LIVERPOOL JOHN MOORES UNIVERSITY

Sentiment Analysis on Climate Change by Using Reddit Data

Big Data Computing - 7012DATSCI

Student Name – Thet Khine Lin

Student ID – 976558

Table of Contents

Abstract

This project focuses on sentiment analysis of Reddit posts related to climate change using data collected from 2010 to 2024. Posts were retrieved by filtering with subreddit and query keywords, which are specific terms relevant to climate change. Sentiment analysis was performed using the TextBlob model, which is selected after evaluating three different sentiment models, TextBlob, VADER, and BERT, by using a sample Reddit dataset from Kaggle. Each model's accuracy was assessed, and TextBlob was chosen because it gave better accuracy score and performance across other metrics. For data collection, processing, and storage, a custom Python implementation was used instead of SparkStreaming or other streaming tools as it gave me greater customization and detailed tracing of each process. An interactive dashboard, developed with the Dash package, is used for visualization for finding insights. Key features include a line graph, a pie chart, a word cloud, and a table. This analysis revealed a steady increase in positive and neutral sentiments in climate change discussions on Reddit, while negative sentiments remained relatively stable.

Section – 1: Introduction

Social media platforms have become powerful communication channels by enabling individuals to share their opinions, thoughts, and experiences on various topics. These platforms have become significant sources of information, providing real-time insights into public sentiment. Among these platforms, Reddit is one of the popular platforms where users engage in discussions, share content, and express their views through posts and comments. With its wide range of subreddits dedicated to specific topics, Reddit offers a rich resource for understanding how people react to and feel about various issues, including climate change.

Climate change is one of the most critical challenges facing humanity today. According by United Nation, the impacts of climate change are far-reaching, affecting ecosystems, economies, and human societies globally. From extreme weather events like hurricanes, wildfires, and heatwaves to long-term consequences such as sea levels rising and biodiversity loss, climate change makes significant risks to the planet [1]. Understanding public sentiment around climate change is essential because it can shape policies, drive collective action, and influence people's attitudes toward adopting sustainable practices.

Sentiment analysis, a key application of Natural Language Processing (NLP), involves evaluating and categorizing textual data into sentiments such as positive, negative, or neutral. NLP techniques allow machines to interpret human language, which makes sentiment analysis a valuable tool for analyzing public opinions. Various models can be used for sentiment analysis, including lexicon-based models like VADER and TextBlob, or machine learning-based models like BERT.

Researchers have employed various sentiment analysis models to study climate change discussions. For example, Rosenberg et al. (2023) utilized BERT-based models to analyze sentiment in climate-related Twitter data by demonstrating the advantages of using deep learning approaches in understanding context [2]. Similarly, Qiao and Williams (2022) applied latent Dirichlet allocation (LDA) for topic modeling along with sentiment analysis to explore global warming discussions on Twitter [3].

In this project, I will evaluate the performance of three sentiment analysis models, VADER, TextBlob, and BERT, by using sample Reddit data from Kaggle. Then, the best-performing model will be selected for sentiment analysis of Reddit posts related to climate change. This approach ensures that the chosen model aligns with the nature of the Reddit data and the objectives of this analysis.

The goal of this project is to perform sentiment analysis on Reddit posts related to climate change, allowing us to gain a deeper understanding of public sentiment on the climate change issue. By analyzing the collective opinions of Reddit users, this project aims to provide valuable

insights into how climate change is perceived by the public and track any shifts and trends in sentiment over time.

Section – 2: Data Collection and Consideration

For this project, Reddit was chosen as the data source not only it provides free access to its API, unlike other platforms such as Twitter and Facebook, but also the richness and diversity of discussions on the platform. Additionally, subreddits, dedicated to specific topics, allow for targeted data collection on climate change. To handle the data from Reddit API, Python PRAW package was used because it simplifies the data collection process by efficiently handling JSON data conversion, API rate limits, and error handling.

In this project, only posts were considered for analysis, not comments. This decision was made to avoid the noise that often exists in the comments section, such as irrelevant or off-topic discussions. Analyzing the posts directly allows for more meaningful and focused data related to climate change, ensuring that the content is relevant to the project. Including comments would have introduced unnecessary complexity and noise, making the analysis less focused. By limiting the dataset to posts, the analysis stays more aligned with the project's goal to examine public sentiment about climate change.

Regarding the ethicality of data usage, the data collected for this project is publicly available on Reddit, meaning users have already agreed to Reddit's terms and conditions, which allow their posts to be accessed via the API for research and analysis purposes. Since the platform governs the data, no additional permissions were required to use the posts for this analysis. Moreover, user anonymity and privacy were inherently preserved because the Reddit API does not provide access to personal or identifiable information, which means we can only take publicly available post content and metadata.

Section – 3: Model Selection for Sentiment Analysis

To determine the most effective model for sentiment analysis in this project, I explored both lexicon-based models and machine learning-based models. Lexicon-based models, such as TextBlob and VADER, rely on predefined sentiment dictionaries to analyze text, offering simplicity and efficiency (Sham and Mohamed, 2022) [5]. In contrast, machine learning-based models like BERT utilize deep learning to understand context and nuances, making them ideal for capturing complex sentiment patterns (Sham and Mohamed, 2022) [5]. Each model offers unique advantages, and I wanted to select the one that performs the best in the context of analyzing Reddit posts related to climate change.

To test the models, I used a sample dataset from Kaggle, which consists of over 30,000 Reddit posts with sentiment labels. This dataset was useful because it provided labeled data for testing the models' performance. I randomly selected 1,000 samples from this dataset to evaluate each model based on key classification metrics, including Precision, Recall, F1-Score, and Accuracy. The formulas for these metrics are as follows

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

$$\text{F1-Score} = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

$$\text{Accuracy} = \frac{True\ Positive + True\ Negatives}{Total\ Samples}$$

Interpretation for the above metrics:

- Precision measures the proportion of correctly predicted positive results out of all predicted positive results.
- Recall measures the proportion of correctly predicted positive results out of all actual positive results in the dataset.
- F1 Score gives the harmonic mean of Precision and Recall by balancing their trade-offs.
- Accuracy measures the proportion of correctly classified results (both positives and negatives) out of all predictions.

Here are the results for each model by using Python sklearn package:

```
VADER Classification Report (Cleaned):
              precision    recall  f1-score
Negative       0.447811  0.627358  0.522593
Neutral        0.765957  0.591781  0.667697
Positive       0.695962  0.692671  0.694313
accuracy       0.642000  0.642000  0.642000

VADER Confusion Matrix:
[[293  38  92]
 [ 77 216  72]
 [ 51  28 133]]
```

Fig. 3.1: Evaluation on VADER

```
TextBlob Classification Report (Cleaned):
              precision    recall  f1-score
Negative       1.000000  0.599057  0.749263
Neutral        0.667276  1.000000  0.800439
Positive       1.000000  0.770686  0.870494
accuracy       0.818000  0.818000  0.818000

TextBlob Confusion Matrix:
[[326  97   0]
 [  0 365   0]
 [  0  85 127]]
```

Fig. 3.2: Evaluation on TextBlob

```
BERT Classification Report (Cleaned):
              precision    recall  f1-score
Negative       0.275148  0.877358  0.418919
Neutral        0.588235  0.027397  0.052356
Positive       0.514658  0.373522  0.432877
accuracy       0.354000  0.354000  0.354000

BERT Confusion Matrix:
[[158   5 260]
 [125  10 230]
 [ 24   2 186]]
```

Fig. 3.3: Evaluation on BERT

Based on the Fig 3.1, 3.2, and 3.3, we can clearly see that TextBlob gave the best model performance scores. It achieves the highest accuracy of 0.818, significantly outperforms than the other models. It also delivers strong F1-scores, particularly for the Positive and Neutral classes, with F1-scores of 0.870 and 0.800, respectively. In terms of precision and recall, TextBlob performs well across all classes, especially for Positive (precision of 1.000 and recall of 0.771) and Neutral (precision of 0.667 and recall of 1.000). Given these results, I selected TextBlob as the sentiment analysis model for my climate change sentiment analysis project.

Section – 4: Random Posts Selection

Negative: < -0.1, Neutral: between -0.1 and +0.1, Positive: > +0.1

Post 1

ID: wgy69d

Context:

Getting rid of fossil fuels is important even if anthropomorphic global warming is minimal. CO2 is about 410 ppm, up from a preindustrial level of 280 ppm. The RCP 8.5 scenario, a terribly unlikely scenario, has CO2 at 1200 ppm by 2100. Around 1000 ppm, CO2 starts having impact on human health. [https://www.earthclinic.com/cures/high-carbon-dioxide.html] (https://www.earthclinic.com/cures/high-carbon-dioxide.html). Once we get above 800, it will be difficult to keep buildings well enough ventilated to prevent health impacts.

I started looking into this when I was given a home atmosphere monitor. My home CO2 was running over 2000 ppm. I had to install an Energy Recovery Ventilator to keep CO2 down without the open windows leaving me with an uncomfortably drafty house.

Sentiment Score: -0.123

Date: 2022-08-05 15:20:26

Analysis and Connection to Project Objective:

This post discusses the importance of reducing fossil fuel use, even if anthropogenic global warming is minimal, and highlights the potential health impacts of rising CO2 levels. It incorporates both scientific data (e.g., CO2 concentration trends and scenarios like RCP 8.5) and personal experience with indoor air quality issues. The sentiment score calculated using TextBlob is slightly negative (-0.123), reflecting a tone of concern for potential health risks. This post aligns with the project objective of understanding how people engage with climate-related topics on Reddit, particularly focusing on health implications and practical solutions related to CO2 emissions.

