

COMBINING TEXTUAL INFORMATION AND SENTIMENT DIFFUSION PATTERNS FOR TWITTER SENTIMENT ANALYSIS

Submitted in partial fulfillment of the requirements for
the award of Bachelor of Engineering degree in

Computer Science and Engineering

By

SAMA SAIPRATHUSHA (Reg.No.37110670)

M.RUPA (Reg. No. 37110650)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade "A" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI
SALAI, CHENNAI – 600 119.TAMILNADU.**

MARCH - 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in



SCHOOL OF COMPUTING

BONAFIDE CERTIFICATE

This is to certify that this project report is the bonafide work of **SAMA SAIPRATHUSHA** (Reg. No. 37110670) and **M.RUPA** (Reg. No.37110650) who carried out the project entitled "**COMBINING TEXTUAL INFORMATION AND SENTIMENT DIFFUSION PATTERNS FOR TWITTER SENTIMENT ANALYSIS**" under my supervision from **November 2020 to March 2021**.

Internal Guide

Ms.INDIRA K,M.TECH.,(Ph.D)

Head of the Department

Dr.S.VIGNESHWARI,M.E.,Ph.D.,

Submitted for Viva voce Examination held on_____

Internal Examiner

External Examiner

DECLARATION

I **SAMA SAIPRATHUSHA** hereby declare that the Project Report entitled **“COMBINING TEXTUAL INFORMATION AND SENTIMENT DIFFUSION PATTERNS FOR TWITTER SENTIMENT ANALYSIS”** is done by me under the guidance of Ms.INDIRA K,M.TECH.,(Ph.D) Department of Computer Science and Engineering at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala, M.E., Ph.D., Dean**, School of Computing, **Dr. S. Vigneswari, M.E., Ph.D., and Dr. L. Lakshmanan, M.E., Ph.D., Heads of the Department** of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Ms.INDIRA K,M.TECH.,(Ph.D)** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Over the recent years, Twitter has grown from a vague invention to become a mainstream medium for dissemination of messages and the public discussion of news and events. The rapid proliferation of Twitter posts presents a big obstacle for efficient information acquisition. It is impossible for a user to get an overview of important topics on Twitter by reading all tweets every day. In addition, because of information redundancy and the informal writing style, it is time consuming to find useful information about a topic from a huge number of tweets. The tremendous volume of tweets suggests summarization as the key to facilitating the requirements of topic exploration, navigation, and search from hundreds of thousands of tweets. Specifically, a summary that provides representative information of topics with no redundancy and well-written sentences would be preferred. Twitter is a popular microblogging service where users create status messages (called "tweets"). These tweets sometimes express opinions about different topics. The purpose of this project is to build an algorithm that can accurately classify Twitter messages as positive or negative, with respect to a query term. Our hypothesis is that we can obtain high accuracy on classifying sentiment in Twitter messages using machine learning techniques.

TABLE OF CONTENTS

CHAPTER No.	TITLE	PAGE No.
	ABSTRACT	V
	LIST OF FIGURES	Viii
1	INTRODUCTION	1
	1.1OVERVIEW	1
	1.2 OBJECTIVE	3
	1.3 NEED FOR TRENDIND TOPIC ANALYSIS	3
	1.4 CHALLENGES IN TRENDIND TOPIC ANALYSIS	4
2	LITERATURE SURVEY	5
3	SYSTEM ARCHITECTURE	7
	3.1PROPOSED SYSTEM	7
	3.2SYSTEM ARCHITECTURE	7
	3.3LIST OF MODULES	8
	3.4SENTIMENT CLASSIFICATION MODULE	17
4	SYSTEM DESIGN	20
	4.1USE CASE DIAGRAM	20
	4.2ACTIVITY DIAGRAM	20
	4.3SEQUENCE DIAGRAM	21
	4.4COLLABORATION DIAGRAM	21
5	SOFTWARE SPECIFICATION	22
	5.1PYTHON	22
	5.2SPYDER	22
	5.3KERAS	24
	5.4ANACONDA	24
	IMPLEMENTATION AND TESTING THE MODULE	25
6	6.1WHITE BOX TEXTING	25
	6.2WHITE BOX TESTING FOR PREPROCESSING AND FEATURE EXTRACTION	26
	6.3WHITE BOX TESTING FOR SPLITING DATA	27
	6.4WHITE BOX TESTING FOR PREDICTION AND ACCURACY	29
7	SYSTEM REQUIREMENT	32
8	APPLICATIONS	33
9	CONCLUSION AND FUTURE ENHANCEMENTS	35

APPENDICES	37
REFERENCES	42

LIST OF FIGURES

FIGURE No.	FIGURE NAME	PAGE No.
3.1	SYSTEM ARCHITECTURE OF TRENDING TOPIC ANALYSER	8
3.2	TRENDING TOPIC COLLECTION	9
3.3	TOPIC BASED TWEET EXTRACTION	10
3.4	TWEET PRE-PROCESSING	11
3.5	TEXT FEATURE EXTRACTION	12
3.6	PARTS OF SPEECH TAGGED TWEETS	14
3.7	TEXT FEATURES WITH FREQUENCY COUNT	14
3.8	NON TEXT FEATURE EXTRACTION	15
3.9	NETWORK FEATURES EXTRACTION USING PARENT CHILD RELATIONSHIP	16
3.10	TASC MODEL LEARNING	18
4.1	USE CASE DIAGRAM	20
4.2	ACTIVITY DIAGRAM	20
4.3	SEQUENCE DIAGRAM	21
4.4	COLLABORATION DIAGRAM	21
6.1	WHITE BOX TESTCASE ERROR SNAPSHOT FOR PRE-PROCESSING AND FEATURE EXTRACTION	26
6.2	WHITE BOX TESTCASE ERROR SNAPSHOT	27

	FORSPLITING DATA	
6.3	WHITE BOX TESTCASE ERROR SNAPSHOT FOR PREDICTION AND ACCURACY	28
6.4	BLACKBOX TESTING FOR ALL MODULES	30
6.5	SCREENSHOT FOR MODULE 1	31
6.6	SCREENSHOT FOR MODULE 1	31

1.1 OVERVIEW

CHAPTERTER 1 INTRODUCTION

Over the recent years, Twitter has grown from a vague invention to become a mainstream medium for dissemination of messages and the public discussion of news and events. The rapid proliferation of Twitter posts presents a big obstacle for efficient information acquisition. It is impossible for a user to get an overview of important topics on Twitter by reading all tweets every day. In addition, because of information redundancy and the informal writing style, it is time consuming to find useful information about a topic from a huge number of tweets. The tremendous volume of

tweets suggests summarization as the key to facilitating the requirements of topic exploration, navigation, and search from hundreds of thousands of tweets. Specifically, a summary that provides representative information of topics with no redundancy and well-written sentences would be preferred.

To alleviate this problem, many applications have evolved from Twitter (serving as their clients), like “echofon”, “whatthetrend”, which provide services to explain why the term becomes a trending topic or to give a short description of the topic. These systems generally track the topics in Twitter and use existing tweets or encourage users to edit a new tweet to explain the topics. “Whatthetrend” encourages users to edit explanatory tweets about topics. It ranks the submitted explanatory tweets by readers’ agreements. These explanatory tweets can be regarded as tiny summaries about the topic, providing a good way to help users understand the topic. However, a short summary can only sketch the topic in a simple way. Some researchers attempt to aggregate several explanations into one long summary using traditional summarization approaches, but it still loses much useful information, such as the change of twitters’ focus and the temporal information. A well-generated traditional summary can reflect the overall picture of topic, but performs poorly in summarizing these temporal changes of the crowd’s focus in Twitter. Note that the focus of tweets changes much more frequently than that of the traditional mainstream media.

Other applications like “Summary Card” provides a full-width prominent image alongside a tweet. It is designed to give the reader a rich photo experience, and clicking on the image takes the user to hostwebsite.

Analysis of topics is truly strengthened by performing a sentiment classification but summarization applications lack this component and as a result produce conflicting summaries. Topics discussed in Twitter are very diverse and unpredictable. Sentiment classifiers always dedicate themselves to a specific domain or topic. Namely, a classifier trained on sentiment data from one topic often performs poorly on test data from another. Most of the recent applications for sentiment analysis deal with a single topic under interest. Another concern regarding sentiment classification is if carried out as a supervised approach it will incur heavy manual effort in annotation.

1.2 OBJECTIVE

Given a topic, generate a serial of chronologically related sub-summaries (extractive) by using Multi Tweet Sequential Extractive Summarization.

- o To avoid generation of conflicting summaries
- o To resolve lack of temporal summaries showcasing a complete topic evolution
- o To detect sub topics in tweet stream for sequential sub summary generation

PROBLEM STATEMENT

To implement a Trending Topic Analysis System that would analyze the given topic by performing Topic adaptive sentiment classification thereby summarizing public views on selected trending topics and being able to generate extractive sub summaries of sub topics over the time period.

1.3 NEED FOR TRENDING TOPIC ANALYSIS

Twitter serves as an information-delivering platform where it collects millions of tweets every day. However, some users, especially new users, often find it difficult to understand trending topics in Twitter when confronting the overwhelming and unorganized tweets. Existing work has attempted to provide a short snippet to explain a topic, but this only provides limited benefits and cannot satisfy the users' expectations.

Trending Topic Analysis system provides a deep analysis via topic adaptive sentiment classification and multi tweet sequential summarization, which aims to provide a serial of chronologically, ordered short sub-summaries for a trending topic in order to provide a complete story about the development of the topic while retaining the order of information presentation.

Communication experts are using Twitter data to measure the pulse of public opinion and generate business intelligence for which Trending Topic Analysis system will be of great value.

One of the applications of trending topic analyzer finds its place in Political domains where a political party can track the public's views on topics of their interest which had been trending once and use it improve their political status. Commercially content generators to perform a deep analysis of any trending topic under study and use to improve their commercial benefits could use it. In recent times, Trending Topic

Analysis system also plays a crucial role in Research areas due to the increased popularity of Twitter. This analysis cum summarization system could serve as rich tool

of focusing only on significant areas of topics and overcome the overwhelming nature oftweets.

1.4 CHALLENGES FOR TRENDING TOPIC ANALYSIS

Nature of tweets poses two major challenge namely,

- o Diversity inTopics
- o Noise inTweets

Diversity inTopics

Users discuss range of topics over an increasing time speed, which poses a major challenge for automatic summarization. Diversity also comes in form of varied form of data to handle.

