12-2023

TWITTER POLICING

Hemanth Kumar Medisetty

TWITTER POLICING

A Project

Presented to the

Faculty of

California State University,

San Bernardino

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

by

Hemanth Kumar Medisetty

December 2023

TWITTER POLICING

A Project

Presented to the

Faculty of

California State University,

San Bernardino

---

by

Hemanth Kumar Medisetty

December 2023

Approved by:

Dr. Yan Zhang, Advisor, Computer Science and Engineering

Dr. Qingquan Sun, Committee Member

Dr. Yunfei Hou, Committee Member

ABSTRACT

Police departments are frequently utilizing social media platforms to actively interact with the public. Social media offers an opportunity to share information, facilitate communication, and foster stronger connections between police departments and the communities they serve. In this context sentiment analysis of social media data has become a tool, for identifying sentiments and tracking emerging trends.

This project utilizes sentiment analysis to examine the social media interactions with particular data obtained from the Twitter (X). Initially, the project gathers social media data, from twitter mentioned accounts on Twitter utilizing web scraping techniques. Afterwards, we perform a thorough sentiment analysis using techniques, in Natural Language Processing (NLP). We utilize two reliable sentiment analysis tools, TextBlob and Natural Language Toolkit (NTLK) to classify media posts into three distinct sentiment categories positive, negative, and neutral. Additionally, it monitors the opinions of public sentiments on a monthly basis. The project utilizes data visualization techniques like pie charts, line charts, and clustered column charts to represent sentiment analysis data in appealing ways that highlight the distribution and the trends of public opinion.

ACKNOWLEDGEMENTS

I am writing this message to express my appreciation and thankfulness to Dr. Yan Zhang, who has served as the chair of my committee. Her unwavering support and dedication to excellence throughout my journey have been truly invaluable. I would also like to extend my gratitude to Dr. Qingquan Sun and Dr. Yunfei Hou members of my committee for their belief, in my abilities and their recognition of the contributions I have made to this study. Their confidence in me has been a driving force behind my work.

Furthermore, I would like to take this opportunity to express my gratitude to the faculty at the university who have provided me with guidance and mentorship that has played a crucial role in shaping my academic journey and helping me progress towards achieving my educational goals.

I am particularly grateful for the curriculum offered by the School of Computer Science at California State University, San Bernardino. This curriculum has supported my aspirations and also equipped me with the necessary skills for a successful future.

Throughout my master’s program, I have held onto a saying that education is a treasure that can never be taken away from us. My commitment and determination are evident in this pursuit as it reflects both resilience and an unwavering dedication, towards achieving endeavors.

DEDICATION

With gratitude and utmost admiration, I would like to dedicate my master’s project to Dr. Yan Zhang, my professor. Their guidance, mentorship, and invaluable perspectives have played a role in shaping my journey throughout my this project. I am also profoundly thankful to my parents, whose unwavering affection, backing, and motivation have served as the foundation of my strength during this endeavor. This accomplishment stands as evidence of your belief, in my capabilities. I offer it to you with sincerest appreciation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

In the digital age, social media has become a major platform for communication and engagement between the Police Department and the public. This project mainly focuses on sentiment analysis, specifically Twitter (which is referred to as “X”). Web scraping is used to collect data of a specified mention account from Twitter offering valuable insights into public sentiments, concerns, and trends. Advanced Natural Language Processing (NLP) techniques, with the two tools, TextBlob and Natural Language Toolkit (NLTK), classify social media posts into positive, negative, and neutral sentiments. This classification provides important insights into public sentiment regarding the respective departments. Beyond sentiment analysis, the project keeps track of the evolution of public opinion over time using data visualization techniques like pie charts, line charts, and custom clustered column charts. These visuals effectively convey sentiment distributions and emerging trends on a monthly basis. When transparency and community involvement are crucial, this project uses social media and sentiment analysis to improve communication and cooperation between police departments and the communities they serve.

Purpose

The main purpose of the project is to analyze public sentiments towards the police department on Twitter. This analysis will be done by collecting tweets containing relevant hashtags, keywords, and mentioned accounts related to the police department. But this project works on tweets that are related to the mentioned accounts to the respective police department. The sentiment of each tweet will be analyzed using techniques called TextBlob and NLTK to determine if it is positive, negative, or neutral in sentiment. The findings of this analysis will be used to identify areas where the police department may need to improve public relations and attract public opinion over time. This project will provide valuable insights into how the police department is perceived by the public and help the department improve its image and reputation by addressing the areas of public concern. Sentiment analysis combined with social media monitoring allows you to determine how interested your target audience is in any emerging trends. And their opinions toward the trends in the question [2].

Project Milestones

The project is about sentiment analysis on Twitter data using Python. Sentiment analysis is an important tool to understand public opinions on any current trends. This project shows comparison results of two sentiment analysis methods – TextBlob and NLTK. The process starts with collecting the real data from Twitter and pre-processing the data and ensuring the processed data is suitable to perform analysis. After analysis, the data has been processed to generate a few charts that show how sentiments are changing monthly. The below steps explain each stage.

1. Data Collection and Preprocessing:
    a. At this stage the project includes gathering a set of tweets from the Twitter platform.
    b. Applying data preprocessing methods to tidy up and organize the Twitter data for examination.
    c. Making sure that the data is prepared for sentiment analysis.
2. Sentiment Analysis Implementation:
    a. Currently we are in the stage of developing and implementing sentiment analysis by utilizing the TextBlob and NLTK libraries.
    b. We calculate the sentiment polarity and subjectivity for each tweet using the TextBlob.
    c. Utilizing the NLTK sentiment intensity analyzer to evaluate sentiment based on compound scores.
3. Key Findings and Insights:
    a. Analyzing the sentiment results to identify key findings and insights.
    b. Identifying trends in public opinions and emotional responses over time.
    c. Documenting differences in sentiment classification.
4. Data Visualization:
    a. The main goal of this stage is to create charts that improve our understanding of sentiment data.

   b. Pie charts, line charts, and clustered column charts are used to show the sentiment distribution and how it changes over time. These pictures make it easy for us to see what people are feeling.
   c. By making these visualizations we can quickly see the patterns and trends in people’s emotions. These pictures help those who want to understand how people feel on social media.

5. <u>Documentation And Knowledge Transfer:</u> In this last stage, preparing the document to serve as a valuable resource for researchers, data analysts, and organizations sharing their techniques and knowledge gained from this project for studying and tracking sentiments in various contexts.

CHAPTER TWO

TOOLS FOR DATA COLLECTION, PREPROCESSING, AND ANALYSIS

Libraries, Modules and Software

To complete the project libraries, modules and software have been used. Starting from Visual Studio code, by using Python programming language in VS code with the libraries scweet and selenium the data has been extracted and processed. With the use of libraries - TextBlob and NLTK, sentiments have been classified accordingly.

Visual Studio:

Visual Studio code is also known as VS code which is one of the widely used software as it is an open-source code editor which is known for extensibility, versatility, and developer friendly. In this project version 1.83.1 is used. VS code is a cross-platform which can run on Mac OS, Linux, and Windows. It supports a huge library of extensions that are contributed by the community. It supports various programming languages and file types by giving choices to the developers. It even allows developers to run command line tools and scripts directly within VS code, eliminating the need to switch between a terminal and editor.

Python:

Python is a high-level language, functional, and popular programming language with a readable syntax. Python 3.10.9 is the version used in this project. In Python, the braces-free, intended index which makes code easier to read and reduces errors. It can be used for a variety of tasks such as data analysis, machine learning, web development, and automation. It has a wide range of libraries and frameworks. Python doesn't need to be compiled to be executed because it is an interpreted language. This functionality allows for development and troubleshooting. Python is open source making it accessible to everyone for usage and contribution towards its progress. The comprehensive Python standard library includes functions and modules that cater to an array of tasks eliminating the need, for external dependencies.

Scweet:

Scweet is a robust method, for extracting and examining Twitter data based on specific search parameters. It is a Python library that offers users the capability to scrape types of information from Twitter including tweets, user details, and related data. With scweet users have the flexibility to define their search terms hashtags, user mentions and date ranges to gather tweets. In this project, users can conveniently scrape tweets using the mentioned accounts of users. By providing an interface for configuring search criteria, scweet streamlines the setup process. Additionally, it allows users to specify their timeframes, for data collection enabling analysis during specific periods.

Moreover, scweet supports Twitter data collection making it well suited for research and analysis purposes.

Selenium Webdriver:

Selenium webdriver is a widely used open-source tool that allows automation of web browsers. It provides an interface that enables users to interact with web applications similar to humans. Selenium webdriver is compatible with browsers such as Firefox, Edge, Safari, and Chrome making it possible to run automation tests across platforms. It supports programming languages like Java, JavaScript, Python, Ruby, and C# which makes it versatile for developers and QA engineers. With Selenium webdriver web applications can be tested by emulating user actions like clicking buttons filling forms and navigating through pages. Additionally, its functionality can be enhanced by utilizing third-party libraries and browser-specific drivers. Overall Selenium webdriver plays a role in web-related projects, for developers and individuals involved in quality assurance testing.

TextBlob:

TextBlob, a Python library, is widely used for processing data. It greatly aids in performing natural language processing (NLP) tasks, including sentiment analysis noun phrase extraction, part-of-speech tagging, translation, and more. Through sentiment analysis, TextBlob can determine whether text data holds a neutral sentiment by leveraging its pre-trained sentiment analysis model. One of the benefits of using TextBlob is its ability to identify the part of speech, for each word in a sentence – whether it’s a noun, verb, adjective, or other. When it comes to sentiment analysis with TextBlob it focuses primarily on two aspects: subjectivity and polarity. Polarity measures the tone of the text on a scale from 1 (negative) to +1 (positive) while subjectivity indicates how subjective or objective the text is on a scale from 0 to 1.

Natural Language Toolkit (NLTK):

NLTK is a Python library widely used for language processing and text analysis. When it comes to sentiment analysis with NLTK we rely on the Sentiment Intensity Analyzer module. This module uses a lexicon of words and their associated sentiment scores to calculate the sentiment of a given text. By analyzing the words and their scores, in the lexicon, the Sentiment Intensity Analyzer assigns a polarity score to determine whether the text is positive, negative, or neutral. The classification of sentiment in texts is based on these polarity scores, which help us identify whether a text expresses positivity, negativity, or neutrality.

Pandas:

Pandas is an open source powerful and widely used library module for data manipulation and analysis. Panda provides two main data structures which are Series and Data frames. While a data frame is a two-dimensional table with labeled rows and columns, a Series is effectively a labeled array. These structures make it simple to represent and work with data. Data frames were used in this project. Pandas supports both import and export of data, to and from various file formats like CSV, Excel, SQL database, and JSON. It effortlessly interfaces with frameworks for data visualization like Matplotlib and Seaborn, enabling users to produce helpful visualizations. Matplotlib library has been used in this project. Pandas is incredibly effective and capable of handling huge datasets.

Dataset Information

To perform sentiment analysis on the Dataset which had been collected in real-time. In the code, scrape is a function from the Python library. Scweet which was customized according to the project. By using this function, it scrapes the tweets from the mentioned Twitter account. Below are the following columns in the dataset,

1. UserScreenName: The Twitter user’s screen name who posted the tweet.
2. Username: The user’s full name on Twitter.
3. Timestamp: The timestamp when the tweet was posted.
4. Text: The content of the tweet after cleaning (removing mentions, hashtags, RT indicators, and URLs).
5. Emojis: Emojis present in the tweet (if any).
6. Tweets URL: The URL of the tweet.
7. Image link: The link to any images or media associated with the tweet.

