Analyzing Public Opinion on Twitter through Big Data Techniques

BDA mini project submitted in partial fulfillment of the requirements
of the degree of

B. E. Computer Engineering

By

Justin Fernandes 11 222036
Valour Moraes 18 222063
Bhavik Solanki 21 222109

Name of the Guide: Ms. Jayashri Mittal
Designation: Assistant Professor

Department of Computer Engineering
St. Francis Institute of Technology
(Engineering College)

An Autonomous Institute, Affiliated to University of Mumbai
2025-2026

CERTIFICATE

This is to certify that the mini project entitled "**Analyzing Public Opinion on Twitter through Big Data Techniques"** is a bonafide work of **Justin Fernandes(11), Valour Moraes (18), Bhavik Solanki (21)** submitted to the University of Mumbai in partial fulfillment of the requirement for the BDA subject in final year of Computer Engineering.

Ms. Jayashri Mittal
Guide

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Justin Fernandes (11)

Valour Moraes (18)

Bhavik Solanki (21)

Date:

Abstract

In today's digital age, social media platforms such as Twitter serve as major hubs for public opinion, where millions of users express their views daily on topics ranging from politics and entertainment to sports and brands. This project, *Analyzing Public Opinion on Twitter through Big Data Techniques*, aims to classify tweets from a user's profile into positive, negative, or neutral sentiments. The system employs Python along with Tweepy for data extraction, NLTK and VADER (Valence Aware Dictionary for Sentiment Reasoning) for text processing and sentiment classification, and visualization tools such as Streamlit, Matplotlib, and WordCloud to present the results. By integrating Natural Language Processing (NLP) and Big Data Analytics (BDA), the project effectively interprets large volumes of real-world social media data, providing valuable insights into public emotions, opinions, and emerging trends.

Contents

List of Figures

List of Abbreviations

Sr. No.	Abbreviation	Expanded form
1	API	Application Programming Interface
2	BERT	Bidirectional Encoder Representations from Transformers
3	BDA	Big Data Analytics
4	CSV	Comma-Separated Values
5	DFD	Data Flow Diagram
6	LSTM	Long Short-Term Memory
7	NLP	Natural Language Processing
8	NLTK	Natural Language Toolkit
9	VADER	Valence Aware Dictionary for Sentiment Reasoning
10	IDE	Integrated Development Environment
11	UI	User Interface
12	URL	Uniform Resource Locator

Chapter 1
Introduction

1.1 Description

Social media platforms like Twitter produce large volumes of textual data daily, where users share their opinions and experiences. This data represents an opportunity for sentiment analysis, which determines whether a piece of text expresses a positive, negative, or neutral emotion.
Analyzing Public Opinion on Twitter through Big Data Techniques involves automatically processing this data to determine public opinions toward specific topics or individuals.

This project develops an application that fetches tweets from a specified Twitter user using the Twitter API, processes the text, and determines the sentiment using **VADER**. The project helps visualize user sentiment in an interactive and simple interface.

1.2 Problem Formulation

With the exponential increase in online communication, especially on platforms like Twitter, manual sentiment analysis is practically impossible. Millions of tweets are generated daily, making it difficult to assess the general tone or opinion of users toward a particular topic or individual.

Some key challenges that motivated this project include:

- **Unstructured Data:** Tweets contain informal language, abbreviations, emojis, URLs, hashtags, and slang that make automated analysis complex.
- **High Volume of Data:** Processing and analyzing large quantities of real-time tweets requires efficient tools and algorithms.
- **Ambiguity of Language:** Words can have multiple meanings based on context (e.g., sarcasm or irony), which complicates sentiment classification.

To address these challenges, a system is required that can:

- Fetch tweets directly from Twitter in real-time.
- Preprocess and clean unstructured text data.
- Classify sentiments automatically using a reliable model.
- Provide meaningful, easy-to-interpret visual results.

The **Analyzing Public Opinion on Twitter through Big Data Techniques** project fulfills these objectives using a simple, scalable, and efficient architecture that combines NLP and visualization technologies.

1.3 Motivation

The rise of digital media has transformed the way individuals and organizations engage with the world. Every opinion shared on social media holds potential insights that can influence public relations, business strategies, and political campaigns.

Monitoring these sentiments manually is impossible due to the massive volume of online content. Hence, automating this process not only saves time but also provides analytical advantages.
Some motivating factors for developing this project include:

- To gain practical understanding of **NLP**, **Data Mining**, and **Big Data Analytics**.
- To design a real-world application that can assist businesses in understanding customer opinions.
- To explore social media as a valuable data source for trend detection and emotional insight.

From a technical perspective, this project also provides an opportunity to understand **API integration**, **data visualization**, and **data-driven decision-making** in real-time environments.

1.4 Proposed Solution

The proposed system is a sentiment analysis tool that automatically collects, processes, and evaluates tweets from a user's Twitter profile. The solution integrates several components to perform this analysis efficiently.

1) **Data Collection Layer:**
   Uses **Tweepy** to connect to the **Twitter API** and fetch recent tweets from the given username. This ensures that the system can access live, authentic data directly from the Twitter platform.
2) **Preprocessing Layer:**
   Removes unwanted elements like URLs, mentions, hashtags, punctuation marks, emojis, and stop words. Text normalization is also performed (lowercasing, tokenization, etc.) to improve analysis accuracy.
3) **Analysis Layer:**
   Applies the **VADER sentiment analyzer**, a lexicon and rule-based model optimized for

social media text. It calculates a *compound score* representing the overall sentiment of each tweet and classifies it as *positive*, *negative*, or *neutral*.

4) **Visualization Layer:**
   Uses **Streamlit**, **Matplotlib**, and **WordCloud** to display the analysis results. The user can view sentiment distribution through pie charts, bar graphs, and a word cloud of the most frequent terms.
5) **Export Layer:**
   Allows users to export analyzed results into a **CSV file** for offline use or reporting.

This architecture ensures efficiency, modularity, and scalability, allowing future extensions such as real-time streaming and multilingual analysis.

1.5 Scope of The Project

The **scope** of this project defines its boundaries, features, and applications.

In Scope:

- Fetching tweets using Twitter API through Tweepy.
- Preprocessing text and removing unwanted tokens.
- Sentiment classification using the VADER model.
- Interactive visualization using Streamlit.
- Exporting analyzed data to CSV format.

Out of Scope:

- Real-time tweet streaming and sentiment updates.
- Deep learning-based approaches such as BERT or LSTM.
- Sentiment detection for non-English languages.

Applications:

- Business intelligence for brand reputation tracking.
- Political sentiment analysis.
- Social trend and opinion monitoring.
- Academic research on public emotion and data analytics.

Chapter 2
Review of Literature

2.1 Introduction

The domain of **Sentiment Analysis** has received significant attention in the last decade due to the explosion of user-generated content on social media. Millions of opinions expressed daily on platforms like **Twitter**, **Facebook**, and **Reddit** form an immense source of data for analyzing public mood and perception.

This chapter explores the existing work, research papers, algorithms, and tools related to sentiment analysis. It also provides a comparison between **machine learning**, **lexicon-based**, and **deep learning** approaches, and explains why the **VADER** model was chosen for this project.

2.2 Background

Sentiment analysis, a subfield of **Natural Language Processing (NLP)** and **Text Mining**, focuses on identifying the emotional tone conveyed in text data. It can be performed at multiple levels:

- **Document Level:** Determines the overall sentiment of an entire document.
- **Sentence Level:** Classifies each sentence as positive, negative, or neutral.
- **Aspect Level:** Analyzes sentiment toward a specific feature or topic within a sentence.

