

**A
Project Report
On
"Twitter Sentiment Analysis and Topic Modelling"**

(CE447 - Software Project Major)

Prepared by
Patel Neel (16CE077)
Patel Pathik (16CE080)
Patel Ravi (16CE085)

Under the Supervision of
Prof. Mrugendra Rahevar

Submitted to
Charotar University of Science & Technology (CHARUSAT)
for the Partial Fulfillment of the Requirements for the
Degree of Bachelor of Technology (B.Tech.)
in U & P U. Patel Department of Computer Engineering (CE)
for B.Tech Semester 8

Submitted at



Accredited with Grade A by NAAC



**U & P U. PATEL DEPARTMENT OF COMPUTER ENGINEERING
Chandubhai S. Patel Institute of Technology (CSPIT)
Faculty of Technology & Engineering (FTE), CHARUSAT
At: Changa, Dist: Anand, Pin: 388421.**

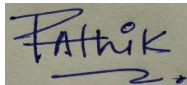
April-May, 2020

DECLARATION BY THE CANDIDATES

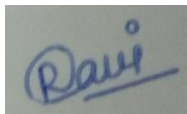
We hereby declare that the project report entitled “**Twitter Sentiment Analysis and Topic Modelling**” submitted by us to Chandubhai S. Patel Institute of Technology, Changa in partial fulfilment of the requirements for the award of the degree of **B.Tech Computer Engineering**, from U & P U. Patel Department of Computer Engineering, CSPIT, FTE, is a record of bonafide CE447 Software Project Major (project work) carried out by us under the guidance of **Prof. Mrugendra Rahevar**. We further declare that the work carried out and documented in this project report has not been submitted anywhere else either in part or in full and it is the original work, for the award of any other degree or diploma in this institute or any other institute or university.



(Patel Neel – 16CE077)

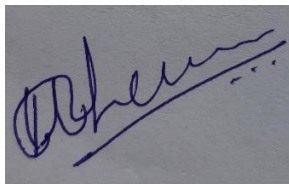


(Patel Pathik – 16CE080)



(Patel Ravi – 16CE085)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.



Prof. Mrugendra Rahevar
Assistant Professor
U & P U. Patel Department of Computer Engineering,
Chandubhai S Patel Institute of Technology (CSPIT)
Faculty of Technology (FTE)
Charotar University of Science and Technology (CHARUSAT) - Changa.



Accredited with Grade A by NAAC

CERTIFICATE

This is to certify that the report entitled “**Twitter Sentiment Analysis and Topic Modelling**” is a bonafied work carried out by **Patel Neel (16CE077), Patel Pathik (16CE080) and Patel Ravi (16CE085)** under the guidance and supervision of **Prof. Mrugendra Rahevar** for the subject **Software Project Major (CE447)** of 8th Semester of Bachelor of Technology in **Computer Engineering** at Chandubhai S. Patel Institute of Technology (CSPIT), Faculty of Technology & Engineering (FTE) – CHARUSAT, Gujarat.

To the best of my knowledge and belief, this work embodies the work of candidates themselves, has duly been completed, and fulfills the requirement of the ordinance relating to the B.Tech. Degree of the University and is up to the standard in respect of content, presentation and language for being referred by the examiner(s).

Under the supervision of,

Prof. Mrugendra Rahevar
Assistant Professor
U & P U. Patel Dept. of Computer
Engineering
CSPIT, FTE, CHARUSAT, Changa, Gujarat

Dr. Ritesh Patel
Head - U & P U. Patel Department of Computer Engineering,
CSPIT, FTE, CHARUSAT, Changa, Gujarat.

Chandubhai S. Patel Institute of Technology (CSPIT)
Faculty of Technology & Engineering (FTE), CHARUSAT

At: Changa, Ta. Petlad, Dist. Anand, Pin: 388421. Gujarat.

ABSTRACT

Twitter Sentiment analysis and Topic Modelling is web application to classify the tweets related to particular topic into positive, negative and neutral tweets. Additionally it also provides most discussed word/topics related to particular event or topic or hashtag.

The project is made using Natural Language Processing (NLP) and Deep Learning techniques. It is developed using Python and Flask, a python framework for developing web application.

The goal of the project is to provide platform for social media monitoring, market research, customer feedback, brand monitoring, customer services and other fields which include social interaction of people.

ACKNOWLEDGEMENT

It is a matter of great pleasure to present this progress report on development of “Twitter Sentiment Analysis and Topic Modelling”. We are grateful to U & P U. Patel Department of Computer Engineering, Chandubhai S. Patel Institute of Technology (CSPIT) for providing us with this great opportunity to develop a web application for the Software Project Major.

We are also grateful to our project guide Prof. Mrugendra Rahevar, for providing understanding on the ways of preparing a project report and for the guidance and support for Software Project Major.

Also, we would like to thank Dr. Ritesh Patel, Head of Department, for providing us all the necessary resources that meets our project requirements.

Table of Contents

ABSTRACT.....	iii
ACKNOWLEDGEMENT	iv
ABBREVIATIONS	ix
CHAPTER 1 INTRODUCTION	1
1.1 PROJECT SUMMARY	1
1.2 PURPOSE	1
1.3 OBJECTIVE.....	2
1.4 SCOPE	2
1.5 TECHNOLOGY AND LITERATURE REVIEW	2
CHAPTER 2 PROJECT MANAGEMENT.....	7
2.1 PROJECT PLANNING.....	7
2.1.1 Project Development Approach and Justification	7
2.1.2 Project Effort and Time Estimation.....	7
2.1.3 Roles and Responsibilities.....	9
2.2 PROJECT SCHEDULING	9
CHAPTER 3 SYSTEM REQUIREMENTS STUDY	10
3.1 USER CHARACTERISTICS	10
3.2 HARDWARE AND SOFTWARE REQUIREMENTS.....	10
3.2.1 Development Environment.....	10
3.2.2 Hosting Environment.....	10
3.3 ASSUMPTIONS AND DEPENDENCIES.....	10
CHAPTER 4 SYSTEM ANALYSIS	11
4.1 STUDY OF CURRENT SYSTEM.....	11
4.2 PROBLEM AND WEAKNESS OF CURRENT SYSTEM.....	11
4.3 REQUIREMENTS OF NEW SYSTEM.....	12
4.3.1 Functional Requirements	12
4.3.2 Non Functional Requirements	12
4.4 FEASIBILITY STUDY	12
4.4.1 Technical Feasibility.....	13
4.4.2 Operational Feasibility	13
4.4.3 Economic Feasibility	13
4.4.4 Schedule Feasibility.....	14
4.5 FEATURES OF NEW SYSTEM.....	14

4.6	SYSTEM DIAGRAM	15
4.7	ACTIVITY DIAGRAM.....	16
4.8	USE CASE DIAGRAM	18
4.9	SEQUENCE DIAGRAM.....	20
4.10	FLOW CHART	22
CHAPTER 5	SYSTEM DESIGN	24
5.1	INPUT/OUTPUT AND INTERFACE DESIGN	24
CHAPTER 6	IMPLEMENTATION PLANNING	27
6.1	IMPLEMENTATION ENVIRONMENT	27
6.2	PROGRAM/MODULES SPECIFICATION	27
6.3	CODING STANDARDS	27
CHAPTER 7	TESTING.....	32
7.1	TESTING PLAN.....	32
7.1.1	Features to be tested	32
7.1.2	Approach	32
7.2	TESTING STRATEGY	32
7.2.1	Unit Testing	32
7.2.2	Bottom-up Integration Testing	32
7.2.3	System Testing	33
7.2.4	Performance Testing.....	33
7.3	TEST SUITE.....	33
CHAPTER 8	CONCLUSION AND DISCUSSION	36
8.1	PROBLEM ENCOUNTERED AND POSSIBLE SOLUTIONS	36
8.2	CONCLUSION OF PROJECT WORK.....	36
CHAPTER 9	LIMITATION AND FUTURE ENHANCEMENT	37
9.1	LIMITATIONS	37
9.2	FUTURE ENHANCEMENT	37
REFERENCES	38

List of Figures

Figure 2.1	Gantt chart	9
Figure 4.1	System design of Twitter Sentiment Analysis.....	15
Figure 4.2	System Design of Twitter Topic Modelling	15
Figure 4.3	Activity Diagram of Twitter Sentiment Analysis.....	16
Figure 4.4	Activity Diagram of Twitter Topic Modelling	17
Figure 4.5	Use Case Diagram of Twitter Sentiment Analysis	18
Figure 4.6	Use Case Diagram of Twitter Topic Modelling	19
Figure 4.7	Sequence Diagram of Twitter Sentiment Analysis	20
Figure 4.8	Sequence Diagram of Twitter Topic modelling	21
Figure 4.9	Flow Chart of Twitter Sentiment Analysis.....	22
Figure 4.10	Flow Chart of Twitter Topic Modelling.....	23
Figure 5.1	Sentiment Analysis Home Screen	24
Figure 5.2	Topic Modelling Home Screen	24
Figure 5.3	Sentiment Analysis Result.....	25
Figure 5.4	Topic Modelling Output	26

List of Tables

Table 2.1	Technical Complexity Factor.....	8
Table 2.2	Unadjusted Function Point.....	8
Table 2.3	Coefficients for COCOMO.....	8
Table 2.4	Roles and Responsibilities	9
Table 7.1	Test case for LSTM model	33
Table 7.2	Test case for Twitter Sentiment Analysis	34
Table 7.3	Test case for Twitter Topic Modelling	35

ABBREVIATIONS

NLP – Natural Language Processing

API – Application Programing Interface

NLTK – Natural Language ToolKit

LSTM – Long Short-Term Memory

LDA – Latent Dirichlet Allocation

SVM – Support Vector Machine

TF-IDF – Term Frequency–Inverse Document Frequency

RNN – Recurrent Neural Network

CHAPTER 1 INTRODUCTION

1.1 PROJECT SUMMARY

This project is a web application which is used to analyse the tweets. We will be performing sentiment analysis of tweets and determining whether it is positive, negative or neutral. Additionally we could also perform topic modelling to know the most discussed topics/words related to particular topic. This web application can be used by any organization office to review their works or by political leaders or by any other company to review about their products or brands.

The main feature of our web application is that it helps to determine the opinion about the peoples on products, government work, politics or any other by analysing the tweets.