Post 2

ID: 14mlo1x

Context:

"I live in Merida, Venezuela, and it is a city that relatively has no industries or pollution etc, besides being surrounded by mountains and a green environment in general. It has always been a cold city (for being a tropical country) but for about 2 months we have been feeling an intensity of heat never before felt, to the point that it is a collective feeling the heat and the intensity of the sun that we see and feel every day. The phenomenon started from one day to the next, and it had never happened before. My question is: can this be due to global warming or climate change? Can you really feel changes from one day to the next or from one week to another? Thank you very much, I am ignorant of the subject but I have this doubt."

Sentiment Score: -0.0371

Date: 2023-06-30 00:43:22

Analysis and Connection to Project Objective:

This post reflects personal concern and curiosity about sudden changes in the local climate. The writer, residing in Merida, Venezuela, describes an unusual increase in heat intensity in a city typically known for its cool climate, raising questions about whether these changes are due to global warming or climate change. The slightly negative sentiment score (-0.0371) calculated using TextBlob reflects a tone of unease and genuine inquiry rather than outright negativity. This aligns with the project objective, particularly by showing the emotional impact of climate change based on their own experience on climate change.

Post 3

ID: 12cf4od

Context:

"Sultan Al Jaber, CEO of Adnoc, has been tasked with the incredible responsibility of finding ways to sell energy indefinitely without planet-warming emissions. This is a monumental task for someone who previously spent much of his career making investments in renewable energy and attempting to build a zero-carbon city in the desert. His selection to lead COP28, the most important climate summit, is a testament to his commitment to finding sustainable solutions for the future. We are excited to see what he will accomplish in this role and beyond."

Sentiment Score: 0.325

Date: 2023-04-05 10:01:58

Analysis and Connection to Project Objective:

This post conveys an optimistic perspective on leadership in addressing climate change, focusing on Sultan Al Jaber's influential role in advancing renewable energy and leading global climate initiatives like COP28. The positive sentiment score (0.325), calculated using TextBlob, reflects a sense of hope and confidence in his ability to drive sustainable energy solutions. This aligns with the project objective of analyzing public sentiment on climate-related topics by showcasing how discussions of leadership, innovation, and actionable strategies can inspire constructive engagement and foster support for climate action.

Section - 5: Architecture of Batch and Streaming Data

5.1. Architecture of Batch Processing

The batch process is used in the initial data collection. It involves the following steps:

1. **Data Collection**: Collect posts from specified subreddits and query keywords until a predefined limit is reached or no new posts are available.
2. **Data Wrangling**: Clean the data by dropping duplicated posts and posts without SelfText.
3. **Backup Storage**: Save the collected data in a CSV file (File 1) as a backup to ensure the raw data is preserved.
4. **Sentiment Analysis**: Apply the TextBlob to the collected data to compute sentiment scores for each post.
5. **Processed Data Storage**: Save the data along with their computed sentiment scores in another CSV file (File 2) for further analysis.
6. **Data Visualization**: Generate plots, such as line graphs, pie charts, and bar charts, to analyze trends, sentiment distributions, and frequently used words in the dataset.

Batch Data Processing Workflow

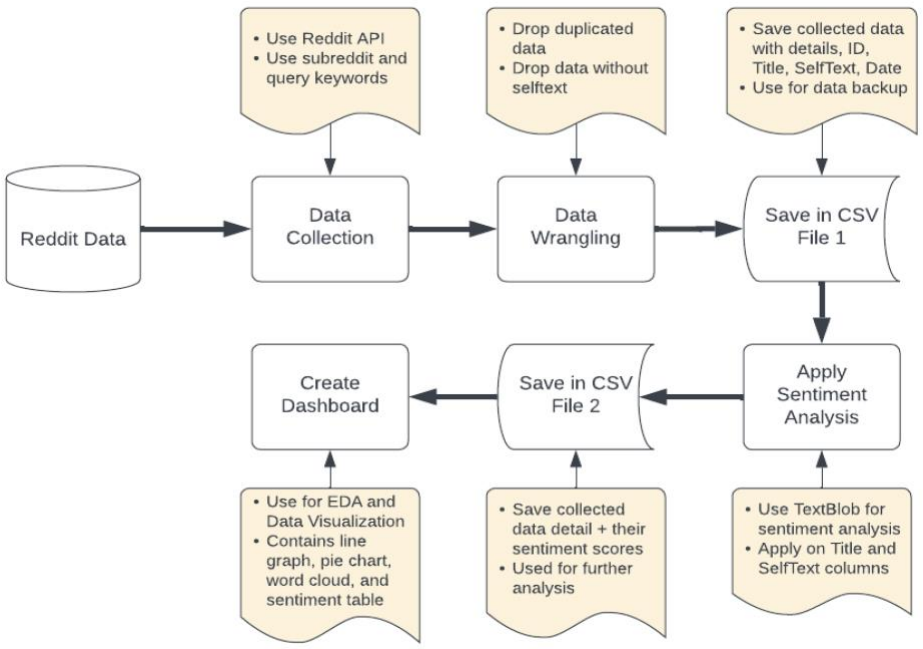


Fig. 5.1: Flowchart of Batch Data Processing

5.2. Architecture of Streaming Processing

The streaming process builds upon the batch approach to enable near-real-time data collection and analysis. It consists of the following steps:

1. **Data Collection**: Collect the posts and check whether new posts are collected or not. No new posts are collected, then the following processes are not performed.
2. **Data Wrangling**: Clean the data by dropping duplicated posts and posts without SelfText.
3. **Backup Storage**: Save the newly collected data in a CSV file (File 1) for backup.
4. **Sentiment Analysis**: Apply the SentimentIntensityAnalyzer to compute sentiment scores for the new data.
5. **Processed Data Storage**: Save the new data along with their sentiment scores in the CSV file (File 2), ensuring that the file contains an updated dataset.
6. **Interactive Dashboard Creation**: Use File 2 to create an interactive dashboard that dynamically visualizes updated trends, sentiment distributions, and insights.

The streaming process operates within a **while loop**, which repeats steps 1 to 5 at adjustable intervals (e.g., seconds or minutes). This allows continuous updates to the dataset. So, we can create a dashboard, which can provide near-real-time monitoring.

Streaming Data Processing Workflow

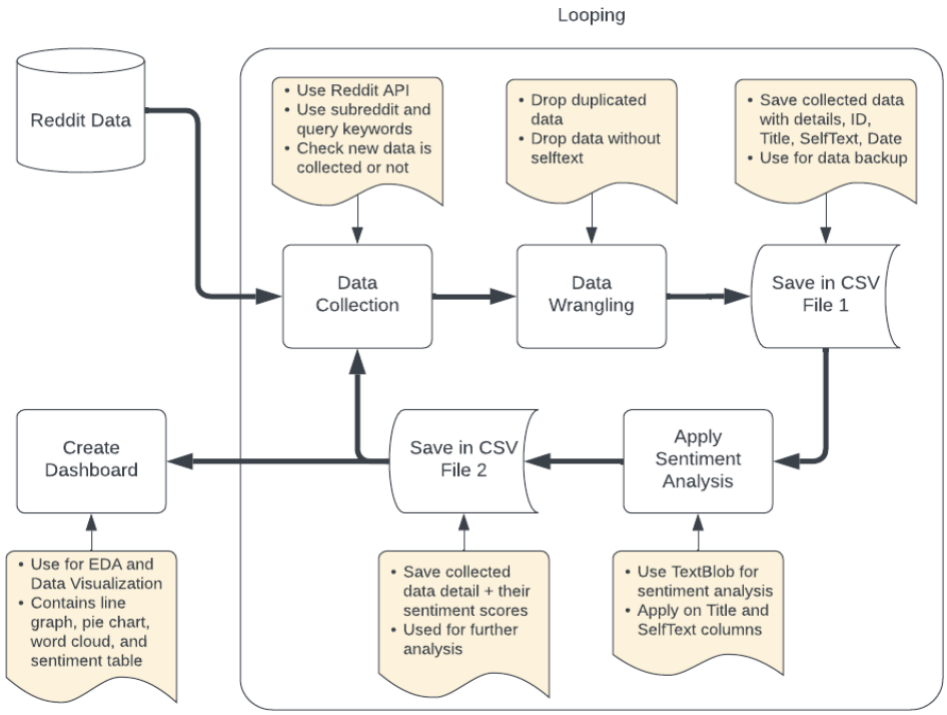


Fig. 5.2: Flowchart of Streaming Data Processing

Section – 6: Technologies and Methodologies

6.1. Programming Language and Environment

For this project, Python was chosen as the programming language because of its rich packages and ease of integration with external APIs. Additionally, Python offers a variety of packages for data manipulation, analysis, and visualization. In this project, Jupyter Notebook was used for developing and testing the code, and creating interactive dashboard.

6.2. Reddit Developer Account and API Authentication

A Reddit Developer Account was created to generate the required API credentials (client ID and secret) that allow the project to access Reddit's data. These credentials were used to authenticate the requests made to the Reddit API.

6.3. For Streaming

The explanation of the technology and methodology for the streaming data process can be divided into two main parts:

1. **Sentiment Analysis Pipeline**: This includes explaining the code responsible for Collecting Data, Storing Data, and Applying Sentiment Analysis on the collected posts.
2. **Dashboard Creation**: This focuses on explaining the code used to build an interactive dashboard for visualizing the processed data in real time.

6.3.1. Explanation on Sentiment Analysis Pipeline

Used Packages:

The project uses several Python packages for data collection, processing, and analysis:

1. **praw** is used to interact with Reddit's API to collect posts based on specific subreddits and queries.
2. **pandas** handles data manipulation, converting the collected data into DataFrames for storage in CSV files.
3. **time** package manages the intervals between requests to ensure near real-time data collection.
4. **os** helps open the CSV file and retrieve existing post IDs to prevent saving duplicate data.
5. **datetime** converts Unix timestamps into human-readable dates.
6. **textblob** performs sentiment analysis to extract insights from post content.