For Twitter data, **sentence-level analysis** is most appropriate because tweets are short (limited to 280 characters) and often self-contained.

2.3 Existing Approaches to Sentiment Analysis

2.3.1 Machine Learning-Based Approaches

In traditional sentiment analysis, machine learning models are trained using labeled datasets to classify text into sentiment categories.
Common algorithms include:

1) **Naïve Bayes Classifier:**
   Works on the principle of conditional probability and assumes feature independence. It performs well for text classification but requires a large labeled dataset.

2) **Support Vector Machine (SVM):**
   Creates a separating hyperplane in high-dimensional space to classify text. SVMs are accurate but computationally heavy for large datasets.
3) **Logistic Regression:**
   A supervised method that predicts probabilities of categorical outcomes. It's interpretable but limited for complex linguistic structures.

While these models can perform well, they need extensive preprocessing and training data, which is not always feasible for dynamic and real-time Twitter data.

2.3.2 Lexicon-Based Approaches

Lexicon-based models rely on predefined lists of words tagged with sentiment scores (positive, negative, neutral).
When a text is processed, each word's score is summed to calculate the overall sentiment.
Popular lexicon-based systems include:

1. **SentiWordNet:** Provides polarity values for each WordNet term.
2. **AFINN:** Assigns numeric scores to English words based on emotional valence.
3. **VADER (Valence Aware Dictionary and sEntiment Reasoner):** Specifically tuned for social media text, considers capitalization, punctuation, emojis, and slang.

Lexicon-based methods are advantageous because they do not require labeled data and can work directly on new datasets. However, they may not handle context, sarcasm, or domain-specific vocabulary effectively.

2.3.3 Deep Learning-Based Approaches

Recent years have seen the rise of deep learning models such as:

1) **Convolutional Neural Networks (CNN):** Used to capture spatial relationships between words.
2) **Recurrent Neural Networks (RNN) and LSTM:** Efficient in capturing long-term dependencies in text sequences.
3) **BERT (Bidirectional Encoder Representations from Transformers):** Provides state-of-the-art contextual understanding by processing text bidirectionally.
4) While these models offer higher accuracy, they require large computational resources, huge labeled datasets, and GPU support—making them unsuitable for small-scale or academic projects with limited infrastructure.

2.5 Overview of VADER Sentiment Analysis Model

VADER (Valence Aware Dictionary and sEntiment Reasoner) is a **lexicon and rule-based sentiment analysis tool** that performs exceptionally well on social media text. It was

introduced by **C.J. Hutto and Eric Gilbert in 2014** in the paper *"VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text."*

Key Features of VADER:

- Recognizes **capitalization, punctuation, emojis, and emoticons**.
- Understands **intensifiers and negations** (e.g., "very good" > "good"; "not good" = negative).
- Returns a **compound score** between –1 (negative) and +1 (positive).
- Lightweight and doesn't require model training.
- Integrated with **NLTK**, making it easy to use in Python projects.

The **compound score** is the normalized sum of the valence scores of each word.
Interpretation:
Compound ≥ 0.05 → Positive
Compound ≤ –0.05 → Negative
Otherwise → Neutral

VADER's efficiency, interpretability, and social-media tuning make it ideal for real-time Twitter analysis.

2.6 Related Work

1. **Hutto & Gilbert (2014)** – Proposed VADER, proving superior accuracy over machine learning classifiers for social media datasets.
2. **Saif et al. (2016)** – Developed semantic-enhanced models using word embeddings to improve polarity detection accuracy.
3. **Zhang et al. (2018)** – Applied deep learning (LSTM) for multilingual sentiment analysis, showing high accuracy but requiring significant resources.

These studies collectively demonstrate the evolution of sentiment analysis techniques—from basic machine learning to rule-based and deep learning models.

2.7 Tools and Technologies Identified

From the literature, the following tools were identified as effective for this project:

- **Python:** The core programming language due to its robust NLP and visualization libraries.
- **NLTK (Natural Language Toolkit):** Provides the VADER sentiment analyzer and text preprocessing utilities.
- **Tweepy:** Used for connecting to the Twitter API to fetch tweets.
- **Streamlit:** For building an interactive, web-based interface.
- **Matplotlib & WordCloud:** For visualization of sentiment and keyword frequency.
- **Pandas & NumPy:** For data manipulation and statistical analysis.

Chapter 3
System Analysis

3.1 Functional Requirements

Functional requirements define what the system should do — the tasks, services, and operations performed by the application.

1. **User Input Module:**
   The system should allow the user to enter a valid Twitter username. It must validate the input and handle errors such as invalid usernames, absence of tweets, or API issues.
2. **Tweet Retrieval Module:**
   The system should connect to the Twitter API using Tweepy to fetch recent tweets. Retweets and replies should be excluded unless the user specifies otherwise. The system should also retrieve tweet metadata, including timestamp, text, and ID.
3. **Preprocessing Module:**
   The system should clean the tweet text by removing special characters, URLs, hashtags, mentions, and punctuation. It should convert all text to lowercase, remove stopwords, and prepare the data for sentiment analysis.
4. **Sentiment Analysis Module:**
   The system should perform sentiment scoring using the VADER sentiment analyzer. Each tweet should be categorized as Positive, Negative, or Neutral, and the sentiment score along with its label should be stored.
5. **Visualization Module:**
   The system should display sentiment distributions using bar or pie charts. It should show time-series trends of sentiments over time and generate a word cloud of frequently used words. All results should be displayed interactively on the Streamlit interface.
6. **Export and Reporting Module:**
   The system should allow users to export the analyzed tweet data in CSV format. It should also support real-time refresh and update of results.

3.2 Non Functional Requirements:

3.2.1 Performance Requirements

- The system should retrieve and analyze tweets within a few seconds (depending on network and API limits).
- Handle at least 100–200 tweets per request efficiently.
- Streamlit interface should load within 5 seconds after launching.
- Keep resource usage (RAM and CPU) minimal during operation.

- Visualizations should render interactively without lag.

3.2.2 Software Quality Attributes

- Usability: Simple and intuitive interface for non-technical users.
- Reliability: Handle API failures or missing data gracefully.
- Maintainability: Modular code structure for easier maintenance or extension.
- Scalability: Support future integration of other APIs or advanced ML models.
- Portability: Deployable across platforms (local, cloud, Streamlit sharing).
- Security: Store sensitive credentials (API keys) securely using environment variables or Streamlit secrets.

3.3 Specific Requirements:

Hardware :

- Processor: Intel Core i3 or higher
- RAM: Minimum 4 GB (8 GB recommended)
- Storage: At least 500 MB free space
- Internet Connection: Required for accessing Twitter API and fetching tweets
- Display: Minimum resolution 1366×768

Software :

- Operating System: Windows / Linux / macOS
- Programming Language: Python 3.8 or higher
- Framework / Libraries: Streamlit, Tweepy, NLTK (VADER), Pandas, Matplotlib, WordCloud
- API Access: Valid Twitter Developer Account and Bearer Token
- IDE / Tools: VS Code / PyCharm / Jupyter Notebook

3.4 Use-Case Diagrams and description

The Use Case Diagram illustrates the interaction between different actors and the functionalities of the Analyzing Public Opinion on Twitter through Big Data Techniques System built using Streamlit. The system is designed to analyze sentiments expressed in tweets by fetching data from the X (Twitter) API, processing it, and presenting analytical insights through visualizations.

