Twitter Sentiment Analysis Tool

A Dockerized Natural Language Processing Project

By: Meshwa Patel

Table of Contents:

1. **Abstract**

Today, social media platforms, especially Twitter, are more than just places for conversation. They serve as a mirror of public opinion in real-time, whether it's about world events, trending topics, or major brands. This project introduces a **Twitter Sentiment Analysis Tool** that taps into that public conversation and analyzes it using **Natural Language Processing** (NLP) techniques.

The tool is built with Python and brings together various components like **data collection**, **text cleaning**, **sentiment classification**, and **interactive visualizations**. It uses Twitter's API to fetch tweets based on any search query and then processes them using both traditional (TextBlob) and modern (RoBERTa) NLP models to detect the emotional tone behind the text. Each tweet is labeled as positive, neutral, or negative, along with a confidence score.

What makes this project accessible is its user-friendly Streamlit web app, where anyone can:

- Search for tweets on any topic
- See sentiment trends and engagement metrics
- View and explore tweet-level analysis
- Export the results as CSV or JSON files

Everything runs inside Docker, so there are no messy installations or environment setup issues. Just one command and the dashboard is ready to use. This project is a practical example of applying NLP to understand social media better, and it can be useful for researchers, businesses, and everyday users alike.

2. Introduction

In today's hyper-connected world, social media platforms have evolved into real-time public forums where individuals express their thoughts, feelings, and opinions on a wide range of topics. Among these platforms, **Twitter** stands out as a global microblogging service that offers valuable insights into public sentiment due to its brevity, speed, and widespread usage.

With the increasing volume of tweets being generated every second, manually analyzing them for meaningful insights is infeasible. This challenge calls for an automated system that can efficiently capture, process, and interpret the sentiments expressed in tweets. **Sentiment analysis**, also known as opinion mining, is a subfield of Natural Language Processing (NLP) that determines the emotional tone behind text data. It is widely used in domains such as market research, politics, customer service, and social analytics.

This project, titled **"Twitter Sentiment Analysis Using Streamlit and Docker"**, aims to address this need by building a fully functional, modular, and interactive system that performs real-time sentiment analysis on Twitter data. The system is implemented in **Python** and integrates the following capabilities:

- **Data Collection** using the Twitter API v2 to fetch tweets based on a user-defined query.
- **Data Preprocessing** using advanced regular expression techniques to clean raw tweet text.
- **Sentiment Classification** using both traditional models (TextBlob) and transformer-based models (RoBERTa) to label tweets as positive, neutral, or negative.
- **Visualization & Insights** using intuitive charts, graphs, and engagement metrics for deeper understanding.
- **User Interface** built with Streamlit that enables anyone (with no coding experience) to run analyses via a friendly web dashboard.
- **Deployment** using Docker to ensure reproducibility, scalability, and ease of deployment across any machine.

The motivation behind this project stems from the increasing importance of social media analysis in business intelligence, consumer feedback monitoring, and social research. By developing a comprehensive and user-friendly sentiment analysis pipeline, this project bridges the gap between raw social media content and actionable insights.

