POLARITY OF HASH TAG USING PATTERN MINING

A PROJECT REPORT

Submitted by

DHARANI R	**(814817104008)**
GANESAMOORTHY S	**(814817104012)**
KEERTHIKA N	**(814817104017)**

in partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

UNIVERSITY COLLEGE OF ENGINEERING, ARIYALUR

(A Constituent College of Anna University, Chennai)

ANNA UNIVERSITY : CHENNAI 600 025

AUGUST 2021

ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certificate that this project report entitled, **"POLARITY OF HASHTAG USING PATTERN MINING"** is the bonafide work of **DHARANI R (814817104008), GANESAMOORTHY S (814817104012), KEERTHIKA N (814817104017)** Who carried out the project under my supervision.

SIGNATURE	SIGNATURE
Dr.S.JAYANTHI M.TECH, Ph.D	**Dr.S.JAYANTHI M.TECH, Ph.D**
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Assistant Professor	**Assistant Professor**
Department of Computer Science and Engineering	Department of Computer Science and Engineering
University College of Engineering, Ariyalur – 621 704.	University College of Engineering, Ariyalur – 621 704.

Submitted for the University viva voce Examination held on.....

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We express our breathless thanks to our **Dr.V.VENKATESAN, ME, Ph.D.,** Dean i/c, University College of Engineering, Ariyalur, for giving constant motivation in succeeding in our goal.

We acknowledge our sincere thanks to **Dr.S.JAYANTHI M.Tech, Ph.D,** HOD i/c**,** Department of Computer Science and Engineering, University College of Engineering, Ariyalur, for the co-ordinating us throughout this project.

We are highly grateful to thank our project co-ordinator **Dr.R.ARUN PRAKASH M.Tech, Ph.D,** and our project guide **Dr.S.JAYANTHI M.Tech, Ph.D,** Department of Computer Science and Engineering, University College of Engineering, Ariyalur, for the co-ordinating us throughout this project.

We are very much indebted to thank all faculty members of Department of Computer Science and Engineering in our Institute, for their excellent moral support and suggestions to complete our project work successfully.

Finally our acknowledgement does our parents, sisters and friends those who had extended their excellent support and idea to make our project a pledge one.

DHARANI R

GANESAMOORTHY S

KEERTHIKA N

ABSTRACT

Nowadays, people from all around the world use social media sites to share information. Twitter for example is a platform in which users send, read posts known as ‘tweets’ and interact with different communities. Users share their daily lives, post their opinions on everything such as brands and places. Companies can benefit from this massive platform by collecting data related to opinions on them. The aim of this system is to present a model that can perform sentiment analysis of real data collected from Twitter. Data in Twitter is highly unstructured which makes it difficult to analyze. However, our proposed model is different from prior work in this field because it combined the use of supervised and unsupervised machine learning algorithms. The process of performing sentiment analysis as follows: Tweet extracted directly from Twitter API, then cleaning and discovery of data performed. After that the data were fed into several models for the purpose of training. Each tweet extracted classified based on its sentiment whether it is a positive, negative or neutral. Data were collected on two subjects McDonalds and KFC to show which restaurant has more popularity. Different machine learning algorithms were used. The results from these models were tested using various testing metrics like cross validation and f-score. Moreover, our model demonstrates strong performance on mining texts extracted directly from Twitter.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVATIONS

API	**:**	Application Program Interface
CSV	**:**	Comma Separated Value
DFD	**:**	Data Flow Diagram
FP	**:**	Frequent Pattern
GUI	**:**	Graphical User Interface
IR	**:**	Information Retrieval
IVR	**:**	Interactive Voice Response
KDD	**:**	Knowledge Discovery and Data
NLP	**:**	Natural Language Processing
OOP	**:**	Object Oriented Programming
SVM	**:**	Support Vector Machine
TF-IDF	**:**	Term Frequency-Inverse Document Frequency
TK	**:**	Tkinter
UI	**:**	User Interface
UML	**:**	Unified Modelling Language
URL	**:**	Unified Resource Locator

1 INTRODUCTION

1.1 DATA MINING

The process of extracting information to identify patterns, trends, and useful data that would allow the business to take the data-driven decision from huge sets of data is called Data Mining.

In other words, we can say that Data Mining is the process of investigating hidden patterns of information to various perspectives for categorization into useful data, which is collected and assembled in particular areas such as data warehouses, efficient analysis, data mining algorithm, helping decision making and other data requirement to eventually cost-cutting and generating revenue.

Data mining is the act of automatically searching for large stores of information to find trends and patterns that go beyond simple analysis procedures. Data mining utilizes complex mathematical algorithms for data segments and evaluates the probability of future events. Data Mining is also called Knowledge Discovery of Data (KDD).

Data Mining is a process used by organizations to extract specific data from huge databases to solve business problems. It primarily turns raw data into useful information. Data Mining is similar to Data Science carried out by a person, in a specific situation, on a particular data set, with an objective. This process includes various types of services such as text mining, web mining, audio and video mining, pictorial data mining, and social media mining. It is done through software that is simple or highly specific. By outsourcing data mining, all the work can be done faster with low operation costs. Specialized firms can also use new technologies to collect data that is impossible to locate manually. There are tonnes of information available on various platforms, but very little knowledge is accessible. The biggest challenge is to analyse the data to extract important information that can be used to solve a problem or for

company development. There are many powerful instruments and techniques available to mine data and find better insight from it.

1.2 PATTERN MINING

Pattern mining concentrates on identifying rules that describe specific patterns within the data. Market-basket analysis, which identifies items that typically occur together in purchase transactions, was one of the first applications of data mining. For example, supermarkets used market-basket analysis to identify items that were often purchased together—for instance, a store featuring a fish sale would also stock up on tartar sauce. Although testing for such associations has long been feasible and is often simple to see in small data sets, data mining has enabled the discovery of less apparent associations in immense data sets. Of most interest is the discovery of unexpected associations, which may open new avenues for marketing or research. Another important use of pattern mining is the discovery of sequential patterns; for example, sequences of errors or warnings that precede an equipment failure may be used to schedule preventative maintenance or may provide insight into a design flaw.

1.3 NATURAL LANGUAGE PROCESSING

Natural Language Processing, usually shortened as NLP, is a branch of artificial intelligence that deals with the interaction between computers and humans using the natural language.

The ultimate objective of NLP is to read, decipher, understand, and make sense of the human languages in a manner that is valuable.

Most NLP techniques rely on machine learning to derive meaning from human languages. Natural Language Processing is the driving force behind the following common applications: Language translation applications such as Google Translate Word Processors such as Microsoft Word and Grammarly that employ NLP to check grammatical accuracy of texts. Interactive Voice

Response (IVR) applications used in call centre's to respond to certain users' requests. Personal assistant applications such as OK Google, Siri, Cortana, and Alexa.

NLP entails applying algorithms to identify and extract the natural language rules such that the unstructured language data is converted into a form that computers can understand. When the text has been provided, the computer will utilize algorithms to extract meaning associated with every sentence and collect the essential data from them. Sometimes, the computer may fail to understand the meaning of a sentence well, leading to obscure results.

1.4 LITERATURE SURVEY

[1] Retrieving information for micro blog using Pattern Mining and Relevance feedback

Retrieving information from Twitter is always challenging due to its large volume, inconsistent writing and noise. Most existing information retrieval (IR) and text mining methods focus on term-based approach, but suffer from the problems of terms variation such as polysemy and synonymy. This problem deteriorates when such methods are applied on Twitter due to the length limit. Over the years, people have held the hypothesis that pattern-based methods should perform better than term-based methods as it provides more context, but limited studies have been conducted to support such hypothesis especially in Twitter. This system presents an innovative framework to address the issue of performing IR in micro blog. The proposed framework discovers patterns in tweets as higher level feature to assign weight for low-level features (i.e. terms) based on their distributions in higher level features. We present the experiment results based on TREC11 micro blog dataset and shows that our proposed approach significantly outperforms term-based methods Okapi BM25, TF-IDF and pattern based methods, using precision, recall and F measures.

ADVANTAGES:

- It addresses the issue of performing IR in micro blog.
- It discovers patterns in tweets as higher level features to assign weight for low level features.

DISADVANTAGES:

- Micro blog is not popular.
- It has only 140 character limit.
- In this system, Twitter is not used.

[2] Mining Frequent Patterns without using candidate generation

Mining frequent patterns in transaction databases, time-series databases, and many other kinds of databases has been studied popularly in data mining research. Most of the previous studies adopt an Apriori-like candidate set generation-and-test approach. However, candidate set generation is still costly, especially when there exist a large number of patterns and/or long patterns. In this study, we propose a novel frequent-pattern tree (FP-tree) structure, which is an extended prefix-tree structure for storing compressed, crucial information about frequent patterns, and develop an efficient FP-tree-based mining method, FP-growth, for mining the complete set of frequent patterns by pattern fragment growth. Efficiency of mining is achieved with three techniques: (1) a large database is compressed into a condensed, smaller data structure, FP-tree which avoids costly, repeated database scans, (2) our FP-tree-based mining adopts a pattern-fragment growth method to avoid the costly generation of a large number of candidate sets, and (3) a partitioning-based, divide-and-conquer method is used to decompose the mining task into a set of smaller tasks for mining confined patterns in conditional databases, which dramatically reduces the search space. Our performance study shows that the FP-growth method is efficient and scalable for mining both long and short frequent patterns, and is

about an order of magnitude faster than the Apriori algorithm and also faster than some recently reported new frequent-pattern mining methods.

ADVANTAGES:

- It develops an FP-growth, for mining the complete set of frequent patterns by Pattern growth.
- It is efficient and scalable.

DISADVANTAGES:

- Previous studies adopt an Apriori like candidate set generation and test approach.
- Apriori will scan database many times repeatedly for finding candidate sets.

[3] Sentimental analysis of Twitter data

Nowadays, people from all around the world use social media sites to share information. Twitter for example is a platform in which users send, read posts known as `tweets' and interacts with different communities. Users share their daily lives, post their opinions on everything such as brands and places. Companies can benefit from this massive platform by collecting data related to opinions on them. The aim of this system is to present a model that can perform sentiment analysis of real data collected from Twitter. Data in Twitter is highly unstructured which makes it difficult to analyse. However, our proposed model is different from prior work in this field because it combined the use of supervised and unsupervised machine learning algorithms. The process of performing sentiment analysis as follows: Tweet extracted directly from Twitter API, then cleaning and discovery of data performed. After that the data were fed into several models for the purpose of training. Each tweet extracted classified based on its sentiment whether it is a positive, negative or neutral. Data were

collected on two subjects McDonalds and KFC to show which restaurant has more popularity. Different machine learning algorithms were used. The result from these models was tested using various testing metrics like cross validation and f-score. Moreover, our model demonstrates strong performance on mining texts extracted directly from Twitter.

ADVANTAGES:

- It performs a real data collection from Twitter.
- It combines the use of supervised and unsupervised Machine Learning algorithm.

DISADVANTAGES:

- Support Vector Machine (SVM) algorithm is not suitable for large data sets.
- SVM does not perform very well when the data set has more noise.

[4] Prediction of 2019 Indian election using Sentimental Analysis

Sentiment analysis is considered to be a category of machine learning and natural language processing. It is used to extricate, recognize, or portray opinions from different content structures, including news, audits and articles and categorizes them as positive, neutral and negative. It is difficult to predict election results from tweets in different Indian languages. We used Twitter Archiver tool to get tweets in Hindi language. We performed data (text) mining on 42,235 tweets collected over a period of a month that referenced five national political parties in India, during the campaigning period for general state elections in 2016. We made use of both supervised and unsupervised approaches. We utilized Dictionary Based, Naive Bayes and SVM algorithm to build our classifier and classified the test data as positive, negative and neutral. We identified the sentiment of Twitter users towards each of the considered

Indian political parties. The results of the analysis for Naive Bayes were the BJP (Bhartiya Janta Party), for SVM it was the BJP (Bhartiya Janta Party) and for the Dictionary Approach it was the Indian National Congress. SVM predicted a 78.4% chance that the BJP would win more elections in the general election due to the positive sentiment they received in tweets. As it turned out, BJP won 60 out of 126 constituencies in the 2016 general election, far more than any other political party as the next party (the Indian National Congress) only won 26 out of 126 constituencies.

