c) \sum_{i=1}^m P(f|c)^{n_i(d)})}{P(d)}$$

f – Features

$n_i(d)$ – number of feature f found in the tweet d

m – Total number of features

5.4.6 Maximum Entropy Classifier (ME)

Maximum entropy is also known as multinomial logistic regression MaxEnt in short. This belongs to exponential classifiers where it extracts some set of weighted features linearly as stated by (Jurafsky DanMartin, 2000). MaxEnt can be used in classification when there are large amount of features to be classified. Maximum entropy classifier converts labels features sets to vectors using encoding and using these encoded vectors it is possible to calculate weights for each feature that can then be used to detect the most likely label for a feature set according to Medhat et al. (2014). Though Naïve Bayes classifier is a generative classifier, Logistic regression is a “**discriminative classifier**”.

Unlike Naïve Bayes algorithm, MaxEnt can't directly compute $P(c / d)$ since the weights of the features can get negative values. As a solution for this problem Jurafsky DanMartin (2000) has stated that by wrapping the **exp** function around the weight-feature dot product $\mathbf{w} \cdot \mathbf{f}$ will make all the values positive.

$$p(c|x) = \frac{1}{Z} \exp \sum_i w_i f_i$$

w_i – the tweet

f_i – the feature list

c – Information (can be unigram, bigram, trigram etc.)

x – The polarity class

This equation further can be redesign after taking the normalization factor Z and specifying the number of features as N;

$$p(c|x) = \frac{\exp\left(\sum_{i=1}^N w_i f_i\right)}{\sum_C \exp\left(\sum_{i=1}^N w_i f_i\right)}$$

Indicator functions where features that takes on only the values 0 and 1 are common in natural language processing. And features are not just a property of observation x , but a property in both polarity class c and observation x . Since that it is possible to use $f_i(c, x)$ rather than using f_i or $f_i(x)$ as explained by Jurafsky DanMartin (2000).

After the above adjustments, the maximum entropy classifier can be illustrated as below;

$$p(c|x) = \frac{\exp\left(\sum_{i=1}^N w_i f_i(c, x)\right)}{\sum_{c' \in C} \exp\left(\sum_{i=1}^N w_i f_i(c', x)\right)}$$

5.4.7 Linear Classifiers

Medhat et al. (2014) states in a given \vec{X} (where \vec{X} is normalized document word frequency), \vec{A} (vector of linear coefficients) and b is a scaler, the result of the linear classification predictor is p is defined as;

$$p = \vec{A} \cdot \vec{X} + b$$

P separates a hyperplane between different classes. There are many kinds of linear classifiers among them the following two classifiers are the mostly used;

- **Support Vector Machines Classifiers (SVM)**
- **Neural Network Classifiers**

5.4.8 Support Vector Machines Classifiers (SVM)

SVM mainly calculate the linear separator in a search space where it can best separate the polarity classes. And also SVM perform best in text classification due to the sparse nature of text as mentioned by Medhat et al. (2014). The key concept of SVM classification is to detect the right hyper plane which segregates the two classes (x and 0).

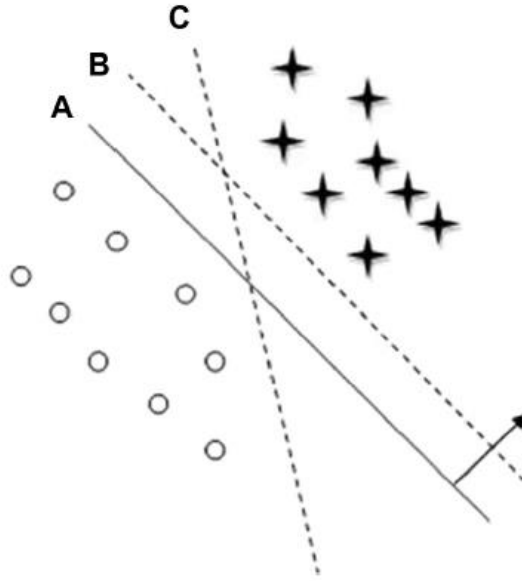


Figure 8 - SVM Illustration

Source: Medhat et al. (2014)

According to the above SVM classification diagram, there are two classes 'x' and '0' and three hyperplanes A, B and C. In SVM also there are several variations which are used depending on the problem. The basic idea in SVM training procedure is that to find the hyperplane represented by the vector w . Where w helps to separate the document vectors in a single class and making the representation as large as possible as mentioned in Pang et al. (2002).

$$\vec{w} := \sum_j \alpha_j c_j \vec{d}_j, \quad \alpha_j \geq 0,$$

c_j – positive and negative polarity

d_j – tweet

There are fine tuning methods like '**Kernel**' function to increase the accuracy level of the SVM algorithm.

5.4.9 Neural Network Classifiers

In the modern machine learning techniques, neural networks are widely used due to the high accuracy level of the classifier. In a neural network neurons are the basic unit and many neurons connect to each other and form a neural network. Though many researchers have used other supervised machine learning techniques, there are less number of researches are conducted using neural network classifiers. But there are certain amount of researches that have used recursive neural network classifiers and deep learning techniques where in deep learning polarity classification can be done in a more natural way.

According to Medhat et al. (2014):

“Multilayer neural networks are used for non-linear boundaries. These multiple layers are used to induce multiple piecewise linear boundaries, which are used to approximate enclosed regions belonging to a particular class. The training process of is more complex because the errors need to be back-propagated over different layers.”

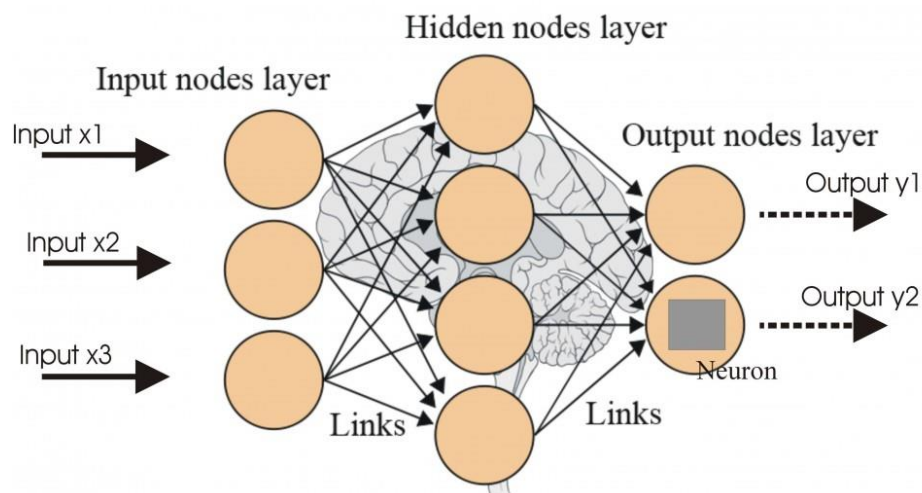


Figure 11 - Neural Network Architecture

Source: declare.com (2013)

5.4.10 Decision Tree Classifiers

Decision tree classifiers are used in sentimental classification under supervised machine learning. A condition on the attribute value is used to divide the training data according to hierarchical decomposition. According to Medhat et al. (2014) this division of data is happened continuously (recursively) till the leaf node of the decision tree contain minimum number of records which are used in the classification process. Medhat et al. (2014) further states that there are more predicators that is used to further breaking down of documents; depending on the similarity like wise.

In decision tree classification also there are variation developed by the researchers, according to Medhat et al. (2014) variation may differ depending on the concept of the tree; he further states that one researcher has proposed an approach to mine the content structures of the topical terms in sentence level context using **Maximum Spanning Tress (MST)**. Using this structure it is possible to discover the links among the term “t” and its context words. This approach is known as “**Topical Term Description Model**” for sentimental classification.

5.4.11 Rule based classifiers

This is also another supervised machine learning technique which is widely used in text classification. Due to the set of rules which the data space is modeled it is known as rule based classifier. In standards the right had side denotes the class label while the left hand side denotes the condition on the feature set expressed in disjunctive normal form Medhat et al. (2014). Mostly the rules depends on the term presence because of that terms are mostly included. The rules are mostly generated by the training phase depending on criteria.

As mentioned by Medhat et al. (2014) there are two main criteria in order to generate the rules; *support* and *confidence*. Support is the absolute number of instances in the training data set which are relevant to the rule. Confidence refers to the conditional probability of right hand of the rule is satisfied if left side of the equation is satisfied.

5.5 Similar Approaches

This section thoroughly discuss about the similar systems and similar approaches which have been used by the researchers. The following discussed approaches are mainly focused on how sentimental analysis and predictive analysis can be used in twitter data classification. Similar approaches have been categorized in to two parts;

- General sentimental analysis classification
- Political sentimental analysis classification

5.5.1 Political sentimental analysis classification

5.5.1.1 Predicting Elections with Twitter: What 140 Characters reveal about Political Sentiment (2010) - *Andranik Tumasjan, Timm O. Sprenger, Philipp G. Sandner, Isabell M. Welp*

This research was one of the earliest to evaluate the feasibility of using Twitter as a data source and perform sentimental analysis on the opinions and predict the winning probabilities of elections. This research is based on the **2009 German Federal election** which they have analyzed the tweets according to the different parties. According to Tumasjan et al. (2010) they have analyzed 104,003 tweets containing about the political parties or politicians. Their results shows that politicians extensively used Twitter as their primary social media to reach the public mass and due to high number of tweets regarding the election reflect the engagement of the users or the public.

Operational Analysis

Tumasjan et al. (2010) states that by the June 2009, 71% of all 1.8 million German Twitter users had visited Twitter only once and 15% of them have visited back 3 times. Where it shows high interaction between users towards the election in 2009. They have conducted the research based on the following questions;

- Does Twitter provide a platform for political deliberation online?
- How accurately can Twitter inform us about the electorate's political sentiment?
- Can Twitter serve as a predictor of the election result?

(Tumasjan et al., 2010)

The collection of tweets has taken over a period of a month prior to the German election with a large number of tweets have been collected nearer to the election. They have collected roughly 70,000 tweets each for a political party to obtain fair results. After the data is cleaned and preprocessed and then **LIWC2007 (Linguistic Inquiry and word count;** Pennebaker, Chung and Ireland 2007), text analysis software which was developed to assess structural, emotional and cognitive components of text samples has been used to extract the sentiments out from the tweets.

LIWC2007 determine the rate which certain cognitions and emotions like positive or negative emotions and future orientation are present in the text. Using those rate, the system calculates a relative frequency with words related to that dimension in the text sample. After the classification of tweets they have discovered 12 dimensions in order to classify political sentiment;

Future orientation, Past orientation, positive and negative emotions, sadness, anxiety, anger, tentativeness, certainty, work, achievement and money as stated by Tumasjan et al. (2010)

After selecting the leaders of the six main political parties, the researchers have illustrated the dimensions with the polarity of the sentiments.

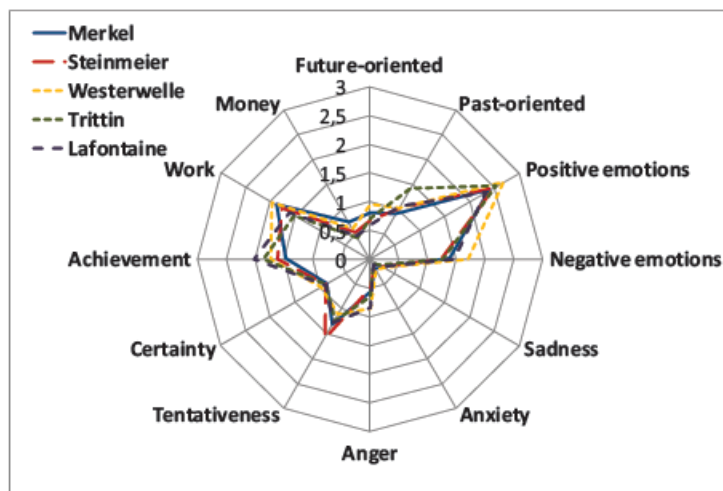


Figure 14 - Dimensions with the polarity of the Sentiments

Source: Tumasjan et al. (2010)

After all the tweets are classified, it's been ranked according to the tweet volume share for each political party. The below mentioned table illustrates the results. Tumasjan et al. (2010) has used MAE (mean absolute error) to measure the forest accuracy of this research. The MAE was 1.65% according to the calculations.

Party	All mentions		Election	
	Number of tweets	Share of Twitter traffic	Election result*	Prediction error
CDU	30,886	30.1%	29.0%	1.0%
CSU	5,748	5.6%	6.9%	1.3%
SPD	27,356	26.6%	24.5%	2.2%
FDP	17,737	17.3%	15.5%	1.7%
LINKE	12,689	12.4%	12.7%	0.3%
Grüne	8,250	8.0%	11.4%	3.3%
			MAE:	1.65%

* Adjusted to reflect only the 6 main parties in our sample

Figure 15 – Prediction of Errors

Limitations of the system

- Inability to clearly identify the twitter users only in the German electorate.
- Data was limited to the tweets containing only names of the political parties and politicians.
- Used a dictionary based classification software to identify the polarity of the tweets. The use of emoticons was not used since the LIWC dictionary based classifier doesn't provide those features.
- The convention of tweets from German language to English language made lots of noisy errors in the tweets.

5.5.1.2 The Predictive Power of Social Media: On the Predictability of U.S. Presidential Elections using Twitter (2014) - Kazem Jahanbakhsh, Yumi Moon

This paper examines the predictive power of Twitter using Twitter regarding the US presidential election of 2012. They have collected over 32 million tweets regarding only about the election over a four month of period and used machine learning technique to analyze.

Operational Analysis

Jahanbakhsh & Moon, (2014) states that by the time this project was initiated, there were only few researches that have been done to analyze and compare the tweets using machine learning techniques and deep comparison the results with traditional polls. They have performed content analysis using topic modelling and sentimental analysis on sample tweets and have compared sentimental analysis results with traditional polls' results within the same time span.

They have based their research on the following questions;

- Can one use Twitter data to predict the 2012 US presidential election?
- Are the content analysis results of Twitter comparable within traditional pollster results?
- Can one use topic modeling in order to discover topics of Twitter based discussions? Are those extracted topics in match with offline discussions?

(Jahanbakhsh & Moon, 2014)

Each tweet that was collected using the Official twitter API and Tweepy python library was comprised with *time frame of the tweet, source, latitude, longitude and text*. They compute the distributions of tweets according to the PST time zone and for key specific words. (E.g. *Obama*) Tweets was mainly filtered using keywords like Barack Obama, Mitt Romney, US and so on. Only 0.43% was geo-tagged out of the 39 million tweets collected. All the collected data was stored in a *MySQL* database.

Below illustration define the architectural design of the system.

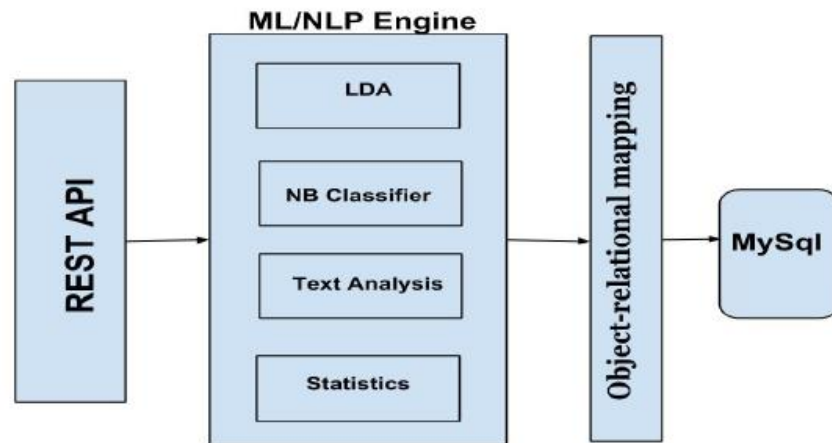


Figure 8 - ML/NLP Engine - Design Architecture

Source: Jahanbakhsh & Moon (2014)

Jahanbakhsh & Moon (2014) has stated that they developed an advanced machine learning and language processing engine to perform the text classification of the tweets. The ML-NLP engine constructed with four major components;

- **Statistical component** – Compute the basic statistics like mapping tweet posted time
- **Text analysis** – basic text analysis tasks like text cleaning and preprocessing
- **NB classifier** – Naïve Bayes supervised machine learning algorithm was used for sentimental analysis
- **LDA** – LDA algorithm was used for topic modelling.

After the tweets were cleaned and preprocessed the **Naïve Bayes algorithm** was used in classification purposes due to the higher accuracy of the classifier. General probabilistic framework the **LDA model** (Latent Dirichlet Allocation) was used in topic modeling since unsupervised learning technique was used there.

Conclusion

- Twitter can be used to predict the elections results of the US election.
- This research suggests that in order to find the accurate predictions; the polarity of the tweets should be concerned rather than based on the frequency of the names or the keywords being mentioned.
- Predictions results from political tweets can be matched with pollster results; especially during the election is getting closer.
- Geographical sentiment analysis predicted the final outcomes of 76% of the states in the US.
- Topic modeling can be used to discover the trending topics in the social media.

5.5.2 General sentimental analysis classification

5.5.2.1 Twitter Sentiment Classification using Distant Supervision (2009)

Alec Go, Richa Bhayani, Lei Huang

This research paper is based on an existing online sentimental analysis solution “**Sentiment 140**”. Go et al. (2009) states that this system is developed to give the sentiment to the consumers about any product or company with respect to the query term. The main contribution of this research is the idea of using tweets with emoticons for distant supervised learning.

Operational Analysis

Researchers have proposed a method to automatically extract the sentiment (positive or negative) from a tweet. They have cleaned the tweets well as it matters a lot in the final accuracy levels. In the tweet preprocessing stage they have extracted the noisy features like usernames, URLs, repeated letters and so on.

They have used multiple supervised machine learning techniques like,

- **Naïve Bayes**
- **Maximum Entropy (MaxEnt)**
- **Support Vector Machines (SVM)**

And different keywords to classify the polarities. According to (Go et al., 2009) they have used Multinomial Naïve Bayes model a variation of Naïve Bayes as the first classification technique. And also they have used the Stanford Classifier to perform the Maximum Entropy classification. SVM light software with a linear kernel has been used to perform the SVM classification technique. Authors have conducted the research on all three machine learning techniques to obtain the highest accurate classifier.

By using the official Twitter API they have collected the tweets for period of two months and collected 1,600,000 of training tweets. 177 negative tweets and 182 positive tweets were manually annotated for the training data set.

To identify the features in the tweets they have used Unigrams, bigrams, unigrams and bigrams and parts of speech. In the research the authors have mentioned that the POS tags were not useful than n-grams. According to the authors the highest accuracy which they have gained is 83% out of this system.

The authors have proposed some techniques to improve the accuracy of the classifiers;

Semantics – A polarity if a tweet can be depend on different perspectives but if the overall sentiment is classified for a tweet the accuracy might be reduced.

Domain specific tweets- If the domain or the scope of the tweets are limited, a higher accuracy can be gained using s good testing data.

Internationalization – In this research the authors have focused on English and Spanish language sentences. This is mainly due to the lack of training data in other languages.

Using emoticon data in test data – Authors propose that the polarity of emoticons don't have to influence the towards a polarity class if the emoticons are stripped out from the training data.

6. SOLUTION CONCEPT

6.1 Selection of Social Media

Facebook and Twitter are considered as the largest social media platforms in the world right now. As of by first quarter of 2016 monthly active users of Facebook is close to 1.65 Billion and among them close to 175 million users are from the United States of America. While Twitter has 310 million monthly active users and among them 65 million are from the US. According to statistics, there are roughly 240 million citizens who are above 18 years in the US.

When comparing the registered voters in US with the above mentioned social media monthly active users, Facebook represents closely about half the citizens while Twitter represents closely one third of the citizens. According to the statistics mentioned above, there is a high active usage of both Social Medias daily. But the structure of the two services are different. Twitter is widely considered as microblogging social media platform, one status message or a post cannot exceed over 140 characters. Due to this phenomenon users tend to post summarized opinions or emotions in this platform. These 140 characters posts are known as '*tweets*'. Tweets however structured with informal structure such as emoticons, slang language, links etc. But this 140 character limit phenomenon is widely helpful in data mining and opinion mining.

Most of the Twitter audience is consisted with celebrities, businessmen, diplomats, almost all the country heads like Presidents, Prime Ministers, politicians and general public. Due to this, twitter users can be considered as a good sample category in opinion mining.

Facebook in the other hand has enabled the users to publish content without any limit. Facebook users can post status messages, photos, videos and many more. Facebook also provide a feature called 'Pages', to promote businesses, events, personals. These pages are used mostly by artists, celebrities and also by politicians. Both Twitter and Facebook has provided public APIs to collect data from their websites. Facebook has provided a public API named as '**Social Graph API**' while Twitter has published several APIs like **REST API**, Media API and Collections API etc. The main problem in Facebook's social graph API is that, it only streams the public posts which have been posted. Public posts are mostly posted by the Facebook public pages and small number of users. Most of the general users have enabled the privacy setting to private due to security reasons. In many cases Facebook posts don't include attributes like the geo-location or the timestamp.

The Twitter API provides vital information regarding to opinion mining and considered as one of the best source to collect raw data. Most of the Twitter posts are public and the Twitter REST API enables to obtain attributes like geo-location, timestamp. As each of the tweet is consisted with 140 characters or less the size of each tweet is considered to be small and easy to store in the databases. Twitter REST API also enables to stream the tweets live where the researchers can obtain real-time tweets which is considered as a significant feature in data mining.

As this research is focused only about text opinion mining, photos and video contents posted by the users are not taken in to consideration. Though Facebook has larger audience than Twitter, by considering the high amount of public posts (opinions) that are able to be collected from Twitter, **Twitter** can be selected as the best social media platform to perform to collect raw data and perform opinion mining.

6.2 Selection of Sentiment Classification Technique

Sentimental Analysis is considered as the most prominent section in modern natural language processing. As mentioned by Medhat et al. (2014) and described in the Literature review chapter, sentimental analysis can be performed by using both Lexicon-based technique and Machine learning techniques. Though there are small number of research that have followed Hybrid Techniques (combination of machine learning and lexicon-based techniques), the accuracy levels are not clearly defined. Due to above mentioned reasons, this research is focused on the machine learning approach in sentimental classification which can be used in Twitter opinion mining. The machine learning approach has two categories; supervised machine learning classification and unsupervised machine leaning classification.

Unsupervised machine learning classification is mainly used when it is unable to find annotated or labeled training data or when there is less amount of annotated training data. One of the earliest unsupervised learning sentimental classification was performed by Peter D. Turney in 2001. According to Turney (2002), he has used terms *excellent* and *poor* to define positivity and negativity of a sentence. His unsupervised approach proposed to use **semantic orientation** on each phrase after extracting phrases which contains adjectives or adverbs. But he suggests that though it produced accuracy levels from 66% to 84% depending on the topic, semantic orientation can

further be used in machine learning classification techniques. The accuracy of sentiment classification highly relies on the training data set, if the number of topics to be used in the sentiment classification is higher the training data should also have wide spreaded sentiments and a large amount. Supervised machine learning approaches have better performance in this project since large sets of annotated training data can be gathered and this project only focus about a single topic; Politics. And also to train an unsupervised classification algorithm, it takes tremendous amount of time as the algorithm itself should classify the training data set. Due to the time constraint of this project and due to the availability of higher amount of annotated training data set, it is appropriate to use **machine learning technique** as the classification technique.

As mentioned in the Literature review, there are three main machine learning classification techniques; Naïve Bayes, Maximum Entropy and Support Vector Machines. According to Medhat et al. (2014) most frequent used supervised machine learning algorithms are Naïve Bayes and Support vector machines. The accuracies of the algorithms also depend on the feature extraction methods used in the research such as Part-of-Speech, negation, n-grams etc. Since that in many research, variations of the above mentioned classifiers are used to obtain higher accuracies.

The following table illustrates how the supervised classification techniques perform with each feature extraction method according to Go et al. (2009) research based on Twitter sentiment classification.

Feature	Keyword	Naïve Bayes	SVM	MaxEnt
Unigram	65.2	81.3%	80.5%	82.2%
Bigram	N/A	81.6%	79.1%	78.8%
Unigram + Bigram	N/A	82.7%	83.0%	81.6%
Unigram + POS	N/A	79.9%	79.9%	81.9%

Table 2 - Classifier Accuracy

Source : (Go et al., 2009)

According to Tripathy et al. (2015) '*precision*', '*recall*' and '*F-measure*' performance evaluation parameters are used to calculate the accuracy. Precision provides the exactness of the classifier where it is the ratio of number of correctly predicted positive tweets to the total number of tweets

predicted as positive. Recall means the completeness of the classifier. The ratio of number of correctly predicted positive tweets to the actual number of positive tweets in the corpus. F-measure means the mean of precision and recall. Tripathy et al. (2015) performance results are mentioned below.