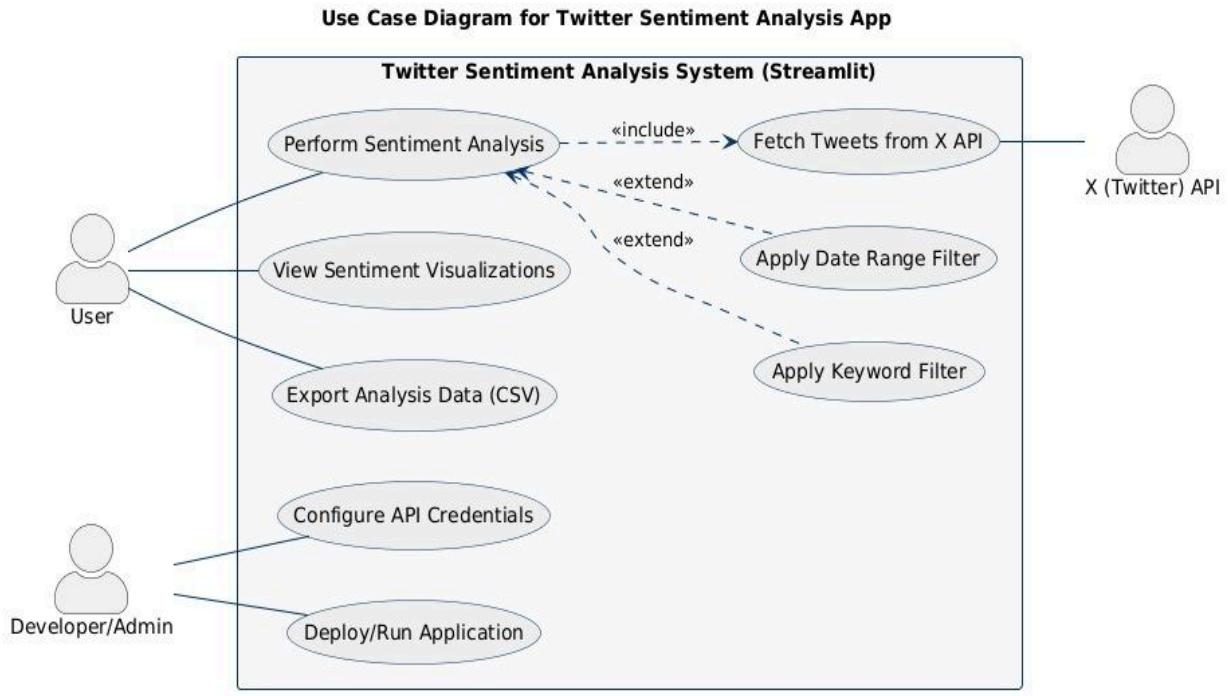

Figure 3.1: Use case diagram for Analyzing Public Opinion on Twitter through Big Data Techniques

Actors and Their Roles:

- **User**
  - The **User** interacts with the application to analyze public sentiment on specific topics or hashtags.
  - Primary actions include:
    i. **Perform Sentiment Analysis** – Initiates the process of collecting tweets and analyzing their sentiments (positive, negative, neutral).
    ii. **View Sentiment Visualizations** – Views graphical representations (charts, graphs) of sentiment distribution and trends.
    iii. **Export Analysis Data (CSV)** – Exports processed sentiment data for external use or further analysis.

- **Developer/Admin**
  - Responsible for setting up and maintaining the application.
  - Primary actions include:
    i. **Configure API Credentials** – Sets up authentication keys required to access the X (Twitter) API.
    ii. **Deploy/Run Application** – Launches and monitors the Streamlit application.

- **X (Twitter) API**
  - An external system that provides tweet data to the application.

- Communicates with the “Fetch Tweets from X API” use case to supply the required tweet datasets.

Use Case Relationships

- **Perform Sentiment Analysis**
  a. **Includes → Fetch Tweets from X API**:
     Fetching tweets is a mandatory step before performing sentiment analysis.
  b. **Extends → Apply Date Range Filter**:
     Users can optionally narrow analysis results to a specific time period.
  c. **Extends → Apply Keyword Filter**:
     Users can optionally filter tweets based on specific keywords or hashtags.

System Overview

The **Analyzing Public Opinion on Twitter through Big Data Techniques System (Streamlit)** integrates real-time tweet fetching, sentiment classification, and data visualization. It provides users with intuitive insights into public opinions on various topics. The admin ensures smooth system operation by managing configurations and deployments.

This diagram effectively captures how the user and admin interact with the system and how the system communicates with the X (Twitter) API. It highlights core functionalities such as sentiment analysis, visualization, and export features, ensuring a clear understanding of the system’s operational flow and scope.

Chapter 4

Analysis Modeling

4.1 Class Diagram

The **Class Diagram** represents the structural design of the **Analyzing Public Opinion on Twitter through Big Data Techniques Application**, highlighting its main classes, their attributes, methods, and relationships. It explains how various components interact to perform sentiment analysis and visualization.

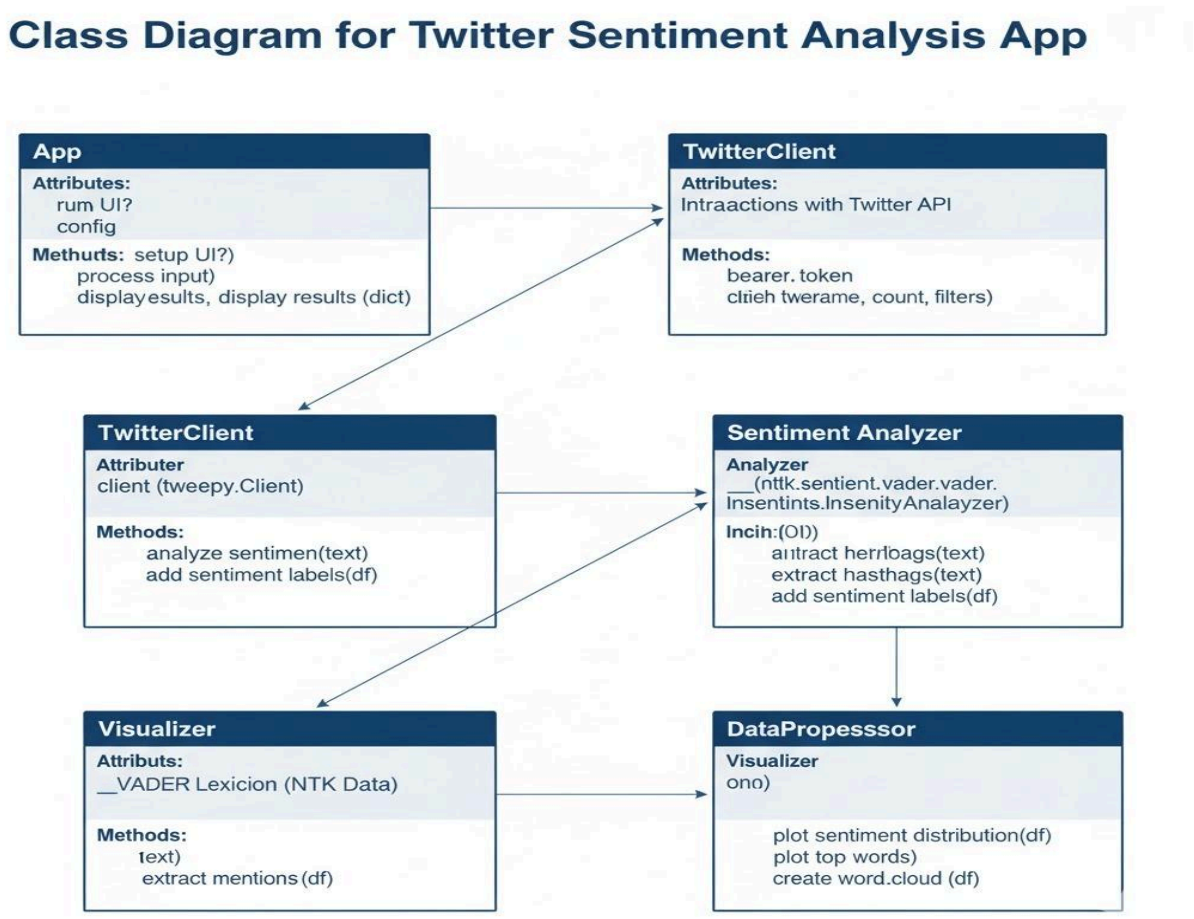

Figure 4.1: Class diagram for Analyzing Public Opinion on Twitter through Big Data Techniques system

