Project Report: Real-Time Twitter Sentiment Analysis System

CS210

Team: Vraj Patel (vmp130), Vraj Soni(vhs35), Anurag Vattipalli(vv270)
GitHub Repo: https://github.com/pvraj1904/twitter-sentiment-cs210

DemoVideo:
https://drive.google.com/file/d/1wZvlbirJzzbVDd3ewQMs99h9gdz2PhQe/view?usp=drive_link

---

1. Introduction

Social media is one of the fastest, most public forms of opinion and news sharing today. Platforms like Twitter (now called X) are crucial not only for personal expression, but also for real-time event monitoring, news tracking, and brand management. Accurately tracking sentiment as it develops on Twitter can help organizations and researchers understand public mood, react to crises, or identify emerging trends.

This project aims to build a real-time sentiment analysis system for Twitter, combining **historical datasets** (for context and model training) with **live tweet streams** (for up-to-the-minute updates). Our final deliverable is an interactive dashboard where users can view sentiment trends, popular topics, and detailed tweet samples—all updated in near real-time.

---

2. Problem Statement and Motivation

Traditional sentiment analysis tools often run on static, previously downloaded data. This can miss sudden shifts in public mood, making them less effective for monitoring events as they happen. Our goal was to close this gap, offering a platform that both **learns from past data and reacts instantly to new information**. This dual approach is especially valuable in fields such as journalism, marketing, politics, and crisis management, where timing is everything.

---

3. Literature Review / Related Work

Existing open-source sentiment analysis tools, such as those found on Kaggle or in basic NLP packages, generally use standard classifiers (like VADER or TextBlob) and work on batches of static data. More advanced, cloud-based commercial systems can provide real-time results, but are rarely open and often lack transparency. Our system builds on these foundations by integrating batch and streaming data pipelines, using a modular Python codebase and a lightweight, interactive dashboard.

---

4. System Architecture and Implementation

A. High-Level Architecture

The project consists of several clear steps:

1. **Historical Data Ingestion:**
   We use the Sentiment140 dataset (1.6 million labeled tweets) from Kaggle as our foundation. This is loaded into a MongoDB collection for easy processing and querying.
2. **Real-Time Data Collection:**
   Tweets are streamed from the Twitter/X API (using Tweepy), filtered by relevant keywords or hashtags. Data is inserted into MongoDB for live analysis.
3. **Data Cleaning:**
   Tweets are processed to remove URLs, mentions, hashtags, emojis, and non-English content. We use Python's regex and the langdetect package for this purpose.
4. **Sentiment Analysis:**
   The VADER sentiment model (from vaderSentiment) is used for fast, social-media-friendly sentiment scoring (positive/negative/neutral).
5. **Aggregation:**
   Sentiment scores are grouped by day for historical trends, and by other metadata (such as user, hashtag, or keyword) for deep dives.
6. **Visualization:**
   All processed data is visualized in a **Streamlit dashboard**, with charts, word clouds, sample tweets, and live filter controls.

B. Technologies Used

- **Python 3.12**
- **MongoDB** (NoSQL database, flexible for both structured/unstructured data)
- **Tweepy** (Twitter API client)
- **VADER Sentiment** (sentiment scoring)
- **Streamlit** (dashboard)
- **Pandas, Matplotlib, wordcloud** (data processing and charts)

C. Code Structure

```
/sentiment-analysis-project
  |-- src/
      |-- collect.py        # Load historical data
      |-- collect_v2.py     # Fetch live tweets (Twitter/X API)
```

```
    |-- clean.py          # Preprocess and clean tweets
    |-- analyze.py        # Sentiment analysis
    |-- aggregate.py      # Aggregation for dashboard
    |-- dashboard.py      # Streamlit dashboard
|-- data/
    |-- historical/
        |-- sentiment140.csv  # (excluded from repo, must download from Kaggle)
|-- requirements.txt
|-- .gitignore
|-- README.md
```

