PYTHON FOR DATABASE

Project 2: Use Case Study - Twitter Data Analysis

Name : Rengasyami sankar Abisekkamthan

M2

Project report

Date: 31/12/2024

Section 1: Setting Up the Environment

This project utilized various Python libraries and tools to facilitate the analysis of Twitter data. The primary development environment was **Jupyter Notebook**, chosen for its user-friendly interface and support for step-by-step analytical workflows. For advanced visualization tasks during the later stages, **Visual Studio Code (VS Code)** was also employed, leveraging its robust features for detailed analysis and presentation.

Python (Version 3.10): Python is the core programming language used throughout the project. Version 3.10 is used to ensure compatibility with libraries such as Pandas, NumPy, and others.

Jupyter Notebook: Used for exploratory data analysis (EDA), testing code snippets, and data processing tasks.

Libraries and Tools Utilized :

To address the diverse requirements of the project, the following Python libraries were employed:

NumPy (import numpy as np)

- **Purpose**: NumPy is a powerful numerical computing library in Python that supports multidimensional arrays and matrices. It also provides a large collection of high-level mathematical functions to operate on these arrays.
- **Use in Project**: Although not directly used in the initial steps of the project, NumPy is foundational for numerical operations. It would be helpful for tasks such as handling large datasets, performing matrix operations, or working with complex numerical analyses.

Pandas (import pandas as pd)

- **Purpose**: Pandas is a data manipulation and analysis library, providing data structures like DataFrames and Series. It is used for handling and analyzing structured data, including the ability to read/write different file formats like CSV, Excel, and SQL databases.
- **Use in Project**: Pandas will be used throughout the project for loading datasets, cleaning, transforming, and analyzing data. The DataFrame object will hold the tweet data, and we will perform operations like sorting, filtering, grouping, and summarizing the data.

Regular Expressions (import re)

- **Purpose**: The re library in Python provides functions for working with regular expressions, allowing us to search, match, and manipulate strings based on specific patterns.
- **Use in Project**: Regular expressions will be used to clean tweet texts by removing URLs, mentions, hashtags, and special characters. This step is crucial to ensure the text is in a consistent and analyzable format.

Natural Language Toolkit (NLTK) (import nltk)

- **Purpose**: NLTK is a comprehensive library for natural language processing (NLP). It offers tools for tokenization, lemmatization, stemming, part-of-speech tagging, and many other NLP tasks.
- **Use in Project**: NLTK can be used for tokenizing the tweet text, removing stopwords, and performing other NLP preprocessing steps. It will be helpful when we want to break down the text into meaningful components or analyze its structure.

SpaCy (import spacy)

- **Purpose**: SpaCy is an advanced NLP library designed for high-performance text processing. It excels at tasks such as named entity recognition (NER), part-of-speech tagging, and dependency parsing.
- **Use in Project**: SpaCy will be used for Named Entity Recognition (NER) to identify important entities like people, organizations, locations, and other named entities in the tweet text. This will help us extract key topics and insights from the tweets.

String (import string)

- **Purpose**: The string module provides a collection of string constants and functions, which help with handling text data more easily, especially with punctuation and whitespace.
- **Use in Project**: This library will be used to handle punctuation marks when cleaning the text, ensuring that only meaningful text is kept. For example, it can help remove punctuation during the preprocessing phase.

TextBlob (from textblob import TextBlob)

- **Purpose**: TextBlob is a library for simple NLP tasks such as sentiment analysis, noun phrase extraction, and translation. It is built on top of NLTK and Pattern.
- **Use in Project**: TextBlob will be used for performing sentiment analysis on tweet text. Based on the polarity score returned by TextBlob, we can categorize the sentiment of each tweet as positive, negative, or neutral.

Matplotlib (import matplotlib.pyplot as plt)

- **Purpose**: Matplotlib is a plotting library used for creating static, animated, and interactive visualizations in Python. It provides a wide variety of chart types, including line charts, bar charts, scatter plots, and histograms.
- **Use in Project**: Matplotlib will be used for generating basic visualizations like histograms and bar charts to understand the distribution of numeric data such as tweet likes, retweets, and sentiment scores.

Seaborn (import seaborn as sns)

- **Purpose**: Seaborn is a higher-level data visualization library based on Matplotlib that provides a more user-friendly interface and attractive themes for statistical plots.

- **Use in Project**: Seaborn will be used for creating more advanced visualizations like heatmaps, boxplots, and pair plots. It will also enhance the aesthetics of the visualizations, making them easier to interpret.

Transformers (from transformers import AutoTokenizer, AutoModelForSequenceClassification, pipeline)

- **Purpose**: The transformers library, developed by Hugging Face, is designed to work with pretrained models for NLP tasks, such as sentiment analysis, text generation, and more. It supports transformer models like BERT, GPT, and others.
- **Use in Project**: We will use this library for advanced NLP tasks like fine-tuned sentiment analysis using transformers like BERT or RoBERTa, which could give more accurate sentiment classifications compared to simpler models like TextBlob.

Torch (import torch)

- **Purpose**: PyTorch is a machine learning library that provides tools for working with deep learning models. It supports tensor computations and automatic differentiation.
- **Use in Project**: Torch will be used to run deep learning models for NLP tasks. It is required for loading and running transformer models from the Hugging Face transformers library, which are built to work with PyTorch.

WordCloud (from wordcloud import WordCloud)

- **Purpose**: WordCloud is a library used for generating word clouds, a type of visualization that displays word frequencies in a visual format.
- **Use in Project**: Word clouds will be used to visually represent the most frequent words in the tweet text, giving a quick overview of the topics discussed.

Pillow (from PIL import Image)

- **Purpose**: Pillow is a Python Imaging Library (PIL) fork that allows opening, manipulating, and saving image files in various formats.
- **Use in Project**: Pillow will be used to load and process images, such as customizing the shape of the word cloud, if needed.

Workflow Optimization :

To streamline the data manipulation workflow, warnings such as the **SettingWithCopyWarning** from Pandas were suppressed, ensuring uninterrupted analysis. This measure enhanced the efficiency of iterative operations without compromising data integrity.

The combination of these tools and libraries enabled a robust environment for the comprehensive analysis of Twitter data, supporting both foundational tasks and advanced insights.

Section 2: Data Collection

The dataset utilized in this project was sourced from **Kaggle**, which provided a CSV file containing pre-scraped Twitter data. This approach simplified the data acquisition process by eliminating the need to fetch tweets directly via Twitter's API, thereby bypassing the constraints of API authentication and rate limiting.

Dataset Management:

To optimize the analysis, all the 10,000 **rows** of the dataset were loaded. This decision was made to reduce memory usage while ensuring a sufficiently large and representative subset of data for analysis. Among the various columns in the dataset, the **"Text" column** was identified as the primary focus of this study, as it contained the raw tweet content.

To ensure compatibility during text preprocessing tasks, the "Text" column was converted to string format. A preliminary review of the dataset confirmed successful loading and the proper structure of the data, validating its readiness for subsequent cleaning and analysis stages.

Rationale for Pre-Collected Data:

The use of pre-collected data from Kaggle allowed the project to concentrate on exploring the content and extracting meaningful insights rather than on data acquisition challenges. This facilitated a seamless transition to the critical stages of data cleaning, preprocessing, and analysis, laying a strong foundation for the overall workflow.

Data overview:

	Tweet_ID	Username	Text	Retweets	Likes	Timestamp
0	1	julie81	Party least receive say or single. Prevent pre...	2	25	2023-01-30 11:00:51
1	2	richardhester	Hotel still Congress may member staff. Media d...	35	29	2023-01-02 22:45:58
2	3	williamsjoseph	Nice be her debate industry that year. Film wh...	51	25	2023-01-18 11:25:19
3	4	danielsmary	Laugh explain situation career occur serious. ...	37	18	2023-04-10 22:06:29
4	5	carlwarren	Involve sense former often approach government...	27	80	2023-01-24 07:12:21

Section 3: Data Cleaning and Preprocessing

Effective data preprocessing is a critical step in any data analysis project, especially when dealing with unstructured text data like tweets. In this project, several preprocessing techniques were employed to clean and structure the raw tweet data, ensuring it was suitable for analysis. The preprocessing steps applied include:

Lowercasing:

Lowercasing is a fundamental text preprocessing technique. It involves converting all characters in the text to lowercase, ensuring that variations like "text," "Text," and "TEXT" are treated as the same word.

This step is particularly useful for tasks such as frequency analysis and TF-IDF vectorization, where consistent casing helps consolidate identical words, leading to more accurate counts or TF-IDF values. By normalizing text, we reduce redundancy and simplify downstream analysis.

However, lowercasing may not always be beneficial. For example:

- In Part-of-Speech (POS) tagging, proper casing can provide crucial hints about nouns, proper nouns, and other grammatical elements.
- For sentiment analysis, uppercase words might indicate emphasis or emotion, such as anger or excitement, and removing this distinction could reduce the quality of insights.

Modern text vectorizers and tokenizers, such as sklearn's TfidfVectorizer and Keras Tokenizer, often perform lowercasing automatically. It's essential to carefully consider whether to disable this behavior based on the specific requirements of our analysis.

In this project, lowercasing will be applied to standardize the tweet text for trend and sentiment analysis, ensuring consistency across all entries while preserving meaningful patterns.

Removal of Punctuations:

Removing punctuation is another essential text preprocessing step. It standardizes the text by eliminating symbols that typically do not carry significant meaning in most analysis contexts. For example, this ensures that "hurray" and "hurray!" are treated as identical, avoiding redundancy in data representation.

However, the list of punctuation marks to remove should be carefully selected based on the project's objectives. For instance, Python's string.punctuation provides a comprehensive list of common punctuation symbols:

!"#$%&\'()*+,-./:;<=>?@[\\]^_{|}~`

In some cases, retaining specific punctuation marks (e.g., hashtags # and mentions @ in Twitter data) might be crucial for extracting trends or user interaction patterns. Conversely, removing all punctuation may simplify text for tasks like frequency analysis or machine learning models.