Class Descriptions

1. **App Class**
   - **Purpose:** Acts as the main controller for initializing and running the Streamlit application.
   - **Attributes:**
     i. runUI: Launches the graphical interface.

ii. config: Stores configuration details such as API credentials or UI settings.

- **Methods:**
  i. setupUI(), processInput(), displayResults(): Manage user inputs and display analytical results.

2. **TwitterClient Class**
   - **Purpose:** Handles interactions with the **X (Twitter) API** using libraries like Tweepy.
   - **Attributes:**
     i. client (tweepy.Client): Interface to connect with the API.
   - **Methods:**
     i. fetchTweets(): Retrieves tweets based on filters.
     ii. analyzeSentiment(text): Processes and sends data for sentiment classification.
     iii. addSentimentLabels(df): Adds sentiment results to the data frame.

3. **SentimentAnalyzer Class**
   - **Purpose:** Performs the actual sentiment classification using **VADER (Valence Aware Dictionary and sEntiment Reasoner)**.
   - **Methods:**
     i. extractHashtags(text), analyzeSentiment(text), addSentimentLabels(df): Process text and assign sentiment polarity (positive, neutral, negative).

4. **Visualizer Class**
   - **Purpose:** Generates graphical representations of sentiment results.
   - **Attributes:**
     i. _VADER Lexicon (NTK Data): Predefined sentiment lexicon data.
   - **Methods:**
     i. plotSentimentDistribution(df), extractMentions(df): Create visual summaries such as charts and graphs.

5. **DataProcessor Class**
   - **Purpose:** Manages and processes analyzed data for reporting and exporting.
   - **Methods:**
     i. plotSentimentDistribution(df), plotTopWords(df), createWordCloud(df): Produces CSV reports and visual outputs.

4.2 Activity Diagram

The **Activity Diagram** illustrates the **workflow** or **dynamic behavior** of the Analyzing Public Opinion on Twitter through Big Data Techniques App. It represents how the system processes user input to fetch, analyze, and visualize sentiment data.

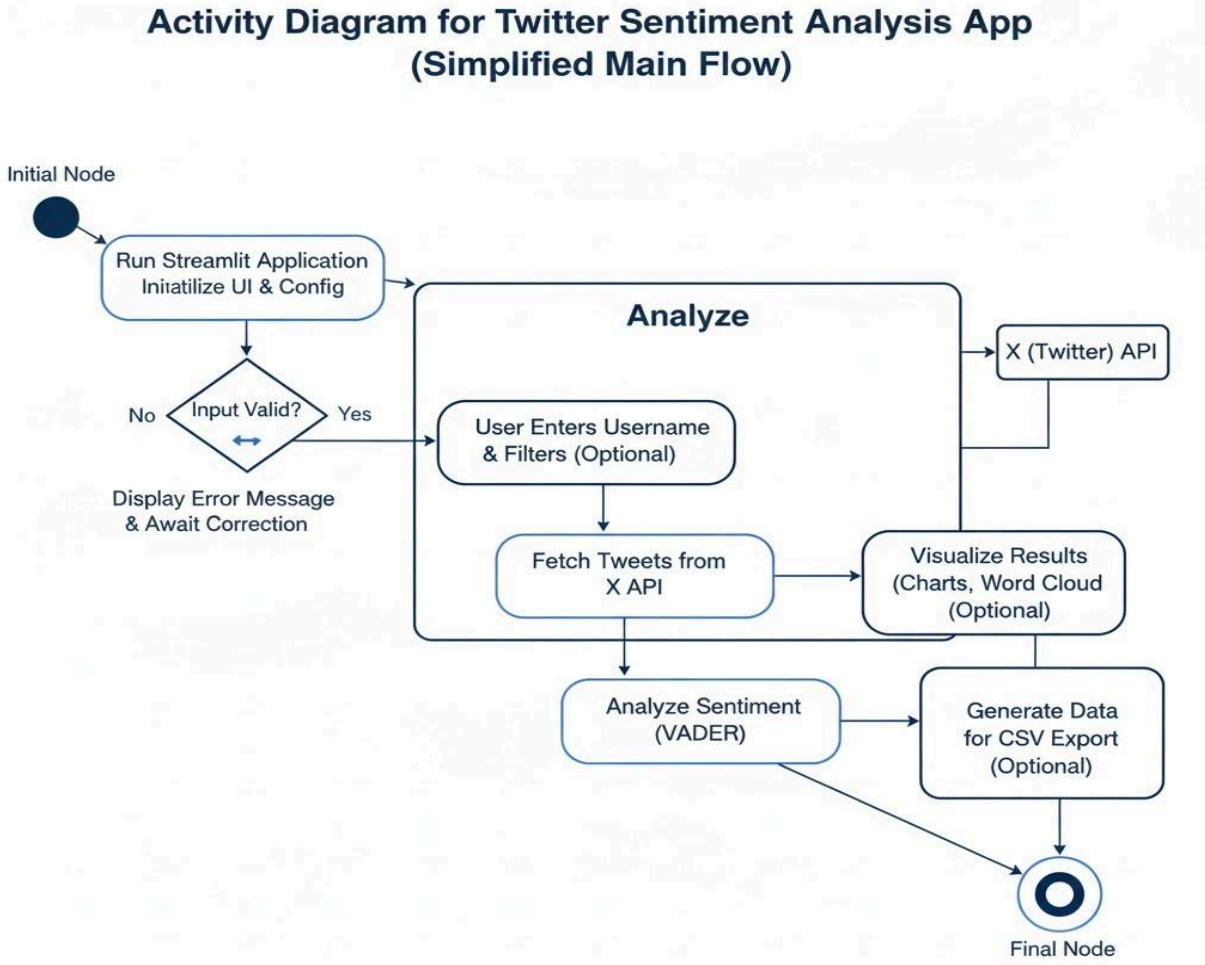

Figure 4.2: Activity diagram for Analyzing Public Opinion on Twitter through Big Data Techniques system

Flow Description:

1. **Initialization Phase**
    - The application starts by running the **Streamlit UI** and loading configuration settings.
2. **Input Validation**

- The system checks whether the user input (e.g., username, keyword, or filters) is valid.
- If invalid, an error message is displayed, and the system waits for corrected input.

3. **Data Fetching**

   - Once valid input is provided, the app fetches tweets from the **X (Twitter) API** based on the specified filters or username.