ADVANTAGES:

- It made up of use of both supervised and unsupervised approaches.
- To build classifier, they use Naïve Bayes and SVM algorithm.

DISADVANTAGE:

- It is difficult to predict election results from tweets in different Indian languages.

[5] Sentimental analysis on product reviews

Now, more than ever, it's key for companies to pay close attention to the voice of customer (VoC) to improve their products. Product managers need insights that will help them develop a robust product roadmap; it's about providing customers with what they actually want, rather than with what businesses think they need. A good place to start collecting product feedback is from online review sites (such as Capterra, G2Crowd, and Google Play). But manually analysing this unstructured data would take far too long. That's where sentiment analysis can help to: Understand what your customers like and dislike about your product. Compare your product reviews with those of your competitors. Get the latest product insights in real-time, 24/7.Save hundreds of hours of manual data processing. Sentiment analysis is the automated process of

understanding the sentiment or opinion of a given text. You can use it to automatically analyse product reviews and sort them by Positive, Neutral, and Negative.

ADVANTAGES:

- Understand what your customers like and dislike about your product.
- Compare your product reviews with those of your competitors.
- Save hundreds of hours of Manual data processing.

DISADVANTAGES:

- Naïve Bayes algorithm requires training.
- Prediction may go wrong due to lack of training.

[6] Real time sentimental analysis of 2019 election tweets using word2vec and Random forest model

Sentiment analysis of social media data consists of attitudes, assessments, and emotions which can be considered a way human think. Understanding and classifying the large collection of documents into positive and negative aspects are a very difficult task. Social networks such as Twitter, Facebook, and Instagram provide a platform in order to gather information about people's sentiments and opinions. Considering the fact that people spend hours daily on social media and share their opinion on various different topics helps us analyse sentiments better. More and more companies are using social media tools to provide various services and interact with customers. Sentiment Analysis (SA) classifies the polarity of given tweets to positive and negative tweets in order to understand the sentiments of the public. This system aims to perform sentiment analysis of real-time 2019 election twitter data using the feature selection model word2vec and the machine learning algorithm random forest for sentiment classification. Word2vec with Random Forest improves the accuracy of

sentiment analysis significantly compared to traditional methods such as BOW and TF-IDF. Word2vec improves the quality of features by considering contextual semantics of words in a text hence improving the accuracy of machine learning and sentiment analysis.

ADVANTAGES:

- Random forest model improves the accuracy of sentiment analysis significantly compared to any other methods.
- Word2Vec improves the quality of features by considering contextual semantics of words in a test.

DISADVANTAGES:

- It requires much time for training.
- It combines a lot of decision trees to determine the class.

[7] Sentimental analysis in social media and its application

This system is a report of a review on sentiment analysis in social media that explored the methods, social media platform used and its application. Social media contain a large amount of raw data that has been uploaded by users in the form of text, videos, photos and audio. The data can be converted into valuable information by using sentiment analysis. A systematic review of studies published between 2014 to 2019 was undertaken using the following trusted and credible database including ACM, Emerald Insight, IEEE Xplore, Science Direct and Scopus. After the initial and in-depth screening of system, 24 out of 77 articles have been chosen from the review process. The articles have been reviewed based on the aim of the study. The result shows most of the articles applied opinion-lexicon method to analyses text sentiment in social media, extracted data on micro blogging site mainly twitter and sentiment analysis application can be seen in world events, healthcare, politics and business.

ADVANTAGES:

- It offers immense opportunities to study and analyse the human opinions and sentiments.
- It discusses challenges in sentiment analysis and its application in disaster relief.

DISADVANTAGES:

- It computes document similarity directly in the word count space.
- It making really slow for large documents.

2 SYSTEM REQUIREMENTS

The system requirements including software and hardware configuration are explained in detail.

2.1 HARDWARE SYSTEM CONFIGURATION

The minimum hardware requirements of a computer are given below,

- Processor : Pentium-IV
- RAM : 4 GB
- Hard Disk : 20 GB

2.2 SOFTWARE SYSTEM CONFIGURATION

The system must have at least following configuration,

- Operating System : Windows 7 /10
- Software : Python 3.8 / above

2.3 SOFTWARE ENVIRONMENT

This application is created using python and Tkinter (TK) Graphical User Interface (GUI). A brief introduction of python is given below.

2.3.1 Python Programming Language

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

1. Simple
2. Easy-to-learn
3. Easy-to-maintain
4. A broad standard library
5. Interactive mode

6. Portable
7. Extendable
8. Databases
9. GUI Programming
10. Scalable

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

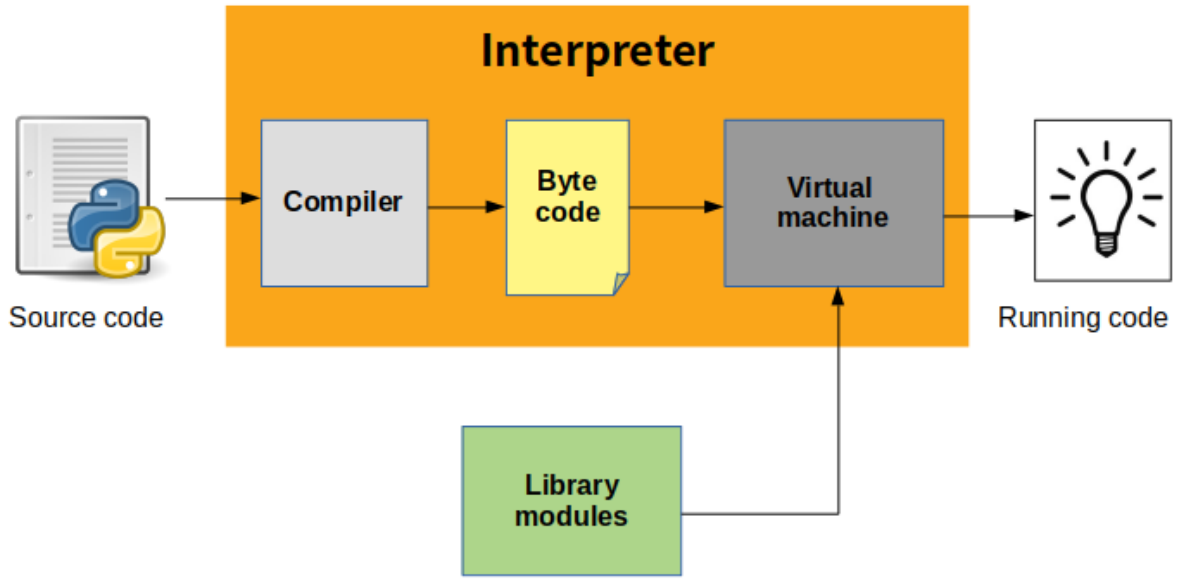

Figure 2.1 : Python Platform

3 DESCRIPTION OF THE PROBLEM

A brief discussion of existing system problem and how we overcome that problem in proposed method are stated in next section.

3.1 PROBLEM DEFINITION

1. Many Social Media networks like Twitter, Instagram does not able to find how many people support or against the post of any tweets by using Hash tags (#). For Example, #FarmerProtest100Days and #SpeakUpAgainstPriceRise.
2. But some people posting negative comment (or) feedback on some public figures in social media that projects the particular figure in negative manner.

DRAWBACKS:

- Difficult to find how many people support the post.
- It does not separate the people who supports and against the post.

3.2 PROPOSED SYSTEM

1. This project is used to separate the Supporters and Haters who against the tweets by using frequent pattern growth algorithm and natural language processing.
2. In frequent pattern growth algorithm there is no candidate generation.
3. It constructs conditional, frequent pattern tree and conditional pattern base from database
4. The frequent item set are represented in tree format and it scan the database only twice.
5. Tree structure maintain association rule between the item set.

6. With this frequent pattern growth algorithm search for frequent item set is reduced comparatively.
7. This algorithm is efficient and scalable for mining long and short frequent patterns.
8. Pre-Trained NLP Module.
9. The natural language processing is used to identify the text as positive, neutral or negative.
10. Which deals with the interaction between computers and human using the natural language?
11. To identify original frequency of tags and finding meaning of tags.

ADVANTAGES:

- To Find Supporters and Haters for the post.
- Separate Hash tags from Description by using Natural Language Processing.

4 DESIGN OF THE SYSTEM

The architecture of the proposed method is show below.

4.1 ARCHITECHTURAL DESIGN

In proposed method the user have to enter their query which analysis they want to perform, based on the three options we access the twitter API (Application Program Interface) and collect the data from particular twitter account using tweepy module, then the data's are saved as dataset in CSV (Comma Separated Value) file.

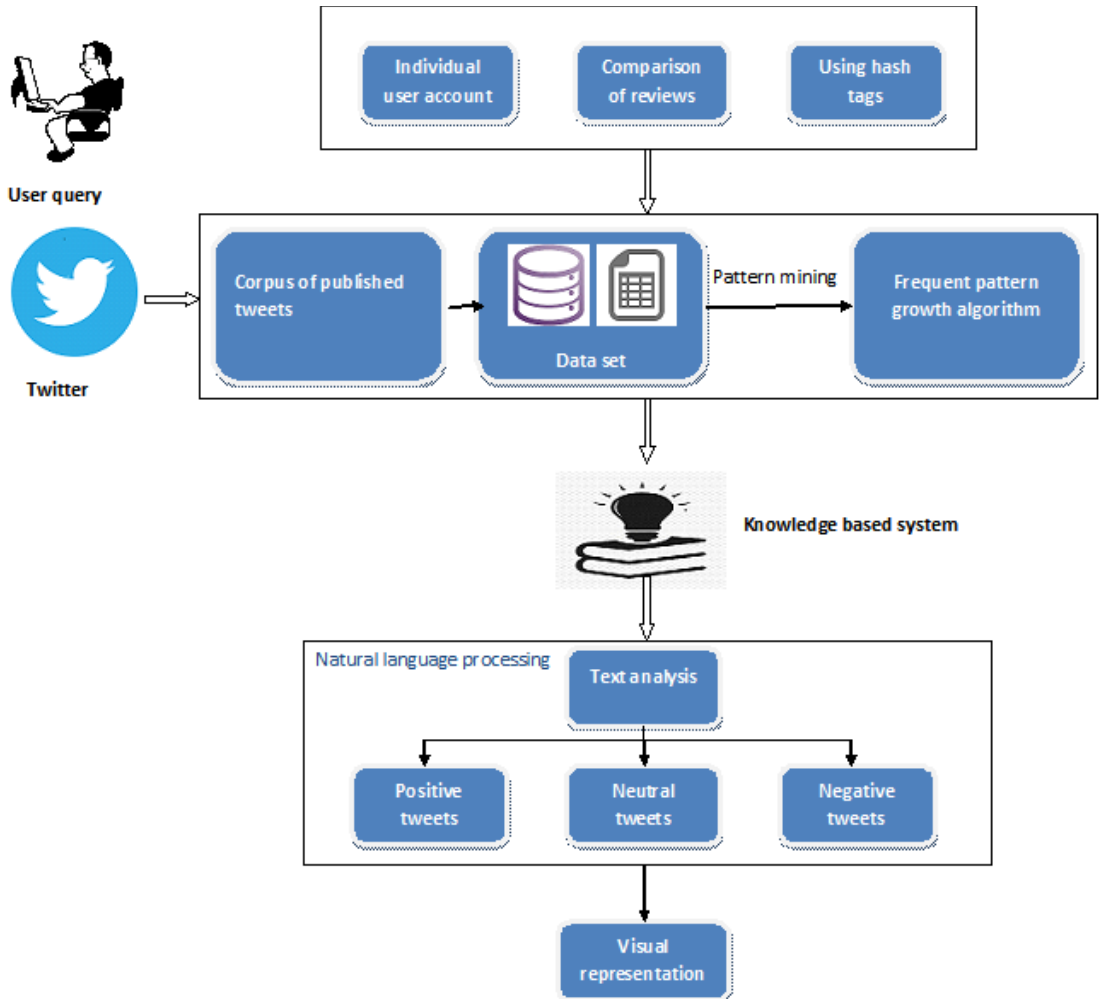

Figure 4.1 : Architectural Design

The figure 4.1 shows the architecture of hash tag polarity using pattern mining.