	Precision	Recall	F-measure
Negative	0.80	0.89	0.84
Positive	0.87	0.77	0.82

Table 3 - Evaluation parameters for Naïve Bayes Classifier

Source: Tripathy et al. (2015)

Maximum accuracy achieved by cross validation analysis of **Naïve Bayes classifier is 0.8953**

	Precision	Recall	F-measure
Negative	0.87	0.89	0.88
Positive	0.89	0.86	0.88

Table 4 - Evaluation parameters for Support Vector Machine classifier

Source: Tripathy et al. (2015)

Maximum accuracy achieved by cross validation analysis of **Support vector machine classifier is 0.9406**

According to above results SVM classifier is slightly more accurate than the Naïve Bayes classifier. But according to many research, Naïve Bayes classifier is considered as the most widely used classifier due to the less execution time, less CPU memory is acquired when classification and less amount of training data is needed comparing to SVM. Since above accuracies are mostly tested on review or other corpus, in Twitter sentimental classification these accuracy might be changed due the character limitation. Considering above mentioned reasons, **Naïve Bayes classifier** is selected as the sentimental classification technique.

6.3 Data cleaning

Accuracy of any classifier is primarily depends on how well the data or text is cleaned. In the cleaning process of tweets, items such as usernames, stop words, URLs, HTML tags, emoticons etc. are removed before the feature extraction process. Data cleaning process also helps to reduce the unnecessary words or letters in the dataset which will be benefited to the sentiment classifier. Go et al. (2009) has mentioned how data cleaning is important to reduce unimportant features in a data set. According to him features were reduced down to 45.85% after removing the above mentioned items in a twitter dataset.

Due to high availability of slang words and misspellings, those words should also be avoided but as slang words might also contain certain polarity towards and emotion, it will be more accurate to maintain a slang word dictionary as mentioned by Neethu & Rajasree (2013).

The below mentioned features will be cleaned in a tweet to obtain a higher accuracy in this research.

Feature	Original Tweet	Cleaned tweet
Username	@HilaryClinton	<USER>
URLs	https://www.election.gov.us	<URL>
Hash Tags	#Election	Election
Stop words	From, since, am, a, is ,are, at etc.	Such letter or words will be removed
Repeated letters	Baaaad, neeeeds	Bad, needs
Special features	RT	-

Table 5 - Feature to be reduced

Parikh & Movassate (2009) clearly state the accuracy change in Multinomial Naïve Bayes classifier when the tweets are cleaned or pre-processed and not pre-processed.

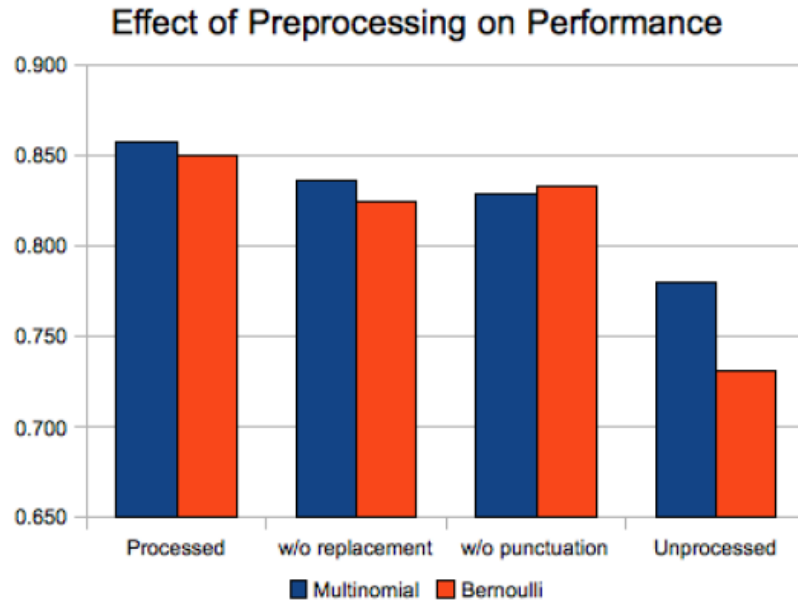


Figure 9 - Effect of Pre-processing on Performance

Source: (Parikh & Movassate, 2009)

Accuracy of 85.7% is achieved using pre-processed tweets classified using Multinomial Naïve Bayes with Unigrams while unprocessed tweets using Multinomial Naïve Bayes with unigrams provided 78% of accuracy.

6.4 Feature Extraction

Selection of feature extraction methods are depend on the sentiment classifier as it affects the accuracy of the final result. Feature extraction also depends on the type of media the sentimental classification performed on such as reviews, Tweets and other text corpus.

Research	Type of media	Number of features	Term Frequency / Presence	Features Extracted	Accuracy with NB
(Pang et al., 2002)	Movie Review	16165	Frequency	Unigrams	78.7%
		16165	Presence	Unigrams	81.0%
		32330	Presence	Unigrams+bigrams	80.6%
		16165	Presence	Bigrams	77.3%
		16695	Presence	Unigrams+POS	81.5%
		2633	Presence	Adjectives	77.0%
		2633	Presence	Top 2633 unigrams	80.3%
		22430	Presence	Unigrams+position	81.0%
(Go et al., 2009)	Twitter.com	Not Mentioned	Frequency	Unigram	81.3%
			Presence	Bigram	81.6%
			Presence	Unigram+Bigram	82.7%
(Pak & Paroubek 2010)	Twitter.com	Not mentioned	Frequency	Unigram	71%
				Bigram	82%
				Trigram	80%

Table 6 Feature Extraction Summary

Pang et al. (2002) claims that depending on the type of research medium and nature of the training data the usage of Unigrams, Bigrams, POS or Unigrams+Bigrams can be differ. Both Pang et al. (2002) and Go et al. (2009) suggests that in feature extraction process, term presence always perform better than term frequency. From (Pak & Paroubek 2010), it can be justified that usage of higher n-gram doesn't provide higher accuracy. But according to the above mentioned data n-

grams usually provide much accuracy with Naïve Bayes classifier than POS tagging. Since **unigram+bigram** provides higher accuracy, it will be used in the implementation.

6.5 Gathering Training data

Proper training data should be collected in order to obtain high accuracy rate. But gathering training data is considered to be a time consuming process. As a solution it is discovered that previously gathered rich data sets can be used as training data in the system. By using a rich data set to train the system, the sentiments of the tweets can be easily determine. According to Go et al. (2009), they have collected training data over a certain period and have made the dataset public. Likewise Jahanbakhsh & Moon (2014) has conducted their research on predicting the U.S. election 2012 and collected annotated training data set over 1 million.

It is possible to use those above mentioned training data as its public. By using equal amount of positive and negative annotated training data the classifier can be trained to identify the accurate polarity of the tweets. Neither of the researches mentioned above provide the neutral dataset to the public.

After a thorough research conducted on gathering the dataset, it was decided to use the 1.6 million positive and negative data sets in English provided by the A. Go. They have annotated this dataset by mapping the happy and sad emoticons with the tweets. For this process they had to consume close to 3 months of a time period. Since the neutral data set wasn't provided by any of the researchers, it was assumed to collect tweets from renowned news sources like '**CNN**', '**BBC**' to be neutral data.

Different amounts of tweets will be used in training the classifier to achieve a stable accuracy in the system.

6.6 Selection of the Programming Language

Selection of the right programming language for this project is a crucial task as the optimization of the system relies on the selected programming language. Since this system uses machine learning techniques and mathematical operations, it is required to use a programming language which supports rich set of mathematical and statistical libraries. There are 3 main languages that are used in data mining or data science projects. ‘**R**’ is a statistical programming language which is heavily used in data mining projects. ‘**Python**’ is a scripting language which supports number of external libraries which are used in natural language processing and in machine learning. ‘**Java**’ is another high level programming language which is used in data science. Though Java isn’t built for process data mining calculations, it can be optimized by using frameworks such as Apache Hadoop. The chart mentioned below displays the most popular tools that are used in Data Science in 2016.

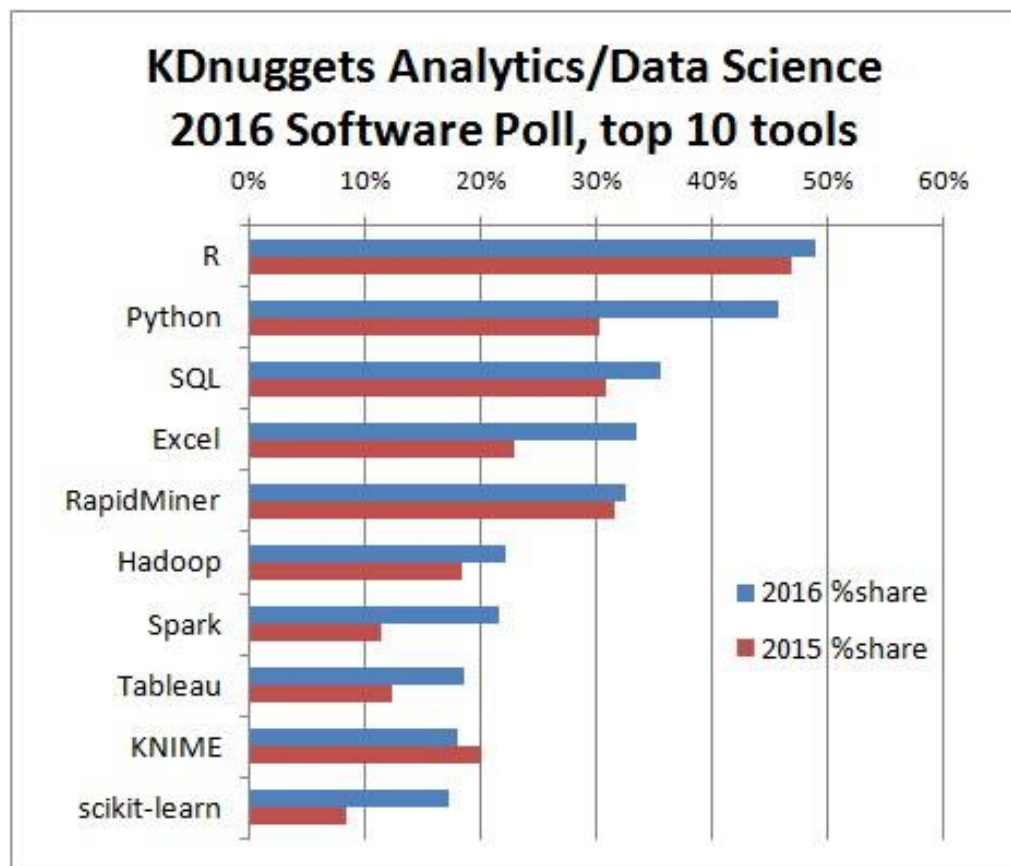


Figure 17 - Popular software in Data Science

(Source: kdnuggets.com, 2016)

According to the above poll, R is the most widely used programming language in both 2015 and 2016 in Data Science. But Python has increased its popularity heavily in 2016 comparing to 2015. Java programming language needs lots of coding to be done than both R and python. When considering machine learning and natural language processing tools, many tools such as ‘NLTK’, ‘Scikit-Learn’, ‘Pandas’ are supported for Python and R. Due to this phenomenon it is flexible to choose a language out of Python and R.

Since this proposed project requires a language that supports machine learning, NLP tools as well as to perform visualization of data, object oriented concepts python was selected as the programming language.

6.7 Selection of the Natural Language Processing Tool

Before applying machine learning techniques, the raw data should be processed using natural language processing techniques. NLP is a field in computational linguistics which connects human language which is known as natural language with computers. Since this project uses tweets as the source of input, using natural language processing those inputs should be processed to enable computers to understand.

According to opensource.com (2015), there are few NLP tools that can be used;

Benchmarks	Stanford Core NLP Suite	Natural Language Toolkit	Apache OpenNLP
Programming languages support	Java	Python	Java
Supports machine learning algorithms	Yes	Yes	Yes
Tokenization	Yes	Yes	No
Open Source	Yes	Yes	Yes

Table 7- Comparison NLP Tools

Re-writing of the same algorithms during a short period of time won't increase the efficiency of the system. Due to that reason the use of NLP tool kit would extensively help to increase the accuracy of the system. The 3 NLP tools which mentioned above are open source, due to this the author is able to understand the backend processes of the tool. Natural Language Toolkit is considered to be much more efficient than the other tools mentioned above. Also, NLTK provides efficient implementations of machine learning techniques. NLTK also provides rich content of documentation which is crucial in the implementation. Since Python has been chosen as the primary programming language, Natural Language Toolkit is more effective in the usage.

Earlier in this chapter, the '*Naïve Bayes Algorithm*' was selected as the classification algorithm in this proposed project. The implementation of the Naïve Bayes algorithm in the Natural Language Toolkit would be used in this project. Using the methods such as '*Most informative features*', it is possible to keep track of the accuracy of the classifier. Due to above mentioned reason, Natural Language Toolkit would be used as the NLP tool in this project.

6.8 Selection of the Database

In data mining projects, the way of storing the data plays an important task. In general software projects, SQL Databases are commonly used as the primary database. Primarily there are two types of databases, *Relational Databases* and *Non-Relational Databases*. In Relational databases the data is stored in tables as records with relationships with rest of the tables in the same database. RDB mainly uses SQL also known as Structured Query Language for the functionalities. '*Relationships*' between the data or the tables is the key feature in RDB. Non-relational databases are also known as '*No-SQL*' databases. These type of databases are commonly used in data mining or Big Data projects as the data which are collecting have no relationships with the rest of the data.

Due to this phenomenon large amount of data can be stored as objects in No-SQL databases. Databases such as 'MongoDB', 'Couchbase' can be considered as Document-Oriented No-SQL databases.

There are several disadvantages in using RDB for data mining projects.

- High Latency in data saving and retrieving
- Difficulty in scaling the database
- Inability to store Big Data

In this project the main task of the database will be to store the tweets which are streamed using the Twitter API each day. Since the Twitter API streams humongous number of tweets in real time, the database should include the capacity to save data without in latency. Twitter API streams the tweets as JSON objects, which generally include 26-32 fields in a single object. Tweets are inconsistent due to that, the number of fields may vary. The key feature of tweets is that they don't require any relationship when storing data.

Due to above reasons, use of No-SQL database rather than using a traditional SQL database is crucial in this proposed system. According to the research of Singh & Sachdeva (2014), MongoDB has been recommended as an efficient No-SQL database which can be used in data mining. Though there are no much of a differences between Couchbase and MongoDB, MongoDB is ranked 1st in document based No-SQL databases while Couchbase is ranked 6th according to (db-engines.com, 2016). MongoDB database supports the python development environment.

Considering the above facts, it was decided to use MongoDB database as the primary database of the proposed system.

7. DESIGN

This chapter is consisted of the detailed designs and the architectural diagrams which were used to design the overall system. These designs were iteratively changed during the system modelling phases. Below mentioned diagrams are responsible for the final implementation of the system.

7.1 Activity Diagrams

Activity diagrams are used to illustrate the primary action flow of the system. This also can be used to illustrate the action flow of each individual components. The overall activity diagram and individual activity diagrams of several individual components are depicted below.

7.1.1 Overall Activity Diagram

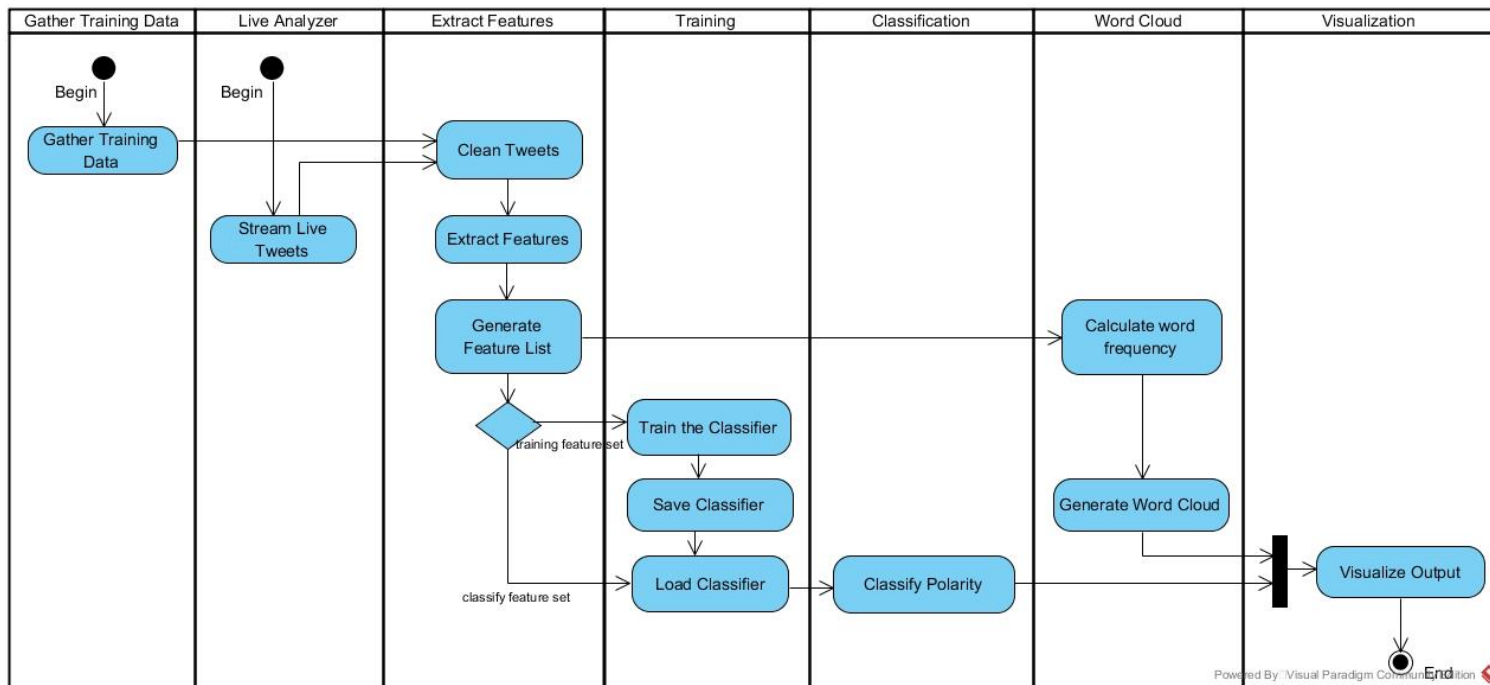


Figure 18 - Overall Activity Diagram

7.1.2 Extract Features Activity Diagram

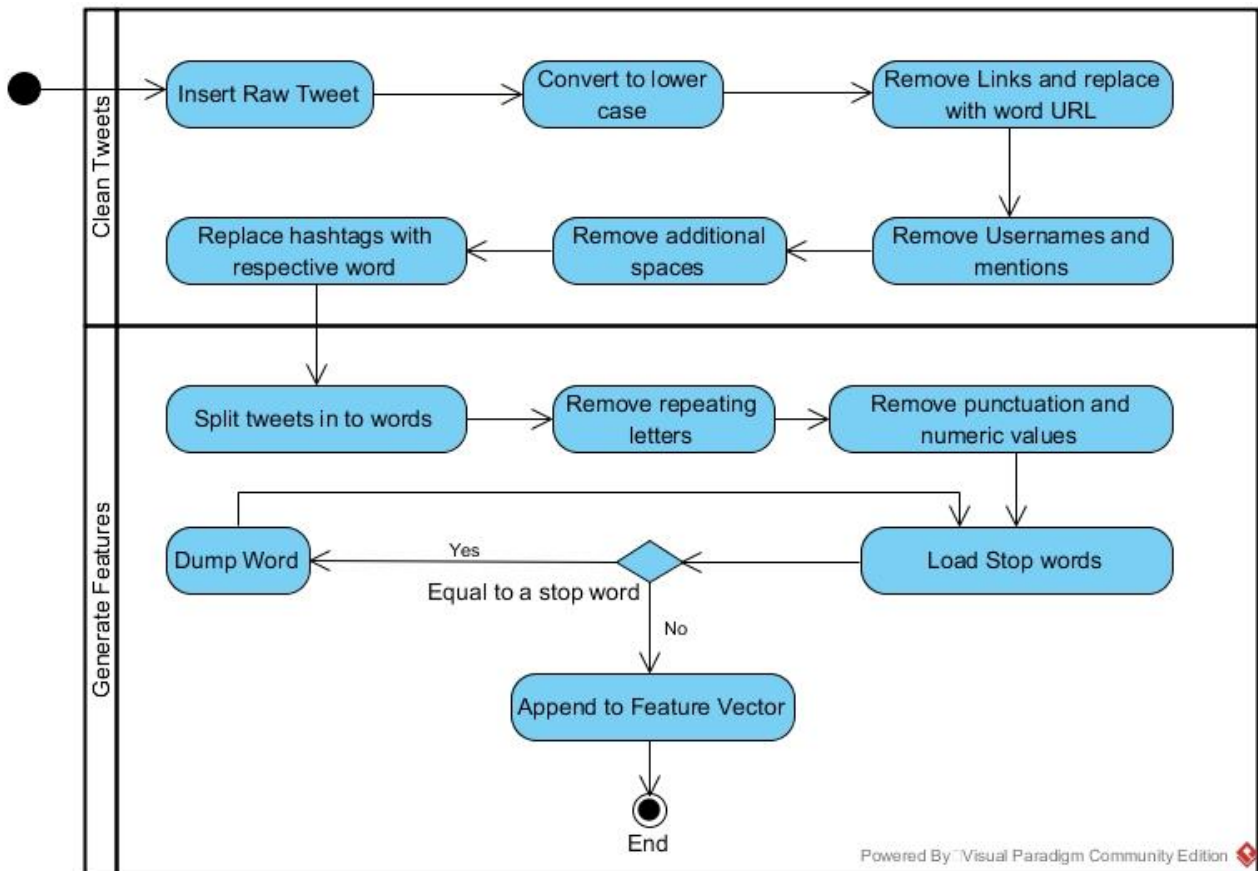


Figure 19 - Extract Features Activity Diagram

In this activity diagram, two classes are get involved. Due to the clarification of the diagram, the processes of the classes are separately illustrated. Initially the basic cleaning processed will be executed, later tweets will be split into individual words and remove punctuations and numerical values. Finally those words will be compared with stop words and only meaningful words will be append to the feature vector.