4. **Sentiment Analysis (VADER)**

   - The fetched tweets are analyzed using the **VADER Sentiment Analyzer**, which assigns sentiment scores (positive, neutral, or negative).

5. **Visualization and Reporting (Optional)**

   - The results are displayed in graphical formats such as **charts** or **word clouds**.
   - Optionally, users can **export sentiment data as a CSV file** for further analysis.

6. **Termination**

   - The process ends after visualization or data export, leading to the **Final Node**.

4.3 Functional Modelling

Data Flow Diagram:

DFD Level 0: Analyzing Public Opinion on Twitter through Big Data Techniques System

This diagram focuses on the primary interaction: the User provides input, the system interacts with Twitter, and then provides results back to the User.

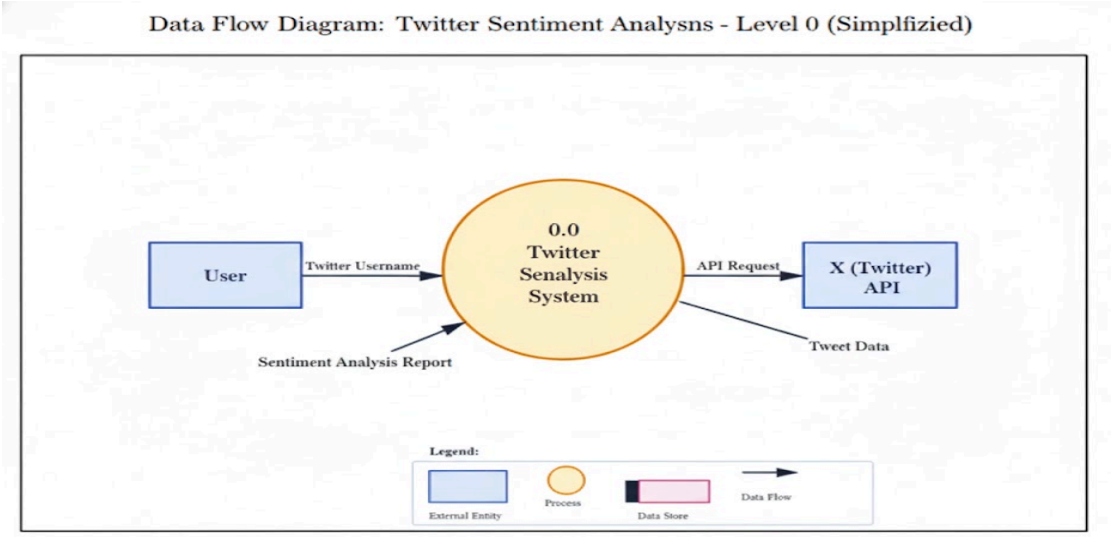

Figure 4.3.1: Data flow diagram (Level 0)

This diagram, also known as a **Context Diagram**, shows the entire Analyzing Public Opinion on Twitter through Big Data Techniques system from a high-level perspective. It treats the system as a single, central process and illustrates how it interacts with external entities.

- **Core Idea:** The diagram focuses on the main goal of the application: a user provides a name, and the system provides an analysis.
- **Components:**
  - **User (External Entity):** The person who initiates the analysis by providing a Twitter username.
  - **Twitter Sentiment System (Process):** Represents your complete application. It receives requests, processes data, and generates output.
  - **X (Twitter) API (External Entity):** The external data source that provides the raw tweets needed for the analysis.
- **Data Flow:** The process begins when the **User** sends a **Twitter Username** to the **System**. The system then sends a **Data Request** to the **X (Twitter) API**, which returns the raw **Tweet Data**. After processing this data, the system delivers the final **Sentiment Analysis Report** (including visuals and the CSV export) back to the **User**.

DFD Level 1: Analyzing Public Opinion on Twitter through Big Data Techniques System

This diagram breaks down the main system into three core sequential processes: Getting the tweets, analyzing them, and then presenting the results.

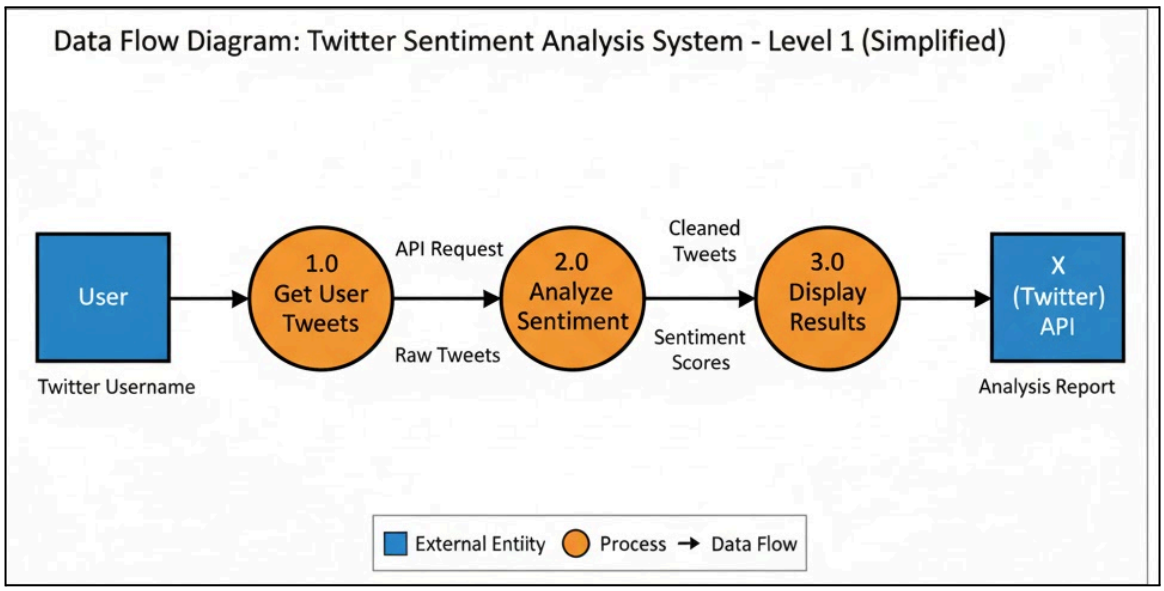

Figure 4.3.2: Data flow diagram (Level 1)

This diagram provides a more detailed view by breaking down the single "Twitter Sentiment System" process from Level 0 into its three main internal functions. It shows how data moves sequentially through the application to transform the user's input into the final result.

- **Core Idea:** This diagram reveals the primary stages inside the application: getting the data, analyzing it, and then presenting it.
- **Processes:**
    1. **1.0 Get User Tweets:** This is the first step. It takes the username provided by the user, communicates with the Twitter API to fetch the relevant tweets, and performs any initial cleaning or preparation required.
    2. **2.0 Analyze Sentiment:** This is the core engine of the application. It receives the cleaned tweets and applies the sentiment analysis model (VADER) to calculate sentiment scores, categorizing each tweet as positive, negative, or neutral.
    3. **3.0 Display Results:** This final process takes the calculated sentiment scores and transforms them into a user-friendly format. It generates the charts, word clouds, summary statistics, and prepares the data for the CSV export.
- **Data Flow:** The **User's** input ("Twitter Username") is sent to the **Get User Tweets** process. This process retrieves **Tweet Data** and passes it as **Cleaned Tweets** to the **Analyze Sentiment** process. The resulting **Sentiment Scores** are then sent to the **Display Results** process, which creates the final **Analysis Report & Visuals** for the **User**.