Using frequent pattern growth algorithm the frequent item sets get separated for easy access which stored in a knowledge based system after that NLP(Natural Language Processing) starts analysing the text as positive, negative and neutral. Before classifying the text it removes the stop words, URL, and links.

4.2 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

4.2.1 USE CASE DIAGRAM

A use case diagram in the Unified Modelling Language (UML) is a type of behavioural diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

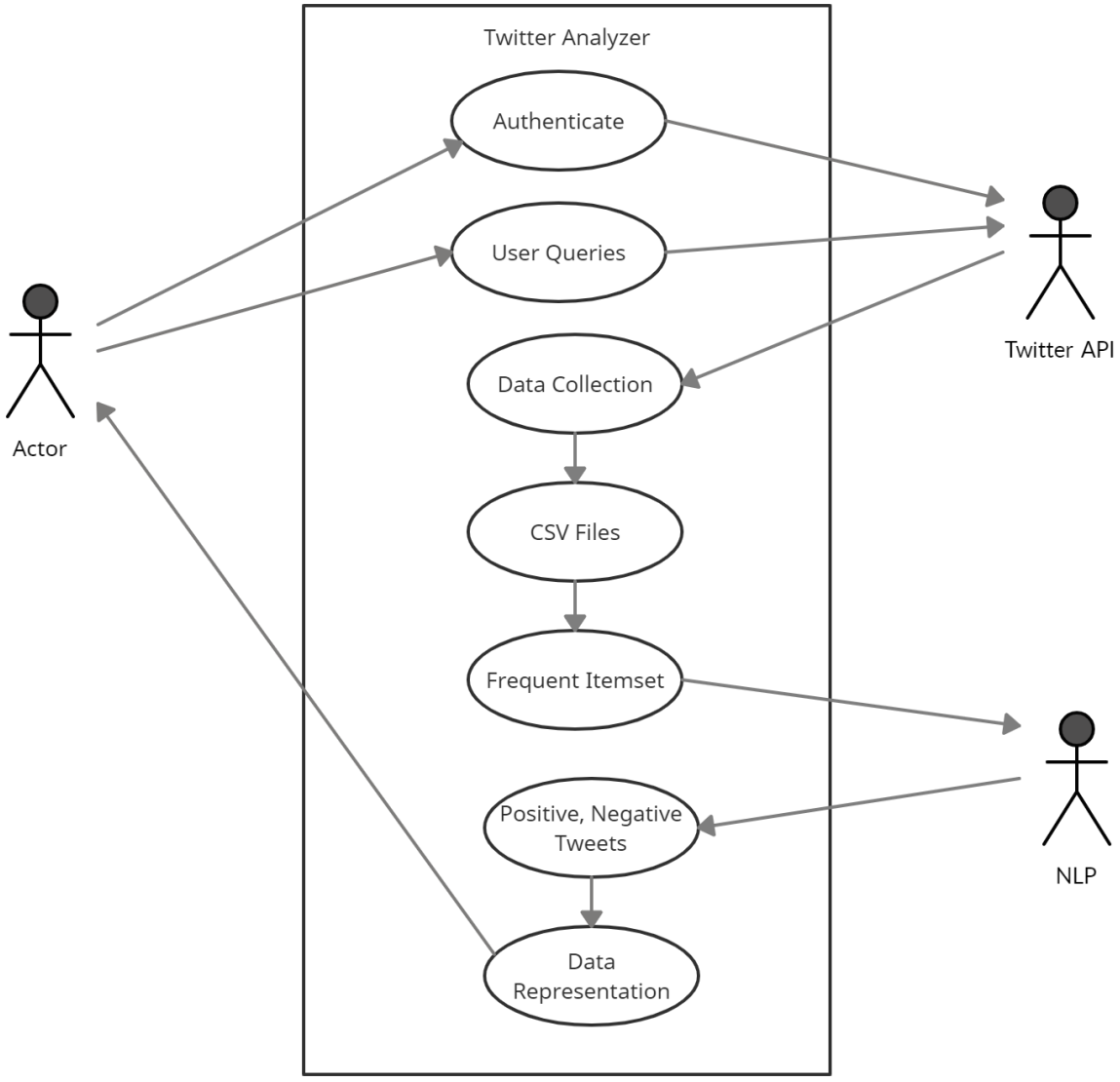

Figure 4.2 : Use Case Diagram

4.2.2 CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

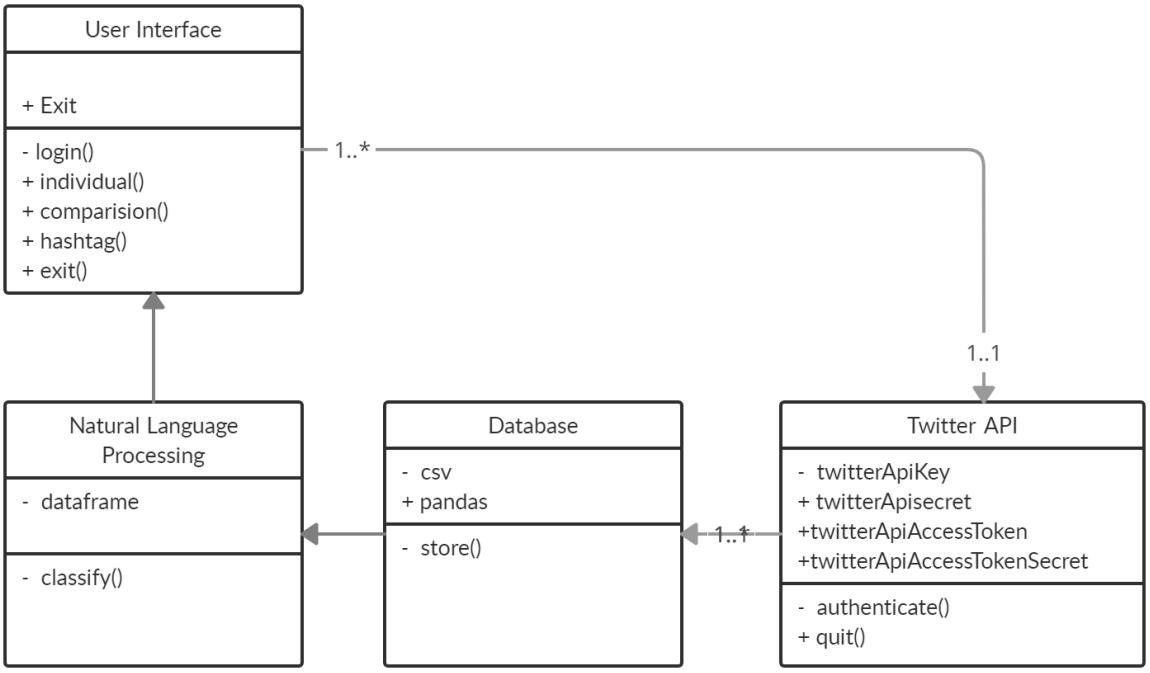

Figure 4.3 : Class Diagram

4.2.3 SEQUENCE DIAGRAM

A sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

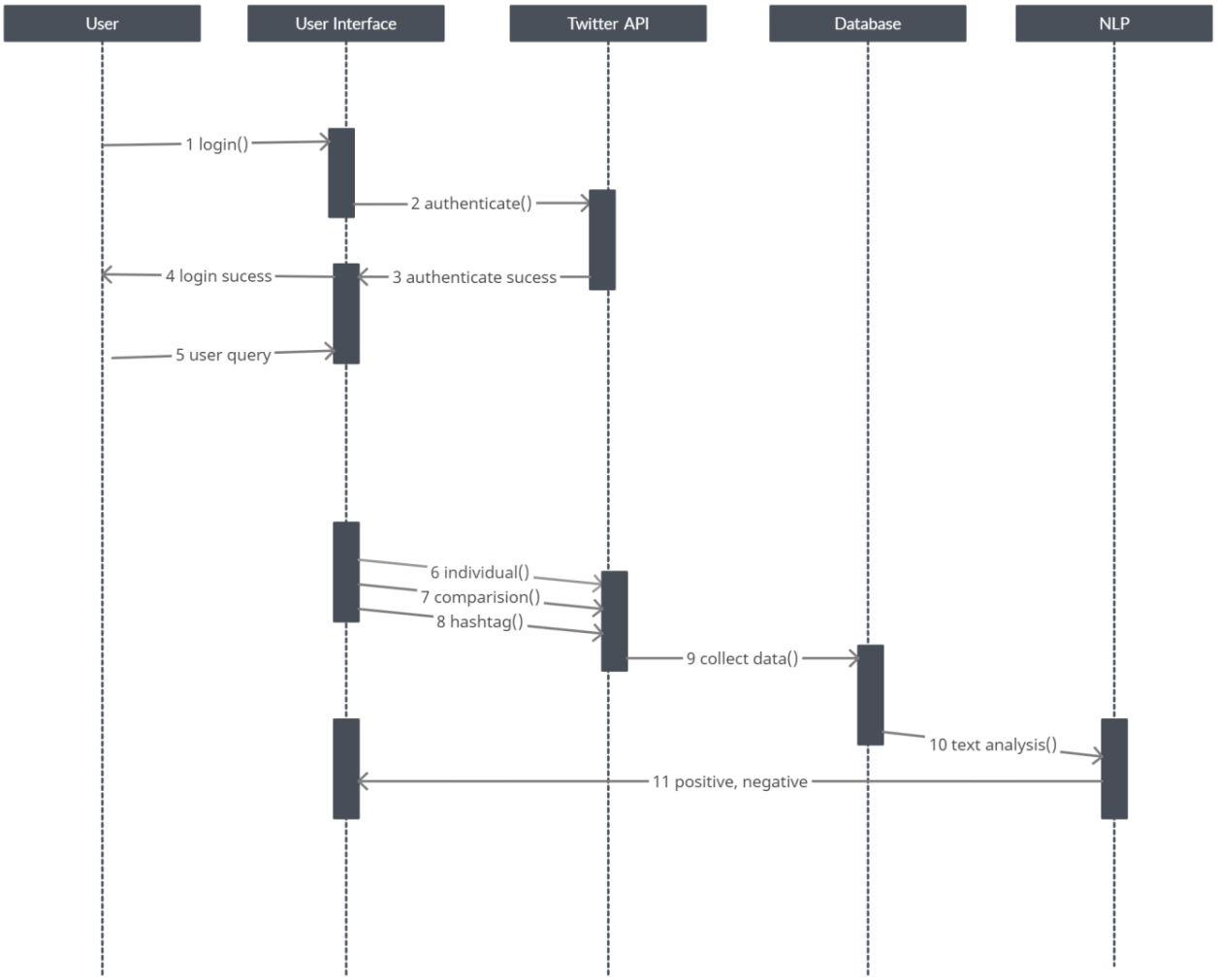

Figure 4.4 : Sequence Diagram

4.3 DATA FLOW DIAGRAM

The Data Flow Diagram (DFD) is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.