D. Implementation Details

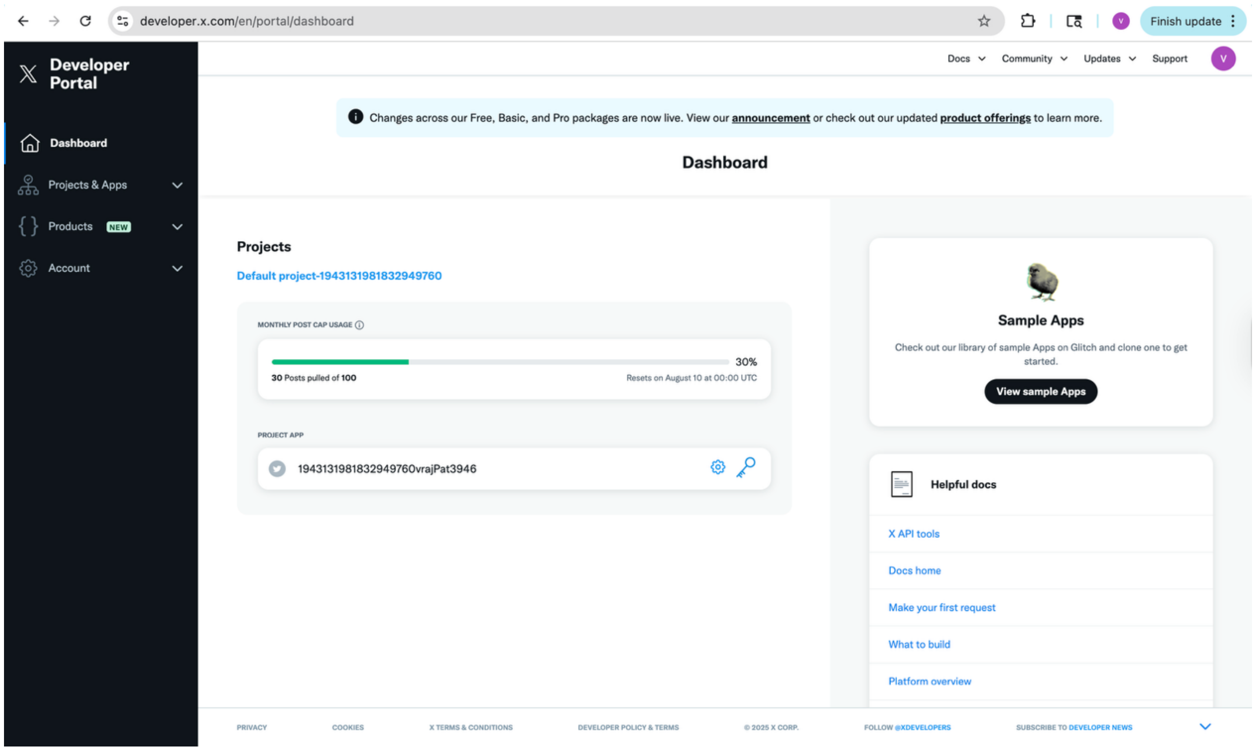
- **Historical data loading:**
  collect.py reads the CSV, processes each tweet, and inserts it into MongoDB, preserving metadata (date, user, text, etc.).
- **Live tweet collection:**
  collect_v2.py authenticates with Twitter/X API, streams tweets by topic/hashtag, and writes them to the raw data collection.
- **Cleaning and feature extraction:**
  clean.py removes noise (URLs, @mentions, emojis), standardizes text, and uses langdetect to filter for English-language tweets. Results are stored in a separate "cleaned" collection.
- **Sentiment analysis:**
  analyze.py assigns a sentiment label to each tweet using VADER. These are written to an "analyzed" collection.
- **Aggregation:**
  aggregate.py uses pandas to group sentiments by date and other fields, for efficient dashboard queries.
- **Dashboard:**
  dashboard.py runs a Streamlit app to visualize trends, word clouds, sample tweets, and more, with interactive filters.

---

5. Results & Outputs

Figure 1: X API App Setup

Description:
Screenshots showing the setup of the Twitter/X Developer Portal, project app registration, and API usage limits. This step is required to access live tweets from Twitter/X API.



developer.x.com/en/portal/dashboard

Finish update

Developer Portal

Dashboard

Projects & Apps

Products NEW

Account

Docs Community Updates Support

Changes across our Free, Basic, and Pro packages are now live. View our **announcement** or check out our updated **product offerings** to learn more.

Dashboard

Projects

Default project-1943131981832949760

MONTHLY POST CAP USAGE

30%

30 Posts pulled of 100 — Resets on August 10 at 00:00 UTC

PROJECT APP

1943131981832949760vrajPat3946

Sample Apps

Check out our library of sample Apps on Glitch and clone one to get started.