7.1.3 Train Classifier Activity Diagram

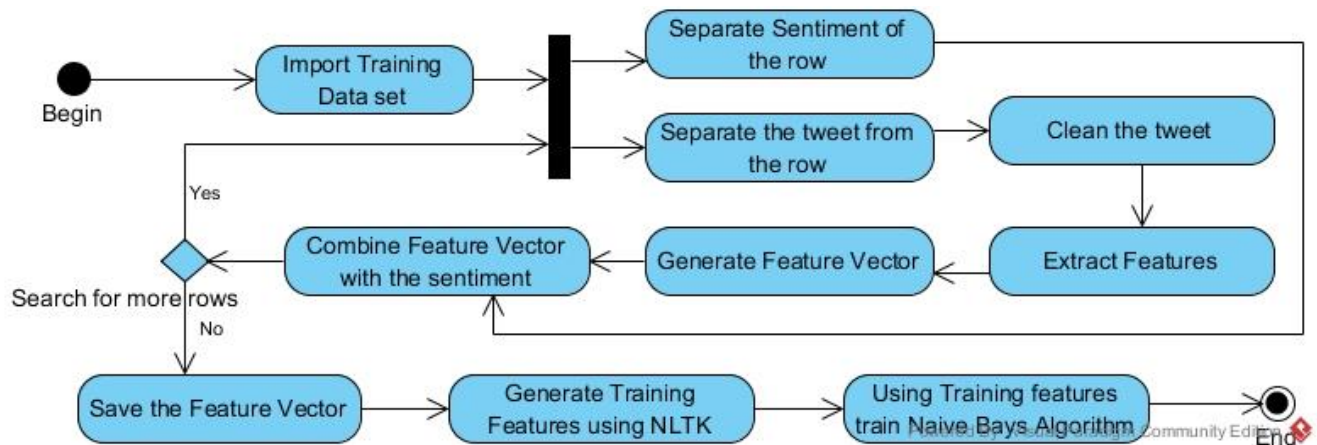


Figure 20 - Train Classifier Activity Diagram

7.1.4 Generate Word Cloud Activity Diagram

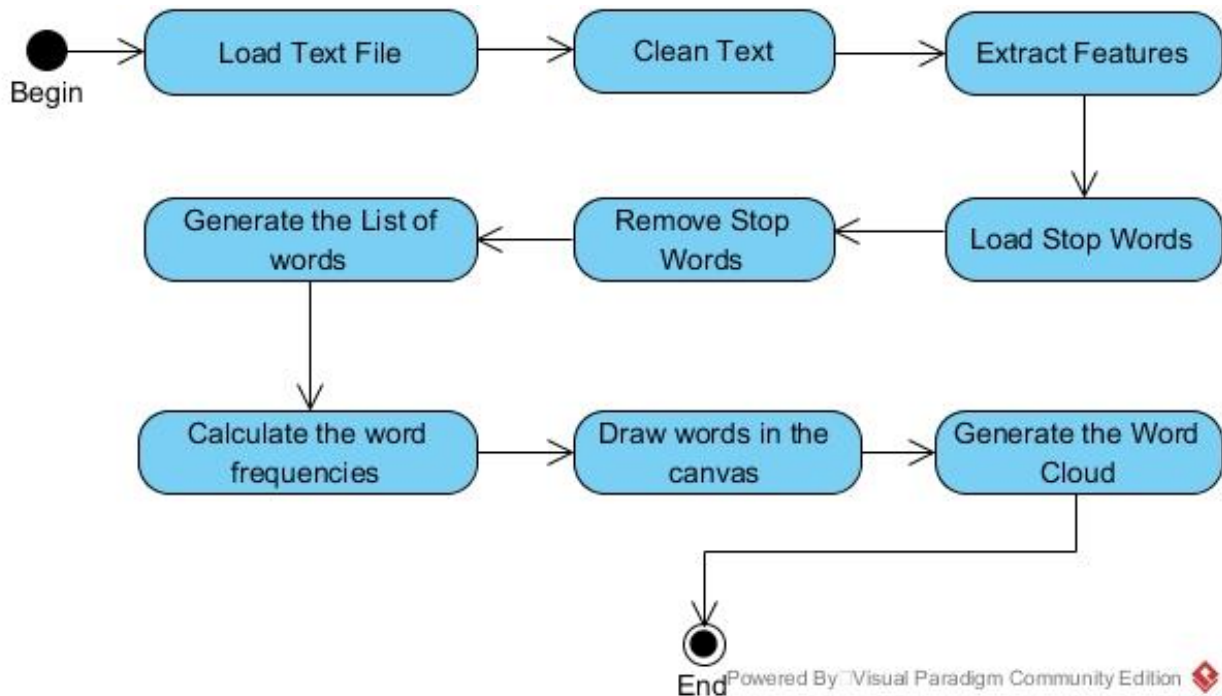


Figure 21 - Generate Word Cloud Activity Diagram

7.1.5 Live Sentiment Analysis and Visualization activity diagram

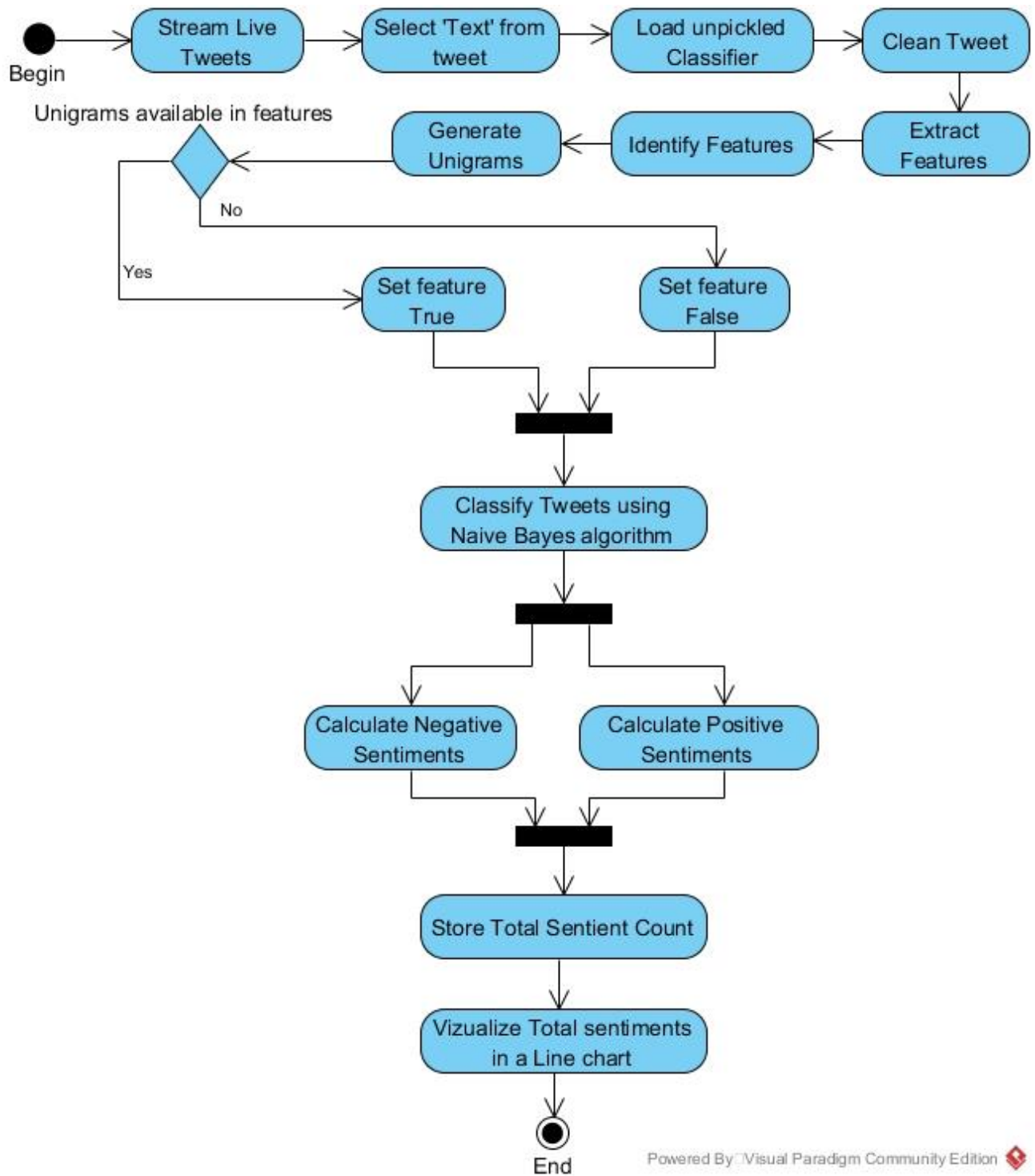


Figure 22 - Live Sentiment Analysis Activity Diagram

7.2 Overall Class Diagram

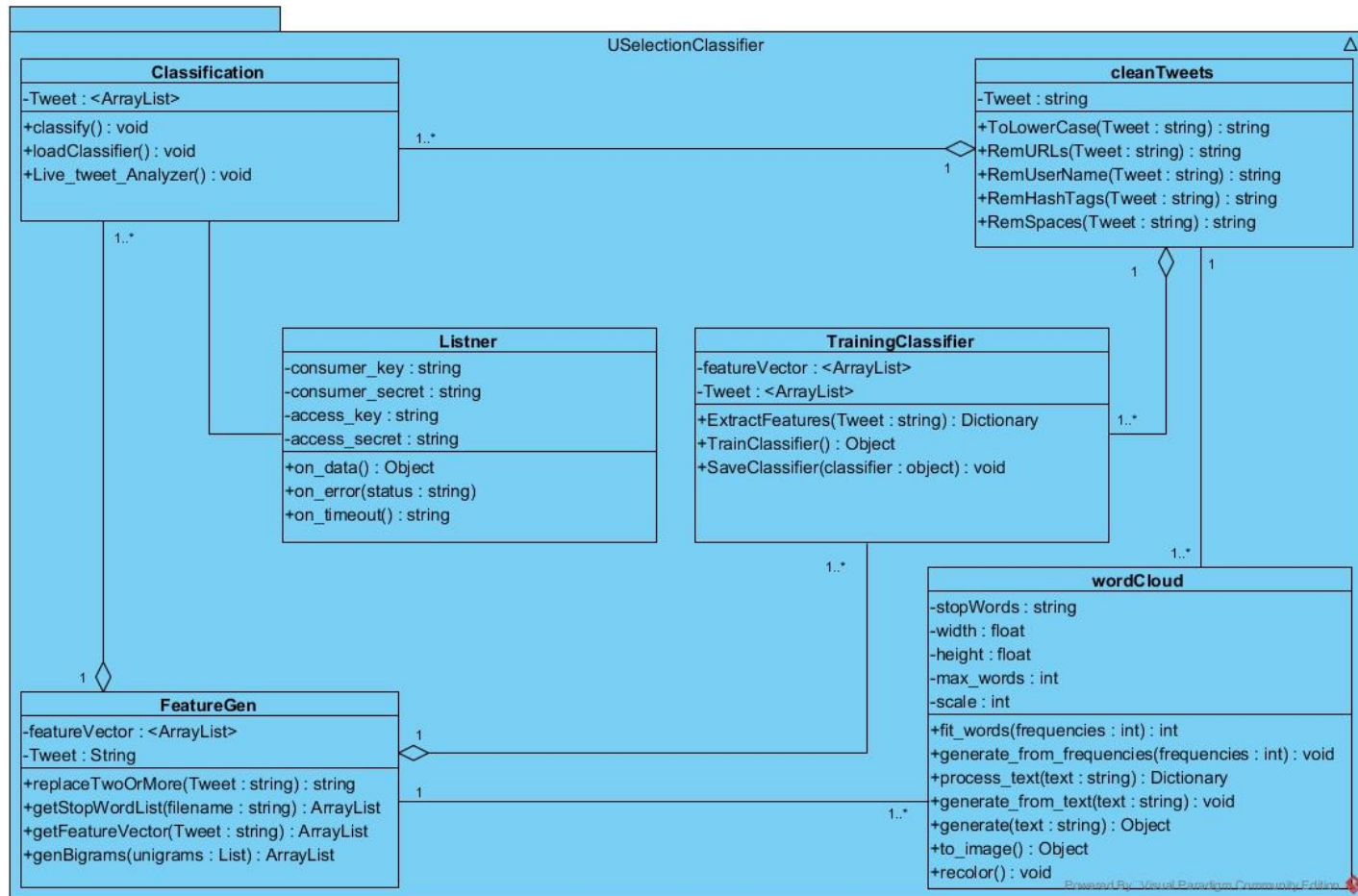


Figure 23 - Overall Class Diagram

The proposed system is consisted of the above mentioned classes. Since python which is the programming language proposed to implement the system, supports object oriented concepts, the above overall class diagram is designed according to such concepts. The Naïve Bayes classifier can be trained using the **TrainingClassifier** class. **CleanTweets** class is capable of cleaning the tweet while **FeatureGen** class is capable of generating the required features whenever necessary.

Live tweets can be classified using the **Listner** class along with the **Classification** class. **WordCloud** class has the ability to generate the word clouds, where the underlying topics of the tweets can be discovered easily.

7.2 Sequence Diagrams

The communication of functions or data within the classes mentioned in the class diagram is illustrated using sequence diagrams. Sequence diagrams include the respective functions that are required to communicate with each class.

7.2.1 Feature Extraction

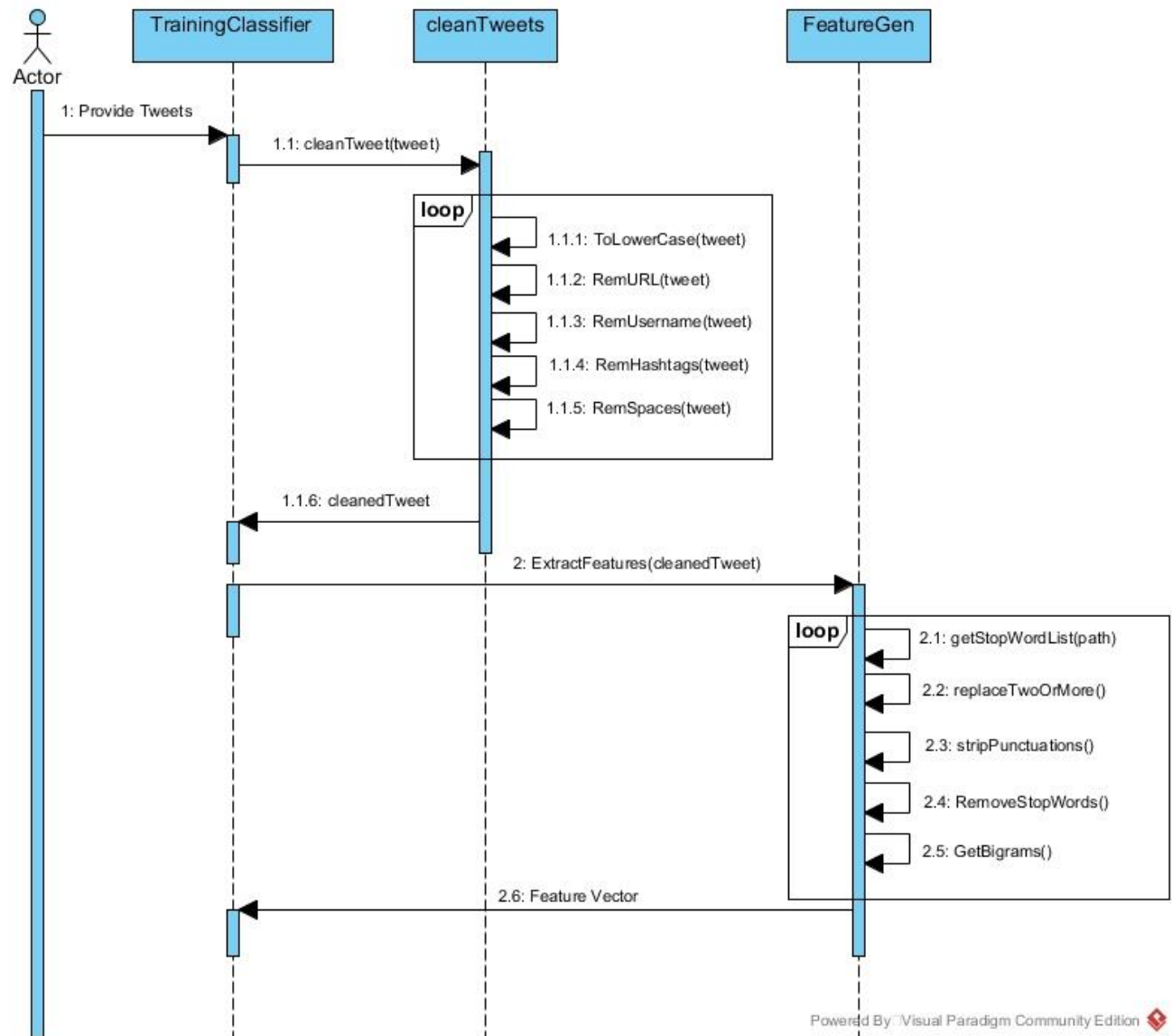


Figure 25 - Extract Features Sequence Diagram

7.2.2 Train Classifier

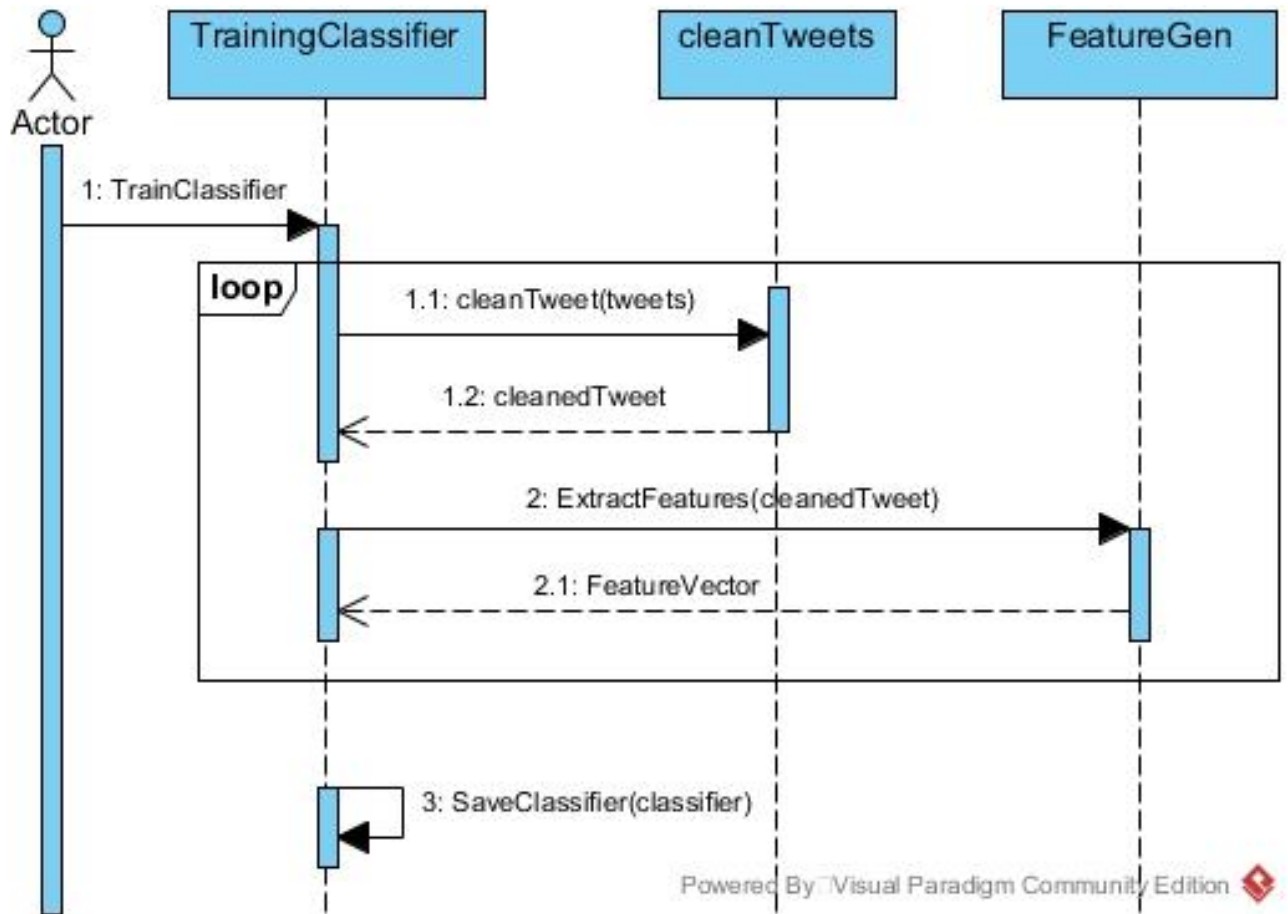


Figure 26 - Train Classifier Sequence Diagram

7.2.3 Generate Word Cloud

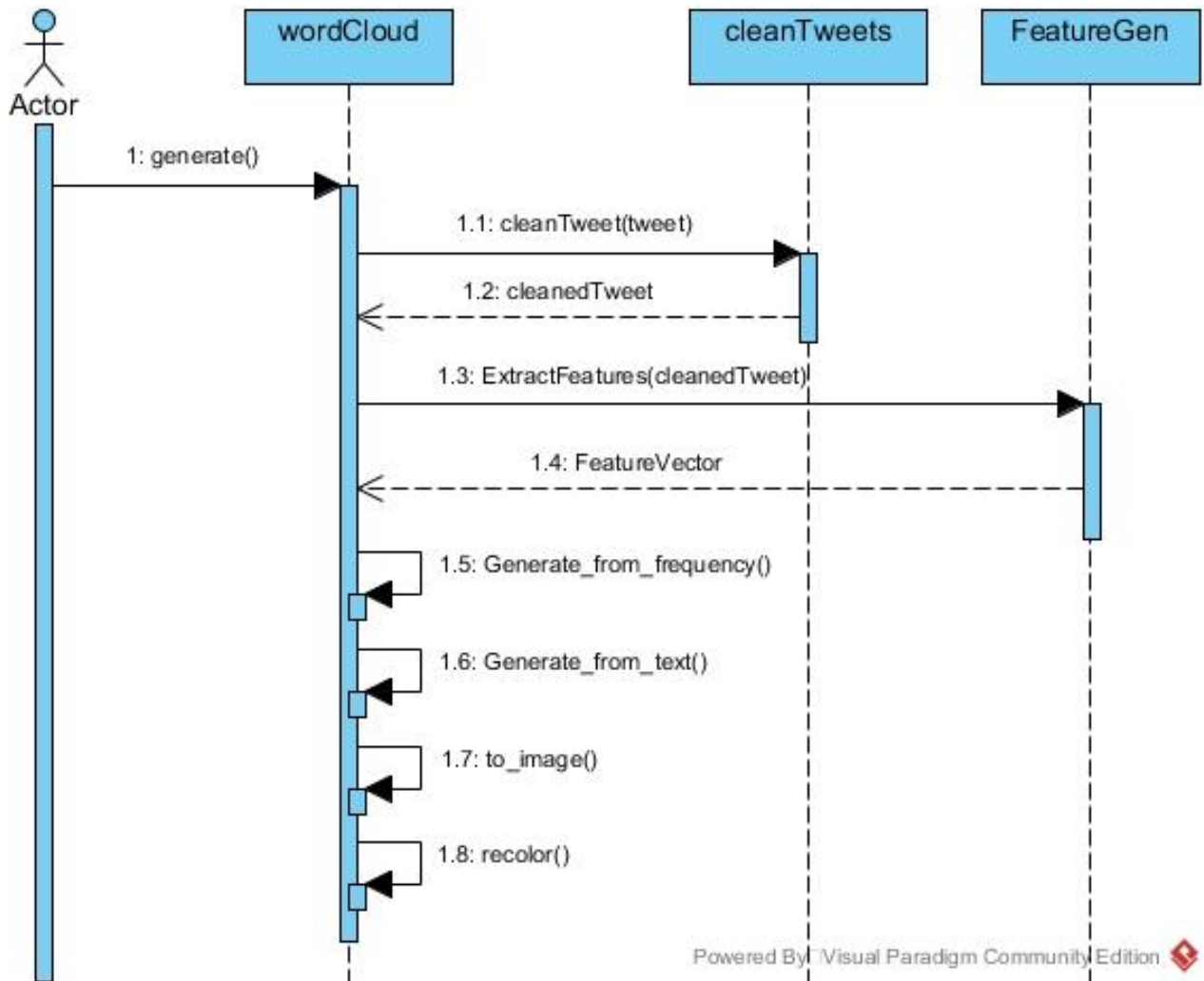


Figure 27 - Generate Word Cloud Sequence Diagram

7.2.4 Live Sentiment Analysis and Visualization

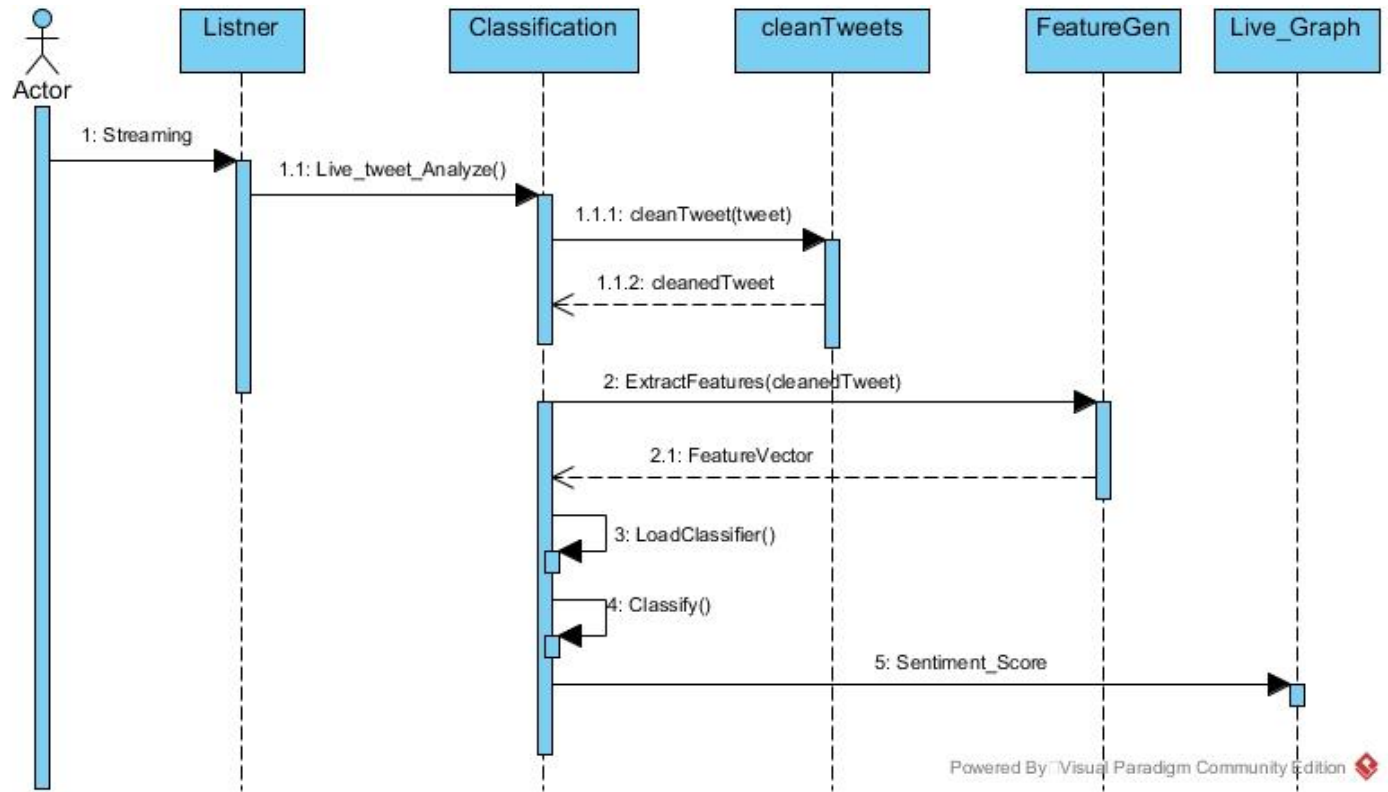


Figure 28 - Live Sentiment Analysis - Sequence Diagram

8. IMPLEMENTATION

This chapter thoroughly elaborates the implementation phase of the system. The implementation was initially executed using the research and approaches that were filtered out from the *Solution Concept* chapter. Additionally more research and experimental implementations were carried out to choose the most sophisticated solution whenever necessary. The implementation reflects the *Design Models* and *Object Oriented Software Architecture* which were designed in the *Detailed Design* chapter.