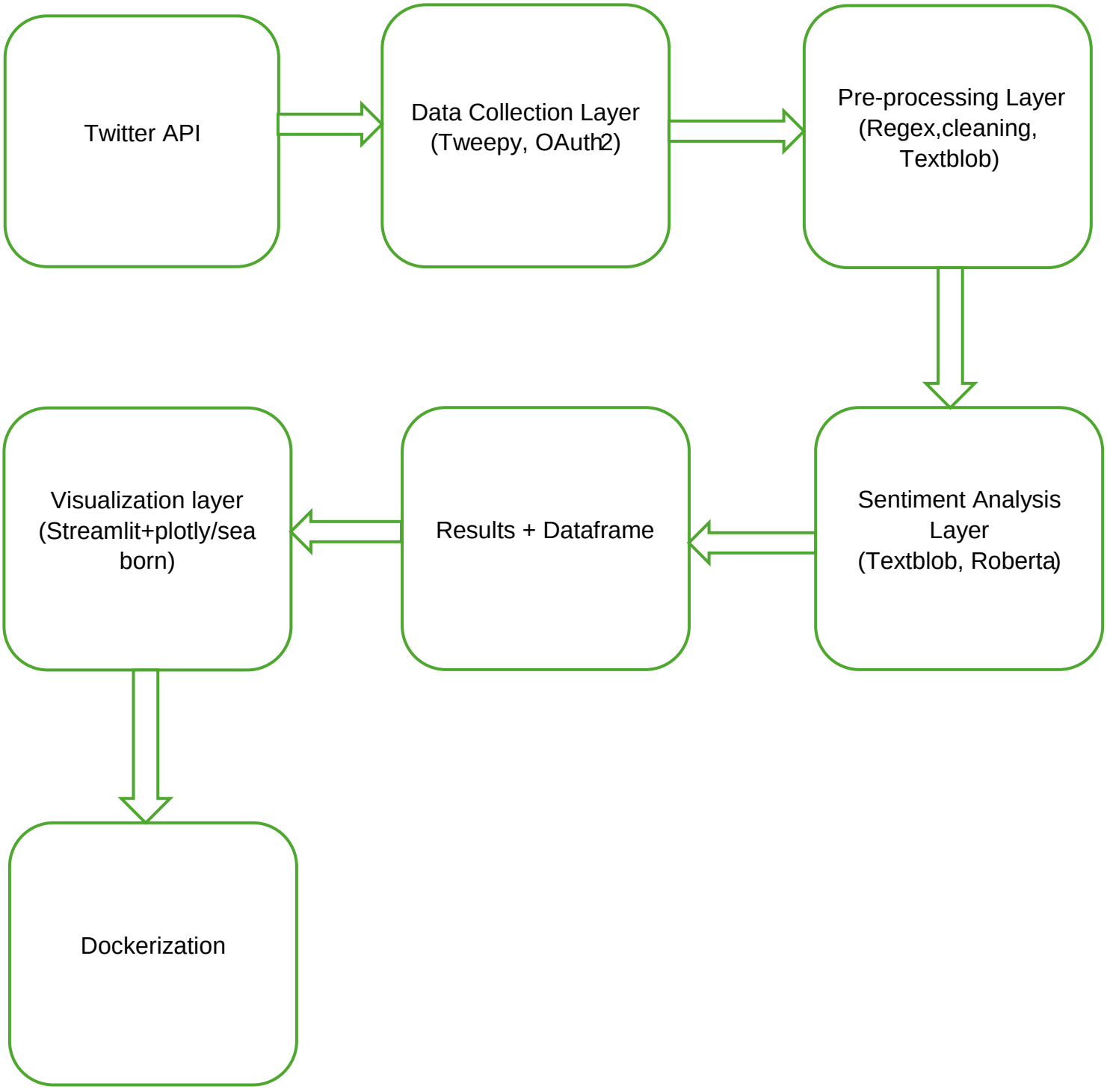
Ultimately, the system provides a robust foundation for tracking trends, measuring public mood, and making data-driven decisions based on real-time social feedback from the Twitter verse.

3. System Architecture

The architecture of the Twitter Sentiment Analysis system is designed to be modular, scalable, and user-friendly, supporting seamless integration of various components responsible for data collection, pre-processing, sentiment analysis, visualization, and deployment. The system follows a **layered architecture** with clearly defined responsibilities at each layer, ensuring maintainability and extensibility.

Below is a breakdown of the architecture:

3.1 Overview Diagram



3.2 Component Breakdown

1. Twitter Data Collection Layer

- **Technology Used:** Tweepy, Twitter API v2
- **Function:** This layer connects to Twitter's API using secure OAuth 2.0 credentials and fetches tweets based on user-defined queries.
- **Features:**
    - Rate limit handling
    - Query-based filtering (e.g., language ='en')
    - Metadata collection (likes, retweets, author ID, timestamp)

2. Data Pre-processing Layer

- **Technology Used:** Regex, Python string methods, TextBlob
- **Function:** Cleans the raw tweet text by:
    - Removing URLs, mentions, emojis, extra whitespace, and punctuation
    - Filtering short/duplicate tweets
    - Prepares data for robust sentiment analysis

3. Sentiment Analysis Layer

- **Technology Used:** TextBlob, ROBERTa (Hugging Face Transformers)
- **Function:**
    - Provides multiple sentiment analysis methods:
        - Lightweight (TextBlob) for fast analysis
        - Deep learning-based (ROBERTa) for higher accuracy
    - Outputs both sentiment label and confidence score