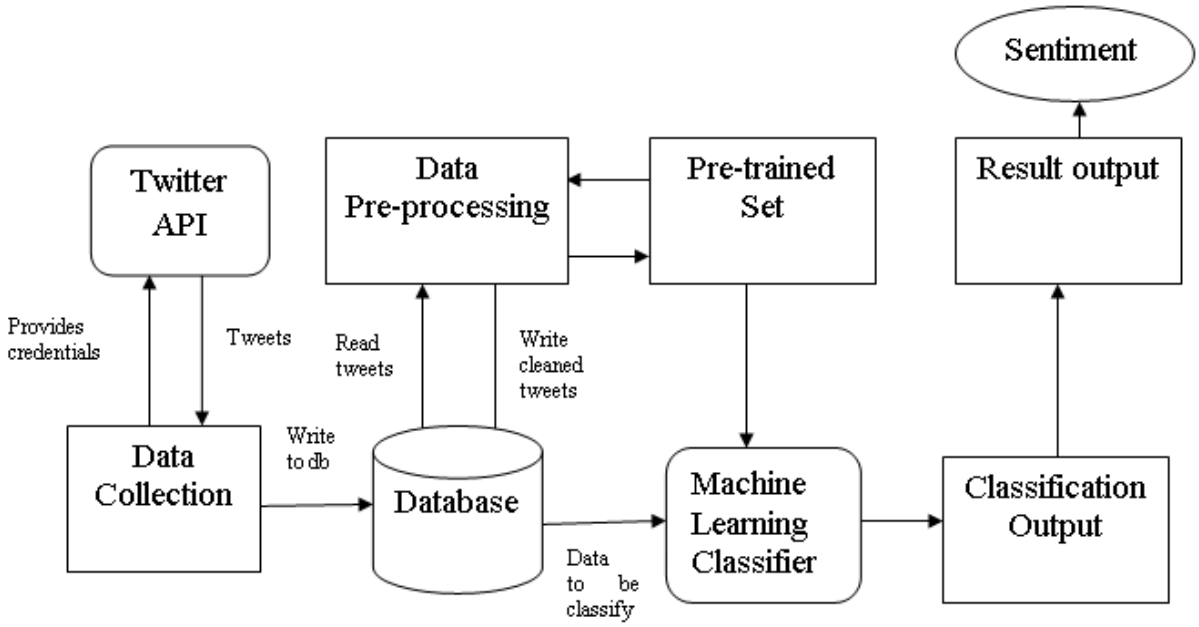

Figure 4.5 : Data Flow Diagram

5 SYSTEM IMPLEMENTATION

The module separation, system implementation and testing process of the proposed method are detailed below.

5.1 MODULE

In the proposed method, the module separation are listed below

1. GUI
2. Twitter developer login
3. Data collection
4. Clean-up tweets
5. Natural language processing
6. Decision

5.1.1 GUI

GUI is a Graphical User Interface used to get input queries from the user.

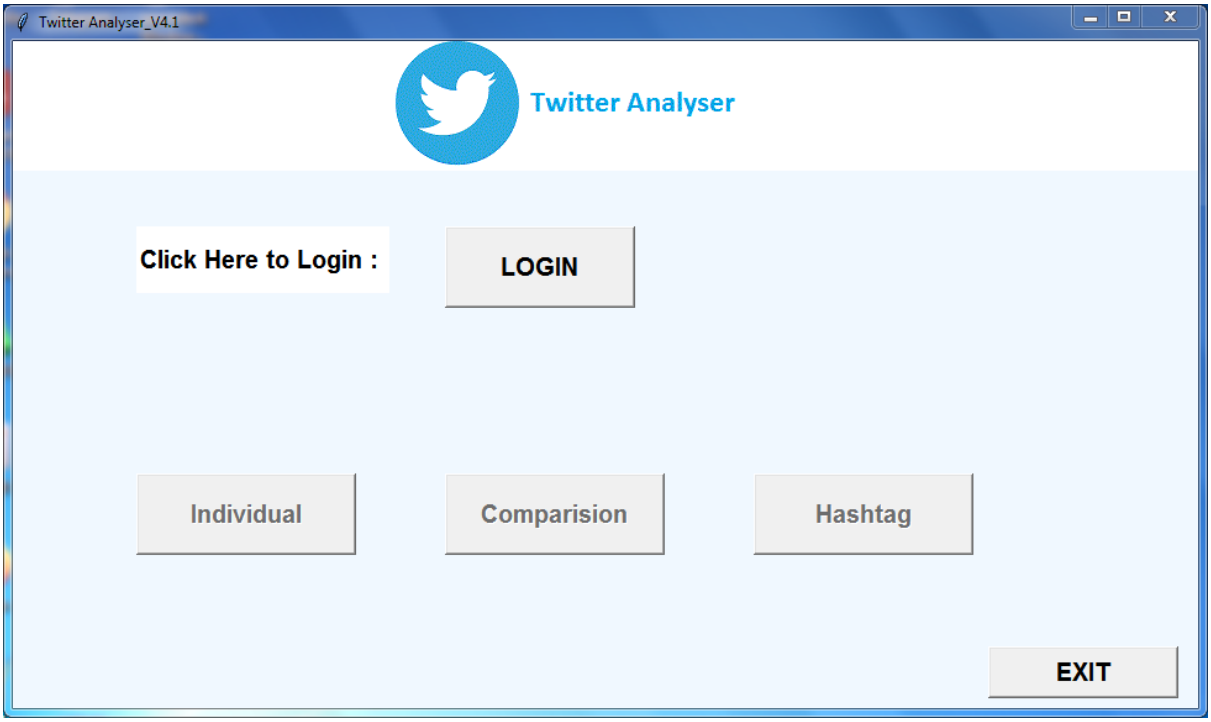

Figure 5.1 : Main GUI

- Tkinter is a GUI Toolkit for creating User friendly Interfaces.
- Fast Simple and reliable.
- It's a cross platform toolkit which supports windows, Mac and Linux.
- Here we are using GUI toolkit for getting user queries.

5.1.2 Twitter developer login

- The first step for this project is to apply for a twitter developer account.
- Twitter will provide a unique access token and access token secret key using these keys.
- These keys can be used for the API Authentication.

5.1.3 Data collection

- After authentication, we can either collect tweets by a particular user or by a hash tag by using the Tweepy Module
- Then the captured content is stored together as dataset.
- The datasets are text file saved as table structured format called CSV (comma separated values) file.
- The frequent items in dataset are separated by using FP-Growth algorithm.

5.1.4 Clean-up tweets

For getting better result in Natural language processing, we need to clean tweets

- Remove Links
- Remove Hash tags
- Remove Retweets
- Remove URL's
- Spelling correction
- Grammatical Error Correction

5.1.5 Natural language processing

- NLP (Natural Language Processing) is used to classify the data from tweets.
- Text Blob is simplified Text processing Module
- Classification is the process of identifying opinions in text and labelling them as positive, negative, or neutral based on the tweets.
- It can also give score for each tweet on a scale of -1 to 1

6 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 provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product it is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

6.1 UNIT TESTING

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

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

6.1.1 Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

6.1.2 Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

6.1.3 Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

6.2 INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up-software applications at the company level – interact without error.

6.2.1 Test Results

All the test cases mentioned above passed successfully. No defects encountered

6.3 FUNCTIONAL TESTING

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

Functional testing is centred on the following items:

- Valid Input : identified classes of valid input must be accepted.
- Invalid Input : identified classes of invalid input must be rejected.
- Functions : identified functions must be exercised.
- Output : identified classes of application outputs are shown.
- Systems : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

6.4 SYSTEM TESTING

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

6.4.1 White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

6.4.2 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

6.4.3 Acceptance Testing

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

6.5 TEST RESULTS

All the test cases mentioned above passed successfully. No defects encountered.