Noise inTweets

Tweets poses some unique characteristics, which makes it noisy to handle. Firstly, all tweets are limited to 140 characters. Some tweets are news headlines from the official media; users generate others with various degrees of familiarity with the social media. The resulting tweets can be very different regarding the text quality and word usage. Secondly, Tweets lack structure information, contain various ill-formed sentences and grammatical errors. There are lots of noisy nonstandard tokens, such as abbreviations (“feelin” for “feeling”), substitutions (“Pr1mr0se” for “Primrose”), emoticons, etc. Third, Twitter invented its own mark-up language. “@user” is used to reply to a specific user or call for attentions. The hashtag “#topic” aims to assign a topic label to the tweet, and is frequently employed by the twitter users. Finally, Tweets frequently contain embedded URLs that direct users to other online content, such as news web pages, blogs, and organization homepages.

CHAPTER 2

LITERATURE SURVEY

LIST OF LITERATURE SURVEY

1. Phrase ReinforcementAlgorithm
2. Non-Parametric BayesianModel
3. Concept Based OptimizationFramework
4. Simple List ofMessages
5. Stream and Semantic BasedApproaches

SUMMARY

1. G. MANE ET AL - PHRASE REINFORCEMENTALGORITHM

He proposed a combined approach of Phrase Reinforcement algorithm along with Word Sense Disambiguation and Textual Entailment techniques for generating one line summary. Phrase Reinforcement algorithm aimed at constructing a graph that helps in identifying the most commonly occurring phrases for a central topic by simply searching for the most weighted set of paths through the graph. This methodology lacked temporal nature of summaries and created coherence issues in the summary generated.

2. D. WEN ET AL - NON-PARAMETRIC BAYESIANMODEL

He accomplished Summarization using a non-parametric Bayesian model applied to Hidden Markov Models. A novel observation model was designed to allow ranking based on selected predictive characteristics of individual tweets. Major focus was to investigate the possibility of using a temporal probabilistic data model known as Hierarchical Dirichlet Process Hidden Markov Model (HDP-HMM) to process a stream of tweets pertaining to a single subject and cluster the tweets into groups or rankings based on the value of the individual tweets .But Summaries generated did not project the temporal nature.

3. F. LUI ET AL - CONCEPT BASED OPTIMIZATIONFRAMEWORK

He proposed a Concept Based optimization framework for topic Summarization using Integer Linear Programming. Target data comprised of original and normalized tweets along with web content relevant to the topic. The focus was not on developing new summarization systems but rather utilizing and integrating diverse text sources to generate summaries that are more informative but there were lack of series of sub events identification to show topic- evolving process.

4. B. O'CONOR ET AL - SIMPLE LIST OF MESSAGES

He explained Twitter topics by presenting a simple list of messages. An exploratory search application for Twitter called TweetMotif grouped messages by frequent significant terms and result set is subtopics facilitated navigation and drilldown through a faceted search interface. The topic extraction system was based on syntactic filtering, language modelling, near-duplicate detection, and set cover heuristics. However, the system lacked temporal aspect in summaries generated and topic development was not observed.

5. D. GAO ET AL - STREAM AND SEMANTIC BASED APPROACHES

He generated sequential summaries on a topic using Stream and Semantic based approaches. However, the approach fell short of readability of summaries generated. System failed to capture the opinion expressed in the data thus leading to conflicting summaries.

CHAPTER 3 SYSTEM ANALYSIS

3.1 PROPOSED SYSTEM

The proposed system “Trending Topic Analyser” is accomplished through topic adaptive sentiment classification and multi tweet summarization. The proposed system aims at generating chronologically ordered sub summaries, which throws light in understanding the topic evolution of the trending topics. Topic under study is assumed to contained several hidden sub topics, which are revealed using the proposed system through Foreground Dynamic Topic Modelling. This enables an end user to completely analyse a trending topic to a greater level of detail.

Target data collection is performed by extracting the trending topic on a region basis. Topic under study is assumed to contain several hidden sub topics, which are revealed using the proposed system. This enables an end user to completely analyse a trending topic to a greater level of detail. Necessary pre-processing is performed to prepare the target useable data. Along with the pre-processing phase, the proposed system also handles non-English tweet translation that is essential to prevent discarding of public opinion about the trending topic.

Pre-processed data is later analysed through topic adaptive sentiment classification to identify the public opinion. Sentiment labelled data will then be processed for sequential summarization. Sub topic detection is achieved using Stream based approach and Semantic based approach. Finally, sub summary candidate selection is executed which gives the highest scored tweets incorporating certain unique features of tweets. Redundancy check is performed to remove duplicates from the selected tweets followed by a threshold check to ensure fair mixture of user's opinions.

3.2 SYSTEM ARCHITECTURE

The system architecture is illustrated in Figure 3.1. The proposed framework analyses a trending topic based on a particular region. Execution of the proposed may be broadly viewed into three categories.

Target data collection which involves extracting trending topic and filtering the

selected topics for further analysis. Following it is the tweet extraction based on the topic. Then, Pre- processing module cleans and prepares the workable data set.

Pre-processed tweets are then fed into the sentiment classifier which will be labelled adaptively. Classifier trained is evaluated as an online process. Sentiment classification process address the issue of generation of conflicting summaries.

Finally the sequential summarization phase in this project handles Topic evolution issue. Latent topics are hidden in the dataset which needs an efficient topic detection models. In this project Stream based sub topic detection models and Semantic based sub topic detection models help in achieving the task of discovering the sub topics within the trending topic. Then, using graph based approach significant tweets will be selected and topic wise sub summaries are generated in chronological order and presented as extractive sub summaries. Detailed system architecture is sketched as shown in Figure 3.1.

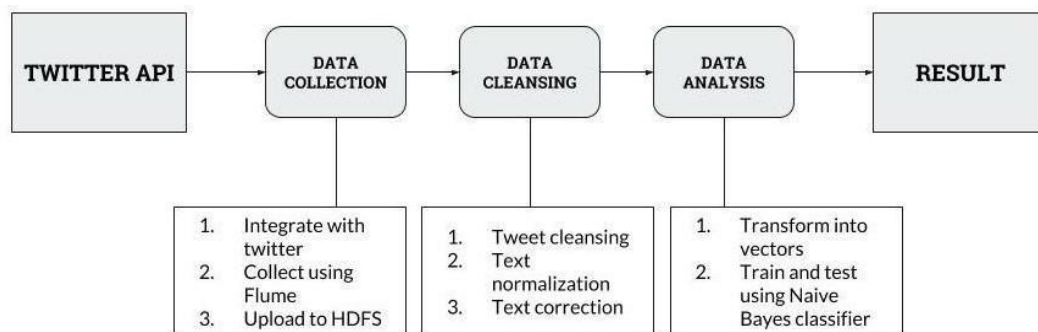


Fig 3.1 System architecture of trending topic analyser

3.3 LIST OFMODULES

The proposed system is composed of the following modules:

1. Data ExtractionModule
2. Pre-processingModule
3. Feature ExtractionModule
4. Sentiment ClassificationModule

DATA EXTRACTIONMODULE

Trending TopicCollection

Trending topics will be extracted using the region's "WOEID" which uniquely identifies any region in the world. Twitter API will enable extraction of region wise trends. Extracted trend will be analysed to confirm to the requirement that the topic involves several sub topics hidden in it. Trending topics will be stored in the

database. While storing the trending topics into the database, they will be tagged

with the month range during which it trended. This will enable user to pick a topic to analyse based on the month range. *tagCurrentMonthRange()* method mentioned in the following algorithm performs the above-mentioned task. Trending topic extraction steps is illustrated in Figure 3.2.



Fig 3.2 Trending topic collection

Following algorithm discusses the steps involved in Trending Topic Collection.

Algorithm- TRENDING TOPIC COLLECTION

Input: Target place's *WOEID*

Output: Set of Trending

Topics ***begin***

```

    connectto Twitter ()
    for each trend do ()
        addto target topic database
    endfor
  
```

end

Function ()

tags a month range for the set of trends extracted

Function ()

gets top trending topics of that region

WOEID mentioned in method *getTrends()* decides the region of interest to extract top ten trends per request.

Topic based TweetExtraction

Once the trends are extracted, keyword based tweet extraction will be

performed using the extracted trend as the keyword as shown in Figure 3.3. Twitter's JSON response will be parsed and relevant details will be recorded in database.

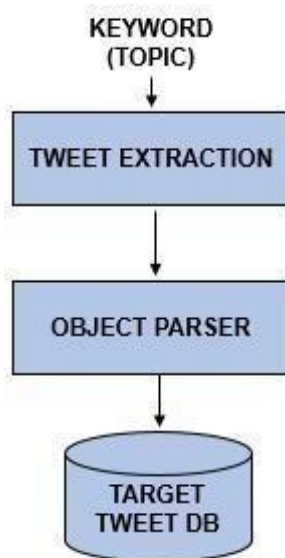


Fig. 3.3 Topic based tweet extraction

Following Algorithm describes the procedure for Topic Based Tweet Extraction where it is shown that tweets of language “English” and “Hindi” are to be extracted.

Algorithm- TOPIC BASED TWEET EXTRACTION

Input: A Trending Topic as keyword

Output: Target Tweets

Begin

C
o
n
n
e
c
t
End **t**
t
o
T
w
i
t
t
e
r

(	t
)	w
J	e
s	e
o	t
n	—
=	s
f	t
o	r
r	i
e	n
a	g
c	=
h	(
j	
s	)
o	If (.== . == h) add the target
n	tweets to tweetdatabase
d	End if
o	End for

Function ()

Parse and fetch required fields.

PRE-PROCESSINGMODULE

Tweets by nature involve many noisy content and ill-formed words because the users are allowed to tweet only within limited characters. Therefore, cleaning and preparing the meaningful data is essential for any kind of analysis to be performed on the data. Tweet pre- processing involves the following task: URL Removal- URL and links will be removed from the tweet content. Slang Word Replacement-Slang words like LOL, OMG will be replaced its appropriate English words i.e. Laugh out C

Though tweets could be extracted using specific language, some noisy terms and words corrupt the target data. Non-English words will be filtered with the help of a dictionary. Stemmer and Stop word removal-English words will be stemmed and only the root words will be retained. Stop words in the tweet content will be removed.

Detailed flow of the pre-processing steps is shown in Figure 3.4 along with the algorithm steps described as below.

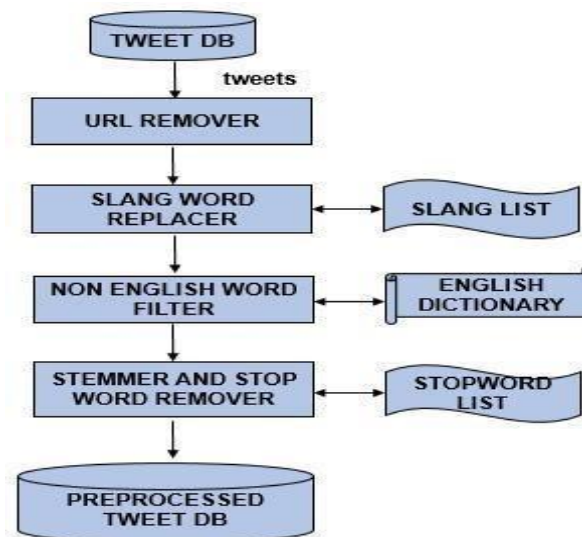


Fig.3.4 Tweet pre-processing

Algorithm below details each of the above-mentioned pre-processing steps and results are stored in the database for further modules to act on.