Before the initialization of the implementation, the relevant software and tools that were discussed in the *Solution Concept* was installed. Since Python 2.7 was chosen as the programming language to implement the solution, JetBrains PyCharm 5.0 used as the text editor. Furthermore NLTK, MongoDB, Robomongo and Tweepy was installed to build the system.

This chapter will be ordered according to the development of the components of the system.

1. Implementation of the NoSQL Database
2. Implementation of Twitter streamer to stream daily tweets
3. Pre-processing of tweets
4. Classification Algorithm
5. Implementation of Live tweet analyser module
6. Implementation of Word cloud map to discover underlying topics

The implementation of the above mentioned components were executed iteratively in order to gain the maximum accuracy for the system. The development of the above mentioned components were divided in to three main categories and then executed. Component number one and two were the initial components that were developed since this system handles large amount of data with regarding to data mining and storing of the particular data. Afterwards, component number three and four were developed with several number of iterations in development. Component number five and six were developed to test the performance of the category number one and two by visualizing the processed data.

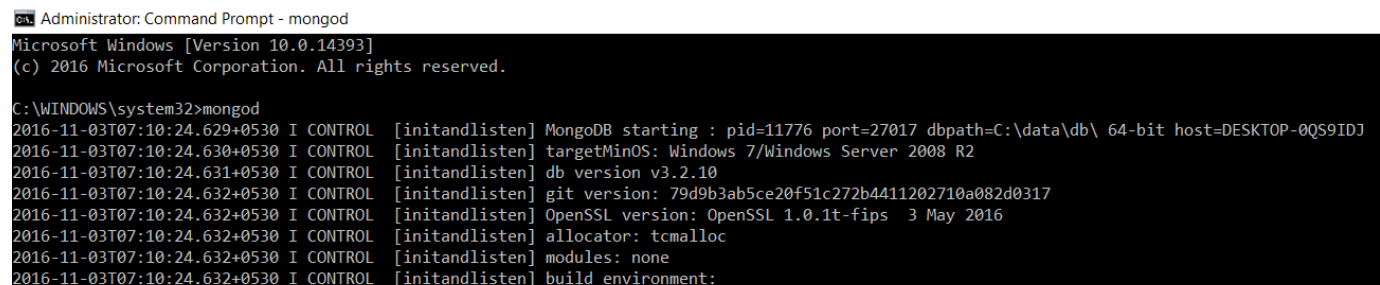
Afterwards, the below mentioned components were developed to visualize the analysis processed by the system,

7. Integration of the Graphical User interface to the system
8. Development of the real-time visualizer to analyse the trend

8.1 Implementation of the NoSQL Database

Since this system requires historical data to be analysed, a database should be integrated to the system. In the initial development stages comma separated value (CSV) files were used to store the real time tweets that were streamed through the official Twitter streaming API. The speed of the Twitter streaming API is very high, due to that phenomenon the JSON formatted tweets streamed through the API tend to get corrupted. Additionally CSV writer was unable to write the real time streaming tweets to the CSV. To avoid the above mentioned circumstances, a database had to be implemented. As the tweets are streamed in JSON format and there are no relations in the tweets a NoSQL database was implemented. In NoSQL databases data are saved as objects. This feature is significantly important for this project as the JSON formatted tweets can be stored as JSON objects in the NoSQL database. Unlike relational databases, NoSQL databases are considered to be efficient and lightweight in data mining.

From a thorough research as mentioned in the *Solution Concept*, MongoDB was selected as the NoSQL database. After the installation of the MongoDB through the command line, the following command needs to be executed,



```
Administrator: Command Prompt - mongod
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>mongod
2016-11-03T07:10:24.629+0530 I CONTROL [initandlisten] MongoDB starting : pid=11776 port=27017 dbpath=C:\data\db\ 64-bit host=DESKTOP-0QS9IDJ
2016-11-03T07:10:24.630+0530 I CONTROL [initandlisten] targetMinOS: Windows 7/Windows Server 2008 R2
2016-11-03T07:10:24.631+0530 I CONTROL [initandlisten] db version v3.2.10
2016-11-03T07:10:24.632+0530 I CONTROL [initandlisten] git version: 79d9b3ab5ce20f51c272b4411202710a082d0317
2016-11-03T07:10:24.632+0530 I CONTROL [initandlisten] OpenSSL version: OpenSSL 1.0.1t-fips 3 May 2016
2016-11-03T07:10:24.632+0530 I CONTROL [initandlisten] allocator: tcmalloc
2016-11-03T07:10:24.632+0530 I CONTROL [initandlisten] modules: none
2016-11-03T07:10:24.632+0530 I CONTROL [initandlisten] build environment:
```

Figure 29 - Initializing MongoDB

Storing and retrieving of data from the MongoDB will be discussed in the following section.

```
db.getCollection('Hillary_10_13').find({})
```

Using the above query the data can be viewed in the command line.

8.2 Implementation of the Twitter streamer

In order to access the twitter streamer, the user should obtain secret access credentials from the official Twitter website, www.twitter.com. This access tokens provide the authentication to access the live twitter stream. Though twitter provides real time tweets for the developers, it is considered to be only 5% from the real time generating tweets as mentioned in the dev.twitter.com (2016). The following mentioned access tokens were obtained by the author in order to get the access to the Official twitter streaming API.

```
consumer_key = " "  
consumer_secret = " "  
access_key = " "  
access_secret = " "
```

Consumer key, consumer secret key, access key and access secret key aren't mentioned in the documentation due to Privacy policy of Twitter.

```
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)  
auth.set_access_token(access_key, access_secret)  
api = tweepy.API(auth)
```

Using the above mentioned code, the secret keys are passed to the Twitter website to get authenticated. Using 'auth' as the parameter for the Tweepy API, it is possible to connect with the REST API of the Twitter.

The following coded implementation was used to connect with the Tweepy API. Since tweepy API supports Python after the authentication process the streaming class is able to connect with the Tweepy StreamListener to stream the real time tweets.

```
def __init__(self, api):
    self.api = api
    super(tweepy.StreamListener, self).__init__()
    self.db = pymongo.MongoClient().USelection
```

The above mentioned implementation is the constructor of the Streaming class. As MongoDB is used to store the live streaming tweets, the initialization of the database is implemented inside the constructor of the Streaming class. **PyMongo** is a Python distribution containing tools for working with MongoDB. PyMongo enables to connect Python with MongoDB database efficiently. Using the MongoClient() method in PyMongo, the database which is created in the MongoDB is connected with the streaming class. “**USelection**” is the database which is created in MongoDB.

```
def on_data(self, tweet):

    New_tweet = json.loads(tweet)
    text = New_tweet ["text"]
    created_at = New_tweet ["created_at"]
    print(tweet)
    self.db.Trump_11_05.insert(json.loads(tweet))
```

The *on_data* method streams the real time tweet to the streamer class from the Tweepy API. Since tweets are streamed in JSON format it can be easily stored to the database. Developers can print the incoming live tweets to the console in order to check the subjectivity of the tweet to the keyword. Using MongoDB functions, a collection (a table in Relational data model) is created to load the incoming tweets.

```
sapi = tweepy.streaming.Stream(auth, CustomStreamListener(api))
sapi.filter(track=['trump'], languages=['en'])
```

Using the filter method, it is possible to track the necessary data. ‘Track’ is used to only stream the given word or words. This helps to stream tweets which are related to either ‘Hillary Clinton’ or ‘Donald Trump’. Additionally the language of the streaming tweets are defined to filter out the English tweets since this system is only capable of analysing English language.

```
{
  "_id" : ObjectId("57ff792b9885a4101051ed3f"),
  "contributors" : null,
  "truncated" : false,
  "text" : "RT @scrowder: Hillary Clinton. A role model for all young women! You too can make it if you husband's rape, sell your soul and ki...",
  "is_quote_status" : false,
  "in_reply_to_status_id" : null,
  "id" : NumberLong(786538601576366080),
  "favorite_count" : 0,
  "source" : "<a href=\"http://twitter.com/download/iphone\" rel=\"nofollow\">Twitter for iPhone</a>",
  "retweeted" : false,
  "coordinates" : null,
  "timestamp_ms" : "1476360390453",
}
```

Figure 30 - Tweet as a JSON object in MongoDB

8.3 Pre-processing of tweets

8.3.1 Cleaning Tweets

Before the training process or the classification process of the tweets, it needs to be pre-processed. Generally in Sentimental Analysis of documents data isn't consist of unwanted words of word phrases other than part of speech tags of stop-words. Unlike traditional documents tweets are consisted of various unwanted data in the tweets such as URLs, Usernames and Emoticons etc.

```
class cleanTweets():  
  
    def cleanTweet(self, tweet):  
        tweet = tweet.lower()  
        tweet = re.sub('((www\.[^\s]+)|(https?://[^\s]+))', 'url', tweet)  
        tweet = re.sub('@[^\s]+', 'user', tweet)  
        tweet = re.sub('[\s]+', ' ', tweet)  
        tweet = re.sub(r'#([^\s]+)', r'\1', tweet)  
        tweet = tweet.strip('\n')  
  
        return tweet
```

Figure 31 - Clean Tweets Class

CleanTweets() class is implemented to remove all unnecessary data which are consisted in the tweets. Tweets are passed to the *cleanTweet()* method in *cleanTweets()* class. Using the natively built in regular expression module in Python 2.7, firstly, the tweet is converted to lower case. Afterwards words and symbols like '*www*', '*http?://*' replaced with the word '*url*'. The word '*url*' will be later removed by adding to stop-words which will be elaborated later in this chapter. Usernames and mentioned user names are consistently mentioned in tweets. Both type of usernames are started from the '@' sign. Due to that feature usernames and mentioned usernames are replaced as '*user*'. Hashtags are commonly used feature in tweets. Due to that hashtags can be important in understanding the sentiment of the tweet. Only the hash (#) symbol will be removed and the particular word will be remained. Using the strip method the unwanted spaces will be removed in the tweets.

8.3.2 Generating Feature List

To generate the feature list FeatureGen() class is implemented. This class is comprised of three methods which are responsible for the process. In the process of generating the Feature List, part of speech tags in English language, Stop-words like ‘am’, ‘are’ and numerical values are removed as those features doesn’t carry any sentiment or affect to the sentiment of the tweet.

```
class FeatureGen():  
  
    def replaceTwoOrMore(self, s):  
        pattern = re.compile(r"(\1{1,})", re.DOTALL)  
        return pattern.sub(r"\1\1", s)
```

Figure 32 - Replace Two or More method

replaceTwoOrMore() function is the first method in the FeatureGen() class. Generally twitter users use informal language in communication. Due to this phenomenon it is difficult in natural language processing to eliminate words or letters that are repeated several times in a sentence or in a word. Using regular expressing module in python it is possible to match the repeated words or letters as a pattern. ‘re.DOTALL’ enables to identify each word in the tweet as a new word. In this method it checks for two or more repeating letters and it will be replaced by the same letter itself.

```

def getStopWordList(self, stopWordListFileName):

    stopWords = []
    stopWords.append('user')
    stopWords.append('url')
    stopWords.append('rt')

    fp = open(stopWordListFileName, 'r')
    line = fp.readline()
    while line:
        word = line.strip()
        stopWords.append(word)
        line = fp.readline()
    fp.close()
    return stopWords

```

Figure 33 - GetStopWordList method

Usage of stop words to pre-process the tweets is one of the most crucial part for the final accuracy of the classifier. As mentioned above stop words are words in English Language like negations, prepositions or simply words such as ‘the’, ‘are’ and etc. In the implementation of this system a collection of stop words were gathered from ravikiranj.net (2012). This particular list was consisted of stop words more than 500, which are only related to English Language. Other than this list of stop words, words such as ‘**user**’, ‘**url**’, ‘**rt**’ were append to the list. This is due to the replacement of these words in the cleanTweets() class. The text file consisted of stop words will be added to a list by reading line by line from the text file. Later all these stop words will be added to a list named as ‘**stopWords**’ which is defined inside the method.

The final method in the FeatureGen() class is getFeatureVector(). This is another crucial method responsible for the accuracy of the final classification. This method generate a python list named as ‘**FeatureVector**’. This list will be consisted of all the meaningful words included in all the tweets after removing unwanted features, words and symbols.

```

def getFeatureVector(self, tweet):
    stopWords = self.getStopWordList('stopwords.txt')
    featureVector = []
    words = tweet.split()
    for w in words:
        w = self.replaceTwoOrMore(w)
        w = w.strip('\[-.?!,:;()|0-9]')

        val = re.search(r"^[a-zA-Z][a-zA-Z0-9]*$", w)
        if(w in stopWords or val is None):
            continue
        else:
            featureVector.append(w.lower())
    return featureVector

```

Figure 34 - getFeatureVector Method

Initially an empty featureVector list and stop words from getStopWordList are initialized in the method. Then the words in the cleaned tweets are assigned to words which then will be directed towards a 'for' loop. There will be several more steps in pre-processing the cleaned tweets. First replaceTwoOrMore method will be called which was discussed earlier. Afterwards symbols like question mark, exclamation mark, brackets, colons and other punctuations as well as numerical values which are considered to be unimportant for the sentiment will be removed using the regular expression method built in. Additionally this methods checks whether the each word starts with an alphabetical letter, if not that word will not be considered furthermore. If the word starts with an alphabetical letter, those words will be compared with the stop words list. The words that are not in the stop words then will be append to the initially initialized featureVector list.

At this point of the implementation, the system has identified the meaningful words in the tweets and has generated a python list for the further process. Since a single word is append to the list at a time the single word can be considered as a '**Unigram**'. But according to the research of Pang et al. (2002) and Go et al. (2009) Naïve Bayes algorithm provides best results when '**Unigrams**' + '**Bigrams**' are generated as the featureVector list.

The below mentioned method generates the Bigrams list when the Unigrams list is provided.

```
def genBigrams(self, unigrams):  
    bigrams = []  
    for i in range(len(unigrams)-1):  
        bigrams.append(unigrams[i]+" "+unigrams[i+1])  
    return bigrams
```

Figure 35 - Generate Bigrams method

An empty python list is instantiated as 'bigrams'. Using the unigrams passed to the method it acquires the two closest words together and build a single phrase and include it to the bigrams list mentioned above. This method helps to increase the final accuracy of the '*Naïve Bayes Algorithm.*'

8.4 Classification Algorithm

The initial phase of the Classification algorithm is to train the algorithm. The above mentioned pre-processing of tweets are used in the classification algorithm. An empty list named as 'tweets' are defined to store the features and the sentiment relevant to that particular feature. Then the method is provided the training data set.

```
def trainNBClassifier (self):
    tweets = []
    reader = csv.reader(open('trainingData_1.csv', 'rt'))
    for row in reader:
        sentiment = row[0]
        tweet = row[1].decode('utf8')
        clean = cleanTweets()
        cleanedTweet = clean.cleanTweet(tweet)
        featureman = FeatureGen()
        features = featureman.getFeatureVector(cleanedTweet)
        self.featureVector.extend(features)
        tweets.append((features, sentiment))
    self.featureVector = list(set(self.featureVector))

    training = TrainingClassiffier()
    NBFeatures = training.identifyFeatures
    traa = nltk.classify.util.apply_features(NBFeatures, tweets)

    classifier = nltk.NaiveBayesClassifier.train(traa)
    self.saveFeatures(self.featureVector)
    self.saveNBClassifier(classifier)
```

Figure 36 - Train Naive Bayes Classifier

The training data set CSV is consisted of two columns, first column with the sentiment and the second column with the tweet. The cleanTweets() class is called and passes the tweet. Then the cleanTweets() class will return the cleaned tweet. The cleaned tweet will be passed to extract the relevant features. Those features will be included to the featureVector list. Set(self.featureVector) helps to identify the duplicate features and eliminate the duplicates.

Afterwards an special method named as 'extractFeatures()' in the TrainingClassifier class is executed. It will be explained in the next section. NLTK provides the valuable feature named,

`'apply_features'`. This extracts the whole features of the tweets as a bulk when `extractFeatures`, and tweets are given as the parameters. Afterwards the Naïve Bayes Algorithm will be trained using the extracted features.

Since training the Naïve Bayes algorithm is time consuming the `featureVector` and the trained Classifier are saved as Pickles, where pickle module is used to serialize the python objects.

```
def extractFeatures(self, tweet):
    words = set(tweet)
    features = {}
    for w in self.featureVector:
        features['contains(%s)' % w] = (w in words)
    return features
```

Figure 37 - Extract Features Algorithm

The above mentioned method checks whether for the features in the `featureVector` with each tweet. If any word is present in both `featureVector` and the tweet, the method will return `'True'` for the respective word. This will be returned as a dictionary named as `'features'`. This method helps to calculate the probability of the occurrence in a positive tweet or a negative tweet.

If the `featureVector` list is consisted as the following format;

```
featureVector = ['Kite', 'hey', 'hurts', 'hey', 'heard', 'head']
```

If the following tweet is passed to the `extractFeatures()` method;

```
Tweet = 'Donald Trump hurts Mexicans.'
```

The features dictionary will contain the following as the output;

```
{ 'contains (head)': False,
  'Contains (heard)': False,
  'Contains (hey)': False,
  'contains (hurts)': True}
```

The following code snippet demonstrates how the trained Naïve Bayes algorithms can be serialized in order to eliminate the training time of the dataset which consists of 50,000 tweets. Due to this condition, the classifier can be trained only once and save it as a pickle object.

```
save_classifier = open('classifier_20.pickle', 'wb')
pickle.dump(classifier, save_classifier)
save_classifier.close()
```

Figure 38 - Save Classifier as Pickle

The pickle object should be write into bytes in Python 2.7. The pickle.dump takes the trained classifier and the file name as the parameters. The same approach can be used to load the classifier by de-serializing the pickle, whenever necessary.

After the training process of the Naïve Bayes Algorithm, it can used to classification of new tweets. Since pickled training data is used the classification doesn't require much time. If the pickled classifier isn't used, the classification would require greater amount of time depending on the training data set.

```
connection = MongoClient('localhost', 27017)
db = connection.USelection
data = db.Trump_10_22
tweets_iterator = db.Trump_10_22.find()
pos = 0
neg = 0
for i in tweets_iterator:
    tweet = i['text'].encode('utf8')
    clean = cleanTweets()
    cleanedTweet = clean.cleanTweet(tweet)
    featureman = FeatureGen()
    test_features = featureman.getFeatureVector(cleanedTweet)
    classification = TrainingClassiffier()
    New_features = classification.identifyFeatures(test_features)
    sentiment = classifier.classify(New_features)
    if sentiment=='4':
        pos = pos +1
    elif sentiment=='0':
        neg = neg + 1
    print "testTweet = %s, sentiment = %s\n" % (tweet, sentiment)
print ("Positive:", pos)
```

Figure 39 - Classification of tweets

The above code snippet demonstrates the classification of the tweets. Since the historical data is stored in the MongoDB database, database connection should be established in order to read the relevant data. Afterwards, the tweets will be cleaned and extract the necessary features from the tweets. Then the extracted features would be sent to the classifier to classify the polarity of the respective tweet. The classifier would return a polarity value according to the polarity. A score of '4' for a positive tweet and a score of '0' for a negative tweet.

After the whole document is classified, an overall percentage can be obtained for the positively classified tweets and negatively classified tweets.

8.4.1 Live Tweet Analyser Module

Live tweet analyser module is consisted of two sections, live tweet streamer and Live Classifications. Live tweet streamer is responsible to stream real time tweets when the necessary keyword is given. The live classification module is responsible for the real time classification of the streamed tweets. Both these modules should be efficient in order to classify the sentiment of the tweets. A real time chart would be generated demonstrating the trend of the classification which was build using Python matplotlib.

```
class listener(StreamListener):
    def on_data(self, data):
        try:
            all_data = json.loads(data)
            tweet = all_data["text"].encode('utf-8')
            Live_tweet = live_classifier.TrainingClassiffier()
            sentiment_value = Live_tweet.trainClassifier(tweet)
            print(tweet, sentiment_value)
            output = open("twitter-out2.txt", "a")
            output.write(sentiment_value)
            output.write('\n')
            output.close()
            return True
        except:
            return True

    def on_error(self, status):
        print(status)
```

Figure 40- Live Twitter streamer

A listener class is implemented in order to retrieve real time tweets from the Twitter streaming API. A JSON object would be loaded to all_data variable. Using the JSON object, only the text will be extracted after encoding in to the standard 'UTF-8' format. The text value will be passed into the Live_tweet analyser class in order to perform live classification. The sentiment_value which is returned then be saved in to a text file. (It is possible to store the sentiment_value in the MongoDB database or in a CSV file.) This text file can be used in the generation of the live trend graph.

```
classifierFile = open("classifier_20.pickle", 'rb')
classifier = pickle.load(classifierFile)
classifierFile.close()

clean = cleanTweets()
cleanedTweet = clean.cleanTweet(live)
featureman = FeatureGen()
test_features = featureman.getFeatureVector(cleanedTweet)
tweet_analyser = TrainingClassiffier()
Live_classif = tweet_analyser.identifyFeatures(test_features)
return classifier.classify(Live_classif)
```

Figure 41 - Live Tweet Analyser

Initially, the pickled trained classifier is loaded to the system. Then it will be de-serialized in to python object. After that, the tweet will be cleaned, feature extraction process will be executed. Finally the features will be classified using the de-serialized classifier. The sentiment will be returned by the classifier itself to the live tweet streamer class.

8.4.2 Word Cloud Map to Discover Underlying Topics of Tweets

Word Clouds are frequently used in the modern data mining applications. Word Clouds helps to represent the most frequently used words in a given text. In this applications, it is used to discover the most frequently used words and topics. Using these cloud maps, the users of this system are able to discover the most ‘popular’ topics and words between the twitter users where it is practically unable to discover the topics or words by reading all the tweets through-out the day.

```
def __init__(self, font_path=None, width=400, height=200, margin=2,
             ranks_only=None, prefer_horizontal=0.9, mask=None, scale=1,
             color_func=random_color_func, max_words=200, min_font_size=4,
             stopwords=None, random_state=None, background_color='black',
             max_font_size=None, font_step=1, mode="RGB", relative_scaling=0, regexp=None):
    if font_path is None:
        font_path = FONT_PATH
    self.font_path = font_path
    self.width = width
```

Figure 42 - Constructor of the WordCloud class

The above code snippet demonstrates the constructor of the word cloud class. It requires the height, width of the canvas where the plotting will be executed, the number of maximum words that can be displayed in the canvas and many more.

```
frequencies = sorted(frequencies, key=item1, reverse=True)
frequencies = frequencies[:self.max_words]
max_frequency = float(frequencies[0][1])
frequencies = [(word, freq / max_frequency) for word, freq in frequencies]
self.words_ = frequencies

if self.random_state is not None:
    random_state = self.random_state
else:
    random_state = Random()

if len(frequencies) <= 0:
    print("We need at least 1 word to plot a word cloud, got %d."
          % len(frequencies))
```

Figure 43 - Generating frequency of words

Above code snippet describes how the frequency for each word is calculated. ‘*frequencies*’ is a defined array of tuples, where the tuple contains the word and its frequency. Initially the frequencies are sorted and normalized in order to organize the list of words given in the document.

