**Project – I Report on**

**“UTILIZING MACHINE LEARNING FOR AN EFFICIENT ANALYSIS OF CUSTOMERS PERSPECTIVE FROM TWEETS”**

*Submitted to*

**GURU NANAK INSTITUTIONS TECHNICAL CAMPUS (AUTONOMUS)**

*In partial fulfillment of the requirement for the award of the degree of*

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**BY**

**EARATI PRIYANKA (20WJ1A0569)**

**ESPI SHAROON RAJ (20WJ1A0576)**

**JAKKA DURGA PRASAD REDDY (20WJ1A05A8)**

**Under the Esteemed Guidance Of**

**Mrs.** **K. SAILAJA**

**Assistant Professor**

****

**Department of Computer Science Engineering**

**GURU NANAK INSTITUTIONS TECHNICAL CAMPUS (AUTONOMOUS)**

**School of Engineering & Technology**

**Ibrahimpatnam, R.R District- 501506**

**2023 - 2024**



Date 11-12-2023

**Department of Computer Science Engineering**

**CERTIFICATE**

This is to certify that this mini project report entitled **“Utilizing Machine Learning for an Efficient Analysis of Customers Perspective from Tweets”** by **EARATI PRIYANKA (20WJ1A0569),** **ESPI SHAROON RAJ (20WJ1A0576), JAKKA DURGA PRASAD REDDY (20WJ1A05A8)** was submitted in partial fulfillment of the requirements for the degree of **Bachelor of Technology** in **Computer Science and Engineering** of the **Guru Nanak Institutions Technical Campus** during the academic year 2023-2024, is a bonafide record of work carried out under our guidance and supervision.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**INTERNAL GUIDE PROJECT CO-ORDINATOR HOD CSE**

**Mrs.** **K. Sailaja Mr. B. Samirana Acharya Dr. J. Rajeshwar**

**\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ \_**

**EXTERNAL EXAMINER**



Computer Science & Engineering

EARATI PRIYANKA 20WJ1A0569

ESPI SHAROON RAJ 20WJ1A0576

JAKKA DURGA PRASAD REDDY 20WJ1A05A8

MACHINE LEARNING

“UTILISING MACHINE LEARNING FOR EFFICIENT ANALYSIS OF CUSTOMERS PERSPECTIVR FROM TWEETS”

RAM INNOVATIVE INFOTECH

**ACKNOWLEDGEMENT**

We wish to express our sincere thanks to **Dr. Harvinder Singh Saini, Managing Director, GNITC** for providing us with the conducive environment for carrying through our academic schedules and Project with ease.

We wish to express our sincere thanks to **Dr. Koduganti Venkat Rao, Director, GNITC** for providing us with the conducive environment for carrying through our academic schedules and Project with ease.

We wish to express our sincere thanks to **Dr. Rishi Sayal, Associate Director, GNITC** for providing us with the conducive environment for carrying through our academic schedules and Project with ease.

We would like to say our sincere thanks to **Dr. J. Rajeshwar, Professor & Head, Department of CSE, GNITC** for providing seamless support and right suggestions are given in the development of the project.

We would like to say our sincere thanks to **Mr. B. Samirana Acharya, Assistant Professor, Department of CSE**, Project Coordinator, for providing seamless support and right suggestions are given in the development of the project.

We have been truly blessed to have a wonderful internal guide **Mrs.** **K. Shailaja, Assistant Professor of CSE**, **GNITC** for guiding us to explore the ramification of our work and we express our sincere gratitude towards him for leading me through the completion of Project.

Finally, we would like to thank our family members for their moral support and encouragement to achieve goals.

**EARATI PRIYANKA (20WJ1A0569)**

**ESPI SHAROON RAJ (20WJ1A0576)**

**JAKKA DURGA PRASAD REDDY (20WJ1A05A8)**

**UTILISING MACHINE LEARNING FOR EFFICIENT ANALYSIS CUSTOMERS PERSPECTIVE FROM TWEETS**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
|  | **LIST OF FIGURES**  **LIST OF SYMBOLS**  **LIST OF ABBREVIATIONS**  **ABSTRACT** | i  ii  vi  vii |
| 1. | **CHAPTER 1 INTRODUCTION**   * 1. GENERAL   2. OBJECTIVE   3. EXISTING SYSTEM   1.3.1 EXISTING SYSTEM DISADVANTAGES  1.3.2 LITERATURE SURVEY  1.4 PROPOSED SYSTEM  1.4.1 PROPOSED SYSTEM ADVANTAGES | 1  2  3  3  4  9  9 |
| 2. | **CHAPTER 2 PROJECT DESCRIPTION**  2.1 GENERAL  2.2 METHODOLOGIES  2.2.1 MODULES NAME  2.2.2 MODULES EXPLANATION | 10  10  10  10 |
| 3. | **CHAPTER 3 REQUIREMENTS**  3.1 General  3.2 Hardware REQUIREMENTS  3.3 Software REQUIREMENTS  3.4 FUNCTIONAL REQUIREMENTS  3.5 NON-FUNCTIONAL REQUIREMENTS | 13  13  13  14  14 |
| 4. | **CHAPTER 4 SYSTEM DESIGN**  4.1 general  4.1.1 USE CASE DIAGRAM  4.1.2 CLASS DIAGRAM  4.1.3 OBJECT DIAGRAM  4.1.4 COMPONENT DIAGRAM  4.1.5 DEPLOYMENT DIAGRAM  4.1.6 SEQUENCE DIAGRAM  4.1.7 COLLABORATION DIAGRAM  4.1.8 STATE DIAGRAM  4.1.9 activity diagram  4.1.10 MODULE DIAGRAM  4.1.11 E-R DIAGRAM  4.1.12 SYSTEM ARCHITECTURE | 15  16  17  18  19  20  21  22  23  24  25  27  28 |
| 5. | **CHAPTER 5 SOFTWARE SPECIFICATION**  5.1 general | 29 |
| 6. | **CHAPTER 6 IMPLEMENTATION**  6.1 general | 32 |
| 7. | **CHAPTER 7 SNAPSHOTS**  7.1 GENERAL  7.2 VARIOUS SNAPSHOTS | 36  36 |
| 8. | **CHAPTER 8 SOFTWARE TESTING**  8.1 GENERAL  8.2 DEVELOPING METHODOLOGIES  8.3 TYPES OF TESTING | 38  38  38 |
| 9. | **CHAPTER 9 APPLICATIONS AND FUTURE ENHANCEMENT**  9.1 GENERAL  9.2 APPLICATIONS  9.3 FUTURE ENHANCEMENTS | 41  41  41 |
| 10 | **CHAPTER 10**  10.1 CONCLUSION  10.2 REFERENCES | 43  44 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **NAME OF THE FIGURE** | **PAGE NO.** |
| 4.1.1 | USE CASE DIAGRAM | 16 |
| 4.1.2 | CLASS DIAGRAM | 17 |
| 4.1.3 | OBJECT DIAGRAM | 18 |
| 4.1.4 | COMPONENT DIAGRAM | 19 |
| 4.1.5 | DEPLOYMENT DIAGRAM | 20 |
| 4.1.6 | SEQUENCE DIAGRAM | 21 |
| 4.1.7 | COLLABORATION DIAGRAM | 22 |
| 4.1.8 | STATE DIAGRAM | 23 |
| 4.1.9 | activity diagram | 24 |
| 4.1.10 | MODULE DIAGRAM | 25 |
| 4.1.11 | E-R DIAGRAM | 27 |