The computer data will be represented in various diagrams such as Pie-Chart, Bar graph and word cloud.

1.2 PURPOSE

Sentiment Analysis is the most common text classification tool that analyses an incoming message and tells whether the underlying sentiment is positive, negative or neutral. It is a contextual mining of text which identifies and extracts subjective information from the text and helping a business to understand the social sentiment of their brand, product or service while monitoring online conversations. In today's highly competitive world, it is very difficult to remain in market for any company. This sentiment analysis system is to provide a sentiment of particular text, in terms of positive, neutral, or negative, which can help data analysts within large enterprises gauge public opinion, monitor brand and product reputation, conduct nuanced market research, and understand customer experiences, and based on which, certain action or changes can be taken. Moreover, this project also contains a feature of topic modelling. It is a frequently used text-mining tool for discovery of hidden semantic structures in a text body. This feature is to provide a clusters of abstract "topics" that occur in a collection of documents which can be used by data analyst.

1.3 OBJECTIVE

- Sentiment Analysis to determine the attitude of the people is positive, negative or neutral towards the subject of interest.
- Graphical representation of the sentiment in form of Pie-Chart.
- To implement an algorithm for Topic Modelling of tweets fetched on particular topic.
- Graphical representation of topics in form of various charts.

1.4 SCOPE

- This project can be used in social media monitoring, market research, customer feedback, brand monitoring, customer services and other fields which include social interaction of people.
- It would be helpful to the companies, political parties as well as to the common people.
- It would be helpful to political party for reviewing about the program that they are going to do or the program that they have performed.
- Similarly companies also can get review about their new product or newly released hardware or software.
- Also the movie maker can take review on the currently running movie by analysing tweets.

1.5 TECHNOLOGY AND LITERATURE REVIEW

Flask

Flask is a lightweight WSGI web application framework. It is designed to make getting started quick and easy, with the ability to scale up to complex applications. It began as a simple wrapper around Werkzeug and Jinja and has become one of the most popular Python web application frameworks.

Flask offers suggestions, but doesn't enforce any dependencies or project layout. It is up to the developer to choose the tools and libraries they want to use. There are many extensions provided by the community that make adding new functionality easy [5].

Tensorflow

TensorFlow is an open-sourced library that's available on GitHub. It is one of the more famous libraries when it comes to dealing with Deep Neural Networks. The primary reason behind the

popularity of TensorFlow is the sheer ease of building and deploying applications using TensorFlow [6].

TensorFlow excels at numerical computing, which is critical for deep learning. It provides APIs in most major languages and environments needed for deep learning projects: Python, C, C++, Rust, Haskell, Go, Java, Android, iOS, Mac OS, Windows, Linux, and Raspberry Pi.

Moreover, TensorFlow was created keeping the processing power limitations in mind. Implying, we can run this library on all kinds of computers, irrespective of their processing powers. It can even be run on a smartphone.

Keras

Keras is a high-level library that's built on top of Theano or TensorFlow. It provides a scikit-learn type API (written in Python) for building Neural Networks. Developers can use Keras to quickly build neural networks without worrying about the mathematical aspects of tensor algebra, numerical techniques, and optimization methods [6].

The key idea behind the development of Keras is to facilitate experimentations by fast prototyping. The ability to go from an idea to result with the least possible delay is key to good research.

This offers a huge advantage for scientists and beginner developers alike because they can dive right into Deep Learning without getting their hands dirty with low-level computations. The rise in the demand for Deep Learning has resulted in the rise in demand for people skilled in Deep Learning.

Every organization is trying to incorporate Deep Learning in one way or another, and Keras offers a very easy to use as well as intuitive enough to understand API which essentially helps you test and build Deep Learning applications with least considerable efforts. This is good because Deep Learning research is such a hot topic right now and scientists need a tool to try out their ideas without wasting time on putting together a Neural Network model.

Salient Features of Keras:

- Keras is a high-level interface and uses Theano or Tensorflow for its backend.
- It runs smoothly on both CPU and GPU.
- Keras, being modular in nature, is incredibly expressive, flexible, and apt for innovative research.

- Keras supports almost all the models of a neural network – fully connected, convolutional, pooling, recurrent, embedding, etc. Furthermore, these models can be combined to build more complex models.
- Keras is a completely Python-based framework, which makes it easy to debug and explore.

Gensim

Gensim = “Generate Similar” is a popular open source natural language processing (NLP) library used for unsupervised topic modelling. It uses top academic models and modern statistical machine learning to perform various complex tasks such as –

- Building document or word vectors
- Corpora
- Performing topic identification
- Performing document comparison (retrieving semantically similar documents)
- Analysing plain-text documents for semantic structure

Apart from performing the above complex tasks, Gensim, implemented in Python and Cython, is designed to handle large text collections using data streaming as well as incremental online algorithms. This makes it different from those machine learning software packages that target only in-memory processing.

It is a great package for processing texts, working with word vector models (such as Word2Vec, FastText etc) and for building topic models. Also, another significant advantage with gensim is: it lets you handle large text files without having to load the entire file in memory [7].

Sentiment Analysis

Sentiment analysis is the interpretation and classification of emotions (positive, negative and neutral) within text data using text analysis techniques. Sentiment analysis allows businesses to identify customer sentiment toward products, brands or services in online conversations and feedback.

Sentiment Analysis Basics

Sentiment analysis models detect polarity within a text (e.g. a *positive* or *negative* opinion), whether it's a whole document, paragraph, sentence, or clause.

Understanding people's emotions is essential for businesses since customers are able to express their thoughts and feelings more openly than ever before. By automatically analysing customer feedback, from survey responses to social media conversations, brands are able to listen attentively to their customers, and tailor products and services to meet their needs.

For example, one of our customers used sentiment analysis to automatically analyse 4,000+ reviews about their product, and discovered that customers were happy about their pricing but complained a lot about their customer service:

Benefits of Sentiment Analysis

It's estimated that 80% of the world's data is unstructured, in other words it's unorganized. Huge amounts of text data (emails, support tickets, chats, social media conversations, surveys, articles, documents, etc.), is created every day but it's hard to analyse, understand, and sort through, not to mention time-consuming and expensive [8].

Benefits of sentiment analysis include:

- **Processing Data at Scale** Can you imagine manually sorting through thousands of tweets, customer support conversations, or customer reviews? There's just too much data to process manually. Sentiment analysis helps businesses process huge amounts of data in an efficient and cost-effective way.
- **Real-Time Analysis** Sentiment analysis can identify critical issues in real-time, for example is a PR crisis on social media escalating? Is an angry customer about to churn? Sentiment analysis models can help you immediately identify these kinds of situations, so you can take action right away.
- **Consistent criteria** it's estimated that people only agree around 60-65% of the time when determining the sentiment of a particular text. Tagging text by sentiment is highly subjective, influenced by personal experiences, thoughts, and beliefs. By using a centralized sentiment analysis system, companies can apply the same criteria to all of their data, helping them improve accuracy and gain better insights.

Topic Modelling

Topic modelling is an unsupervised machine learning technique that's capable of scanning a set of documents, detecting word and phrase patterns within them, and automatically clustering word groups and similar expressions that best characterize a set of documents.

Topic modelling is a machine learning technique that automatically analyses text data to determine cluster words for a set of documents. This is known as ‘unsupervised’ machine learning because it doesn’t require a predefined list of tags or training data that’s been previously classified by humans.

Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation (LDA) and LSA are based on the same underlying assumptions: the distributional hypothesis, (i.e. similar topics make use of similar words) and the statistical mixture hypothesis (i.e. documents talk about several topics) for which a statistical distribution can be determined. The purpose of LDA is mapping each document in our corpus to a set of topics which covers a good deal of the words in the document.

What LDA does in order to map the documents to a list of topics is assign topics to arrangements of words, e.g. n-grams such as best player for a topic related to sports. This stems from the assumption that documents are written with arrangements of words and that those arrangements determine topics. Yet again, just like LSA, LDA also ignores syntactic information and treats documents as bags of words. It also assumes that all words in the document can be assigned a probability of belonging to a topic. That said, the goal of LDA is to determine the mixture of topics that a document contains [9].

CHAPTER 2 PROJECT MANAGEMENT

2.1 PROJECT PLANNING

2.1.1 Project Development Approach and Justification

To develop this project, we have chosen Incremental software development model. According to this software development model we have divided our project work into small chunks. In every iteration, we have designed, developed and tested additional features in our program. The purpose of working iteratively is to allow more flexibility for changes. When requirements and design of a major application are done in the traditional method, there can be unforeseen problems that don't surface until development begins. By working iteratively, we can go through a cycle where we evaluate with each iteration, and determine what changes are needed to produce a satisfactory product.

2.1.2 Project Effort and Time Estimation

System General Characteristics	Description	Degree of Influence (0-5)
1. Data communications	How many communication facilities are there to aid in the transfer or exchange of information with the application or system?	5
2. Performance	Did the user require response time or throughput?	5
3. Heavily used configuration	How heavily used is the current hardware platform where the application will be executed?	5
4. Transaction rate	How frequently are transactions executed daily, weekly, monthly, etc.?	0
5. Online data entry	What percentage of the information is entered On-Line?	0
6. End user efficiency	Was the application designed for end-user efficiency?	4
7. Online update	How many ILF's are updated by On-Line transaction?	2
8. Complex processing	Does the application have extensive logical or mathematical processing?	5
9. Reusability	Was the application developed to meet one or many user's needs?	4
10. Installation ease	How difficult is conversion and installation?	4
11. Operations ease	How effective and/or automated are start-up, back up, and recovery procedures?	3

12. Multiple sites	Was the application specifically designed, developed, and supported to be installed at multiple sites for multiple organizations?	0
13. Facilitate Change	Was the application specifically designed, developed, and supported to facilitate change?	0
14. Distributed functions	How are distributed data and processing functions handled?	4
Technical Complexity Factor (TCF)		41

Table 2.1 Technical Complexity Factor

Function Point	Low	Average	High	Total
External Input	$\underline{2} * 3 = 6$	$\underline{0} * 4 = 0$	$\underline{0} * 6 = 0$	6
External Output	$\underline{0} * 4 = 0$	$\underline{1} * 5 = 5$	$\underline{0} * 7 = 0$	5
External Inquiries	$\underline{0} * 3 = 0$	$\underline{2} * 4 = 8$	$\underline{0} * 6 = 0$	8
Internal Logical File	$\underline{0} * 7 = 0$	$\underline{1} * 10 = 10$	$\underline{0} * 15 = 0$	10
External Interface File	$\underline{0} * 5 = 0$	$\underline{0} * 7 = 0$	$\underline{0} * 10 = 0$	0
Unadjusted Function Point (UFP)				29

Table 2.2 Unadjusted Function Point

Technical Complexity Factor (TCF) = 41

Unadjusted Function Point (UFP) = 19

Complexity Adjustment Factor (CAF) = $0.65 + (0.01 * \text{TCF}) = 1.06$

Function point (FP) = $\text{UFP} * \text{CAF} = 1.06 * 29 = 30.74$

Estimation of Lines of Code:

Perl is probably about the closest to Python in spirit and code density. Therefore, Lines of code (LOC) per Function Point (FP) for Python is 50 [1].