Data Collection Using the Reddit API:

The data collection process is carried out using the praw (Python Reddit API Wrapper) package. A user-defined function, “search_reddit_by_subreddit()”, is used to fetch posts related to climate change by specifying subreddit and query keywords. In this function, the Reddit API is accessed through the praw package, which allows easy interaction with Reddit's platform. The function searches for posts within a specific subreddit and filters out those that do not contain any text (selftext). The for loop in the function iterates through the posts by fetching them one by one until it either reaches the specified limit (1000 posts) or there are no more posts that match the query. To prevent duplication, the ID of each collected post is checked against the IDs of posts already saved in the CSV file to ensure it has not been collected previously. Once a post is confirmed to be new, it is appended to the list with its relevant details, including "ID," "Title," "SelfText," "Score," "URL," and "Date." Additionally, the function handles errors that may arise during the data fetching process, such as issues with connectivity or invalid subreddit access.

Saving Data to CSV:

A user-defined function called “save_to_csv()” was created to save the collected posts into a Pandas DataFrame. The function drops the duplicated posts and appends new posts to the CSV file 1. This CSV file serves both for further analysis and as a backup, preserving the data's integrity in case of any issues.

Applying Sentiment Analysis and Saving Results to CSV:

Sentiment analysis is applied to the newly fetched posts using a separate function, “analyze_sentiment()”, which use the TextBlob library. This function calculates the sentiment polarity of a given text, returning a score between -1 (negative) and 1 (positive). If the text is empty, it returns a neutral score of 0.0.

This is the function for applying sentiment analysis by using TextBlob model:

Explanation on Main Loop

The main loop handles the continuous process of fetching, processing, and storing Reddit posts based on specific keywords and subreddits. Below is a clearer breakdown of the steps:

1. **Defining Keywords and Subreddits:**
   - A list of query keywords and target subreddits are predefined. These are used to filter and fetch relevant Reddit posts.
2. **Avoiding Duplicates:**
   - Before collecting new posts, the script checks for an existing CSV file. If the file exists, the ID column of already-stored posts is extracted into a set (collected_ids).

This set is passed to the search_reddit_by_subreddit function to ensure only new posts are fetched.

3. **Fetching Posts:**
   - For each combination of query keyword and subreddit, posts are retrieved using the "search_reddit_by_subreddit()" function. These posts are stored as dictionaries in a Python list called new_posts. Each dictionary represents a single post with its details like ID, Title, SelfText, and Date.
4. **Saving New Posts:**
   - The list of new posts is delivered to "save_to_csv()" function. If it contains any new posts after dropping any duplicated posts, the function appends them to the CSV file 1.
5. **Applying and Storing Sentiment Analysis data:**
   - Sentiment analysis is performed on the title and selftext columns of the CSV file 1 using the "analyze_sentiment()" function and save the posts' details and the sentiment scores in the CSV file 2.
6. **Looping with Interval:**
   - The process is wrapped in a while loop to run continuously with a defined time interval. This ensures near-real-time data collection and processing.
7. **Error Handling:**
   - The while loop contain error handling process to interrupt manually with a keyboard shortcut and handle unexpected errors gracefully by printing the error message.

6.3.2. Explanation on Dashboard Creation

The dashboard is created by using the Dash package. It is designed to provide a dynamic visualization of sentiment trends. The process of creating the dashboard consists of two main components: App Layout and Callback Functions.

Used Packages:

1. **Dash** is used to **i**nitialize the web application framework for your dashboard.
2. **dash_bootstrap_components** is used to structure the layout with responsive components like dbc.Row() and dbc.Col() for better alignment and organization of elements in the dashboard.
3. **plotly** is used to create interactive visualizations like bar charts, scatter plots, and line graphs to display insights from your climate change sentiment analysis.
4. **pandas** is used to load and process the Reddit data (e.g., reading, cleaning, filtering) for analysis and visualization.
5. **base64** is used to encode image files (like word clouds) into Base64 format for embedding them directly into the dashboard without requiring external file links.

6. **io** is used to generate in-memory streams to handle dynamically created files, such as exporting word clouds or graphs as downloadable images.
7. **Wordcloud** is used to create word clouds from Reddit text data to visualize the most frequent words related to climate change.
8. **subprocess** is used to launch the dashboard automatically in the Chrome browser by executing a system command.
9. **PIL** is used to handle image-related tasks like opening and processing images (e.g., resizing or modifying word cloud outputs before embedding).

App Layout

The dashboard layout includes interactive components like dropdown menus and range sliders to filter data by year and month. The year slider enables to select a year range, while the month slider allows filtering within a specific year. The year dropdown is used to choose a particular year for a detailed month-by-month analysis. Visualizations include a line graph to show sentiment trends over time, a pie chart for sentiment distribution, a word cloud to display frequent terms, and a data table summarizing sentiment counts and data points.

Callback Functions

Callback functions enable interactivity by responding to user inputs and updating visualizations. In the "By Year" view, data is filtered by the selected year range using the year slider, and all visualizations are updated. In the "By Month" view, the data is filtered by the chosen year from the dropdown and month range from the slider, with visualizations adjusting accordingly to reflect sentiment data for the selected months.

Data Handling

The dashboard loads sentiment data from a CSV file containing columns such as Date, SelfTextSentimentScore, and SelfText. The Date column is cleaned to extract Year and Month. Sentiment scores are categorized as positive, neutral, or negative based on the sentiment score, allowing for detailed analysis of sentiment trends.

Dashboard Update Mechanism

The dashboard updates every 60 seconds using the “dcc.Interval” component, which triggers a callback to refresh the data. By dynamically reloading the dataset, the visualizations reflect newly collected data, enabling real-time monitoring and ensuring that the displayed information remains current. This part is defined in the app-layout creation.

6.4. For Batch

For batch data, the process is nearly identical to streaming, covering data collection, sentiment analysis, and saving the results in a CSV file. The dashboard creation remains the same. The key difference lies in the data processing step, the batch processing doesn't have a while loop.

Section – 7: Analysis on Batch Data

7.1. Findings from Batch Data Analysis

Histogram Findings

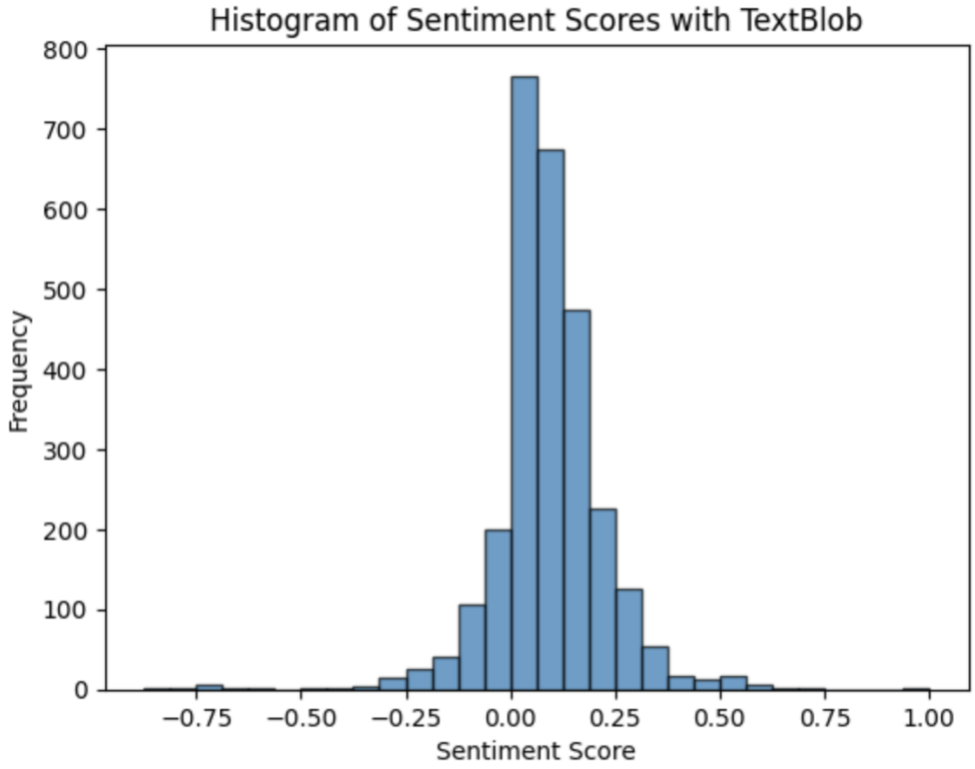
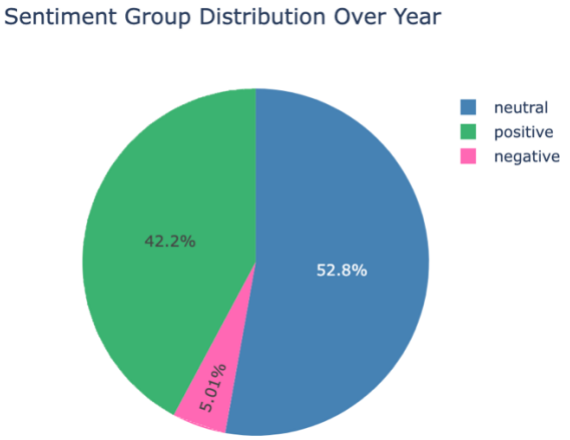


Fig. 7.1.1: Histogram of Sentiment Scores with TextBlob

Fig. 7.1.1 shows that the majority of data points are concentrated between 0.0 and 0.2, indicating a predominance of positive and neutral sentiments in the dataset. Negative sentiment, less than -0.1, appears to be minimal. This distribution forms a bell curve resembling a normal distribution, with the density peaking near neutral scores. The histogram highlights the overall public discussions towards either neutrality or slight positivity.