If the given document is empty, it will display that ‘*at least a single word is necessary to generate the word cloud*’.

```
mask = np.array(Image.open(path.join(d, "mask.png")))
text = open("features.txt").read()
stopwords = set(STOPWORDS)
stopwords.add("int")
stopwords.add("ext")
wc = WordCloud(max_words=2000, mask=mask, stopwords=stopwords, margin=10,
               random_state=1).generate(text)
default_colors = wc.to_array()
plt.title("Custom colors")
plt.imshow(wc.recolor(color_func=grey_color_func, random_state=3))
wc.to_file("mask_1.png")
plt.axis("off")
plt.figure()
plt.title("Trump vs Hillary - 2016/10/20")
plt.imshow(default_colors)
plt.axis("off")
plt.show()
```

Figure 44 - Draw the Word Cloud using Matplotlib

To draw the word cloud, the mask is provided where the words will be fit in to. The document will be provided to the function using a text file. Then the word cloud object is generated. Word cloud object will return the words with the respective frequencies of the words. Using these features, the word cloud will be generated inside the given canvas using Matplotlib.

8.5 Final Implementation of the Application



Figure 46- Main GUI of the solution

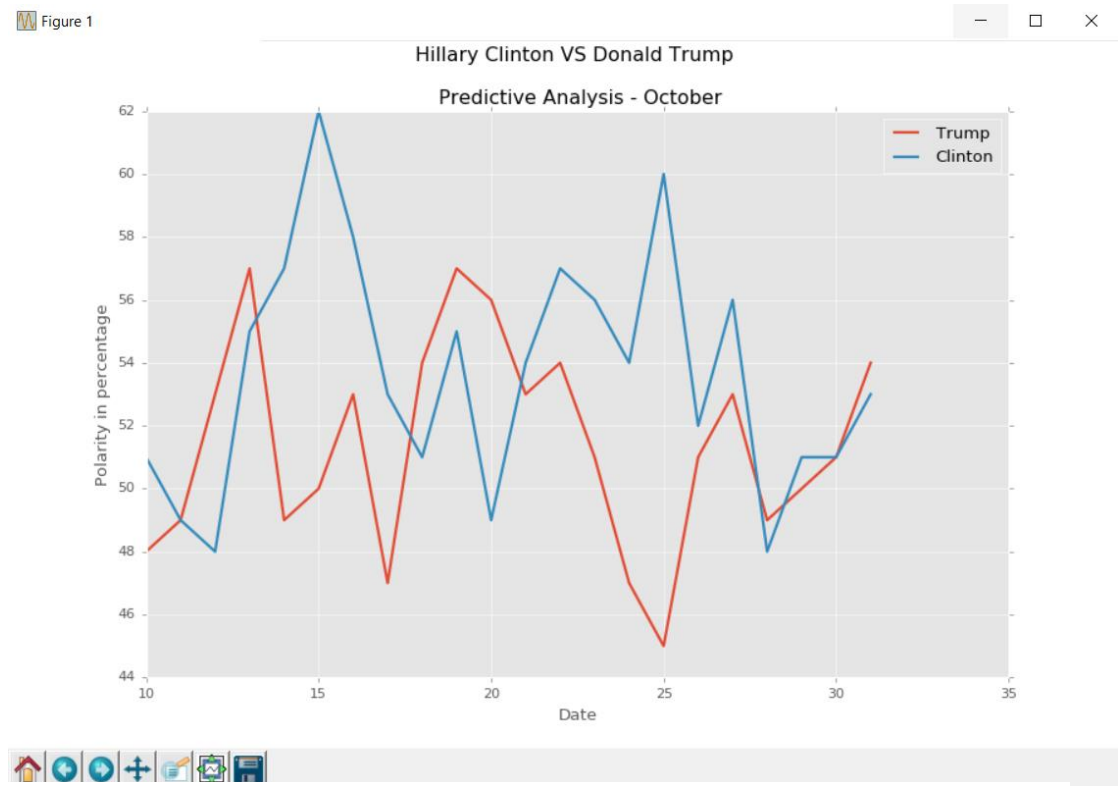


Figure 45 - Predictive Analysis Dashboard

9 TESTING AND PERFORMANCE EVALUATION

Testing and performance evaluation is considered to be a crucial section of any software engineering project. Since this project is based on predictive analysis, the system accuracy would be thoroughly tested in several iterations. It is important to test the system effectively with **white box** testing and **black box** testing in order to build robust software solutions. Other than the system accuracy testing, the system was tested under the following test categories.

1. Unit Testing
2. Integration Testing
3. Accuracy Testing

The test plan is illustrated below.

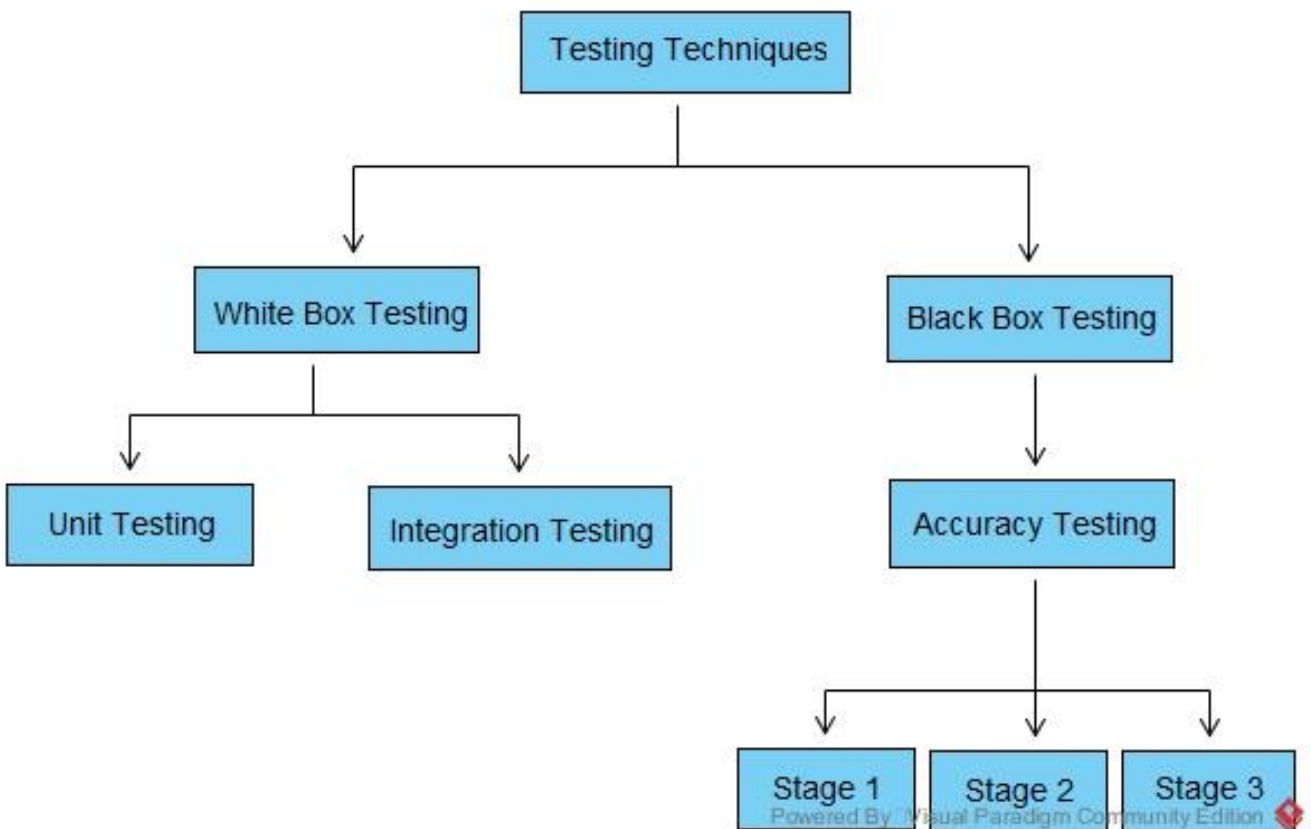


Figure 47 - Test Plan

9.1 Testing Environment

In order to perform the above mentioned test categories, a laptop computer with the below mentioned hardware specifications was used.

- Processor: Intel Core i5-6300HQ CPU @ 2.30GHz (4 CPUs)
- RAM : 8192 MB
- HDD: 120 GB
- VGA: 4160 MB - NVIDIA GeForce GTX 960M

A high capacity RAM was required in conducting the *Accuracy Testing*, since it needs higher computational power to train data sets consisted with more than 10,000 tweets. Due to this phenomenon a considerable amount of time was required to conduct the accuracy testing.

9.2 Unit Testing

Unit testing was conducted for each functions which was developed in the 1st iteration of the implementation. Functions under cleaning of tweets, extraction of features were given priority in the 1st iteration of implementation. Rest of the functions were implemented during the 2nd and 3rd iterations of the implementation.

‘**Python Unit Testing**’ framework, also known as ‘**PyUnit**’ is the standard unit testing framework for python language. PyUnit is also considered to be the Python language version of *JUnit*, which is the standard unit testing library for Java language. ‘**PyUnit 1.4.1**’ was selected as the unit testing framework for this project.

The python code written for the PyUnit tests will be included in the appendix.

9.2.1 Clean Tweets – Unit Tests

Clean tweets class has six different methods which handle the pre-processing of a tweet before the classification process. These methods had to be thoroughly unit tested as these methods directly responsible for the final accuracy of the classifier. Using PyUnit, it was accurately tested due to PyUnit's strong testing capabilities.

Test Case No: 01	Test Scenario: Remove usernames and mentioned usernames
Test Description	Tweets are consisted with usernames and mentioned usernames most of the time. Username doesn't carry a specific sentiment towards the tweet.
Input Data	Tweets E.g.- '@Trump this is for you'
Actions	Pass the live streaming tweets or the training dataset tweets
Expected Results	<ul style="list-style-type: none">• Select all the usernames using '@' symbol.• Replace the selected words with the word 'USER'.• Return the tweet with the replaced word. E.g.- 'user this is for you'
Actual Results	After passing the original tweet, the usernames and the mentioned usernames were replaced with the word 'USER'. E.g. - 'user this is for you'
Pass / Fail	Pass
Any changes Required	No changes required.

Table 8- Remove Usernames Test Case

Test Case No: 02	Test Scenario: Remove hyperlinks, URLs and other links
Test Description	Most of the tweets consisted with URLs or other links. Since this system doesn't access to related links, all those links will be considered as words with no sentiment.
Input Data	Tweets E.g. - 'www.cnn.com we got the results!'
Action	Pass the live streaming tweets or the training dataset tweets
Expected Results	<ul style="list-style-type: none"> Using standard formats like 'www', 'http: //' identify the URLs. Replace the relevant url with the word 'URL' Return the tweet with replaced tweet. E.g. - 'URL we got the results!'
Actual Results	After passing a tweet with URLs, all the linked will be replaced by the word 'URL'. E.g. - 'URL we got the results!'
Pass / Fail	Pass
Any changes Required	No changes required.

Table 9 - Remove URL Test Case

Test Case No: 03	Test Scenario: Convert all the words in the tweet to lower case.
Test Description	To detect the letters accurately, all the tweets would be converted to lower case.
Input Data	Tweets E.g. - 'DONALD TRUMP'
Action	Pass the live streaming tweets or the training dataset tweets

Expected Results	Using the regular expression module in python, all the letters would be converted into lower case letters. E.g - 'donald trump'
Actual Results	All the letters were converted to lower case letters in English language. E.g. - 'donald trump'
Pass / Fail	Pass
Any changes Required	No changes required.

Table 10- Convert to lower case Test Case

Test Case No: 04	Test Scenario: Remove hashtags
Test Description	One of the most popular feature of twitter is including hashtags in the tweets. These hashtags can be important in calculating the sentiment. But '#' symbol should be removed from the tweets.
Input Data	Tweets E.g. - '#ElectionNight'
Actions	Pass the live streaming tweets or the training dataset tweets
Expected Results	<ul style="list-style-type: none"> • Detect all the letters consisted with '#' symbol. • Remove the # symbol of the word. • Return the tweet with the removed # symbol. E.g. - 'electionnight'
Actual Results	After passing the original tweet, the words which were considered as hashtags would be converted to words with no # symbol. E.g. - 'electionnight'
Pass / Fail	Pass

Any changes Required	No changes required.
----------------------	----------------------

Table 11 - Remove hashtags Test Case

Test Case No: 05	Test Scenario: Remove Additional Spaces between words
Test Description	In many cases tweets are consisted with informal format. Due to this phenomenon there can be additional white spaces. This can be a problem in training the dataset or in the classification.
Input Data	Tweets E.g. - 'Hillary Clinton'
Actions	Pass the live streaming tweets or the training dataset tweets
Expected Results	<ul style="list-style-type: none"> Using the regular expression module in python detect the additional spaces in the tweet. Remove additional spaces within words. The tweet will be converted to lower case. Return the tweet with the removed additional spaces. E.g. - 'hillary clinton'
Actual Results	After passing the original tweet, the tweets which had additional spaces between the words are removed and normalized. E.g. - 'hillary clinton'
Pass / Fail	Pass
Any changes Required	No changes required.

Table 12 - Remove Additional spaces Test Case

Test Case No: 06	Test Scenario: Test the above mentioned test cases all at once.
Test Description	Testing all the pre-processing techniques from a single tweet.
Input Data	<p>Tweets</p> <p>E.g. - '@Rav94 Hillary Clinton is winning the final debate http: //cnn.com #FinalDebate.'</p>
Actions	Pass the live streaming tweets or the training dataset tweets
Expected Results	<ul style="list-style-type: none"> • Covert all the letters to lower case. • Remove additional spaces within words. • Remove usernames and URLs. • Remove hash symbol. • Return the tweet with the removed additional spaces. <p>E.g. - 'USER hillary Clinton is winning the final debate URL finaldebate.'</p>
Actual Results	<p>After passing the original tweet, pre-processed tweet is generated.</p> <p>E.g. - ' USER hillary Clinton is winning the final debate URL finaldebate.'</p>
Pass / Fail	Pass
Any changes Required	No changes required.

Table 13 - Clean tweets final Test Case

Unittests in TestCleanTweet: 6 total, 6 passed 1 ms

[Collapse](#) | [Expand](#)

test_cleanTweet.TestCleanTweet			1 ms
test_cleanTweet	passed		1 ms
test_cleanTweet_URL	passed		0 ms
test_cleanTweet_hash	passed		0 ms
test_cleanTweet_lower	passed		0 ms
test_cleanTweet_space	passed		0 ms
test_cleanTweet_spaces	passed		0 ms

Generated by PyCharm on 11/19/16 11:44 AM

Figure 48 - PyUnit Test Results

The above mentioned six test cases were passed with 100% successful rate using the PyUnit testing framework. As clean tweets represent the initial pre-processing stage of the tweets, there are remaining features to be processed before using for training purposes or classification purposes. Since the initial stage of implementation is successful, Unit testing on **extracting features** was conducted afterwards.

9.2.2 Extract Features – Unit Test

Unlike cleaning tweets, extracting features has few complex functions. These functions were tested using cleaned tweets which have been already pre-processed up to a certain limit. To improve the accuracy several iterations were conducted in this section.

Test Case No: 01	Test Scenario: Replace repeating letters in words
Test Description	Since twitter users use informal language when communicating, repeating letters are used to express excitement, sadness or happiness. Removing the additionally repeated letters is a crucial factor due to this phenomenon.
Input Data	Tweets E.g. - 'Hillllary Clinton just won the Texas state!'
Actions	Pass cleaned training tweets or the cleaned real time tweets
Expected Results	<ul style="list-style-type: none">• Detect all the words which has repeating letters in the word• Using regular expression, remove excess letters from the word. E.g. - 'hillary clinton just won the texas state!'
Actual Results	<ul style="list-style-type: none">• Many words were corrected using the function but words such as 'Hillary' was converted to 'Hilary' which is incorrect. E.g. - 'hilary clinton just won the texas state!'
Pass / Fail	Fail
Any changes Required	Needed to change the detection of letters in a word. Remove only heavily repeating letters.

Table 14 - Replace Repeating letters Test Case

Test Case No: 02	Test Scenario: Replace repeating letters in words
Test Description	Since the first implementation was failed, after several experiments another function was developed to remove repeating letters without any error.
Input Data	Tweets E.g. - 'Hillllary Clinton just won the Texas state!'
Actions	Pass cleaned training tweets or the cleaned real time tweets
Expected Results	<ul style="list-style-type: none"> • Detect all the words which has repeating letters in the word • Using regular expression, remove excess letters from the word. E.g. - 'hillary clinton just won the texas state!'
Actual Results	<ul style="list-style-type: none"> • This implementation worked perfectly even with words which has 2 repeating letters itself. E.g. - 'hillary clinton just won the texas state!'
Pass / Fail	Pass
Any changes Required	No changes required.

Table 15 - Second implementation of remove repeating letters test case

Test Case No: 03	Test Scenario: Read stop words from the text file and append new stop words.
Test Description	In English language stop words are considered to be words with no meaning, such as 'a', 'the', 'an' etc. In the cleaning process of the tweets, username and URLs are replaced with the terms such as 'USER', 'URL'. These words should also be removed from the tweets due to the null sentiment value.
Input Data	Stop words list
Actions	Append new stop words to the stop word list and create a python list.
Expected Results	<ul style="list-style-type: none"> • A python list is created including all the stop words from the stop words text file and newly included words such as 'USER' etc. • Return the created stop word list
Actual Results	<ul style="list-style-type: none"> • A list is created with all the stop words including newly added words.
Pass / Fail	Pass
Any changes Required	No changes required.

Table 16 - Generate stop words list Test Case

Test Case No: 04	Test Scenario: Remove numeric values and punctuations from the tweets
Test Description	Tweets are consisted with various punctuations and numerical values. All these don't carry any sentiment towards the tweet. Due to this, punctuations and numerical values are removed using regular expressions methods.
Input Data	Cleaned tweets

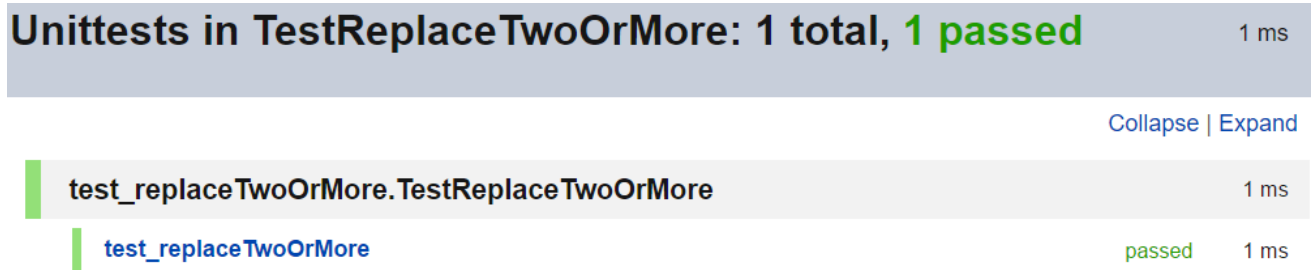
Actions	The input data is processed in order to find punctuations and numerical values.
Expected Results	<ul style="list-style-type: none"> • All the punctuations marks and numerical values would be removed from the tweets. • Return the remaining words in the tweet
Actual Results	<ul style="list-style-type: none"> • All the unwanted punctuations and numerals values are removed from the tweet. • Words which have meaning and carries a sentiment are remained in the tweet.
Pass / Fail	Pass
Any changes Required	No changes required.

Table 17 - Remove punctuations test case

Test Case No: 05	Test Scenario: Generate feature vector
Test Description	Up to this point in the implementation, all the unwanted words, punctuations and numeric values are removed from the tweets. From the remaining words, it is possible to generate a Feature Vector; a list containing all the meaningful words.
Input Data	Cleaned tweets which doesn't contain punctuations and numerical values.
Actions	The input data is compared with stop words. All the stop words which contained in the input data would be removed and words which have sentiment would be remained.
Expected Results	<ul style="list-style-type: none"> • A python list containing all the meaningful words which can affect to the sentiment of a tweet. • Return the python list to the training and classification process.
Actual Results	<ul style="list-style-type: none"> • The feature vector list was created, and it only contained words which has sentiment towards a polarity.

Pass / Fail	Pass
Any changes Required	No changes required.

Table 18 - Generate feature vector Test Case



Generated by PyCharm on 11/19/16 11:57 AM

Figure 49 - PyUnit Test Results - 2

9.2.3 Generating Word Cloud – Unit Test

Test Case No: 01	Test Scenario: Generate the word cloud
Test Description	When a text file containing tweets are given, after pre-processing of tweets generate a word cloud using the word frequencies calculated.
Input Data	A text file containing all the tweets in a particular day
Actions	The input data should be cleaned and pre-processed. Afterwards the stop words should be removed from data. Outcome of these data would be used to calculate the frequencies of words.
Expected Results	<ul style="list-style-type: none"> All the processed words would be inserted to a python list. After calculating the frequencies of each word, draw the word cloud in the given canvas. Generate the world cloud in matplotlib framework.

Actual Results	<ul style="list-style-type: none"> • Word cloud generated using the matplotlib. • The word cloud was generated in black and white and color formats.
Pass / Fail	Pass
Any changes Required	No changes required.

Table 19 - Generate world Cloud Test Case

8.5 Integration Testing

Integration testing was conducted during the 2nd and 3rd iterations of the implementation. During the 1st iteration, the basic components such as cleaning of tweets, feature extraction of tweets were implemented separately. The effectiveness of those components can only be tested after they are integrated with each component.

In this section, all the components which were implemented in the 1st iteration of implementation were integrated together to verify the quality of the system. Integration testing was conducted manually without using a testing framework.

8.5.1 Training Classifier – Integration Test

Test Case No.	Test Case Name	Expected Results	Actual Results	Test Results
1	Real time tweets are streamed via the custom built streamer.	Tweets should be stored in the MongoDB database, for historical data analysis.	Real time tweets are stored directly in the MongoDB database. Higher streaming speed doesn't cause any drawback.	Pass
2	Storing real time tweets during the Final Presidential Debate. (A large number of tweets streamed in a unit time.)	Tweets should be stored in the MongoDB database for historical data analysis.	During the final presidential debate a higher number of tweets were streamed, due to this phenomenon 10% of the tweets were corrupted.	Fail
3	Cleaning tweets of the final-	Tweets should be converted to lower case, usernames, URLs should be -	Exceptions were thrown when a corrupted tweet was	Fail

	presidential debate.	replaced with relevant words.	taken for the cleaning process.	
4	Training the classifier with training data collected.	Cleaning of tweets, feature extraction processes should be executed successfully. Feature Vector is generated and used for training the classifier.	Training the classifier was successful with the collected training data set.	Pass
5	Serialization of the trained classifier object.	The trained classifier should be serialized to a python pickle object.	A pickle object was created but it was partially corrupted.	Fail

Table 20 - Training Classifier - Integration Test

>	(43) ObjectId("5808185b9885a41394fa3292")	{ 31 fields }	Object
▼	(44) ObjectId("5808185b9885a41394fa3293")	{ 2 fields }	Object
	_id	ObjectId("5808185b9885a41394fa3293")	ObjectId
	> limit	{ 2 fields }	Object
>	(45) ObjectId("5808185b9885a41394fa3294")	{ 26 fields }	Object

Figure 50 - Test Case No. 2

Twitter streams tweets as JSON objects which generally contain 26 – 32 fields, but in some instances tweets were streamed with only 2 fields, which were the corrupted objects.

8.5.2 Word Cloud Generator – Integration Testing

Test Case No.	Test Case Name	Expected Results	Actual Results	Test Results
1	Generate word cloud when the feature vector is given.	The feature vector is consisted of pre-processed tweets. Accurate word cloud can be generated.	The word cloud class can directly access the Extract features class and retrieve the feature vector. The word cloud generated successfully.	Pass
2	Generate the word cloud using unprocessed tweets.	Access the clean tweets class and extract feature class. Generate the feature vector and the word cloud afterwards.	The word cloud was generated including stop words and unprocessed data.	Fail
3	Generate the word cloud with random colours assigned to each word.	The final outcome of the word cloud should include words representing different colours in order to identify the words clearly.	The word cloud was successfully generated with random colours assigned to each word.	Pass

Table 21 - Word Cloud Generator – Integration Testing

9.4 Accuracy Testing

Accuracy testing was performed during the final or the 3rd iteration of the implementation. This testing phase can be considered as the most crucial section of this project. Since one of the core functions of this project is to classify the polarity of tweets, the accuracy can be considered as the key measurement of the project.

Accuracy testing phase was conducted in 3 stages.

- Stage 1 – Testing the accuracy of the classifier using 6,000 and 9,000 training data set.
- Stage 2 – Testing the accuracy of the classifier using 10,000 and 20,000 training data set.
- Stage 3 – Testing the accuracy of the classifier using 20,000 and 50,000 training data set.