**LIST OF SYSMBOLS**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **NOTATION**  **NAME** | **NOTATION** | **DESCRIPTION** |
| 1. | Class | *Class Name*  *-attribute*  *-attribute*  *+operation*  *+operation*  *+operation*  *+ public*  *-private*  *# protected* | Represents a collection of similar entities grouped together. |
| 2. | Association | name  Class B  Class A  Class A  Class B | Associations represents static relationships between classes. Roles represents the way the two classes see each other. |
| 3. | Actor | Class A  Class A  Class B  Class B | It aggregates several classes into a single classes. |
| 4. | Aggregation | Interaction between the system and external environment |

|  |  |  |  |
| --- | --- | --- | --- |
| 5. | Relation  (uses) | uses | Used for additional process communication. |
| 6. | Relation  (extends) | extends | Extends relationship is used when one use case is similar to another use case but does a bit more. |
| 7. | Communication |  | Communication between various use cases. |
| 8. | State | State | State of the processs. |
| 9. | Initial State |  | Initial state of the object |
| 10. | Final state |  | F inal state of the object |
| 11. | Control flow |  | Represents various control flow between the states. |
| 12. | Decision box |  | Represents decision making process from a constraint |
| 13. | Usecase |  | Interact ion between the system and external environment. |
| 14. | Component |  | Represents physical modules which are a collection of components. |

|  |  |  |  |
| --- | --- | --- | --- |
| 14. | Component |  | Represents physical modules which are a collection of components. |
| 15. | Node |  | Represents physical modules which are a collection of components. |
| 16. | Data Process/State |  | A circle in DFD represents a state or process which has been triggered due to some event or acion. |
| 17. | External entity |  | Represents external entities such as keyboard,sensors,etc  . |
| 18. | Transition |  | Represents communication that occurs between processes. |
| 19. | Object Lifeline |  | Represents the vertical dimensions that the object communications. |
| 20. | Message | Message | Represents the message exchanged. |

**LIST OF ABBREVATION**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **ABBREVATION** | **EXPANSION** |
| 1**.** | DB | DataBase |
| 2. | JVM | Java Virtual Machine |
| 3. | JSP | Java Server Page |
| 4. | PWS | Personalised Web Search |
| 5. | UPS | User Personalised Search |
| 6. | JRE | Java Runtime Environment |

**ABSTRACT**

Social media have received more attention nowadays. Public and private opinion about a wide variety of subjects is expressed and spread continually via numerous social media.Twitter is one of the social media that is gaining popularity. Twitter offers organizations a fast and effective way to analyze customers’ perspectives toward the critical to success in the market place. Developing a program for sentiment analysis is an approach to be used to computationally measure customers’ perceptions.This project reports on the design of a sentiment analysis, extracting a vast number of tweets. Prototyping is used in this development. Results classify customers’ perspective via tweets into positive and negative, which is represented in a pie chart and html page. However, the program has planned to develop on a web application system, but due to limitation of Django which can be worked on a Linux server or LAMP, for further this approach need to be done.

**CHAPTER 1**

**INTRODUCTION**

* 1. **GENERAL**

Sentiment analysis [SA] is the area which discuss about people’s opinion towards product, service, tourism, movies, political issues, education system etc. Sentiment analysis plays vital role in the internet era due to extensive range of business applications and social media. Inspiration behind sentiment analysis is that it provides people ‘s opinion about the product, which helps to improve the product quality. It also supports to take purchase/manufacturing decisions. In this project we apply sentiment analysis to catch people feedback about Cricket using social media messages. For this we use tweeter data set to analyze tourist opinion about this game. In this project , we recommend innovative sentiment analysis method based on common sense knowledge (Domain Specific Ontology). We created our own Cricket ontology based on Concept Net. Entities are identified from the tweets using POS tagger and entities are compared with concepts in the domain specific ontology. Further the sentiment of the extracted entities is determined by the combined sentiment lexicon approach. Finally semantic orientations of domain specific features are combined with respect to the domain. We deliberate conceptual semantic as feature which can be combined with machine learning algorithm to enhance the performance of sentiment analysis of Cricket .