Lines of Code (LOC) = $\text{FP} * 67 = 30.74 * 50 = 1537 = \mathbf{1.5 \text{ KLOC}}$

Estimation of Time and Effort using Constructive Cost Model (COCOMO):

Software Project	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semidetached	3.0	1.12	2.5	0.35
Embedded	3.6	1.2	2.5	0.32

Table 2.3 Coefficients for COCOMO

According to Boehm's definition, a software project is said to be an organic type if the team size required is adequately small, the problem is well understood and has been solved in the past and also the team members have a nominal experience regarding the problem [2]. Therefore, we would consider coefficient of organic system for further calculation.

$$\begin{aligned}\text{Effort} &= a * (\text{KLOC})^b \text{ PM (Person per Month)} \\ &= 2.4 * (1.5)^{1.05} \text{ PM} = 3.67 \text{ PM}\end{aligned}$$

$$\begin{aligned}\text{Time of development} &= c * (\text{Effort})^d \text{ Months} \\ &= 2.5 * (3.67)^{0.38} = 4.09 \approx 4 \text{ Months}\end{aligned}$$

2.1.3 Roles and Responsibilities

Patel Neel (16CE077)	Sentiment Analysis, Data Pre-processing
Patel Pathik (16CE080)	Topical Modelling, Data Visualization
Patel Ravi (16CE085)	Topical Modelling, GUI (Flask)

Table 2.4 Roles and Responsibilities

2.2 PROJECT SCHEDULING

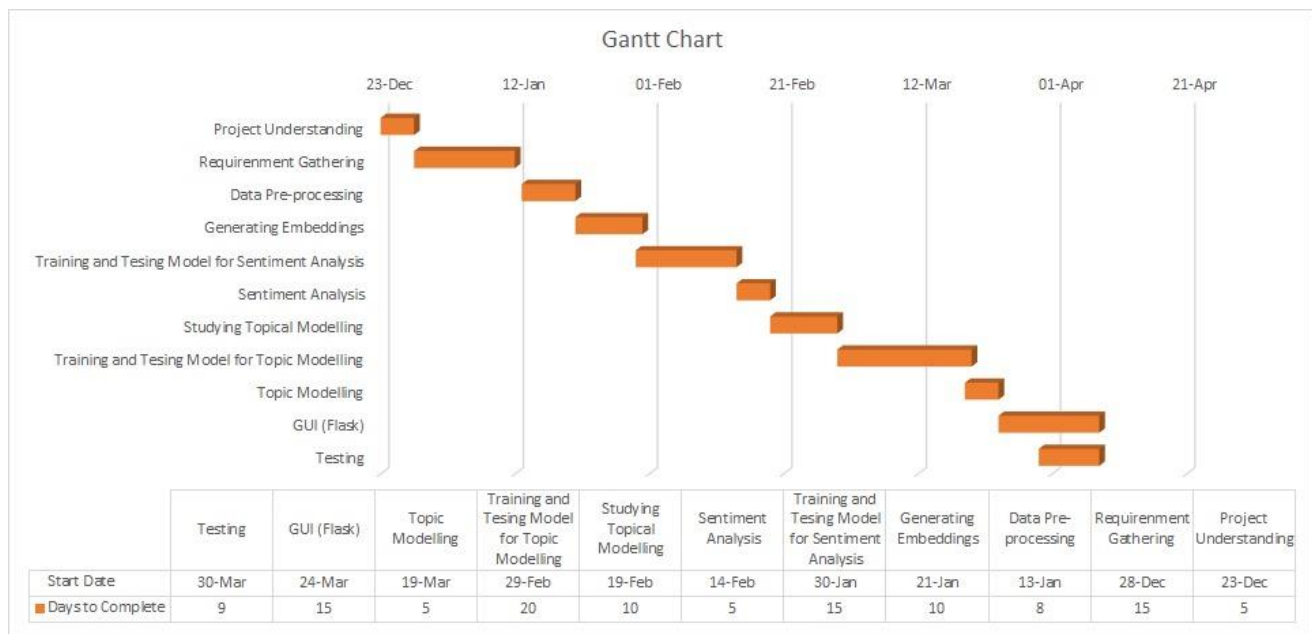


Figure 2.1 Gantt chart

CHAPTER 3 SYSTEM REQUIREMENTS STUDY

3.1 USER CHARACTERISTICS

This system can be used by any companies, political parties as well as to the common people.

3.2 HARDWARE AND SOFTWARE REQUIREMENTS

3.2.1 Development Environment

Hardware Requirement

- RAM : 16 GB
- Processor : Intel i5
- Speed : 2.5 GHz

Software Requirement

- Operating System : Windows 10
- Coding Language : Python
- Frameworks : Bootstrap, Flask
- API : Twitter API
- Libraries : NLTK, Tensorflow, Keras, Gensim, Sklearn

3.2.2 Hosting Environment

Hardware Requirement

- RAM : 8 GB
- Processor : Intel i5
- Speed : 1.8 GHz

Software Requirement

- Operating System : Windows/Linux
- API : Twitter API
- Dependencies : NLTK, Tensorflow, Keras, Gensim, Sklearn

3.3 ASSUMPTIONS AND DEPENDENCIES

- It is assumed that user has twitter developer account's credentials.
- This system is depended on internet so it is assumed that user has internet connection.

CHAPTER 4 SYSTEM ANALYSIS

4.1 STUDY OF CURRENT SYSTEM

Sentiment Analysis, also called opinion mining or emotion AI, is the process of determining whether a piece of writing is positive, negative, or neutral. A common use case for this technology is to discover how people feel about a particular topic. Sentiment analysis is widely applied to reviews and social media for a variety of applications.

Sentiment analysis can be performed in many different ways. Many brands and marketers use keyword-based tools that classify data (i.e. social, news, review, blog, etc.) as positive/negative/neutral.

Automated sentiment tagging is usually achieved through word lists. For example, mentions of 'hate' would be tagged negatively.

There can be two approaches to sentiment analysis.

- Lexicon-based methods
- Machine Learning-based methods.

In current time developers are using tweepy or twython to access Twitter API to fetch tweets. Additionally, they are using Textblob library to carry out sentiment analysis.

TextBlob is a Python (2 and 3) library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more [4].

The *sentiment* function of textblob returns two properties, **polarity**, and **subjectivity**.

Polarity is float which lies in the range of [-1, 1] where 1 means positive statement and -1 means a negative statement. Subjective sentences generally refer to personal opinion, emotion or judgment whereas objective refers to factual information. Subjectivity is also a float which lies in the range of [0, 1] [4].

4.2 PROBLEM AND WEAKNESS OF CURRENT SYSTEM

- slow
- no neural network models
- no integrated word vectors

4.3 REQUIREMENTS OF NEW SYSTEM

4.3.1 Functional Requirements

Functional requirements are the functions or features that must be included in any system to satisfy the business needs and be acceptable to the users. Based on this, the functional requirements that the system must require are as follows:

- System should be able to process live tweets and should be able to analyse data and classify each tweets polarity.
- System should be able to fetch live tweets on the basis of hashtags, Geo-location, key word etc. and should be able to store it in file.
- System should be able to use stored data to perform Topical Modelling on that.

4.3.2 Non Functional Requirements

Non-functional requirements is a description of features, characteristics and attribute of the system as well as any constraints that may limit the boundaries of the proposed system. They are essentially based on the performance, information, economy, control and security efficiency and services. Based on these the non-functional requirements are as follows:

- User friendly and attractive GUI.
- System should provide better accuracy.
- To perform with efficient throughput and response time.

4.4 FEASIBILITY STUDY

A feasibility study is a preliminary study which investigates the information of prospective users and determines the resources requirements, costs, benefits and feasibility of proposed system. A feasibility study takes into account various constraints within which the system should be implemented and operated. In this stage, the resource needed for the implementation such as computing equipment, manpower and costs are estimated. The estimated are compared with available resources and a cost benefit analysis of the system is made. The feasibility analysis activity involves the analysis of the problem and collection of all relevant information relating to the project. The main objectives of the feasibility study are to determine whether the project would be feasible in terms of economic feasibility, technical feasibility and operational feasibility and schedule feasibility or not. It is to make sure that the input data which are required for the project are available. Thus we evaluated the feasibility of the system in terms of the following categories:

- Technical feasibility
- Operational feasibility
- Economic feasibility
- Schedule feasibility

4.4.1 Technical Feasibility

Evaluating the technical feasibility is the trickiest part of a feasibility study. This is because, at the point in time there is no any detailed designed of the system, making it difficult to access issues like performance, costs (on account of the kind of technology to be deployed) etc. A number of issues have to be considered while doing a technical analysis; understand the different technologies involved in the proposed system. Before commencing the project, we have to be very clear about what are the technologies that are to be required for the development of the new system. Is the required technology available? Our system is technically feasible since all the required tools are easily available. Python and HTML with CSS and Javascript can be easily handled. Although all tools seems to be easily available there are challenges too.

4.4.2 Operational Feasibility

Proposed project is beneficial only if it can be turned into information systems that will meet the operating requirements. Simply stated, this test of feasibility asks if the system will work when it is developed and installed. Are there major barriers to Implementation? The proposed was to make a simplified web application. It is simpler to operate and can be used in any webpages. It is free and not costly to operate.