7 SCREENSHOTS

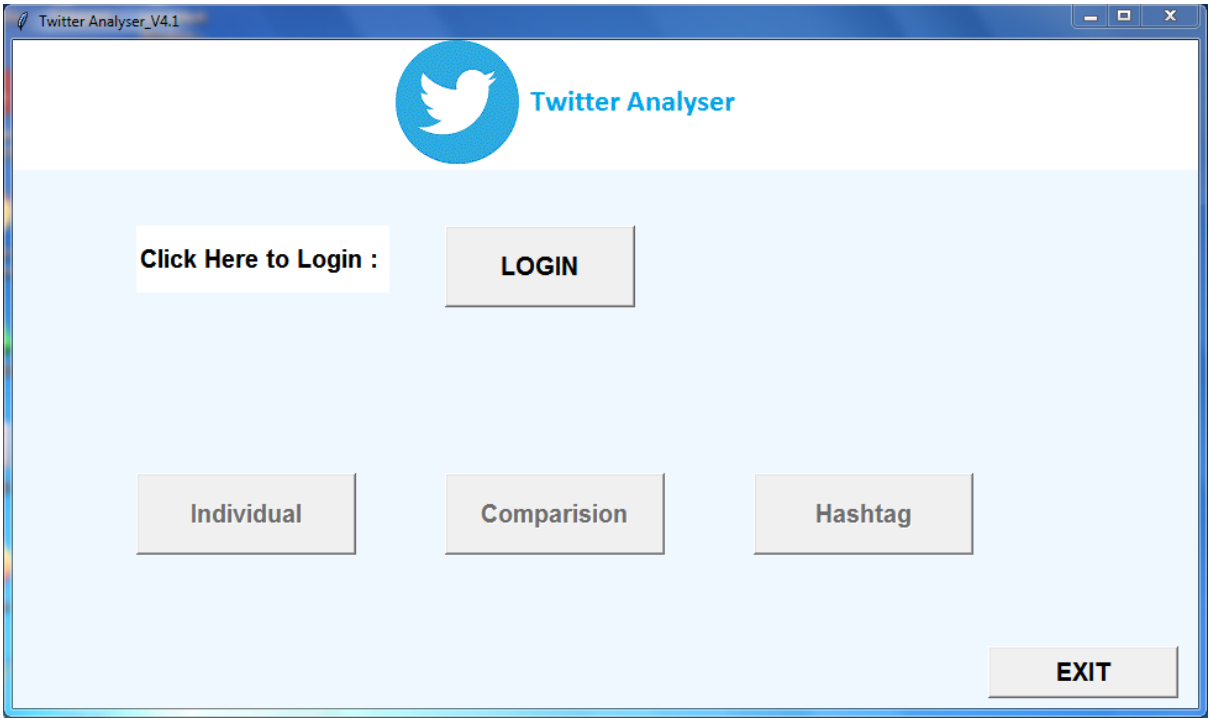

Figure 7.1 : Home Screen

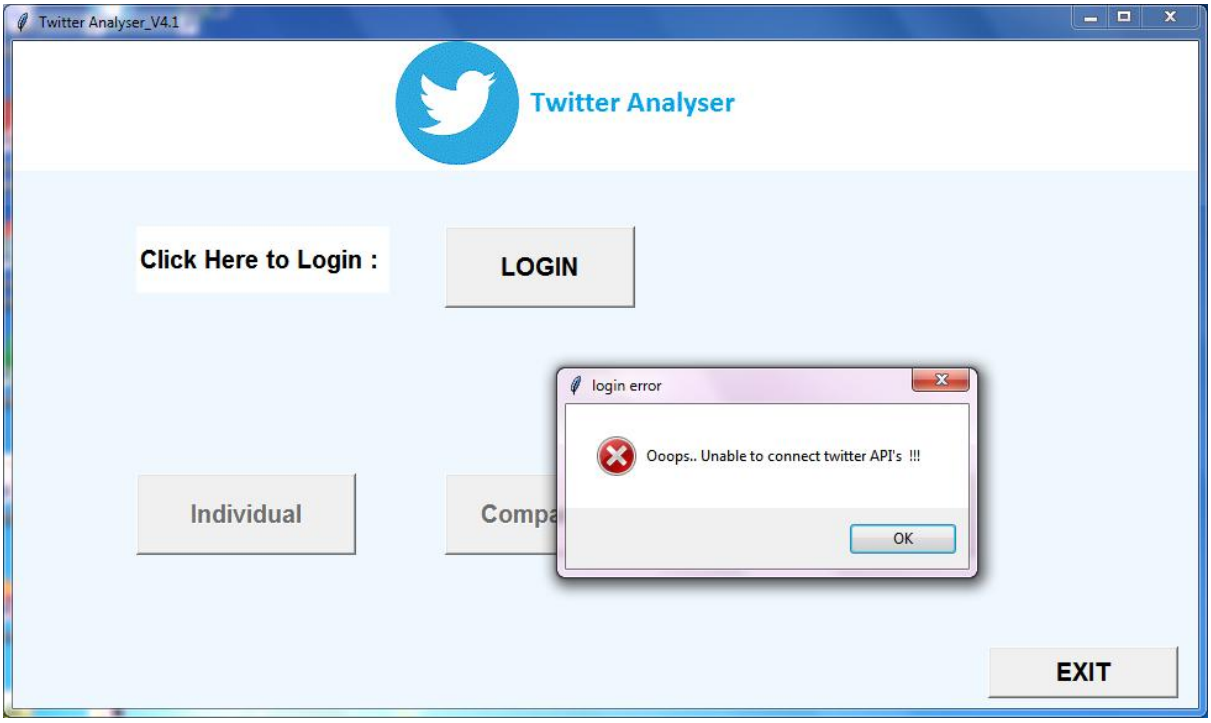

Figure 7.2 : Login Fail

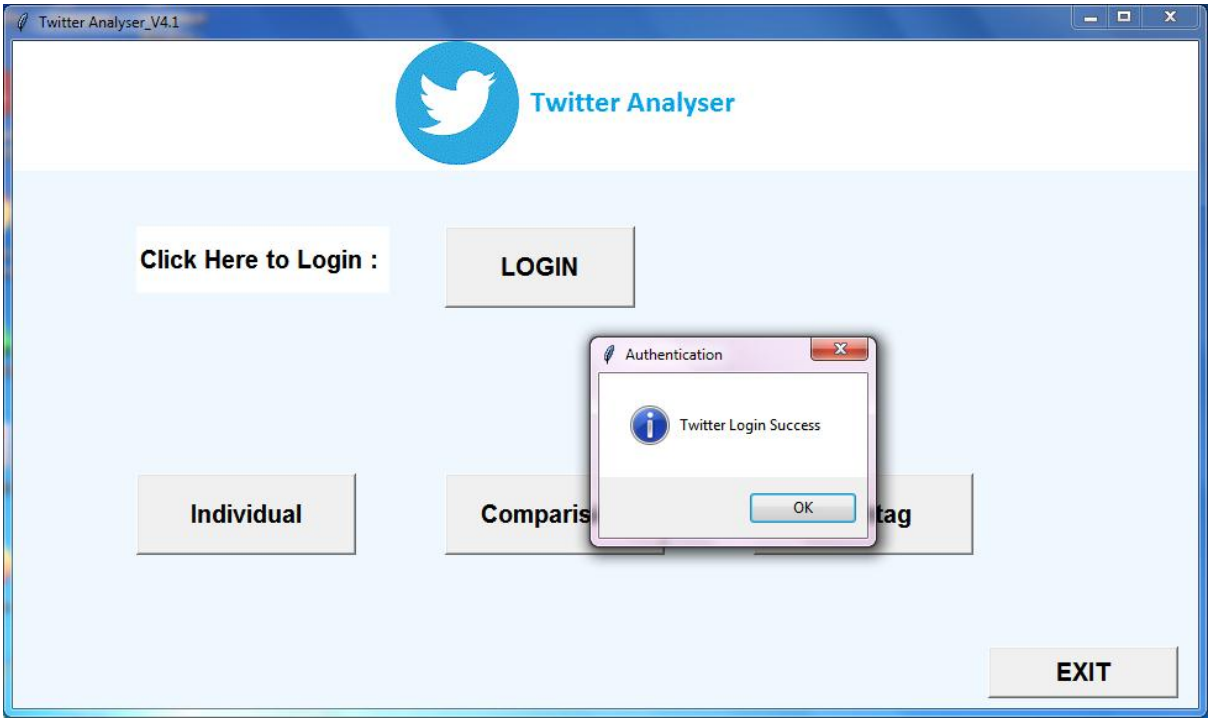

Figure 7.3 : Login Success

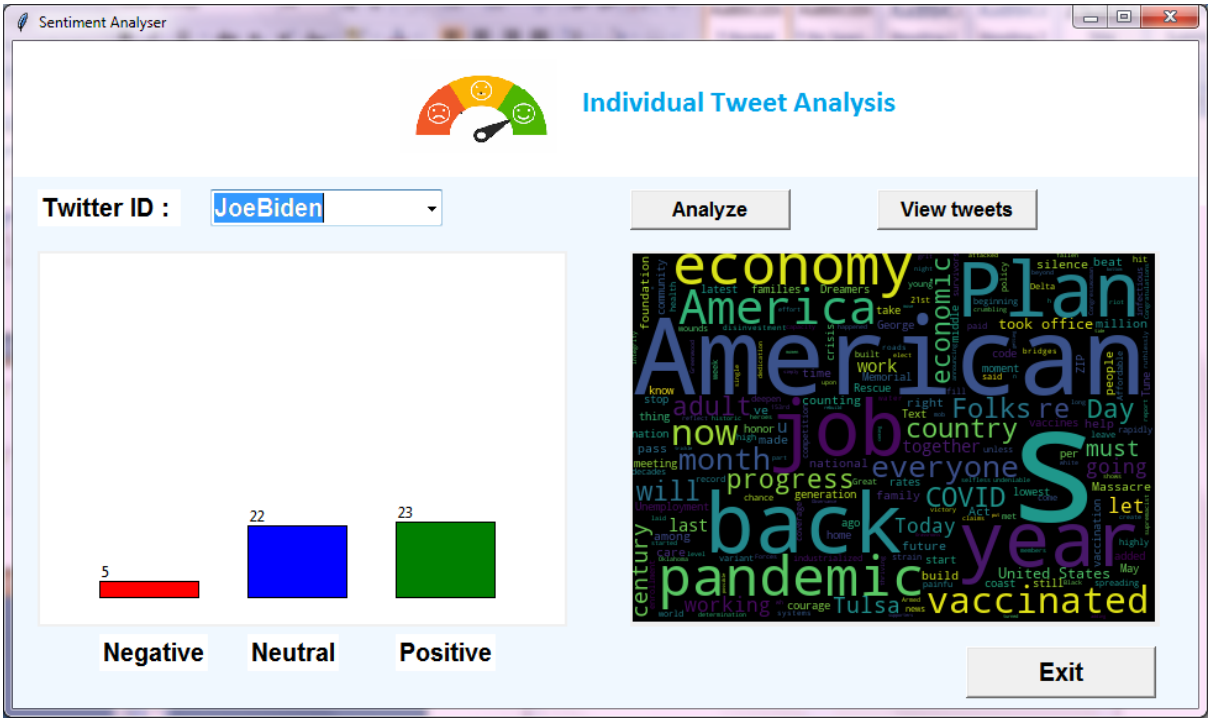

Figure 7.4 : Individual Twitter ID Analysis

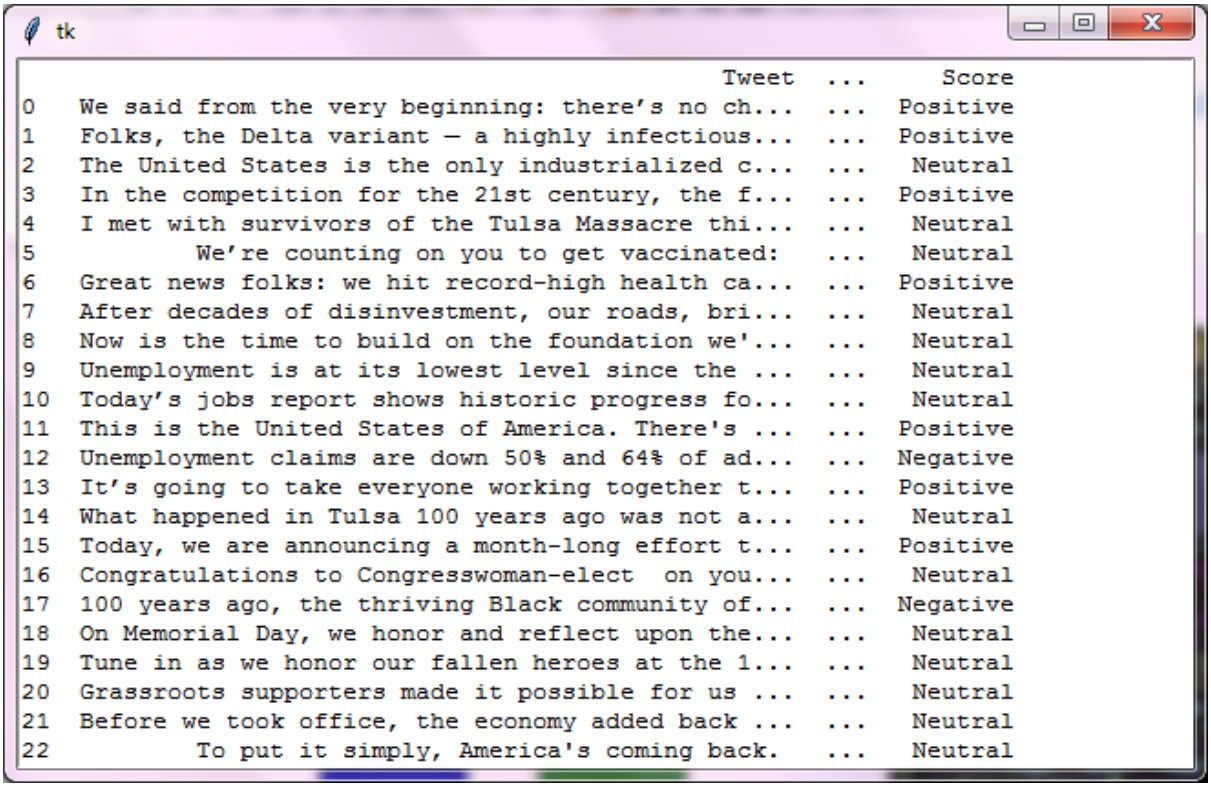

Figure 7.5 : View Tweets

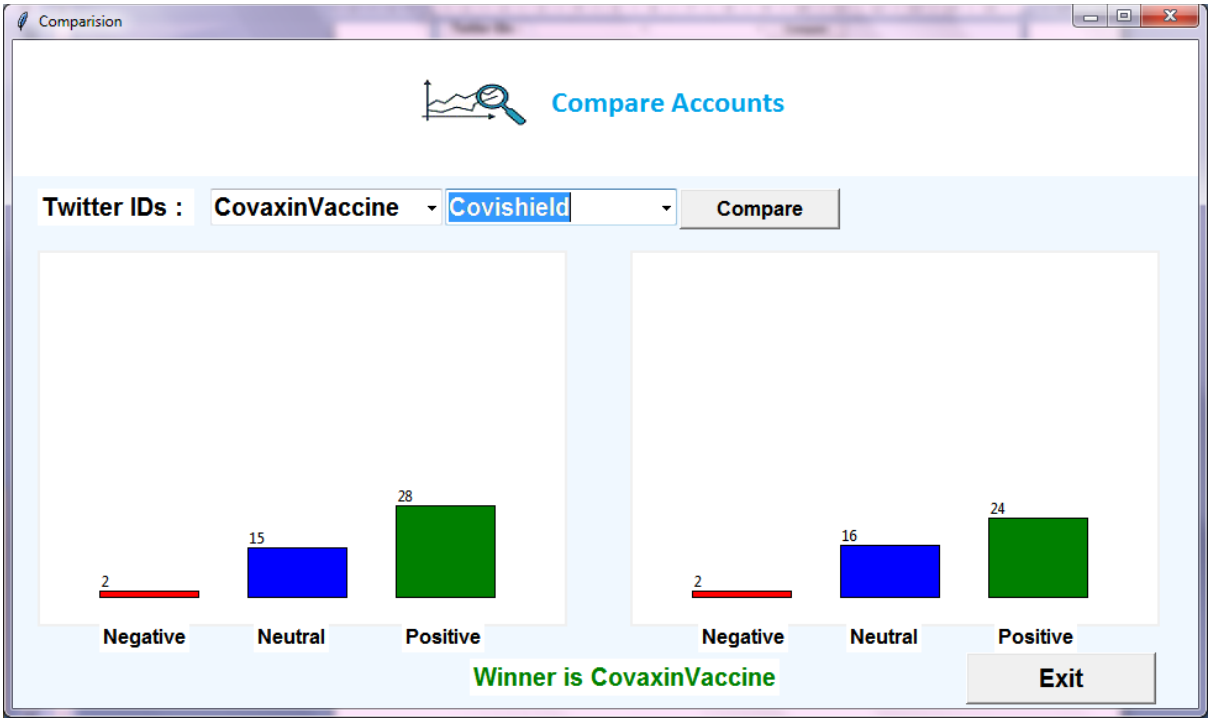

Figure 7.6 : Comparison of Accounts 1

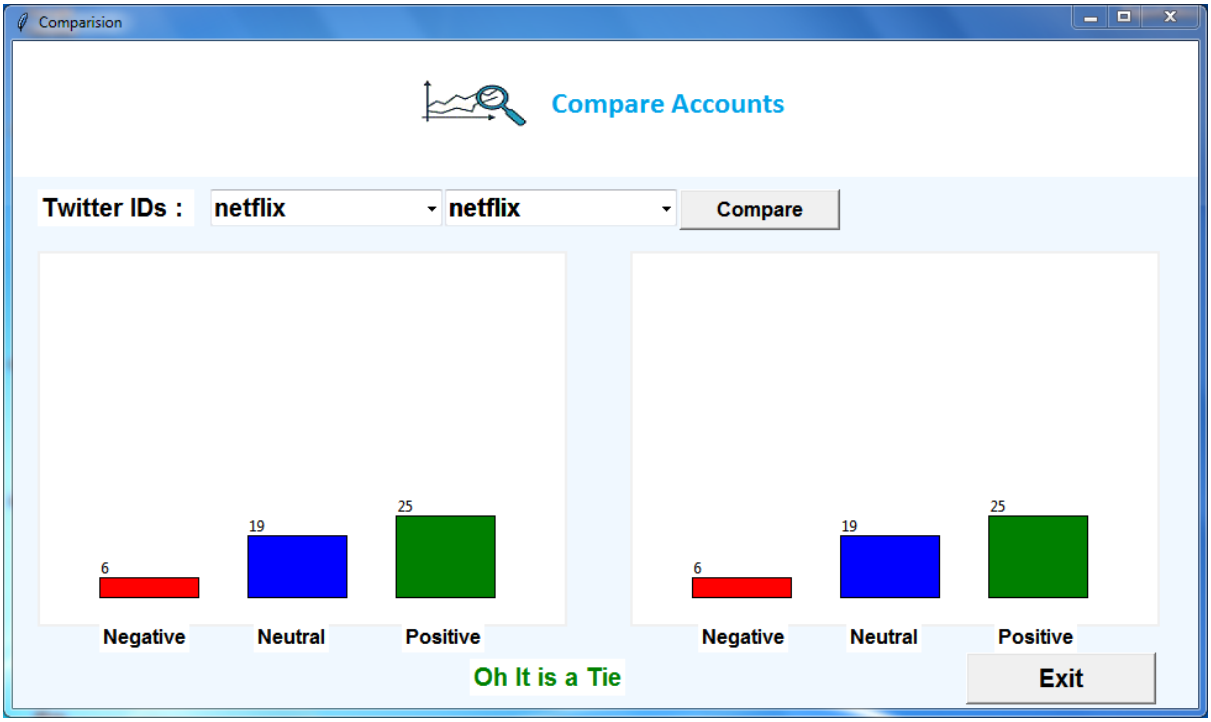

Figure 7.7 : Comparison of Accounts 2

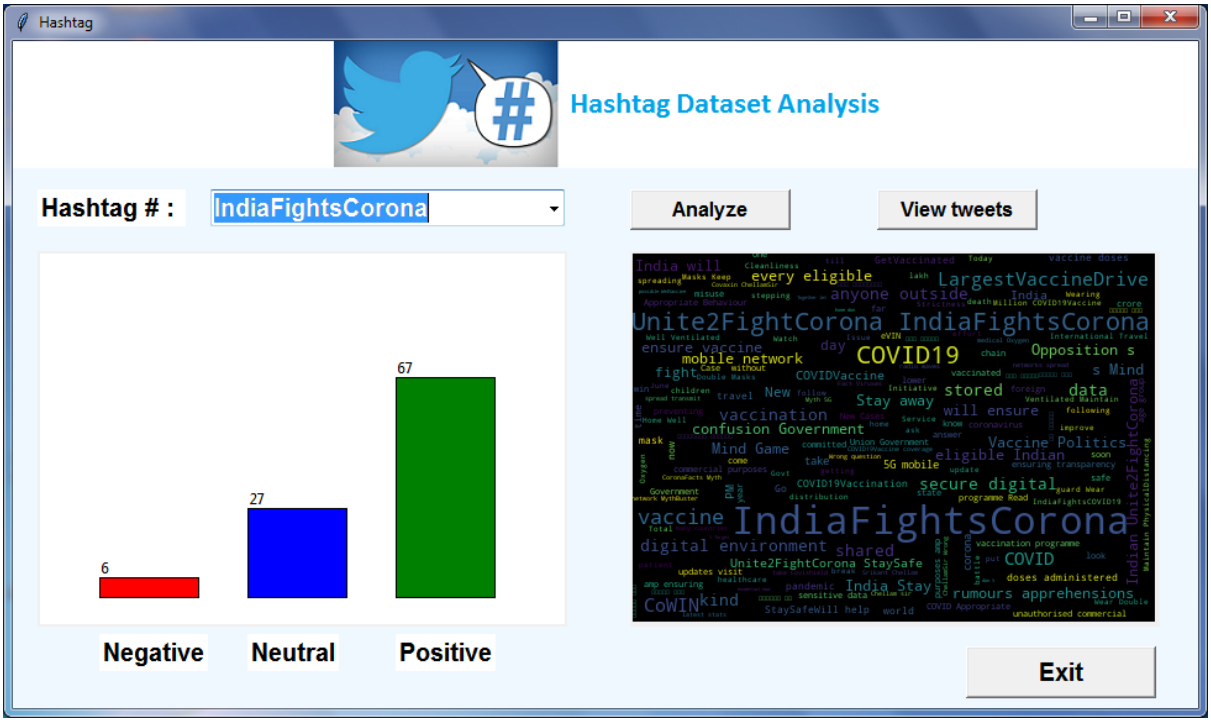