Algorithm- PRE-PROCESSING OF TWEETS

Input: Target Tweets

Output: Pre-processed target tweets

Begin

End

F
o
r
e
a
c
h
t
w
e
e
t
d
o
I
f
(

.

(
)
)

Remove

End if

For each word in tweetdo()

End for

For each word in tweetdo ()

End for

For each word in tweetdo()

End for End for

FEATURE EXTRACTION MODULE

Feature extraction involved in this system creates a significant impact in the performance of sentiment classification task. Tweets have a special nature unlike normal English sentences like @ symbol used to refer to a user, emoticons expressed in the tweets etc. Incorporating such features while performing feature extraction enhances the overall process.

Text Feature Extraction

Text features are a combination of Common Sentiment words and topic sentiment words which will be used to train the classifier. With POS tagging for tweets on a topic and removing the common sentiment words, proposed framework selects the frequent adjectives, verbs, nouns and adverbs as candidates of topic-adaptive. Figure 3.5 depicts the process involved.

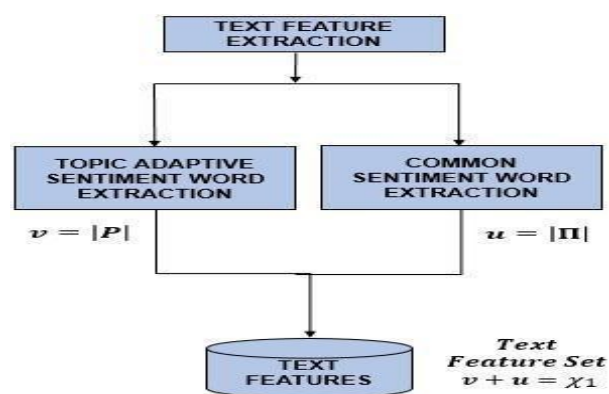


Fig. 3.5 Text feature extraction

Algorithm- TEXT FEATURE EXTRACTION USING POS TAGGER

Input: Pre-processed target tweets

Output: Topic adaptive sentiment

word **For each** pre-processed tweet

in topic do

 words [] = POSTagger

 (tweet) frequencyCount (words

 []) targetTagList (v, n, a_v, a_j)

For each word **in** targetTagList

If (word **not in**

 CommonSentimentWordList)

addTopicAdaptiveWordList (word)

End if

End for End for

```

<terminated> POSTagging [Java Application] C:\Program Files\Java\jre1.8.0_51\bin\javaw.exe (17-Aug-2015 7:32:07 am)
@kaminiba_NN final_JJ happen_VB rt_NN theflashat9_IN
@coloursinfin_NN amaz_NN best_JJS goodi_NN hope_NN theflashat9_RB want_VBP
@coloursinfin_JJ arrow_NN green_JJ theflashat9_NN
eagerli_JJ theflashat9_NN wait_NN
@coloursinfin_NN a1_NN grant_NN theflashat9_IN
@coloursinfin_NN coloursinfin_NN episod_NN love_NN meet_VBP rt_NN theflashat9_NN three_CD till_IN tonight_NN wait_NN
@coloursinfin_NN awesom_NN come_VBP dont_NN goodi_NN last_JJ on_IN question_NN rt_NN soon_RB stop_VB theflashat9_RB tweet_JJ
@coloursinfin_NN answer_NN team_NN theflashat9_NN
6_CD @coloursinfin_NN contestalert_NN hour_NN question_NN readi_NN rt_NN start_NN theflashat9_NN todai_NN
1_CD @coloursinfin_NN contest_NN number_NN on_IN question_NN rt_NN theflashat9_NN tweet_NN winner_NN
@coloursinfin_JJ ad_NN contest_NN last_JJ on_IN question_NN readi_NN rt_NN theflashat9_NN
@coloursinfin_NN aliv_NN allen_NN barri_NN fastest_JJS man_NN name_NN rt_NN theflashat9_NN
@coloursinfin_JJ contest_NN continu_NN frenzi_NN goodi_NNS grab_VBP rt_NN theflashat9_NNS win_VBP
@coloursinfin_NN air_NN answer_NN coloursinfin_NN easi_NNS flash_VBP gui_NNS on_IN rt_NN theflashat9_NN will_MD
@coloursinfin_NN @magicianbobo_NN cant_NN enough_RB gui_FW mondai_FW never_RB thank_VB theflashat9_NN will_MD
9_CD @dildeep_NN fresh_JJ mondai_NNS new_JJ pm_NN show_NN theflashat9_IN
1_CD @coloursinfin_NN contest_NN number_NN on_IN question_NN rt_NN theflashat9_NN tweet_NN winner_NN
@coloursinfin_JJ awai_NNS back_RB blue_JJ coloursinfin_NN go_VB hour_NN just_RB mondai_FW shew_FW theflashat9_NN three_CD

```

Fig.3.6 Parts of speech tagged tweets

Pre-processed tweets are selected for text feature where POS Tagging is applied as shown in Figure 3.6. Each word will be tagged with its respective parts-of-speech. Most frequent verbs, nouns and adjectives are taken as the text features. Figure 3.7 displays the verbs, nouns, adjectives with itscount.

```

<terminated> POSTagging [Java Application] C:\Program Files\Java\jre1.8.0_51\bin\javaw.exe (17-Aug-2015 7:32:07 am)
@coloursinfin_nn ==== 4192
theflashat9_nn ==== 3908
rt_nn ==== 1604
@coloursinfin_jj ==== 993
theflashat9_in ==== 993
question_nn ==== 936
theflashat9_rb ==== 922
contest_nn ==== 875
answer_nn ==== 775
us_prp ==== 764
super_jj ==== 695
coloursinfin_nn ==== 692
win_vb ==== 655
win_vbp ==== 518
goodi_nn ==== 503
on_in ==== 472
tonight_nn ==== 471
speed_nn ==== 433
readi_nn ==== 421
back_rb ==== 390
wait_vb ==== 386
want_vbp ==== 364
next_jj ==== 362
allen_nn ==== 342
power_nn ==== 339
awesom_nn ==== 335
arrow_nn ==== 330
your_prp$ ==== 311

```

Fig.3.7 Text features with frequency count

Words that satisfy a threshold will be filtered and treated as Topic Adaptive words for the set. Same procedure is repeated for the topics understudy.

Non-Text Feature Extraction

Non-text features include the @-network features, user sentiment features and emoticons. These features capture the special nature of tweets as features. In twitter, @ symbol in a tweet is used to refer to a user whom a person would tweet to e.g. @abc means that the tweet is referring the user abc. Using this symbol useful feature can be extracted there by eliminating the need to label the entire dataset. Figure 3.8 shows the three types of non-text features extracted.

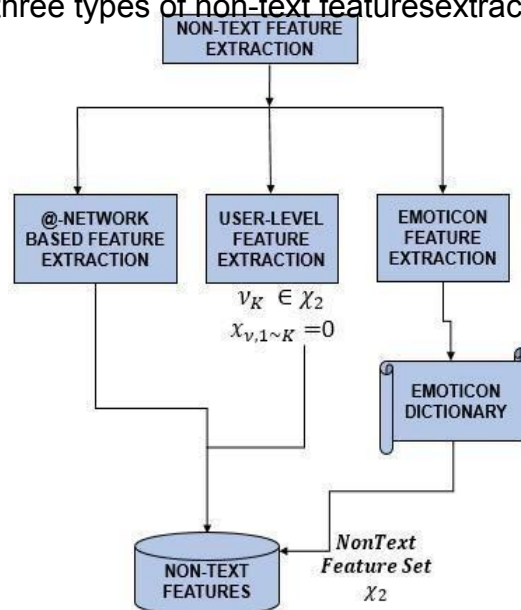


Fig.3.8 Non text feature extraction

(a) @ Network Feature

"@" is a commonly used convention in a tweet, e.g. user a_i mentioned a_j via a tweet t containing "@ a_j ". Such a @-relation reflects the dependencies between the sentiments of tweet t and @-user a_j 's tweets in the same context. Thus, the proposed system builds a graph to model @-network, with tweets as vertices and @-relations between tweets as directed edges. There are directed edges from tweet t to all a_j 's tweets in the example. Those a_j 's tweets posted earlier than tweet t are called t 's parents; otherwise, those tweets posted later are its children. The parent with the nearest posting time to tweet t is its direct parent, and similarly the nearest child is its direct child. Algorithm below details the flow of extracting network based graph model.

Algorithm- EXTRACTING @-NETWORK GRAPH MODEL

Input: Topic specific target tweets

Output: A directed @-network graph model

For each tweet in topic e in training set L do

initialize (vertices-tweet)

initialize (d-edges- @-relations)

End for

For every $vertex_i$ **in** vertices **do**

For every $vertex_j$ **in** vertices **do**

If ($postTime(vertex_i) < postTime$

$(vertex_j)$) **createCEdge**

$(vertex_i, vertex_j)$

Else **createPEdge** ($vertex_i, vertex_j$)

End if

End for

End for

```
<terminated> ParentChildRelation [Java Application] C:\Program Files\Java\jre1.8.0_51\bin\javaw.exe (14-Oct-2015 1:52:35 pm)
parent Count-->0----Child Count-->0
String Array Values-->[Ljava.lang.String;@4de5031f
arrayvalues length-->0
Dynamically assigned variable names-->[]
String graph values-->([27226], [])
Count Value-->12
parent Count-->0----Child Count-->0
String Array Values-->[Ljava.lang.String;@5d47c63f
arrayvalues length-->0
Dynamically assigned variable names-->[]
String graph values-->([27249], [])
Count Value-->12
parent Count-->2----Child Count-->2
String Array Values-->[Ljava.lang.String;@3bbc39f8
arrayvalues length-->4
Dynamically assigned variable names-->[v0, v1, v2, v3]
v0---30129
v1---30564
v2---30685
v3---34631
String graph values-->([27249, 30129, 30564, 30685, 34631], [{27249,30129}, {27249,30564}, {27249,30685}, {27249,34631}])
Count Value-->13
parent Count-->0----Child Count-->0
String Array Values-->[Ljava.lang.String;@52e677af
arrayvalues length-->0
Dynamically assigned variable names-->[]
String graph values-->([27256], [])
Count Value-->13
parent Count-->2----Child Count-->2
String Array Values-->[Ljava.lang.String;@8e0379d
arrayvalues length-->4
Dynamically assigned variable names-->[v0, v1, v2, v3]
v0---30129
v1---30564
v2---30685
v3---34631
```

Fig 3.9 Network features extraction using parent child relationship

(b) UserFeature

A user's sentiment on a topic is consistent, which could be determined by his tweets. It in turn means the sentiment of tweets posted by a user is correlated to his opinion. Algorithm below discusses the user level feature extraction in detail.