For each testing stage, the accuracy of the classifier will be calculated by the author, comparing system classified tweets manually. For each testing stage 100 tweets will be used to calculate the percentage of the accuracy.

9.4.1 Accuracy Testing – Stage 1

Accuracy testing stage 1 was initialized after conducting *Unit tests* and *Integration tests*. The primary goal of this stage was to train the *Naïve Bayes Algorithm* in order to classify the polarity of the historical tweets which were stored in the MongoDB.

For the training purpose, two data sets were used which had 6,000 and 9,000 tweets respectively. The first training data set was consisted of 2000 positive tweets, 2000 negative tweets and 2000 neutral tweets. Positive and negative training dataset was obtained by the research of Alec Go (Go et al., 2009), where the data was publically available for further research. The neutral data set was created by collecting tweets from the news sources such as CNN.com, BBC.com etc. An assumption was taken that the tweets published by the renowned news sources to be neutral.

The table accuracy test 1 displays the results of the classifier when 6,000 training tweets were used to train the classifier. As the testing data set, 100 tweets which were collected during 2016/10/16 on Donald Trump was used.

Polarity	Results of the Classifier	Results of the Author	Difference between Classifier and the Author
Positive	21 %	34 %	13 %
Negative	41 %	51 %	10 %
Neutral	38%	15 %	23 %

Table 22 - Accuracy Test 1

Accuracy of the Classifier = $100 - 46 = 54 \%$

Afterwards, the second training data set was used to train the classifier. This set was consisted of 9,000 tweets, 3000 annotated tweets from each polarity. The same testing data set which was used for the Accuracy test 1 was used.

Polarity	Results of the Classifier	Results of the Author	Difference between Classifier and the Author
Positive	23 %	34 %	11 %
Negative	33 %	51 %	18 %
Neutral	44%	15 %	29 %

Table 23 - Accuracy Test 2

Accuracy of the Classifier = $100 - 58 = 42 \%$

9.4.2 Conclusion of the Accuracy Testing – Stage 1

At the end of the stage 1 in accuracy testing, the classifier had a low accuracy rate though the unit testing and integration testing were conducted successfully and taken required measurements for the failures that were in the implementation.

Since the implementation was thoroughly tested, it was assumed by the author that to analyse the training data set which was used to train the classifier. The positive and the negative training tweets were annotated properly, but it was discovered that the neutral data set which was collected using renowned news providers wasn't accurate. The below displayed tweets would elaborate this phenomenon.

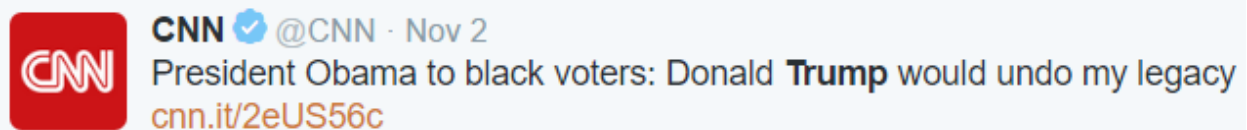


Figure 52 - CNN Tweet 1

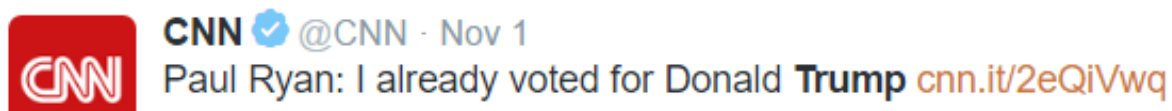


Figure 51 - CNN Tweet 2

The sentiment of the tweet in the figure 41 is negative and the sentiment in the figure 40 is positive. Though this was the actual scenario, since all the tweets which were published in news sources were taken as **neutral tweets**, the classifier considered negative words like 'undo' to be neutral rather than being negative. Due to this phenomenon most of the negative and positive words in the tweets were considered as neutral during the classification.

Although the news sources should act as neutral data providers due to the structure they use to communicate news on Twitter, it is unable to use these tweets as neutral data set. This is one of the major discoveries which was found during the accuracy testing – stage 1.

After this discovery, the author decided to remove the neutral training data set and to use only positive and negative data sets.

9.4.3 Accuracy Testing – Stage 2

Accuracy testing stage 2 was conducted after the changes that were taken in the stage 1. In this stage, 2 training data sets were used for the training purpose; one set with 10,000 training tweets (5000 positive tweets and 5000 negative tweets) and the second set with 20,000 training tweets (10,000 positive tweets and 10,000 negative tweets).

The table 24 – Accuracy Test 3, displays the accuracy results which were obtained after training 10,000 training tweets. Classifier was tested using the tweets set on 2016/10/22 – Donald Trump.

Polarity	Results of the Classifier	Results of the Author	Difference between Classifier and the Author
Positive	65 %	53 %	14 %
Negative	35 %	47 %	17 %

Table 24- Accuracy Test 3

Accuracy of the Classifier = $100 - 31 = 69\%$

The Naïve Bayes classifier identifies the words in a tweet as independent words such that each word will be assigned a probabilistic value towards positive or negative polarity. After the training process is completed which was based on the 10,000 training tweets, it is possible to check the accuracy of the classifier using ‘**Most Informative Features**’. This displays the words which are included in the trained classifier and the polarity towards negative or positive in a ratio.

The list mentioned below displays the 12 most informative features in the above trained classifier.

Most Informative Features

Contains (sad) = True	0 : 1	=	17.5 : 1.0
Contains (stuck) = True	0 : 1	=	14.6 : 1.0
Contains (upset) = True	0 : 1	=	13.0 : 1.0
Contains (headache) = True	0 : 1	=	12.2 : 1.0
Contains (sucks) = True	0 : 1	=	12.1 : 1.0

Contains (argh) = True	0 : 1	=	11.7 : 1.0
Contains (dentist) = True	0 : 1	=	11.7 : 1.0
Contains (welcome) = True	1 : 0	=	11.6 : 1.0
Contains (snow) = True	0 : 1	=	11.0 : 1.0
Contains (poor) = True	0 : 1	=	10.8 : 1.0
Contains (crying) = True	0 : 1	=	10.3 : 1.0
Contains (sick) = True	0 : 1	=	10.0 : 1.0

The ratio ‘1:0’ means the ratio between ‘**positive : negative**’, where ‘1’ denotes positive and ‘0’ denotes negative. The ratio in the right corner denotes the occurrence of each word either positive or negative tweet.

e.g. - Contains (sad) = True 0 : 1 = 17.5 : 1.0

The word ‘*sad*’ has occurred 17.5 times on negative tweets than in positive tweets. By analyzing these informative features, it is possible to determine whether the classifier has been trained accurately using the training data.

The table 25 – Accuracy Test 4, displays the accuracy results which were obtained after training 20,000 training tweets. Classifier was tested using the tweets set on 2016/10/22 – Donald Trump.

Polarity	Results of the Classifier	Results of the Author	Difference between Classifier and the Author
Positive	63 %	53 %	10 %
Negative	37 %	47 %	11 %

Table 25 - Accuracy Test 4

Accuracy of the Classifier = 100 - 21 = **79 %**

The list mentioned below displays 12 the most informative features in the above trained classifier.

Most Informative Features

Contains (lonely) = True	0 : 1	=	26.3 : 1.0
Contains (dentist) = True	0 : 1	=	18.3 : 1.0
Contains (snowing) = True	0 : 1	=	16.7 : 1.0
Contains (luv) = True	1 : 0	=	16.3 : 1.0
Contains (sick) = True	0 : 1	=	15.1 : 1.0
Contains (stomach) = True	0 : 1	=	14.3 : 1.0
Contains (tummy) = True	0 : 1	=	14.3 : 1.0
Contains (sad) = True	0 : 1	=	14.0 : 1.0
Contains (proud) = True	1 : 0	=	13.7 : 1.0
Contains (throat) = True	0 : 1	=	13.4 : 1.0
Contains (bummed) = True	0 : 1	=	13.0 : 1.0
Contains (headache) = True	0 : 1	=	12.6 : 1.0

The ratio '**1:0**' means the ratio between '**positive : negative**', where '1' denotes positive and '0' denotes negative. The ratio in the right corner denotes the occurrence of each word either positive or negative tweet. Based on the above mentioned informative features, it can be considered that most of the negative words have polarity towards negative and positive words have polarity towards positive, however there are words that are not either positive or negative such as '*stomach*' which has been assigned a polarity value depending on the training tweets. This phenomenon can caused for the reduction of the accuracy.

9.4.4 Conclusion of the Accuracy Testing – Stage 2

The accuracies of the 'stage 2' were considerably increased compared to the 'stage 1' accuracy testing. The prime reason for this change was due to the removal of neutral training data set from the system. Thus, the accuracies purely depend on the testing data set. It is possible to receive lower accuracy rates even after removing neutral data set from the training process. In order to improve the accuracy, than 79%; 'stage 3' was conducted by including more training data set.

9.4.5 Accuracy Testing – Stage 3

This testing stage was conducted to improve the accuracy of the classifier from 79%. In this stage two training data sets were used for the training purpose, the first set consisted of 20,000 training tweets and the second set consisted of 50,000 training tweets. Both training set were consisted of equal amounts of positive and negative annotated tweets.

The table 26 – Accuracy Test 5, displays the accuracy results which were obtained after training 20,000 training tweets. Classifier was tested using the tweets set on 2016/10/22 – Donald Trump.

Polarity	Results of the Classifier	Results of the Author	Difference between Classifier and the Author
Positive	63 %	53 %	10 %
Negative	37 %	47 %	11 %

Table 26 - Accuracy Test 5

Accuracy of the Classifier = $100 - 21 = 79\%$

The table 27 – Accuracy Test 6, displays the accuracy results which were obtained after training 50,000 training tweets. Classifier was tested using the tweets set on 2016/10/22 – Donald Trump.

Polarity	Results of the Classifier	Results of the Author	Difference between Classifier and the Author
Positive	47 %	53 %	6 %
Negative	58 %	47 %	11 %

Table 27 - Accuracy Test 6

Accuracy of the Classifier = $100 - 17 = 83\%$

The list mentioned below displays 14 the most informative features in the above trained classifier which used 50,000 training tweets.

Most Informative Features

Contains (bummed) = True	0 : 1	=	39.7 : 1.0
Contains (lonely) = True	0 : 1	=	25.9 : 1.0
Contains (followfriday) = True	1 : 0	=	23.7 : 1.0
Contains (tummy) = True	0 : 1	=	20.6 : 1.0
Contains (infection) = True	0 : 1	=	17.0 : 1.0
Contains (ankle) = True	0 : 1	=	16.3 : 1.0
contains (cancelled) = True	0 : 1	=	15.0 : 1.0
contains (heyy) = True	1 : 0	=	15.0 : 1.0
contains(boom) = True	1 : 0	=	15.0 : 1.0
contains (hurts) = True	0 : 1	=	14.9 : 1.0
contains (sad) = True	0 : 1	=	14.8 : 1.0
contains (depressed) = True	0 : 1	=	14.6 : 1.0
contains (hating) = True	0 : 1	=	14.3 : 1.0
contains (worst) = True	0 : 1	=	13.9 : 1.0

9.4.6 Conclusion of the Accuracy Testing – Stage 3

After training 50,000 training tweets it is observed that the accuracy was increased from 79% to 83 %. Though this accuracy may vary according to the testing dataset, it is possible to improve the accuracy by training more tweets. By observing the most informative features in this stage it can be noted that most of the words are belonged to the right sentiment except for few words such as *'followfriday'*, *'tummy'* etc. This accuracy range was considered as a stable rate and finalized the accuracy testing phase of the system.

9.4.7 Testing Live Sentiment Classifier

Since a stable accuracy rate was achieved through the 3 accuracy testing stages, the author decided to use the final classifier for the rest of the project. 2016 November 8th, the U.S. Presidential Election date was selected to test the live sentiment classifier. The live sentiment classifier was tested before the final results were announced.

Topic – Hillary Clinton

Date – 2016/11/09 (2016/11/08 Washington, DC USA)

Time – 9:10 am – 9:30 am (10:40 pm – 11pm Washington, DC USA)

Number of tweets analysed - 113

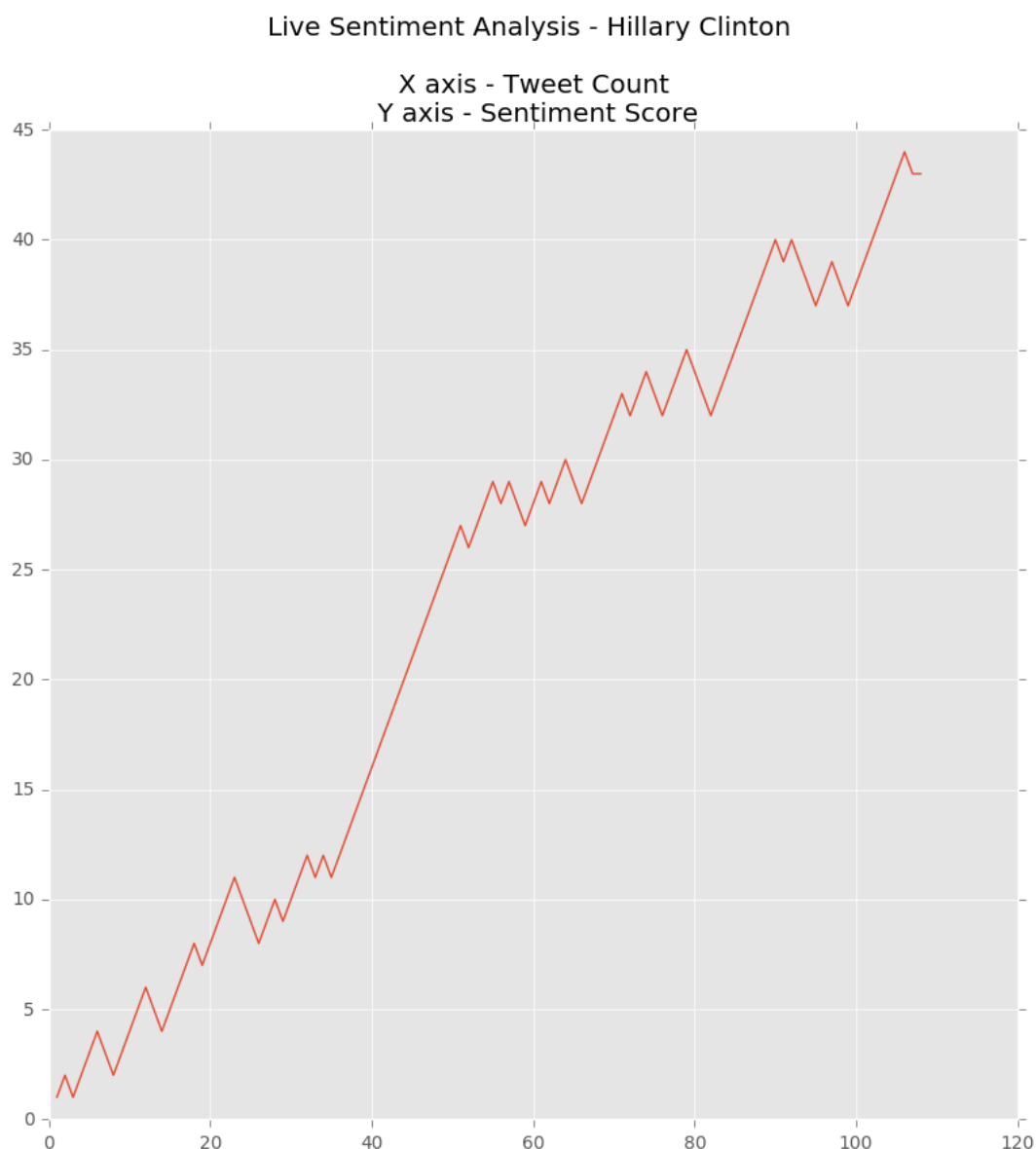


Figure 53 - Live Sentiment Analysis - Hillary Clinton

Topic – Donald Trump

Date – 2016/11/09 (2016/11/08 Washington, DC USA)

Time – 9:10 am – 9:30 am (10:40 pm – 11pm Washington, DC USA)

Number of tweets analysed - 112

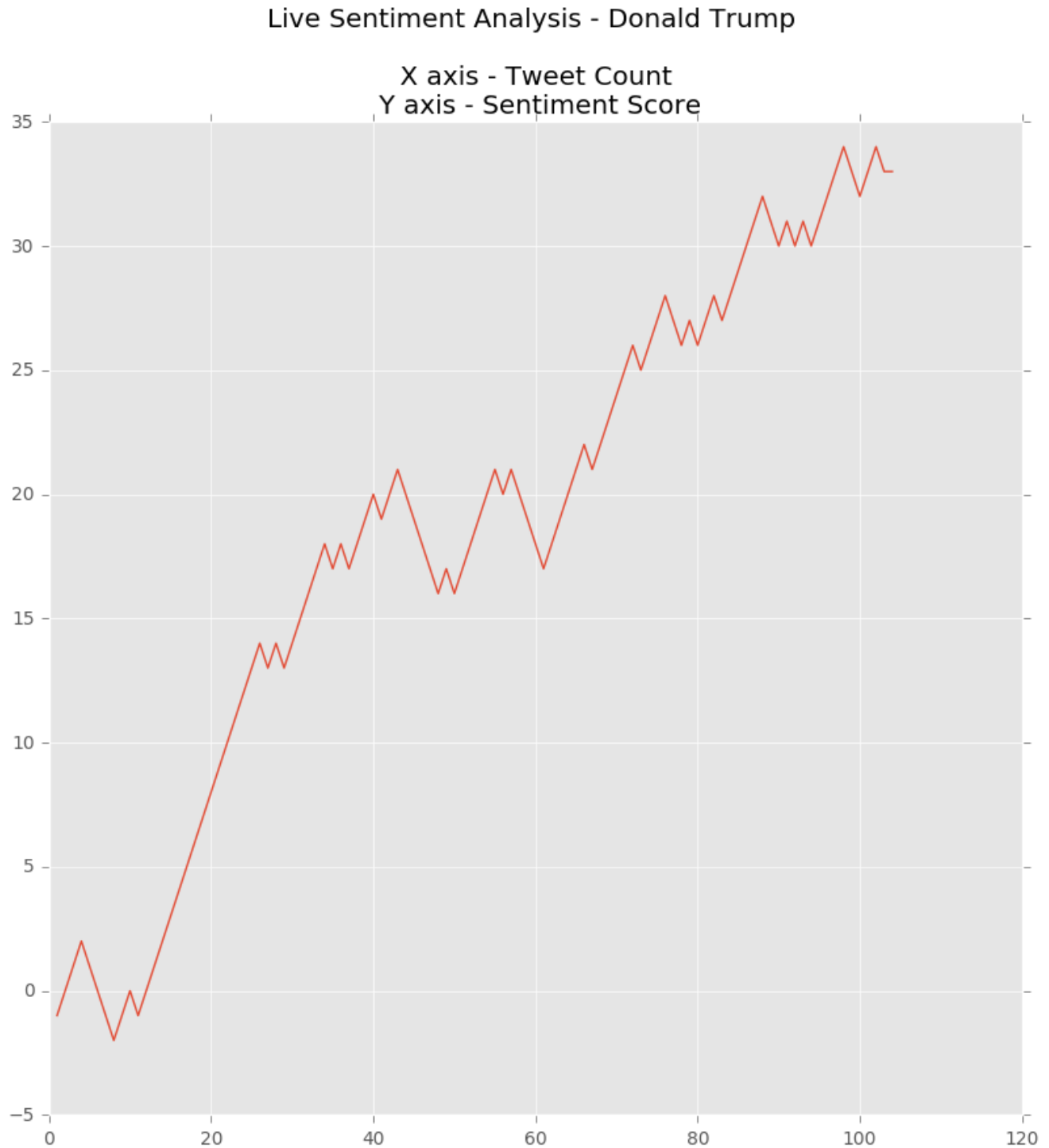


Figure 54 - Live Sentiment Analysis – Donald Trump

The above charts were generated during the live sentiment classifications of the each presidential candidate. In both scenarios were tested in equal conditions to achieve non-biased results and real time tweets were used for the classification. The real time tweets were streamed through the live streamer and the sentiment values (positive or negative) were graphed in the same time frame. The ‘X axis’ indicates the number of tweets streamed while the ‘Y axis’ indicates the sentiment score. The sentiment score is calculated by assigning +1 for every positive tweet and -1 for every negative tweets, in simple terms the candidate who can achieves the height sentiment score in a given number of tweets, has the highest favourable conditions in the election.

According to the figure 43 – the live sentiment analysis of Hillary Clinton, the node started from positive 1 which means the initial tweet had carried a positive sentiment towards her. The rest of the tweets contains peaks and troughs indicating negative and positive sentiment score towards Hillary Clinton. At the end of **100 tweets** according to the graph, total sentiment score of **37 points** was calculated towards **Hillary Clinton**. Likewise according to the figure 44 – the live sentiment analysis of Donald Trump, the node started from negative 1 which means the initial tweet had carried a negative sentiment towards him. In the initial stage Trump’s sentiment score was scoped in negative values but gradually the score was increased towards the positive section. At the end of **100 tweets** according to the graph, total sentiment score of **32 points** was calculated towards **Donald Trump** by the classifier.

The Live sentiment classifier was again tested 2 hours before the officials result was announced. Compared to the first analysis of the candidates, the sentiment score was ranged between lower values in the second analysis. The second live sentiment analysis results are mentioned below.

Topic – Hillary Clinton

Date – 2016/11/09 (2016/11/09 Washington, DC USA)

Time – 11:10 am – 11:30 am (12:40 am – 1am Washington, DC USA)

Number of tweets analysed – 80

Live Sentiment Analysis - Hillary Clinton



Figure 55 - Final Live Sentiment Analysis - Hillary Clinton

Topic – Donald Trump

Date – 2016/11/09 (2016/11/09 Washington, DC USA)

Time – 11:10 am – 11:30 am (12:40 am – 1am Washington, DC USA)

Number of tweets analysed – 113

Live Sentiment Analysis - Donald Trump

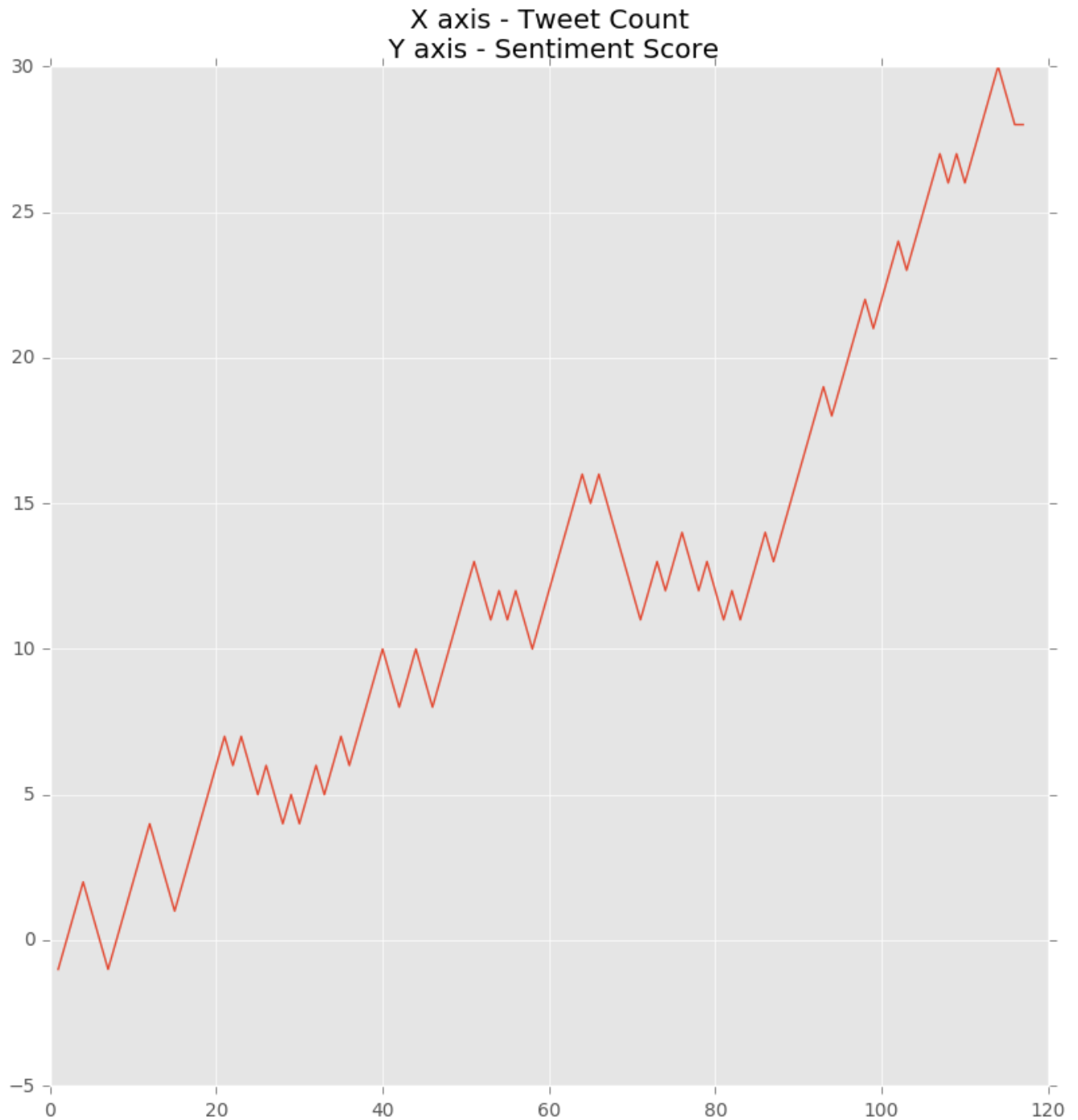


Figure 56 - Final Live Sentiment Analysis - Donald Trump

The final live sentiment analysis was performed during the final stage of calculating the electoral votes. From the graphs it was found that more negative sentiments were aimed towards Donald Trump than Hillary Clinton. This is mainly due to by the time Trump had won more electoral votes than Clinton and leading the way. Due to this situation more tweets with hatred were published mentioning 'Donald Trump'. In the other hand more tweets having sympathetic sentiments were published mentioning 'Hillary Clinton'. But in a certain stage Donald Trump was able to score a very high sentiment score than his opponent Clinton due to more favourable tweets towards him.