View sample Apps

Helpful docs

X API tools

Docs home

Make your first request

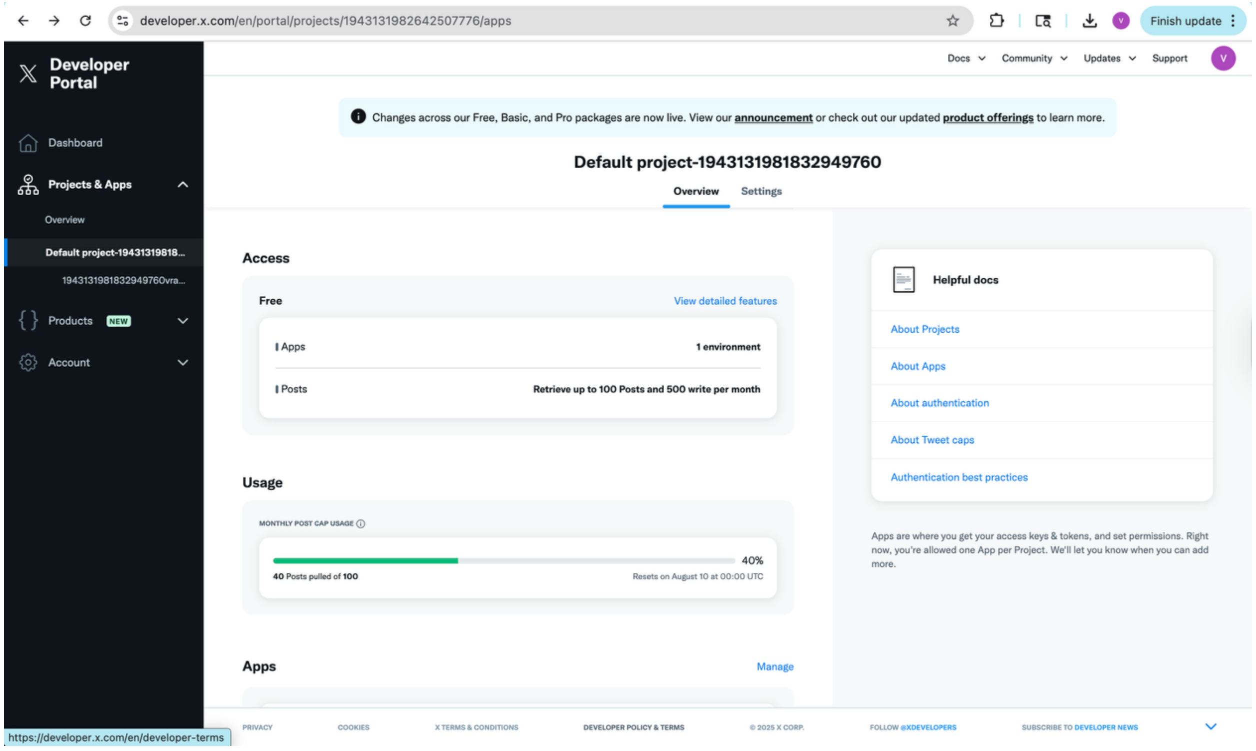
What to build

Platform overview

PRIVACY COOKIES X TERMS & CONDITIONS DEVELOPER POLICY & TERMS © 2025 X CORP. FOLLOW @XDEVELOPERS SUBSCRIBE TO DEVELOPER NEWS

Description:

X Developer Portal screenshot showing API usage and rate limits, proving proper usage of the developer account for data collection.



developer.x.com/en/portal/projects/1943131982642507776/apps

Finish update

Developer Portal

Dashboard

Projects & Apps

Overview

Default project-19431319818...

1943131981832949760vra...

Products NEW

Account

Docs Community Updates Support

Changes across our Free, Basic, and Pro packages are now live. View our **announcement** or check out our updated **product offerings** to learn more.

Default project-1943131981832949760

Overview Settings

Access

Free — View detailed features

Apps	1 environment
Posts	Retrieve up to 100 Posts and 500 write per month

Usage

MONTHLY POST CAP USAGE

40%

40 Posts pulled of 100 — Resets on August 10 at 00:00 UTC

Apps — Manage

Helpful docs

About Projects

About Apps

About authentication

About Tweet caps

Authentication best practices

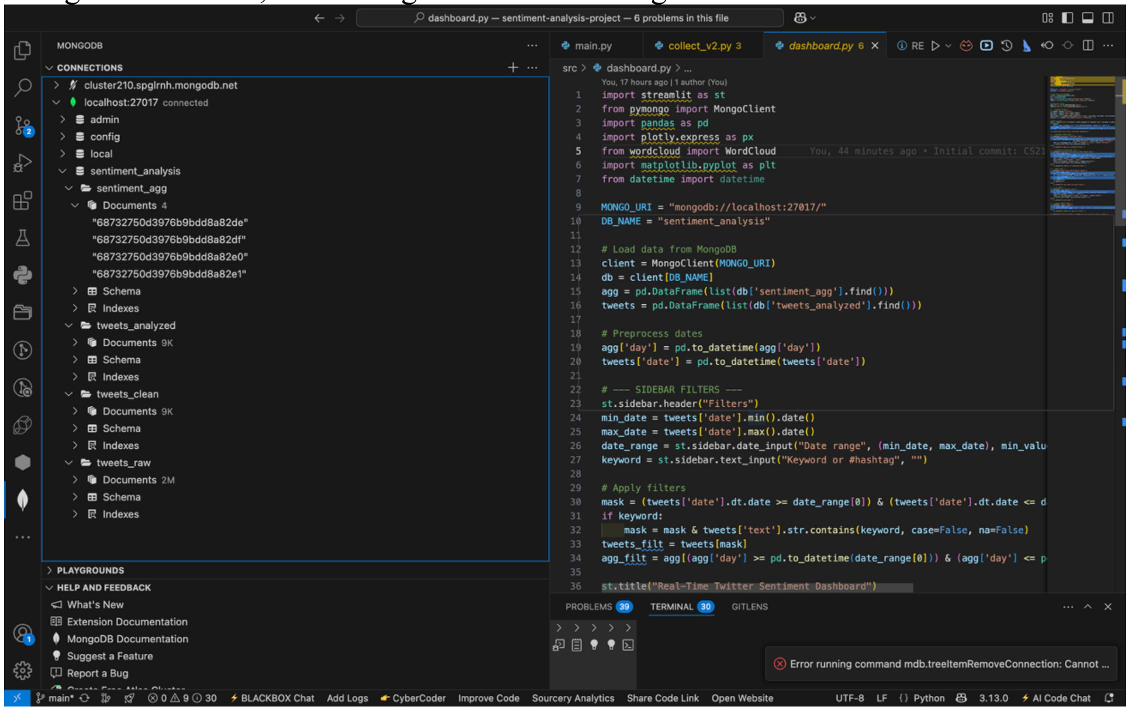
Apps are where you get your access keys & tokens, and set permissions. Right now, you're allowed one App per Project. We'll let you know when you can add more.