Pie Chart and Sentiment Table Findings



Sentiment Summary

Sentiment	Count
Negative	139
Neutral	1466
Positive	1170
Total	2775

Fig. 7.1.2: Sentiment Distribution with Pie Chart and Summary Table

Fig. 7.1.2 provides specific proportions and counts for the sentiment distribution across posts:

- Neutral sentiment accounts for 52.8% (1,466 posts) of the total, making it the most common sentiment.
- Positive sentiment comprises 42.2% (1,170 posts) of the posts.
- Negative sentiment is the smallest category, at 5.01% (139 posts).

These findings show that public opinions around the climate change topic have more neutral or optimistic tones, with relatively few negative perspectives.

Line Chart Findings with Sentiment Table data

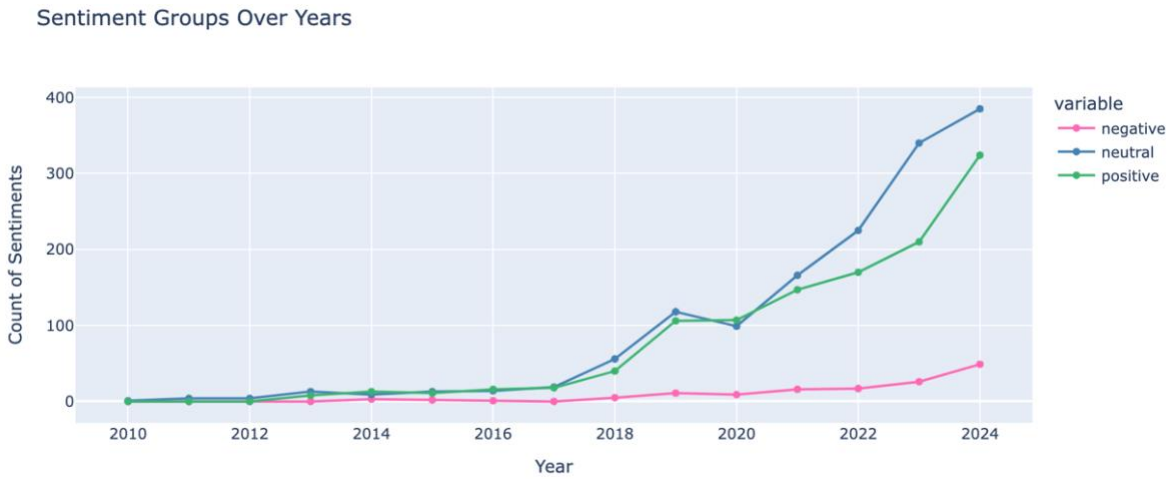


Fig. 7.1.3: Sentiment Trend Line Chart

For 2017

Sentiment Summary

Sentiment	Count
Negative	0
Neutral	19
Positive	18
Total	37

For 2018

Sentiment Summary

Sentiment	Count
Negative	5
Neutral	56
Positive	40
Total	101

For 2024

Sentiment Summary

Sentiment	Count
Negative	49
Neutral	385
Positive	324
Total	758

Fig. 7.1.3 tracks sentiment trends from 2010 to 2024. Until 2017, the total post count remained consistently low, with no significant increase in sentiment activity. For instance, in 2017, there were only 37 posts (18 positive, 19 neutral, and 0 negative).

However, starting from 2018, a marked increase is observed, rising to 101 total posts. By 2024, there are 758 total posts, with a dramatic increase in positive (324 posts) and neutral (385 posts) sentiments, while negative posts increased steadily to 49. This trend implies a growing public engagement in climate change discussions, with a great amount positive and neutral sentiment.

7.2. Possible Assumptions for Batch Data Findings

The followings are two possible assumptions that could explain the sudden increase observed in Fig. 7.1.3:

Key Global Events:

- International agreements like the 2019 UN Climate Action Summit and ongoing discussions about renewable energy and sustainability could have encouraged more optimistic discourse. Additionally, increased focus on environmental solutions may have generated neutral and positive conversations.

Rise in Reddit's Popularity:

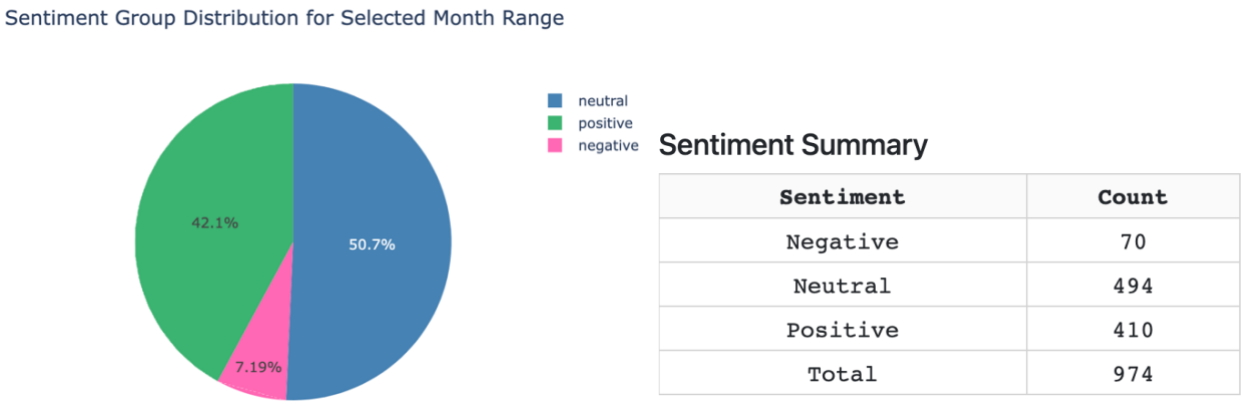
- Reddit's growing user base after the mid-2010s likely contributed to the surge in total posts. In February 2024, Reddit went public, boasting over 267.5 million weekly active users. This growth likely played in increasing public engagement in climate change discussions (Dean, 2024).

Section – 8: Analysis on Streaming Data

For this streaming data analysis, the primary challenge was the limited amount of data collected during the streaming process, with only around 150 posts available for November 2024. Because of that constraint, I decided to extend the analysis to the whole year, 2024, to gain more usable insights. By analyzing data across the entire year, I was able to observe broader trends and then focused on November 2024 to examine its sentiment activity.

8.1. Findings from Analysis on whole 2024 data

Pie Chart and Sentiment Table Findings



Sentiment Summary

Sentiment	Count
Negative	70
Neutral	494
Positive	410
Total	974

Fig. 8.1.1: Sentiment Distribution of 2024 data with Pie Chart and Summary Table

Fig 8.1.1 reveals the following sentiment distribution across posts in 2024:

- Neutral sentiment makes up 50.7% of the posts, totaling 494 posts, and emerges as the highest sentiment category.
- Positive sentiment accounts for 42.1% of the posts, corresponding to 410 posts.
- Negative sentiment is the smallest category, at 7.19%, with a total of 70 posts.

These findings suggest that public opinions on climate change in this subset of streaming data are mostly neutral or positive, with relatively few negative sentiments. This aligns with the general sentiment trend seen in the batch data analysis.

Line Graph Findings

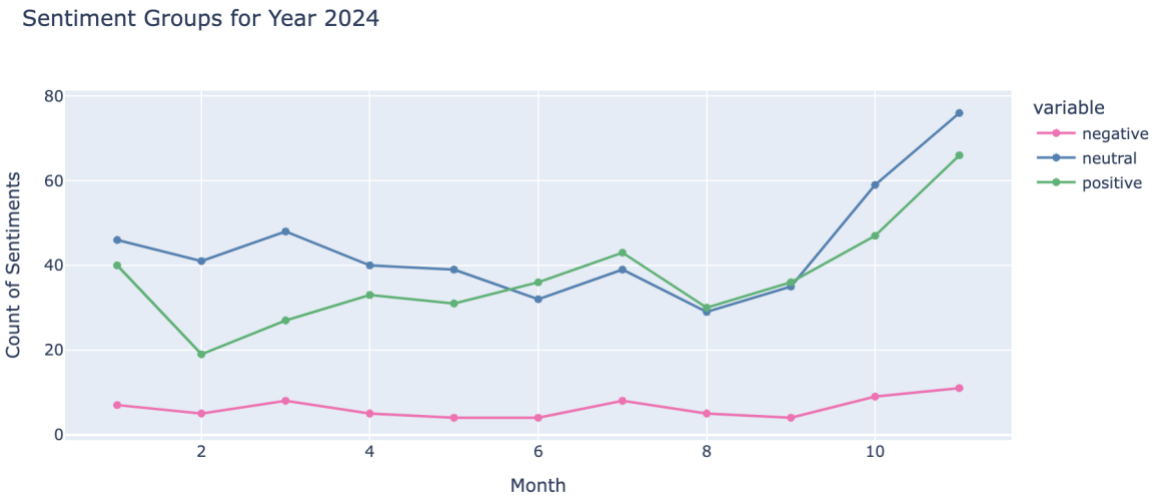


Fig. 8.1.2: Sentiment Trend Line Chart of 2024 data

X-Axis: Month (January to December)
Y-Axis: Count of sentiments in each month

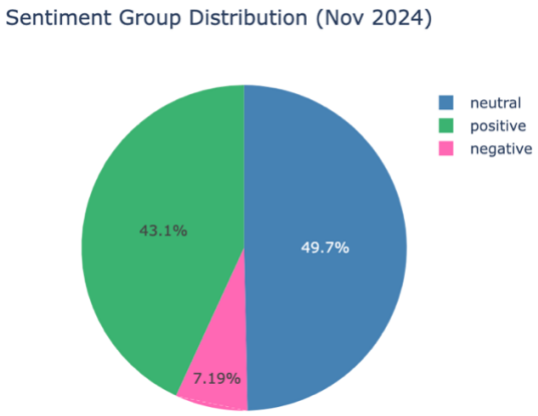
Observation:

- From January to September, the sentiment trends remained relatively steady. Positive and neutral sentiments fluctuated between 20–50 posts, while negative sentiment remained under 10 posts.
- Post-September, all sentiments saw a sharp rise. Neutral sentiment peaked at 76 posts, and positive sentiment reached 66 posts, showing a steep increase. Negative sentiment also rose but more gradually, reaching 11 posts in November.