Algorithm- EXTRACTING USER LEVEL FEATURES

Input: Topic specific target tweets

Output: User's sentiment and tweet post time

For each tweet in topic e in training set

do **If** (user.tweetCount > 2)

 addUserList (user)

End if

End for

For every user in

 addUserList **do**

 extractSentiment (user)

 extractPostTime

 (user.tweet)

End for

(c) EmoticonFeature

Emoticons in tweets are considered as a non-text feature. Extract the emoticons in each tweet and assign +1 for positive emoticon, -1 for negative and 0 for neutral. Add the values which will be the emoticon feature of the tweet. The system uses an emoticon dictionary for assigning values.

3.4 SENTIMENT CLASSIFICATIONMODULE

In traditional system, summaries may contain conflicting content that reduces the readability and the overall impact of summaries generated. Partially similar to [8], this system aims to perform sentiment classification and then using topic wise sentiment labelled data proceeds to the summarization module.

Sentiment classification is a topic-sensitive task, i.e., a classifier trained from one topic will perform worse on another. This is especially a problem for the tweets sentiment analysis. Since the topics in Twitter are very diverse, it is impossible to train a universal classifier for all topics. Moreover, compared to product review, Twitter lacks data labelling and a rating mechanism to acquire sentiment labels. The extremely sparse text of tweets also brings down the performance of a sentiment classifier. In this paper, we propose a semi-supervised topic- adaptive sentiment classification model, which starts with a classifier built on common features and mixed labelled data from various topics

Semi supervised support vector machines S3SVM is one of the most promising

candidates to utilize unlabelled data combining with a small amount of labelled ones, since SVM minimizes the structural risk. In addition, collaborative training (co-

training) framework is an alternative wrapper, and achieves a good performance, which is often used in the scenarios whose features are easily split into different views.

This system adopts the text and non-text features, x_1 and x_2 as independent views for co-training. In co-training scheme, two classifiers C_1 and C_2 are trained based on x_1 and x_2 separately using initial labelled data L . The corresponding feature values are denoted as x and x' respectively for text and non-text feature values.

The unlabelled data are selected collaboratively to augment labelled data set L , which is used for the next iteration. In addition, the classifier trained on the combining features using augmented labelled data set L . obtains the final sentiment classification result.

Figure 3.10 depicts the process flow of the algorithm followed by TASC learning algorithm is described where adapting to unlabelled data and adapting to features is performed iteratively returning the model , augmented labelled data and features.

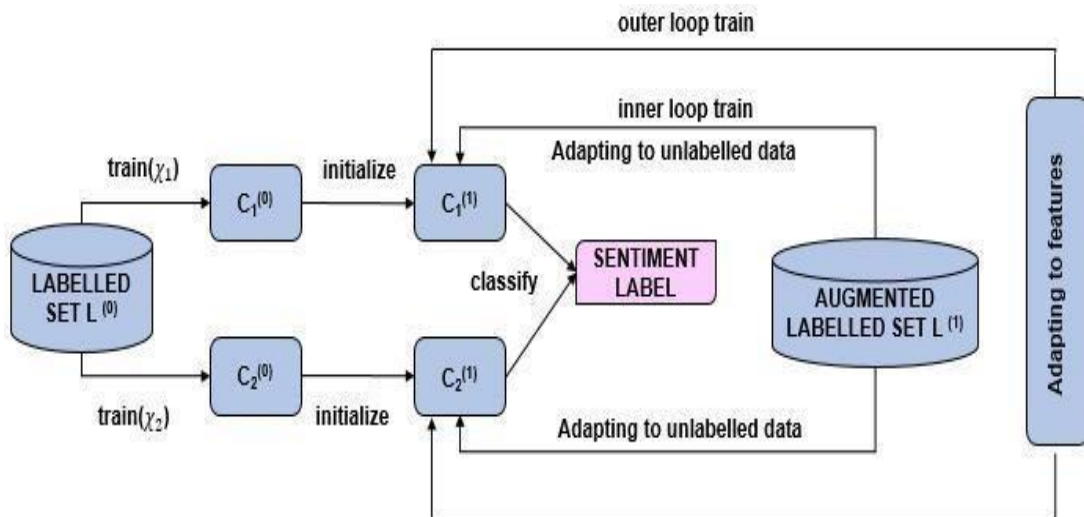


Fig. 3.10 TASC model learning

Figure 3.11 displays the results of sentiment classification showing the number of correctly classified records and its accuracy.

Tomcat v8.0 Server at localhost [Apache Tomcat] C:\Program Files\Java\jdk1.8.0_51\bin\javaw.exe (14-Oct-2015 1:26:57 pm)

```

testFileName-->C:/Files/CoTrain/YakubMemon/data/test/day1/test.arff
testFileName-->C:/Files/CoTrain/YakubMemon/data/test/day1/test.arff
Classifiers Label :negative
testFileName-->C:/Files/CoTrain/YakubMemon/data/test/day1/test.arff
Classifiers Label :positive
testFileName-->C:/Files/CoTrain/YakubMemon/data/test/day1/test.arff
Classifiers Label :positive
testFileName-->C:/Files/CoTrain/YakubMemon/data/test/day1/test.arff
Classifiers Label :positive
testFileName-->C:/Files/CoTrain/YakubMemon/data/test/day1/test.arff
Classifiers Label :positive
testFileName-->C:/Files/CoTrain/YakubMemon/data/test/day1/test.arff
Classifiers Label :neutral
testFileName-->C:/Files/CoTrain/YakubMemon/data/test/day1/test.arff
Classifiers Label :positive
testFileName-->C:/Files/CoTrain/YakubMemon/data/test/day1/test.arff
Classifiers Label :positive
testFileName-->C:/Files/CoTrain/YakubMemon/data/test/day1/test.arff
Classifiers Label :positive
testFileName-->C:/Files/CoTrain/YakubMemon/data/test/day1/test.arff
Classifiers Label :negative
testFileName-->C:/Files/CoTrain/YakubMemon/data/test/day1/test.arff
Classifiers Label :positive
testFileName-->C:/Files/CoTrain/YakubMemon/data/test/day1/test.arff
Classifiers Label :positive