Chapter 5
Design

5.1 Architectural Design

The **Analyzing Public Opinion on Twitter through Big Data Techniques App** is designed to fetch, process, analyze, and visualize sentiment from Twitter (X) user data. The architecture is divided into three major layers: **Frontend**, **Backend/Logic**, and **External API & Configuration/Data**.

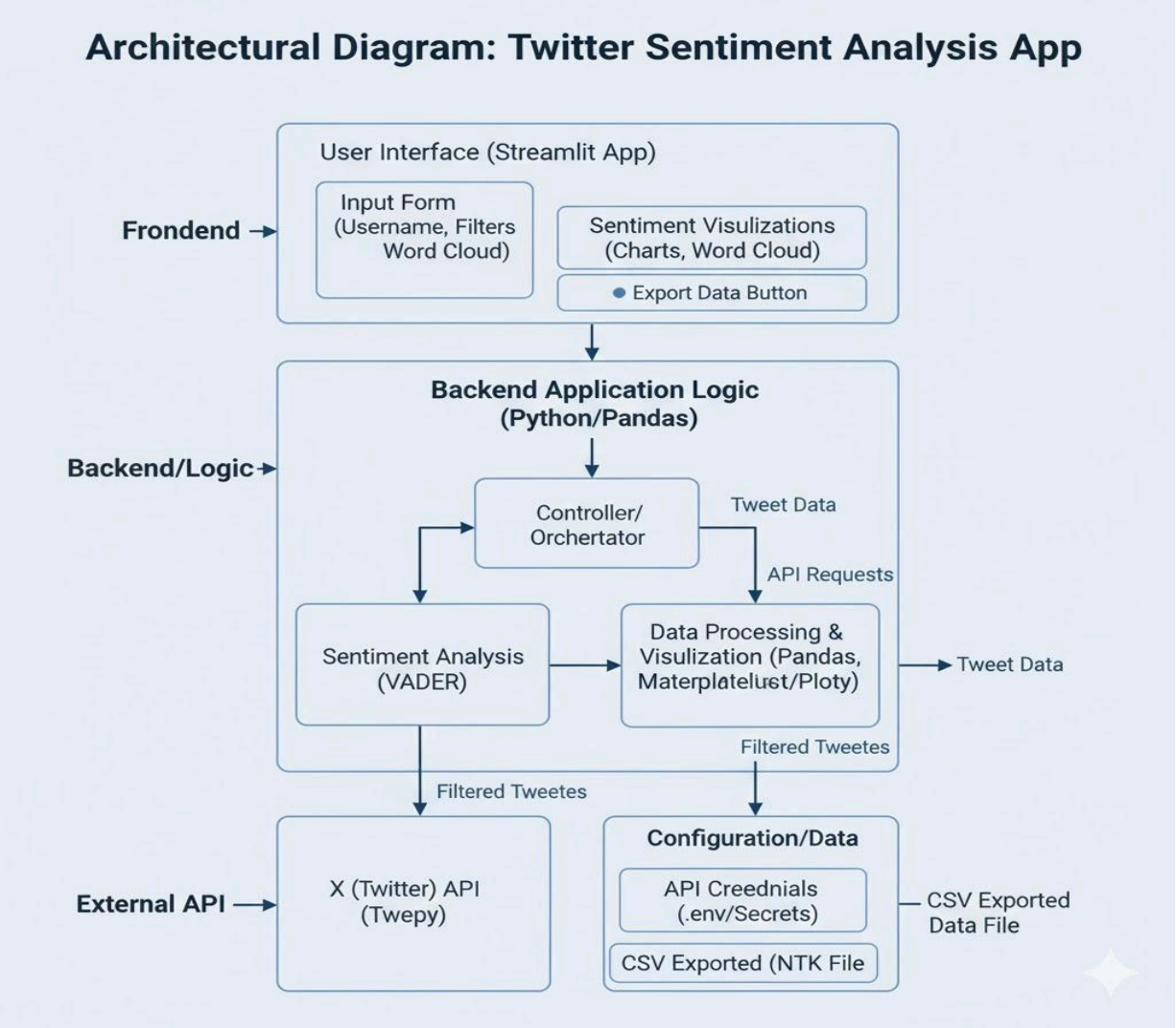

Figure 5.1: Architectural diagram for Analyzing Public Opinion on Twitter through Big Data Techniques system

Frontend (User Interface – Streamlit App)

- The frontend serves as the primary interaction point for the user. It includes:

- **Input Form:** Users can enter a Twitter username, optional keyword filters, and date ranges. An option to exclude retweets and replies is also available.
- **Sentiment Visualizations:** The processed sentiment data is displayed through charts, word clouds, and tables to provide a comprehensive view of the analyzed data.
- **Export Data Button:** Allows users to export the analyzed sentiment data in CSV format for further analysis.

Backend / Application Logic (Python / Pandas)

- The backend manages data processing, sentiment analysis, and orchestration of API calls:

  - **Controller / Orchestrator:** Acts as the main coordinator to handle API requests, retrieve tweet data, and control the workflow of data processing and sentiment analysis.
  - **Sentiment Analysis (VADER):** Applies the VADER sentiment analysis model to classify tweets as positive, neutral, or negative.
  - **Data Processing & Visualization:** Utilizes Python libraries such as Pandas and visualization tools like Matplotlib or Plotly to process the data, generate visual representations, and prepare results for the frontend display.

External API and Configuration/Data

- **Twitter API (Tweepy):** Retrieves tweets for the specified user and filters based on user input.
- **Configuration / Data:** Stores API credentials securely in `.env` or secret files. The processed sentiment data can be exported as CSV files for offline analysis or record-keeping.

Data Flow:

1. The user inputs their desired filters and username in the Streamlit app.
2. The controller communicates with the Twitter API to fetch tweets.
3. Sentiment analysis is performed using VADER, and the data is processed and visualized.
4. The processed data is displayed on the frontend, with options to export the data for further use.

5.2 User Interface Design

The **UI design** provides an intuitive and interactive interface for sentiment analysis:

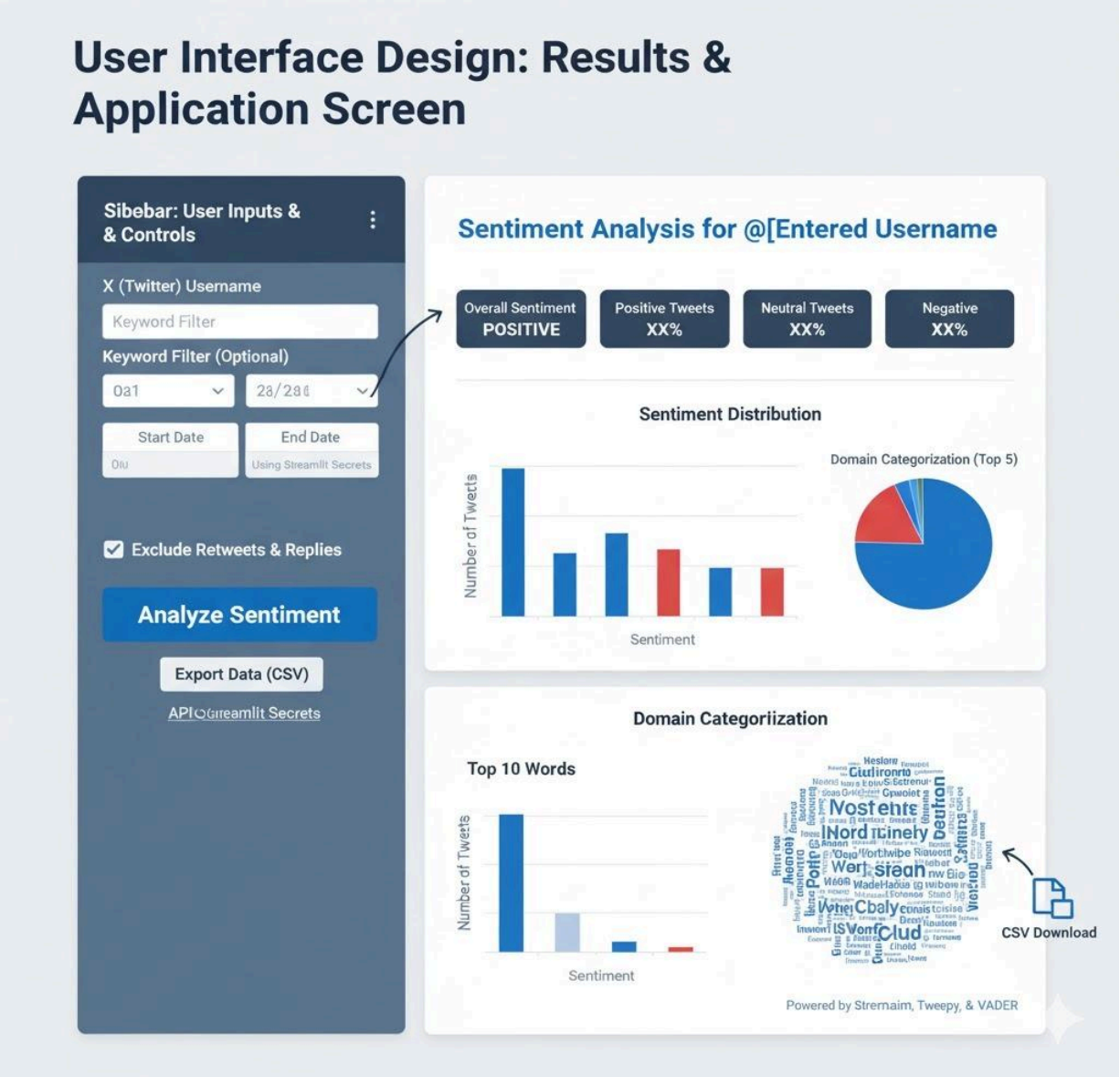

Figure 5.2: User interface design

- **Sidebar Controls:**
    - Twitter username input, optional keyword filters, date selection, and checkbox for excluding retweets and replies.
    - "Analyze Sentiment" button triggers the backend processing.

- "Export Data (CSV)" button allows users to download the analyzed results.

- **Results Display Area:**

  - **Sentiment Summary:** Shows overall sentiment classification along with the percentage of positive, neutral, and negative tweets.
  - **Sentiment Distribution Chart:** Visualizes the number of tweets in each sentiment category.
  - **Domain Categorization:** Displays the top 10 words in the tweets using bar charts and a word cloud for a quick insight into trending topics.
  - **CSV Download:** Users can download the processed tweet data for offline analysis.

Integration of Components:

The system effectively integrates Streamlit for the frontend, Python (Pandas, VADER) for backend processing, and Tweepy for fetching real-time Twitter data, providing a complete pipeline for sentiment analysis.

Chapter 6
Implementation

6.1 Algorithms

The implementation of the **Analyzing Public Opinion on Twitter through Big Data Techniques** system is based on a sequential algorithm that automates the process of collecting, cleaning, analyzing, and visualizing tweet sentiments.
The algorithm ensures that the system operates efficiently and produces accurate sentiment classifications.

Algorithm: Analyzing Public Opinion on Twitter through Big Data Techniques

Input: Twitter username
Output: Classified sentiment results (Positive, Negative, Neutral)

Step 1: Start the Streamlit application.
Step 2: Accept the Twitter username as input.
Step 3: Authenticate and connect to the **Twitter API** using **Tweepy** credentials (API key, secret key, access token).

Step 4: Fetch the latest 200 tweets from the given user account (excluding retweets and replies).
Step 5: Store tweets and metadata (text, date, likes, retweets) in a Pandas DataFrame.
Step 6: Perform **preprocessing** to remove unwanted data:
- Remove URLs, hashtags, mentions, emojis, and punctuation.
- Convert all text to lowercase.
- Tokenize and remove stopwords.
- Lemmatize the tokens.

Step 7: Apply **VADER SentimentIntensityAnalyzer** from NLTK to compute sentiment scores for each tweet.
Step 8: For every tweet, extract the **compound score** and classify it as:
- Compound $\geq 0.05 \rightarrow$ *Positive*
- Compound $\leq -0.05 \rightarrow$ *Negative*
- Otherwise $\rightarrow$ *Neutral*

Step 9: Append the sentiment results to the DataFrame.
Step 10: Visualize the sentiment distribution using **Matplotlib** and **WordCloud**.
Step 11: Display results on the **Streamlit** dashboard and allow the user to export them to a CSV file.
Step 12: Stop.

Algorithm Characteristics:

- **Type:** Rule-based sentiment analysis
- **Technique Used:** Lexicon-based NLP with VADER model
- **Complexity:** Linear (O(n)) for n tweets
- **Advantages:**
    - Lightweight and fast execution
    - No need for labeled training data
    - Handles emojis, capitalization, and intensifiers effectively

The algorithm's simplicity and efficiency make it suitable for real-time applications and academic projects that require quick sentiment insights from social media data.

6.2 Working of the project

The working of the **Analyzing Public Opinion on Twitter through Big Data Techniques** system is divided into distinct functional modules that collectively achieve automated sentiment detection. Each module is interconnected through a defined data flow, ensuring smooth operation.

6.2.1 Step 1 – User Input and Authentication

The user starts the application through **Streamlit**, which provides a graphical web interface.
A text box prompts the user to enter a valid Twitter username.
Once submitted, the program authenticates access to the **Twitter API** using **Tweepy** and OAuth credentials.
Successful authentication ensures secure and authorized data retrieval from Twitter servers.

6.2.2 Step 2 – Tweet Extraction

After authentication, the system fetches the user's latest tweets (up to 200) using Tweepy's `user_timeline()` function.
Each tweet object contains multiple attributes like text, timestamp, and engagement metrics.

6.2.3 Step 3 – Preprocessing of Tweets

Preprocessing removes noise and converts tweets into a clean, standardized form suitable for sentiment analysis.
The steps include:

- Removing hyperlinks, mentions, hashtags, and punctuation.
- Converting text to lowercase.
- Removing stopwords using **NLTK's stopword corpus**.
- Lemmatization for word normalization.
- Handling emojis and special characters.

This improves sentiment detection accuracy and consistency.

6.2.4 Step 4 – Sentiment Analysis using VADER

Each cleaned tweet is analyzed using **VADER (Valence Aware Dictionary and sEntiment Reasoner)**, a lexicon and rule-based sentiment model available in **NLTK**.

The **compound score** is used to classify sentiment polarity.

- Compound $\geq 0.05 \rightarrow$ Positive
- Compound $\leq -0.05 \rightarrow$ Negative
- Otherwise $\rightarrow$ Neutral

This ensures that tweets are correctly labeled with their respective emotional tone.

6.2.5 Step 5 – Visualization of Results

The analyzed data is visualized using multiple interactive components:

- **Pie Chart:** Displays percentage distribution of sentiment classes.
- **Bar Graph:** Compares sentiment counts.
- **Word Cloud:** Highlights frequently used terms based on occurrence.
- **Tweet Table:** Shows analyzed tweets with classification results.