https://developer.x.com/en/developer-terms

PRIVACY COOKIES X TERMS & CONDITIONS DEVELOPER POLICY & TERMS © 2025 X CORP. FOLLOW @XDEVELOPERS SUBSCRIBE TO DEVELOPER NEWS

Figure 2: Loading Historical Tweets from Kaggle Dataset

Description:
Output of the main script that loads 1.6 million historical tweets from the Kaggle dataset into the MongoDB database, confirming successful data ingestion.



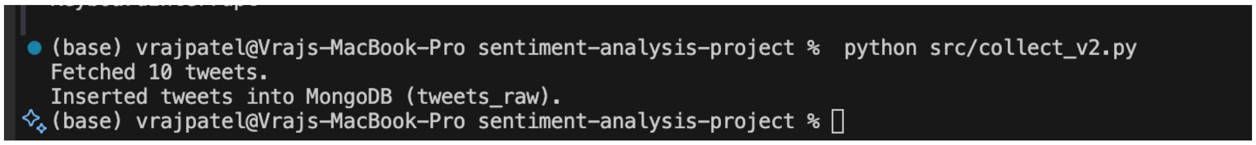
```
CONNECTIONS
> cluster210.spglrnh.mongodb.net
v localhost:27017 connected
  > admin
  > config
  > local
  v sentiment_analysis
    v sentiment_agg
      v Documents 4
          "68732750d3976b9bdd8a82de"
          "68732750d3976b9bdd8a82df"
          "68732750d3976b9bdd8a82e0"
          "68732750d3976b9bdd8a82e1"
      > Schema
      > Indexes
    v tweets_analyzed
      > Documents 9K
      > Schema
      > Indexes
    v tweets_clean
      > Documents 9K
      > Schema
      > Indexes
    v tweets_raw
      > Documents 2M
      > Schema
      > Indexes
```

```python
import streamlit as st
from pymongo import MongoClient
import pandas as pd
import plotly.express as px
from wordcloud import WordCloud
import matplotlib.pyplot as plt
from datetime import datetime

MONGO_URI = "mongodb://localhost:27017/"
DB_NAME = "sentiment_analysis"

# Load data from MongoDB
client = MongoClient(MONGO_URI)
db = client[DB_NAME]
agg = pd.DataFrame(list(db['sentiment_agg'].find()))
tweets = pd.DataFrame(list(db['tweets_analyzed'].find()))

# Preprocess dates
agg['day'] = pd.to_datetime(agg['day'])
tweets['date'] = pd.to_datetime(tweets['date'])

# --- SIDEBAR FILTERS ---
st.sidebar.header("Filters")
min_date = tweets['date'].min().date()
max_date = tweets['date'].max().date()
date_range = st.sidebar.date_input("Date range", (min_date, max_date), min_valu
keyword = st.sidebar.text_input("Keyword or #hashtag", "")

# Apply filters
mask = (tweets['date'].dt.date >= date_range[0]) & (tweets['date'].dt.date <= d
if keyword:
    mask = mask & tweets['text'].str.contains(keyword, case=False, na=False)
tweets_filt = tweets[mask]
agg_filt = agg[(agg['day'] >= pd.to_datetime(date_range[0])) & (agg['day'] <= p

st.title("Real-Time Twitter Sentiment Dashboard")
```

```
Error running command mdb.treeItemRemoveConnection: Cannot ...
```

Figure 3: Collecting Live Tweets from X API

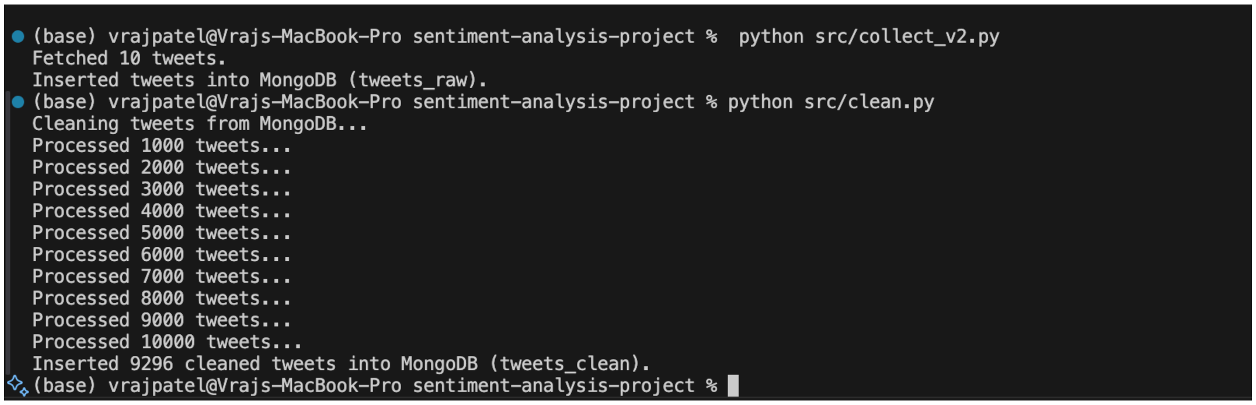
Description:
Terminal output showing the collection of live tweets from the X (Twitter) API and insertion into the tweets_raw collection in MongoDB. Demonstrates real-time tweet ingestion.



```
(base) vrajpatel@Vrajs-MacBook-Pro sentiment-analysis-project %  python src/collect_v2.py
Fetched 10 tweets.
Inserted tweets into MongoDB (tweets_raw).
(base) vrajpatel@Vrajs-MacBook-Pro sentiment-analysis-project %
```