4.4.3 Economic Feasibility

Economic feasibility attempts to weigh the costs of developing and implementing a new system, against the benefits that would accrue from having the new system in place. This feasibility study gives the top management the economic justification for the new system. A simple economic analysis which gives the actual comparison of costs and benefits are much more meaningful in this case. In addition, this proves to be useful point of reference to compare actual costs as the project progresses. There could be various types of intangible benefits on account of automation. These could increase improvement in product quality, better decision making, and timeliness of information, expediting activities, improved accuracy of operations, better documentation and record keeping, faster retrieval of information.

This is a web based application. Creation of application is not costly.

4.4.4 Schedule Feasibility

A project will fail if it takes too long to be completed before it is useful. Typically, this means estimating how long the system will take to develop, and if it can be completed in a given period of time using some methods like payback period. Schedule feasibility is a measure how reasonable the project timetable is. Given our technical expertise, are the project deadlines reasonable? Some project is initiated with specific deadlines. It is necessary to determine whether the deadlines are mandatory or desirable. A minor deviation can be encountered in the original schedule decided at the beginning of the project. The application development is feasible in terms of schedule.

4.5 FEATURES OF NEW SYSTEM

- Sentiment Analysis of tweets using Word Embedding's and Long Short-term Memory (LSTM).
- Topic Modelling of tweets using Word2Vector Model, TF-IDF Model, Linear SVM Model and Latent Dirichlet Allocation (LDA).
- Visualization of results using various charts like Pie-chart, Scatter plot, Word Cloud etc.
- Live sentiment analysis of tweets as well as offline sentiment analysis of tweets using csv file.