Text mining refers to extracting the useful information from the natural language text. Generally, most of the people express their opinion through social media like Twitter, Facebook etc. Sentiment analysis or opinion mining is the study of extracting user’s opinion from the given text. With rapid development of internet, people are using social media sites, blogs, channels, forums and review sites to express their opinion about any topic in the current trend.

**1.2 OBJECTIVE**

In this project we apply sentiment analysis to catch people feedback about Cricket using social media messages. For this we use tweeter data set to analyze tourist opinion about this game. In this project , we recommend innovative sentiment analysis method based on common sense knowledge (Domain Specific Ontology).

**1.3 Existing System**

Sentiment analysis is the role of identifying opinions, emotions, and evaluations in each text. The important concept in sentiment analysis is to identify the polarity of overall text. It helps to indicate whether the text is positive, negative, or neutral.

**1.3.1 Existing System Disadvantages**

The existing system may have certain limitations and drawbacks that pave the way for the proposed system. Some potential disadvantages of the existing system might include

**Limited Accuracy**  The sentiment analysis in the existing system may not achieve high accuracy due to challenges in handling nuanced language, sarcasm, or context-specific expressions.

**Scalability Issues**  As social media data grows, the existing system may face challenges in efficiently processing and analyzing a large volume of tweets, leading to scalability issues.

**Dependency on Pre-built Lexicons**  If the existing system heavily relies on pre-built sentiment lexicons, it may struggle to adapt to domain-specific nuances and emerging trends, resulting in less accurate sentiment classification.

**Inability to Handle Multimodal Data**  The existing system might focus solely on text data and may not effectively incorporate or analyze additional multimedia elements like images, videos, or emojis accompanying the tweets.

**1.3.2 LITERATURE SURVEY**

**Title** Sentiment Analysis and Opinion Mining

**Author**  Bing Liu

**Year** 2010

**Description**

Sentiment analysis and opinion mining is the field of study that analyzes people's opinions, sentiments, evaluations, attitudes, and emotions from written language. It is one of the most active research areas in natural language processing and is also widely studied in data mining, Web mining, and text mining. In fact, this research has spread outside of computer science to the management sciences and social sciences due to its importance to business and society. The growing importance of sentiment analysis coincides with the growth of social media such as reviews, forum discussions, blogs, micro-blogs, Twitter, and social networks. For the first time in human history, we now have a huge volume of opinionated data recorded in digital form for analysis.

Sentiment analysis systems are being applied in almost every business and social domain because opinions are central to almost all human activities and are key influencers of our behaviors. Our beliefs and perceptions of reality, and the choices we make, are largely conditioned on how others see and evaluate the world. For this reason, when we need to decide, we often seek out the opinions of others. This is true not only for individuals but also for organizations.

**Title** SENTIWORDNET A Publicly Available Lexical Resource for Opinion Mining

**Author**  Andrea Esuli , Fabrizio Sebastiani

**Year** 2006

**Description**

Opinion mining (OM) is a recent subdiscipline at the crossroads of information retrieval and computational linguistics which is concerned not with the topic a document is about, but with the opinion it expresses. OM has a rich set of applications, ranging from tracking users' opinions about products or about political candidates as expressed in online forums, to customer relationship management. In order to aid the extraction of opinions from text, recent research has tried to automatically determine the "PN-polarity" of subjective terms, i.e. identify whether a term that is a marker of opinionated content has a positive or a negative connotation. Research on determining whether a term is indeed a marker of opinionated content (a subjective term) or not (an objective term) has been, instead, much scarcer. In this work we describe SENTIWORDNET, a lexical resource in which each WORDNET synset s is associated to three numerical scores Obj(s), Pos(s) and Neg(s), describing how objective, positive, and negative the terms contained in the synset are. The method used to develop SENTIWORDNET is based on the quantitative analysis of the glosses associated to synsets, and on the use of the resulting vectorial term representations for semi-supervised synset classification. The three scores are derived by combining the results produced by a committee of eight ternary classifiers, all characterized by similar accuracy levels but different classification behaviour. SENTIWORDNET is freely available for research purposes, and is endowed with a Web-based graphical user interface.

**Title** Mining and Summarizing Customer Reviews .

**Author**  Minqing Hu , Bing Liu

**Year** 2004

**Description**

Merchants selling products on the Web often ask their customers to review the products that they have purchased and the associated services. As e-commerce is becoming more and more popular, the number of customer reviews that a product receives grows rapidly. For a popular product, the number of reviews can be in hundreds or even thousands. This makes it difficult for a potential customer to read them to make an informed decision on whether to purchase the product. It also makes it difficult for the manufacturer of the product to keep track and to manage customer opinions. For the manufacturer, there are additional difficulties because many merchant sites may sell the same product and the manufacturer normally produces many kinds of products. In this research, we aim to mine and to summarize all the customer reviews of a product. This summarization task is different from traditional text summarization because we only mine the features of the product on which the customers have expressed their opinions and whether the opinions are positive or negative. We do not summarize the reviews by selecting a subset or rewrite some of the original sentences from the reviews to capture the main points as in the classic text summarization. Our task is performed in three steps (1) mining product features that have been commented on by customers; (2) identifying opinion sentences in each review and deciding whether each opinion sentence is positive or negative; (3) summarizing the results. This project proposes several novel techniques to perform these tasks. Our experimental results using reviews of several products sold online demonstrate the effectiveness of the techniques.