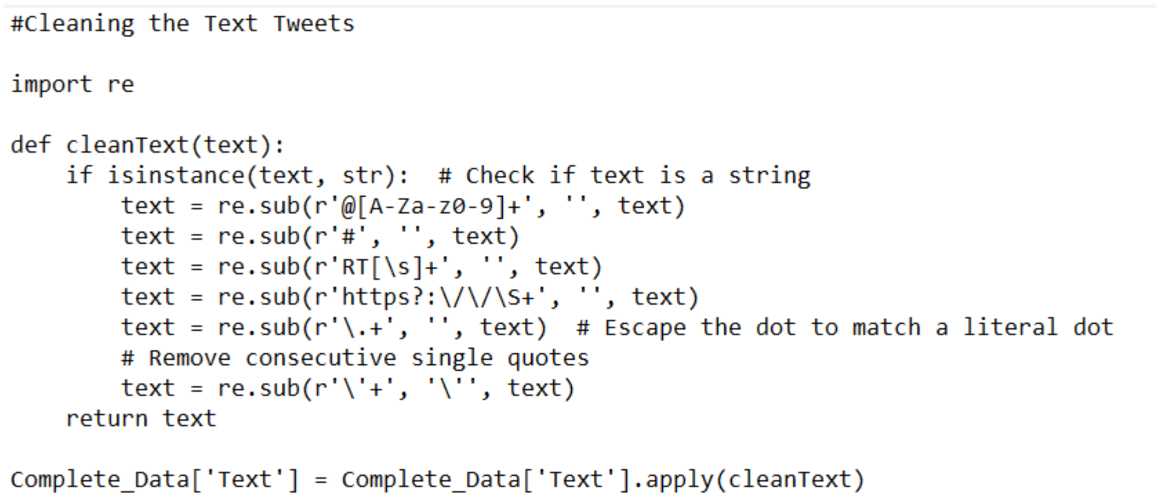
8. Month: The full name of the month in which the tweet was posted extracted from the timestamp,
9. Subjectivity: Subjectivity is a measure of a tweet's text, indicating how opinionated or subjective it is.
10. Polarity: The polarity is a measure of a tweet's text indicating its sentiment (positive, negative, or neutral).
11. TextBlob Sentiment: The sentiment analysis results from TextBlob, classifying the tweet as positive, negative, or neutral based on polarity scores.
12. NLTK_Sentiment scores: A dictionary containing sentiment scores from the NLTK library, including compound, neg, neu, and pos scores.
13. Compound_Value: The compound score is derived from the sentiment score, which is used to classify the tweet as positive, negative, or neutral based on certain thresholds.
14. NLTK_Analysis: The sentiment analysis results from NLTK classifying the tweets as positive, negative, or neutral based on the Compound_Value.

Data Preprocessing

Data Preprocessing is an important part of the code which is required for further analysis and classification of sentiments. All the unwanted content in the text which is not useful for the analysis will get cleaned. Along with the cleaning of text, there is one more important part separating the tweets according to monthly. This process can help when month-based analysis is initiated.

1. Cleaning Text Tweets: In this step, the code cleans unwanted details from the tweet's textual content, making the text more suitable for sentiment analysis. In the code 'cleanText(text)' is a custom function used for cleaning the text tweets. It checks whether the input text is a string or not. Cleaning includes the following operations:
    a. Removing Mentions: Twitter usernames starts with '@'such as '@username' are removed from the text. This helps in removing user references from the tweet.
    b. Removing Hashtags: Hashtags starting with '#' are removed, as they do not contribute to sentiment analysis.
    c. Removing RT indicators: If the tweet is a retweet, it often starts with "RT @username". The code removes this indicator as well.
    d. Removing URLs: Any web links (such as https://example.com) are also removed from the text.
    e. Handling consecutive single quotes: It replaces a single quote for each set of successive single quotes. This is done to standardize text and prevent problems brought on by repetitive quotation marks.

    By removing the text data components that are unimportant for sentiment analysis, this clearing method ensures that the text data is clearer, more focused, and more concisely on tweet's content.



```python
#Cleaning the Text Tweets

import re

def cleanText(text):
    if isinstance(text, str):  # Check if text is a string
        text = re.sub(r'@[A-Za-z0-9]+', '', text)
        text = re.sub(r'#', '', text)
        text = re.sub(r'RT[\s]+', '', text)
        text = re.sub(r'https?:\/\/\S+', '', text)
        text = re.sub(r'\.+', '', text)  # Escape the dot to match a literal dot
        # Remove consecutive single quotes
        text = re.sub(r'\'+', '\'', text)
    return text

Complete_Data['Text'] = Complete_Data['Text'].apply(cleanText)
```

Figure 1: Method to Clean the Text.

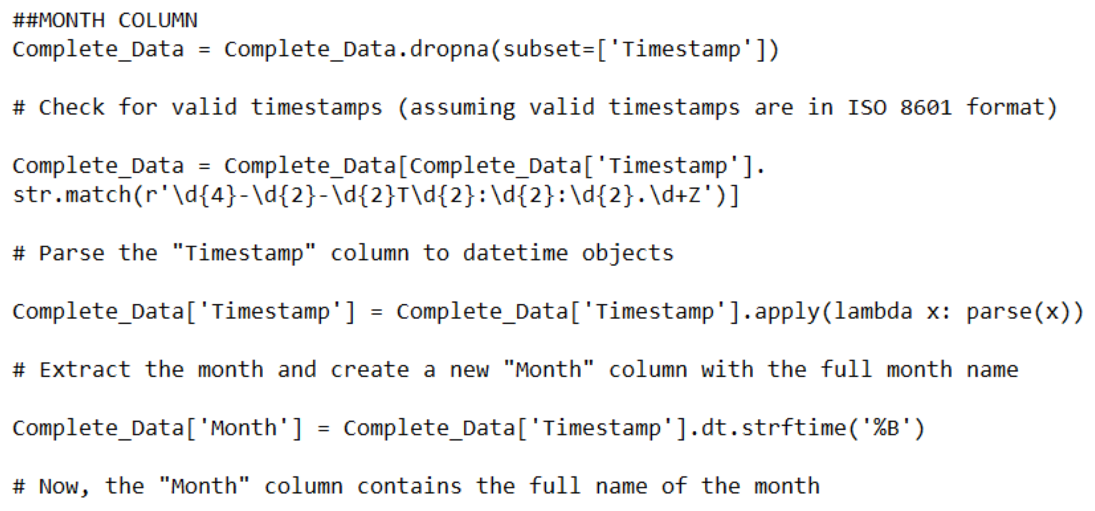
2. Parsing Timestamps: The “Timestamp” column in the data set is processed by the code in this stage, which creates datetime objects from timestamp strings for further analysis. This step is essential because it allows you to segregate all of the tweets by month column which in turn makes it easier and used for time-based analysis and visualization.

   The parsing process involves the following steps:

   a. <u>Check for valid timestamps</u>: In the code ‘dateutil. parser’ library is used to parse these timestamps into strings. The code verifies that timestamp strings in the timestamp match the expected ISO 8601 format. This helps filter out invalid or improperly formatted timestamps.

b. <u>Conversions to datetime objects</u>: Timestamp strings that pass the validation are converted into datetime objects, which can be used for various time-related analysis.

c. The code creates a new column ‘Month’ based on the parsed timestamps. It extracts the full name of the month from the date time objects allowing for further analysis.



```
##MONTH COLUMN
Complete_Data = Complete_Data.dropna(subset=['Timestamp'])

# Check for valid timestamps (assuming valid timestamps are in ISO 8601 format)

Complete_Data = Complete_Data[Complete_Data['Timestamp'].
str.match(r'\d{4}-\d{2}-\d{2}T\d{2}:\d{2}:\d{2}.\d+Z')]

# Parse the "Timestamp" column to datetime objects

Complete_Data['Timestamp'] = Complete_Data['Timestamp'].apply(lambda x: parse(x))

# Extract the month and create a new "Month" column with the full month name

Complete_Data['Month'] = Complete_Data['Timestamp'].dt.strftime('%B')

# Now, the "Month" column contains the full name of the month
```

Figure 2: Create a New Column – Month.

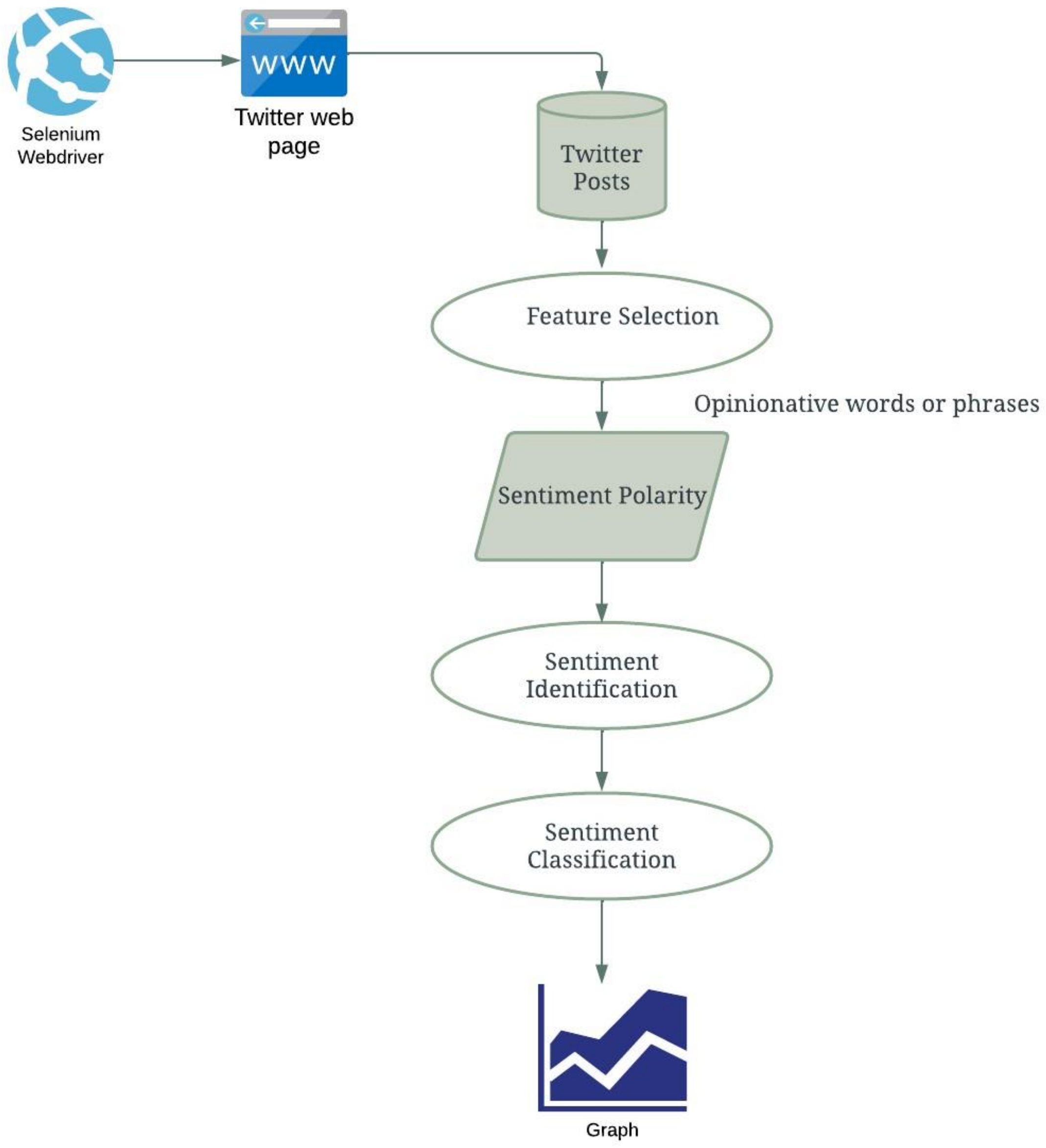


Figure 3: State Diagram.