```

Fig.311 Sentiment classification results

CHAPTER 4 SYSTEM DESIGN

4.1 USE CASE DIAGRAM



Fig.4.1 Use case

4.2 ACTIVITY DIAGRAM

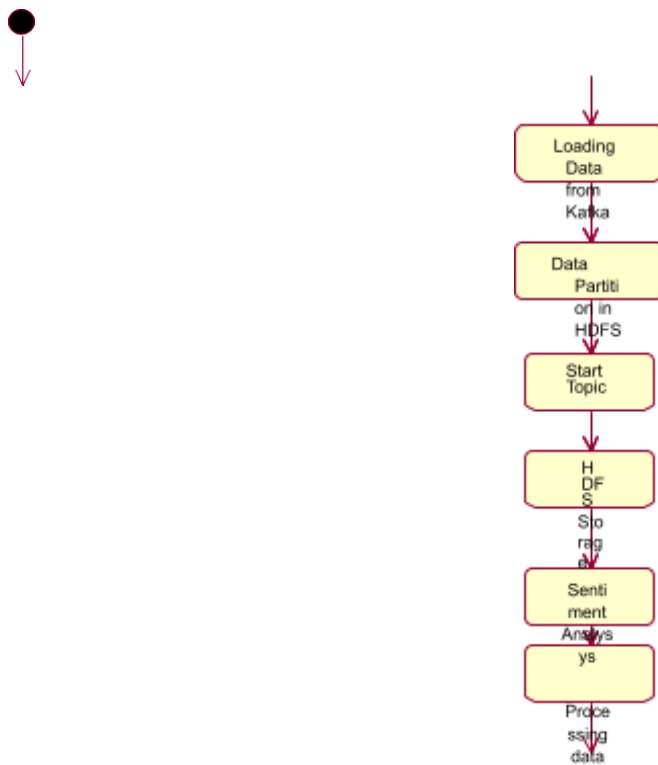


Fig.4.2 Activity

4.3 SEQUENCEDIAGRAM

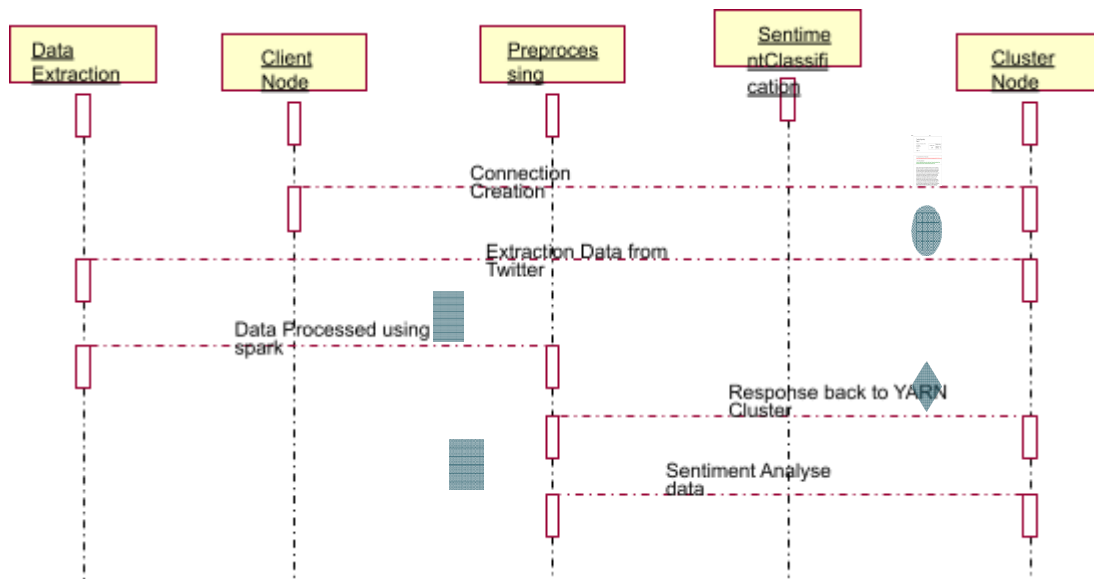


Fig.4.3 Sequence

4.4 COLLABORATIONDIAGRAM

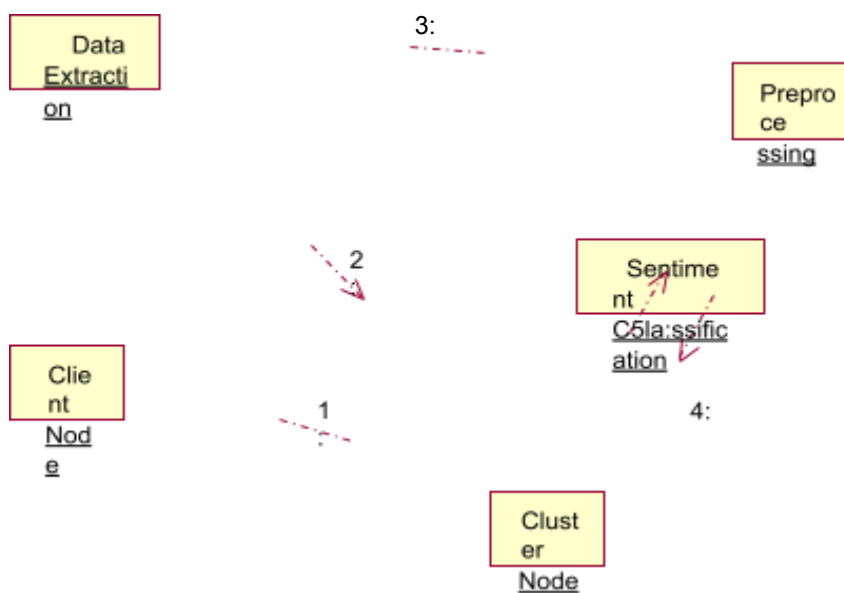


Fig.4.4 Collaboration

CHAPTER

5

SOFTWARE

RE

SPECIFI

CATION

5.1 PYTHON

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object oriented, imperative, functional and procedural, and has a large and comprehensive standard library. Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of Python's other implementations. Python and CPython are managed by the non-profit Python Software Foundation. Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming. Many other paradigms are supported via extensions, including design by contract and logic programming.

Python uses dynamic typing, and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution, which binds method and variable names during program execution. Python's design offers some support for functional programming in the Lisp tradition. It has `filter()`, `map()`, and `reduce()` functions; list comprehensions, dictionaries, and sets; and generator expressions. The standard library has two modules that implement functional tools borrowed from Haskell and Standard ML.

5.2 SPYDER

Spyder is an open source cross-platform integrated development environment (IDE) for scientific programming in the Python language. Spyder integrates with a number of prominent packages in the scientific Python stack, including NumPy, SciPy, Matplotlib, pandas, IPython, SymPy and Cython, as well as other open source software. It is released under the MIT license.

Initially created and developed by Pierre Raybaut in 2009, since 2012 Spyder has been maintained and continuously improved by a team of scientific Python developers and the community.

Spyder is extensible with first- and third-party plugins, includes support for interactive tools for data inspection and embeds Python-specific code quality and Ubuntu. Spyder uses Qt for its GUI, and is designed to use either of the PyQt or PySide Python bindings. QtPy, a thin abstraction layer developed by the Spyder project and later adopted by multiple other packages, provides the flexibility to use either backend.

5.3 KERAS

Keras is an Open Source Neural Network library written in Python that runs on top of Theano or Tensorflow. It is designed to be modular, fast and easy to use. It was developed by François Chollet, a Google engineer. Keras doesn't handle low-level computation. Instead, it uses another library to do it, called the "Backend. So Keras is high-level API wrapper for the low-level API, capable of running on top of TensorFlow, CNTK, or Theano.

Keras High-Level API handles the way we make models, defining layers, or set up multiple input-output models. In this level, Keras also compiles our model with loss and optimizer functions, training process with fit function. Keras doesn't handle Low-Level API such as making the computational graph, making tensors or other variables because it has been handled by the "backend" engine.

5.4 ANACONDA

Anaconda is a free and open-source distribution of the Python programming languages for scientific computing that aims to simplify package management and deployment. Package versions are managed by the package management system conda. The Anaconda distribution is used by over 12 million users and includes more than 1400 popular data-science packages suitable for Windows, Linux, and MacOS.

Anaconda is a Python-based data processing and scientific computing platform. It has built in many very useful third-party libraries. Installing Anaconda is equivalent to automatically installing Python and some commonly used libraries such as Numpy,

Pandas, Scipy, and Matplotlib, so it makes the installation so much easier than regular Python installation. It is painful and you need to consider compatibility, thus it is highly recommended to directly install Anaconda.

The Machine Learning Concepts can be well implemented through PYTHON . We have a numerous Python Tools but for the DNN implementation Anaconda can meet our needs. Spyder in Anaconda is chosen as the IDE setup . The latest Anaconda Spyder has Python 3.6 version . Python 3.6 is unstable to hold the KERAS backend . Apart from the inability to build over Python 3.6 , KERAS requires two additional library packages THEANO and TENSORFLOW . Tensorflow can be implemented over Python 3.5 only .

IDE Setup

A cloning environment is built using the Anaconda Prompt . The Python 3.5 is cloned using the command .

conda create -n py35 python=3.5 anaconda

After activating the cloned environment the Spyder is installed using the command . So the Python3.5 has its own Spyderversion , which does not interfere with the base version of Anaconda .

conda install spyder

The required libraries can be set up using the below commands

```
conda install theano
conda install tensorflow
conda install keras
conda install wordclo
```

CHAPTER 6

IMPLEMENTATION

Tweets from twitter will be in JSON structure. A sample tweet JSON is shown below. These JSON can be parsed using any programming languages like Java, JavaScript. There are many third party frameworks are available in Maven repository. Appendix A1.1 explains parsing JSON using Java.

SYSTEM TESTING

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 provide a way to check the functional of components. It is the process of exercising software with the intent 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 address a specific testing requirement.

6.1 WHITE BOX TESTING

White box testing's testing which include the error in coding section it includes the error occurred during compliant and also during the development of the project. This project is verified and tested using white box testing. The following screenshots are the sample of the error coding and also how it is rectified. The error in the sample is because the coding is not proper. The coding is rewritten and corrected to solve the error.

6.2 WHITE BOX TESTING FOR PRE-PROCESSING AND FEATURE EXTRACTION

The below Figure 6.1.1 explains white box testing which describes the importing of label encoder and one hot encoder to transform the data from its bit level 0. Here found String Attribute value missing without suitable in label encoding declaration.

ERROR CODE:

```
dataset = pd.read_csv('C:\\Users\\Win 10\\Downloads\\EmergencyData.csv')
"dataset = pd.read_csv('C:\\Users\\BALAJI KRISHNARAJ\\Downloads\\EmergencyData.csv')"
X = dataset.iloc[:, 0:14].values
y = dataset.iloc[:, 14].values
```

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
```

SYNTAX ERROR

```
labelencoder_X_0 = LabelEncoder()
```

```
X[:, 0] = labelencoder_X_0.fit_transform(X[:, 0])
```

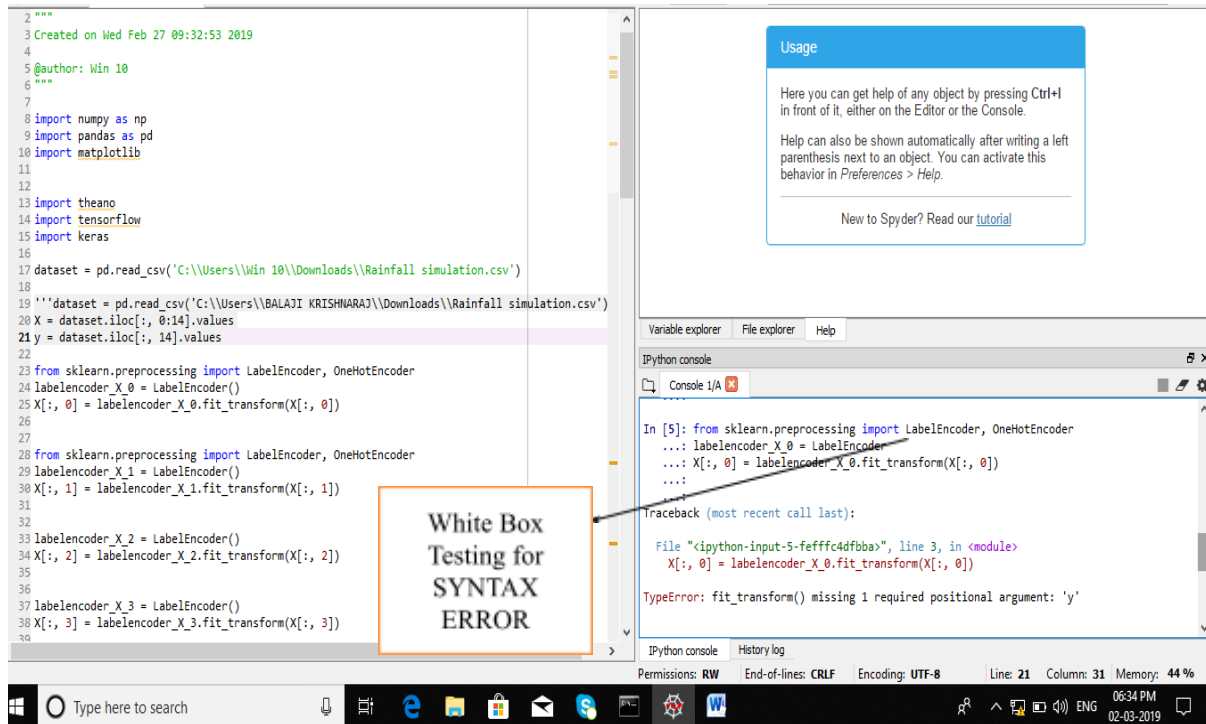


Figure 6.1 White Box Test Case Error Snapshot for Pre-processing and Feature Extraction

Error Description and Rectified Code:

Here, White Box Testing for Pre-processing is done.

Data were collected and pre-processed within the dataset.

Label encoding and one hot encoding were done.

Error Code: `labelencoder_X_0 = LabelEncoder()`

`X[:, 0] = labelencoder_X_0.fit_transform(X[:, 0])`

Rectified Code: `labelencoder_X_0 = LabelEncoder()`

`X[:, 0] = labelencoder_X_0.fit_transform(X[:, 0])`

6.3 WHITE BOX TESTING FOR SPLITTING DATA

The Figure 6.1.2 explains white box testing which describes the White Box Testing for Name Error in the Splitting of weather data by layering the input and output. Here the error is naming the sequential classifier is not defined.

ERROR CODE:

`from keras.models import Sequential`

`from keras.layers import Dense`

```
classifier = Sequential()
```

```
classifier.add(Dense(output_dim = 6, init = 'uniform', activation = 'relu', input_dim = 18))
```

```
classifier.add(Dense(output_dim = 6, init = 'uniform', activation = 'relu'))
```

```
classifier.add(Dense(output_dim = 1, init = 'uniform', activation = ' '))
```

NAME ERROR