Unfortunately during the live sentiment analysis of Hillary Clinton, the live tweet streamer was stopped due to the higher rate of tweets. Due to this the streamer was able to analyse **80 tweets** during the given time period. The final **sentiment score** by the 80th tweet was **24**. After analysing 80 tweets mentioning 'Donald Trump', he had achieved a sentiment score of 13 but surprisingly after analysing 20 more tweets he had reached up to a **sentiment score** of **24** and kept the trend towards the positive way with a higher rate than the initial stages.

Summarizing the above graphs, it was found out that though Hillary Clinton had a high positive sentiment score in the early stages but with the electoral results indicating Donald Trump had a lead in votes, more positive sentiment was scored by Donald Trump during the final live sentiment analysis.

After performing thorough Unit testing and Integration testing on the implementation of the word cloud, it was possible to generate much more accurate word clouds to discover underlying topics of the tweets in a given time frame. Since testing data was started to collect a month prior to the election, it was possible to observe underlying topics and the most frequently used word phrases. This provided opportunity to evaluate the trended topics of the general public.



The above word cloud was generated on 2016/10/11, to evaluate the final two presidential candidates. After observing the word cloud that more tweets have been published mentioning the words 'Donald' and 'Trump' than 'Hillary' or 'Clinton'. This phenomenon was proved correct, by evaluating the number of tweets which were streamed on both candidates in a unit time.

This word cloud also proved correct after comparing most trended topics during that week. Topics such as '*Russia*', '*GOP*', '*email*', '*women*', '*bill*', and '*tape*' were the most popular topics during that day. Though irrelevant words such as '*re*', '*twitter*' were displayed in the word cloud, it was further improved to remove such word phrases from the word clouds. However as a drawback, it was only possible to illustrate the 2000 most frequently used words in a word cloud.

9.5 Performance Evaluation

This section will summarize the Testing phase of this project including the results observed from Unit, Integration and Accuracy tests. As mentioned earlier in this chapter, rigours testing iterations had to be performed to obtain accurate information through this system.

In the Unit testing phase all the functions and the algorithms were testes individually. In this phase it was noted that all the individual functions were properly working up to the standards required. Only few functions were needed minor adjustments in this stage.

Integration testing phase was highly effective for the improvement of this project. During the integration testing phase all the individual functions and algorithms were integrated each other, in order to build the complete system. In this stage many functions were failed to integrate with the rest of the components. Due to this, several functions were needed to be altered and the system designs were adjusted accordingly.

The implementation of MongoDB database to store tweets increased the performance of data retrieving and saving drastically, than using a traditional SQL database. This was achieved due to the structure of the NoSQL databases, which are ideal for **Data Mining** projects.

The final and the most crucial testing phase was the 'Accuracy Testing phase'. To achieve a reliable accuracy this stage was divided in to 3 phases. By training fewer amount of tweets in the 1st phase the classifier was tested with two testing data sets. The 1st phase revealed a major breakthrough in this project. The accuracy of the 1st phase was less due to the usage of neutral data set. The problem was identified that the quality of the neutral data set was low and it had directly affected the performance of the classifier.

The 2nd phase of the accuracy testing was conducted without using a neutral data set and more positive and negative training data. This achieved the performance of the classifier to a stable stage. To improve the accuracy much higher, 50,000 training data set was used to train the classifier. It was able to provide a satisfactory accuracy rate for this project. Since the performance was gradually increased throughout the testing stages, the classifier was able to perform real time sentiment analysis on the U.S. Presidential election date before the final results were announced. The word cloud was also tested during all the test phases. It was able to clearly illustrate the trending topics, word phrases and to demonstrate the results in a user friendly manner.

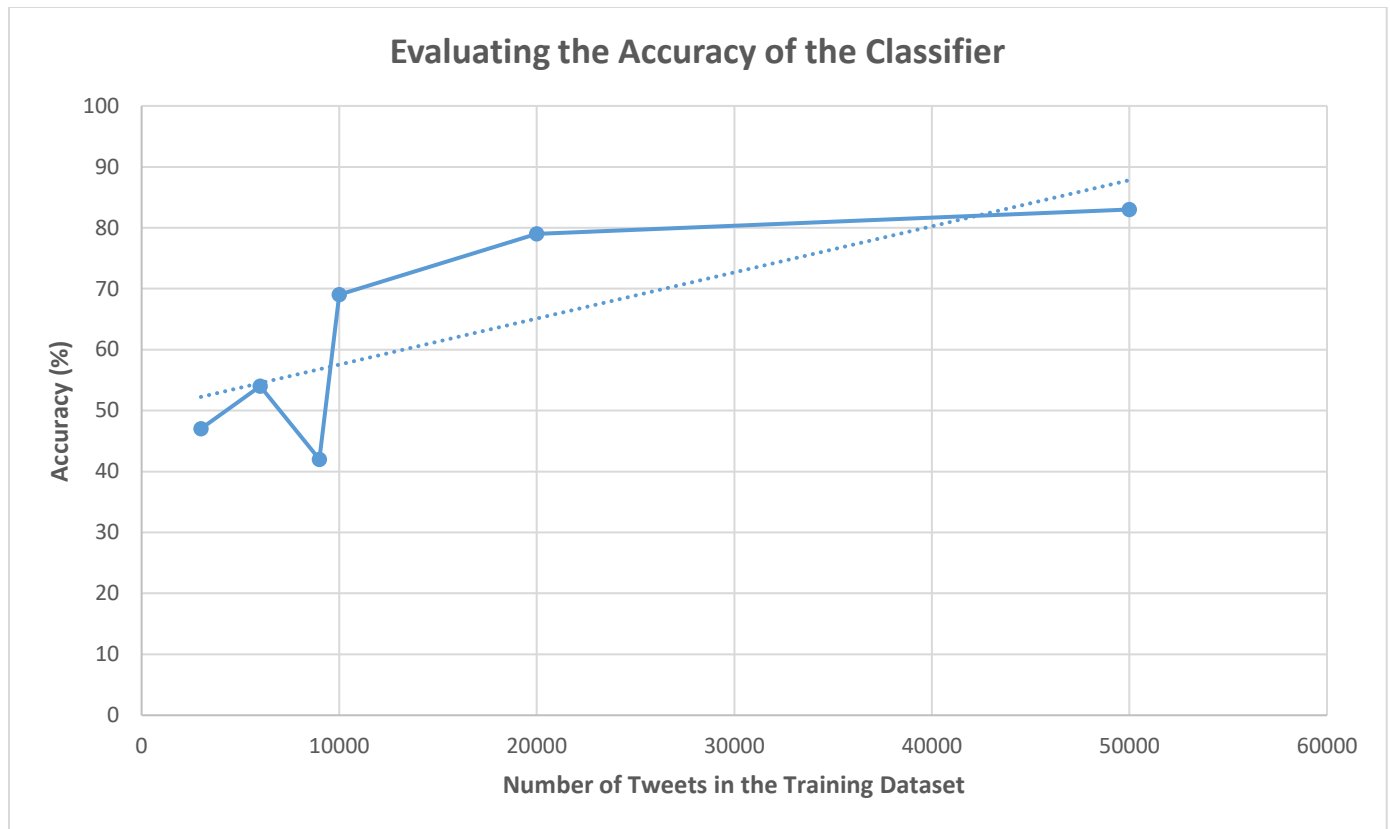


Figure 58 - Evaluating Accuracy of the Classifier

The above chart displays the summarised details of the Accuracy testing phase. When the classifier was trained with neutral training dataset, the accuracy rate of the classifier was poor. But after removing the neutral training dataset the accuracy of the classifier tend to get improved. After using 50,000 training tweets, the classifier achieved the maximum accuracy rate of 83%.

The dotted line displayed in the chart illustrates the trend line of the accuracy. By the observations of this chart, it can be assumed that higher the number of training data higher the accuracy of the classifier. However the quality of the training data also effects on this hypothesis.

9 PREDICTIVE ANALYSIS

After analysing the historical data collected from **2016/10/11 – 2016/11/07**, the system was able to discover the following facts,

- Frequency of mentioning the names ‘Hillary Clinton’ and ‘Donald Trump’
- The most frequently discussed and trended topics.
- Inability to detect sarcasm in the tweets.
- When both candidate names were mentioned in a single tweet, the system wasn’t accurate enough to detect the polarity towards each candidate.

During the early iterations of implementation, it was highlighted that the name ‘Donald Trump’ appeared in tweets **1.5 to 2 times** as frequently as the tweets mentioning ‘Hillary Clinton’. This phenomenon was observed by collecting tweets in a unit time, which included the names of the candidates. Keywords such as ‘Hillary’, ‘Clinton’, ‘Donald’ and ‘Trump’ were used to gather the tweets. The chart below was generated using the historical data collected to illustrate the frequency of tweets per each candidate.

How frequently are Clinton/Trump mentioned in Tweets?

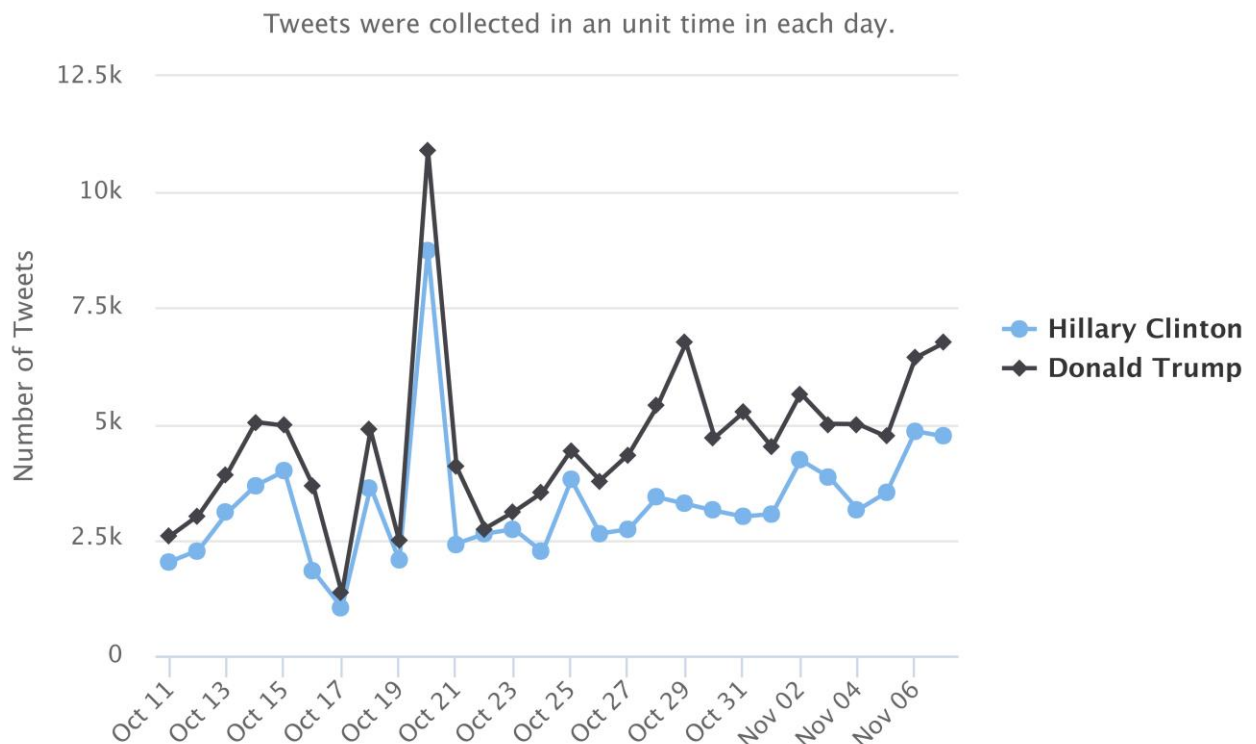


Figure 59 - Frequency of Tweets

This was the 1st phenomenon that was discovered using the historical data. Since the initial day the data started collecting till the Election Day on 8th, Tweets which mentioned ‘Donal Trump’ was higher than the tweets mentioned ‘Hillary Clinton’. During the Final Presidential Debate on October 20th both candidates were mentioned in higher number of tweets than the normal days. This clearly demonstrate that the general public were interested in the debate as well as highly active in the Twitter than the rest of the days. Without considering the sentiment of these tweets, it can be assumed that the name ‘*Donal Trump*’ was considerably popular than the name ‘*Hillary Clinton*’.

The above hypothesis can be proved correct using the official statistics provided by the Twitter.com regarding the tweets mentioning each candidate.

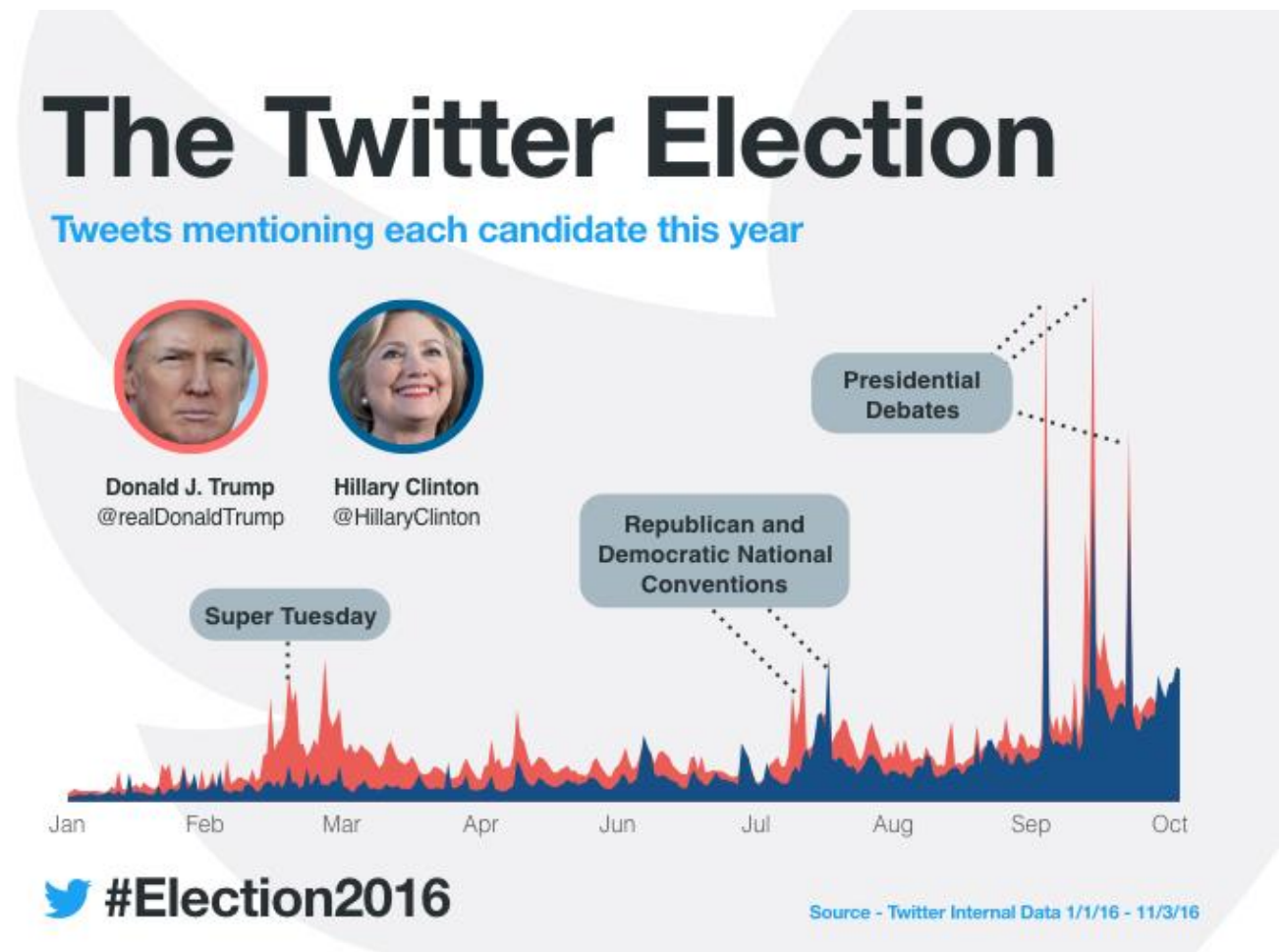


Figure 60 - Official Twitter Statistics

(Source: blog.twitter.com, 2016)

highlighted the attention towards Trump were the prime topics during the final debate. The word phrases such as ‘*iamwithher*’ was also frequently used to support Hillary Clinton. These results depict the most interested topics that were used during these time period.

However there were few drawbacks of the system that were discovered when analysing the tweets manually in the testing phase.

- **Inability to accurately detect the sentiment of the sarcastic tweets.**

e.g. - @RaviJ *If Trump wins the election, God bless America!!! – Positive*

Though the underlying meaning of this tweet carries a negative sentiment, due to the words like ‘wins’ and ‘bless’ the system considered this to be a tweet with positive polarity.

- **When both candidate names were mentioned in a single tweet, the system wasn’t accurate enough to detect the polarity towards each candidate.**

e.g. - @RealDonaldTrump *Real Americans are with you not with crooked Hillary! – Positive*

The first part of the tweets carries a positive sentiment towards Trump but the final part of the tweet carries a negative sentiment towards Clinton. The implemented solution is possible to detect only a single sentiment in a tweet.

Using the tweets that were collected from **2016/10/11 – 2016/11/07**, predictive analysis graphs were generated using the implemented solution for both candidates. The graphs which are demonstrated below displays the predictive analysis for the month of October,

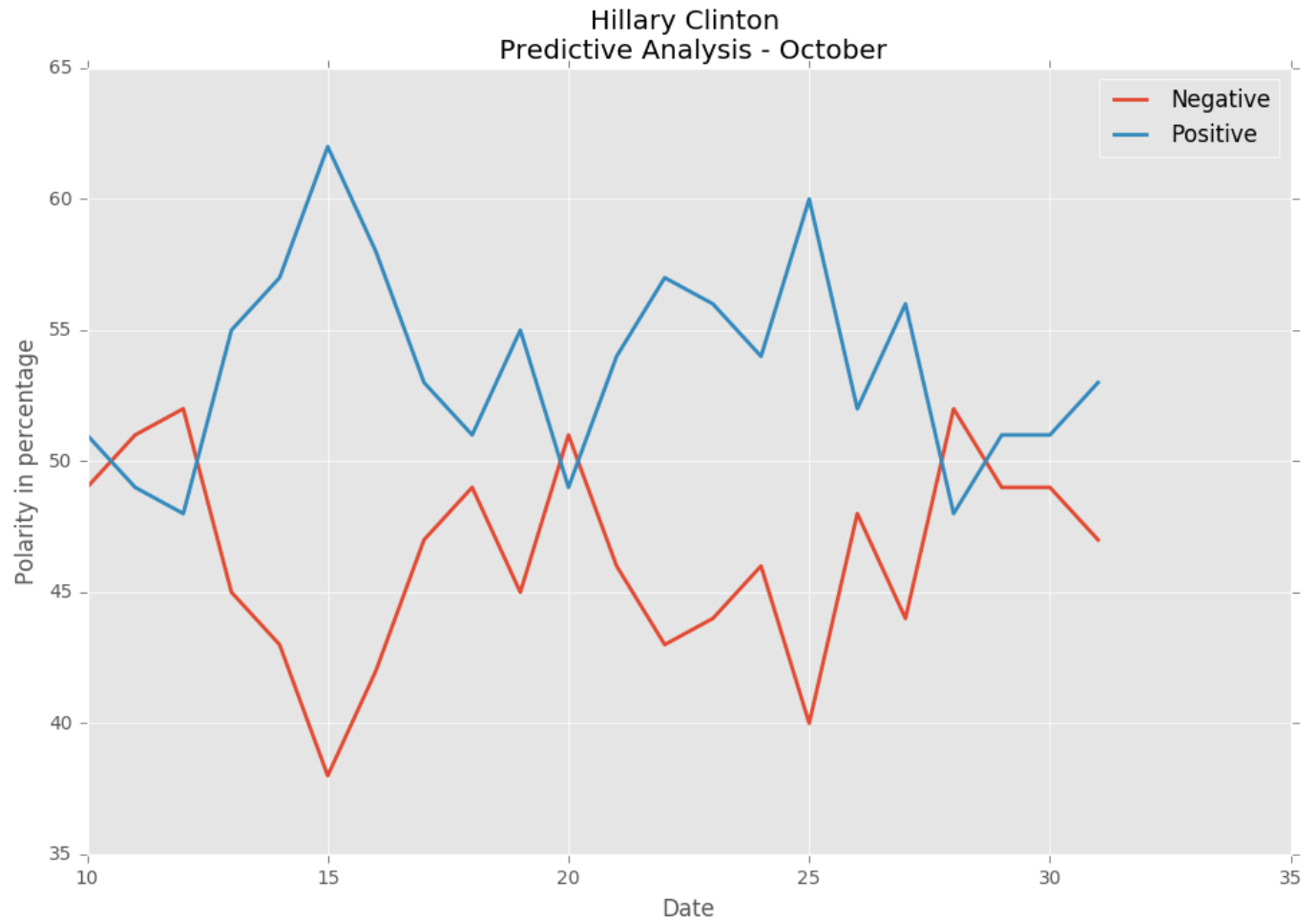


Figure 62 - Predictive Analysis - Hillary Clinton

The blue line displays the variation of the positive polarity tweets in percentage while the red line shows the negative polarity tweets in percentage. By analysing the above graph, it can be observed that during this particular period Hillary Clinton has achieved a positive polarity sentiment in most of the days except in few days such as October 20th (Final Presidential Debate day) etc. After analysing the above graph it is possible to evaluate that more positive sentiments are published by the public towards 'Hillary Clinton'.