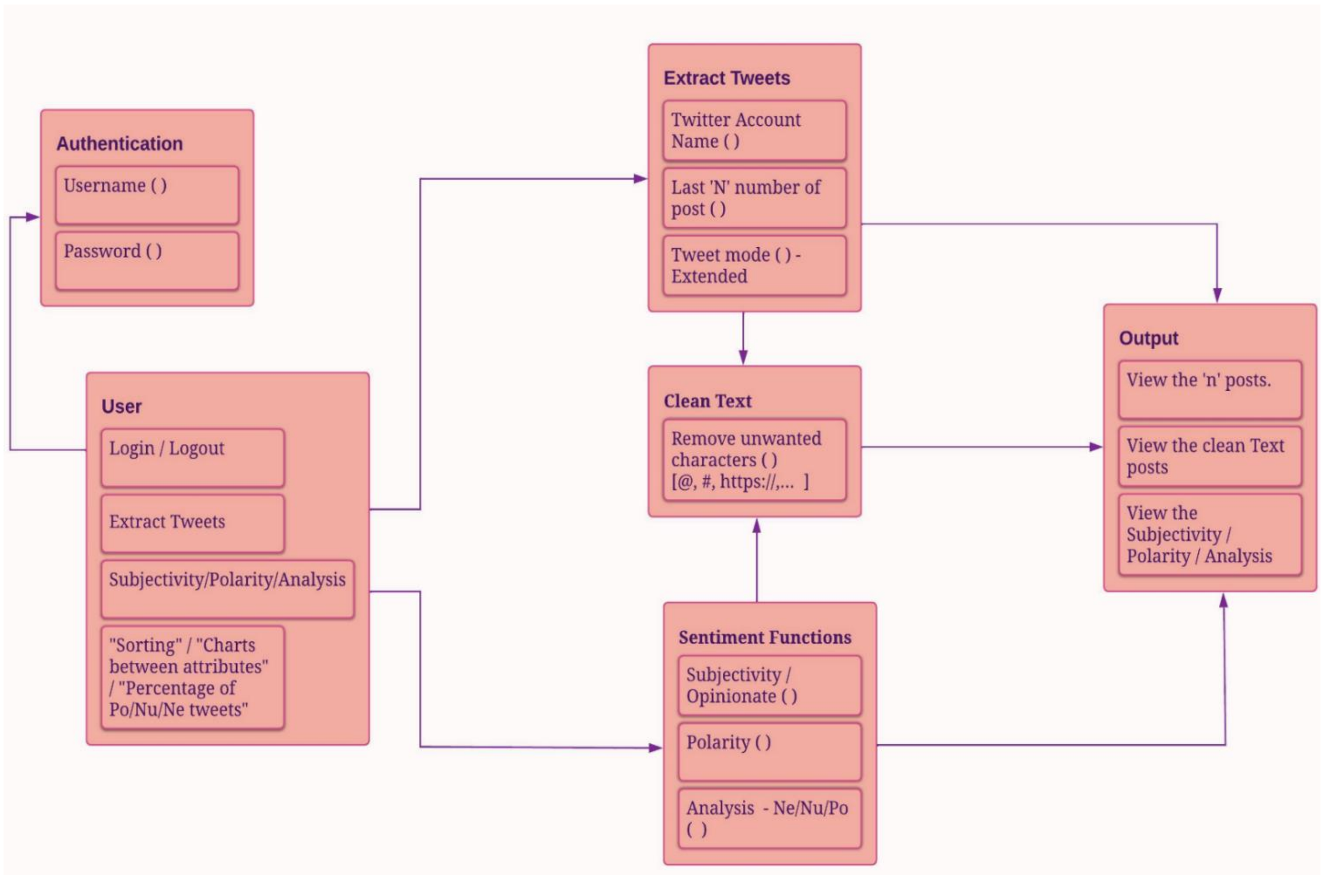

Authentication
Username ( )
Password ( )
User
Login / Logout
Extract Tweets
Subjectivity/Polarity/Analysis
"Sorting" / "Charts between attributes" / "Percentage of Po/Nu/Ne tweets"
Extract Tweets
Twitter Account Name ( )
Last 'N' number of post ( )
Tweet mode ( ) - Extended
Clean Text
Remove unwanted characters ( ) [@, #, https://,...]
Sentiment Functions
Subjectivity / Opinionate ( )
Polarity ( )
Analysis - Ne/Nu/Po ( )
Output
View the 'n' posts.
View the clean Text posts
View the Subjectivity / Polarity / Analysis

Figure 4: Class Diagram.

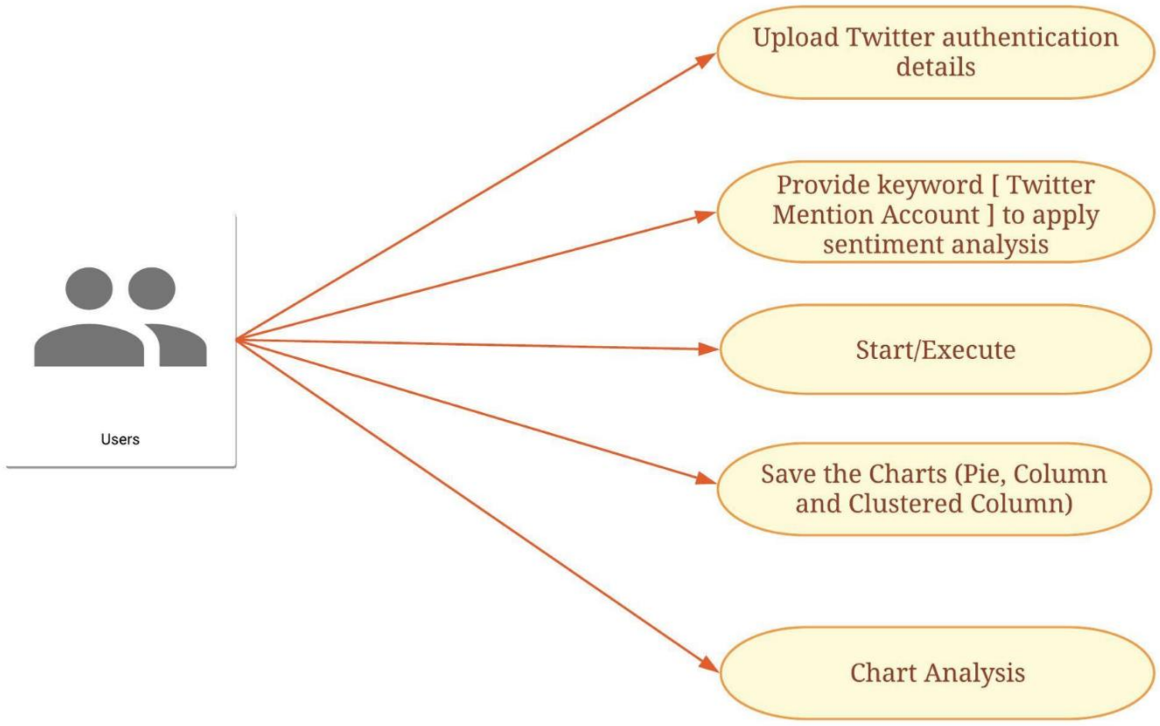

Users
Upload Twitter authentication details
Provide keyword [Twitter Mention Account] to apply sentiment analysis
Start/Execute
Save the Charts (Pie, Column and Clustered Column)
Chart Analysis

Figure 5: Use Case Diagram.

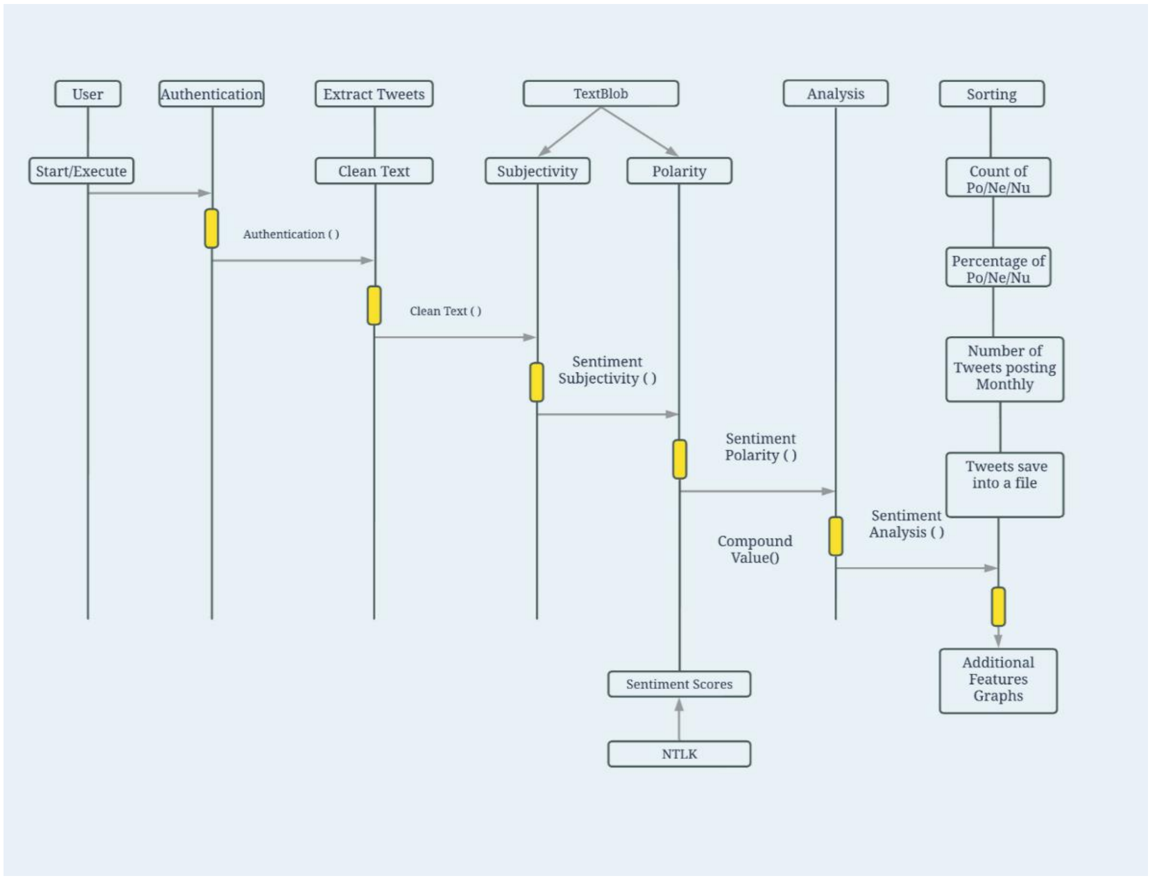


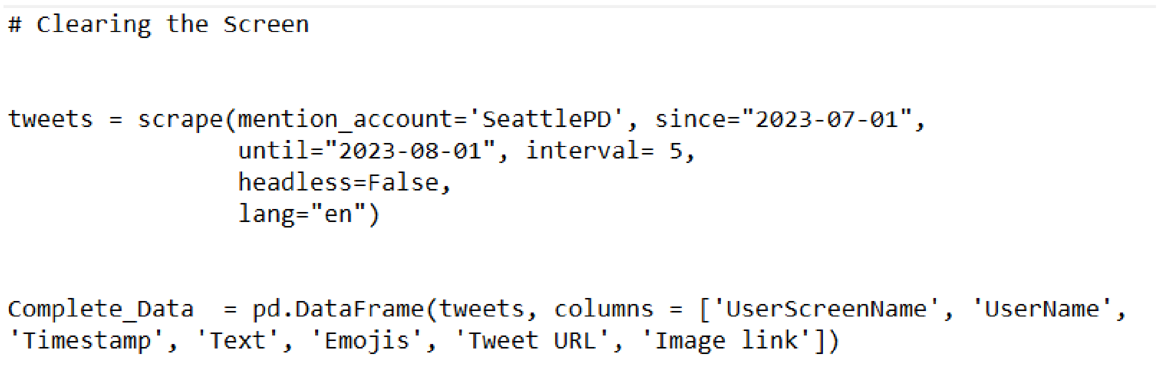
Figure 6: Sequence Diagram.