```
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
```

```
classifier.fit(X_train, y_train, batch_size = 10, nb_epoch = 100)
```

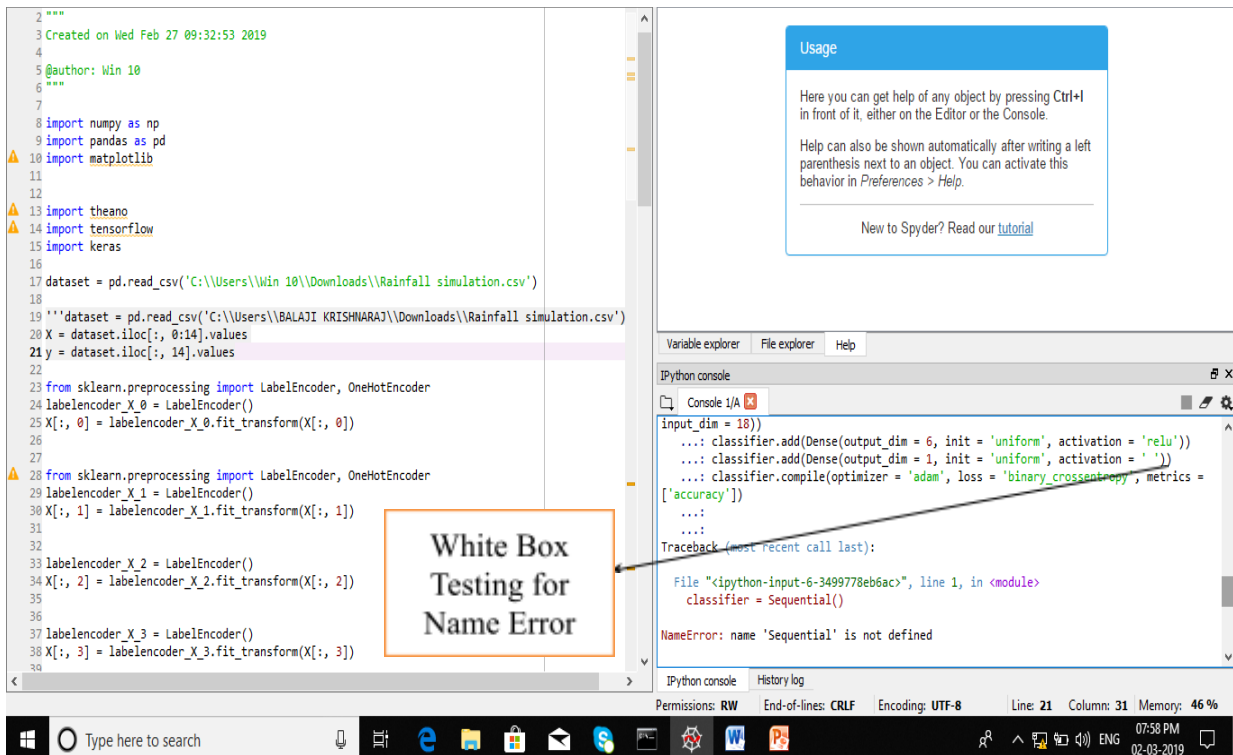


Figure 6.2 White Box Test Case Error Snapshot for Splitting Data

Error Description and Rectified Code:

For Splitting Data the dataset are splited into training data and testing data, the name of the classifier are not defined. So, it's a Naming Error for Splitting Data.

Error Code:

```
classifier = Sequential()
```

```
classifier.add(Dense(output_dim = 6, init = 'uniform', activation = 'relu', input_dim = 18))
```

```
classifier.add(Dense(output_dim = 1, init = 'uniform', activation = ' '))
```

Rectified Code:

```
classifier.add(Dense(output_dim = 6, init = 'uniform', activation = 'relu', input_dim = 18))
```

```
classifier.add(Dense(output_dim = 1, init = 'uniform', activation = 'sigmoid'))
```

6.4 WHITE BOX TESTING FOR PREDICTION AND ACCURACY

Figure 6.1.3 explains white box testing which describes the White Box Testing for Compilation error for finding the emergency department admission prediction and accuracy. Naming index are not defined and it is out of range also database stored values are not match properly.

ERROR CODE:

```
classifier = Sequential()
classifier.add(Dense(output_dim = 6, init = 'uniform', activation = 'relu', input_dim = 18))
classifier.add(Dense(output_dim = 6, init = 'uniform', activation = 'relu'))
classifier.add(Dense(output_dim = 1, init = 'uniform', activation = 'sigmoid'))
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
classifier.fit(X_train, y_train, batch_size = 10, nb_epoch = 100)
y_pred = classifier.predict(X_test)
y_pred = (y_pred > 0.5)
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
cm = confusion_matrix(y_test, y_pred)
print(cm)
print('Accuracy' + str(accuracy_score(y_test, y_pred)))
print(classification_report(y_test, y_pred))
```

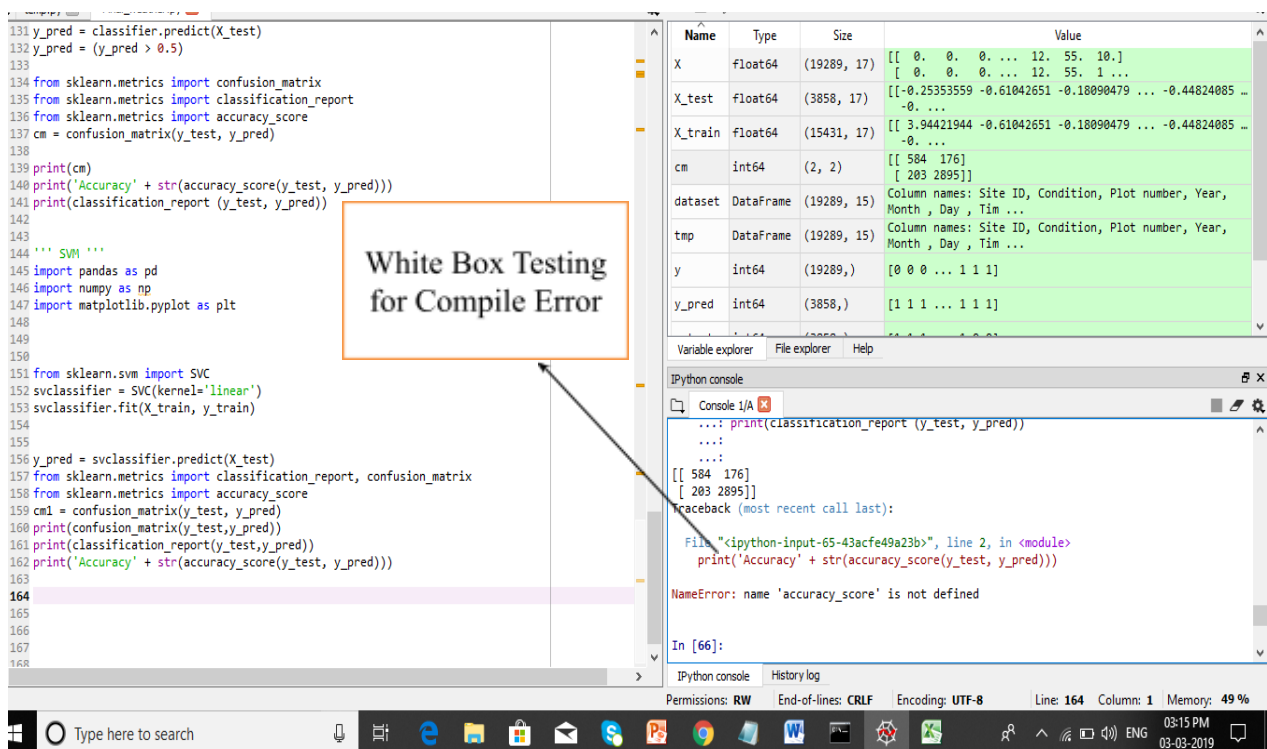


Figure 6.3 White Box Test Case Error Snapshot for Prediction and Accuracy

Error Description and Rectified Code:

In Prediction and Accuracy,
An error while declaring the values compile error occurs.

The values are not initialized here it's an out of range of index.

ERROR CODE:

```
classifier = Sequential()  
classifier.add(Dense(output_dim = 6, init = 'uniform', activation = 'relu', input_dim = 18))
```

CORRECT CODE:

```
classifier = Sequential()  
classifier.add(Dense(output_dim = 6, init = 'uniform', activation = 'relu', input_dim = 17))
```

6.5BLACK BOX TESTING

Block box testing is a table analysis which helps the analyzer to verify the project. It shows the number of attempts taken to resolve an error test cases are built based on the specification and requirement of the project. This project is verified and tested using test cases. These test case are built in the form of table this table includes the test description, input given expected output and the actual output it also tells the status of the test briefing whether it is fail or pass.

S NO	INPUT	TEST DISCRIPTION	EXPECTED OUTPUT	ACTUAL OUTPUT	STATUS
MOD 1	Data Pre-processi ng	If the dataset is given for pre-processing without cleaning.	Unwanted data are in dataset accuracy prediction is difficult.	Accuracy prediction is difficult due to unwanted data.	Pass
MOD 2	Data Visualizatio n and descriptive statistics	Converting the layers from Float or Double to String.	Visualization is converted to String the expected graphs are generated.	While the String is converted the graph generation occurs.	Pass
MOD 3	Data splitting	Split the data for training and Testing process.	Data are splitted and the system were feed the input and	Data were not splitted.	Fail

			output corrected.		
MOD 4	Predicting admissions	Using DNN and SVM classification of data occurs.	Epoch will occur until the count and comparison of two algorithm occurs.	Comparison of two algorithm and epoch occur until the count.	Pass
MOD 5	The evaluation of model performance	Comparison occurs and accuracy prediction of weather analyzed.	Confusion matrix generated and accuracy predicted.	Confusion matrix generated and accuracy predicted.	Pass

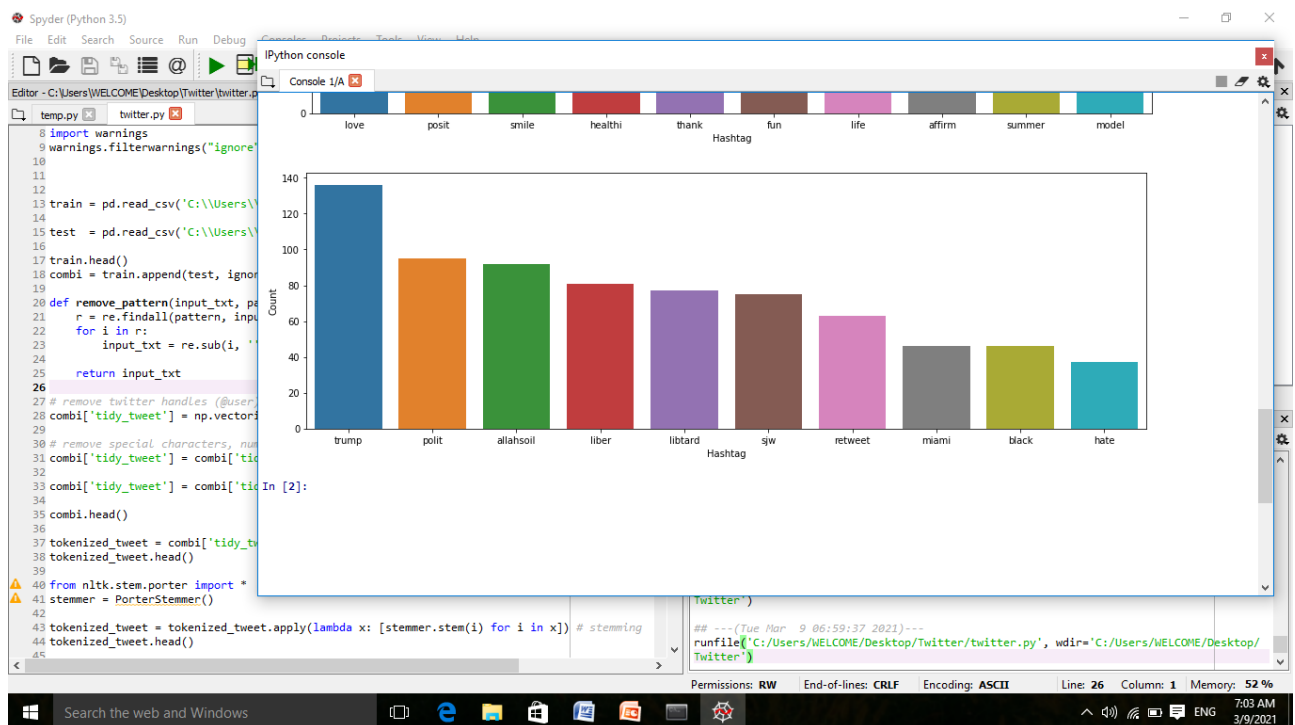
Table 6.4 BLACK BOX TESTING FOR ALL MODULES

1	Site ID	Condition	Plot num	Year	Month	Day	Time	Precipitat	Run-on	Runoff	Sediment	Sediment	Velocity n	Velocity n	Run
2	ER2	N	1	2013	7	30	0	73.66	0	0	NA	NA	NA	NA	0
3	ER2	N	1	2013	7	30	6.33	73.66	0	0	NA	NA	NA	NA	0
4	ER2	N	1	2013	7	30	7	73.66	0	4.86	0.42	0.06	NA	NA	0
5	ER2	N	1	2013	7	30	9	73.66	0	8.35	0.54	0.15	NA	NA	0
6	ER2	N	1	2013	7	30	11	73.66	0	10.25	0.33	0.11	NA	NA	0
7	ER2	N	1	2013	7	30	15	73.66	0	13.68	0.4	0.18	NA	NA	0
8	ER2	N	1	2013	7	30	20	73.66	0	13.59	0.33	0.15	NA	NA	0
9	ER2	N	1	2013	7	30	25	73.66	0	13.59	0.26	0.12	NA	NA	0
10	ER2	N	1	2013	7	30	35	73.66	0	9.28	0.11	0.03	NA	NA	0
11	ER2	N	1	2013	7	30	40	73.66	0	9.44	0.28	0.08	NA	NA	0
12	ER2	N	1	2013	7	30	45	0	0	9.44	0.15	0.05	NA	NA	0
13	ER2	N	1	2013	7	30	45.67	0	0	3.9	0.08	0.01	NA	NA	0
14	ER2	N	1	2013	7	30	46.33	0	0	0	NA	NA	NA	NA	0
15	ER2	N	1	2013	7	30	0	73.66	0	0	NA	NA	NA	NA	1
16	ER2	N	1	2013	7	30	4.58	73.66	0	0	NA	NA	NA	NA	1
17	ER2	N	1	2013	7	30	6	73.66	0	5.91	0.38	0.07	NA	NA	1
18	ER2	N	1	2013	7	30	8	73.66	0	14.25	0.23	0.11	NA	NA	1
19	ER2	N	1	2013	7	30	9	73.66	0	15.32	NA	NA	NA	NA	1
20	ER2	N	1	2013	7	30	10	73.66	0	16.62	0.2	0.11	0.083	0.014	1
21	ER2	N	1	2013	7	30	13	73.66	0	16.62	0.15	0.08	NA	NA	1
22	ER2	N	1	2013	7	30	14	73.66	0	17.76	0.16	0.1	NA	NA	1
23	ER2	N	1	2013	7	30	17	93.72	0	19.59	0.15	0.09	NA	NA	1
24	ER2	N	1	2013	7	30	19	93.72	0	42.59	0.22	0.32	NA	NA	1
25	ER2	N	1	2013	7	30	21	93.72	0	49.83	0.18	0.3	NA	NA	1

Fig 6.5 SCREENSHOT FOR MODULE 1

19266	ER2		4	2010	8	25	20	93.72	rate0	23.4	7.61	7.61	1.508	0.016
19267	ER2		4	2010	8	25	20	93.72	rate0	23.4	7.61	7.61	1.508	0.016
19268	ER2		4	2010	8	25	20	93.72	rate0	23.4	7.61	7.61	1.508	0.016
19269	ER2		4	2010	8	25	20	93.72	rate0	23.4	7.61	7.61	1.508	0.016
19270	ER2		4	2010	8	25	20	93.72	rate0	23.4	7.61	7.61	1.508	0.016
19271	ER2		4	2010	8	25	20	93.72	rate0	23.4	7.61	7.61	1.508	0.016
19272	ER2		4	2010	8	25	20	93.72	rate0	23.4	7.61	7.61	1.508	0.016
19273	ER2		4	2010	8	25	20	93.72	rate0	23.4	7.61	7.61	1.508	0.016
19274	ER2		4	2010	8	25	20	93.72	rate0	23.4	7.61	7.61	1.508	0.016
19275	ER2		4	2010	8	25	20	93.72	rate0	23.4	7.61	7.61	1.508	0.016
19276	ER2		4	2010	8	25	20	93.72	rate0	23.4	7.61	7.61	1.508	0.016
19277	ER2		4	2010	8	25	20	93.72	rate0	23.4	7.61	7.61	1.508	0.016
19278	ER2		4	2010	8	25	20	93.72	rate0	23.4	7.61	7.61	1.508	0.016
19279	ER2		4	2010	8	25	20	93.72	rate0	23.4	7.61	7.61	1.508	0.016
19280	ER2		4	2010	8	25	20	93.72	rate0	23.4	7.61	7.61	1.508	0.016
19281	ER2		4	2010	8	25	20	93.72	rate0	23.4	7.61	7.61	1.508	0.016
19282	ER2		4	2010	8	25	20	93.72	rate0	23.4	7.61	7.61	1.508	0.016
19283	ER2		4	2010	8	25	20	93.72	rate0	23.4	7.61	7.61	1.508	0.016
19284	ER2		4	2010	8	25	20	93.72	rate0	23.4	7.61	7.61	1.508	0.016
19285	ER2		4	2010	8	25	20	93.72	rate0	23.4	7.61	7.61	1.508	0.016
19286	ER2		4	2010	8	25	20	93.72	rate0	23.4	7.61	7.61	1.508	0.016
19287	ER2		4	2010	8	25	20	93.72	rate0	23.4	7.61	7.61	1.508	0.016
19288	ER2		4	2010	8	25	20	93.72	rate0	23.4	7.61	7.61	1.508	0.016
19289	ER2		4	2010	8	25	20	93.72	rate0	23.4	7.61	7.61	1.508	0.016
19290	ER2		4	2010	8	25	20	93.72	rate0	23.4	7.61	7.61	1.508	0.016

Fig 6.6 SCREENSHOT FOR MODULE 1



Output



OutPut

CHAPTER 7

SYSTEM REQUIREMENT

The system requirement is a main part in the analyzing phase of the project. The analyzer of the project has to properly analyze the hardware and the software requirements, otherwise in future the project designer will face more trouble with the hardware and software required. Below specified are the project hardware and software requirements.

HARWARE 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 the software engineers as the starting point for the system design. It shows what the system does and not how it should be implemented.

- SYSTEM : PENTIUM DUAL CORE
- HARD DISK : 120 GB
- MONITOR : 15” LCD
- INPUT DEVICE : KEYBOARD, MOUSE
- RAM : 8 GB

SOFTWARE REQUIREMENTS

The software requirements document is the software specification of the system. It should include both a definition and a specification of a 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 team's progress throughout the development activity.

- OPERATING SYSTEM : WINDOWS 10
- PROGRAMMING LANGUAGE : PYTHON
- TOOLS : ANACONDA, SPYDER, PYTHON
- DATABASE : KERAS LIBRARY

CHAPTER 8 APPLICATIONS

8.1 VARIOUS APPLICATIONS

- **Adjust marketing strategy**

Most companies, if not all, are active in social media, and use the public forum to promote their brands and services. But how can you know if you are doing the right things in social media? Are you randomly posting promotions or do you have strategic plan that you follow? When and how do you know if you should change what you do in social media?

From the managerial perspective, social media is not just a platform where you can post and promote your services. It is a place where your customers chitchat about your brand and is full of information about how brand is being perceived by your target customers. The information you get from sentiment analysis provides you with means to optimize your marketing strategy. By listening to what your customers feel and think about your brand, you can adjust your high-level messaging to meet their needs.

From the tactical point of view, you can build a short-term marketing campaign to provide customers with what they want. By continuously having sentiment analysis in place, you can adjust your campaign to fit even more to your target audience. Hours of work typically goes into preparing marketing campaigns. Why ignore your audience's sentiment when it will help you improve your campaign?

- **Improve customer service**

Once your customer purchases your product, you want to keep them loyal to your brand as long as possible, and be an evangelist for your brand. Your customers can essentially become micro-influencers for you. That is why it is of utmost importance to have the best customer service in place and keep your current customers happy.

There are many factors that contribute to great customer service, such as on-time delivery, being responsive in social media, and adequate compensation for product's errors. Sentiment analysis can pick up negative discussions, and give you real-time alerts so that you can respond quickly. If customers complain about something

related to your brand, the faster you react, the more likely customers will forget being annoyed in the first place, and be satisfied with the great customer service. Sentiment analysis as part of social listening to manage complaints helps you avoid leaving your customers feeling ignored and angry.

For example, Finnair is trying to actively to reply to customers' complaints to quickly solve their issues. Finnair's Twitter page shows quick response to customer's tweets.

- **Leadgeneration**

Your sales colleagues will definitely love this benefit; by having accurate sentiment analysis in place and because of adjusting your marketing campaigns, having great customer service, and improving your product quality to meet the needs of the market, you will be able to increase leads. Loyal and happy customers, who will act as your brand ambassadors, will bring you new customers.

In addition, since sentiment analysis will tell you what your audience wants, needs, and feels about something, you can grasp better ideas to create content that can attract new customers to you.

- **Online Commerce**

The most general use of sentiment analysis is in ecommerce activities. Websites allows their users to submit their experience about shopping and product qualities. They provide summary for the product and different features of the product by assigning ratings or scores. Customers can easily view opinions and recommendation information on whole product as well as specific product features. Graphical summary of the overall product and its features is presented to users. Popular merchant websites like amazon.com provides review from editors and also from customers with rating information. <http://tripadvisor.in> is a popular website that provides reviews on hotels, travel destinations. They contain 75 millions opinions and reviews worldwide. Sentiment analysis helps such websites by converting dissatisfied customers into promoters by analyzing this huge volume of opinions

CHAPTER 9

CONCLUSION AND FUTURE ENHANCEMENTS

CONCLUSION

Social media is a crucial part of the world and it will only keep on building its influence in the future. Analysis of this humongous datasets can improve many networks and industries. We have used one way of analysis called sentiment analysis; there are many ways to improve its accuracy by deploying large datasets considering the emoticons and internationalization. Many futuristic compressor and filter algorithms will play a major role in bridging the language barrier and in handling noisy content and ill-formed words. Sentiment analysis of Twitter using big data helped us to analyse huge amount of datasets.

FUTURE ENHANCEMENTS

Apache Spark is a fast and general-purpose cluster computing system. It provides high-level APIs in Java, Scala, Python and R, and an optimized engine that supports general execution graphs. It also supports a rich set of higher-level tools including Spark SQL for SQL and structured data processing, MLlib for machine learning, GraphX for graph processing, and Spark Streaming. In future, Spark tool can be used to improve the performance as Spark is 100 times faster when compared to MapReduce.

One more feature we that is worth exploring is whether the information about relative position of word in a tweet has any effect on the performance of the classifier. Although Pang et al. explored a similar feature and reported negative results, their results were based on reviews that are very different from tweets and they worked on an extremely simple model.

In this research, we are focussing on general sentiment analysis. There is potential of work in the field of sentiment analysis with partially known context. For example we noticed that users generally use our website for specific types of keywords which can divided into a couple of distinct classes, namely: politics/politicians, celebrities, products/brands, sports/sportsmen, media/movies/music. So we can attempt to perform separate sentiment analysis on tweets that only belong to one of these classes (i.e. the training data would not be

general but specific to one of these categories) and compare the results we get if we apply general sentiment analysis on it instead.

Last but not the least, we can attempt to model human confidence in our system. For example if we have 5 human labellers labelling each tweet, we can plot the tweet in the 2- dimensional objectivity / subjectivity and positivity / negativity plane while differentiating between tweets in which all 5 labels agree, only 4 agree, only 3 agree or no majority vote is reached. We could develop our custom cost function for coming up with optimized class boundaries such that highest weightage is given to those tweets in which all 5 labels agree and as the number of agreements start decreasing, so do the weights assigned. In this way the effects of human confidence can be visualized in sentiment analysis.

APPENDICES

APPENDIX 1:

```
import re
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import nltk
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)

train = pd.read_csv('C:\\Users\\WELCOME\\Desktop\\Twitter\\train_E6oV3IV.csv')

test = pd.read_csv('C:\\Users\\WELCOME\\Desktop\\Twitter\\test_tweets_anuFYb8.csv')

train.head()
combi = train.append(test, ignore_index=True)

def remove_pattern(input_txt, pattern):
    r = re.findall(pattern, input_txt)
    for i in r:
        input_txt = re.sub(i, "", input_txt)

    return input_txt

# remove twitter handles (@user)
combi['tidy_tweet'] = np.vectorize(remove_pattern)(combi['tweet'], "@[\w]*")

# remove special characters, numbers, punctuations
combi['tidy_tweet'] = combi['tidy_tweet'].str.replace("[^a-zA-Z#]", " ")
```

```
combi['tidy_tweet'] = combi['tidy_tweet'].apply(lambda x: ' '.join([w for w in x.split() if len(w)>3]))
```

```
combi.head()
```

```
tokenized_tweet = combi['tidy_tweet'].apply(lambda x: x.split())
```

```
tokenized_tweet.head()
```

```
from nltk.stem.porter import *
```

```
stemmer = PorterStemmer()
```

```
tokenized_tweet = tokenized_tweet.apply(lambda x: [stemmer.stem(i) for i in x]) # stemming
```

```
tokenized_tweet.head()
```

```
for i in range(len(tokenized_tweet)):
```

```
    tokenized_tweet[i] = ' '.join(tokenized_tweet[i])
```

```
combi['tidy_tweet'] = tokenized_tweet
```

```
all_words = ' '.join([text for text in combi['tidy_tweet']])
```

```
from wordcloud import WordCloud
```

```
wordcloud = WordCloud(width=800, height=500, random_state=21,  
max_font_size=110).generate(all_words)
```

```
plt.figure(figsize=(10, 7))
```

```
plt.imshow(wordcloud, interpolation="bilinear")
```

```
plt.axis('off')
```

```
plt.show()
```

```
normal_words = ' '.join([text for text in combi['tidy_tweet'][combi['label'] == 0]])
```

```
wordcloud = WordCloud(width=800, height=500, random_state=21,  
max_font_size=110).generate(normal_words)
```

```
plt.figure(figsize=(10, 7))
```

```
plt.imshow(wordcloud, interpolation="bilinear")
```

```
plt.axis('off')
```

```
plt.show()
```

```

negative_words = ' '.join([text for text in combi['tidy_tweet'][combi['label'] == 1]])
wordcloud = WordCloud(width=800, height=500,
random_state=21, max_font_size=110).generate(negative_words)
plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.show()

```

function to collect hashtags

```

def hashtag_extract(x):
    hashtags = []
    # Loop over the words in the tweet
    for i in x:
        ht = re.findall(r"#(\w+)", i)
        hashtags.append(ht)

    return hashtags

```

extracting hashtags from non racist/sexist tweets

```

HT_regular = hashtag_extract(combi['tidy_tweet'][combi['label'] == 0])

```

extracting hashtags from racist/sexist tweets

```

HT_negative = hashtag_extract(combi['tidy_tweet'][combi['label'] == 1])

```

unnesting list

```

HT_regular = sum(HT_regular,[])
HT_negative = sum(HT_negative,[])

```

```

a = nltk.FreqDist(HT_regular)
d = pd.DataFrame({'Hashtag': list(a.keys()),
                  'Count': list(a.values())})

```

selecting top 10 most frequent hashtags

```

d = d.nlargest(columns="Count", n = 10)
plt.figure(figsize=(16,5))

```

```
ax = sns.barplot(data=d, x= "Hashtag", y = "Count")
ax.set(ylabel = 'Count')
plt.show()
```

```
b = nltk.FreqDist(HT_negative)
e = pd.DataFrame({'Hashtag': list(b.keys()), 'Count': list(b.values())})
# selecting top 10 most frequent hashtags
e = e.nlargest(columns="Count", n = 10)
plt.figure(figsize=(16,5))
ax = sns.barplot(data=e, x= "Hashtag", y = "Count")
ax.set(ylabel = 'Count')
plt.show()
```

```
from sklearn.feature_extraction.text import CountVectorizer
bow_vectorizer = CountVectorizer(max_df=0.90, min_df=2, max_features=1000,
stop_words='english')
# bag-of-words feature matrix
bow = bow_vectorizer.fit_transform(combi['tidy_tweet'])
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer(max_df=0.90, min_df=2, max_features=1000,
stop_words='english')
# TF-IDF feature matrix
tfidf = tfidf_vectorizer.fit_transform(combi['tidy_tweet'])
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
```

```
train_bow = bow[:31962,:]
test_bow = bow[31962,:]
```

```
# splitting data into training and validation set
xtrain_bow, xvalid_bow, ytrain, yvalid = train_test_split(train_bow, train['label'], random_state=42,
```

```
test_size=0.3)
```

```
lreg = LogisticRegression()
```

```
lreg.fit(xtrain_bow, ytrain) # training the model
```

```
prediction = lreg.predict_proba(xvalid_bow) # predicting on the validation set
```

```
prediction_int = prediction[:,1] >= 0.3 # if prediction is greater than or equal to 0.3 then 1 else 0
```

```
prediction_int = prediction_int.astype(np.int)
```

```
f1_score(yvalid, prediction_int) # calculating f1 score
```

```
test_pred = lreg.predict_proba(test_bow)
```

```
test_pred_int = test_pred[:,1] >= 0.3
```

```
test_pred_int = test_pred_int.astype(np.int)
```

```
test['label'] = test_pred_int
```

```
submission = test[['id','label']]
```

```
submission.to_csv('sub_lreg_bow.csv', index=False) # writing data to a CSV file
```

```
train_tfidf = tfidf[:31962,:]
```

```
test_tfidf = tfidf[31962:,:]
```

```
xtrain_tfidf = train_tfidf[ytrain.index]
```

```
xvalid_tfidf = train_tfidf[yvalid.index]
```

```
lreg.fit(xtrain_tfidf, ytrain)
```

```
prediction = lreg.predict_proba(xvalid_tfidf)
```

```
prediction_int = prediction[:,1] >= 0.3
```

```
prediction_int = prediction_int.astype(np.int)
```

```
f1_score(yvalid, prediction_int)
```

REFERENCES

- [1].Monu Kumar, Dr. AnjuBala, 'Analyzing Twitter Sentiments through Big Data' IEEE 2016.
- [2].Kim,M.H.Yang,Y.J.Hwang,S.H.Jeon,K.Y.Kim,I.S.Jung,C.H.Choi, W.S.Cho and J.H. Na, 'Customer Preference Analysis Based on SNS data', 2012 Second International Conference on Cloud and Green Computing ,pp.106-113,2012.
- [3].AnisZarrard, AbdulazizAljaloud, Izzat Alsmadi,³The Evaluation of The Public Opinion',IEEE/ACM 7th International Conference on Utility Cloud Computing,2014
- [4].Ye Wu, Fuji Ren,³Learning Sentimental Influence in Twitter', International Conference on Future Computer Sciences and Application, 2011
- [5].M.SaravananM,D.Sundar and S.Kumaresh, ³Probing of Geospatial Stream Data To Report Disorientation', IEEE Recent Advances in Intelligent Computational Systems(RAICS),2013
- [6].BeimingSun,Vincent TY Ng, ³Analyzing Sentimental influence of Posts on Social Networks', Proceedings of the 2014 IEEE 18th International Conference on Computer Supported Cooperative Work in Design.
- [7].Masahiro Ohmura,KohKakusho, Takeshi Okadome, ³Social Mood Extraction from Twitter posts with Document Topic Model.
- [8].R.Sanjay,'Big Data and Hadoop with components like Flume,Pig,Hive and Jaql',2013.