4.6 SYSTEM DIAGRAM

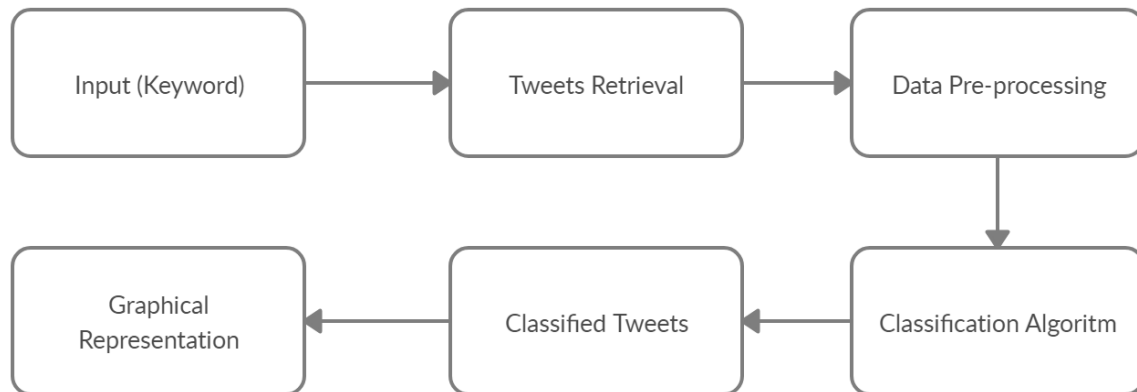


Figure 4.1 System design of Twitter Sentiment Analysis

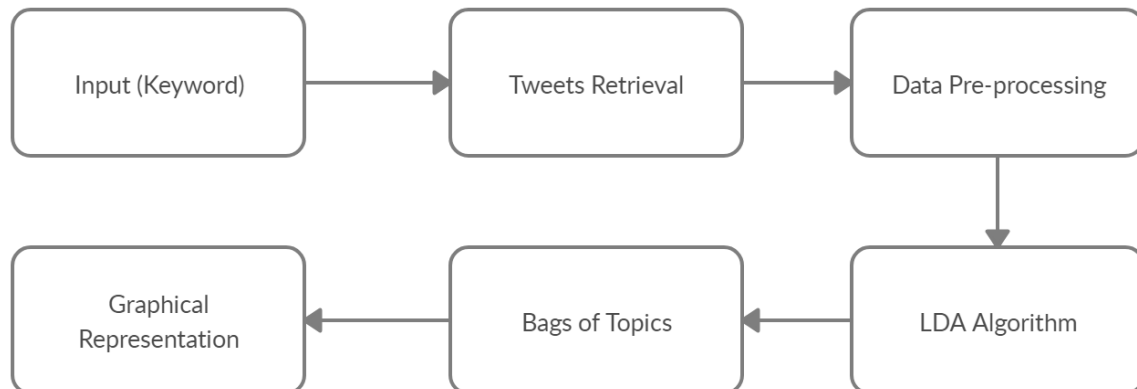


Figure 4.2 System Design of Twitter Topic Modelling

4.7 ACTIVITY DIAGRAM

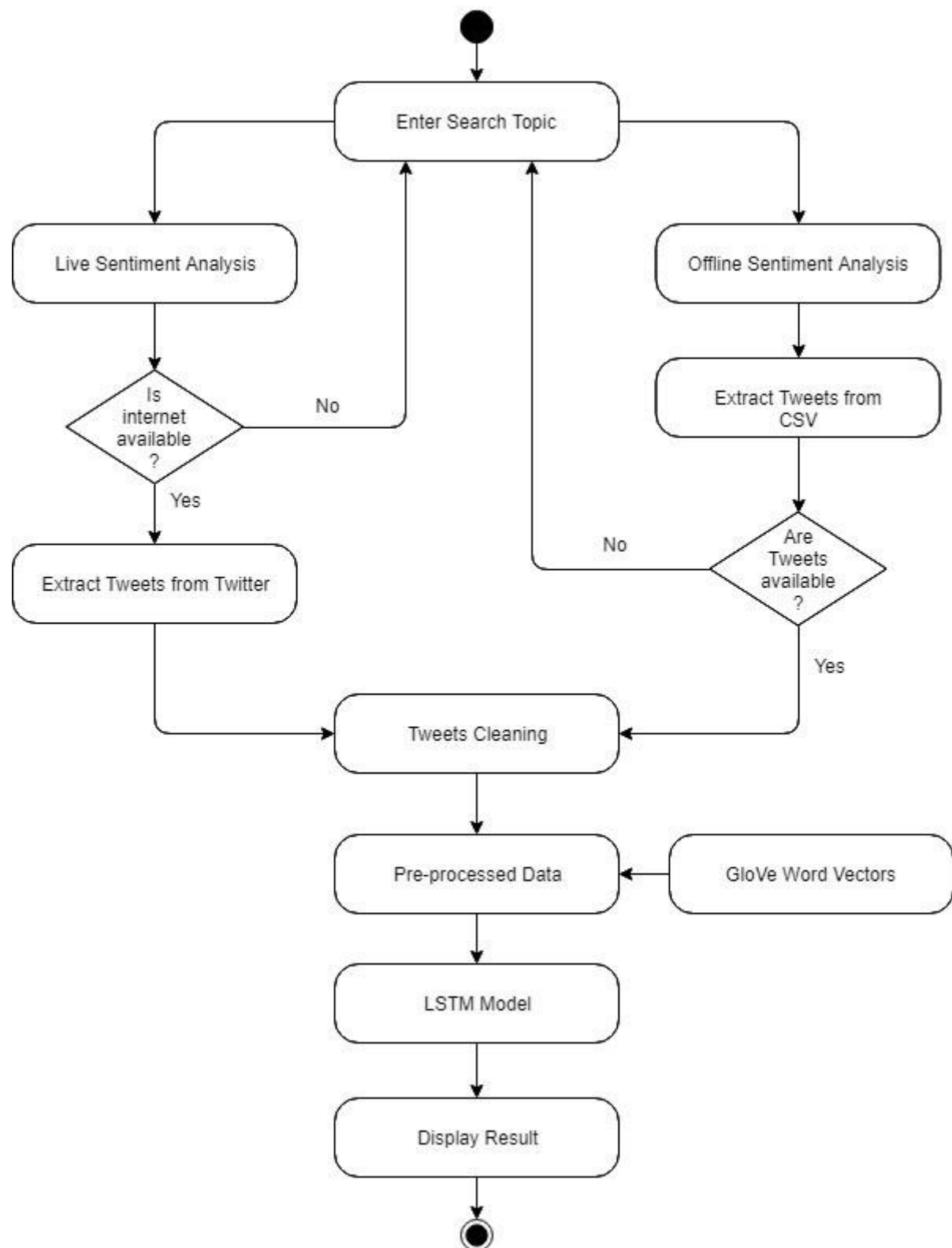


Figure 4.3 Activity Diagram of Twitter Sentiment Analysis

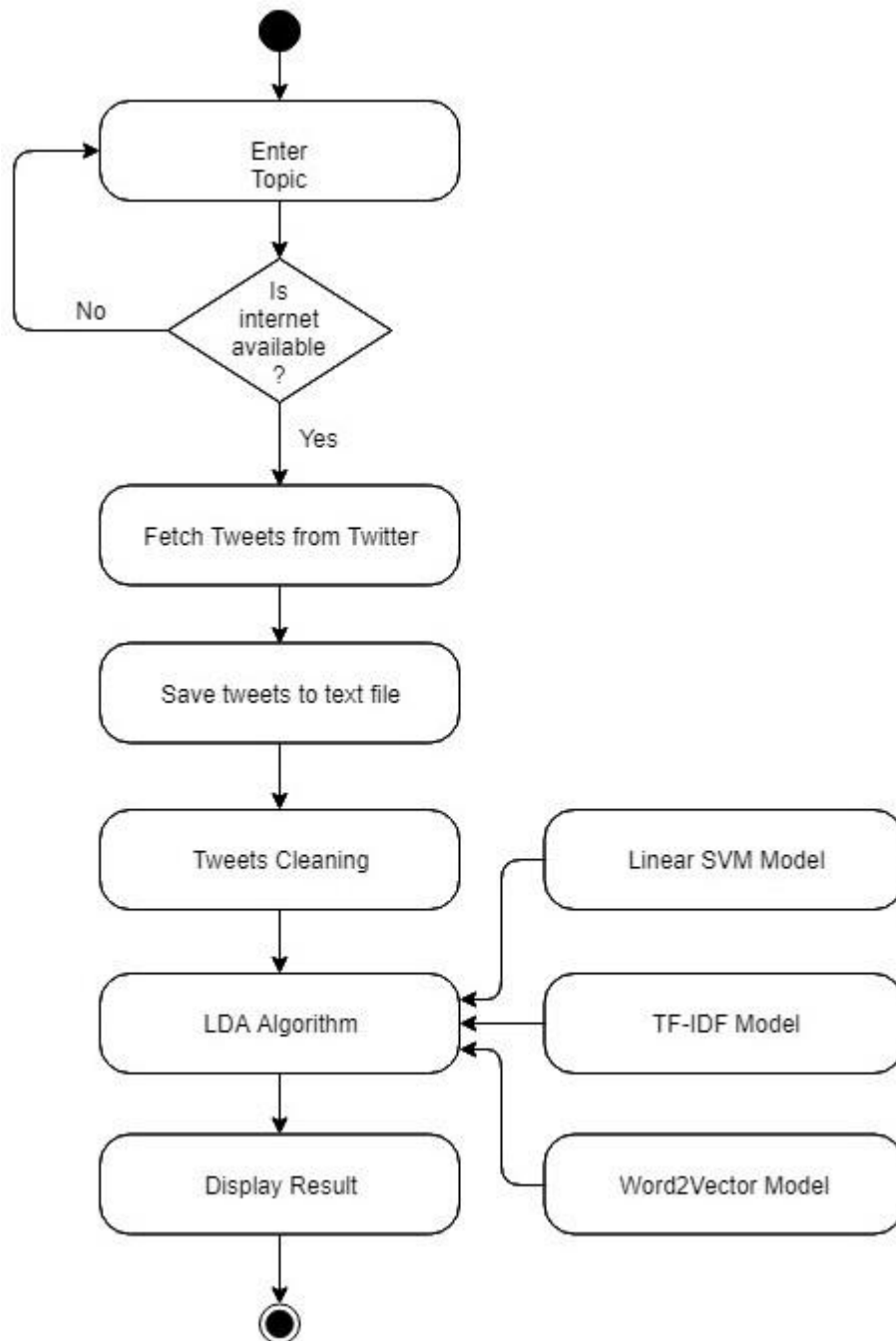


Figure 4.4 Activity Diagram of Twitter Topic Modelling

4.8 USE CASE DIAGRAM

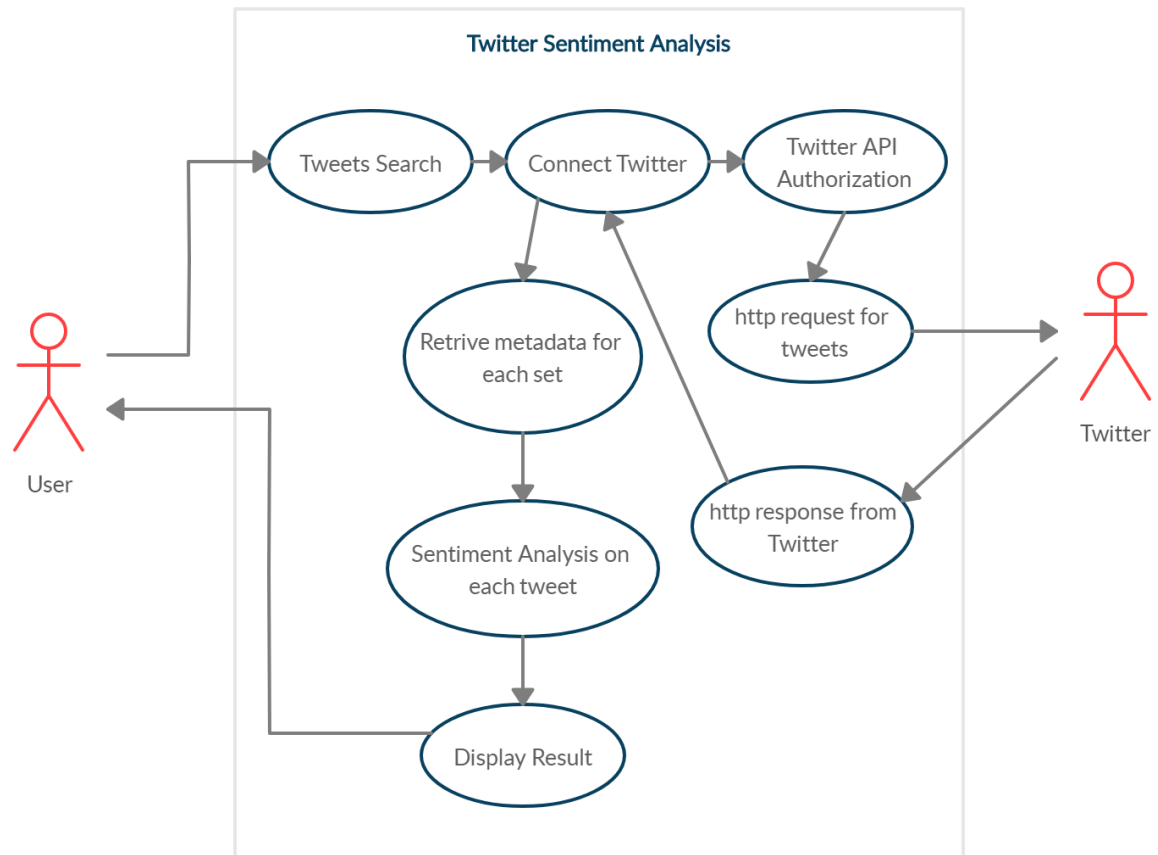


Figure 4.5 Use Case Diagram of Twitter Sentiment Analysis

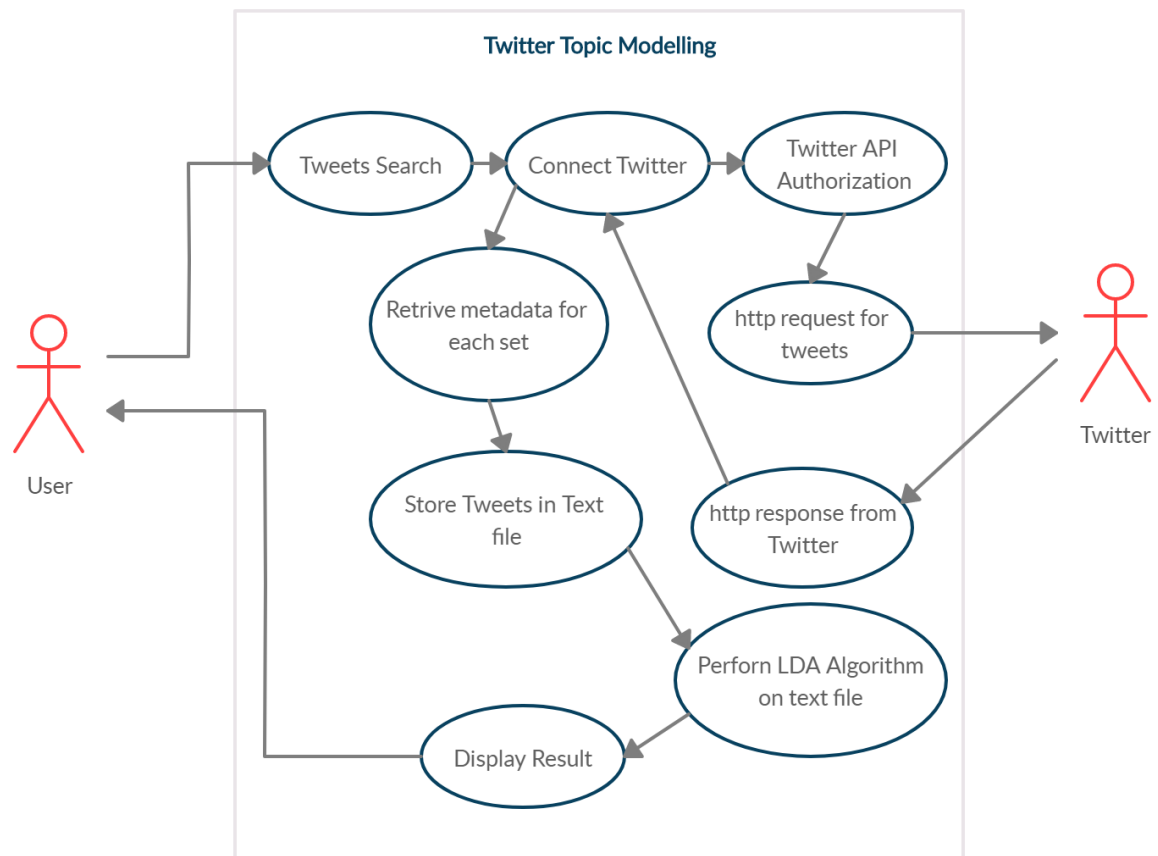


Figure 4.6 Use Case Diagram of Twitter Topic Modelling

4.9 SEQUENCE DIAGRAM

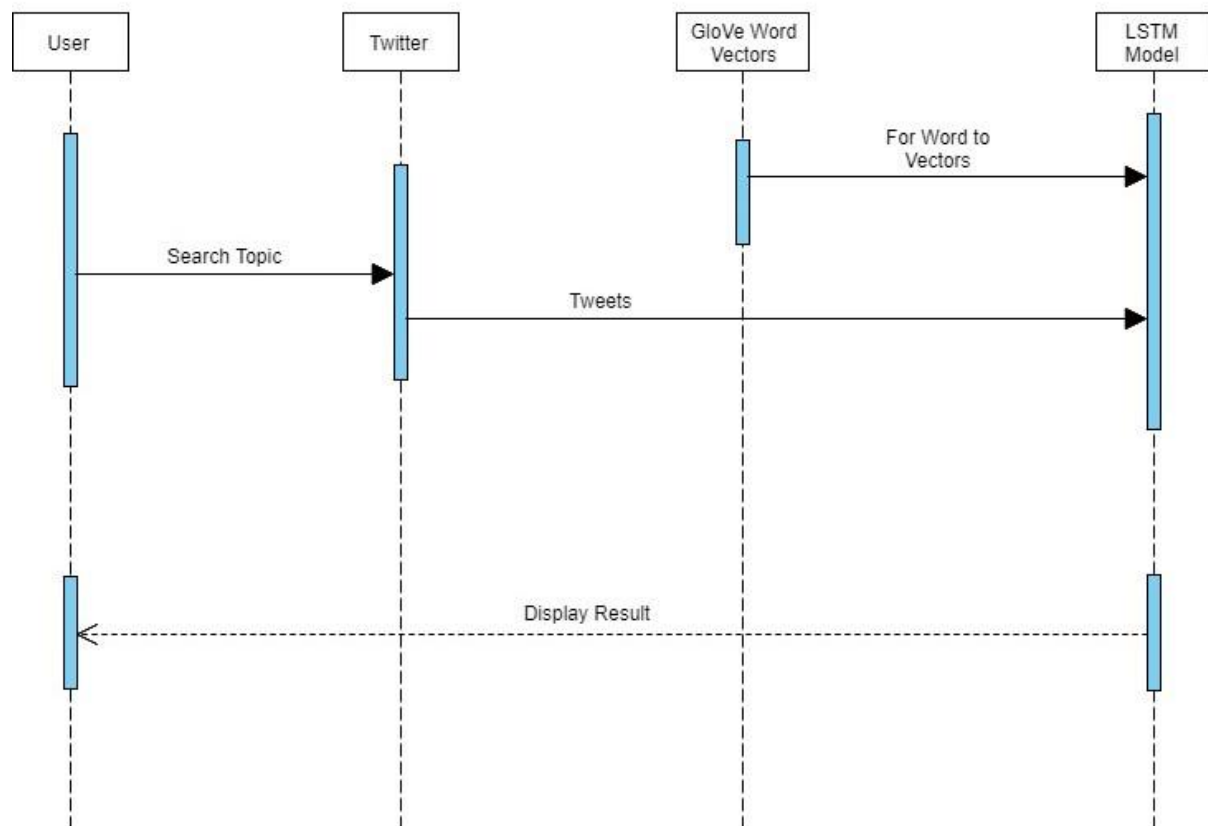


Figure 4.7 Sequence Diagram of Twitter Sentiment Analysis

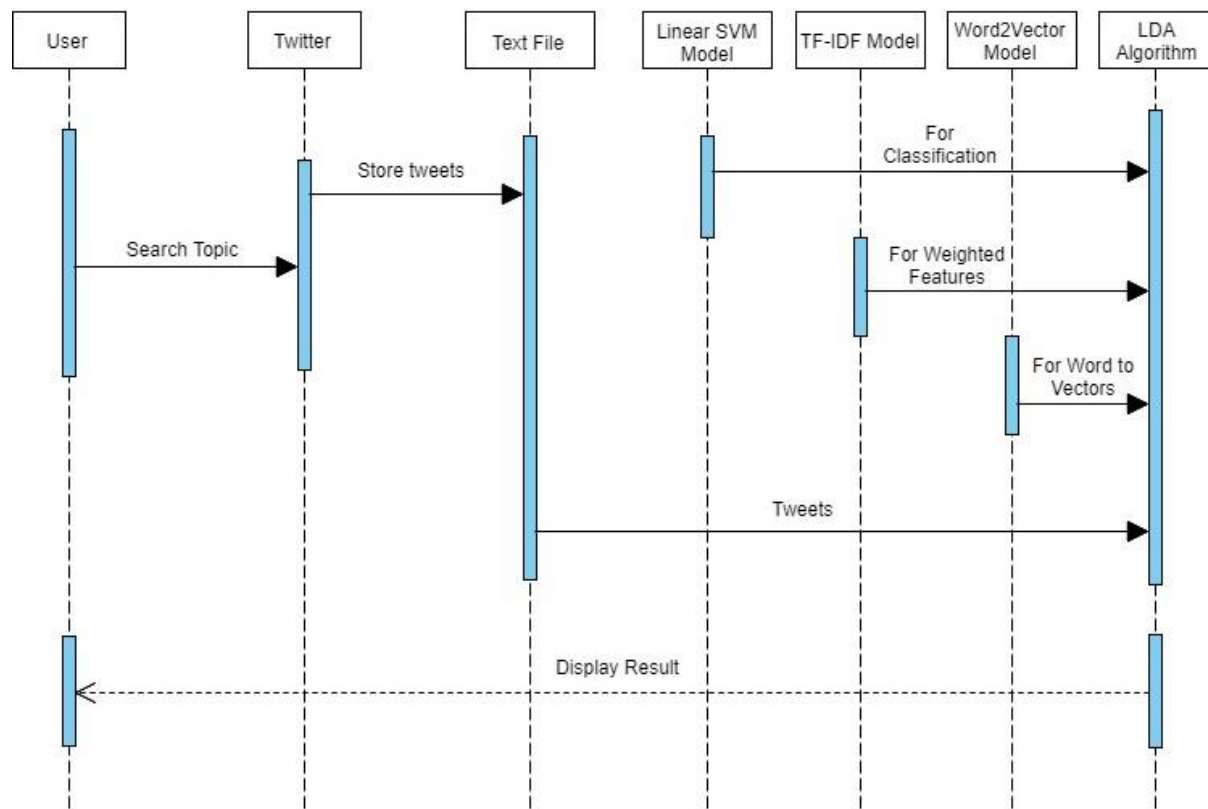


Figure 4.8 Sequence Diagram of Twitter Topic modelling

4.10 FLOW CHART

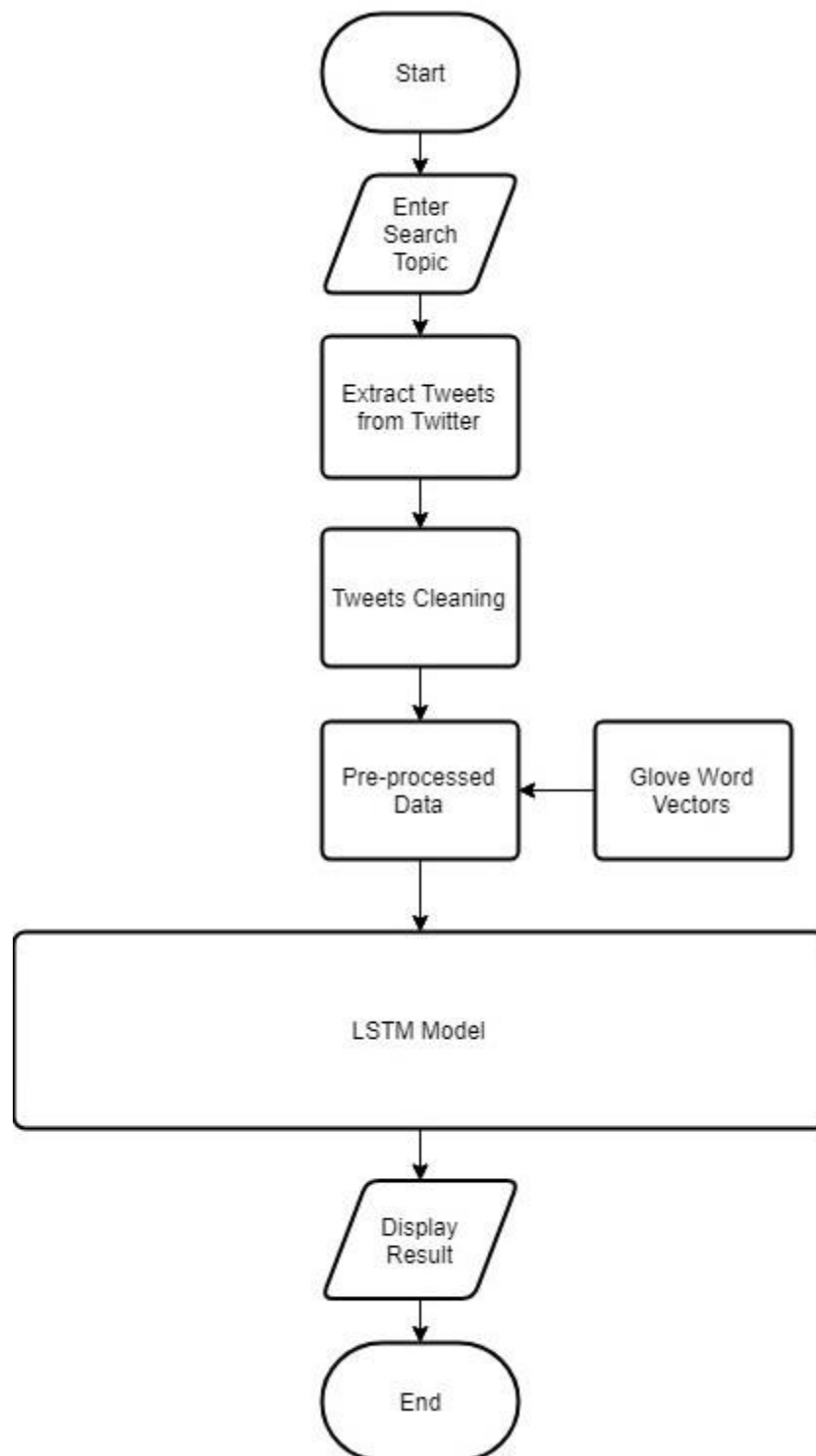


Figure 4.9 Flow Chart of Twitter Sentiment Analysis

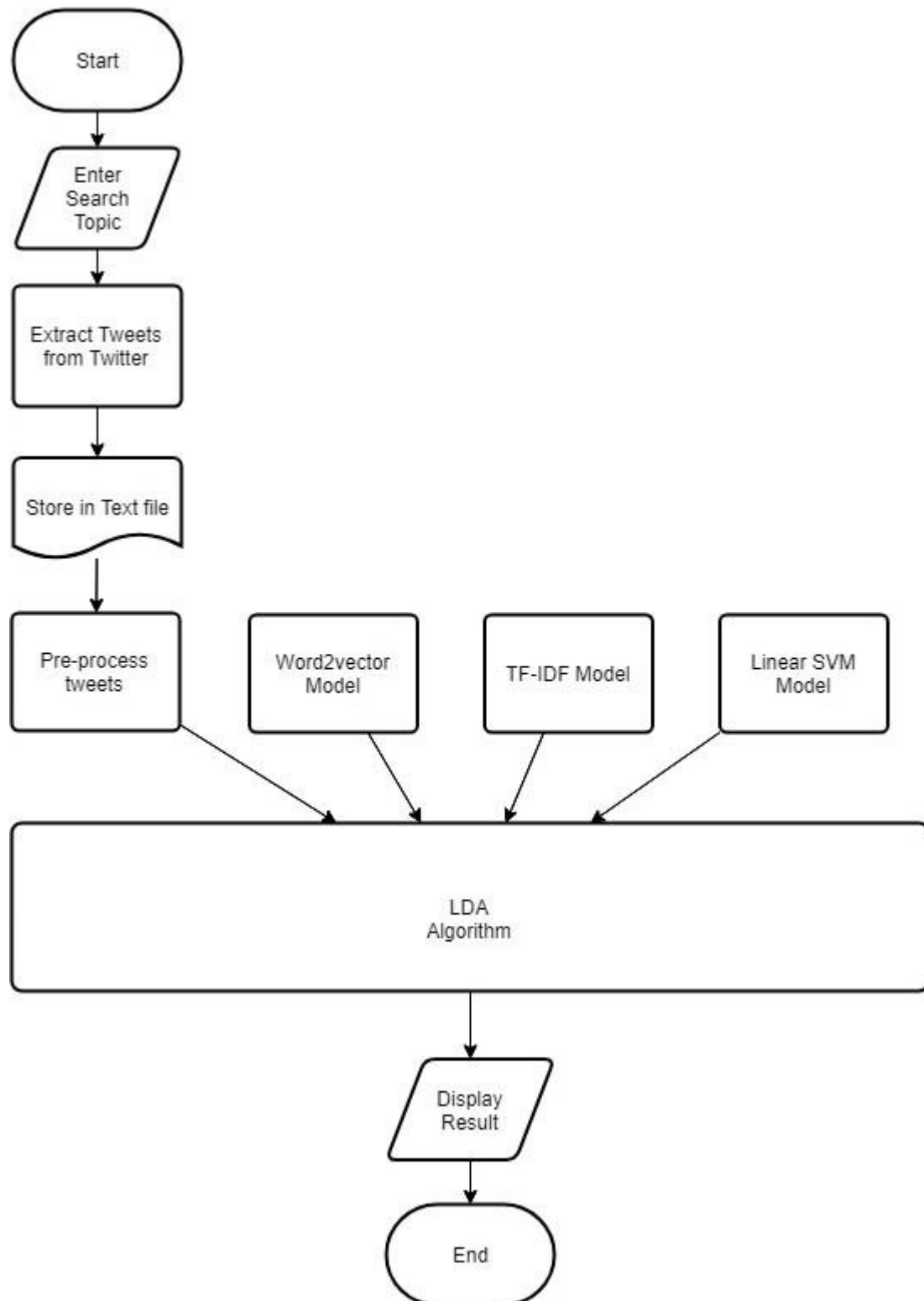


Figure 4.10 Flow Chart of Twitter Topic Modelling

CHAPTER 5 SYSTEM DESIGN

5.1 INPUT/OUTPUT AND INTERFACE DESIGN

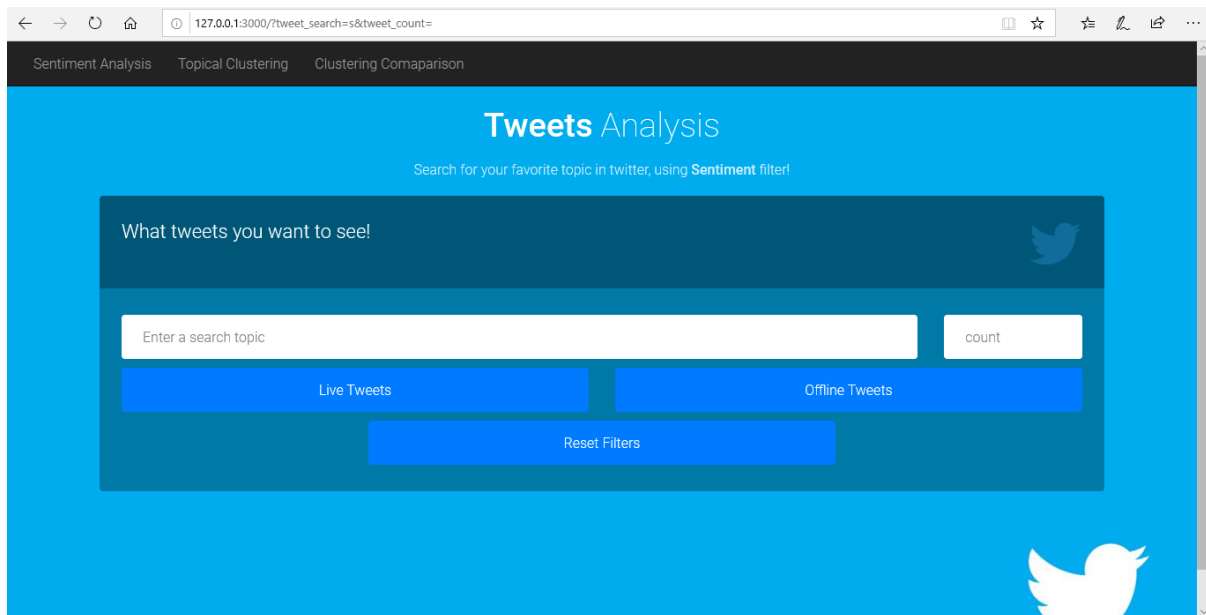


Figure 5.1 Sentiment Analysis Home Screen

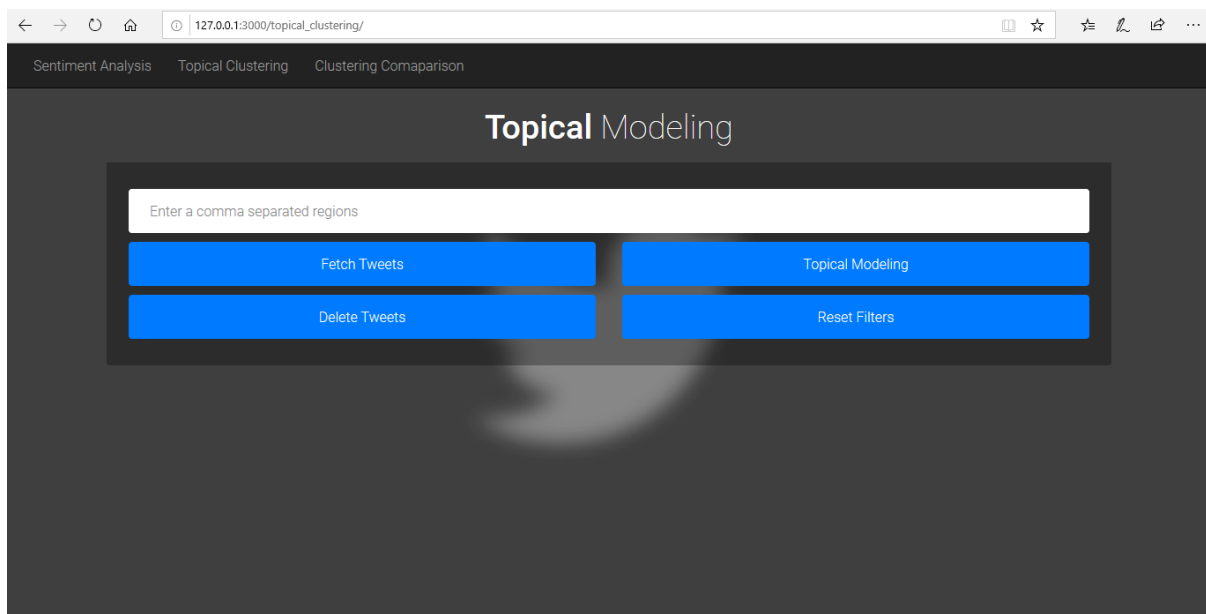


Figure 5.2 Topic Modelling Home Screen



Figure 5.3 Sentiment Analysis Result



Figure 5.4 Topic Modelling Output

CHAPTER 6 IMPLEMENTATION PLANNING