CHAPTER THREE

METHODOLOGIES

Dataset Collection Using Scrape Method

The scrape method is called from scweet in the main code, and a few methods from utils.py are called in scweet to collect the data. In accordance with the project requirement, these two files have been modified.



```
# Clearing the Screen


tweets = scrape(mention_account='SeattlePD', since="2023-07-01",
                until="2023-08-01", interval= 5,
                headless=False,
                lang="en")


Complete_Data  = pd.DataFrame(tweets, columns = ['UserScreenName', 'UserName',
'Timestamp', 'Text', 'Emojis', 'Tweet URL', 'Image link'])
```

Figure 7: Scrape Method

1. <u>Initialization and Configuration:</u>
   A. The code supports the Chrome browser and uses Selenium for web scrapping.
   B. The required libraries such as Selenium, Pandas, and other modules are imported into the code.

2. <u>Data Extraction:</u>
    A. The ‘scrape’ function is defined in scweet.py for scraping Twitter data.
    B. It takes various parameters like start date, end date, keywords, mention account, hashtags, languages, and more.
    C. A webdriver, in this case Chrome is initiated to access Twitter.
    D. The code, logs into a Twitter account using credentials (username and password) with the help of selenium webdriver.
3. <u>Web Scraping:</u>
    A. The code opens a Twitter search page for a specific query (defined by parameters) and starts scraping.
    B. It continues scrolling through the page until a specific condition is met (for example reaching the tweet limit).
4. <u>Data Extraction:</u>
    A. The ‘get_data’ function is defined in utils.py to extract data from a tweet card.
    B. It collects information such as username, handle, post date, tweet text, emojis, and image links from the tweet card.
    C. Promoted tweets are also handled to exclude them from the data.
5. <u>Data Processing</u>:
    A. Extracted tweet data is checked for uniqueness and added to a list of data.

    B. Images associated with the tweets are optionally saved if specified in the parameters.

6. <u>Data Storage:</u>
    A. Scrapped data is stored in pandas DataFrame for further analysis.
    B. A CSV file is created to save the scrapped data.
7. <u>Scrolling and Pagination:</u> The code keeps scrolling through the Twitter search results to retrieve more tweets.
8. <u>User Interaction</u>: The code includes functions to interact with web elements such as links and XPath to get the required information from the posts.

Text Blob Method Analysis

In accordance with the earlier mention of TextBlob in this document, this section offers an actual illustration of TextBlob's use in this project. This method demonstrates how TextBlob is used for particular tasks in this project and is responsible for conducting sentiment analysis and categorizing tweets into sentiment categories (Positive, Negative or Neutral).

1. <u>getSubjectivity Function</u>: The 'getSubjectivity(text)' function calculates the subjectivity of a given text. The degree of objectivity or subjectivity in the text is referred to as subjectivity.
    A. isinstance (text, str) is used to determine whether this input string, "text" is a string or not.

B. if ‘text’ is a string, the TextBlob library is used to analyze the text and extract the subjectivity score with the help of the function ‘Textblob’s(text).sentiment.subjectivity’

C. If text is not a string (for example missing or non-string data) it returns a default value of 0.0.

2. <u>getPolarity Function:</u>

   A. The polarity of a text is determined using ‘getPolarity(text) function.

   B. The polarity describes the tone of the text, which can be either positive, negative, or neutral.

   C. Similar to ‘getSubjectivity’, it checks whether input ‘text’ is a string.

   D. If ‘text’ is a string, it uses the TextBlob library is used to analyze the text and retrieve the polarity score using ‘TextBlob(text). sentiment. polarity’.

   E. if text is not a string it returns a default value of 0.0

3. <u>Calculating Subjectivity and Polarity:</u>

   A. Using the ‘.apply’ method, the code applies this function to ‘Text’ column of the ‘Complete_Data’ DataFrame.

   B. The polarity scores are kept in the ‘Polarity’ column, but the subjectivity scores are kept in a new column called “Subjectivity”.

4. <u>Sentiment Analysis:</u>

   A. Tweets are categorized into sentiment groups based on their polarity scores using the ‘getAnalysis(value)’ function.

I. if the polarity score < 0 the tweet is categorized ‘Negative’.

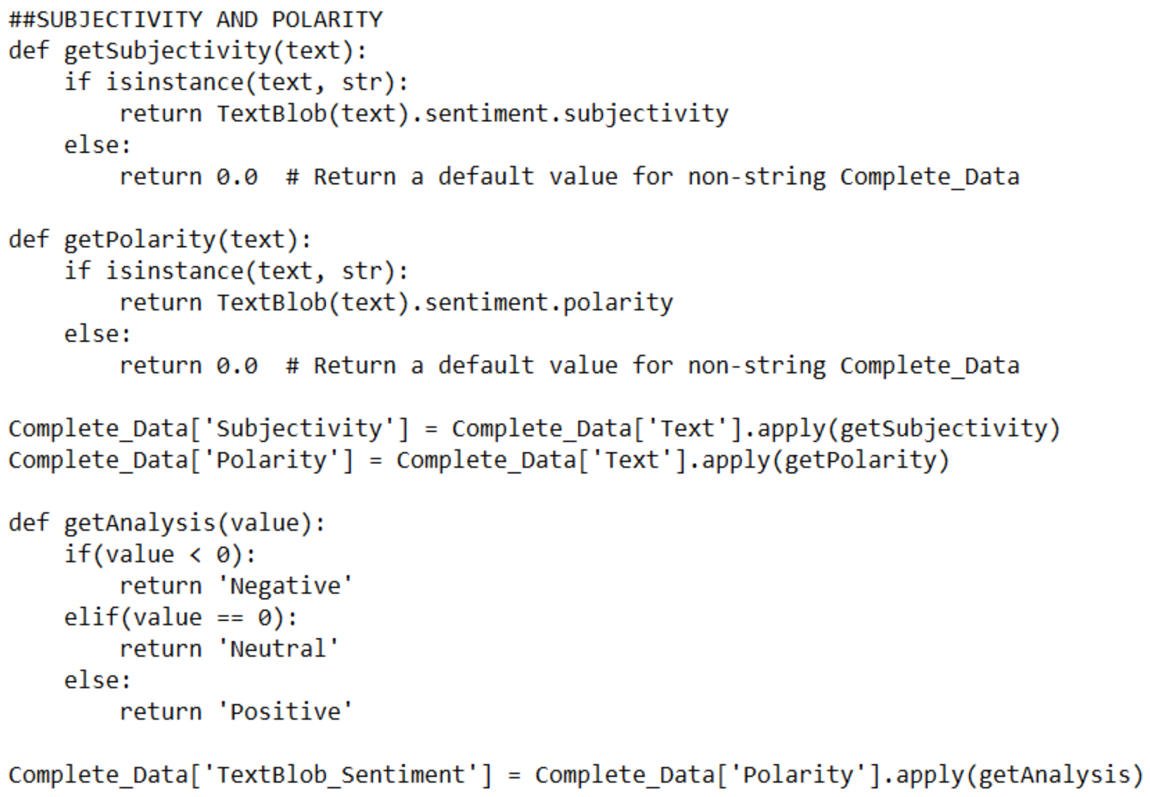
II. if the polarity score = zero the tweet is categorized ‘Neutral’.

III. if the polarity score > 0 the tweet is categorized ‘Positive’.

5. <u>Storing Sentiment Analysis Results:</u>

   A. To save the sentiment labels the code adds a new column called “TextBlob_Sentiment” to “Complete_Data” DataFrame.

   b. It is populated by using ‘getAnalysis’ function on Polarity column.



```
##SUBJECTIVITY AND POLARITY
def getSubjectivity(text):
    if isinstance(text, str):
        return TextBlob(text).sentiment.subjectivity
    else:
        return 0.0  # Return a default value for non-string Complete_Data

def getPolarity(text):
    if isinstance(text, str):
        return TextBlob(text).sentiment.polarity
    else:
        return 0.0  # Return a default value for non-string Complete_Data

Complete_Data['Subjectivity'] = Complete_Data['Text'].apply(getSubjectivity)
Complete_Data['Polarity'] = Complete_Data['Text'].apply(getPolarity)

def getAnalysis(value):
    if(value < 0):
        return 'Negative'
    elif(value == 0):
        return 'Neutral'
    else:
        return 'Positive'

Complete_Data['TextBlob_Sentiment'] = Complete_Data['Polarity'].apply(getAnalysis)
```

Figure 8: TextBlob - Subjectivity, Polarity and Sentiment Classification.

NLTK Method Analysis

This method is used to perform sentiment analysis on text data using the NLTK library and its SentimentIntensityAnalyzer.

1. <u>sentiment_scores Functions:</u>
   - A. The ‘sentiment_scores(text)’ function calculates sentiment scores (compound, negative, neutral, positive) for the input text.
   - B. If ‘text’ is not a string (example: missing or non-string data), it returns a default value of 0.0.
   - C. ‘isinstance (text, str) is used to determine whether the input string, “text”, is a string or not.
   - D. If the input is a string, it proceeds to perform sentiment analysis using the SentimentIntensityAnalyzer.
2. <u>Applying Sentiment Analysis:</u>
   - A. This line applies the sentiment_scores functions to evaluate each text in the Complete_Data Dataframe ‘Text’ column.
   - B. Each text’s sentiment score is determined, and it is stored in a new column called “NLTK_Sentiment_Scores.”
3. <u>Extracting Compound Values:</u>
   - A. ‘get_compound(text)’ This function takes NLTK_Sentiment_Scores as a text as input and it returns the ‘compound’ score from the input dictionary.

B. ‘get_compound function’ applies to each row in NLTK_Sentiment _Scores extract the ‘compound’ score and stores it in a new column called ‘Compound_Value’.

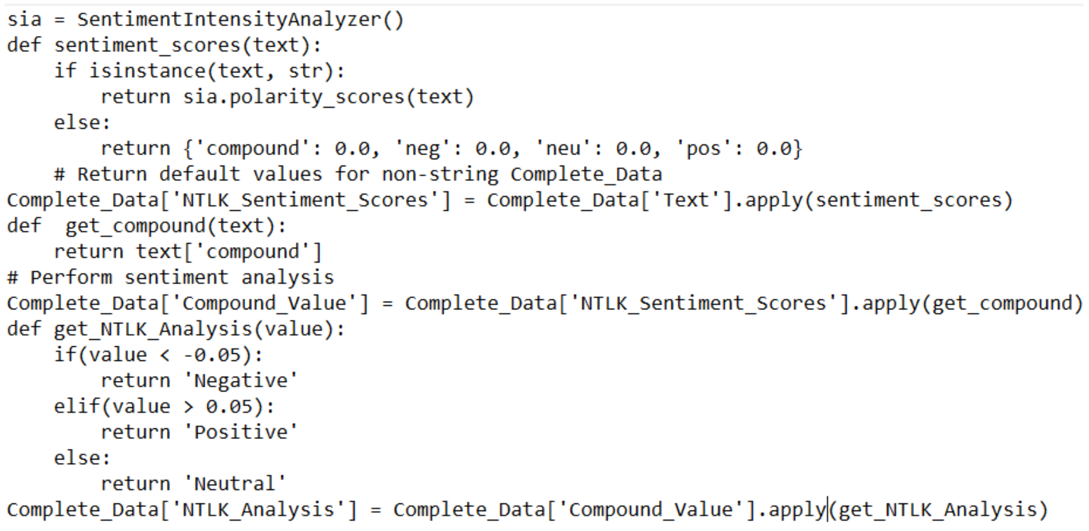
4. Performing Sentiment Analysis: Tweets are categorized into sentiment groups based on their compound scores using the ‘get_NLTK_Analysis(value)’ function.