**Title** Opinion observer analyzing and comparing opinions on the Web

**Author**  Bing Liu, Minqing Hu, Junsheng Cheng

**Year** 2005

**Description**

The Web has become an excellent source for gathering consumer opinions. There are now numerous Web sites containing such opinions, e.g., customer reviews of products, forums, discussion groups, and blogs. This project focuses on online customer reviews of products. It makes two contributions. First, it proposes a novel framework for analyzing and comparing consumer opinions of competing products. A prototype system called Opinion Observer is also implemented. The system is such that with a single glance of its visualization, the user is able to clearly see the strengths and weaknesses of each product in the minds of consumers in terms of various product features. This comparison is useful to both potential customers and product manufacturers. For a potential customer, he/she can see a visual side-by-side and feature-by-feature comparison of consumer opinions on these products, which helps him/her to decide which product to buy. For a product manufacturer, the comparison enables it to easily gather marketing intelligence and product benchmarking information. Second, a new technique based on language pattern mining is proposed to extract product features from Pros and Cons in a particular type of reviews. Such features form the basis for the above comparison. Experimental results show that the technique is highly effective and outperform existing methods significantly.

**Title** An Introduction to concept-level sentiment analysis

**Author**  Erik Cambria

**Year** 2013

**Description**

The ways people express their opinions and sentiments have radically changed in the past few years thanks to the advent of social networks, web communities, blogs, wikis, and other online collaborative media. The distillation of knowledge from the huge amount of unstructured information on the Web can be a key factor for marketers who want to create an image or identity in the minds of their customers for their product or brand. These online social data, however, remain hardly accessible to computers, as they are specifically meant for human consumption. The automatic analysis of online opinions, in fact, involves a deep understanding of natural language text by machines, from which we are still very far. To this end, concept-level sentiment analysis aims to go beyond a mere word-level analysis of text and provide novel approaches to opinion mining and sentiment analysis that enable a more efficient passage from (unstructured) textual information to (structured) machine-processable data, in potentially any domain.

**1.4 Proposed System**

We propose novel method based on domain specific ontology. We constructed our own Cricket ontology. Generally, nouns are considered as entities, so nouns are extracted from the tweets using POS tagger which could be compared with concepts in ontology. Further entities/opinion extraction to be done by inquiring object-attribute pair in ontology. In addition to that we construct our own sentiment lexicon based on three existing lexicons such as SentiStrength, SentiWordNet and Opinion lexicon.

**1.4.1 Proposed System Advantages**

The proposed system offers several advantages over the existing system

**Improved Accuracy**  Incorporating domain-specific ontology and entity-specific opinion extraction enhances the accuracy of sentiment analysis by capturing nuances and context.

**Enhanced Performance** Conceptual semantic sentiment analysis, combined with machine learning algorithms, contributes to better performance in understanding and classifying sentiments in tweets.

**Adaptability to Emerging Trends**  The proposed system is designed to adapt to emerging trends and changing language patterns, making it more robust in handling evolving user expressions.

**Multimodal Analysis** Future enhancements may include the ability to analyze not only text but also images, videos, and other multimedia elements in tweets, providing a more comprehensive understanding of customer sentiments.

**CHAPTER 2**

**PROJECT DESCRIPTION**

**2.1 GENERAL**

In this project we concern individual’s opinion about Cricket since Cricket is the beautiful game, all are interested to visit Cricket . Since twitter is the large carpus of data, twitter message has taken for this research to know the people sentiment.

**2.2 METHODOLOGIES131**

**MODULES**

1. Data Gathering

2. Pre-Processing

3. Polarity Calculation

4. Classification

**DATA GATHERING**

To gather public opinion based on collected hash tags related to views about political parties including Twitter top trends, we used Tweepy API [27,57]. We have created an account on Tweepy API linked to our Twitter account. To retrieve the tweets, Tweepy API accepts parameters and provides the Twitter account’s data in return. Retrieved tweets, from Twitter accounts, were saved in the database under the following fields twitter\_id, hashtag, tweet\_created, user\_id, screen\_name, tweet\_text, retweet\_count, follower\_count, and favourite\_count of each tweet. The gathered tweets numbered 100,000.

**PRE-PROCESSING**

Twitter is a micro-blog where people generally write in a conversational style. Tweets are known to be very noisy for any text mining task as they contain a number of symbols that do not have any useful information and make further processing ineffective. Therefore, this model includes effective pre-processing phase which removes meaningless symbols from tweets and hence, effective keywords can be extracted. The steps for pre-processing are as follows

**i. Remove username and re-tweet symbol**  Tweets often contain usernames beginning with the symbol ‘@’. Sometimes a tweet is also re-tweeted, which means a tweet by any user is shared again by other users and it contains the symbol RT. These user- names and re-tweet symbol do not contribute any significance to keyword extraction and act as noise. So, usernames and re-tweet symbols are removed.

**ii. Remove URLs**  Any URL links appearing in the tweets are re- moved as the model focuses only on the textual part of the tweet and URLs act as unnecessary noise while keywords are extracted.

**iii. Remove hash tags**  The Hash tag i.e. # before a word such as #KarnatakaWithCongress is removed to get ‘Karnataka With- Congress ’.

**iv. Tokenization**  Each term in a tweet is treated as a token. To- kens are the basic constituents of a tweet/text. Let T be the set of tweets which is represented as T = { T 1 , T 2 , T 3 ,…, T i | i is the number of tweets}. Then each tweet in T is pre-processed and its terms are treated as tokens. Let t be the set of tokens rep- resented as t = { t 1 , t 2 , t 3 ,…, t k }. t includes tokens from all the tweets of T where the number of tokens in the set T is k .