6.1 IMPLEMENTATION ENVIRONMENT

OS: Windows 10

Coding Language: Python

Frameworks: Bootstrap, Flask

API: Twitter API

Libraries: Tensorflow, Keras, Gensim, Sklearn, NLTK

6.2 PROGRAM/MODULES SPECIFICATION

- **Module 1 – Twitter Sentiment Analysis**
 - User can perform sentiment analysis on live tweets fetched on some particular topic or hashtag or geolocation.
 - Additionally, user can also perform sentiment analysis offline using pre-saved csv file by searching for topic.
 - On both the task mentioned above user get visualization of sentiment in form of Pie-chart.
- **Module 2 – Twitter Topic Modelling**
 - User can fetch the live tweets and save it in text file and using that file topic modelling would be performed.
 - User can delete the fetched tweets and start over for new topic.

6.3 CODING STANDARDS

Naming Conventions

- **Module Names**
 - Short, lowercase names, without underscores.
 - Example: myfile.py
- **Class Names**
 - CapWords convention.
 - Example: MyClass
- **Exception Names**
 - If a module defines a single exception raised for all sorts of conditions, it is generally called “Error”. Otherwise use CapWords convention i.e. MyError.

- Method Names and Instance Variables
 - The “Style Guide for Python Code” recommends using lowercase with words separated by underscores (example: `my_variable`). But since most of our code uses mixed case, we would be using this style (example: `myVariable`).
 - Use one leading underscore only for internal methods and instance variables i.e. protected. (example: `_myProtectedVar`)
 - Use two leading underscores to denote class-private names (example: `__myPrivateVar`).
 - Don’t use leading or trailing underscores for public attributes unless they conflict with reserved words, in which case, a single trailing underscore is preferable (example: `class_`).

Organizing Imports

- They should be always put at the top of the file, just after any module comments and docstrings, and before module globals and constants.
- Imports should be on separate lines.
 - Wrong: `import sys, os`
 - Right: `import sys`
`import os`
 - The following is ok, though:
`from types import StringType, ListType`
- Imports should be grouped in the following order with a blank line between each group of imports:
 - Standard library imports
 - Related major packages imports
 - Application specific imports

Indentation and line length

- indentions
 - Single tab.
 - Avoid using more than five levels of indentation.
- Line length
 - Maximum of 72 characters (never exceed 79 characters).
 - You can break a long line using “\”.

Break Lines

- Leave one line between function in a class.
- Extra blank lines may be used to separate groups of related functions.
- Blank lines may be omitted between bunches of related one-liners.
- Use blank lines in functions, sparingly, to indicate logical section.