   I. If the compound value < -0.05 it is categorized as ‘Negative.

   II. If the compound value > 0.05 it is categorized as ‘Positive’.

   III. Otherwise, it is categorized as ‘Neutral’.

5. Storing Sentiment Analysis Results: ‘get_NLTK_Analysis’ function to each ‘Compound_value’ in the DataFrame and save the sentiment labels, the code adds a new column called ‘NLTK_Analysis” to the Complete_Data” DataFrame.



```
sia = SentimentIntensityAnalyzer()
def sentiment_scores(text):
    if isinstance(text, str):
        return sia.polarity_scores(text)
    else:
        return {'compound': 0.0, 'neg': 0.0, 'neu': 0.0, 'pos': 0.0}
    # Return default values for non-string Complete_Data
Complete_Data['NTLK_Sentiment_Scores'] = Complete_Data['Text'].apply(sentiment_scores)
def  get_compound(text):
    return text['compound']
# Perform sentiment analysis
Complete_Data['Compound_Value'] = Complete_Data['NTLK_Sentiment_Scores'].apply(get_compound)
def get_NTLK_Analysis(value):
    if(value < -0.05):
        return 'Negative'
    elif(value > 0.05):
        return 'Positive'
    else:
        return 'Neutral'
Complete_Data['NTLK_Analysis'] = Complete_Data['Compound_Value'].apply(get_NTLK_Analysis)
```

Figure 9: NLTK Sentiment Scores and the Sentiment Classification

CHAPTER FOUR

RESULTS

In contrast, NLTK uses compound values and TextBlob uses polarity values for the determination of sentiment. A notable distinction is the 0.05 threshold, which separates the sentiment categories. This minor change has a big effect on outcomes with NLTK categorizing more negative comments than TextBlob. When comparing the two, NLTK shows a greater tendency than TextBlob to classify feelings as negative. It's crucial to remember that this does not necessarily mean that NLTK is always preferable to TextBlob. The choice between the two, depends on the specific requirement of the user. While NLTK excels in negative sentiment classification. The distribution of positive and neutral attitudes is very constant on TextBlob. In conclusion, the choice of tool depends on the desired results and particular use cases. In this project, sentiment analysis was conducted on the Twitter account of SeattlePD. The results are presented below in tabular form, indicating the counts of positive, negative and neutral sentiments.

Sentiment Analysis SeattlePD		
Sentiment	TextBlob	NLTK
Positive	6073	5183
Negative	4403	7198
Neutral	7507	5602

Table 1: Data on Sentiment Analysis SeattlePD

Result Analysis and Visualization

With the help of visualization, the data can be simplified, and it help one to understand current trends and patterns. In this the analysis shown in three categories with Sentiment Analysis Comparison using Pie Chart which gives numbers and percentages of positive, negative neutral of complete data, secondly, Track of monthly changes using Line Chart which shows individual trends, and Monthly Sentiment Analysis using Clustered column chart which shows all the sentiments and their trends in one place.

<u>Sentiment Analysis Comparison:</u>

The below chart visually compares sentiment analysis results for the Police Department's tweets using text blob and NLTK. It facilitates a quick understanding of the sentiment distribution and allows for insights into public sentiment regarding the Police Department on Twitter. The pie chart shows percentages and number of each sentiment for both the TextBlob and NLTK analysis.

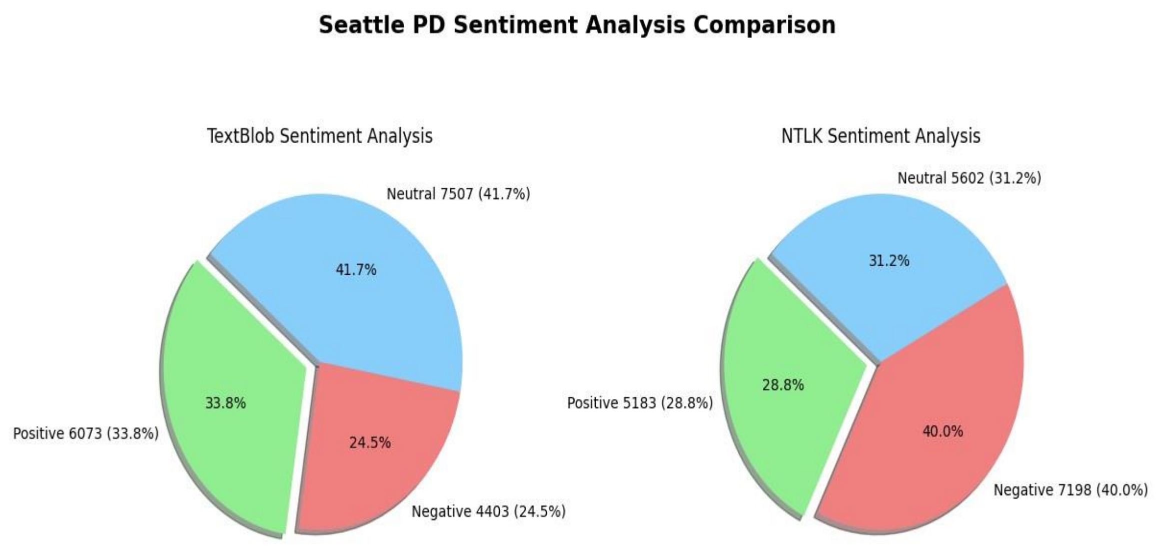


Figure 10: Sentiment Analysis Comparison between TextBlob and NLTK.

<u>Keep Track of Monthly Sentiment Changes:</u>

The following code generates separate line charts to visualize sentiment analysis results for three categories: Positive, Negative and Neutral using TextBlob. It customizes the color palette for each sentiment category and groups the data by month and sentiment which are present on X-axis and Y-axis. The charts provided a clear view of statement trends over time, enabling insights into how public sentiment towards the Police Department on Twitter varies by month and sentiment category. Additionally, the NLTK sentiment analysis uses the same chats with the same function with different variables.

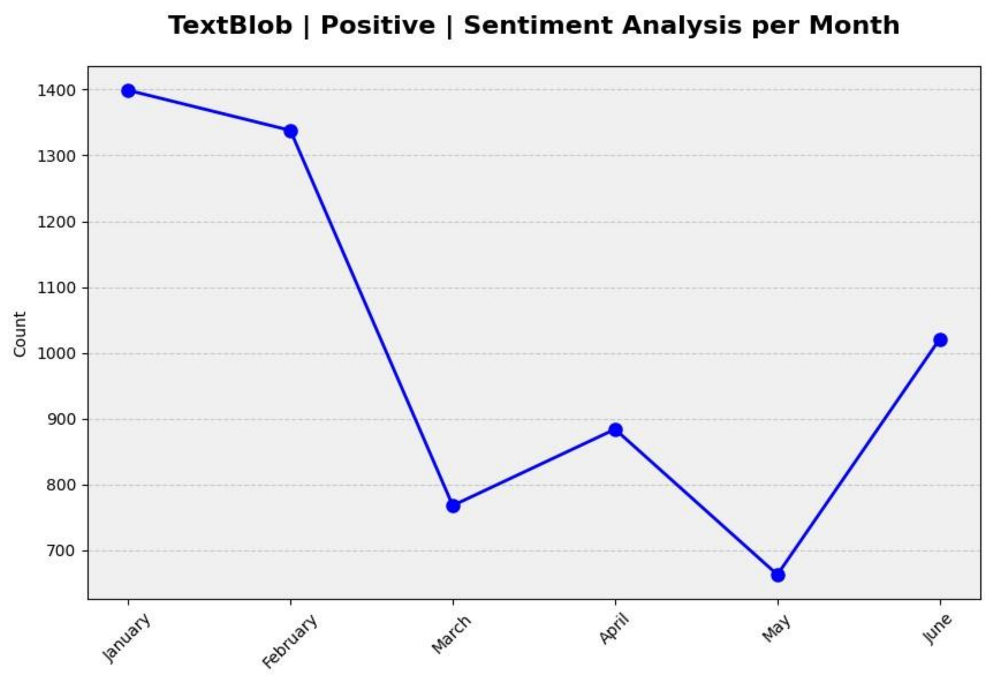


Figure 11: Positive - Monthly Sentiment Analysis using TextBlob.

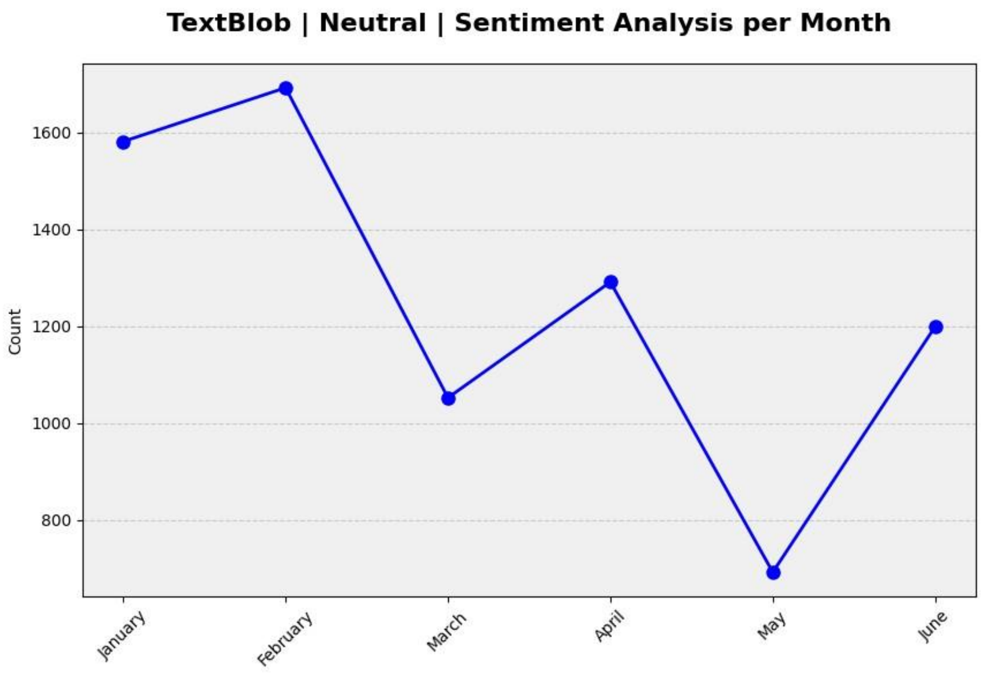


Figure 12: Neutral - Monthly Sentiment Analysis using TextBlob.