Figure 4: Cleaning Tweets

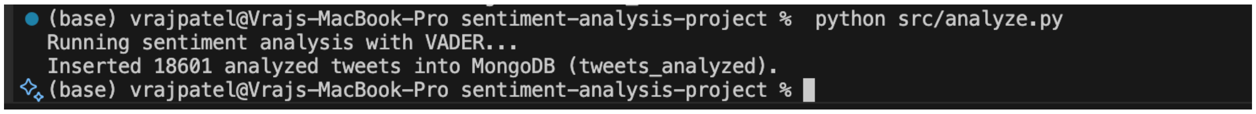
Description:
Script output from cleaning/preprocessing tweets, including removing non-English tweets, links, and special characters. Shows number of tweets processed and inserted into tweets_clean.



```
● (base) vrajpatel@Vrajs-MacBook-Pro sentiment-analysis-project %  python src/collect_v2.py
  Fetched 10 tweets.
  Inserted tweets into MongoDB (tweets_raw).
● (base) vrajpatel@Vrajs-MacBook-Pro sentiment-analysis-project % python src/clean.py
  Cleaning tweets from MongoDB...
  Processed 1000 tweets...
  Processed 2000 tweets...
  Processed 3000 tweets...
  Processed 4000 tweets...
  Processed 5000 tweets...
  Processed 6000 tweets...
  Processed 7000 tweets...
  Processed 8000 tweets...
  Processed 9000 tweets...
  Processed 10000 tweets...
  Inserted 9296 cleaned tweets into MongoDB (tweets_clean).
  (base) vrajpatel@Vrajs-MacBook-Pro sentiment-analysis-project %
```

Figure 5: Sentiment Analysis

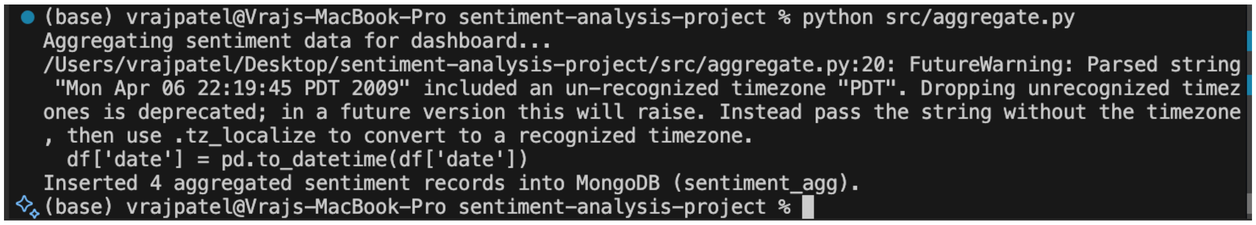
Description:
Terminal output confirming the use of VADER for sentiment analysis and successful insertion of labeled tweets into the tweets_analyzed collection.



```
● (base) vrajpatel@Vrajs-MacBook-Pro sentiment-analysis-project %  python src/analyze.py
  Running sentiment analysis with VADER...
  Inserted 18601 analyzed tweets into MongoDB (tweets_analyzed).
  (base) vrajpatel@Vrajs-MacBook-Pro sentiment-analysis-project %
```