This sudden increase in all sentiments in November warranted a deeper investigation to better understand how recent sentiment trends are going.

8.2. Findings from Analysis on November 2024 Data (Daily Interval)

Pie Chart and Sentiment Table Findings



Sentiment Summary

Sentiment	Count
Negative	11
Neutral	76
Positive	66
Total	153

Fig. 8.2.1: Sentiment Distribution of Nov 2024 data with Pie Chart and Summary Table (daily interval)

- Positive Sentiment: 43.1% (66 posts)
- Neutral Sentiment: 49.7% (76 posts)
- Negative Sentiment: 7.19% (11 posts)
- Total Posts Analyzed: 153 posts

Observation:

- The sentiment distribution pattern observed in November aligns with the whole-year findings. Neutral sentiment consistently dominated, followed by positive, with negative being the lowest.

Line Graph Findings

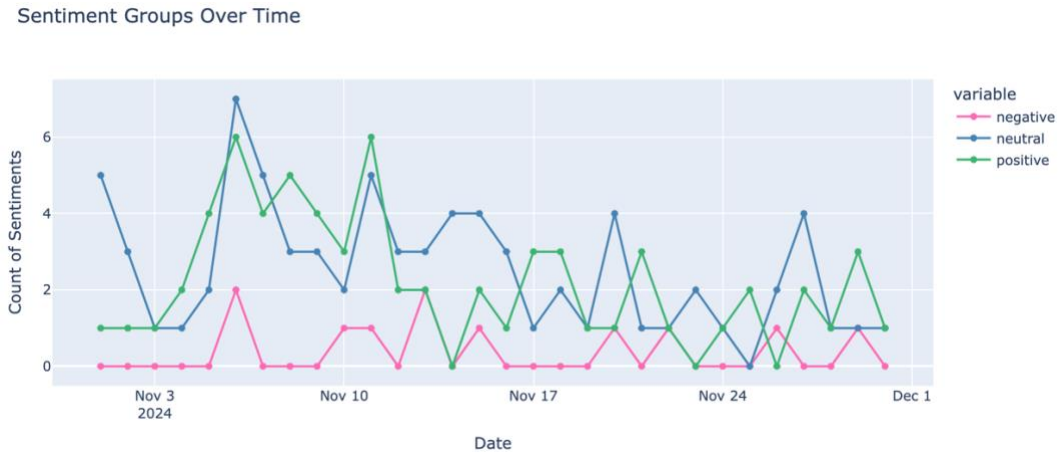


Fig. 8.2.2: Sentiment Trend Line Chart of Nov 2024 data (daily interval)

X-Axis: Date (Daily)
Y-Axis: Count of sentiments

Observation:

- While the graph hinted at a possible decrease in positive and neutral sentiments, noise in the data made the trend unclear. Negative sentiment appeared steady but lacked clarity due to daily fluctuations.

8.3. Findings from Analysis on November 2024 Data (Three-Day Interval)

To reduce noise and reveal clearer trends, data was aggregated into three-day intervals.

Pie Chart and Sentiment Table Findings

Unchanged from the daily analysis because the total data points remained constant.

Line Graph Findings

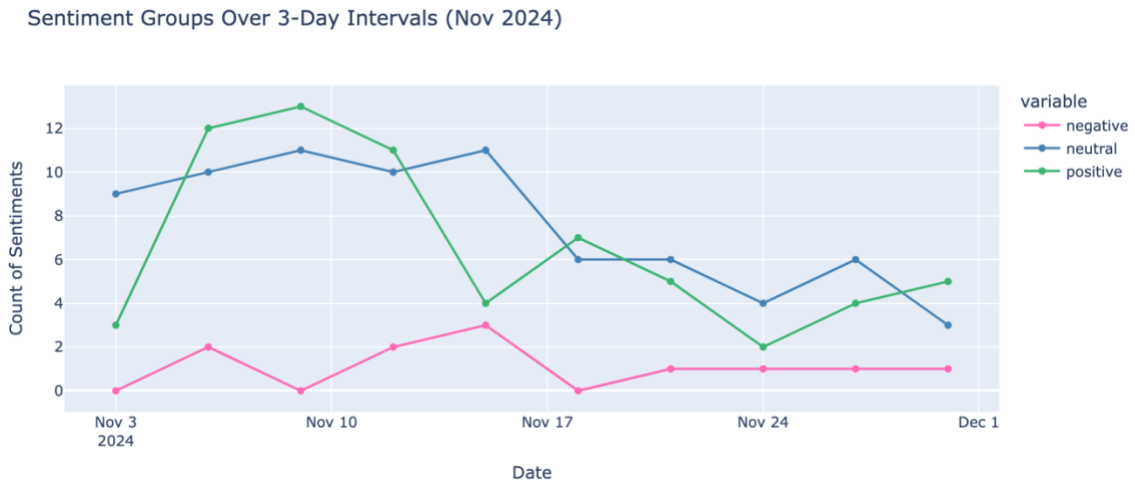


Fig. 8.3.1: Sentiment Trend Line Chart of Nov 2024 data (three-day interval)

X-Axis: Date (Three-Day Intervals)
Y-Axis: Count of sentiments in each interval

Observation:

- Positive Sentiment: Began at 3 posts on November 3, peaked at 13 posts on November 9, and showed a gradual decline, ending with 5 posts on November 30.

- Neutral Sentiment: Mirrored the positive trend, starting at 9 posts on November 3, peaking at 11 on November 15, and decreasing to 3 posts by the end of the month.
- Negative Sentiment: Displayed minimal variation, fluctuating between 0–3 posts throughout the month, remaining relatively stable.

Overall Insight:

- The aggregation uncovered a gradual decline in positive and neutral sentiments over November 2024, while negative sentiment maintained a steady presence.

Section – 9: Discussion on Findings and Limitations

9.1. Discussion on Findings with others' research

In this section, I will compare my findings with two other research papers in the field of climate change sentiment analysis. Both Research Paper 1 by Rosenberg et al. (2023) and Research Paper 2 by Qiao and Williams (2022) show that positive sentiment dominates in global warming discussions. Rosenberg et al. observed that most keywords related to Sustainable Development Goals (SDGs) had 60-70% positive tweets and pointed out positive sentiment were commonly found in his analysis on tweets related to climate action [2]. Similarly, Qiao and Williams highlighted that positive discussions about global warming are more common than negative ones and mentioned that people tend to use more positive language even for a disastrous event [3]. My findings align with these observations, as I found a growing public engagement in climate change discussions with substantial positive and neutral sentiment in batch data. Interestingly, in streaming data, while there was a gradual decline in positive and neutral sentiments in November 2024, the entire year of 2024 showed an overall increase in sentiment with November recording the highest number of posts across all sentiment categories.

9.2. Limitations for Both Batch and Streaming Data Findings

Dependence on TextBlob Sentiment Scoring

- All sentiment analyses in this study rely on TextBlob's sentiment scoring system. While TextBlob proved to be the most effective model for analyzing Reddit posts during testing, it has limitations. TextBlob primarily uses lexicon-based methods and may fail to fully capture nuanced sentiment, sarcasm, or cultural-specific expressions that could influence

the analysis. In some cases, particularly in detecting negative sentiment, other models like VADER have been shown to outperform TextBlob (Mohit, 2024).

Source Exclusivity: Reddit

- The dataset is exclusively drawn from Reddit, which may lead to a biased representation of public opinion. Reddit users tend to be a specific demographic with shared interests, and their discussions may differ significantly from those on other platforms like Twitter or Facebook.

Section – 10: Conclusion

In this project, Reddit was selected as the data source due to its popularity and free API for post collection. For sentiment analysis, TextBlob was chosen for its superior performance in precision, recall, f1-score, and accuracy. The analysis contains two parts, batch and streaming. Batch data analysis focused on initial datasets, identifying sentiment proportions and general trends. Streaming data analysis involved three stages. First, it explored sentiment trends for the entirety of 2024. Next, November data was analyzed daily but lack of clear patterns due to fluctuations, further analysis is required to perform. So, I did November data in three-day intervals, which clarified the trends by showing clear patterns. For limitation, this study relies on TextBlob for sentiment analysis, while effective, may miss nuanced expressions like sarcasm or cultural context, which effect on predicting negative sentiment. Additionally, using Reddit as the sole data source risks demographic bias, as its user base may not represent broader public sentiment across other platforms.