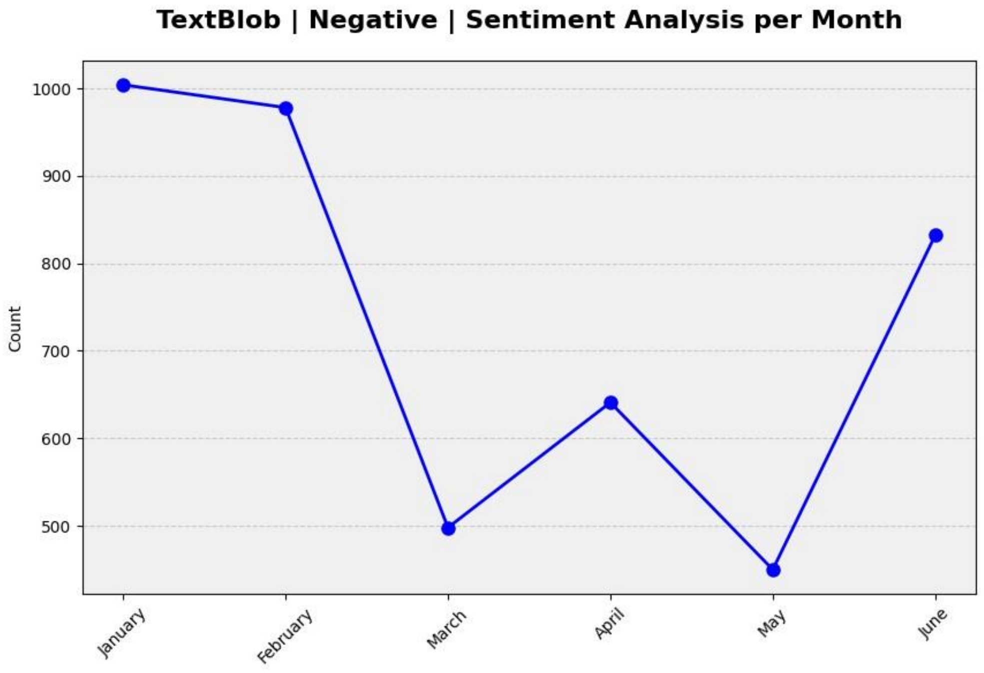


Figure 13: Negative - Monthly Sentiment Analysis using TextBlob.

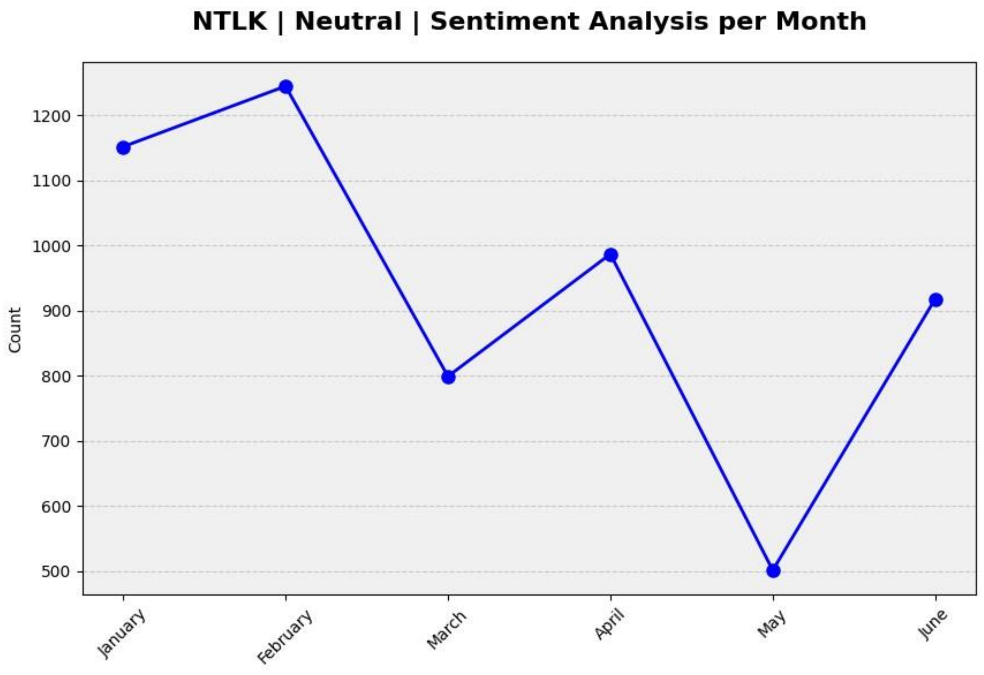


Figure 14: Neutral - Monthly Sentiment Analysis using NLTK .

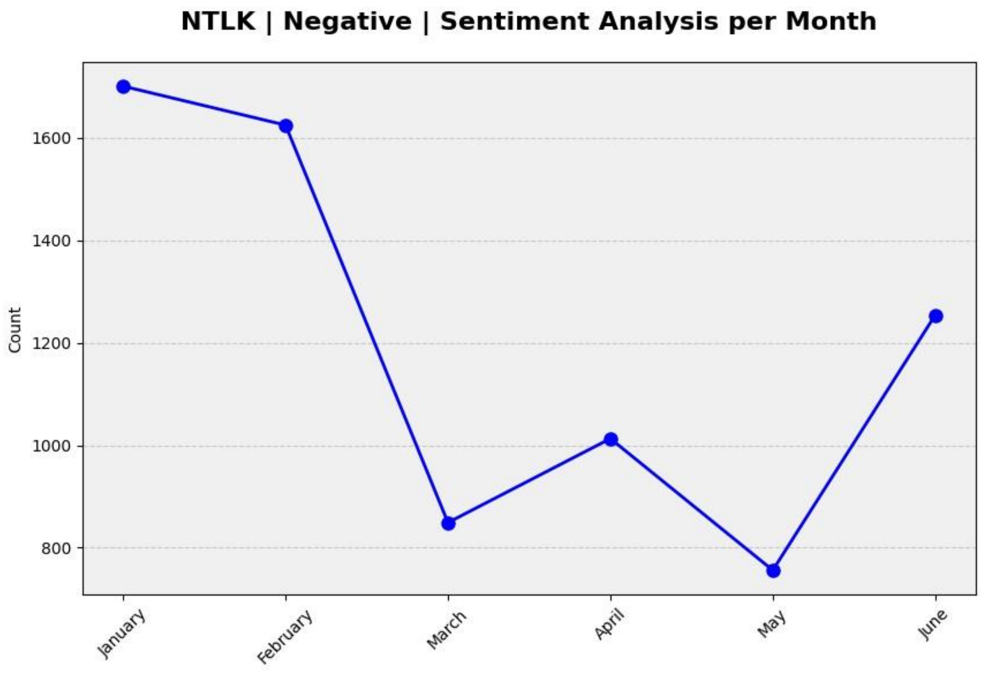


Figure 15: Negative - Monthly Sentiment Analysis using NLTK.

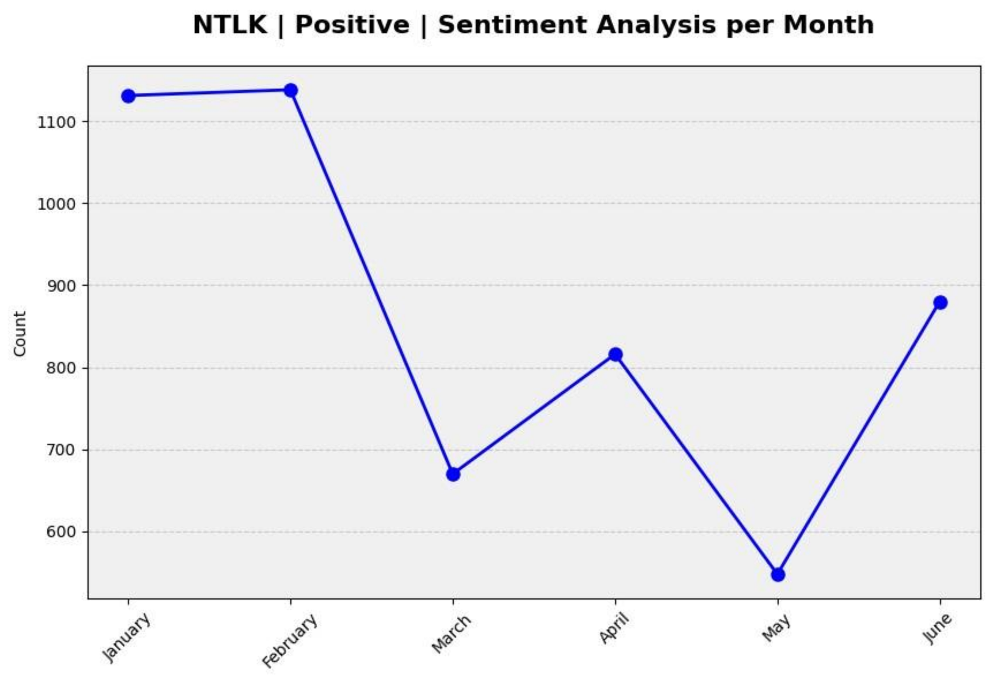


Figure 16: Positive - Monthly Sentiment Analysis using NLTK.

<u>Monthly Sentiment Analysis:</u> The clustered column graph helps to visualize sentiment analysis results using TextBlob across three categories: Positive, Neutral, and Negative. The graph represents sentiment analysis results per month, allowing for easy comparison of sentiment distribution over time. The code ensures that the graph is visually appealing and informative. This visualization helps in understanding how public sentiment towards the Police Department on Twitter varies by sentiment category and month.

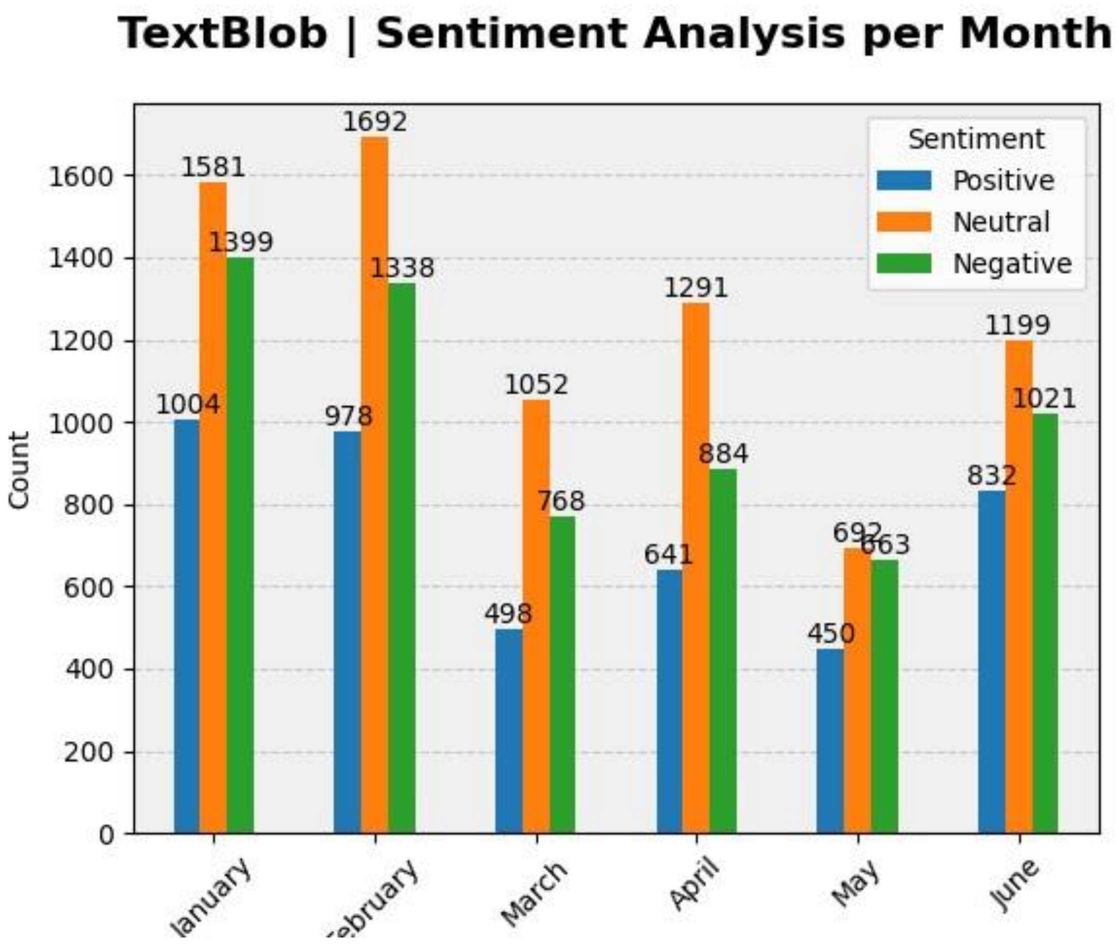


Figure 17: Clustered Column for Monthly Sentiment Analysis using TextBlob.