For our project, we will focus on removing unnecessary punctuation while preserving key symbols such as hashtags and mentions that provide valuable context for sentiment and trend analysis. This tailored approach ensures that the text remains both clean and informative.

Removal of Stopwords:

Stopwords are frequently occurring words in a language such as "the," "a," and "is," that typically do not carry significant meaning for most text analysis tasks. Removing stopwords is a common preprocessing step because they tend to add noise to the data, which can skew results in tasks like frequency analysis or machine learning.

However, in some cases, stopwords may still hold importance. For example, in tasks like **Part-of-Speech (POS) tagging**, these words provide essential clues about the syntactic structure of a sentence and cannot be discarded without losing critical context.

Precompiled stopword lists for various languages are available and can be safely used in most cases. For instance, the **English stopword list** from the nltk package contains common words such as:

['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', ...]

For our project, stopwords will be removed to help us focus on the more meaningful words in tweets, such as hashtags, mentions, and keywords, which provide more insight into trends and sentiments. However, careful consideration will be given to tasks where retaining certain stopwords may be valuable, such as in specific sentiment nuances.

Removal of Frequent Words:

In the previous preprocessing step, we removed stopwords based on general language information. However, when working with a domain-specific corpus (like Twitter data in this case), there may be other words that appear frequently but do not provide significant value for our analysis. These words, although common within the dataset, might not contribute to the insights we're aiming to derive.

This step involves identifying and removing these frequent words to ensure that our analysis focuses on more meaningful terms. In some cases, techniques like **TF-IDF (Term Frequency-Inverse Document Frequency)** can automatically down-weight the impact of frequent words. This is because TF-IDF assigns lower importance to words that appear frequently across many documents, while emphasizing those that are rarer and more informative.

To identify frequent words, we can compute the word frequency distribution and remove those that occur too often in the dataset. For our project, we will first identify the most common words in the tweets and then remove them in the next step to improve the focus of our analysis on relevant keywords, trends, and sentiments.

```
[9]: from collections import Counter
     cnt = Counter()
     for text in df["text_wo_stop"].values:
         for word in text.split():
             cnt[word] += 1
     # Top 10 most frequent words
     cnt.most_common(10)
```

```
[9]: [('Mrs', 346),
      ('American', 346),
      ('tax', 344),
      ('Democrat', 343),
      ('hard', 343),
      ('maybe', 339),
      ('Mr', 338),
      ('I', 337),
      ('Republican', 336),
      ('Congress', 335)]
```

Removal of Rare Words:

The concept of removing rare words is similar to the previous step, but instead of eliminating frequent words, we focus on removing words that appear very infrequently in the dataset.

Rare words may not contribute meaningfully to the analysis, especially when analyzing trends, sentiment, or user interactions in a large corpus like Twitter data. These words may not provide sufficient context or may be too specific to individual tweets, making them less valuable for broader insights.

By removing rare words, we can reduce noise and simplify the dataset, allowing the model or analysis to focus on more common and relevant terms. In this step, we will identify words that occur only a few times across the entire dataset and remove them, ensuring that the analysis centers around more impactful and frequently occurring terms.

For our project, this process helps in cleaning up the dataset, ensuring that we don't waste resources analyzing words that aren't likely to add any valuable insights about trends, hashtags, or sentiment in Twitter data.

Combining Stopwords, Frequent Words, and Rare Words:

To streamline the text preprocessing process, we can combine all the words we want to remove—**stopwords**, **frequent words**, and **rare words**—into a single list. This will allow us to clean the dataset more efficiently by removing all unwanted words in one step, reducing redundancy in the process.

By creating a comprehensive list of words to exclude, we ensure that our analysis is based on meaningful and relevant terms, eliminating noise that could affect our results. This approach is especially useful in handling large datasets, like Twitter data, where there may be an abundance of irrelevant words that do not contribute to sentiment analysis, hashtag trends, or user interactions.

The combined list of words can be used to filter out unnecessary data, leading to a cleaner, more focused dataset that enhances the accuracy of our analysis.

Stemming:

Stemming is a crucial text preprocessing technique in which words are reduced to their base or root form. For instance, words like **"walks"** and **"walking"** are reduced to **"walk"**. While stemming helps in standardizing words, it may sometimes result in non-dictionary terms, as

seen in examples like **"console"** and **"consoling"**, which would both become **"consol"**, a non-standard word.

There are several algorithms for stemming, and one of the most popular is the **Porter Stemmer**, which is widely used for its efficiency. While stemming helps in reducing the complexity of the vocabulary and can improve the performance of models or analysis, it's important to be aware that it may lose some nuances in word meanings.

In our project, we'll use the **NLTK package** to apply stemming to our Twitter dataset. This process will help in reducing variations of words, allowing us to focus on the core concepts in the tweets without being concerned about different word forms.

	Text	text	text_stemmed
0	Party least receive say or single. Prevent pre...	Party least receive say or single. Prevent pre...	parti least receiv say or single. prevent prev...
1	Hotel still Congress may member staff. Media d...	Hotel still Congress may member staff. Media d...	hotel still congress may member staff. media d...
2	Nice be her debate industry that year. Film wh...	Nice be her debate industry that year. Film wh...	nice be her debat industri that year. film whe...
3	Laugh explain situation career occur serious. ...	Laugh explain situation career occur serious. ...	laugh explain situat career occur serious. fiv...
4	Involve sense former often approach government...	Involve sense former often approach government...	involv sens former often approach government. ...

Also this porter stemmer is for English language. If we are working with other languages, we can use snowball stemmer. The supported languages for snowball stemmer are 'danish', 'dutch', 'english', 'finnish', 'french', 'german', 'hungarian', 'italian', 'norwegian', 'porter','portuguese', 'romanian', 'russian', 'spanish', 'swedish'

Lemmatization:

Lemmatization is a crucial text preprocessing step that reduces inflected words to their base or root form, known as the **lemma**. Unlike stemming, which simply chops off word suffixes, lemmatization ensures that the root word is a valid word in the language. For example, the word **"better"** is lemmatized to **"good"**, which is a valid word in English.

Lemmatization generally produces more linguistically accurate results compared to stemming. However, it tends to be slower, as it involves checking the word's dictionary form and considering the word's context to ensure it makes sense. Because of this, the choice between stemming and lemmatization depends on the specific requirements of the task, such as whether linguistic correctness or processing speed is more important.

In our Twitter data analysis project, we will leverage **spaCy**, a fast and efficient NLP library, to perform lemmatization. spaCy not only provides accurate lemmatization but also handles tokenization and part-of-speech tagging automatically. This allows us to accurately lemmatize words like **"running"**, **"ran"**, and **"runner"** into **"run"**, while ensuring proper context is maintained.

Using **spaCy's** lemmatization will help us standardize variations of words and maintain the meaning of the text, which is especially important in our project where understanding the context and sentiment of tweets is crucial for accurate analysis.

	Text	text	text_stemmed	text_lemmatized
0	Party least receive say or single. Prevent pre...	Party least receive say or single. Prevent pre...	parti least receiv say or single. prevent prev...	Party least receive say or single . prevent pr...
1	Hotel still Congress may member staff. Media d...	Hotel still Congress may member staff. Media d...	hotel still congress may member staff. media d...	Hotel still Congress may member staff . medium...
2	Nice be her debate industry that year. Film wh...	Nice be her debate industry that year. Film wh...	nice be her debat industri that year. film whe...	nice be her debate industry that year . film w...
3	Laugh explain situation career occur serious. ...	Laugh explain situation career occur serious. ...	laugh explain situat career occur serious. fiv...	Laugh explain situation career occur serious ....
4	Involve sense former often approach government...	Involve sense former often approach government...	involv sens former often approach government. ...	involve sense former often approach government...

Removal of Emojis:

Emojis have become a significant part of communication on social media platforms, conveying emotions, reactions, or additional context to the text. While emojis can provide valuable sentiment insights in some cases, they might not be relevant or required for certain types of text analysis, such as topic modeling or keyword extraction. Therefore, it is often necessary to remove emojis from the text.

In our Twitter data analysis project, we may choose to remove emojis to focus solely on the text for tasks like sentiment analysis, where the words themselves will be more important than the symbols.

Here's a helper function that efficiently removes emojis from the text. It uses a regular expression to identify and exclude emojis from the tweet text, ensuring that we analyze only the textual content without distraction.

By using this function, we ensure that any emojis are removed from the text, which can then be used for deeper textual analysis without the interference of non-alphabetic symbols. This step is especially useful when preparing data for tasks like sentiment analysis, where the focus is on the actual words used in the tweets.)

Removal of Emoticons :

In the previous step, we removed emojis from the text data. However, emojis are different from emoticons, and we need to treat them separately.

What's the Difference?

- **Emoticons**: These are created using keyboard characters to represent facial expressions. For example, :-) represents a smiley face.
- **Emojis**: These are graphic symbols that depict various expressions, objects, and symbols. For example, 😀 is an emoji, which is different from an emoticon.

While emojis are often graphic symbols, emoticons are text-based symbols commonly used in online communication to express emotions.

Why Remove Emoticons?

Just like emojis, emoticons may not always contribute to meaningful sentiment analysis, topic modeling, or other types of textual analysis. However, the decision to remove them depends on the context of the analysis. For some tasks like **sentiment analysis**, emoticons might carry valuable emotional context, so consider whether to remove them based on your project's objective.

Removal of Emoticons:

To remove emoticons, we can utilize a predefined list of emoticons and remove them from the text. We owe thanks to NeelShah for providing an excellent collection of emoticons, which we will use to clean our text data.