Figure 7.8 : Hashtag Analysis 1

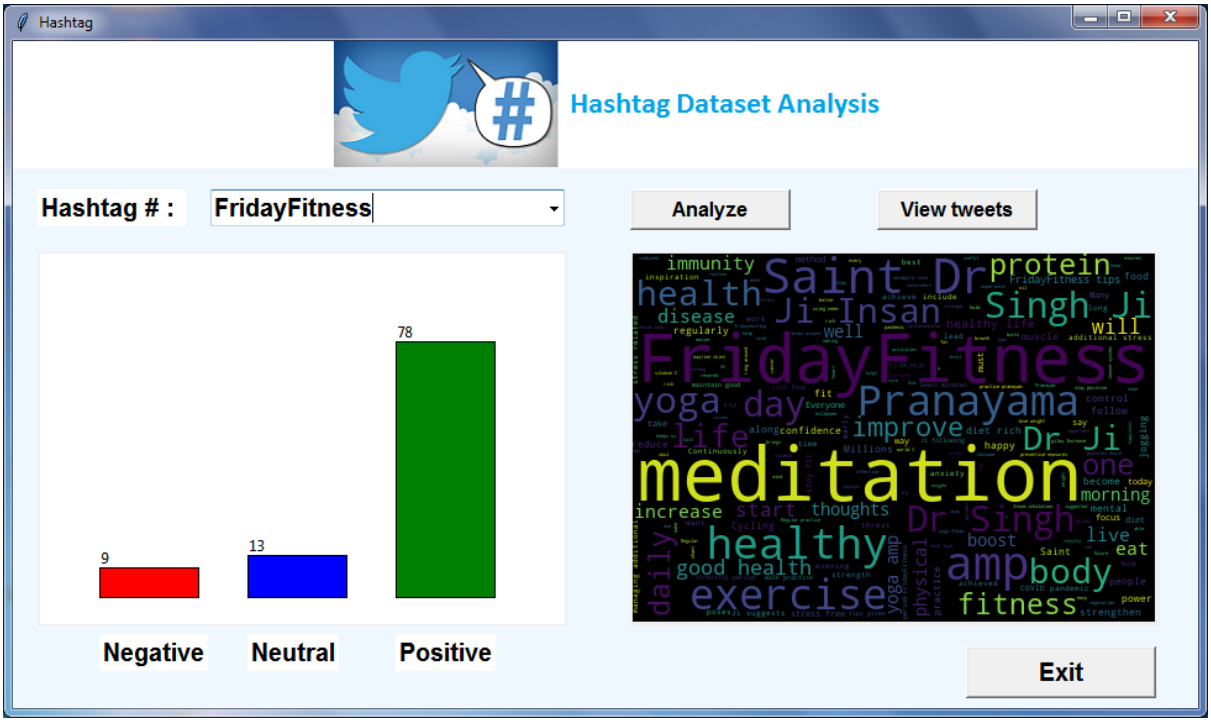

Figure 7.9 : Hashtag Analysis 2

8 CONCLUSION

The proposed system aimed to determine opinion of people on twitter, about large number of dataset (positive and negative) was collected. Normalization of dataset was done. A simple user interface was designed. Representation of decision in bar graph.

9 FUTURE ENHANCEMENT

In future work, we can extend this project to implement by using various upcoming algorithms and analyse the emotions (smiles) also to extend this using various languages and multiple fields.