Figure 6: Data Aggregation

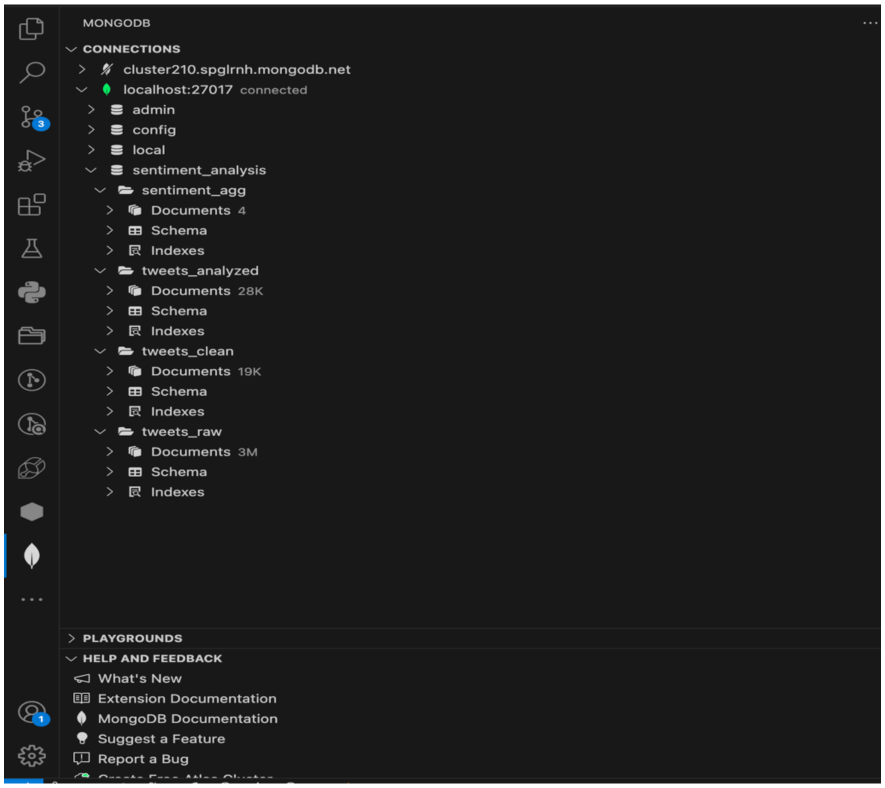
Description:
Script output for aggregating sentiment results by date, storing results for the dashboard in the sentiment_agg collection.



```
● (base) vrajpatel@Vrajs-MacBook-Pro sentiment-analysis-project % python src/aggregate.py
  Aggregating sentiment data for dashboard...
  /Users/vrajpatel/Desktop/sentiment-analysis-project/src/aggregate.py:20: FutureWarning: Parsed string
   "Mon Apr 06 22:19:45 PDT 2009" included an un-recognized timezone "PDT". Dropping unrecognized timez
  ones is deprecated; in a future version this will raise. Instead pass the string without the timezone
  , then use .tz_localize to convert to a recognized timezone.
    df['date'] = pd.to_datetime(df['date'])
  Inserted 4 aggregated sentiment records into MongoDB (sentiment_agg).
  (base) vrajpatel@Vrajs-MacBook-Pro sentiment-analysis-project %
```

Figure 7: MongoDB Collections After All Steps

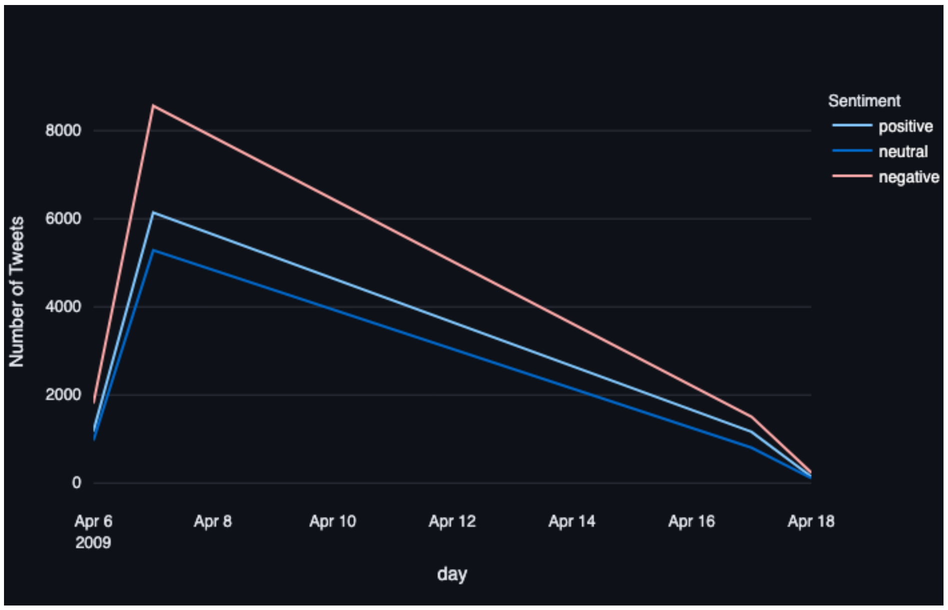
Description:
VS Code view of MongoDB Explorer, showing all collections (tweets_raw, tweets_clean, tweets_analyzed, sentiment_agg) and document counts after all processing steps.