Just like emojis, the removal of emoticons depends on the use case and the nature of the analysis you are performing. It's important to carefully choose preprocessing steps based on your goals for the analysis.

emoticon data from github Neel shah

```
# Thanks : https://github.com/NeelShah18/emot/blob/master/emot/emo_unicode.py
EMOTICONS = {
    u":-\)":"Happy face or smiley",
    u":\)":"Happy face or smiley",
    u":-\]":"Happy face or smiley",
    u":\]":"Happy face or smiley",
    u":-3":"Happy face smiley",
    u":3":"Happy face smiley",
    u":->":"Happy face smiley",
    u":>":"Happy face smiley",
    u"8-\)":"Happy face smiley",
    u":o\)":"Happy face smiley",
    u":-\}":"Happy face smiley",
    u":\}":"Happy face smiley",
    u":-\)":"Happy face smiley",
    u":c\)":"Happy face smiley",
    u":\^\)":"Happy face smiley",
    u"=\]":"Happy face smiley",
    u"=\)":"Happy face smiley",……}
```

Conversion of Emoticon to Words:

In scenarios such as sentiment analysis, removing emoticons might not be ideal since they carry valuable information. Instead of discarding them, we can convert emoticons into their corresponding word formats, preserving their semantic value for downstream modeling processes. Thanks to Neel's excellent dictionary from the previous step, we can reuse it for this conversion.

Removal of URLs:

Next preprocessing step is to remove any URLs present in the data. For example, if we are doing a twitter analysis, then there is a good chance that the tweet will have some URL in it. Probably we might need to remove them for our further analysis.

Removal of HTML Tags:

One another common preprocessing technique that will come handy in multiple places is removal of html tags. This is especially useful, if we scrap the data from different websites. We might end up having html strings as part of our text.

Chat Words Conversion:

This is an important text preprocessing step if we are dealing with chat data. People do use a lot of abbreviated words in chat and so it might be helpful to expand those words for our analysis purposes.

Got a good list of chat slang words from this repo. We can use this for our conversion here. We can add more words to this list.

Spelling Correction:

Spelling correction is an essential text preprocessing step, especially in social media data like Twitter, where informal language, abbreviations, and typos are frequent. These spelling mistakes, if left uncorrected, can impact the quality of text analysis and model predictions.

In our project, we will apply spelling correction to address any typos or inconsistencies in the dataset before performing further analysis, such as sentiment analysis.

For this project, we will use the pyspellchecker library to perform spelling correction on the text data. This library efficiently detects and corrects misspelled words by leveraging a built-in dictionary.

Final data overview after section 3:

	Text	text	text_stemmed	text_lemmatized	Cleaned_Text	Cleaned_Text_No_Emoticons	Text_converted	Cleaned_Text_No_URLs	Cleaned_Text_No_HTML
0	Party least receive say or single. Prevent pre...	Party least receive say or single. Prevent pre...	parti least receiv say or single. prevent prev...	Party least receive say or single . prevent pr...	Party least receive say or single. Prevent pre...	Party least receive say or single. Prevent pre...	Party least receive say or single. Prevent pre...	Party least receive say or single. Prevent pre...	Party least receive say or single. Prevent pre...
1	Hotel still Congress may member staff. Media d...	Hotel still Congress may member staff. Media d...	hotel still congress may member staff. media d...	Hotel still Congress may member staff . medium...	Hotel still Congress may member staff. Media d...	Hotel still Congress may member staff. Media d...	Hotel still Congress may member staff. Media d...	Hotel still Congress may member staff. Media d...	Hotel still Congress may member staff. Media d...
2	Nice be her debate industry that year. Film wh...	Nice be her debate industry that year. Film wh...	nice be her debat industri that year. film whe...	nice be her debate industry that year . film w...	Nice be her debate industry that year. Film wh...	Nice be her debate industry that year. Film wh...	Nice be her debate industry that year. Film wh...	Nice be her debate industry that year. Film wh...	Nice be her debate industry that year. Film wh...
3	Laugh explain situation career occur serious. ...	Laugh explain situation career occur serious. ...	laugh explain situat career occur serious. fiv...	Laugh explain situation career occur serious	Laugh explain situation career occur serious. ...	Laugh explain situation career occur serious. ...	Laugh explain situation career occur serious. ...	Laugh explain situation career occur serious. ...	Laugh explain situation career occur serious. ...
4	Involve sense former often approach government...	Involve sense former often approach government...	involv sens former often approach government. ...	involve sense former often approach government...	Involve sense former often approach government...	Involve sense former often approach government...	Involve sense former often approach government...	Involve sense former often approach government...	Involve sense former often approach government...

natized	Cleaned_Text	Cleaned_Text_No_Emoticons	Text_converted	Cleaned_Text_No_URLs	Cleaned_Text_No_HTML	Cleaned_Text_No_ChatWords	corrected_text_spellchecker
rty least e say or prevent pr...	Party least receive say or single. Prevent pre...	Party least receive say or single. Prevent pre...	Party least receive say or single. Prevent pre...	Party least receive say or single. Prevent pre...	Party least receive say or single. Prevent pre...	Party least receive say or single. Prevent pre...	Party least receive say or single Prevent prev...
otel still ess may er staff . edium...	Hotel still Congress may member staff. Media d...	Hotel still Congress may member staff. Media d...	Hotel still Congress may member staff. Media d...	Hotel still Congress may member staff. Media d...	Hotel still Congress may member staff. Media d...	Hotel still Congress may member staff. Media d...	Hotel still Congress may member staff Media dr...
e be her industry ear . film w...	Nice be her debate industry that year. Film wh...	Nice be her debate industry that year. Film wh...	Nice be her debate industry that year. Film wh...	Nice be her debate industry that year. Film wh...	Nice be her debate industry that year. Film wh...	Nice be her debate industry that year. Film wh...	Nice be her debate industry that years Film wh...
explain n career erious	Laugh explain situation career occur serious. ...	Laugh explain situation career occur serious. ...	Laugh explain situation career occur serious. ...	Laugh explain situation career occur serious. ...	Laugh explain situation career occur serious. ...	Laugh explain situation career occur serious. ...	Laugh explain situation career occur serious F...
ve sense er often pproach nment...	Involve sense former often approach government...	Involve sense former often approach government...	Involve sense former often approach government...	Involve sense former often approach government...	Involve sense former often approach government...	Involve sense former often approach government...	Involve sense former often approach government...

Final Application of all preprocessing and finalising single pre-processed text:

Code:

```
# Final Preprocessing pipeline
def preprocess_text(text):
    text = remove_emoji(text)
    text = convert_emoticons(text)
    text = remove_html(text)
    text = remove_urls(text)
    text = chat_words_conversion(text)
    text = correct_spellings_spellchecker(text)
    return text

# Apply preprocessing pipeline to the 'Cleaned_text' column
df1 = pd.DataFrame()
df1['final_preprocessed_text'] = df['Cleaned_Text'].apply(preprocess_text)

# Save the cleaned data to a new DataFrame and CSV file
df1.to_csv('/content/final_preprocessed_text.csv', index=False)

# Optionally, display the first few rows of the final preprocessed data
df1.head()
```

Dataset overview:

	final_preprocessed_text
0	Party least receive say or single Prevent prev...
1	Hotel still Congress may member staff Media dr...
2	Nice be her debate industry that years Film wh...
3	Laugh explain situation career occur serious F...
4	Involve sense former often approach government...

Section 4: Exploratory Data Analysis

We initialise the new instance of the original dataset and repalce the text column with our new preprocessed text data:

Code:

```
data = pd.read_csv("/kaggle/input/twitter-dataset/twitter_dataset.csv")

data.head()

# Replace the 'Text' column in the original DataFrame with the processed text from df1

data['Text'] = df1['final_preprocessed_text']

# Optionally, save the updated DataFrame to a new CSV file

data.to_csv('updated_data.csv', index=False)

# Display the first few rows of the updated DataFrame

data.head()
```

Data Overview:

	Tweet_ID	Username	Text	Retweets	Likes	Timestamp
0	1	julie81	Party least receive say or single Prevent prev...	2	25	2023-01-30 11:00:51
1	2	richardhester	Hotel still Congress may member staff Media dr...	35	29	2023-01-02 22:45:58
2	3	williamsjoseph	Nice be her debate industry that years Film wh...	51	25	2023-01-18 11:25:19
3	4	danielsmary	Laugh explain situation career occur serious F...	37	18	2023-04-10 22:06:29
4	5	carlwarren	Involve sense former often approach government...	27	80	2023-01-24 07:12:21

Basic exploratory command and the results:

```
data.shape
```

```
(10000, 6)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   Tweet_ID   10000 non-null  int64 
 1   Username   10000 non-null  object
 2   Text       10000 non-null  object
 3   Retweets   10000 non-null  int64 
 4   Likes      10000 non-null  int64 
 5   Timestamp  10000 non-null  object
dtypes: int64(3), object(3)
memory usage: 468.9+ KB
```

```
df1.describe()
```

	final_preprocessed_text
count	10000
unique	10000
top	Body onto understand team about product beauti...
freq	1

Tokenize Tweet Text: Analysis and Insights:

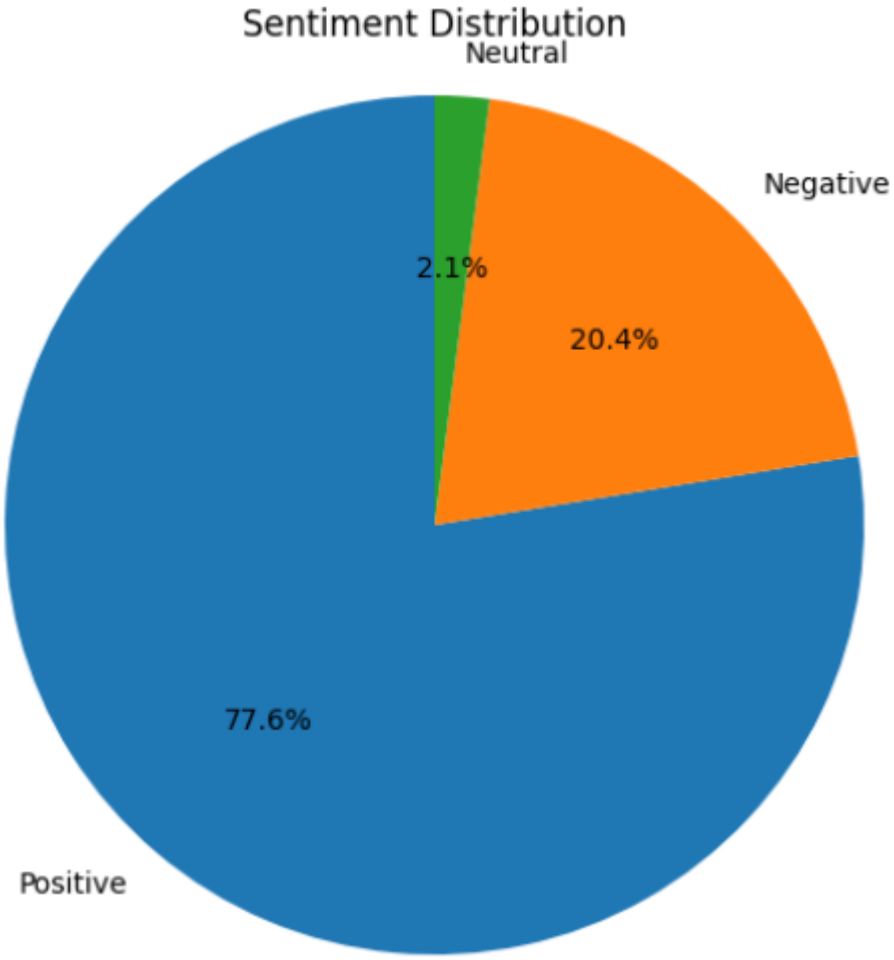
The text preprocessing of tweets involved tokenization, stopword removal, and stemming to refine the data for meaningful analysis. Tokenization, implemented using **NLTK's** word_tokenize() function, divided the content of the **Text** column into individual words (tokens). This step ensured that each tweet's text was represented as a list of words for further processing. Subsequently, common English stopwords were filtered out using **NLTK's** stopword list, retaining only meaningful words in the dataset. Finally, stemming was performed using the **Porter Stemmer**, reducing words to their root forms (e.g., "running" → "run", "played" → "play"), which helped standardize variations of words for consistency in analysis.

In addition to text preprocessing, key summary statistics were calculated to understand user engagement with tweets. The average number of retweets was **49.72**, indicating that, on average, tweets in the dataset were moderately shared among users. The median number of likes was **50.0**, suggesting that half of the tweets received 50 likes or fewer, pointing to a distribution that may include some highly liked outliers skewing the mean.

The correlation between retweets and likes was computed as **0.0128**, revealing a very weak positive relationship. This suggests that while there might be a slight tendency for tweets with higher retweets to also receive more likes, the association is negligible. This insight underscores that other factors, such as tweet content, timing, or user base, might play a more significant role in determining likes and retweets independently.

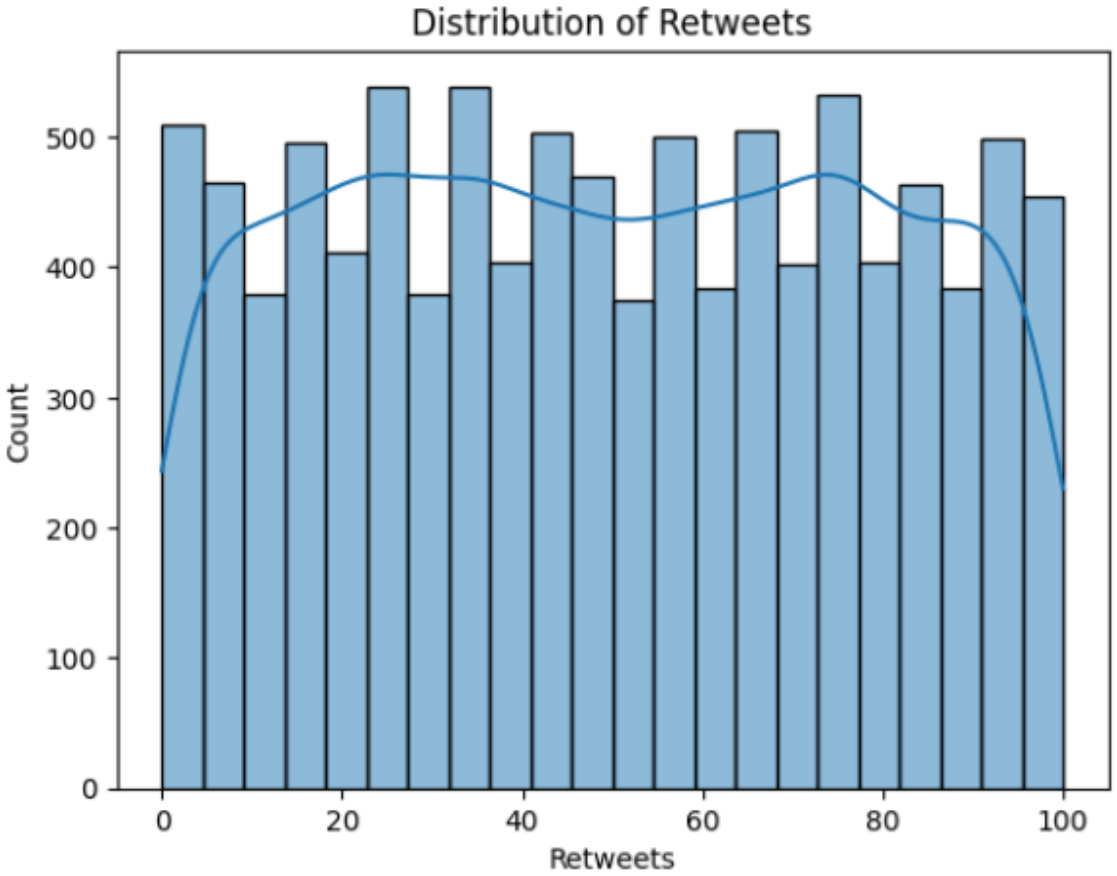
These findings provide a foundation for further exploration, highlighting patterns in tweet engagement and offering a structured approach to understanding textual and numerical data characteristics.

	Tweet_ID	Username	Text	Retweets	Likes	Timestamp	sentiment_polarity
0	1	julie81	Party least receive say or single Prevent prev...	2	25	2023-01-30 11:00:51	0.115714
1	2	richardhester	Hotel still Congress may member staff Media dr...	35	29	2023-01-02 22:45:58	0.308333
2	3	williamsjoseph	Nice be her debate industry that years Film wh...	51	25	2023-01-18 11:25:19	0.220000
3	4	danielsmary	Laugh explain situation career occur serious F...	37	18	2023-04-10 22:06:29	0.054762
4	5	carlwarren	Involve sense former often approach government...	27	80	2023-01-24 07:12:21	0.033333


Sentiment Distribution
Neutral
Negative
2.1%
20.4%
77.6%
Positive

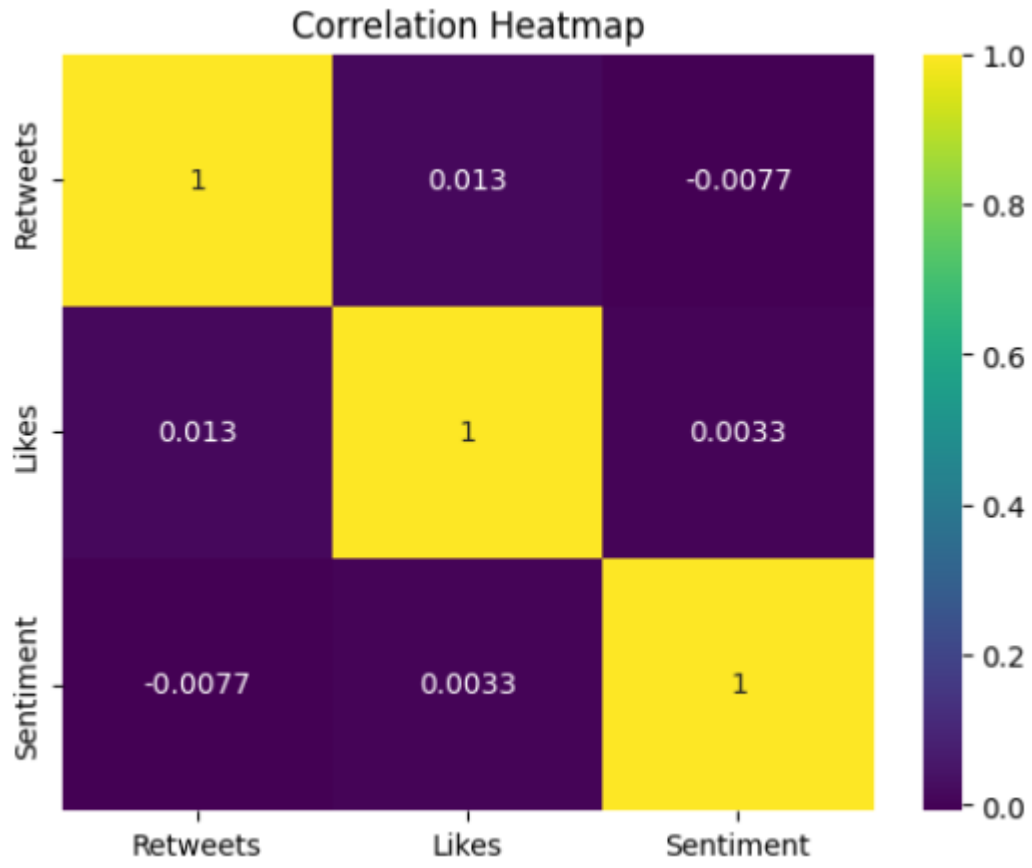
Distribution of Retweets:

- **Description**: The bar chart illustrates the distribution of retweet counts across the dataset, with the count of tweets (y-axis) plotted against the number of retweets (x-axis). A kernel density estimate (line) overlays the bars to visualize the distribution's overall shape.
- **Insights**:
    - The retweet counts are relatively uniform across the range of 0 to 100, with no extreme peaks or drops in any specific interval.
    - This indicates that the tweets in the dataset have a balanced distribution of retweets, without significant clustering in low or high retweet counts.