APPENDIX

SOURCE CODE LISTING

Main.py

```
import os, tweepy, re
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tkinter as tk
from textblob import TextBlob
from wordcloud import WordCloud
from tkinter import *
from tkinter import messagebox, PhotoImage, Label, Text
from tkinter import ttk
from pylab import rcParams
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from PIL import ImageTk,Image


def click():
    'click function used to login into a twitter account'
    global twitterApi
    login_file = "./Settings/login.csv"
    if os.path.isfile(login_file):
        config = pd.read_csv(login_file) #read Login information from csv file
        twitterApiKey = config['twitterApiKey'][0]
        twitterApiSecret = config['twitterApiSecret'][0]
        twitterApiAccessToken = config['twitterApiAccessToken'][0]
```

```
        twitterApiAccessTokenSecret = config['twitterApiAccessTokenSecret'][0]
        auth = tweepy.OAuthHandler(twitterApiKey, twitterApiSecret)
        auth.set_access_token(twitterApiAccessToken,
                              twitterApiAccessTokenSecret)
        api = tweepy.API(auth)
        try:
            test = api.verify_credentials()
            print("auth msg", test)
            if str(test) == 'False':
                print("Error during authentication 1")
                messagebox.showerror("login error", "Please check login.csv
                                     contents and try again !!!")
            else:
                print("Authentication OK")
                twitterApi = tweepy.API(auth, wait_on_rate_limit = True)
                messagebox.showinfo("Login", "login success")
        except:
            print("Connection error")
            messagebox.showerror("login error", "Ooops.. Unable to connect
                                 twitter API's  !!!")
    else:
        print("File does not exist")
        messagebox.showerror("File not found error", "Please check login.csv
                             is present in the settings folder and try again !!!")


def search(list,n):
    for i in range(len(list)):
```

```
        if list[i] == n:
            return i, True
    return i, False

def LabelValueCorrector(Labels, Values):
    L = ["Negative", "Neutral", "Positive"]; V = [0, 0, 0]
    if len(Labels) != 0:
        for x in range(len(L)):
            index, Bool = search(Labels, L[x])
            if Bool: V[x] = Values[index]
    return L,V

def sentiment():
    def back_roots():
        'back_root funtion used to close the main window'
        roots.destroy() #close the UI

    root.destroy()
    roots = Tk()
    roots.title("Sentiment Analyser") #Sentiment Window Title
    roots.geometry("960x540")
    roots.config(bg='alice blue')

    def getvalue():
        canvas.delete("all"); canvas_word.delete("all")#Clear canvas each time.
        roots.update()
        print(mystring.get())
```

```
config = pd.read_csv("C:/Users/ADMIN/Desktop/twitter analyzer/Settings
                /login.csv")
twitterApiKey = config['twitterApiKey'][0]
twitterApiSecret = config['twitterApiSecret'][0]
twitterApiAccessToken = config['twitterApiAccessToken'][0]
twitterApiAccessTokenSecret = config['twitterApiAccessTokenSecret'][0]
auth = tweepy.OAuthHandler(twitterApiKey, twitterApiSecret)
auth.set_access_token(twitterApiAccessToken
                , twitterApiAccessTokenSecret)
twitterApi = tweepy.API(auth, wait_on_rate_limit = True)
twitterAccount = mystring.get()
tweets = tweepy.Cursor(twitterApi.user_timeline,
                    screen_name=twitterAccount,
                    count=None, since_id=None, max_id=None,
                    trim_user=True, exclude_replies=True,
                    contributor_details=False,
                    include_entities=False).items(50);
df = pd.DataFrame(data=[tweet.text for tweet in tweets],
                columns=['Tweet'])
df.head()
# Cleaning the tweets

def cleanUpTweet(txt):
    txt = re.sub(r'@[A-Za-z0-9_]+', '', txt)
    txt = re.sub(r'#', '', txt)
    txt = re.sub(r'RT : ', '', txt)
    txt = re.sub(r'https?:\/\/[A-Za-z0-9\.\/]+', '', txt)
```

```
    return txt
df['Tweet'] = df['Tweet'].apply(cleanUpTweet)
def getTextSubjectivity(txt):
    return TextBlob(txt).sentiment.subjectivity

def getTextPolarity(txt):
    return TextBlob(txt).sentiment.polarity
df['Subjectivity'] = df['Tweet'].apply(getTextSubjectivity)
df['Polarity'] = df['Tweet'].apply(getTextPolarity)
df.head(50)
df = df.drop(df[df['Tweet'] == ''].index)
df.head(50)

def getTextAnalysis(a):
    if a < 0:
        return "Negative"
    elif a == 0:
        return "Neutral"
    else:
        return "Positive"
df['Score'] = df['Polarity'].apply(getTextAnalysis)
df.head(50)
positive = df[df['Score'] == 'Positive']
print(str(positive.shape[0]/(df.shape[0])*100) +
      " % of positive tweets")
labels = df.groupby('Score').count().index.values
values = df.groupby('Score').size().values
```

```
labels, values = LabelValueCorrector(labels, values)
plt.bar(labels, values)
##We visualise matplotlib bar graph in canvas.
data = values
y_stretch = 2.65; y_gap = 20
x_stretch = 40; x_width = 80; x_gap = 50
for x, y in enumerate(data):
    x0 = x * x_stretch + x * x_width + x_gap
    y0 = canvas_bar_height - (y * y_stretch + y_gap)
    x1 = x * x_stretch + x * x_width + x_width + x_gap
    y1 = canvas_bar_height - y_gap
    if x == 0: colour = 'Red'
    elif x == 1: colour = 'Blue'
    elif x == 2: colour = 'Green'
    else: colour = 'Black'
    canvas.create_rectangle(x0, y0, x1, y1, fill=colour)
    canvas.create_text(x0+2, y0, anchor=tk.SW, text=str(y))
Label (roots, text="Negative", bg="white", fg="black", height=1,
       font="none 15 bold") .place(x=70, y=480)
Label (roots, text="Neutral", bg="white", fg="black", height=1,
       font="none 15 bold") .place(x=190, y=480)
Label (roots, text="Positive", bg="white", fg="black", height=1,
       font="none 15 bold") .place(x=310, y=480)
for index, row in df.iterrows():
    if row['Score'] == 'Positive':
        plt.scatter(row['Polarity'], row['Subjectivity'],
                    color="green")
```

```
            elif row['Score'] == 'Negative':
                plt.scatter(row['Polarity'], row['Subjectivity'],
                            color="red")
            elif row['Score'] == 'Neutral':
                plt.scatter(row['Polarity'], row['Subjectivity'],
                            color="blue")
        plt.title('Twitter Sentiment Analysis')
        plt.xlabel('Polarity')
        plt.ylabel('Subjectivity')
        objective = df[df['Subjectivity'] == 0]
        print(str(objective.shape[0]/(df.shape[0])*100) +
              " % of objective tweets")
        # Creating a word cloud
        words = ' '.join([tweet for tweet in df['Tweet']])
        wordCloud = WordCloud(width=425, height=300).generate(words)
        wordCloud.to_file("first_review.png")
        img = ImageTk.PhotoImage(Image.open("first_review.png"))
        canvas_word.create_image(0, 0, anchor=NW, image=img)
        roots.mainloop()
    Frames = Frame(height = 110, width = 990, bg = 'white')
    Frames.place(x=0, y=0)
    headerimages = PhotoImage(file=r'C:\Users\ADMIN\Desktop\twitter
analyzer
                              \sentiment.gif')
    Labels = Label(roots, borderwidth = 0, relief="raised", image=headerimage)
    Labels.place(x=310, y=0, anchor='nw')#Title Image with sentiment logo
    Label (roots, text="Twitter ID : ", bg="white", fg="black", height=1,
```

```
        font="none 15 bold") .place(x=20, y=120)
    mystring = StringVar()
    Entry(roots, textvariable = mystring, width=26, font="none 15")
                        .place(x=160, y=120) #entry textbox
    WSignUp = Button(roots, text="Analyze", width=12, height=1,
                font="none 12 bold", command=getvalue).place(x=500, y=120)
    ##Fixing canvas for bar graph.
    canvas_bar_width = 425; canvas_bar_height = 300
    canvas = tk.Canvas(roots, width=canvas_bar_width,
height=canvas_bar_height
                    , bg= 'white')
    canvas.place(x = 20, y = 170)
    ##Fixing canvas for word cloud.
    canvas_word = Canvas(roots, width = 425, height = 300)
    canvas_word.place(x = 500, y = 170)
    Button(roots, text="Exit", width=12, height=1, font="none 15 bold",
        command=back_roots) .place(x=772, y=490) #button exit
    roots.mainloop()


def comparision():
    def back_roots():
       'back_root function used to close the main window'
       rootc.destroy() #close the UI
    root.destroy()
    rootc = Tk()
    rootc.title("Comparision") #Comparision Window Title
    rootc.geometry("960x540")
```

```
    rootc.config(bg='alice blue')
    def getvalue():
        print(mystring1.get(), mystring2.get())
        config = pd.read_csv("C:/Users/ADMIN/Desktop/twitter analyzer/
                        Settings/login.csv")
    Framec = Frame(height = 110, width = 990, bg = 'white')
    Framec.place(x=0, y=0)
    headerimagec = PhotoImage(file=r'C:\Users\ADMIN\Desktop\twitter
analyzer\
                          Images\Comparision.gif')
    Labelc = Label(rootc, borderwidth = 0, relief="raised", image=headerimagec)
    Labelc.place(x=310, y=0, anchor='nw')#Title Image with comparision LOGO
    Label (rootc, text="Twitter IDs : ", bg="white", fg="black", height=1,
          font="none 15 bold") .place(x=20, y=120)
    mystring1 = StringVar()
    mystring2 = StringVar()
    Entry(rootc, textvariable = mystring1, width=15, font="none 15")
                        .place(x=160, y=120) #entry textbox1
    Entry(rootc, textvariable = mystring2, width=15, font="none 15")
                        .place(x=350, y=120) #entry textbox2
    WSignUp = Button(rootc, text="Compare", width=12, height=1,
                font="none 12 bold", command=getvalue).place(x=540, y=120)
    ##Fixing canvas for bar graphs.
    canvas_bar_width = 425; canvas_bar_height = 300
    canvas = tk.Canvas(rootc, width=canvas_bar_width,
height=canvas_bar_height,
                    bg= 'white')
```

```
    canvas.place(x = 20, y = 170)
    canvas_bar_width1 = 425; canvas_bar_height1 = 300
    canvas1 = tk.Canvas(rootc, width=canvas_bar_width1,
                        height=canvas_bar_height1, bg= 'white')
    canvas1.place(x = 500, y = 170)
    Button(rootc, text="Exit", width=12, height=1, font="none 15 bold",
           command=back_roots) .place(x=772, y=490) #button exit
    rootc.mainloop()
def hashtag():
    def back_roots():
        'back_root function used to close the main window'
        rooth.destroy() #close the UI
    root.destroy()
    rooth = Tk()
    rooth.title("Hashtag") #Hashtag Window Title
    rooth.geometry("960x540")
    rooth.config(bg='alice blue')
    def getvalue():
        canvas.delete("all"); canvas_word.delete("all")#Clear canvas each time.
        rooth.update()
        def scrape(words, numtweet):
            db = pd.DataFrame(columns=['username', 'description', 'location',
            'following', 'followers', 'totaltweets', 'retweetcount', 'Tweet',
                                       'hashtags'])
            tweets = tweepy.Cursor(api.search, q=words, lang="en",
                                   tweet_mode='extended').items(numtweet)
            list_tweets = [tweet for tweet in tweets]
```

```
        i = 1
        for tweet in list_tweets:
            username = tweet.user.screen_name
            description = tweet.user.description
            location = tweet.user.location
            following = tweet.user.friends_count
            followers = tweet.user.followers_count
            totaltweets = tweet.user.statuses_count
            retweetcount = tweet.retweet_count
            hashtags = tweet.entities['hashtags']
            try:
                text = tweet.retweeted_status.full_text
            except AttributeError:
                text = tweet.full_text
            hashtext = list()
            for j in range(0, len(hashtags)):
                hashtext.append(hashtags[j]['text'])
            ith_tweet = [username, description, location, following,
                         followers, totaltweets, retweetcount, text, hashtext]
            db.loc[len(db)] = ith_tweet
            i = i+1
        db.to_csv('scraped_tweets.csv')
    print(Hashtags.get())
    messagebox.showinfo("Title.....", Hashtags.get())
    consumer_key = "OIzAvzLwIV0rvIILvPDYxr9Op"
    consumer_secret =
"4aRXzlqqe0ksvyEdIZmre21P1isFZ46MrH3XBRfk390yrRhM6I"
```

```
    access_key = "1342826884103503876-
pRU6yuw8B8cmhecsjEi0quZ89PmfxS"
    access_secret =
"CxFRJPlOtjDdsy5tZE1wuCBMmduTOBdkM4g0wUgAjeBaA"
    auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_key, access_secret)
    api = tweepy.API(auth)
    scrape(Hashtags.get(), 100) # 100 is number of tweet
    print('Scraping has completed!')
    def filter_columns(filename):
        ''' This will filter the columns in reqlst in the csv file '''
        data_frame = pd.read_csv(filename, low_memory=False,
                                 error_bad_lines=False)
        reqlst = [8] #Filter column Tweet from CSV file Delete all others
        final_df = data_frame.iloc[:, reqlst]
        final_df.to_csv(filename, index=False)
    filter_columns('scraped_tweets.csv')#Filter
    df = pd.read_csv('scraped_tweets.csv')#Read Dataframe from csv
    def cleanUpTweet(txt):
        txt = re.sub(r'@[A-Za-z0-9_]+', '', txt)
        txt = re.sub(r'#', '', txt)
        txt = re.sub(r'RT : ', '', txt)
        txt = re.sub(r'https?:\/\/[A-Za-z0-9\.\/]+', '', txt)
        return txt
    df['Tweet'] = df['Tweet'].apply(cleanUpTweet)
    def getTextSubjectivity(txt):
        return TextBlob(txt).sentiment.subjectivity
```

```
def getTextPolarity(txt):
    return TextBlob(txt).sentiment.polarity
df['Subjectivity'] = df['Tweet'].apply(getTextSubjectivity)
df['Polarity'] = df['Tweet'].apply(getTextPolarity).
df.head(50)
df = df.drop(df[df['Tweet'] == ''].index)
df.head(50)
def getTextAnalysis(a):
    if a < 0:
        return "Negative"
    elif a == 0:
        return "Neutral"
    else:
        return "Positive"
df['Score'] = df['Polarity'].apply(getTextAnalysis)
df.head(50)
print("test here")
positive = df[df['Score'] == 'Positive']
print(str(positive.shape[0]/(df.shape[0])*100) +
      " % of positive tweets")
labels = df.groupby('Score').count().index.values
values = df.groupby('Score').size().values
labels, values = LabelValueCorrector(labels, values)
plt.bar(labels, values)

data = values
y_stretch = 2.65; y_gap = 20
```

```
x_stretch = 40; x_width = 80; x_gap = 50
for x, y in enumerate(data):
    x0 = x * x_stretch + x * x_width + x_gap
    y0 = canvas_bar_height - (y * y_stretch + y_gap)
    x1 = x * x_stretch + x * x_width + x_width + x_gap
    y1 = canvas_bar_height - y_gap
    if x == 0: colour = 'Red'
    elif x == 1: colour = 'Blue'
    elif x == 2: colour = 'Green'
    else: colour = 'Black'
    canvas.create_rectangle(x0, y0, x1, y1, fill=colour)
    canvas.create_text(x0+2, y0, anchor=tk.SW, text=str(y))
Label (rooth, text="Negative", bg="white", fg="black", height=1,
       font="none 15 bold") .place(x=70, y=480)
Label (rooth, text="Neutral", bg="white", fg="black", height=1,
       font="none 15 bold") .place(x=190, y=480)
Label (rooth, text="Positive", bg="white", fg="black", height=1,
       font="none 15 bold") .place(x=310, y=480)
for index, row in df.iterrows():
    if row['Score'] == 'Positive':
        plt.scatter(row['Polarity'], row['Subjectivity'], color="green")
    elif row['Score'] == 'Negative':
        plt.scatter(row['Polarity'], row['Subjectivity'], color="red")
    elif row['Score'] == 'Neutral':
        plt.scatter(row['Polarity'], row['Subjectivity'], color="blue")


plt.title('Twitter Sentiment Analysis')
```

```
        plt.xlabel('Polarity')
        plt.ylabel('Subjectivity')
        objective = df[df['Subjectivity'] == 0]
        print(str(objective.shape[0]/(df.shape[0])*100) +
              " % of objective tweets")
        words = ' '.join([tweet for tweet in df['Tweet']])
        wordCloud = WordCloud(width=425, height=300).generate(words)
        wordCloud.to_file("first_review.png")
        img = ImageTk.PhotoImage(Image.open("first_review.png"))
        canvas_word.create_image(0, 0, anchor=NW, image=img)
        rooth.mainloop()
    Frameh = Frame(height = 103, width = 990, bg = 'white')
    Frameh.place(x=0, y=0)
    headerimageh = PhotoImage(file=r'C:\Users\ADMIN\Desktop\twitter
analyzer\
                              Images\Hashtag.gif')
    Labelh = Label(rooth, borderwidth = 0, relief="raised",
image=headerimageh)
    Labelh.place(x=260, y=0, anchor='nw')#Title Image with hashtag LOGO
    Label (rooth, text="Hashtag # : ", bg="white", fg="black", height=1,
           font="none 15 bold") .place(x=20, y=120)
    Hashtags = StringVar()
    Entry(rooth, textvariable = Hashtags, width=26, font="none 15")
           .place(x=160, y=120) #entry textbox
    WSignUp = Button(rooth, text="Analyze", width=12, height=1,
                     font="none 12 bold", command=getvalue).place(x=500, y=120)
    canvas_bar_width = 425; canvas_bar_height = 300
```

```
    canvas = tk.Canvas(rooth, width=canvas_bar_width,
                       height=canvas_bar_height, bg= 'white')
    canvas.place(x = 20, y = 170)
    canvas_word = Canvas(rooth, width = 425, height = 300)
    canvas_word.place(x = 500, y = 170)
    Button(rooth, text="Exit", width=12, height=1, font="none 15 bold",
          command=back_roots) .place(x=772, y=490) #button exit
    rooth.mainloop()

def close_root():
    'close-root function close the program'
    root.destroy() #close the UI
    #log out of twitter
    os._exit(0) #Forcefully quit python

root = Tk()
root.title("Twitter Analyser") #Main Window Title
root.geometry("960x540")
root.config(bg='alice blue')
Frame1 = Frame(height = 105, width = 990, bg = 'white')#White frame
Frame1.place(x=0, y=0)
headerimage = PhotoImage(file=r'C:\Users\ADMIN\Desktop\twitter analyzer\
                         Label.gif')
Label1 = Label(root, borderwidth = 0, relief="raised", image=headerimage)
Label1.place(x=310, y=0, anchor='nw')#Title Image with twitter LOGO
Label (root, text="Click Here to Login : ", bg="white", fg="black", height=2,
       font="none 15 bold") .place(x=100, y=150)
```

```
Button(root, text="LOGIN", width=12, height=2, font="none 15 bold",
    command=click) .place(x=350, y=150)
Button(root, text="Individual", width=14, height=2, font="none 15 bold",
    command=sentiment) .place(x=100, y=350)
Button(root, text="Comparision", width=14, height=2, font="none 15 bold",
    command=comparision) .place(x=350, y=350)
Button(root, text="Hashtag", width=14, height=2, font="none 15 bold",
    command=hashtag) .place(x=600, y=350)
Button(root, text="EXIT", width=12, height=1, font="none 15 bold",
    command=close_root) .place(x=790, y=490)
twitterApi = "
root.mainloop()
```