4. Visualization and Analysis Layer

- **Technology Used:** Streamlit, Plotly, Seaborn, WordCloud
- **Function:** Displays processed results in an interactive dashboard including:
    - Sentiment distribution (pie, bar, histogram)
    - Engagement metrics (likes, retweets, replies)
    - Sample tweet display
    - Confidence scores
    - Word clouds and downloadable results

5. Containerization and Deployment Layer

- **Technology Used:** Docker, Dockerfile, docker-compose
- **Function:**
    - Encapsulates all system components into a Docker container
    - Ensures consistency across environments
    - Allows one-command deployment of the entire pipeline and frontend interface

3.3 Workflow Summary

1. **User opens the Streamlit app** in a browser.
2. They **initialize the pipeline**, which sets up the backend components.
3. On entering a query (e.g., "AI in healthcare"), tweets are **fetched via Tweepy**.
4. Tweets are **cleaned** and **analyzed** using the selected NLP model.
5. Results are **visualized** with interactive charts and downloadable reports.
6. The **entire flow is containerized** using Docker, requiring no manual setup.

3.4 Architectural Advantages

- **Modularity** – Each module can be improved or replaced independently (e.g., upgrade to BERT instead of RoBERTa).
- **Scalability** – Easily deployable to cloud environments via Docker.
- **User-Friendliness** – Streamlit interface hides all complexity from the end-user.
- **Customizable** – Analysis method, query term, and data volume can be dynamically chosen.

4. Technologies Used

- **Python** – Core programming language used to build the entire project.
- **Tweepy** – Used to connect and fetch tweets from the Twitter API.
- **Pandas** – For structured data handling, cleaning, and storage.
- **TextBlob** – Lightweight library for basic sentiment analysis.
- **Hugging Face Transformers (RoBERTa)** – Used for advanced deep learning-based sentiment classification.
- **Streamlit** – Provides an interactive web interface for user input and visualization.
- **Plotly** – Generates interactive visualizations like pie, bar, and histograms in the dashboard.
- **Seaborn & Matplotlib** – Used for plotting graphs and word clouds in the offline visualizer module.
- **WordCloud** – Creates visual clouds of frequently used words by sentiment.
- **Scikit-learn** – Provides ML utilities like TF-IDF and logistic regression (optional).
- **Docker** – Containerizes the entire project for easy deployment and reproducibility.
- **dotenv** – Manages and secures API keys and tokens via .env file.

5. Implementation

The implementation of this project is structured in a modular, object-oriented fashion to ensure readability, scalability, and maintainability. Each component has a well-defined role within the pipeline. Below is a breakdown of how each module was implemented:

5.1 Project Structure

project/

```
│
├── DataCollection.py            ← Collects tweets from Twitter API
├── DataPreprocessing.py         ← Cleans and preprocesses tweet text
├── SentimentAnalysis.py         ← Performs sentiment analysis using NLP models
├── Visualization_and_analysis.py ← Generates graphs and insights
├── main.py                      ← Core pipeline logic
├── streamlit.py                 ← Frontend interface for user interaction
├── .env                         ← Secure storage for API keys and tokens
├── Dockerfile                   ← Docker setup for containerized deployment
├── requirements.txt             ← List of Python dependencies
└── README.md                    ← Project documentation
```

5.2 Key Module Details

5.2.1 DataCollection.py

- Uses **Tweepy** to connect with Twitter API v2.
- Collects recent English-language tweets using a given query.
- Fetches tweet metadata such as likes, retweets, replies, and timestamps.
- Returns data as a structured Pandas DataFrame.

5.2.2 DataPreprocessing.py

- Applies multiple regex-based cleaning operations:
  - Lowercasing
  - URL removal
  - Hashtag and mention handling
  - Emoji and punctuation stripping
- Removes duplicate and short (less than 3-word) tweets.

5.2.3 SentimentAnalysis.py

- Offers two modes of sentiment analysis:
  - **TextBlob** for simple polarity/subjectivity analysis.
  - **RoBERTa** (via Hugging Face Transformers) for deep-learning-based classification.
- Outputs both sentiment labels and confidence scores.