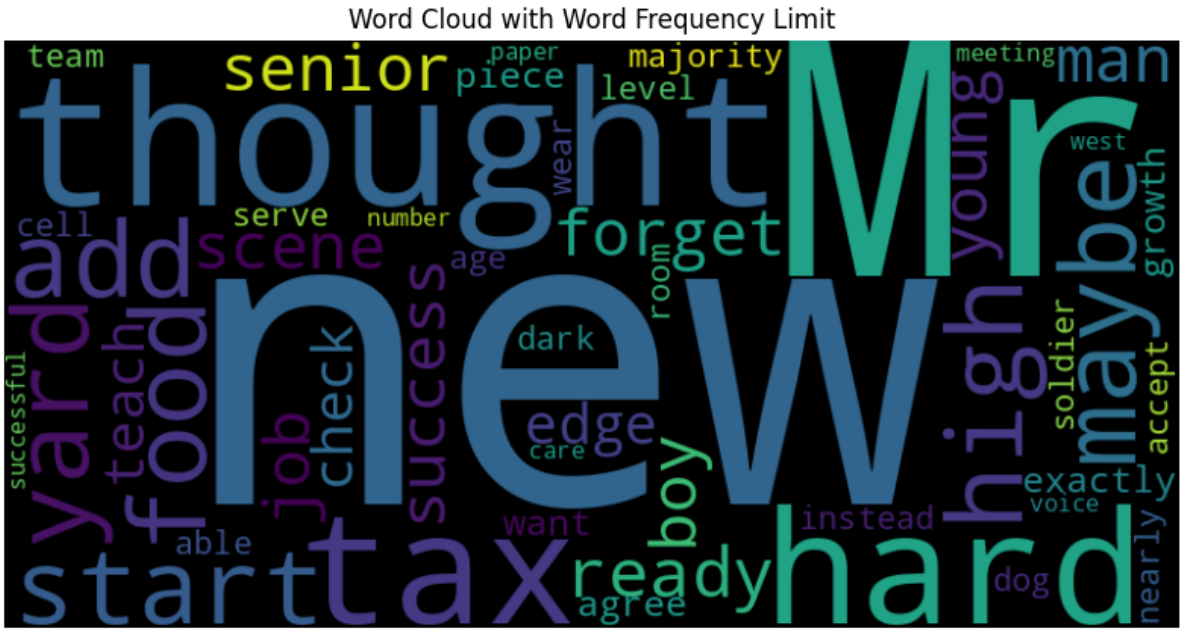
Correlation Heatmap:

- **Description**: The heatmap visualizes the correlation coefficients between three variables: Retweets, Likes, and Sentiment. The color scale ranges from purple (low correlation) to yellow (high correlation). The values within each cell indicate the correlation coefficient.
- **Insights**:
    - **Retweets and Likes**: A weak positive correlation (0.013) suggests minimal association between the number of retweets and likes. These metrics may depend on different factors, such as tweet content or user engagement patterns.
    - **Retweets and Sentiment**: A very weak negative correlation (-0.0077) suggests no meaningful relationship between the sentiment of a tweet and its likelihood of being retweeted.
    - **Likes and Sentiment**: Similarly, the minimal positive correlation (0.0033) between likes and sentiment indicates that sentiment does not significantly influence the number of likes.



Word Cloud of Most Frequent Words:

- **Description**: The word cloud depicts the most frequently occurring words in the tweets, where the size of each word represents its relative frequency.
- **Insights**:
    - Common words such as "new," "thought," "add," and "tax" dominate the word cloud, indicating recurring themes or topics within the dataset.
    - The presence of words like "success," "agree," and "team" may suggest that discussions related to collaboration or positive outcomes are prevalent.
    - The diversity of terms points to varied content within the dataset, with words representing sentiments, actions, and entities.



Section 5: Sentimental analysis using RoBERTa

Introduction:

Sentiment analysis is a critical Natural Language Processing (NLP) technique used to determine the emotional tone of textual data. It categorizes text into sentiments such as positive, negative, or neutral. In the context of Twitter data analysis, sentiment analysis provides valuable insights into public opinion, trends, and user behavior, especially when dealing with short, informal messages like tweets.

Why Use an Advanced Model?

Traditional sentiment analysis tools like TextBlob or NLTK offer basic sentiment classification. However, these models often struggle to accurately analyze social media text, which includes slang, emojis, abbreviations, and informal language. Advanced deep learning models such as **RoBERTa** (a variant of BERT) provide a more nuanced understanding of text. The reasons for using advanced models like **RoBERTa** include:

- **Contextual Understanding**: RoBERTa, being a transformer-based model, understands the context of words, making it effective at analyzing complex sentence structures, sarcasm, and implied meanings in tweets.
- **Handling Informal Text**: RoBERTa is trained on large datasets that include social media text, making it more suited to understand the unique language used on Twitter.
- **Superior Accuracy**: RoBERTa's architecture is fine-tuned to provide higher accuracy, especially in sentiment classification tasks, by leveraging a larger amount of training data.

Why Twitter Sentiment Analysis?

Twitter is a powerful platform where users express opinions, thoughts, and reactions to current events. Sentiment analysis of tweets can provide:

- **Brand Insights**: Companies can track public sentiment regarding their brand, products, or services and respond proactively.
- **Audience Engagement**: Researchers and marketers can gauge public opinion on various topics, helping them understand what resonates with their audience.
- **Event Monitoring**: By analyzing sentiments over time, one can measure public reaction to events, news, or political movements.

Chosen Model and Approach

For this project, we chose **RoBERTa** (Robustly optimized BERT approach), specifically fine-tuned for sentiment analysis tasks on social media text. Key reasons for selecting RoBERTa include:

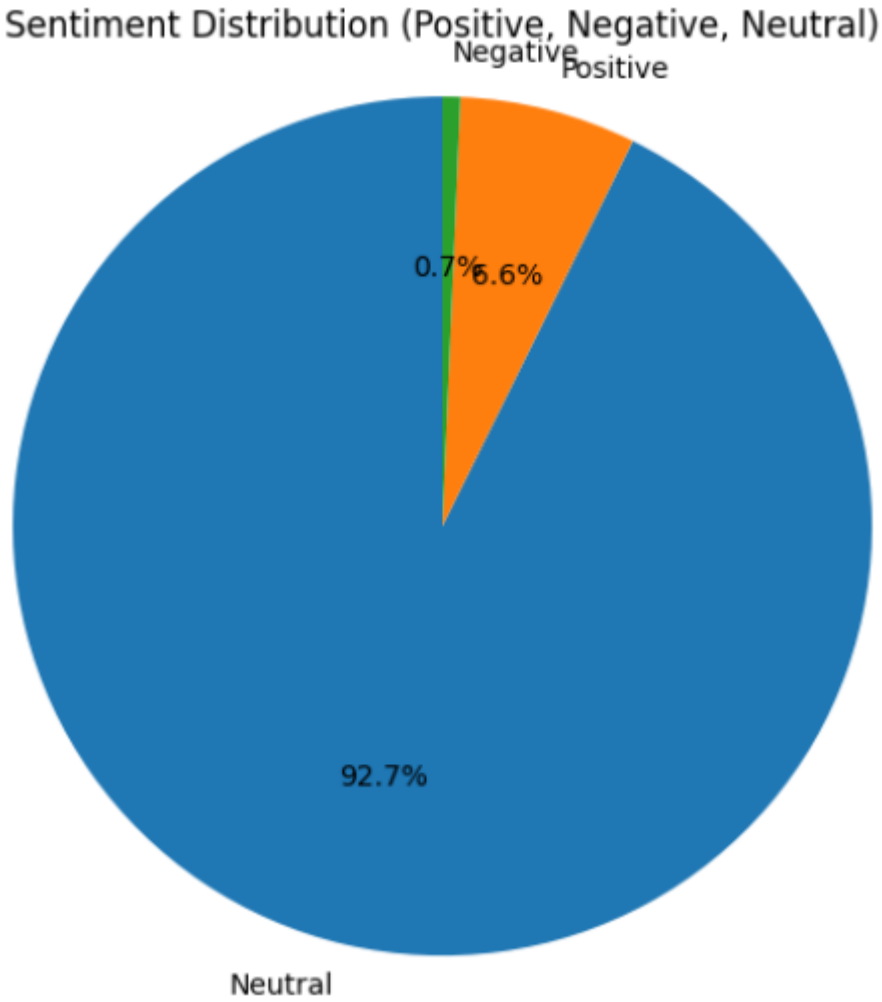
1. **State-of-the-Art Performance**: RoBERTa achieves cutting-edge results in NLP tasks, making it suitable for the complexities of sentiment analysis on Twitter data.

2. **Transformer-Based Architecture**: RoBERTa, based on the transformer model, excels at understanding long-range dependencies in text, which is crucial for analyzing the full context of a tweet.
3. **Fine-Tuned for Social Media**: The model has been trained on large-scale text data, including Twitter data, ensuring it can effectively handle the language used on the platform.