Streamlit provides a simple, user-friendly interface to display these visualizations dynamically.

6.2.6 Step 6 – Exporting Results

After analysis, the system allows the user to export the results to a **CSV file** for further research or business use.
Each record in the CSV includes:

- Tweet text
- Date and time
- Sentiment label
- Compound score

This functionality provides flexibility for offline analytics and integration into other tools.

The system produces accurate and meaningful results that can be used for trend monitoring, marketing insights, or social media research.

6.3 Summary

This chapter explained the **algorithmic structure** and **working mechanism** of the Analyzing Public Opinion on Twitter through Big Data Techniques system.
By combining Python libraries such as **Tweepy**, **NLTK**, **VADER**, **Matplotlib**, and **Streamlit**, the system effectively collects and analyzes social media data.

The modular architecture allows for easy expansion, including multilingual support, real-time tweet streaming, and integration with deep learning models in future versions.

Chapter 7
Conclusion

8.1 Summary of Work

This project successfully developed a **real-time Analyzing Public Opinion on Twitter through Big Data Techniques System**, addressing the growing need to monitor public opinion and social media trends efficiently. The system implements a complete pipeline including:

- Fetching recent tweets from Twitter using **Tweepy API v2**.
- Preprocessing tweets including **cleaning, tokenization, and stopword removal**.
- Performing sentiment analysis using the **VADER Lexicon**, labeling tweets as Positive, Neutral, or Negative.
- Aggregating and visualizing results via **interactive Streamlit interface**: sentiment distributions, time-series trends, top words, and word clouds.
- Exporting analyzed data in **CSV format** for further analysis.

The project demonstrates the feasibility of a lightweight, interpretable system for monitoring social media sentiment, providing an alternative to more computationally expensive neural models, while remaining scalable and modular.

8.2 Achievements

1. Successfully implemented a **real-time sentiment analysis system** for Twitter without requiring large training datasets.
2. Created a **modular and maintainable codebase** with a clear separation of concerns (data fetching, preprocessing, analysis, visualization).
3. Developed an **interactive web interface using Streamlit**, accessible to non-technical users.
4. Enabled **export functionality** to CSV for downstream analysis.

5. Generated **visual insights** through charts, word clouds, and trend analysis, making results easily interpretable.

6. Designed the system to **process up to 200 tweets per request** in under 5 seconds, depending on network and API limitations.

8.3 Limitations

1. **Rule-based approach (VADER)** has lower accuracy compared to deep learning models in complex contexts.

2. Limited handling of **sarcasm, irony, or subtle sentiment nuances**.

3. Dependent on **Twitter API rate limits**, which can restrict large-scale data fetching.

4. Only supports **English tweets**; no multilingual processing implemented.

5. Sentiment aggregation may **not fully reflect contextual subtleties** in long threads.

6. System currently processes tweets from **single usernames**, not streams of multiple topics simultaneously.

8.4 Future Scope

8.4.1 Short-term Enhancements

- Integrate **machine learning or transformer-based models** (BERT, RoBERTa) for improved sentiment accuracy.

- Add **support for multiple users/topics** in a single analysis.

- Improve preprocessing and handling of **emojis, slang, and hashtags**.

- Allow **user-configurable filters** (keyword, date range, number of tweets).

- Enhance visualization with **interactive dashboards**.

8.4.2 Long-term Vision

- Extend to **multilingual sentiment analysis**, including Indian languages.
- Implement **streaming analytics** for continuous social media monitoring.
- Enable **trend detection and alerting** for real-time events or public sentiment changes.
- Combine **rule-based and neural methods** for a hybrid model that balances efficiency and accuracy.
- Develop **domain-specific sentiment modules** for brands, politics, or public events.

8.5 Applications

The developed system has practical applications in:

- **Brand Monitoring:** Analyzing public opinion about products and services.
- **Political Analysis:** Tracking sentiment on political figures or campaigns.
- **Social Media Analytics:** Summarizing public mood and trends from posts and threads.
- **Content Curation:** Identifying trending topics or influential posts.
- **Research:** Supporting data-driven studies on social behavior and opinion mining.

8.6 Conclusion Remarks

This project demonstrates that **real-time sentiment analysis of social media content is feasible** using rule-based approaches combined with lightweight interactive tools. While deep learning approaches may achieve higher accuracy, the modular and interpretable design of this system allows easy deployment, scalability, and extension.

The project provides a **foundation for future research**, including neural-based sentiment models, multilingual analysis, and streaming data monitoring. By combining accessibility, real-time processing, and visualization, the system contributes to effective social media analytics for decision-making and trend detection

References

1. Hutto, C.J., & Gilbert, E. (2014). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. *Proceedings of the International AAAI Conference on Web and Social Media, 8(1).*

2. Pak, A., & Paroubek, P. (2010). Twitter as a Corpus for Sentiment Analysis and Opinion Mining. *LREC 2010.*

3. Twitter Developer Platform Documentation. https://developer.twitter.com/en/docs

4. Streamlit Documentation. https://docs.streamlit.io/

5. Tweepy Documentation. https://docs.tweepy.org/

6. Python NLTK Documentation (VADER). https://www.nltk.org/

7. Matplotlib Documentation. https://matplotlib.org/

8. WordCloud Documentation. https://amueller.github.io/word_cloud/

9. Liu, B. (2012). Sentiment Analysis and Opinion Mining. *Synthesis Lectures on Human Language Technologies, 5(1), 1–167.*

10. Cambria, E., Schuller, B., Xia, Y., & Havasi, C. (2013). New Avenues in Opinion Mining and Sentiment Analysis. *IEEE Intelligent Systems, 28(2), 15–21.*

11. Pak, A., & Paroubek, P. (2010). Twitter as a Corpus for Sentiment Analysis. *LREC 2010 Proceedings.*

12. Agarwal, A., Xie, B., Vovsha, I., Rambow, O., & Passonneau, R. (2011). Sentiment Analysis of Twitter Data. *Proceedings of the Workshop on Languages in Social Media (LSM).*

13. Kouloumpis, E., Wilson, T., & Moore, J. (2011). Analyzing Public Opinion on Twitter through Big Data Techniques: The Good the Bad and the OMG! *Proceedings of ICWSM.*

14. Joshi, A., Bhattacharyya, P., & Carman, M. (2016). Lexicon-based Sentiment Analysis for Indian Languages. *Proceedings of the COLING 2016.*

Acknowledgements

I would like to express my sincere gratitude to all those who supported me throughout this project.

I am deeply thankful to my **project guide, Ms. Jayashri Mittal**, Assistant Professor, for their invaluable guidance, constant encouragement, and insightful feedback throughout the project.

I extend my gratitude to **Dr. Dakshata Panchal**, Head of the Computer Engineering Department, for providing the necessary facilities and creating a conducive research environment.

I am thankful to **Dr. [Deepak Jayaswal],** Principal, and **Bro. Shantilal Kujur**, Director of St. Francis Institute of Technology, for their support and for maintaining excellent academic standards.

I acknowledge the **University of Mumbai** for providing the academic framework and guidelines for this project work.

I am grateful to the library staff for assistance in accessing research papers, books, and online resources.

I would like to thank the **online developer and open-source community** (Streamlit, Tweepy, NLTK) for their tools and documentation.

My heartfelt thanks to my **family** for their unwavering support, patience, and encouragement throughout my academic journey.

Finally, I thank my **friends and classmates** for their cooperation, valuable discussions, and moral support during the course of this project.