**v. Stop word removal**  A standard list of stop words is created and these stop words are then removed from the set.

**POLARITY CALCULATION**

Sentiment analysis can provide valuable insights from social media platforms by detecting emotions or opinions from a large volume of data present in unstructured format. Sentiment analysis includes three polarity classes, which are negative, neutral, and positive. The polarity of each tweet is determined by assigning a score from −1 to 1 based on the words used, where a negative score means a negative sentiment and a positive score means a positive sentiment while the zero value is considered a neutral sentiment. A score of subjectivity assigned to each tweet is based on whether it is representing a subjective meaning or an objective meaning; the range of subjectivity score is also from 0 to 1 where a value near to 0 represents objective and near to 1 subjective.

**CLASSIFICATION**

Based on the polarity result we can classify the tweets into negative, positive, and neutral reviews and calculate the overall percentages. After getting the result we display the graphs based on the reviews.

**CHAPTER 3**

**REQUIREMENTS ENGINEERING**

**3.1 GENERAL**

Finally, we introduce semantic sentiment analysis concept which concern about the semantic meaning of the word. It would be classified in to two types contextual semantic and conceptual semantic. Contextual semantic deals with considering about neighboring word. Conceptual semantic depends on outside knowledge such as ontologies and semantic network. We consider conceptual semantic as feature which can be incorporate with Naïve Bayes machine learning method to increase the performance of sentiment analysis.

**3.2 HARDWARE REQUIREMENTS**

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It shouls what the system do and not how it should be implemented.

* PROCESSOR DUAL CORE 2 DUOS.
* RAM 4GB DD RAM
* HARD DISK 250 GB

**3.3 SOFTWARE REQUIREMENTS**

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team’s progress throughout the development activity.

**SOFTWARE REQUIREMENTS**

* Operating System Windows 7/8/10
* Platform Spyder3
* Programming Language Python, HTML
* Front End Spyder3

**3.4 FUNCTIONAL REQUIREMENTS**

A functional requirement defines a function of a software-system or its component. A function is described as a set of inputs, the behavior, Firstly, the system is the first that achieves the standard notion of semantic security for data confidentiality in attribute-based deduplication systems by resorting to the hybrid cloud architecture.

**3.5 NON-FUNCTIONAL REQUIREMENTS**

**EFFICIENCY**

Our multi-modal event tracking and evolution framework is suitable for multimedia documents from various social media platforms, which can not only effectively capture their multi-modal topics, but also obtain the evolutionary trends of social events and generate effective event summary details over time. Our proposed mmETM model can exploit the multi-modal property of social event, which can effectively model social media documents including long text with related images and learn the correlations between textual and visual modalities to separate the visual-representative topics and non-visual-representative topics.

**CHAPTER 4**

**DESIGN ENGINEERING**

**4.1 GENERAL**

Design Engineering deals with the various UML [Unified Modeling language] diagrams for the implementation of project. Design is a meaningful engineering representation of a thing that is to be built. Software design is a process through which the requirements are translated into representation of the software. Design is the place where quality is rendered in software engineering. Design is the means to accurately translate customer requirements into finished product.

**UML Diagrams**

**Use case diagram**

****

4.1.1 USE CASE DIAGRAM

**EXPLANATION**

The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted. The above diagram consists of user as actor. Each will play a certain role to achieve the concept.

**Class Diagram**



4.1.2 CLASS DIAGRAM

**EXPLANATION**

In this class diagram represents how the classes with attributes and methods are linked together to perform the verification with security. From the above diagram shown the various classes involved in our project.

**Object Diagram**

****

4.1.3 OBJECT DIAGRAM

**EXPLANATION**

In the above digram tells about the flow of objects between the classes. It is a diagram that shows a complete or partial view of the structure of a modeled system. In this object diagram represents how the classes with attributes and methods are linked together to perform the verification with security.

**Component Diagram**



4.1.4 COMPONENT DIAGRAM

**Deployment Diagram**



4.1.5 DEPLOYMENT DIAGRAM

**Sequence Diagram**



4.1.6 SEQUENCE DIAGRAM

**EXPLANATION**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

**Collaboration Diagram**



4.1.7 COLLABRATION DIAGRAM

**State Diagram**



4.1.8 STATE DIAGRAM

**EXPLANATION**

State diagram are a loosely defined diagram to show workflows of stepwise activities and actions, with support for choice, iteration and concurrency. State diagrams require that the system described is composed of a finite number of states; sometimes, this is indeed the case, while at other times this is a reasonable abstraction. Many forms of state diagrams exist, which differ slightly and have different semantics.

**Activity Diagram**

**4.1.9** ACTIVITY DIAGRAM

**EXPLANATION**

Activity diagram are a loosely defined diagram to show workflows of stepwise activities and actions, with support for choice, iteration and concurrency. UML, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. UML activity diagrams could potentially model the internal logic of a complex operation. In many ways UML activity diagrams are the object-oriented equivalent of flow charts and data flow diagrams (DFDs) from structural development.

**4.1.10 MODULE DIAGRAMS**

Level 0

Load Data

Preprocessing

Prediction

Data filtering

Calculate accuracy

Level 1

Training Data, Test Data

Twitter Data

Preprocessing

Prediction

Cleaning, Reconstructing

Features, Unsupervised, Supervised

Filtering

Extraction

**EXPLANATION**

A data flow diagram (DFD) is a graphical representation of the “flow” of data through an information system, modeling its process aspects. Often they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

A DFD shows what kinds of data will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel.

**E-R Diagram**

Twitter

Preprocessing

Filtering

Forecasting

Extraction

User

Supervised

4.1.11 ER DIAGRAM

**EXPLANATION**

Entity-Relationship Model (ERM) is an abstract and conceptual representation of data. Entity-relationship modeling is a database modeling method, used to produce a type of conceptual schema or semantic data model of a system, often a relational database.