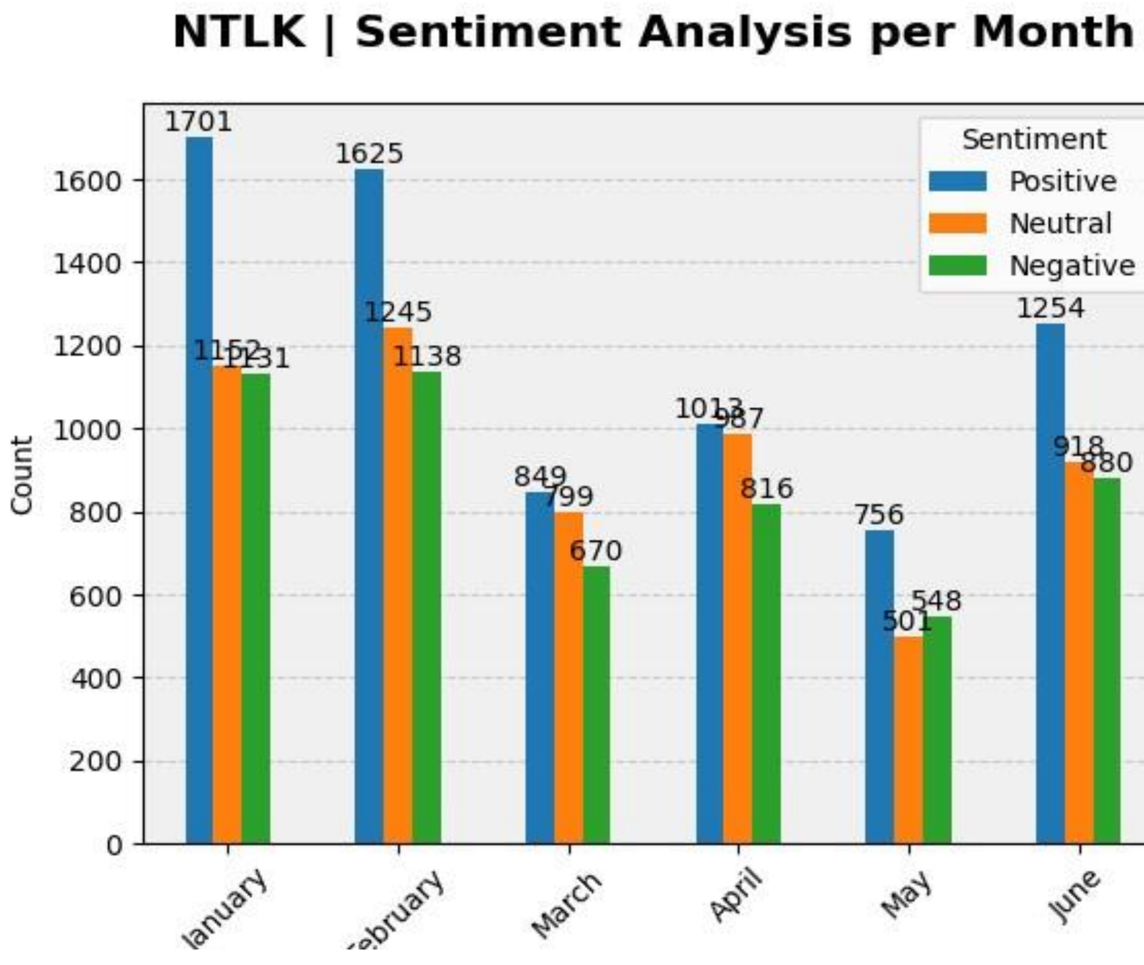


Figure 18: Clustered Column for Monthly Sentiment Analysis using NLTK.

CHAPTER FIVE

CONCLUSION

In conclusion, this study revealed valuable insights of how police and public interact on Twitter showed us important things using sentiment analysis. This project aimed to bring police and communities closer by using the power of natural language processing to understand public sentiments by utilizing two tools TextBlob and Natural Language Toolkit (NTLK), to understand feelings. They showed some differences in how they work. NTLK and TextBlob are tools we used to understand people's feelings on social media, with a certain threshold 0.05, NLTK tends to find more negative comments, while TextBlob shows a more even distribution of positive and neutral sentiments. Neither tool is better overall; it depends on the user specific needs. We looked at the how people feel about this Seattle Police on Twitter. We use charts to show how these feelings change each month this helps us understand what the community thinks. In summary NTL K is good at finding negative feelings while TextBlob is better at keeping a balance between positive and neutral sentiments

Major Contribution

With a Twitter API basic account, for $100 USD per month, users can retrieve up to 10,000 tweets per month. The Twitter API Pro account, on the other hand, allows for a maximum of 1,000,000 tweets and cost $5000 USD per month. In the context of our code, it keeps retrieving data until a server connection is lost. Even though it might take some time to compile a sizable number of tweets, this period of time can handle them by taking the associated cost into account. By executing the code and allowing it to operate for an extended period, a significant volume of data can be collected, and that data can be used for subsequent analysis.

Future Work

In the future, if the budget permits, we can consider using Twitter’s API with a developer account for this project. This involves engaging in discussions through Twitter. We can also make the project more user-friendly by creating front-end interfaces and dashboards. This will help us better connect with our requirements and make the project more accessible and interactive.

APPENDIX A:

KEY PART CODE

The code in the below snippet is used in the main program to collect dataset in real-time. After data collection, the code is used to clean unwanted information from tweets which is not useful for analysis. Subjectivity, Polarity values using TextBlob method and performing sentiment analysis which in return classifying sentiment as well. NLTK, it able to produce sentiment scores, among the scores with help of compound values NLTK sentiment classification will be done.