References

1. United Nations, n.d. *What is Climate Change?* Available at: https://www.un.org/en/climatechange/what-is-climate-change [Accessed 16 November 2024].
2. Rosenberg, E., Tarazona, C., Mallor, F., Eivazi, H., Pastor-Escuredo, D., Fuso-Nerini, F. and Vinuesa, R., 2023. Sentiment analysis on Twitter data towards climate action. Available at: https://www.sciencedirect.com/science/article/pii/S2590123023004140#br0010 [Accessed: 20 November 2024].
3. Qiao, F. and Williams, J., 2022. Topic Modelling and Sentiment Analysis of Global Warming Tweets: Evidence From Big Data Analysis. Available at: https://www.researchgate.net/publication/359327159_Topic_Modelling_and_Sentiment_Analysis_of_Global_Warming_Tweets_Evidence_From_Big_Data_Analysis [Accessed: 20 November 2024].
4. ScienceDirect, 2024. *Natural Language Processing*. Available at: https://www.sciencedirect.com/topics/computer-science/natural-language-processing [Accessed 23 November 2024].
5. Sham, N. and Mohamed, A., 2022. Climate change sentiment analysis using lexicon, machine learning and hybrid approaches. *Available at*: https://www.researchgate.net/publication/360042880 [Accessed 25 November 2024].
6. TextBlob, 2024. *TextBlob: Simplified Text Processing*. Available at: https://textblob.readthedocs.io/en/dev/ [Accessed 25 November 2024].
7. Hutto, C.J. and Gilbert, E., 2014. *VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text*. In: *Proceedings of the International Conference on Weblogs and Social Media (ICWSM)*. Available at: https://ojs.aaai.org/index.php/ICWSM/article/view/14550/14399 [Accessed 25 November 2024].
8. ProjectPro, 2024. *Transformers BART Model Explained for Text Summarization*. Available at: https://www.projectpro.io/article/transformers-bart-model-explained/553 [Accessed 25 November 2024].
9. Mohit, (2024). *Sentiment Analysis with TextBlob and VADER.* Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2021/10/sentiment-analysis-with-textblob-and-vader/ [Accessed: 30 November 2024].
10. Dean, B. (2024). *Reddit Users and Growth Stats.* Backlinko. Available at: https://backlinko.com/reddit-users [Accessed: 30 November 2024].
11. OpenAI, ChatGPT, assistance in coding for data collection, sentiment analysis, and dashboard creation.

Appendix

Code for Data Collection and Sentiment Analysis for Streaming Data Process

```
import praw
import pandas as pd
import time
import os
from datetime import datetime
from textblob import TextBlob  # Import TextBlob

# Reddit API credentials
client_id = 'wvFf0PPQo_1zth6SkDESUQ'
client_secret = 'krMR9Hj-IILecymbX_kf4SdLHNv6Gg'
user_agent = 'climate_sentiment_analyzer:v1.0 (by /u/Ok_Beginning1171)'

# Authenticate with Reddit API using praw
reddit = praw.Reddit(
    client_id=client_id,
    client_secret=client_secret,
    user_agent=user_agent
)
```

```
# Function to search Reddit by subreddit
def search_reddit_by_subreddit(query, subreddit, limit=1000, collected_ids=set()):
    posts = []
    try:
        subreddit_instance = reddit.subreddit(subreddit)
        for submission in subreddit_instance.search(query, sort='new', limit=limit):
            if submission.id not in collected_ids and submission.selftext.strip():
                posts.append({
                    'ID': submission.id,
                    'Title': submission.title,
                    'SelfText': submission.selftext,
                    'Date': datetime.utcfromtimestamp(submission.created_utc).strftime('%Y-%m-%d %H:%M:%S')
                })
    except Exception as e:
        print(f"Error fetching data from subreddit r/{subreddit}: {e}")
    return posts

# Function to save posts to CSV
def save_to_csv(df_new_posts, filename):
    df_updated = pd.DataFrame()
    if os.path.exists(filename):
        df_existing = pd.read_csv(filename)
        df_updated = pd.concat([df_existing, df_new_posts]).drop_duplicates(subset=['ID']).reset_index(drop=True)
    df_updated.to_csv(filename, index=False)
    return len(df_new_posts), len(df_updated)

# Define a function for sentiment scoring using TextBlob
def analyze_sentiment(text):
    if not text:
        return 0.0  # Neutral score if no text
    blob = TextBlob(text)
    return blob.sentiment.polarity  # Polarity score: -1 (negative) to 1 (positive)
```

```
if __name__ == "__main__":
    queries = ["climate change", "global warming", "greenhouse gases", "carbon emissions", "renewable energy",
               "deforestation", "sea level rise", "extreme weather", "climate action", "fossil fuels",
               "deforestation", "carbonfoot print", "united nations"]

    subreddits = ["unitednations", "climatechange", "change", "heat", "weather", "environment", "sustainability", "renewableenergy", "conservation"]

    filename = 'climate_change_posts_test_4.csv'
    filename_sa = "climate_change_sentiment_analysis_test_4.csv"
    collected_ids = set()

    while True:
        try:
            print("Fetching data from Reddit...")
            new_posts = []

            for subreddit in subreddits:
                print(f"Searching in subreddit: r/{subreddit}...")
                for query in queries:
                    if os.path.exists(filename):
                        df_existing = pd.read_csv(filename)
                        if 'ID' in df_existing.columns:
                            collected_ids = set(df_existing['ID'].values)

                    subreddit_posts = search_reddit_by_subreddit(query, subreddit, collected_ids=collected_ids)
                    new_posts.extend(subreddit_posts)
                print(f"r/{subreddit}: Done")

            df_new_posts = pd.DataFrame(new_posts).drop_duplicates(subset=['ID'])
```

```
            if df_new_posts.empty:
                print("No new posts found.")

            else:
                # Save new posts to CSV
                new_count, total_count = save_to_csv(df_new_posts, filename=filename)

                # Perform sentiment analysis
                df_new_posts['TitleSentimentScore'] = df_new_posts['Title'].apply(analyze_sentiment)
                df_new_posts['SelfTextSentimentScore'] = df_new_posts['SelfText'].apply(analyze_sentiment)

                # Save sentiment analysis to a new CSV
                header = not os.path.exists(filename_sa)
                df_new_posts.to_csv(filename_sa, mode='a', index=False, header=header)

                print(f"Appended {new_count} new posts. Total posts: {total_count}")

            print("Waiting for the next interval...")
            time.sleep(5)

        except KeyboardInterrupt:
            print("\nScript interrupted. Exiting...")
            break
        except Exception as e:
            print(f"Unexpected error: {e}")
            break
```

Code for Data Collection and Sentiment Analysis for Batch Data Process

```
import praw
import pandas as pd
import time
import os
from datetime import datetime
from textblob import TextBlob  # Import TextBlob

# Reddit API credentials
client_id = 'wvFf0PPQo_1zth6SkDESUQ'
client_secret = 'krMR9Hj-IILecymbX_kf4SdLHNv6Gg'
user_agent = 'climate_sentiment_analyzer:v1.0 (by /u/Ok_Beginning1171)'

# Authenticate with Reddit API using praw
reddit = praw.Reddit(
    client_id=client_id,
    client_secret=client_secret,
    user_agent=user_agent
)
```

```python
# Function to search Reddit by subreddit
def search_reddit_by_subreddit(query, subreddit, limit=1000, collected_ids=set()):
    posts = []
    try:
        subreddit_instance = reddit.subreddit(subreddit)
        for submission in subreddit_instance.search(query, sort='new', limit=limit):
            if submission.id not in collected_ids and submission.selftext.strip():
                posts.append({
                    'ID': submission.id,
                    'Title': submission.title,
                    'SelfText': submission.selftext,
                    'Score': submission.score,
                    'URL': submission.url,
                    'Date': datetime.utcfromtimestamp(submission.created_utc).strftime('%Y-%m-%d %H:%M:%S')
                })
        print(f"Collected {len(posts)} posts from r/{subreddit} for {query}.")
    except Exception as e:
        print(f"Error fetching data from subreddit r/{subreddit}: {e}")
    return posts

# Function to save posts to CSV
def save_to_csv(new_posts, filename):
    df = pd.DataFrame(new_posts)
    if os.path.exists(filename):
        df_existing = pd.read_csv(filename)
        df = pd.concat([df_existing, df]).drop_duplicates(subset=['ID']).reset_index(drop=True)
    df.to_csv(filename, index=False)
    return len(new_posts), len(df)
```

```python
# Define a function for sentiment scoring using TextBlob
def analyze_sentiment(text):
    if not text:
        return 0.0  # Neutral score if no text
    blob = TextBlob(text)
    return blob.sentiment.polarity  # Polarity score: -1 (negative) to 1 (positive)

if __name__ == "__main__":
    queries = ["climate change", "global warming", "greenhouse gases", "carbon emissions", "renewable energy",
               "deforestation", "sea level rise", "extreme weather", "climate action", "fossil fuels",
               "deforestation", "carbonfoot print", "united nations"]

    subreddits = ["unitednations", "climatechange", "change", "heat", "weather", "environment", "sustainability", "renewableenergy", "conservation"]

    filename = 'climate_change_posts_TextBlob_t.csv'
    filename_sa = "climate_change_sentiment_analysis_TextBlob_t.csv"
    collected_ids = set()

    try:
        print("Fetching data from Reddit...")
        new_posts = []

        for subreddit in subreddits:
            print(f"Searching in subreddit: r/{subreddit}...")
            for query in queries:
                if os.path.exists(filename):
                    df_existing = pd.read_csv(filename)
                    if 'ID' in df_existing.columns:
                        collected_ids = set(df_existing['ID'].values)

                subreddit_posts = search_reddit_by_subreddit(query, subreddit, collected_ids=collected_ids)
                new_posts.extend(subreddit_posts)

        df_new_posts = pd.DataFrame(new_posts).drop_duplicates(subset=['ID'])
```

```python
        if df_new_posts.empty:
            print("No new posts found.")

        else:
            # Save new posts to CSV
            new_count, total_count = save_to_csv(df_new_posts, filename=filename)

            # Perform sentiment analysis
            df_new_posts['TitleSentimentScore'] = df_new_posts['Title'].apply(analyze_sentiment)
            df_new_posts['SelfTextSentimentScore'] = df_new_posts['SelfText'].apply(analyze_sentiment)

            # Save sentiment analysis to a new CSV
            header = not os.path.exists(filename_sa)
            df_new_posts.to_csv(filename_sa, mode='a', index=False, header=header)

            print(f"Appended {new_count} new posts. Total posts: {total_count}")

    except Exception as e:
        print(f"Unexpected error: {e}")
```