**System Architecture**

LOAD TWEETS FROM TWITTER

INDIAN TOURISM

NEGATIVE TWEETS

POSITIVE TWEETS

SENTIMENT ANALYSIS

NEUTRAL TWEELS

4.1.12 SYSTEM ARCHITECTURE

**CHAPTER 5**

**5.1 GENERAL**

**PYTHON**

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

## HISTORY OF PYTHON

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

#### IMPORTANCE OF PYTHON

* **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
* **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
* **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

#### FEATURES OF PYTHON

* **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
* **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
* **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
* **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
* **Databases** − Python provides interfaces to all major commercial databases.
* **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
* **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

* It supports functional and structured programming methods as well as OOP.
* It can be used as a scripting language or can be compiled to byte-code for building large applications.
* It provides very high-level dynamic data types and supports dynamic type checking.
* IT supports automatic garbage collection.
* It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

**LIBRARIES USED IN PYTHON**

* numpy - mainly useful for its N-dimensional array objects.
* pandas - Python data analysis library, including structures such as dataframes.
* matplotlib - 2D plotting library producing publication quality figures.
* scikit-learn - the machine learning algorithms used for data analysis and data mining tasks.



Figure NumPy, Pandas, Matplotlib, Scikit-learn

**CHAPTER 6**

**IMPLEMENTATION**

**6.1 GENERAL**

**Coding**

# -\*- coding utf-8 -\*-

import re

import tweepy

# import tweepy.errors.TweepError

from tweepy import OAuthHandler

from textblob import TextBlob

import numpy as np

import json

class TwitterClient(object)

    '''

    Generic Twitter Class for sentiment analysis.

    '''

    def \_\_init\_\_(self)

        '''

        Class constructor or initialization method.

        '''

        # keys and tokens from the Twitter Dev Console

        consumer\_key = 'IaIdBOJa7ZwNI6xHS11Jg5DVb'

        consumer\_secret = 'IuZ4G5wrK2aYU1yuOwbPqu6G0Rx1hBgHwo6xEOHANhA4gTzJv9'

        access\_token = '2158465844-TYtigqXGBQa5KzshKjz5MFO9SqTVkc03FQLL37N'

        access\_token\_secret = 'ClgQD5kCml8yb70ZlseUQIXOAM6bltOwoPuD5Z3SWguXW'

        # attempt authentication

        try

            # create OAuthHandler object

            self.auth = OAuthHandler(consumer\_key, consumer\_secret)

            # set access token and secret

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

            # create tweepy API object to fetch tweets

            self.api = tweepy.API(self.auth)

        except

            print("Error Authentication Failed")

    def clean\_tweet(self, tweet)

        '''

        Utility function to clean tweet text by removing links, special characters

        using simple regex statements.

        '''

        #import PreProcessing

        #PreProcessing.preProcess(tweet)

        return ' '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+ \/\/\S+)", " ", tweet).split())

    def get\_tweet\_sentiment(self, tweet)

        '''

        Utility function to classify sentiment of passed tweet

        using textblob's sentiment method

        '''

        # create TextBlob object of passed tweet text

        analysis = TextBlob(self.clean\_tweet(tweet))

        # set sentiment

        if analysis.sentiment.polarity > 0

            return 'positive'

        elif analysis.sentiment.polarity == 0

            return 'neutral'

        else

            return 'negative'

    def get\_tweets(self)

        '''

        Main function to fetch tweets and parse them.

        '''

        # empty list to store parsed tweets

        tweets = []

        try

            # call twitter api to fetch tweets

            #fetched\_tweets = self.api.search(q = query, count = count)

            tweets\_file = open('twitter\_data.txt', "r")

            # parsing tweets one by one

            for tweet in tweets\_file

                # empty dictionary to store required params of a tweet

                try

                    parsed\_tweet = {}

                    tweett = json.loads(tweet)

                    tweet =tweett.get("text")

                    parsed\_tweet['text']=tweet

                    # saving text of tweet

                    #parsed\_tweet['text'] = tweet.text

                    # saving sentiment of tweet

                    parsed\_tweet['sentiment'] = self.get\_tweet\_sentiment(tweet)

                    # appending parsed tweet to tweets list

                    if tweett.get("retweet\_count")> 0

                        # if tweet has retweets, ensure that it is appended only once

                        if parsed\_tweet not in tweets

                            tweets.append(parsed\_tweet)

                    else

                        tweets.append(parsed\_tweet)

                except

                    continue

            # return parsed tweets

            return tweets

        except tweepy.errors.TweepyException as e

            # print error (if any)

            print("Error " + str(e))

def main()

    # creating object of TwitterClient Class

    api = TwitterClient()

    # calling function to get tweets

    tweets = api.get\_tweets()

    # picking positive tweets from tweets

    ptweets = [tweet for tweet in tweets if tweet['sentiment'] == 'positive']

    # percentage of positive tweets

    print("Positive tweets per  centage {} %".format(100\*len(ptweets)/len(tweets)))

    # picking negative tweets from tweets

    ntweets = [tweet for tweet in tweets if tweet['sentiment'] == 'negative']

    # percentage of negative tweets

    print("Negative tweets percentage {} %".format(100\*len(ntweets)/len(tweets)))