5.2.4 Visualization_and_analysis.py

- Uses **Matplotlib**, **Seaborn**, and **WordCloud** for:
  - Pie and bar plots of sentiment distribution
  - Confidence histograms
  - Word clouds for each sentiment
  - Box plots of engagement metrics (likes, retweets, replies)
- Displays most frequent words per sentiment class.

5.2.5 main.py

- Coordinates all modules and defines the TwitterSentimentPipeline class.
- Calls data collection, preprocessing, sentiment analysis, and visualizer.
- Saves processed data and summary to disk (CSV and JSON).

5.2.6 streamlit.py

- Provides an interactive UI using **Streamlit**.
- Accepts user input for queries and tweet limits.

- Displays metrics, charts, insights, and lets users download results.
- Also includes single-tweet sentiment classification.

5.3 Docker Integration

The project is containerized using Docker to ensure platform-independent deployment.

Dockerfile

- Based on python:3.10-

  slim.

  - Installs all dependencies from requirements.txt.
  - Exposes port 8501 for Streamlit.
  - Automatically launches the Streamlit app when the container runs.

  Run instructions:

  docker build -t twitter-sentiment .

  docker run -p 8501:8501 twitter-sentiment

5.4 API and Security

- API keys and tokens are stored securely in a .env file.
- Accessed in Python using the python-dotenv library.
- Keeps sensitive credentials out of the source code and version control.

6. Dockerization

To ensure seamless deployment across environments, the entire Twitter Sentiment Analysis project is containerized using Docker. Docker enables the application to be packaged with all its dependencies into a portable container that can be run on any system with Docker installed.

6.1 Why Docker?

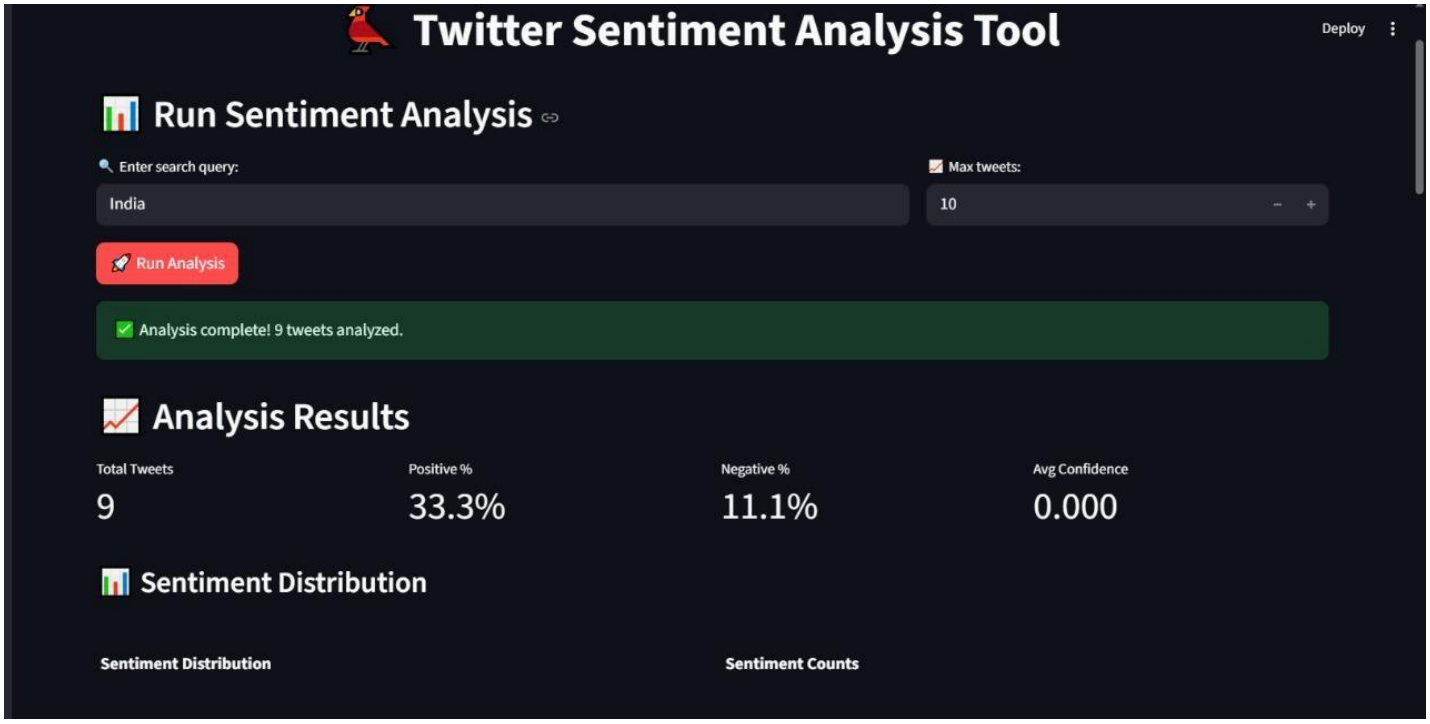
- **Portability:** The application runs the same across Windows, macOS, and Linux.
- **Reproducibility:** All dependencies are installed consistently, preventing environment-related issues.
- **Simplicity:** With a single command, the entire project can be built and launched.