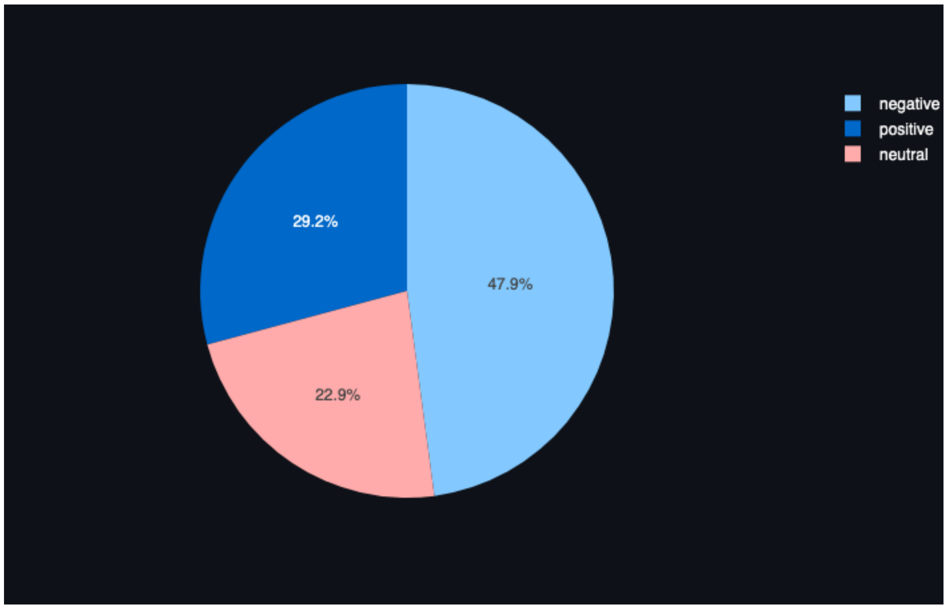
```
MONGODB
CONNECTIONS
  > cluster210.spglrnh.mongodb.net
  v localhost:27017 connected
    > admin
    > config
    > local
    v sentiment_analysis
      v sentiment_agg
        > Documents 4
        > Schema
        > Indexes
      v tweets_analyzed
        > Documents 28K
        > Schema
        > Indexes
      v tweets_clean
        > Documents 19K
        > Schema
        > Indexes
      v tweets_raw
        > Documents 3M
        > Schema
        > Indexes
> PLAYGROUNDS
v HELP AND FEEDBACK
  What's New
  Extension Documentation
  MongoDB Documentation
  Suggest a Feature
  Report a Bug
```

Dashboard (Results & Visualizations)

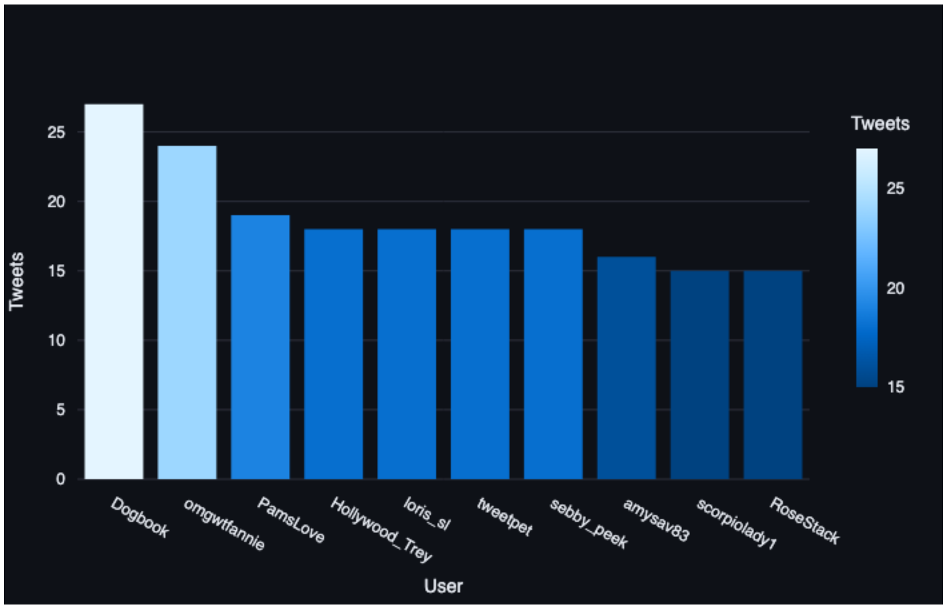
Line/area chart of sentiment over time (done).