    # percentage of neutral tweets

    nutweets= len(tweets)-(len(ntweets) + len(ptweets))

    print("Neutral tweets percentage {} % ".format(100\*nutweets/len(tweets)))

    graph={}

    graph.update({"Positive tweets" 100\*len(ptweets)/len(tweets)})

    graph.update({"Negative tweets" 100\*len(ntweets)/len(tweets)})

    graph.update({"Neutral tweets" 100\*nutweets/len(tweets)})

    import matplotlib.pyplot as pyplot

    import collections

    counts = collections.Counter(graph)

    pyplot.pie([float(v) for v in counts.values()], labels=[k  for k in counts],

           autopct=None)

    pyplot.show()

    pyplot.bar(range(len(counts)), list(counts.values()), align='center')

    pyplot.xticks(range(len(counts)), list(counts.keys()))

    pyplot.show()

    # printing first 5 positive tweets

    print("\n\nPositive tweets ")

    for tweet in ptweets

        print(tweet['text'])

    # printing first 5 negative tweets

    print("\n\nNegative tweets ")

    for tweet in ntweets

        print(tweet['text'])

if \_\_name\_\_ == "\_\_main\_\_"

    # calling main function

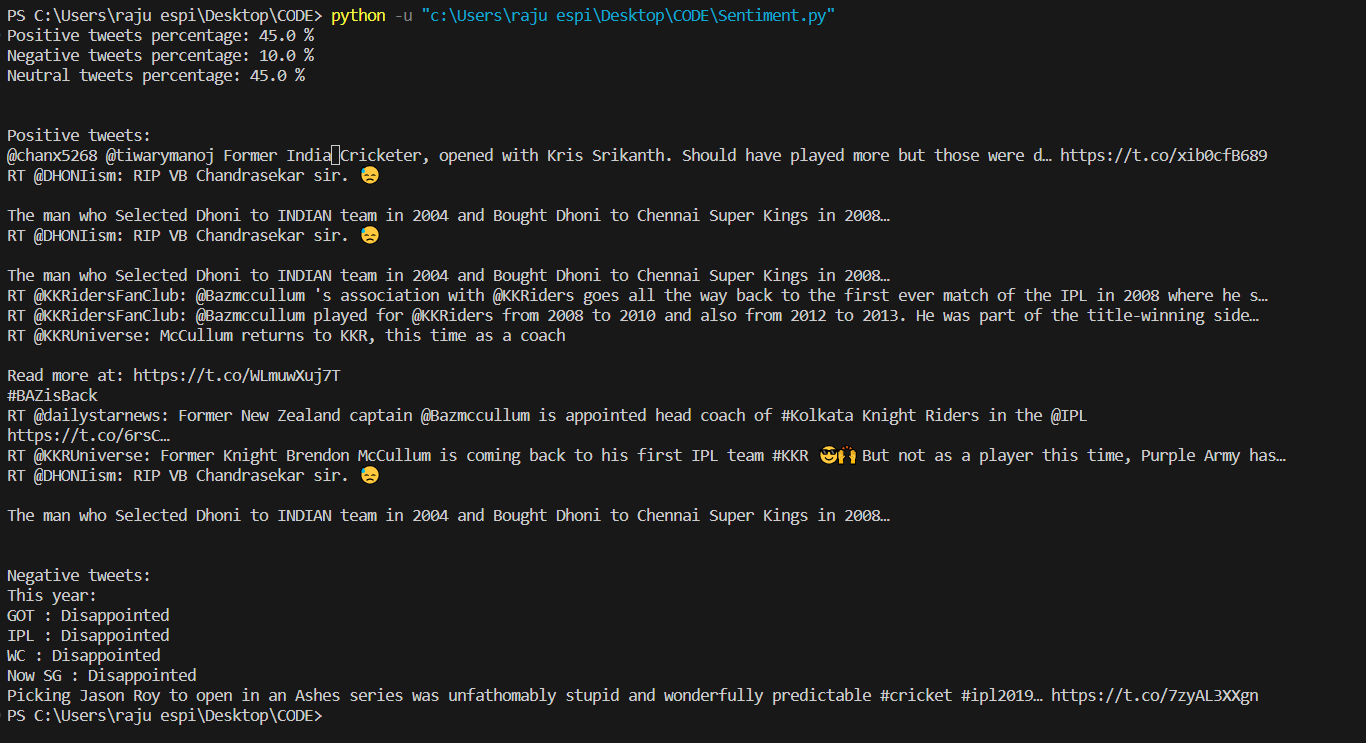
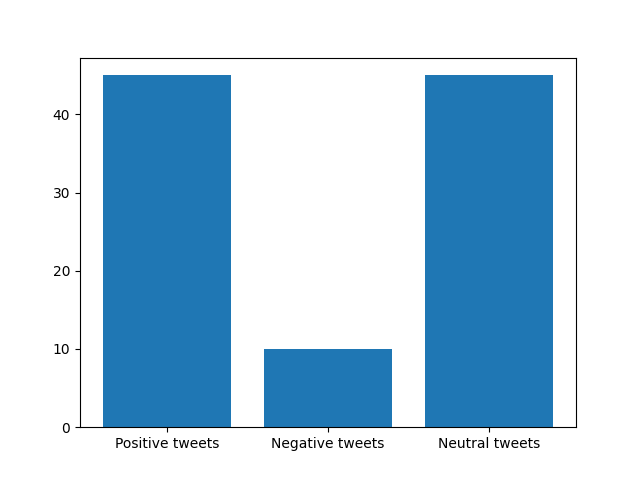
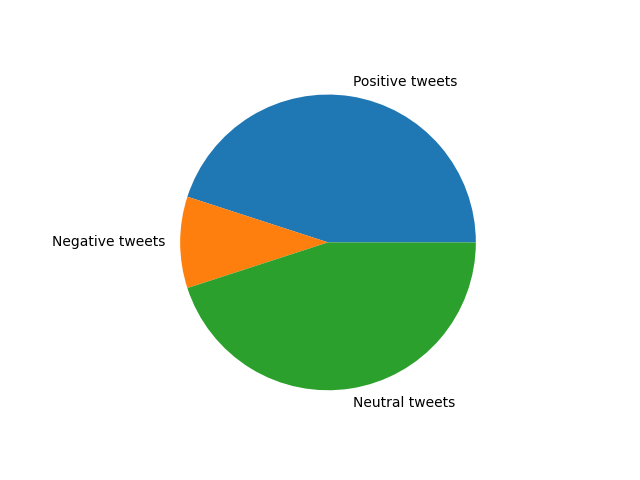
    main()

**CHAPTER 7**

**SNAPSHOTS**

**7.1 GENERAL**

This project is implements like application using python and the Server process is maintained using the SOCKET & SERVERSOCKET and the Design part is played by Cascading Style Sheet.