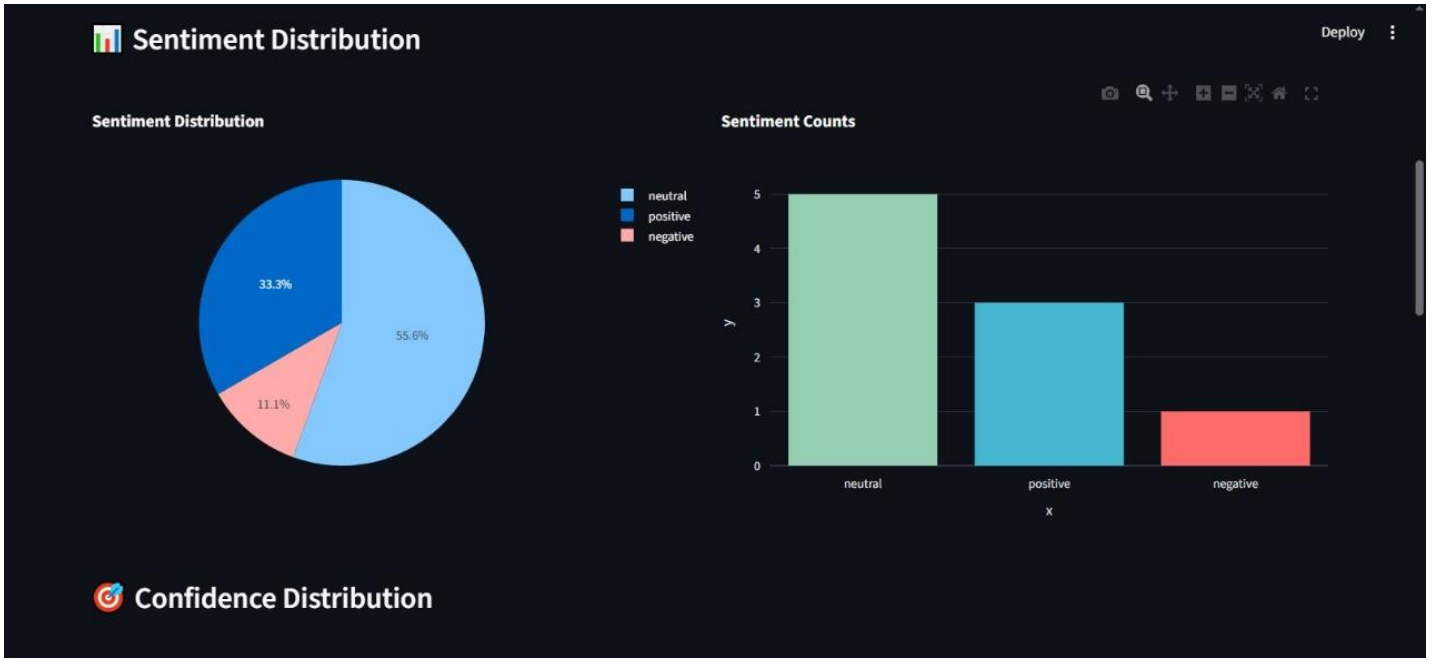
6.2 Dockerfile Overview

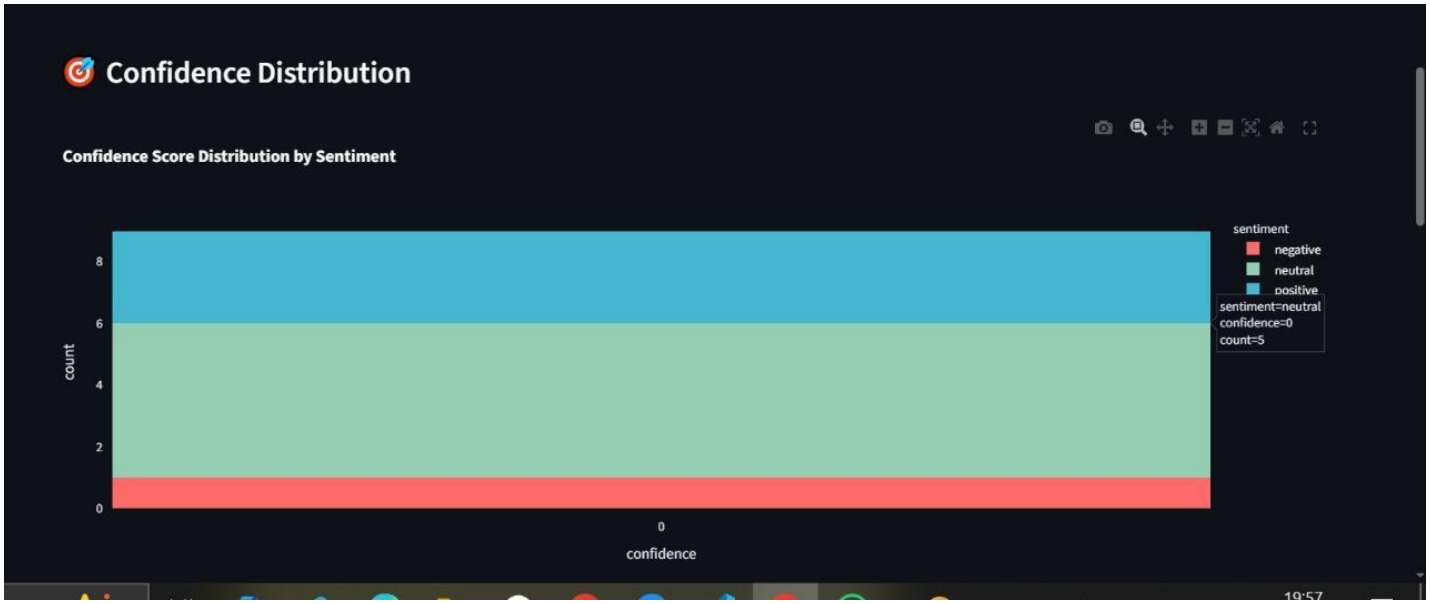
The Dockerfile used in this project performs the following key steps:

1. **Base Image:** Uses the official slim version of Python 3.10 to minimize size.
2. **Working Directory:** Sets /app as the working directory in the container.
3. **Dependency Installation:** Installs all Python libraries listed in requirements.txt.
4. **NLTK Corpus Download:** Downloads punkt tokenizer required by TextBlob.
5. **File Copy:** Copies the full project code into the container.
6. **Port Exposure:** Opens port 8501 for the Streamlit web app.
7. **Entry Point:** Launches the Streamlit app on container start

7. Output Screenshots:







8. Conclusion

This project successfully demonstrates the power of Natural Language Processing (NLP) and data visualization by building an end-to-end **Twitter Sentiment Analysis Tool**. Leveraging real-time data from Twitter, it automates the process of collecting, cleaning, analyzing, and visualizing public sentiment for any given keyword or topic.

The modular architecture ensures high maintainability, and the use of pretrained models like TextBlob and RoBERTa offers flexibility and performance in sentiment classification. The integration of an interactive Streamlit dashboard provides a user-friendly interface that bridges the gap between technical analysis and human interpretation.

The entire system was packaged and deployed using Docker, making it highly portable and production-ready across different environments and machines. With this infrastructure in place, the tool is easily extendable for future improvements like:

- Multi-language support
- Real-time tweet streaming
- Fine-tuned custom sentiment models
- Integration with cloud services or databases

This project not only strengthened practical skills in Python, NLP, visualization, and containerization, but also showcased how AI and social media data can be combined to generate valuable, real-world insights.

Ultimately, this tool can serve as a foundation for future research, market analysis, brand monitoring, or any application where public sentiment plays a critical role.