Code for Dashboard Creation

```
from dash import Dash, dcc, html, Input, Output, dash_table
import dash_bootstrap_components as dbc
import plotly.express as px
import pandas as pd
import base64
from io import BytesIO
from wordcloud import WordCloud, STOPWORDS
import subprocess
from PIL import Image

# Initialize the app
app = Dash(__name__, external_stylesheets=[dbc.themes.BOOTSTRAP])
```

```
# Function to load and clean the latest data
def load_data():

    # Load the data
    df = pd.read_csv('climate_change_sentiment_analysis_test_4.csv')
    #df = pd.read_csv('streaming_sentiment_analysis.csv')

    # Ensure 'Date' column exists and is in datetime format
    if 'Date' in df.columns:
        df['Date'] = pd.to_datetime(df['Date'], errors='coerce')  # Coerce invalid formats to NaT
    else:
        raise KeyError("The 'Date' column is missing in the dataset.")

    # Drop rows with invalid or missing 'Date' values
    df = df.dropna(subset=['Date'])

    # Normalize 'Date' column (strip time information)
    df['Date'] = df['Date'].dt.date

    # Extract year and month from 'Date'
    df['Year'] = df['Date'].apply(lambda x: x.year)
    df['Month'] = df['Date'].apply(lambda x: x.month)
    df['sentiment_group'] = pd.cut(df['SelfTextSentimentScore'], bins=[-float('inf'), -0.1, 0.1, float('inf')],
                                   labels=['negative', 'neutral', 'positive'])

    return df
```

```
# Load the data once
df = load_data()

# Function to generate word cloud image
def generate_wordcloud(text):
    wordcloud = WordCloud(width=400, height=400, background_color='white', stopwords=STOPWORDS, max_words= 50).generate(text)
    img = BytesIO()
    wordcloud.to_image().save(img, format='PNG')
    img.seek(0)
    return f"{base64.b64encode(img.getvalue()).decode()}"

# Layout for the app
app.layout = dbc.Container([
    dbc.Row([
        dbc.Col(html.H1("Climate Change Sentiment Analysis Dashboard"), className="mb-2")
    ]),

    # Dropdown for Year or Month View
    dbc.Row([
        dcc.Dropdown(
            id='view-selector',
            options=[
                {'label': 'By Year', 'value': 'year'},
                {'label': 'By Month', 'value': 'month'}
            ],
            value='year',
            clearable=False
        )
    ]),
```

```
    # Year Slider (Visible for "By Year")
    dbc.Row([
        html.Div(
            dcc.RangeSlider(
                id='year-slider',
                min=df['Year'].min(),
                max=df['Year'].max(),
                step=1,
                value=[df['Year'].min(), df['Year'].max()],
                marks={year: str(year) for year in range(df['Year'].min(), df['Year'].max() + 1)}
            ),
            id='year-slider-container',
            style={'display': 'block'}
        )
    ]),

    # Year Dropdown (Visible for "By Month")
    dbc.Row([
        html.Div(
            dcc.Dropdown(
                id='year-dropdown',
                options=[{'label': str(year), 'value': year} for year in sorted(df['Year'].unique(), reverse=True)],
                value=df['Year'].max(),
                clearable=False
            ),
            id='year-dropdown-container',
            style={'display': 'none'}
        )
    ]),
```

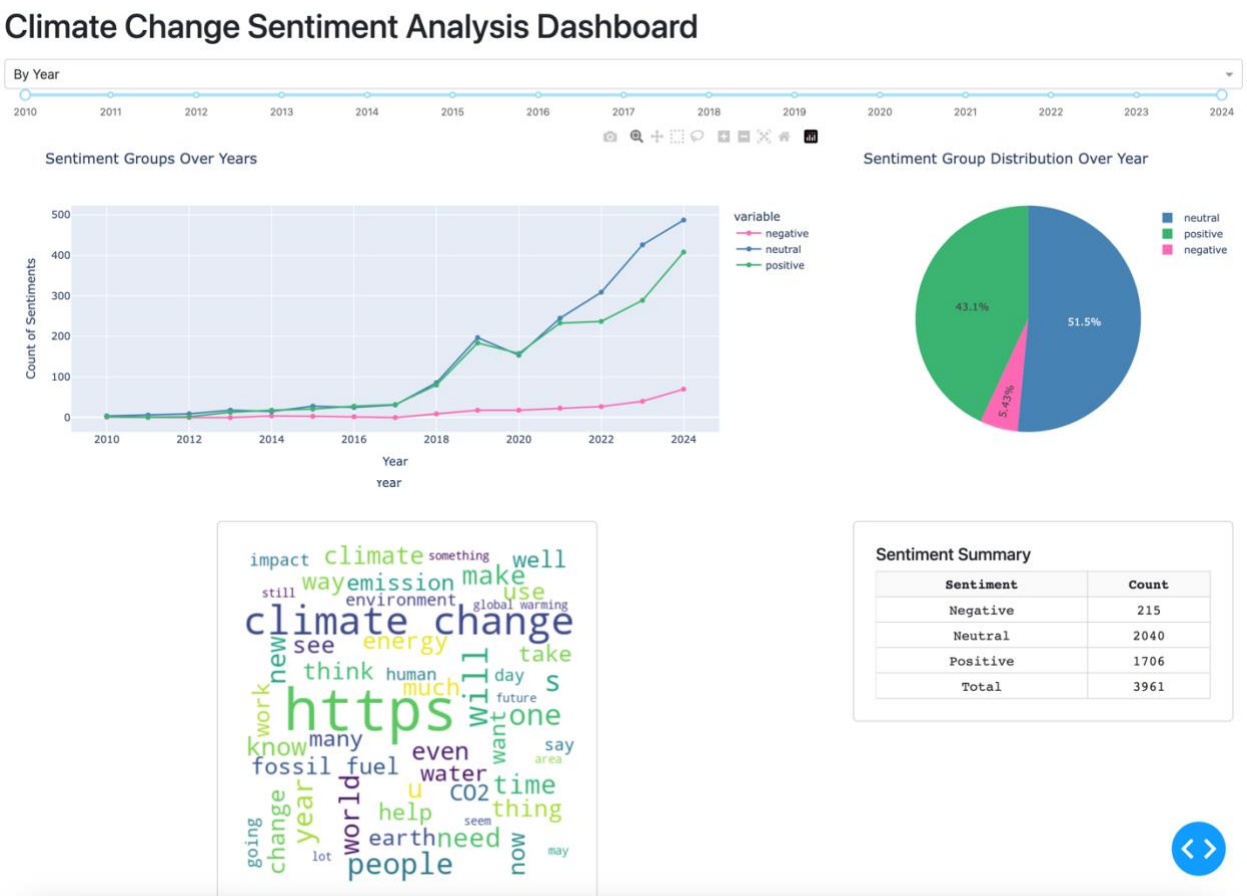
```
    # Range Slider for Month Selection
    dbc.Row([
        html.Div(
            dcc.RangeSlider(
                id='month-slider',
                min=1,
                max=12,
                step=1,
                value=[1, 12],
                marks={month: str(month) for month in range(1, 13)}
            ),
            id='month-slider-container',
            style={'display': 'none'}
        )
    ]),

    dbc.Row([
        dbc.Col(dcc.Graph(id='line-graph'), width=8),
        dbc.Col(dcc.Graph(id='pie-chart'), width=4),
    ]),


    dbc.Row([
        # Word Cloud
        dbc.Col([
            dbc.Card(
                dbc.CardBody([
                    html.Img(id='word-cloud', style={'width': '400px', 'height': '400px', 'display': 'block', 'margin': 'auto'})
                ]),
                style={"width": "450px", "margin": "auto", "border": "1px solid #ccc", "padding": "10px"}
            ),
        ], width=8),
```

```
        # Adding a DataTable for the sentiment counts
        dbc.Col([
            dbc.Card(
                dbc.CardBody([
                    html.H5("Sentiment Summary", className="card-title"),
                    dash_table.DataTable(
                        id='sentiment-table',
                        columns=[
                            {"name": "Sentiment", "id": "sentiment"},
                            {"name": "Count", "id": "count"}
                        ],
                        style_table={'overflowX': 'auto'},
                        style_cell={'textAlign': 'center'},
                        style_header={'fontWeight': 'bold'}
                    )
                ]),
                style={"width": "450px", "margin": "auto", "border": "1px solid #ccc", "padding": "10px"}
            ),
        ], width=4)
    ]),

    dbc.Row([
        dbc.Col(dcc.Interval(id='interval-component', interval=60000, n_intervals=0))
    ])
], fluid=True)
```

```python
# Combined Callback for both Graphs, Word Cloud, Table and Styles
@app.callback(
    [
        Output('line-graph', 'figure'),
        Output('pie-chart', 'figure'),
        Output('year-slider-container', 'style'),
        Output('year-dropdown-container', 'style'),
        Output('month-slider-container', 'style'),
        Output('word-cloud', 'src'),
        Output('sentiment-table', 'data')
    ],
    [
        Input('view-selector', 'value'),
        Input('year-slider', 'value'),
        Input('year-dropdown', 'value'),
        Input('month-slider', 'value'),
        Input('interval-component', 'n_intervals')
    ]
)
def update_graphs_and_styles(view_type, year_range, selected_year, month_range, n_intervals):
    # Prepare styles
    year_slider_style = {'display': 'block'} if view_type == 'year' else {'display': 'none'}
    year_dropdown_style = {'display': 'block'} if view_type == 'month' else {'display': 'none'}
    month_slider_style = {'display': 'block'} if view_type == 'month' else {'display': 'none'}

    df = load_data()

    # Filter Data
    if view_type == 'year':
        filtered_df = df[(df['Year'] >= year_range[0]) & (df['Year'] <= year_range[1])]

        # Line Graph
        grouped = (filtered_df.groupby(['Year', 'sentiment_group'], observed=False)
                   .size()
                   .unstack(fill_value=0)
                   .reset_index())

        line_fig = px.line(grouped, x='Year', y=['negative', 'neutral', 'positive'],
                           labels={'value': 'Count of Sentiments', 'Year': 'Year'},
                           title="Sentiment Groups Over Years",
                           markers=True,
                           color_discrete_map={'negative': 'hotpink', 'neutral': 'steelblue', 'positive': 'mediumseagreen'})

        # Pie Chart
        sentiment_counts = filtered_df['sentiment_group'].value_counts().reset_index()
        sentiment_counts.columns = ['sentiment_group', 'count']

        pie_fig = px.pie(
            sentiment_counts,
            names='sentiment_group',
            values='count',
            title="Sentiment Group Distribution Over Year",
            color='sentiment_group',  # Optional: Assign colors to categories
            color_discrete_map={'negative': 'hotpink', 'neutral': 'steelblue', 'positive': 'mediumseagreen'}
        )
        # Word Cloud
        text = ' '.join(filtered_df['SelfText'].dropna())
        wordcloud_img = generate_wordcloud(text)

        # Sentiment Counts Table
        sentiment_table_data = [
                    {"sentiment": "Negative", "count": sentiment_counts.loc[sentiment_counts['sentiment_group'] == 'negative', 'count'].values[0]
                     if sentiment_counts.loc[sentiment_counts['sentiment_group'] == 'negative', 'count'].size > 0 else 0},
                    {"sentiment": "Neutral", "count": sentiment_counts.loc[sentiment_counts['sentiment_group'] == 'neutral', 'count'].values[0]
                     if sentiment_counts.loc[sentiment_counts['sentiment_group'] == 'neutral', 'count'].size > 0 else 0},
                    {"sentiment": "Positive", "count": sentiment_counts.loc[sentiment_counts['sentiment_group'] == 'positive', 'count'].values[0]
                     if sentiment_counts.loc[sentiment_counts['sentiment_group'] == 'positive', 'count'].size > 0 else 0},
                    {"sentiment": "Total", "count": sentiment_counts['count'].sum()},
                ]

        return line_fig, pie_fig, year_slider_style, year_dropdown_style, month_slider_style, wordcloud_img, sentiment_table_data

    elif view_type == 'month':
        filtered_df = df[(df['Year'] == selected_year) &
                         (df['Month'] >= month_range[0]) &
                         (df['Month'] <= month_range[1])]

        # Line Graph
        grouped = (filtered_df.groupby(['Month', 'sentiment_group'], observed=False)
                   .size()
                   .unstack(fill_value=0)
                   .reset_index())

        line_fig = px.line(grouped, x='Month', y=['negative', 'neutral', 'positive'],
                           labels={'value': 'Count of Sentiments', 'Month': 'Month'},
                           title=f"Sentiment Groups for Year {selected_year}",
                           markers=True,
                           color_discrete_map={'negative': 'hotpink', 'neutral': 'steelblue', 'positive': 'mediumseagreen'})
```

```
        # Pie Chart
        sentiment_counts = filtered_df['sentiment_group'].value_counts().reset_index()
        sentiment_counts.columns = ['sentiment_group', 'count']

        pie_fig = px.pie(
            sentiment_counts,
            names='sentiment_group',
            values='count',
            title="Sentiment Group Distribution for Selected Month Range",
            color='sentiment_group',  # Optional: Assign colors to categories
            color_discrete_map={'negative': 'hotpink', 'neutral': 'steelblue', 'positive': 'mediumseagreen'}
        )

        # Word Cloud
        text = ' '.join(filtered_df['SelfText'].dropna())
        wordcloud_img = generate_wordcloud(text)

        # Sentiment Counts Table
        sentiment_table_data = [
                    {"sentiment": "Negative", "count": sentiment_counts.loc[sentiment_counts['sentiment_group'] == 'negative', 'count'].values[0]
                     if sentiment_counts.loc[sentiment_counts['sentiment_group'] == 'negative', 'count'].size > 0 else 0},
                    {"sentiment": "Neutral", "count": sentiment_counts.loc[sentiment_counts['sentiment_group'] == 'neutral', 'count'].values[0]
                     if sentiment_counts.loc[sentiment_counts['sentiment_group'] == 'neutral', 'count'].size > 0 else 0},
                    {"sentiment": "Positive", "count": sentiment_counts.loc[sentiment_counts['sentiment_group'] == 'positive', 'count'].values[0]
                     if sentiment_counts.loc[sentiment_counts['sentiment_group'] == 'positive', 'count'].size > 0 else 0},
                    {"sentiment": "Total", "count": sentiment_counts['count'].sum()},
                ]

        return line_fig, pie_fig, year_slider_style, year_dropdown_style, month_slider_style, wordcloud_img, sentiment_table_data

if __name__ == '__main__':
    app.run_server(debug=True, port = 8080)
    subprocess.Popen(['/Applications/Google Chrome.app/Contents/MacOS/Google Chrome', 'http://127.0.0.1:8080/'])
```