White space

- Multiple statements on the same line are discouraged.
 - Wrong :
if foo == 'blah': doBlahThing()
 - Correct:
if foo == 'blah':
doBlahThing()
- No white space immediately before an open parenthesis.
 - Wrong: spam (1)
 - Correct: spam(1)
- No white space inside parentheses, brackets or braces.
 - Wrong: spam(ham[1], { eggs : 2 })
 - Correct: spam(ham[1], {eggs:2})
- No white space immediately before a comma, semicolon, or colon.
 - Wrong: if x == 4 :
 - Correct: if x == 4:
- No more than one space around an operator.
 - Wrong:
X = 1
Var1 = 2
 - X = 1
Var1 = 2
- Don't use spaces around the '=' sign when used to indicate a keyword argument or a default parameter value.
 - Wrong: def complex(real, imag = 0.0)
 - Correct : def complex(real, imag=0.0)
- Always surround the following operators with a single space on either side.
 - Assignment (=)
 - Comparisons (==, <, >, !=, <>, <=, >=, in, not in, is, is not)
 - Booleans (and, or, not)
 - Arithmetic operators (+, -, *, /, %)

Programming Recommendations

- Comparisons to singletons like None should always be done with 'is' or 'is not'.
 - Wrong: if x:
 - Right : if x is not None:
- Don't compare Boolean values to True or False.
 - Wrong: if greeting == True:
 - Correct: if greeting:
- Avoid slicing string when checking for prefixes or suffices. Use startswith() and endswith() instead.
 - Wrong: if foo[:3] == 'bar':
 - Correct: if foo.startswith('bar'):

Comments

- Block comments
 - They are indented to the same level as the code they apply to.
 - Each line of a block comment starts with a # and a single space.
 - Paragraphs inside a block comment are separated by a line containing a single #.
 - Block comment are best surrounded by a blank line above and below them.
- Inline comments
 - They should start with a # and a single space.
 - Should be separated by at least two spaces from the statement they apply to.

Documentation String

- Write docstrings for all public modules, functions, classes, and methods.
- Docstrings are not necessary for non-public methods, but you should have a comment that describes what the method does. This comment should appear after the "def" line.
- Insert a blank line before and after all doctsrings that document a class.
- One-line docstrings
 - The opening and closing "" are on the same line.
 - There is no blank line either or after the docstring.
 - Describes the function or method's effect as a command ("Do this", "Return that", not as a description.
- Multi-line docstrings

- The “""" that ends a multiline docstring should be on a line by itself.
- **Script:** The docstring of a script should be usable as its “usage” message. It should document the script’s function, the command line syntax, and the environment variables.
- **Module:** The docstring for a module should generally list the classes, exceptions and functions (and any other objects) that are exported by the module, with a one-line summary of each.
- **Class:**
 - The doctring for a class should summarize its behaviour and list the public methods and instance variables.
 - If the class is intended to be subclassed, and has an additional interface for subclasses, this interface should be listed separately.
 - If a class subclasses another class and its behaviour is mostly inherited from that class, its docstring should mention this and summarize the differences.
 - The class constructor should be documented in the docstring for its `__init__` method.
- **Function or method:**
 - The docstring should summarizes its behaviour and document its arguments, return value, side effects, exception raised, and restriction on when it can be called.
 - Optional arguments should be indicated.
 - Use the verb “override” to indicate that a subclass method replaces a superclass method and does not call the superclass method, use the verb “extend” to indicate that a subclass method calls the superclass method.
 - The docstring should contain a summary line, followed by a blank line, followed by a more elaborate description.

CHAPTER 7 TESTING

7.1 TESTING PLAN

7.1.1 Features to be tested

- Live tweets are been fetched or not in both Sentiment analysis and Topic modelling modules.
- Charts for visualization are generated properly or not.
- File containing tweets is deleted on button click or not in Topic modelling module.
- Result has been displayed or not in both modules.
- LSTM model predict the sentiment correct or not.

7.1.2 Approach

- We have used following strategies for testing
 - Unit Testing
 - Bottom-up Integration Testing
 - System Testing
 - Performance Testing
- We have used White Box method for testing.

7.2 TESTING STRATEGY

7.2.1 Unit Testing

Unit testing is performed for testing modules against detailed design. Inputs to the process are usually compiled modules from the coding process. Each modules are assembled into a larger unit during the unit testing process. Testing has been performed on each phase of project design and coding. We carry out the testing of module interface to ensure the proper flow of information into and out of the program unit while testing. We make sure that the temporarily stored data maintains its integrity throughout the algorithm's execution by examining the local data structure. Finally, all error-handling paths are also tested.

7.2.2 Bottom-up Integration Testing

Testing can be performed starting from smallest and lowest level modules and proceeding one at a time .When bottom level modules are tested attention turns to those on the next level that use the lower level ones they are tested individually and then linked with the previously

examined lower level modules. In a comprehensive software development environment, bottom-up testing is usually done first, followed by top- down testing.

7.2.3 System Testing

We usually perform system testing to find errors resulting from unanticipated interaction between the sub-system and system components. Software must be tested to detect and rectify all possible errors once the source code is generated before delivering it to the customers. For finding errors, series of test cases must be developed which ultimately uncover all the possibly existing errors. Different software techniques can be used for this process. These techniques provide systematic guidance for designing test that Exercise the internal logic of the software components, Exercise the input and output domains of a program to uncover errors in program function, behaviour and performance. We has tested the software using White Box testing method. Internal program logic is exercised using this test case design techniques. This technique helped in finding maximum number of errors with minimal effort and time.

7.2.4 Performance Testing

It is done to test the run-time performance of the software within the context of integrated system. These tests are carried out throughout the testing process. For example, the performance of individual module is accessed during white box testing under unit testing.

7.3 TEST SUITE

Test Case ID: T1					
Test Priority (Low/Medium/High): High					
Test Title: Test of LSTM model					
Description: See whether all text is classified correctly or not					
Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
1	Pass text to model	It is having a best amoled display	Positive	Positive with 0.71 score	Pass
2	Pass text to model	Product is having poor battery life	Negative	Negative with 0.31 score	Pass
3	Pass text to model	Some bugs fixed but looking forward for more optimization	Neutral	Neutral with 0.50 score	Pass

Table 7.1 Test case for LSTM model

Test Case ID: T2					
Test Priority (Low/Medium/High): High					
Test Title: Test of Twitter sentiment analysis module					
Description: See whether all are working					
Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
1	Click “Live Sentiment” button with topic and count at least one field empty	Click on button	Alert box with message	Alert box with message	Pass
2	Click “Live Sentiment” button with Internet disabled	Click on button With both topic and count field filled	Alert box with message	Alert box with message	Pass
3	Click “Live Sentiment” button	Click on button with internet enabled and both field filled	Table of tweets with their classification along with Pie-chart	Table of tweets with their classification along with Pie-chart	Pass
4	Click “Offline Sentiment” button with topic and count at least one field empty	Click on button	Alert box with message	Alert box with message	Pass
5	Click “Offline Sentiment” button with Internet disabled	Click on button With both topic and count field filled	Alert box with message	Alert box with message	Pass
6	Click “offline Sentiment” button	Click on button with internet enabled and both field filled	Table of tweets with their classification along with Pie-chart	Table of tweets with their classification along with Pie-chart	Pass

Table 7.2 Test case for Twitter Sentiment Analysis

Test Case ID: T3					
Test Priority (Low/Medium/High): High					
Test Title: Test of Twitter topic modelling module					
Description: See whether all are working					
Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
1	Click on “Fetch Tweets” button with topic field empty	Click on button	Alert box with message	Alert box with message	Pass
2	Click on “Fetch Tweets” button with Internet disabled	Click on button With both topic field filled	Alert box with message	Alert box with message	Pass
3	Click on “Fetch Tweets” button	Click on button with internet enabled and topic field filled	20 MB of tweets should be saved in text file	20 MB of tweets Saved in text file	Pass
4	Click on “Delete Tweets”	Click on button	Confirmation box followed by alert box with completion message	Confirmation box followed by alert box with completion message	Pass
5	Click on “Topic Modelling” with text file missing	Click on button	Alert box with message	Alert box with message	Pass
6	Click on “Topic Modelling” with text file having size less than 20 MB	Click on button	Alert box with message	Alert box with message	Pass
7	Click on “Topic Modelling”	Click on button	Table of bags of topics	Table of bags of topics	Pass

Table 7.3 Test case for Twitter Topic Modelling

CHAPTER 8 CONCLUSION AND DISCUSSION

8.1 PROBLEM ENCOUNTERED AND POSSIBLE SOLUTIONS

- In live sentiment analysis maximum of 100 tweets can be fetched per request. This can be solved by purchasing premium tweeter API plan.
- In topic modelling minimum 20 MB of tweets in text file is needed to get result. This can be solved by giving user feature to upload their own data.

8.2 CONCLUSION OF PROJECT WORK

We conclude that using word embedding technique in LSTM model it is easier to classify the tweets and more we improve the training dataset more we can get accurate results. Additionally, using word2Vector Model, TF-IDF model, Linear SVM model along with LDA algorithm we can more accurately perform topic modelling.

CHAPTER 9 LIMITATION AND FUTURE ENHANCEMENT

9.1 LIMITATIONS

- Maximum of 100 tweets can be fetched per request while performing live sentiment analysis.
- At least 20 MB of tweets in text file is needed to perform topic modelling.

9.2 FUTURE ENHANCEMENT

- User can upload their files if required.
- Analysing sentiment on emoji's.
- Interpreting sarcasm.
- Future research can be done with possible improvement such as more refined data and more accurate algorithm.
- Potential improvement can be made to our data collection and analysis method.

REFERENCES

1. <https://www.qsm.com/resources/function-point-languages-table>
2. <https://www.geeksforgeeks.org/software-engineering-cocomo-model/>
3. <https://medium.com/@tomaszbak/python-nlp-libraries-features-use-cases-pros-and-cons-da36a0cc6adb>
4. <https://www.analyticsvidhya.com/blog/2018/02/natural-language-processing-for-beginners-using-textblob/>
5. <https://palletsprojects.com/p/flask/>
6. <https://www.upgrad.com/blog/the-whats-what-of-keras-and-tensorflow/>
7. <https://www.machinelearningplus.com/nlp/gensim-tutorial/>
8. <https://monkeylearn.com/sentiment-analysis/>
9. <https://monkeylearn.com/blog/introduction-to-topic-modeling/>