**7.2 SNAPSHOTS **

**CHAPTER 8**

**SOFTWARE TESTING**

**8.1 GENERAL**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**8.2 DEVELOPING METHODOLOGIES**

The test process is initiated by developing a comprehensive plan to test the general functionality and special features on a variety of platform combinations. Strict quality control procedures are used.The process verifies that the application meets the requirements specified in the system requirements document and is bug free. The following are the considerations used to develop the framework from developing the testing methodologies.

**8.3 TYPES OF TESTS**

**8.3.1 Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**8.3.2 Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items

Valid Input identified classes of valid input must be accepted.

Invalid Input identified classes of invalid input must be rejected.

Functions identified functions must be exercised.

Output identified classes of application outputs must be exercised.

Systems/Procedures interfacing systems or procedures must be invoked.

**8.3.3 System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**8.3.4 Performance Test**

The Performance test ensures that the output be produced within the time limits,and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

**8.3.5 Integration Testing**

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**8.3.6 Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Acceptance testing for Data Synchronization**

* The Acknowledgements will be received by the Sender Node after the Packets are received by the Destination Node
* The Route add operation is done only when there is a Route request in need
* The Status of Nodes information is done automatically in the Cache Updation process

**8.2.7 Build the test plan**

Any project can be divided into units that can be further performed for detailed processing. Then a testing strategy for each of this unit is carried out. Unit testing helps to identity the possible bugs in the individual component, so the component that has bugs can be identified and can be rectified from errors.

**CHAPTER 9**

**FUTURE ENHANCEMENT**

**9.1 GENERAL**

An adaptive ensemble member weighting process is designed to emphasize the importance of different ensemble members, and avoid the effect of deleterious ensemble members.

The future internet comes with high requirements of information dissemination, which motivate the research community to find alternative solutions.

**9.2 APPLICATIONS**

The sentiment analysis project outlined in the provided text has diverse applications across various domains. Some notable applications include

**Market Research**

Companies can utilize sentiment analysis to gauge public opinions about their products and services. This information is valuable for market research and helps in understanding consumer preferences.

**Brand Monitoring**

Organizations can monitor social media sentiments related to their brand. Positive sentiments can be reinforced, and negative sentiments can prompt proactive reputation management strategies.

**Customer Support**

Companies can integrate sentiment analysis into their customer support systems to analyze real-time feedback on social media. This allows for quick identification and resolution of customer issues.

**Tourism Industry**

Sentiment analysis can help assess public opinions about tourist destinations. Tourism boards can use this information for destination marketing and improvement strategies

**Movie and Entertainment Industry**

Sentiment analysis can be applied to analyze audience reactions to movies, TV shows, and entertainment events. Studios can use this data for

**CHAPTER 10**

**CONCLUSION & REFERENCE**

**10.1 CONCLUSION**

Factors that is domain specific ontology, entity specific opinion extraction, combined lexicon-based approach and conceptual semantic sentiment analysis to determine the sentiment analysis of tweets about Cricket . The new approach including conceptual semantic sentiment analysis expressively improve the performance of the sentiment analysis. Future work involves how to enrich the sentiment analysis using contextual and conceptual semantic sentiment analysis together. From our research we conclude that most of the people express positive opinion about Cricket .

**10.2 REFERENCES**

[1] Vallikannu Ramanathan, T.Meyyappan ‘An Exhaustive Exploration on Twitter Sentiment Analysis’, Journal of Computer Science and Applications, International Research Publication house, Volume 6, Number 1, 2014. ISSN 2231 -1270

[2] Bing Liu, “Sentiment Analysis and Opinion Mining” Handbook of natural language Processing, 2010.

[3] A.Esuli and F.Sebastiani. ‘SentiWordNet a publicly available lexical resource for opinion mining’ in Proceeding of the 5th International conference on Language Resources and Evaluation, pp.417-422, 2006.

[4] Minging Hu and Bing Liu. ‘Mining and Summarizing Customer Reviews’, Proceedings of the ACM SIGKDD Interenational Conference on Knowledge Discovery and Data Mining (KDD-2004), Aug 22-25, 2004, Seattle, Washington, USA.

[5] Bing Liu, Minging Hu and Junsheng Cheng. ‘Opinion Observer Analyzing and Comparing Opinions on the Web’ Proceedings of the 14th International World Wide Web conference (WWW-2005), May 10-14, 2005, Chiba, Japan.

[6] C.Havasi, R.Speer, and J.Alonso, ‘Conceptnet 3 a flexible, multilingual semantic network for common sense knowledge’ in recent Advances in Natural Language Processing, pp.27-29,2007.

[7] Erik Cambria ‘An Introduction to concept-level sentiment analysis’, In Advances in Soft Computing and Its Applications, pages 478-483. Springer, 2013.

[8] John R. Firth. A Synopsis of linguistic theory. Studies in Linguistic Analysis, 1930-1955.

[9] P.Turney ‘Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews’. In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL’02), Philadelphia, Pennsylvania, 2002.