REFERENCE

[1] Asma Belhadi, Youcef Djenouri, Jerry Chun-Wei Lin, Chongsheng Zhang, Alberto Cano, “Exploring Pattern Mining Algorithms for Hashtag Retrieval Problem”, 2020.

[2] Y. Gong, Q. Zhang, and X. Huang, ``Hashtag recommendation for multimodal microblog posts," Neurocomputing, vol. 272, pp. 170177, Jan. 2018.

[3] G. G. Chowdhury, Introduction to Modern Information Retrieval. Facet Publishing, 2010.

[4] M. Efron, ``Hashtag retrieval in a microblogging environment," in Proc. 33rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr., 2010, pp. 787788.

[5] Cher Han Lau, Xiaohui Tao, Dian Tjondronegoro, Yue Feng. “Retrieving information for micro blog using pattern mining and relevance feedback”, 2018.

[6] J.Han, J.Rei, Y.Yin. “Mining frequent patterns without candidate generation”, 2019.

[7] Sahar A. El Rahman, Feddah Alhumaidi AlOtaibi, Wejdan Abdullah AlShehri. “Sentiment Analysis of Twitter Data”, 2019.

[8] Bharat R. Naiknaware, Seema S. Kawathekar, “Prediction of 2019 Indian Election Using Sentiment Analysis”, 2019.

[9] ZulfadzliDrus, HaliyanaKhalid, “Sentiment Analysis in Social Media and Its Application: Systematic Literature Review”, 2020.

[10] H. K. Azad and A. Deepak, "Query expansion techniques for information retrieval: A survey", Inf. Process. Manage., vol. 56, no. 5, pp. 1698-1735, Sep. 2019.

[11] Q. Li, S. Shah, R. Fang, A. Nourbakhsh and X. Liu, "Hashtag mining: Discovering relationship between health concepts and hashtags" in Public Health Intelligence and the Internet, Springer, pp. 75-85, 2017.

[12] Y. Djenouri, A. Belhadi and P. Fournier-Viger, "Extracting useful knowledge from event logs: A frequent itemset mining approach", Knowl.-Based Syst., vol. 139, pp. 132-148, Jan. 2018.

[13] Y. Djenouri, H. Drias and A. Bendjoudi, "Pruning irrelevant association rules using knowledge mining", Int. J. Bus. Intell. Data Mining, vol. 9, no. 2, pp. 112-144, 2014.

[14] H. Belhadi, K. Akli-Astouati, Y. Djenouri and J. C.-W. Lin, "Exploring pattern mining for solving the ontology matching problem", Proc. Eur. Conf. Adv. Databases Inf. Syst., pp. 85-93, 2019.

[15] Y. Djenouri, Z. Habbas and D. Djenouri, "Data mining-based decomposition for solving the MAXSAT problem: Toward a new approach", IEEE Intell. Syst., vol. 32, no. 4, pp. 48-58, 2017.

[16] Y. Djenouri, Z. Habbas, D. Djenouri and P. Fournier-Viger, "Bee swarm optimization for solving the MAXSAT problem using prior knowledge", Soft Comput, vol. 23, no. 9, pp. 3095-3112, May 2019.

[17] Y. Djenouri, A. Belhadi, P. Fournier-Viger and J. C.-W. Lin, "Fast and effective cluster-based information retrieval using frequent closed itemsets", Inf. Sci., vol. 453, pp. 154-167, Jul. 2018.

[18] Y. Djenouri, A. Belhadi and R. Belkebir, "Bees swarm optimization guided by data mining techniques for document information retrieval", Expert Syst. Appl., vol. 94, pp. 126-136, Mar. 2018.

[19] C. H. Lau, X. Tao, D. Tjondronegoro and Y. Li, "Retrieving information from microblog using pattern mining and relevance feedback", Proc. Int. Conf. Data Knowl. Eng., pp. 152-160, 2019.

[20] H.-J. Choi and C. H. Park, "Emerging topic detection in Twitter stream based on high utility pattern mining", Expert Syst. Appl., vol. 115, pp. 27-36, Jan. 2019.