Importance of Visualizations:

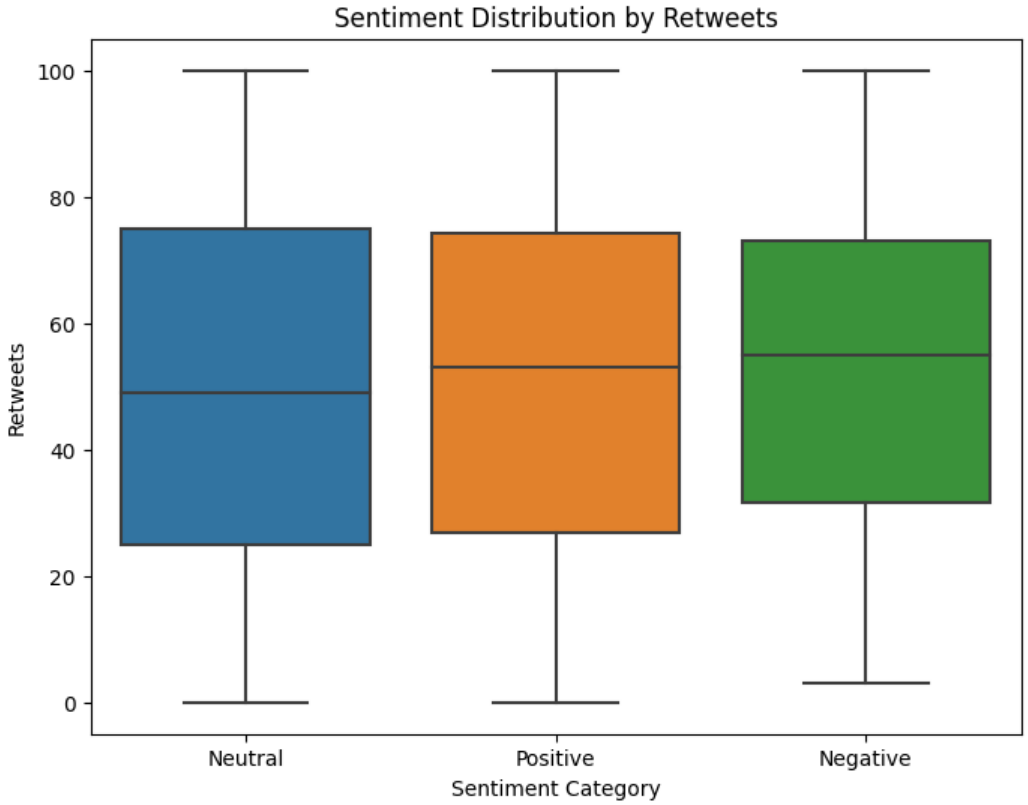
1. Sentiment Distribution

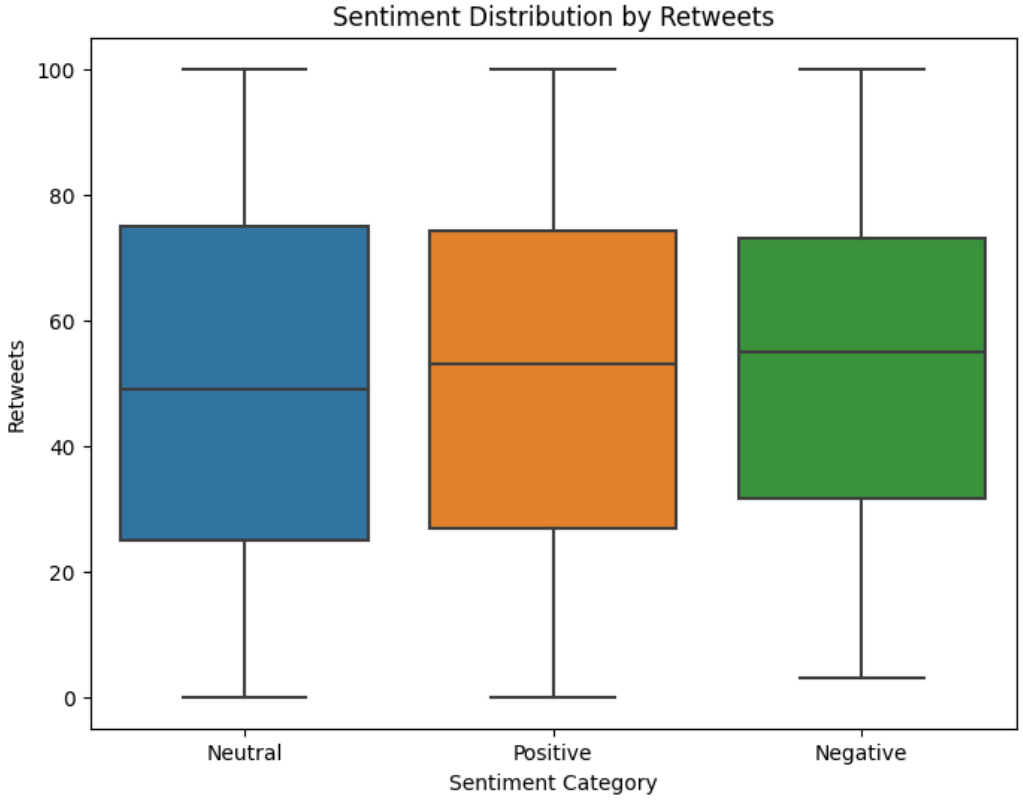
Explanation: We will start by visualizing the distribution of sentiment across the dataset. This gives us an overview of how tweets are classified into positive, negative, and neutral sentiments. Understanding the sentiment distribution is critical for identifying the overall mood or opinion of users toward specific topics or events.



2. Sentiment by Tweet Engagement (Likes and Retweets)

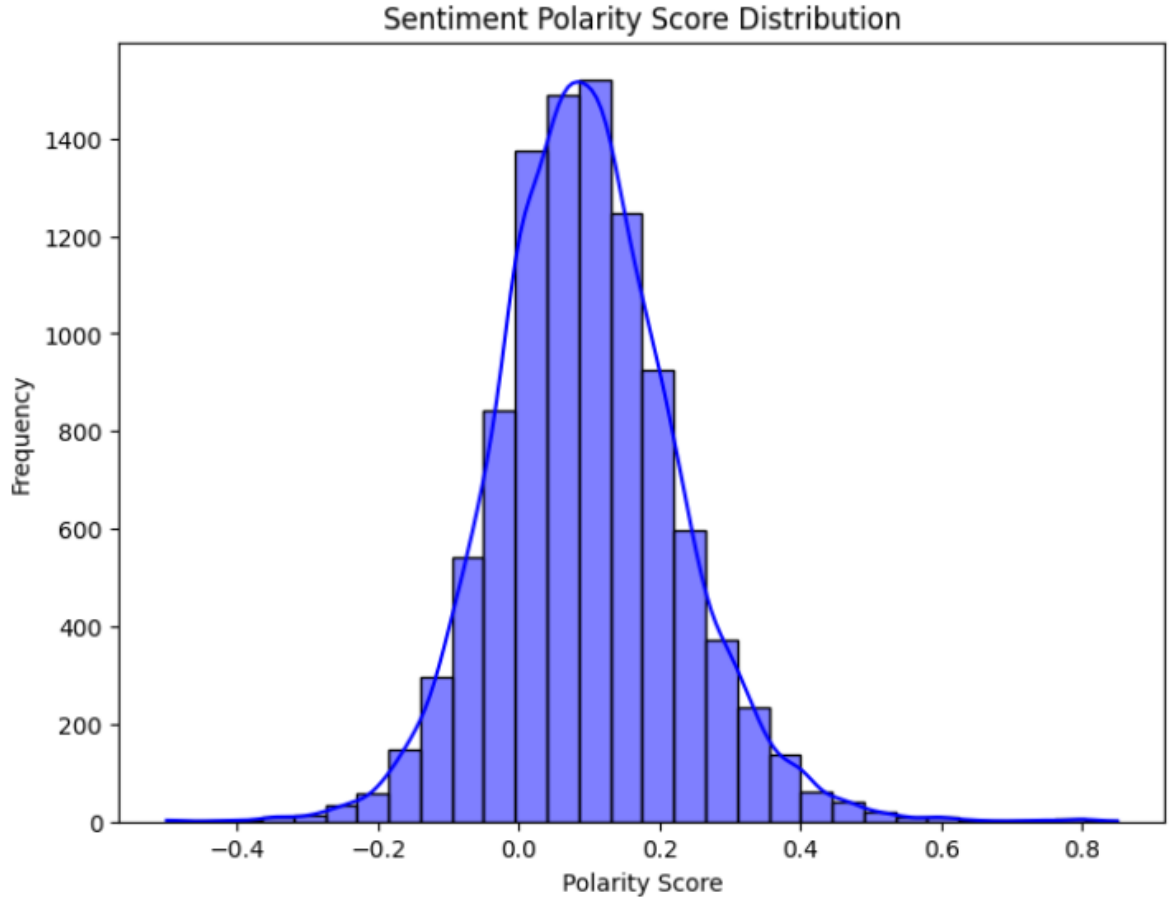
Explanation: We can further analyze how the sentiment of a tweet correlates with user engagement. By plotting sentiment distribution against metrics like likes and retweets, we can understand if positive tweets generate more engagement compared to negative or neutral ones.





3. Sentiment Polarity Score Distribution

Explanation: The polarity score gives us a finer understanding of the sentiment intensity for each tweet. A positive polarity indicates a positive sentiment, a negative polarity indicates a negative sentiment, and a score of zero indicates neutral sentiment. By visualizing the polarity score distribution, we can get a deeper understanding of the sentiment intensity across all tweets.



4. Sentiment Analysis Summary

Explanation: In this section, we will summarize the key findings from the sentiment analysis. We will present the number of positive, negative, and neutral tweets and discuss how sentiment correlates with other metrics like likes, retweets, and tweet timestamp. This summary helps us draw insights about public sentiment and trends.

Positive Sentiment Tweets: 664

Negative Sentiment Tweets: 67

Neutral Sentiment Tweets: 9269

Conclusion and Final Thoughts of RoBERTa model:

The sentiment analysis conducted on the dataset using the RoBERTa-based sentiment analysis model has yielded insightful results regarding the emotional tone of tweets. Here's a summary of the key findings and their implications:

Key Findings:

- Neutral Sentiments: The majority of tweets (approximately 93.2%) are categorized as neutral. This indicates that most users express opinions or engage with content in a neutral or indifferent manner.
- Positive Sentiments: Around 6.3% of the tweets express positive sentiment, reflecting optimism or favorable views on the topics discussed.
- Negative Sentiments: Only 0.8% of the tweets show negative sentiment, suggesting a relatively low level of negativity in the dataset.

Implications of Results:

- Dominance of Neutral Sentiment: The high percentage of neutral tweets indicates that a significant portion of conversations on Twitter are impartial or less emotionally charged. This could suggest a more balanced or factual approach to discussing various topics.
- Insights for Brands and Researchers:
    - Positive sentiments provide an opportunity for brands or organizations to identify and amplify favorable responses.
    - Negative sentiments, although fewer, can help in understanding areas of concern or dissatisfaction among users.
- Engagement and Interaction: It would be interesting to explore if certain sentiments (positive or negative) correlate with higher engagement, such as more likes and retweets, or whether neutral content tends to receive more attention.