Streaming Data Dashboard (with all the posts collected)



Code for Evaluating Sentiment Analysis Models

```
import pandas as pd
# Update the file path to include the full path to the dataset file
df = pd.read_csv("Reddit_Data.csv")

# Remove rows with NaN or None in the 'clean_comment' column
df = df[df['clean_comment'].notna()]

# Reduce the dataset to 300 samples
df = df.sample(n=1000, random_state=42).reset_index(drop=True)

# Map category values to sentiment labels
sentiment_map = {-1: "Negative", 0: "Neutral", 1: "Positive"}
df['Sentiment_Label'] = df['category'].map(sentiment_map)

df.head(5)
```

```
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
from textblob import TextBlob
from transformers import pipeline
from sklearn.metrics import classification_report, confusion_matrix
import pandas as pd

# Initialize sentiment analysis models
vader = SentimentIntensityAnalyzer()
bert = pipeline("sentiment-analysis", model="distilbert-base-uncased-finetuned-sst-2-english")

# Define VADER prediction function with standard ranges
def predict_vader(text):
    compound = vader.polarity_scores(text)['compound']
    if compound > 0.05:
        return "Positive"
    elif compound < -0.05:
        return "Negative"
    else:
        return "Neutral"

# Define TextBlob prediction function with standard ranges
def predict_textblob(text):
    polarity = TextBlob(text).sentiment.polarity
    if polarity > 0.1:  # Updated threshold for positive
        return "Positive"
    elif polarity < -0.1:  # Updated threshold for negative
        return "Negative"
    else:
        return "Neutral"
```

```
# Define BERT prediction function with standard ranges
def predict_bert(text):
    result = bert(text[:512])[0]  # Limit text to 512 tokens
    label = result['label']
    score = result['score']

    if score < 0.6:  # Low confidence means Neutral
        return "Neutral"
    elif label == "POSITIVE":
        return "Positive"
    elif label == "NEGATIVE":
        return "Negative"

# Apply sentiment analysis models to the dataset
df['VADER_Predicted'] = df['clean_comment'].apply(predict_vader)
df['TextBlob_Predicted'] = df['clean_comment'].apply(predict_textblob)
df['BERT_Predicted'] = df['clean_comment'].apply(predict_bert)

# Define function to generate metrics and confusion matrix
def evaluate_model(predicted_col, true_col):
    true_labels = df[true_col]
    predicted_labels = df[predicted_col]

    # Classification report
    report = classification_report(true_labels, predicted_labels, output_dict=True)

    # Confusion matrix
    cm = confusion_matrix(true_labels, predicted_labels, labels=["Positive", "Neutral", "Negative"])

    return report, cm
```

```python
# Function to clean classification report for display
def clean_classification_report(report):
    # Convert report to DataFrame
    report_df = pd.DataFrame(report).transpose()
    # Drop 'support' column and unwanted rows
    report_df = report_df.drop(columns='support', errors='ignore')
    report_df = report_df.drop(index=['macro avg', 'weighted avg'], errors='ignore')
    return report_df

# Evaluate each model
vader_report, vader_cm = evaluate_model('VADER_Predicted', 'Sentiment_Label')
textblob_report, textblob_cm = evaluate_model('TextBlob_Predicted', 'Sentiment_Label')
bert_report, bert_cm = evaluate_model('BERT_Predicted', 'Sentiment_Label')

# Clean and display VADER report
vader_cleaned_report = clean_classification_report(vader_report)
print("\nVADER Classification Report (Cleaned):")
print(vader_cleaned_report)
print("\nVADER Confusion Matrix:")
print(vader_cm)

# Clean and display TextBlob report
textblob_cleaned_report = clean_classification_report(textblob_report)
print("\nTextBlob Classification Report (Cleaned):")
print(textblob_cleaned_report)
print("\nTextBlob Confusion Matrix:")
print(textblob_cm)

# Clean and display BERT report
bert_cleaned_report = clean_classification_report(bert_report)
print("\nBERT Classification Report (Cleaned):")
print(bert_cleaned_report)
print("\nBERT Confusion Matrix:")
print(bert_cm)
```