<u>SOURCE CODE:</u>

```
######################## Libraries ########################
import os
import pandas as pd
from textblob import TextBlob
import matplotlib.pyplot as plt
from datetime import datetime, timedelta #188
import time #55
from dateutil.parser import parse #55
import nltk
from nltk.sentiment import SentimentIntensityAnalyzer
from Scweet.scweet import scrape
```

```
####################### Clearing the Screen #######################
tweets = scrape(mention_account='SeattlePD', since="2023-07-01",
                until="2023-08-01", interval= 5,
                headless=False,
                lang="en")
Complete_Data  = pd.DataFrame(tweets, columns =  ['UserScreenName',
'UserName', 'Timestamp', 'Text', 'Emojis', 'Tweet URL', 'Image link'])


####################### Cleaning the Text Tweets###################
import re
def cleanText(text):
    if isinstance(text, str):  # Check if text is a string
        text = re.sub(r'@[A-Za-z0-9]+', '', text)
        text = re.sub(r'#', '', text)
        text = re.sub(r'RT[\s]+', '', text)
        text = re.sub(r'https?:\/\/\S+', '', text)
        text = re.sub(r'\.+', '', text)  # Escape the dot to match a literal dot
        # Remove consecutive single quotes
        text = re.sub(r'\'+', '\'', text)
    return text
Complete_Data['Text'] = Complete_Data['Text'].apply(cleanText)
```

```
######################## SUBJECTIVITY AND POLARITY ##############

def getSubjectivity(text):

    if isinstance(text, str):

        return TextBlob(text).sentiment.subjectivity

    else:

        return 0.0  # Return a default value for non-string Complete_Data


def getPolarity(text):

    if isinstance(text, str):

        return TextBlob(text).sentiment.polarity

    else:

        return 0.0  # Return a default value for non-string Complete_Data


Complete_Data['Subjectivity'] = Complete_Data['Text'].apply(getSubjectivity)

Complete_Data['Polarity'] = Complete_Data['Text'].apply(getPolarity)


def getAnalysis(value):

    if(value < 0):

        return 'Negative'

    elif(value == 0):

        return 'Neutral'

    else:
        return 'Positive'


Complete_Data['TextBlob_Sentiment'] =

Complete_Data['Polarity'].apply(getAnalysis)


####################### SENTIMENT SCORES USING NLTK ##########

sia = SentimentIntensityAnalyzer()

def sentiment_scores(text):

    if isinstance(text, str):

        return sia.polarity_scores(text)

    else:

        return {'compound': 0.0, 'neg': 0.0, 'neu': 0.0, 'pos': 0.0}

    # Return default values for non-string Complete_Data

Complete_Data['NLTK_Sentiment_Scores'] =

Complete_Data['Text'].apply(sentiment_scores)


def  get_compound(text):

    return text['compound']


# Perform sentiment analysis

Complete_Data['Compound_Value'] =

Complete_Data['NLTK_Sentiment_Scores'].apply(get_compound)
```

```
def get_NLTK_Analysis(value):
   if(value < -0.05):
      return 'Negative'
   elif(value > 0.05):
      return 'Positive'
   else:
      return 'Neutral'

Complete_Data['NLTK_Analysis'] =
Complete_Data['Compound_Value'].apply(get_NLTK_Analysis)

######################## PIE CHARTS ########################
# Define a color map for sentiment categories
colors = {'Positive': 'lightgreen', 'Negative': 'lightcoral', 'Neutral': 'lightskyblue'}
TextBlob_sentiment_counts =
Complete_Data['TextBlob_Sentiment'].value_counts()
NLTK_sentiment_counts = Complete_Data['NLTK_Analysis'].value_counts()
# Sort the data by sentiment categories
TextBlob_sentiment_counts =
TextBlob_sentiment_counts.reindex(index=['Positive', 'Negative', 'Neutral'])
NLTK_sentiment_counts = NLTK_sentiment_counts.reindex(index=['Positive',
'Negative', 'Neutral'])
```

```
# Create a figure with a single pie chart
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 6))

# Labels for the sections of the pie chart
ax1.labels = TextBlob_sentiment_counts.index
ax2.labels = NLTK_sentiment_counts.index

# Values for each section
ax1.sizes = TextBlob_sentiment_counts.values
ax2.sizes = NLTK_sentiment_counts.values

# Define colors for each sentiment category in both charts
ax1_colors = [colors[sentiment] for sentiment in ax1.labels]
ax2_colors = [colors[sentiment] for sentiment in ax2.labels]

# Explode the "Positive" section (you can adjust this for emphasis)
explode = (0.1, 0, 0)

# Calculate the percentages
ax1.percentages = [f'{count} ({count / sum(ax1.sizes) * 100:.1f}%)' for count in
ax1.sizes]
```

```
ax2.percentages = [f'{count} ({count / sum(ax2.sizes) * 100:.1f}%)' for count in
ax2.sizes]

# Create the pie chart with labels, values, colors, and explosion
ax1.pie(ax1.sizes, labels=ax1.labels +" " + ax1.percentages, colors=ax1_colors,
autopct='%1.1f%%', startangle=140, explode=explode, shadow=True)
ax2.pie(ax2.sizes, labels=ax2.labels +" " + ax2.percentages, colors=ax2_colors,
autopct='%1.1f%%', startangle=140, explode=explode, shadow=True)

# Add a title
ax1.set_title('TextBlob Sentiment Analysis')
ax2.set_title('NLTK Sentiment Analysis')

# Set a main title for both subplots
plt.suptitle('Seattle PD Sentiment Analysis Comparison',
fontsize=16,fontweight='bold')

# Display the percentages in a fancy way
plt.gca().set_aspect('equal')  # Equal aspect ratio ensures that pie is drawn as a
circle.
plt.tight_layout()
```

```
# Show the pie chart
plt.show()

###################### TEXT BLOB LINE CHARTS ################
# Now, let's create separate line charts for each sentiment category (Positive,
Negative, Neutral).
# Customize the color palette for the bars
colors = {'Positive': 'blue', 'Neutral': 'orange', 'Negative': 'green'}

# Group the data by month and sentiment
grouped_data = Complete_Data.groupby(['Month',
'TextBlob_Sentiment']).size().unstack(fill_value=0)

# Get the unique months in the data
available_months = Complete_Data['Month'].unique()

# Create a custom order for the available months
custom_order = sorted(available_months, key=lambda x: datetime.strptime(x,
"%B"))

# Create separate line charts for each Text Blob sentiment category
for sentiment in ['Positive', 'Negative', 'Neutral']:
    plt.figure(figsize=(10, 6))

# Filter data for the current sentiment
    sentiment_data = grouped_data.loc[custom_order][sentiment]

# Set custom x-axis ticks and labels
    x_ticks = range(len(custom_order))
    x_labels = custom_order

    ax = sentiment_data.plot(kind='line', color='blue', marker='o', linestyle='-',
linewidth=2, markersize=8)

    # Set custom x-axis ticks and labels
    ax.set_xticks(x_ticks)
    ax.set_xticklabels(x_labels, rotation=45, fontsize=10)

    plt.xlabel('Month')
    plt.ylabel('Count')
    plt.title(f' TextBlob | {sentiment} | Sentiment Analysis per Month')
    ax.set_facecolor('#f0f0f0')
    ax.grid(axis='y', linestyle='--', alpha=0.6)
    ax.set_axisbelow(True)
    plt.title(f'TextBlob | {sentiment} | Sentiment Analysis per Month', fontsize=16,
fontweight='bold', pad=20)
    plt.show()
######################## TEXT BLOB CLUSTERED COLUMN GRAPH ####
# Create a clustered column graph with custom styling
# plt.figure(figsize=(10, 6))
ax = grouped_data.loc[custom_order].plot(kind='bar')

# Set labels and title
plt.xlabel('Month')
plt.ylabel('Count')
plt.title('TextBlob | Sentiment Analysis per Month')

# Customize the legend
plt.legend(title='Sentiment', labels=['Positive', 'Neutral', 'Negative'],
fancybox=True)

# Add count labels to each bar
for p in ax.patches:
    ax.annotate(str(int(p.get_height())), (p.get_x() + p.get_width() / 2.,
p.get_height()), ha='center', va='bottom')
```

```
# Add a shadow effect
ax.set_facecolor('#f0f0f0')

# Add grid lines
ax.grid(axis='y', linestyle='--', alpha=0.6)

# Add a background color to the plot
ax.set_axisbelow(True)
ax.set_facecolor('#f0f0f0')

# Add a title to the graph
plt.title('TextBlob | Sentiment Analysis per Month', fontsize=16, fontweight='bold',
pad=20)

# Customize the tick labels
ax.set_xticklabels(custom_order, rotation=45, fontsize=10)

# Show the plot
plt.show()
```

```
######################## NLTK LINE CHARTS ########################
# Now, let's create separate line charts for each sentiment category (Positive,
Negative, Neutral).
# Customize the color palette for the bars
colors = {'Positive': 'blue', 'Neutral': 'orange', 'Negative': 'green'}
# Group the data by month and sentiment
grouped_data = Complete_Data.groupby(['Month',
'NLTK_Analysis']).size().unstack(fill_value=0)
# Get the unique months in the data
available_months = Complete_Data['Month'].unique()
# Create a custom order for the available months
custom_order = sorted(available_months, key=lambda x: datetime.strptime(x,
"%B"))
# Create separate line charts for each Text Blob sentiment category
for sentiment in ['Positive', 'Negative', 'Neutral']:
    plt.figure(figsize=(10, 6))

    # Filter data for the current sentiment
    sentiment_data = grouped_data.loc[custom_order][sentiment]

    # Set custom x-axis ticks and labels
    x_ticks = range(len(custom_order))
    x_labels = custom_order


    ax = sentiment_data.plot(kind='line', color='blue', marker='o', linestyle='-',
linewidth=2, markersize=8)


    # Set custom x-axis ticks and labels
    ax.set_xticks(x_ticks)
    ax.set_xticklabels(x_labels, rotation=45, fontsize=10)


    plt.xlabel('Month')
    plt.ylabel('Count')
    plt.title(f' NLTK | {sentiment} | Sentiment Analysis per Month')
    ax.set_facecolor('#f0f0f0')
    ax.grid(axis='y', linestyle='--', alpha=0.6)
    ax.set_axisbelow(True)
    plt.title(f'NLTK | {sentiment} | Sentiment Analysis per Month', fontsize=16,
fontweight='bold', pad=20)
    plt.show()
############   NLTK CLUSTERED COLUMN GRAPH
# Create a clustered column graph with custom styling
# plt.figure(figsize=(10, 6))
ax = grouped_data.loc[custom_order].plot(kind='bar')
```

```
# Set labels and title
plt.xlabel('Month')
plt.ylabel('Count')
plt.title('NLTK | Sentiment Analysis per Month')

# Customize the legend
plt.legend(title='Sentiment', labels=['Positive', 'Neutral', 'Negative'],
fancybox=True)

# Add count labels to each bar
for p in ax.patches:
    ax.annotate(str(int(p.get_height())), (p.get_x() + p.get_width() / 2.,
p.get_height()), ha='center', va='bottom')

# Add a shadow effect
ax.set_facecolor('#f0f0f0')

# Add grid lines
ax.grid(axis='y', linestyle='--', alpha=0.6)

# Add a background color to the plot
ax.set_axisbelow(True)
ax.set_facecolor('#f0f0f0')


# Add a title to the graph
plt.title('NLTK | Sentiment Analysis per Month', fontsize=16, fontweight='bold',
pad=20)


# Customize the tick labels
ax.set_xticklabels(custom_order, rotation=45, fontsize=10)


# Show the plot
plt.show()
```

REFERENCES

[1] A. Hernandez-Suarez, G. Sanchez-Perez, K. Toscano-Medina, V. Martinez-Hernandez, V. Sanchez, and H. Perez-Meana, “A web scraping methodology for bypassing Twitter API restrictions,” arXiv.org, Mar. 27, 2018. https://arxiv.org/abs/1803.09875

[2] B. Wagh, J. V. Shinde, and P. Kale, “A Twitter Sentiment Analysis Using NLTK and Machine Learning Techniques,” International Journal of Emerging Research in Management and Technology, vol. 6, no. 12, p. 37, doi: 10.23956/ijermt.v6i12.32.

[3] “Datetime - Basic date and time types," Python Documentation, Available: https://docs.python.org/3/library/datetime.html, [Accessed: September, 2023].

[4] “Dateutil - Date Utilites - Python 3.10.1 documentation," Python Software Foundation, Available: https://docs.python.org/3/library/dateutil.html, [Accessed: September, 2023].

[5] E. Haddi, X. Liu, and Y. Shi, “The role of text pre-processing in sentiment analysis,” Procedia Computer Science, vol. 17, pp. 26–32, Jan. 2013, doi: 10.1016/j.procs.2013.05.005.

[6] I.V. Shravan, "Sentiment Analysis in Python using NLTK," ResearchGate, Available: https://www.researchgate.net/profile/Shravan-Iv/publication/312176414_Sentiment_Analysis_in_Python_using_NLTK/links/5874d26908ae8fce4927e011/Sentiment-Analysis-in-Python-using-NLTK.pdf. [Accessed: June,2023].

[7] J. Yao, “Automated Sentiment Analysis of Text Data with NLTK,” Journal of Physics, vol. 1187, no. 5, p. 052020, Apr. 2019, doi: 10.1088/1742-6596/1187/5/052020.

[8] M. Mahajan, V. Bhakare, A. Bhure, S. Patil, and S. Patil, “Twitter Sentiment Analysis,” International Research Journal of Engineering and Technology (IRJET), vol. 09, issue. 06, Jun. 2022. Available: https://www.irjet.net/archives/V9/i6/IRJET-V9I6234.pdf [Accessed: May,2023].

[9] “Matplotlib: Visualization with Python," Matplotlib, Available: https://matplotlib.org/, [Accessed: September, 2023].

[10] “NLTK Sentiment Analysis,” Available: http://www.nltk.org/howto/sentiment.html, [Accessed: August 2023].

[11] “NLTK Source Code," GitHub, Available: https://github.com/nltk/nltk, [Accessed: August 2023].

[12] “Pandas - Python Data Analysis Library," Pandas, Available: https://pandas.pydata.org/, [Accessed: February, 2023].

[13] S. Elbagir and J. Yang, “Analysis using Natural Language Toolkit and VADER Sentiment,” in International Multiconference of Engineers and Computer Scientists, IMECS 2019, The Royal Garden Hotel, Kowloon, Hong Kong, March 13-15, 2019.

[14] S. Loria, "TextBlob: Simple, Consistent Natural Language Processing," GitHub, Available: https://github.com/sloria/TextBlob, [Accessed: June, 2023].

[15] S. Tiwari, A. Verma, P. Garg and D. Bansal, "Social Media Sentiment Analysis On Twitter Datasets," 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 2020, pp. 925-927, doi: 10.1109/ICACCS48705.2020.9074208.

[16] S. V. Pandey and A. V. Deorankar, "A Study of Sentiment Analysis Task and It's Challenges," 2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT), Coimbatore, India, 2019, pp. 1-5, doi: 10.1109/ICECCT.2019.8869160.

[17] S. Zahoor and R. Rohilla, "Twitter Sentiment Analysis Using Machine Learning Algorithms: A Case Study," 2020 International Conference on Advances

in Computing, Communication & Materials (ICACCM), Dehradun, India, 2020, pp. 194-199, doi: 10.1109/ICACCM50413.2020.9213011.

[18] “Time — Time access and conversions," Python 3.9.7 documentation, Available: https://docs.python.org/3/library/time.html, [Accessed: September, 2023].

[19] V. Prakruthi, D. Sindhu and D. S. Anupama Kumar, "Real Time Sentiment Analysis Of Twitter Posts," 2018 3rd International Conference on Computational Systems and Information Technology for Sustainable Solutions (CSITSS), Bengaluru, India, 2018, pp. 29-34, doi: 10.1109/CSITSS.2018.8768774.

[20] W. Medhat, A. Hassan, and H. Korashy, “Sentiment analysis algorithms and applications: A survey,” Ain Shams Engineering Journal, vol. 5, no. 4, pp. 1093–1113, Dec. 2014, doi: 10.1016/j.asej.2014.04.011.

[21] Y.A. Jeddi, "Scweet - Collect and Analyze Tweets on Twitter," GitHub, Available: https://github.com/Altimis/Scweet, [Accessed: February, 2023].