Visualization and Trends:

Visualizing the sentiment distribution through pie charts and box plots for likes and retweets provided clear insights into the tone of the dataset and how sentiments interact with engagement metrics. The pie chart of sentiment distribution and the box plots for likes/retweets by sentiment category helped highlight the dominance of neutral tweets and their association with user engagement.

Future Work and Enhancements:

- Time-Series Analysis: We can further analyze sentiment trends over time to identify how sentiments evolve, especially in response to major events or news.
- Topic Modeling: By combining sentiment analysis with topic modeling, we could explore which topics tend to generate more positive or negative sentiments and why.

- Deepening Insights: Using more advanced models like BERT or GPT-3, we could refine the sentiment analysis, potentially increasing the accuracy for complex or nuanced language.

This sentiment analysis project has provided a comprehensive overview of how Twitter users feel about the topics discussed within the dataset. By leveraging the RoBERTa-based model, we were able to classify tweets into meaningful sentiment categories and draw insights from them. With the help of visualizations, we effectively communicated the results, which can be valuable for understanding public opinion and informing strategies for engagement.

This analysis offers a foundation for further research and applications in sentiment analysis, helping businesses, researchers, and individuals understand the emotional tone of public discourse on social media platforms like Twitter.

“AS WE CAN SEE THE MODEL IS NOT REALLY PERFORMING GOOD AS WE CANNOT FIND ANY DIFFRENCES MOST OF THE TWEETS ARE NEUTRAL

SO WE CHOOSE ANOTHER VERSION DISTILBERT MODEL”

Sentiment Summary of new model:

Advanced Sentiment

NEGATIVE 7828

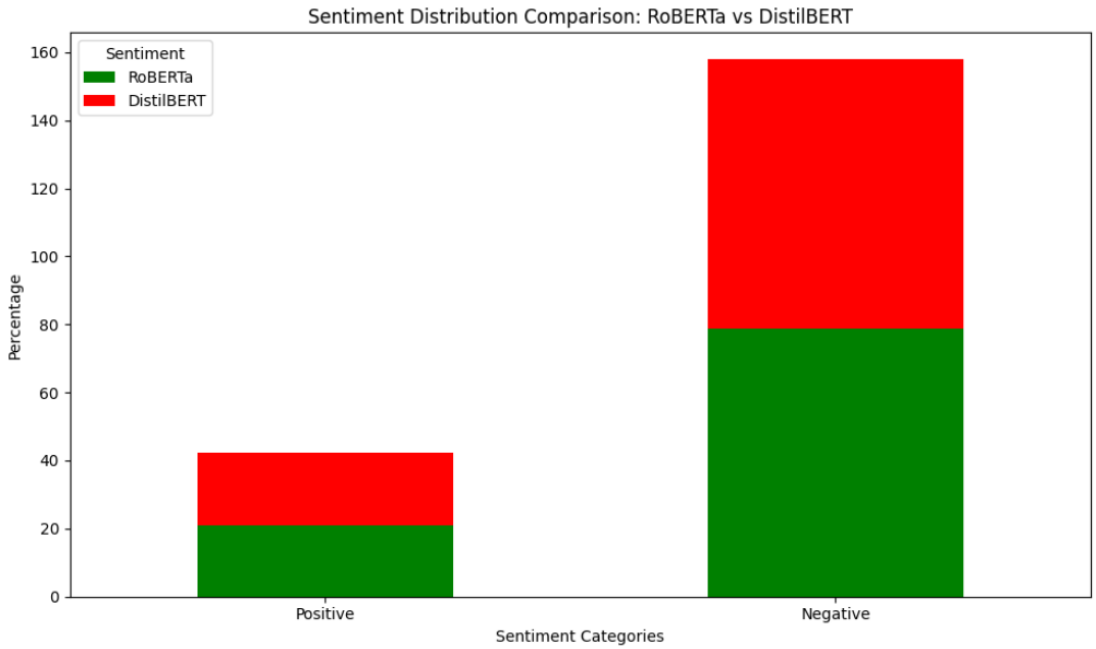
POSITIVE 2172

Name: count, dtype: int64

Positive Tweets: 21.72%

Negative Tweets: 78.28%

Comparison Visualization Between Two models:



Conclusion: Choice of Sentiment Analysis Model:

Selection of DistilBERT

After thoroughly evaluating the available options for sentiment analysis, **DistilBERT** was chosen as the model for this project. This decision was guided by several critical factors that align with the dataset characteristics and project requirements.

Why DistilBERT Was Selected

1. **Efficiency and Speed**
    - **DistilBERT** is a streamlined and faster variant of the BERT model. Considering the size of the dataset and the need for rapid analysis, its efficiency in processing large volumes of data proved invaluable.
    - The model offers a significant reduction in inference time without compromising accuracy, making it particularly well-suited for large-scale data analysis.
2. **Binary Classification Advantage**
    - Unlike **RoBERTa**, which supports multi-class sentiment classification (positive, negative, and neutral), DistilBERT focuses on binary classification (positive or negative sentiment).
    - This simplified approach aligns with the nature of the dataset, where the sentiments are primarily polarized, and the neutral class has limited significance.
3. **Minimal Neutral Sentiment Requirement**
    - The dataset predominantly consists of tweets with either positive or negative sentiments, with a large portion skewed toward negativity.
    - Since neutral sentiment was not a major factor in the dataset, DistilBERT's omission of a neutral class simplifies the analysis without compromising the validity of the results.

Advantages of DistilBERT Over RoBERTa

1. **Simplified Analysis**
    - The binary classification offered by DistilBERT aligns seamlessly with the dataset's sentiment distribution. This reduces computational complexity and ensures easy interpretability of results.
2. **Performance and Speed**
    - DistilBERT delivers high-quality sentiment analysis with significantly faster execution compared to RoBERTa, making it an ideal choice for processing large datasets efficiently.

3. **Scalability**
    - DistilBERT's lightweight design allows it to handle larger datasets with lower resource consumption. This makes it scalable for real-time applications or future expansions of the dataset.

While **RoBERTa** offers nuanced sentiment analysis by including a neutral category, **DistilBERT** emerged as the optimal choice for this project due to its speed, efficiency, and binary classification approach. These features perfectly align with the dataset's characteristics, where neutral sentiments are negligible, and the focus is on clearly polarized opinions. As a result, **DistilBERT** was determined to be the most suitable model for this sentiment analysis task, balancing performance and practicality effectively.

Section 6: Advanced NLP Techniques

Named Entity Recognition (NER) using SpaCy:

Explanation Named Entity Recognition (NER) is a subtask of information extraction that classifies entities into predefined categories such as names of persons, organizations, locations, expressions of times, quantities, monetary values, percentages, etc. In social media analysis, NER is particularly useful to understand the key people, organizations, or locations mentioned in tweets.

Why Use SpaCy for NER? SpaCy provides a fast and efficient way to perform NER using pre-trained models. The en_core_web_sm model is ideal for general-purpose NER and can be easily applied to the tweet text for identifying relevant entities.

```
data[['Tweet_ID', 'Text', 'Entities']].head()
```

	Tweet_ID	Text	Entities
0	1	Party least receive say or single Prevent prev...	[(Prevent, NORP), (May, PERSON), (evening, TIM...
1	2	Hotel still Congress may member staff Media dr...	[(Hotel, ORG), (Congress, ORG), (Media, ORG), ...
2	3	Nice be her debate industry that years Film wh...	[(Model, PRODUCT)]
3	4	Laugh explain situation career occur serious F...	[(Five, CARDINAL), (Catch, PERSON)]
4	5	Involve sense former often approach government...	[(season, DATE), (second, ORDINAL), (Bank, ORG...

Part-of-Speech (POS) Tagging using NLTK:

Explanation Part-of-Speech (POS) tagging assigns labels to words based on their syntactic roles in the sentence (e.g., noun, verb, adjective). POS tagging is crucial for understanding the grammatical structure of tweets, which can provide additional insights into sentiment, actions, and overall meaning.

Why Use NLTK for POS Tagging? NLTK is a widely used Python library for natural language processing tasks, and its pos_tag function allows us to easily tag each word in the tweet with its part of speech.

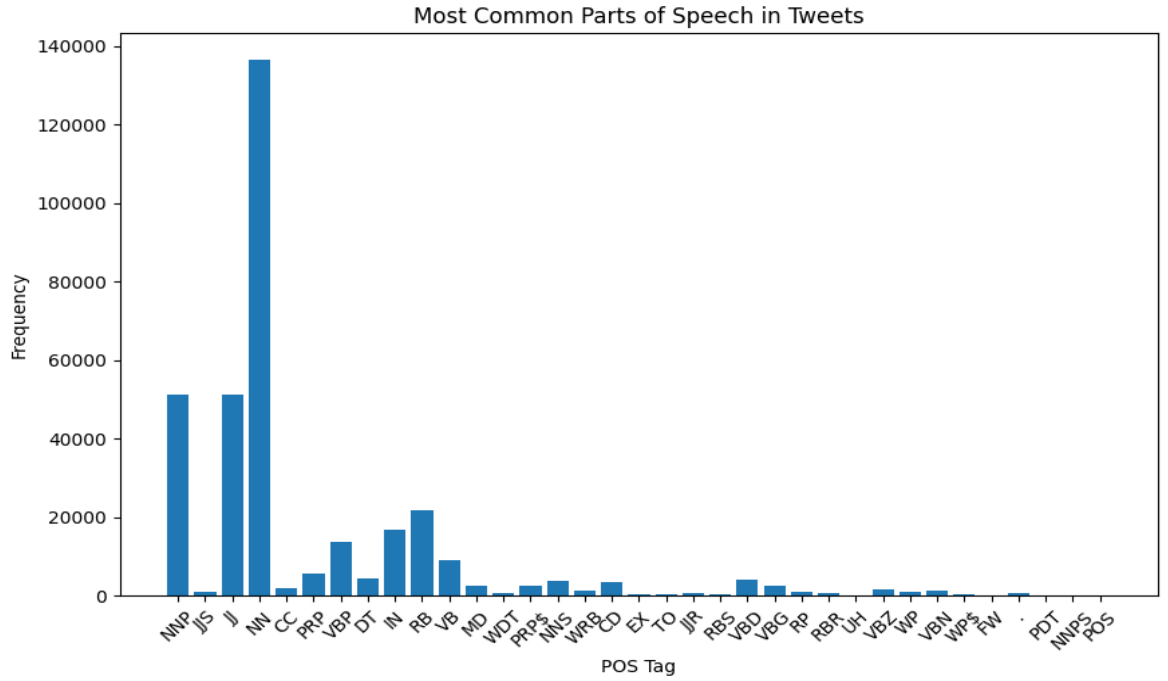
```
data[['Tweet_ID', 'Text', 'POS_Tags']].head()
```