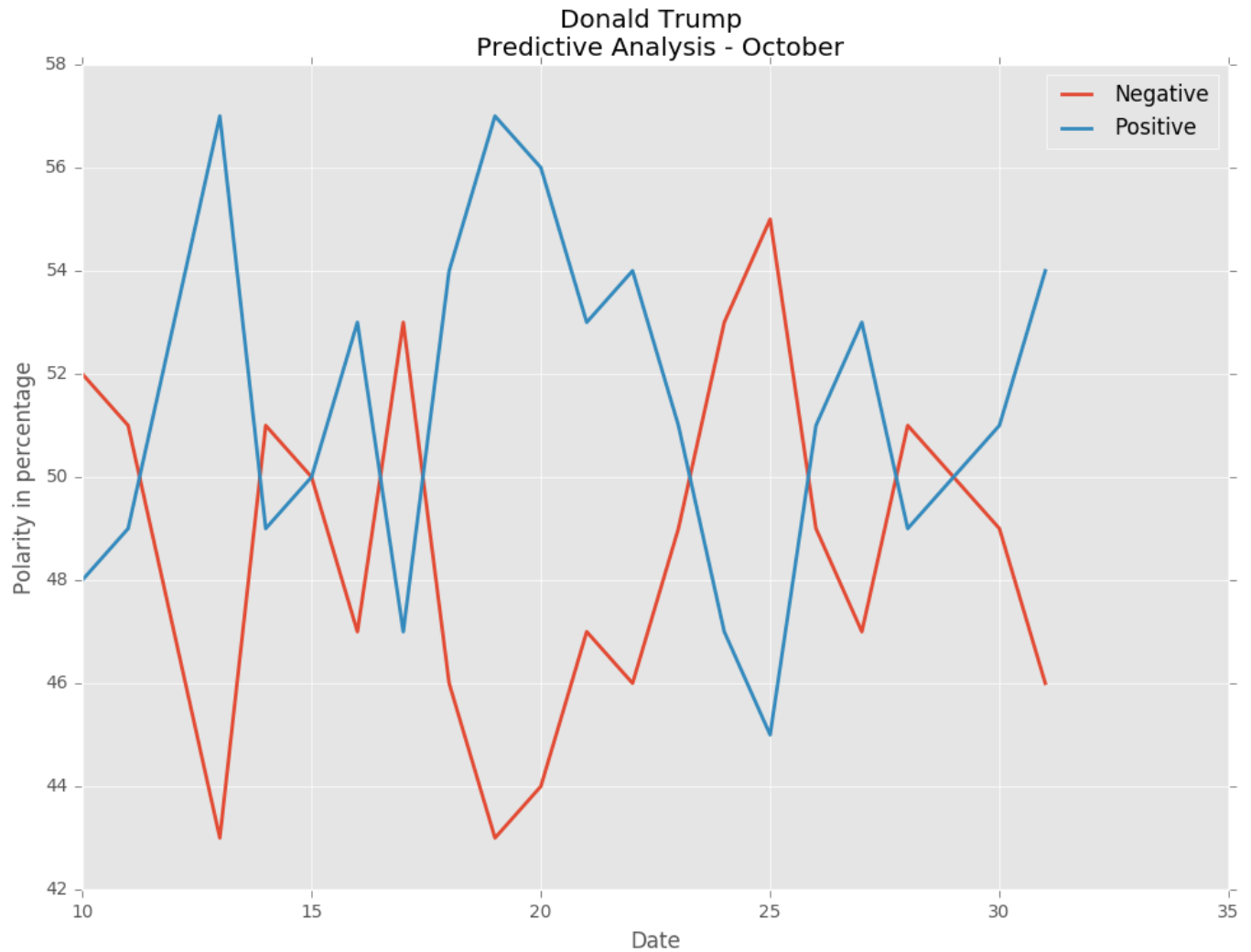


Figure 63- Predictive Analysis - Donald Trump

The above illustrated graph demonstrates the Predictive Analysis of ‘Donald Trump’ for the month of October. The blue line displays the variation of the positive polarity tweets in percentage while the red line shows the negative polarity tweets in percentage. In contrast to the Figure 61, this graph shows high magnitude of variations in the polarity. In most of the days during this period, the positive sentiment curve relies around 50%.

The graph mentioned below is the comparisons of the above two graphs, the positive sentiment analysis of both Hillary Clinton and Donald Trump.

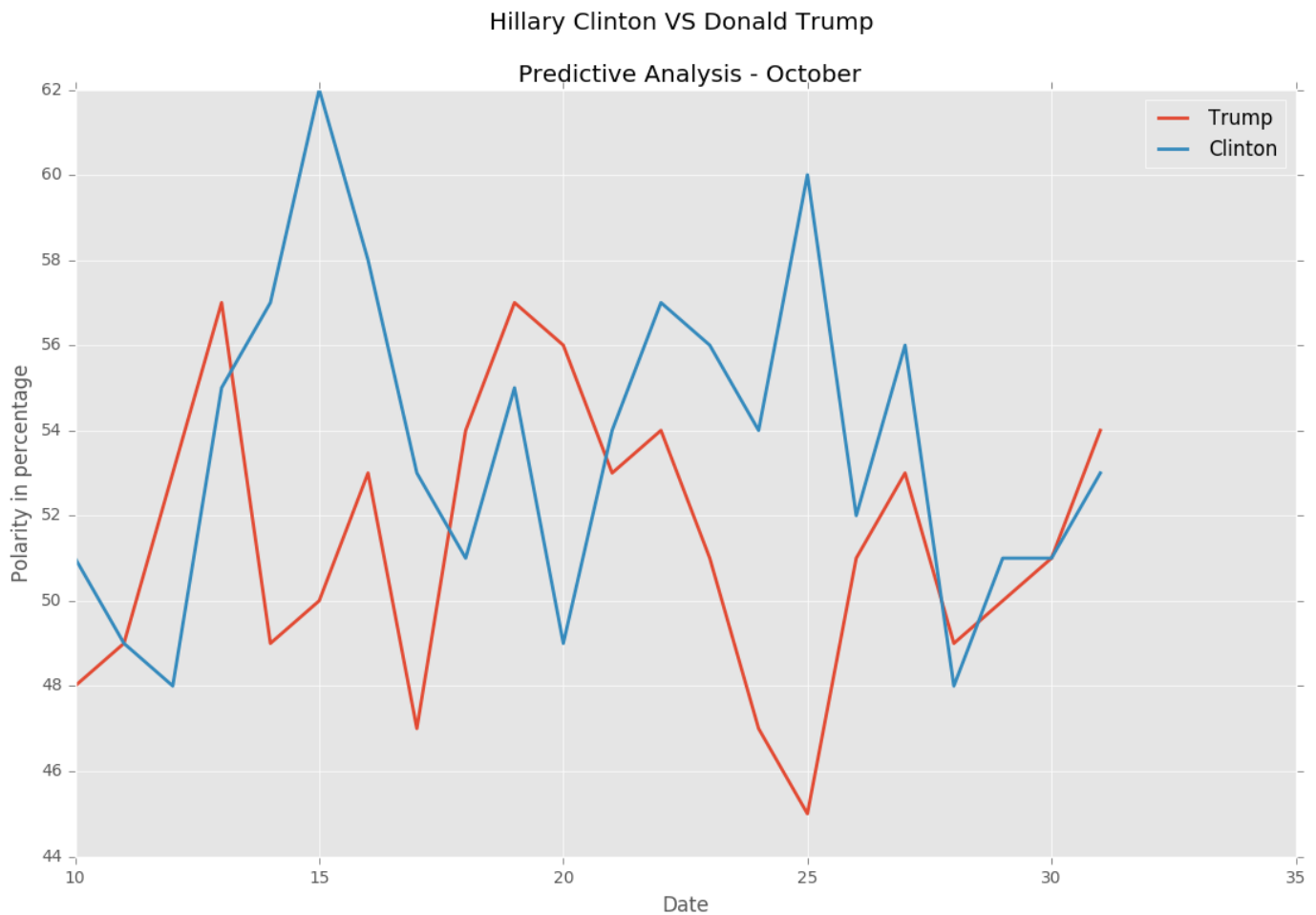


Figure 64 - Clinton VS Trump Analysis

From the above chart it can be observed that, in most of the days Hillary Clinton has obtained more positive sentiments than Donald Trump. For both analysis, equal conditions were used. By using the word cloud, it is possible to analyse the keywords, trended topics recorded in each day.

CRITICAL EVALUATION

This project was primarily implemented to predictively analyse the United States Presidential Election 2016 using machine learning techniques. The proposed solution was implemented as a desktop application which includes historical sentiment analysis as well as real time sentiment analysis. Other than the primary objective, the implemented solution can be used to discover underlying topics or the keywords in a large amount of data.

A thorough analysis was conducted before the project was initiated. Though social media is used by millions of people to present their ideas, opinions and statements there are very few number of systems that are implemented to understand the sentiments of these data. This project was proposed to analyse the sentiments of those opinions published by the public regarding elections. For this purpose, U.S. Presidential Election 2016 was chosen since the public is highly active in social media during the election time period. The popular micro blogging website '*Twitter.com*' was chosen as the data source for this project due to many reasons which were discussed in the Solution Concept chapter.

In the current society the results of any election were predicted by conducting surveys, telephone interviews or one to one interviews. These techniques have been used for over centuries. Initially various research papers and other documents were collected and studied to identify whether it is possible to use social media data in order to perform predictions rather than using the old techniques. Afterwards the requirements for this project were captured by observing various research papers and related documents. By analysing the collected requirements, functional requirements, non-functional requirements, software and hardware requirements were determined.

Afterwards secondary research was conducted to obtain required technical knowledge. During this phase thorough research was conducted on machine learning techniques, sentiment analysis, natural language processing and data mining. The initial section of the literature review was consisted of the observations which were discovered in the secondary research. To understand the fundamentals of these technical concepts, similar research papers, projects and implemented solutions were observed iteratively. Through these observations similar concepts and similar approaches were identified, which were extensively useful in the implementation stage.

After observing the related concepts and approaches, the Solution Concept chapter was written. In this chapter, all the algorithms, functions and concepts were critically evaluated using the results and findings which were documented in the similar projects. Naïve Bayes was selected as the machine learning algorithm to classify the polarity, Python was selected as the programming language, MongoDB database was selected as the primary database and Natural Language Toolkit was selected to perform the natural language processing.

Before the implementation, using UML diagrams the design of the system was created. Activity diagrams, Class diagram and Sequence diagrams were taken into consideration during the design chapter. Proper modelling of the system was finalized during several iterations. Afterwards, the implementation of the system was initialized. Both black box and white box testing was conducted to evaluate the performance of the implemented system. Testing of this project was conducted in three iterations. During the 1st iteration, '*Unit Testing*' was conducted, in the 2nd iteration '*Integration testing*' was conducted. In the 2nd iteration of the system it was observed that several functions were failed in the integration. After the necessary improvements were done, the '*Accuracy testing*' was performed to evaluate the accuracy of the system.

Accuracy testing phase was divided into three stages. During the 1st stage the classifier was trained using 6000 and 9000 tweets. The system performed with poor accuracy rate. After an evaluation done in the end of this stage it was found out that the bad quality of the neutral data set caused the poor functionality of the system. During the 2nd stage the neutral dataset was removed and achieved a higher rate of accuracy. Further training data was used to improve the accuracy in this stage. However expected results couldn't be achieved in the 2nd stage. It was observed that by increasing the number of training data it is possible to improve the accuracy of the classifier.

In the 3rd stage of the accuracy testing it was possible to achieve the expected level of accuracy more than 80%. This accuracy was stable enough to classify the sentiments of the historical tweets as well as the real time tweets. The real time tweet analysis was tested during the presidential election date. The test results and the observations have been included in the testing chapter. The classifier was accurate enough to detect the correct sentiment in most of the tweets. Historical sentiment analysis was also performed by the tweets that were collecting from **2016/10/11** to **2016/11/07**. It was also noticed the difference of the frequencies that mentioned the names 'Donald Trump' and 'Hillary Clinton' was a significant phenomenon.

However there were few drawbacks observed during the testing phase which weren't able to improve. The system was unable to detect the sarcastic tweets. In sarcastic tweets, the meaning of the tweet is different from what's written in it. Due to this phenomenon most of the negative sarcastic tweets were identified as positive sentiments and vice versa. This drawback was directly affected the final accuracy of the implemented system. Another major drawback of the system was the inability to detect the correct sentiments when both candidates were mentioned in a single tweet. The system was implemented to detect a single sentiment for each tweet. In some cases, when the names of the both candidates were mentioned, one section of the tweet carried a positive sentiment while the rest of the tweet carried a negative sentiment. Though it was not implemented, as a solution it was found out that more sophisticated natural language processing techniques should be used to detect these kind of anomalies.

Though the implemented system couldn't solve all the problems that were detected during the project, the system was able to achieve a satisfactory accuracy rate in the analysis. The system was able to provide a predictive analysis using historical data collected, while the real time sentiment analyser was used to detect the sentiments in a given time period.

CONTRIBUTION

This chapter elaborates the author's contribution to the similar type of projects and researches. Though the research area of this project is complicated, the author was able to discover and experiment with some interesting findings.

Implementation of No-SQL Database

Many of the previous researches that were conducted on sentiment analysis and data mining had used SQL databases as the primary database storage. During the technical research it was discovered that SQL databases affects with latency in high speed real time data retrieving. Since this project also deals with high speed real time data retrieving, there was a crucial need for a database type which doesn't affect with latency. As a solution for this problem, No-SQL database was used as the primary data storage. '*MongoDB*' one of the efficient document based No-SQL database was implemented to store the tweets as JSON objects. It was also found out that the data retrieving speed in the classification process was also higher than the use of traditional RDB. Along that it was discovered that the corruption rate of tweets were lower than using CSV and RDB.

Use of Python Object Serialization

Some researchers had claimed that during the real-time sentiment analysis, the classifier took considerable amount of time to classify the tweets. During the research it was found out that this phenomenon was caused by the training process of the classifier. Generally the classifier requires huge amount of time to train the system if the training data set is large. By using python object serialization the author was able to train the classifier once and stored the classifier. Due to this feature, there was no latency during the real time tweet classification.

Removal of Neutral Dataset from the Training

When using neutral training dataset with positive and negative training dataset, the system was able to achieve poor accuracies. Later in the implementation, neutral training dataset was removed from the training purposes and used only positive and negative training data. The classifier was able to provide stable accuracies during this stage. After observing the tweets during the election, it was discovered that most of the tweets contained sentiments which carried negative or positive. The published neutral tweets regarding the election were considerably less.

Implementation of the Word Cloud

Many researchers had used complicated algorithms such as Latent Dirichlet allocation (LDA) to discover the underlying topics in the tweets. The author was able to implement a simpler solution to discover underlying topics inside tweets than using statistical method. This solution was able to identify the frequency of words and illustrate the words inside a mask according to the calculated frequencies. Since this solution can be illustrated in a graphical manner, the users of this system can easily recognize the trending topics and keywords.

SUGGESTED FUTURE ENHANCEMENT

This chapter includes the future improvements which can be executed on the implemented system.

- The accuracy of the implemented system can be further improved by using a larger training dataset. Thus, the quality of the training dataset should be crucial. By using training tweets which were manually annotated would increase the system accuracy drastically.
- One of the major issues that faced during the testing stage was to detect the sentiments of the sarcastic tweets. After referring to similar projects it was decided that by including sarcastic tweets into the training dataset with the correct sentiment would help the system to some extent to detect such tweets in the future analysis.
- More sophisticated natural language processing techniques should be used to detect the sentiment when both candidates were mentioned in the same tweet. In such instances, one section of the tweet might carry a different polarity while the rest of the section carries the opposite polarity. To provide sentiment analysis with better accuracy, this issue should be improved.
- Since twitter users use informal language in the platform, it was observed that in most of the tweets there were spelling mistakes or grammar mistakes. These mistakes directly effects the accuracy of the classifier as the system determines every word that has a spelling mistake or grammar mistake as a new word. Machine learning algorithm could be implemented to determine such issues and correct them accordingly.
- To gather a large number of geo-tagged tweets it is necessary to collect data over a long period of time. Using the geo-tagged tweets it is possible to detect the polarity in each area (state).
- The implemented system can be improved into a web portal, where any user can login to the system and view the predictions or the analysis.
- To increase the processing speed and the classification speed of the system, the classifier can be hosted in a web service such as '*Amazon Web Services*'.
- The user interfaces should be further improved and user friendly.

CONCLUSION

The primary objective of this project was to implement a software solution to analyse the tweets and calculate the sentiment towards each Presidential candidate of the United States Presidential Election 2016. The solution is capable of analysing the tweets that were collected during a month prior to the election. Using those historical data, a predictive analysis can be calculated for each candidate. This system is also capable of conducting real time sentiment analysis, the application will generate a real time graph to demonstrate the trend results.

Thorough research has been conducted in order to implement this system. Several iterations on testing were conducted to evaluate the performance and the accuracy of the system.

Using the implemented solution it is possible to obtain a summarized analytics on the data that was analysed in the dashboard. In separate windows, users are able to observe the historical data analysis which were performed on Hillary Clinton and Donald Trump. Another two graphs are implemented to analyse the real time tweets on the final two candidates. To observe the underlying topics and keywords in tweets, users can move to word cloud window. As a feature, all these analytics and graphs can be saved separately to the computer.

All the deliverables mentioned in the documentation were implemented during this project. To improve this application further few improvements have been documented in the future enhancement chapter. The implemented solution provides stable and accurate results in both historical analysis and live analysis of tweets.

REFERENCES

- Bakliwal, A., Foster, J., van der Puil, J., O'Brien, R., Tounsi, L. & Hughes, M. (2013). *Sentiment analysis of political tweets: Towards an accurate classifier*.
- Bennett, S., McRobb, S. & Farmer, R. (2002). *Object-oriented systems analysis and design using UML*. McGraw-Hill.
- Bowles, M. (n.d.). *Machine learning in Python*.
- Cavnar, W.B., Trenkle, J.M. & others (1994). N-gram-based text categorization. *Ann Arbor MI*. 48113 (2). p.pp. 161–175.
- Collomb, A., Costea, C., Joyeux, D., Hasan, O. & Brunie, L. (n.d.). *A Study and Comparison of Sentiment Analysis Methods for Reputation Evaluation*.
- Cuesta, H. (2013). *Practical data analysis*. Packt Publishing.
- Gayo-Avello, D. (2012). ‘I Wanted to Predict Elections with Twitter and all I got was this Lousy Paper’--A Balanced Survey on Election Prediction using Twitter Data. *arXiv preprint arXiv:1204.6441*.
- Gimpel, K., Schneider, N., O'Connor, B., Das, D., Mills, D., Eisenstein, J., Heilman, M., Yogatama, D., Flanigan, J. & Smith, N.A. (2011). Part-of-speech tagging for twitter: Annotation, features, and experiments. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers- Volume 2*. 2011, pp. 42–47.
- Go, A., Bhayani, R. & Huang, L. (2009). Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*. 1. p.p. 12.
- Godbole, N., Srinivasaiah, M. & Skiena, S. (2007). Large-Scale Sentiment Analysis for News and Blogs. *ICWSM*. 7 (21). p.pp. 219–222.
- Gokulakrishnan, B., Priyanthan, P., Ragavan, T., Prasath, N. & Perera, As. (2012). Opinion mining and sentiment analysis on a twitter data stream. In: *Advances in ICT for Emerging Regions (ICTer), 2012 International Conference on*. 2012, pp. 182–188.
- Hagenau, M., Liebmann, M., Hedwig, M. & Neumann, D. (2012). Automated news reading: Stock price prediction based on financial news using context-specific features. In: *System Science (HICSS), 2012 45th Hawaii International Conference on*. 2012, pp. 1040–1049.
- Hoffer, J.A., George, J.F. & Valacich, J.S. (1999). *Modern systems analysis and design*. Addison-Wesley.

Hussein, D.M.E.-D.M. (2016). A survey on sentiment analysis challenges. *Journal of King Saud University-Engineering Sciences*.

Jahanbakhsh, K. & Moon, Y. (2014). The predictive power of social media: On the predictability of US Presidential Elections using Twitter. *arXiv preprint arXiv:1407.0622*.

Jurafsky DanMartin, J.H. (2000). *Speech and language processing*. Prentice Hall.

Khalid, S., Khalil, T. & Nasreen, S. (2014). A survey of feature selection and feature extraction techniques in machine learning. In: *Science and Information Conference (SAI), 2014*. 2014, pp. 372–378.

De Kok, D. & Brouwer, H. (2011). *Natural language processing for the working programmer*. Del.

Larsen, M.E., Boonstra, T.W., Batterham, P.J., O'Dea, B., Paris, C. & Christensen, H. (2015). We Feel: mapping emotion on Twitter. *IEEE journal of biomedical and health informatics*. 19 (4). p.pp. 1246–1252.

Mahmud, J., Nichols, J. & Drews, C. (2014). Home location identification of twitter users. *ACM Transactions on Intelligent Systems and Technology (TIST)*. 5 (3). p.p. 47.

Medhat, W., Hassan, A. & Korashy, H. (2014). Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal*. 5 (4). p.pp. 1093–1113.

Mount, J. (2011). The equivalence of logistic regression and maximum entropy models. URL: <http://www.win-vector.com/dfiles/LogisticRegressionMaxEnt.pdf>.

Neethu, M.S. & Rajasree, R. (2013). Sentiment analysis in twitter using machine learning techniques. In: *Computing, Communications and Networking Technologies (ICCCNT), 2013 Fourth International Conference on*. 2013, pp. 1–5.

Pak, A. & Paroubek, P. (2010). Twitter as a Corpus for Sentiment Analysis and Opinion Mining. *LREc*. 10. p.pp. 1320–1326.

Pang, B., Knight, K. & Marcu, D. (2003). Syntax-based alignment of multiple translations: Extracting paraphrases and generating new sentences. In: *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*. 2003, pp. 102–109.

Pang, B. & Lee, L. (2004). A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In: *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*. 2004, p. 271.

Pang, B. & Lee, L. (2008). Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*. 2 (1-2). p.pp. 1–135.

- Pang, B., Lee, L. & Vaithyanathan, S. (2002). Thumbs up?: sentiment classification using machine learning techniques. In: *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*. 2002, pp. 79–86.
- Parikh, R. & Movassate, M. (2009). Sentiment analysis of user-generated twitter updates using various classification techniques. *CS224N Final Report*. p.pp. 1–18.
- Richert WilliCoelho, L.P. (2013). *Building Machine Learning Systems with Python*. Packt Publishing.
- Rothfels, J. & Tibshirani, J. (2010). Unsupervised sentiment classification of English movie reviews using automatic selection of positive and negative sentiment items. *CS224N-Final Project*.
- Russell, M.A. (2011). *Mining the social web*. O'Reilly.
- Schouten, K. & Frasincar, F. (2016). Survey on aspect-level sentiment analysis. *IEEE Transactions on Knowledge and Data Engineering*. 28 (3). p.pp. 813–830.
- Thakkar, H. & Patel, D. (2015). Approaches for Sentiment Analysis on Twitter: A State-of-Art study. *arXiv preprint arXiv:1512.01043*.
- Tripathy, A., Agrawal, A. & Rath, S.K. (2015). Classification of Sentimental Reviews Using Machine Learning Techniques. *Procedia Computer Science*. 57. p.pp. 821–829.
- Tugores, A. & Colet, P. (2014). Mining online social networks with Python to study urban mobility. *arXiv preprint arXiv:1404.6966*.
- Tumasjan, A., Sprenger, T.O., Sandner, P.G. & Welpe, I.M. (2010). Predicting Elections with Twitter: What 140 Characters Reveal about Political Sentiment. *ICWSM*. 10. p.pp. 178–185.
- Turney, P.D. (2002). Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In: *Proceedings of the 40th annual meeting on association for computational linguistics*. 2002, pp. 417–424.
- Vohra, S.M. & Teraiya, J. (2013). A comparative study of sentiment analysis techniques. *Journal JIKRCE*. 2 (2). p.pp. 313–317.
- Wazlawick, R.S. (n.d.). *Object-oriented analysis and design for information systems*.
- Wikarsa, L. & Thahir, S.N. (2015). A text mining application of emotion classifications of Twitter's users using Na?? ve Bayes method. In: *2015 1st International Conference on Wireless and Telematics (ICWT)*. 2015, pp. 1–6.

Yessenalina, A., Yue, Y. & Cardie, C. (2010). Multi-level structured models for document-level sentiment classification. In: *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. 2010, pp. 1046–1056.

Yu, B., Kaufmann, S. & Diermeier, D. (2008). Exploring the characteristics of opinion expressions for political opinion classification. In: *Proceedings of the 2008 international conference on Digital government research*. 2008, pp. 82–91.

Yuan, Y. & Zhou, Y. (n.d.). *Twitter Sentiment Analysis with Recursive Neural Networks*.

Appendix – A

This appendix contains all the unit test cases used in the PyUnit testing stage.

```
from unittest import TestCase

class TestCleanTweet(TestCase):
    def test_cleanTweet(self):
        from py_tes import cleanTweet
        self.assertEqual(cleanTweet('@Trump this is for you'), 'user this
is for you')

    def test_cleanTweet_URL(self):
        from py_tes import cleanTweet
        self.assertEqual(cleanTweet('www.cnn.com we got the
results!'), 'url we got the results!')

    def test_cleanTweet_lower(self):
        from py_tes import cleanTweet
        self.assertEqual(cleanTweet('DONALD TRUMP'), 'donald trump')

    def test_cleanTweet_hash(self):
        from py_tes import cleanTweet
        self.assertEqual(cleanTweet('#ElectionNight'), 'electionnight')

    def test_cleanTweet_space(self):
        from py_tes import cleanTweet
        self.assertEqual(cleanTweet('Hillary Clinton'), 'hillary
clinton')

    def test_cleanTweet_spaces(self):
        from py_tes import cleanTweet
        self.assertEqual(cleanTweet(" Hillary Clinton"), 'hillary
clinton')

class TestReplaceTwoOrMore(TestCase):
    def test_replaceTwoOrMore(self):
        from py_tes import replaceTwoOrMore
        self.assertEqual(replaceTwoOrMore('Hillllary Clinton just won
Texas state!'), 'Hillary Clinton just won Texas state!')
```