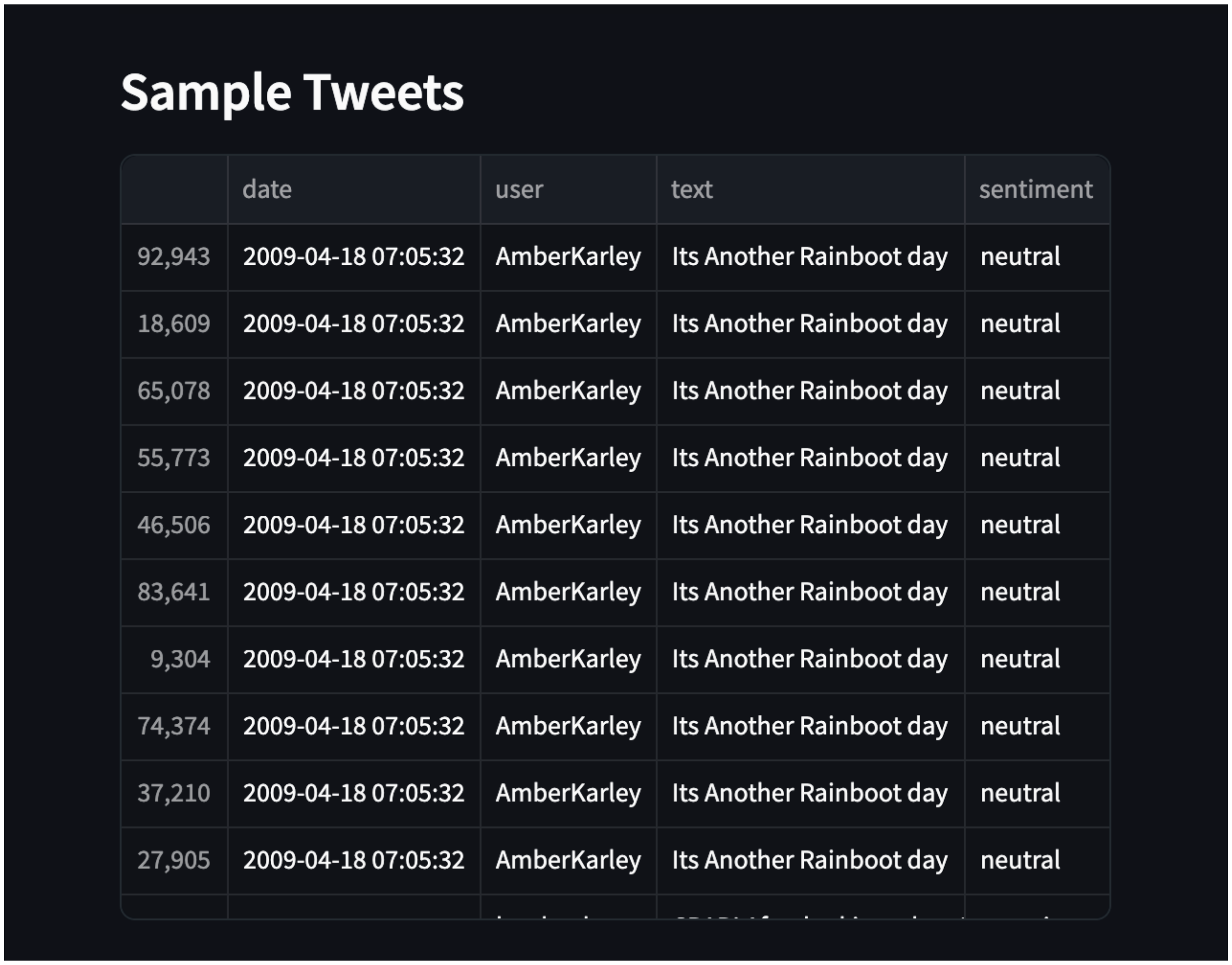
Pie chart **or** bar chart **of sentiment distribution (done).**



Top users **(done, you have bar chart).**



Sample Tweets Table



Sample Tweets

	date	user	text	sentiment
92,943	2009-04-18 07:05:32	AmberKarley	Its Another Rainboot day	neutral
18,609	2009-04-18 07:05:32	AmberKarley	Its Another Rainboot day	neutral
65,078	2009-04-18 07:05:32	AmberKarley	Its Another Rainboot day	neutral
55,773	2009-04-18 07:05:32	AmberKarley	Its Another Rainboot day	neutral
46,506	2009-04-18 07:05:32	AmberKarley	Its Another Rainboot day	neutral
83,641	2009-04-18 07:05:32	AmberKarley	Its Another Rainboot day	neutral
9,304	2009-04-18 07:05:32	AmberKarley	Its Another Rainboot day	neutral
74,374	2009-04-18 07:05:32	AmberKarley	Its Another Rainboot day	neutral
37,210	2009-04-18 07:05:32	AmberKarley	Its Another Rainboot day	neutral
27,905	2009-04-18 07:05:32	AmberKarley	Its Another Rainboot day	neutral

6. Discussion

Key Learnings:

- Real-time and batch data pipelines are both important for flexible analytics.
- Simple models like VADER are surprisingly effective for social media sentiment, especially with high-quality data cleaning.
- MongoDB offers great flexibility for text-heavy, unstructured data.
- API rate limits and data privacy (handling API keys, .env files) were important considerations.
- Visualization is crucial for making raw data actionable.

7. Differences from Proposal / Challenges

- Switched to **only VADER** (instead of BERT/transformers) for faster, more efficient processing—no GPU or heavy cloud infra needed.
- Used **only MongoDB** (no Postgres), as it worked well for all data types and project scope.
- Upgraded API integration to Twitter/X API v2 (some breaking changes vs proposal's v1.1).
- Added extra dashboard features (word clouds, user stats, interactive filters) beyond the original plan.
- Had to **exclude large CSVs** from GitHub due to file size limits; users must download data themselves from Kaggle.

8. Conclusion and Future Work

Our system demonstrates that it is possible to build a fully automated, real-time sentiment tracking dashboard using only open-source tools and free data sources. The dashboard is user-friendly, extensible, and can be adapted for other platforms (Reddit, news sites) with minor changes.
Future directions include using more advanced NLP models, adding geolocation or demographic analysis, and expanding to other languages.

9. Links

- **GitHub:** https://github.com/pvraj1904/twitter-sentiment-cs210
- **DemoVideo:****https://drive.google.com/file/d/1wZvlbirJzzbVDd3ewQMs99h9gdz2PhQe/view?usp=drive_link**
- **Dataset:** Kaggle Sentiment140