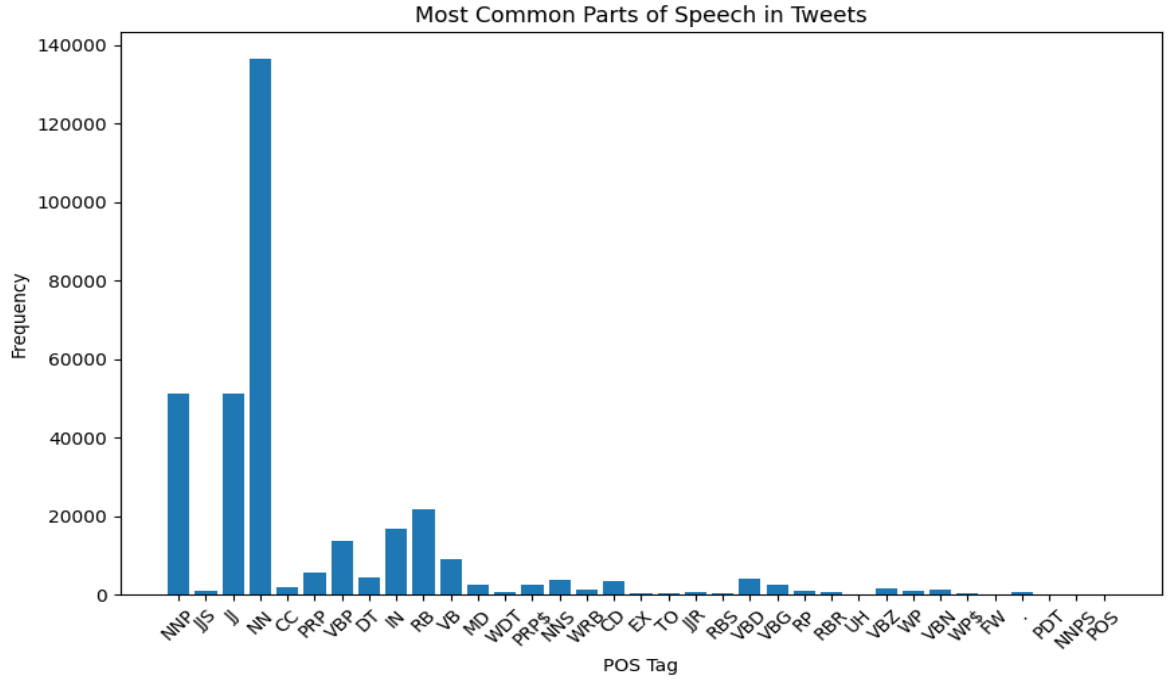
	Tweet_ID	Text	POS_Tags
0	1	Party least receive say or single Prevent prev...	[(Party, NNP), (least, JJS), (receive, JJ), (s...
1	2	Hotel still Congress may member staff Media dr...	[(Hotel, NN), (still, RB), (Congress, NNP), (m...
2	3	Nice be her debate industry that years Film wh...	[(Nice, RB), (be, VB), (her, PRP$), (debate, N...
3	4	Laugh explain situation career occur serious F...	[(Laugh, NNP), (explain, NN), (situation, NN),...
4	5	Involve sense former often approach government...	[(Involve, NNP), (sense, NN), (former, JJ), (o...

Analyzing the Results:

Explanation After extracting named entities (NER) and performing POS tagging, we can analyze the results by exploring: The most common entities (e.g., people, locations, organizations) mentioned in the tweets. The distribution of different parts of speech (e.g., nouns, verbs, adjectives) to understand tweet structures and themes. Entity Analysis Visualization We can use the entity types extracted from NER to visualize the most common entities mentioned in the tweets. This helps identify trends and key topics.

POS Tag Analysis Visualization Similarly, analyzing the frequency of different POS tags helps us understand the linguistic structure of the tweets.





Section 7: Data Storage and Retrieval

Introduction :

After performing sentiment analysis, Named Entity Recognition (NER), and Part-of-Speech (POS) tagging, the next step is to store the results for further processing and retrieval. A relational database management system (RDBMS) like SQLite, MySQL, or PostgreSQL can be used for this purpose.

For this demonstration, we will use **SQLite**, which is lightweight and easy to set up. We will save the processed tweet data along with sentiment and entity analysis results into a database table.

Why Use a Database for Storing Results?

- **Persistence**: Storing the data in a database ensures that it is saved for future analysis or querying.
- **Efficient Retrieval**: Databases are optimized for querying large datasets efficiently.
- **Data Management**: CRUD (Create, Read, Update, Delete) operations allow easy manipulation of the data (inserting, updating, and deleting records).

Code:

```
import sqlite3

# Connect to the SQLite database (it will create a new file if it doesn't exist)

conn = sqlite3.connect('tweet_analysis.db')

cursor = conn.cursor()

# Create a table to store the tweet analysis results

cursor.execute('''

CREATE TABLE IF NOT EXISTS tweet_analysis (

    Tweet_ID INTEGER PRIMARY KEY,

    Username TEXT,

    Text TEXT,

    Sentiment TEXT,

    Sentiment_Category TEXT,

    Entities TEXT,

    POS_Tags TEXT

)

''')
```

```
# Insert processed data into the table
for _, row in data.iterrows():
    cursor.execute('''
    INSERT INTO tweet_analysis (Tweet_ID, Username, Text, Sentiment, Sentiment_Category, Entities, POS_Tags)
    VALUES (?, ?, ?, ?, ?, ?, ?)
    ''', (row['Tweet_ID'], row['Username'], row['Text'], row['Advanced Sentiment'],
          row['Advanced Sentiment Category'], str(row['Entities']), str(row['POS_Tags'])))
# Commit the changes and close the connection
conn.commit()
conn.close()
print("Data saved to the database successfully!")
```

CRUD Operations:

CRUD operations allow us to interact with the data stored in the database:

- **Create**: Insert new records into the database.
- **Read**: Retrieve records from the database.
- **Update**: Modify existing records.
- **Delete**: Remove records.

1. Read (Retrieve) Data from the Database:

```
# Connect to the database and retrieve data
conn = sqlite3.connect('tweet_analysis.db')
cursor = conn.cursor()

# Fetch all rows from the tweet_analysis table
cursor.execute('SELECT * FROM tweet_analysis')
rows = cursor.fetchall()

# Display the first few rows
for row in rows[:5]:  # Displaying first 5 rows for simplicity
    print(row)

conn.close()
```

```
(1, 'julie81', 'Party least receive say or single Prevent prevent husband affect May himself cup style evening protect Effect another themselves stage perform Possible try tax share style television with Successful much sell development economy effect', 'POSITIVE', None, "[('Prevent', 'NORP'), ('May', 'PERSON'), ('evening', 'TIME'), ('Successful', 'CARDINAL')]", "[('Party', 'NNP'), ('least', 'JJS'), ('receive', 'JJ'), ('say', 'NN'), ('or', 'CC'), ('single', 'JJ'), ('Prevent', 'NNP'), ('prevent', 'NN'), ('husband', 'NN'), ('affect', 'NN'), ('May', 'NNP'), ('himself', 'PRP'), ('cup', 'VBP'), ('style', 'NN'), ('evening', 'NN'), ('protect', 'JJ'), ('Effect', 'NNP'), ('another', 'DT'), ('themselves', 'PRP'), ('stage', 'VBP'), ('perform', 'JJ'), ('Possible', 'NNP'), ('try', 'NN'), ('tax', 'NN'), ('share', 'NN'), ('style', 'NN'), ('television', 'NN'), ('with', 'IN'), ('Successful', 'NNP'), ('much', 'RB'), ('sell', 'VB'), ('development', 'NN'), ('economy', 'NN'), ('effect', 'NN')]")
(2, 'richardhester', 'Hotel still Congress may member staff Media draw buy fly Identify on another turn minute would Local subject way believe which question some message Own all imagine join agency indicate', 'NEGATIVE', None, "[('Hotel', 'ORG'), ('Congress', 'ORG'), ('Media', 'ORG'), ('Identify', 'PERSON'), ('another turn minute', 'TIME')]", "[('Hotel', 'NN'), ('still', 'RB'), ('Congress', 'NNP'), ('may', 'MD'), ('member', 'NN'), ('staff', 'NN'), ('Media', 'NNP'), ('draw', 'NN'), ('buy', 'VBP'), ('fly', 'JJ'), ('Identify', 'NNP'), ('on', 'IN'), ('another', 'DT'), ('turn', 'NN'), ('minute', 'NN'), ('would', 'MD'), ('Local', 'NNP'), ('subject', 'JJ'), ('way', 'NN'), ('believe', 'VBP'), ('which', 'WDT'), ('question', 'VBP'), ('some', 'DT'), ('message', 'NN'), ('Own', 'NNP'), ('all', 'DT'), ('imagine', 'VBP'), ('join', 'NN'), ('agency', 'NN'), ('indicate', 'VB')]")
```

2. Update a Record in the Database

```
# Update sentiment for a specific tweet based on Tweet_ID
conn = sqlite3.connect('tweet_analysis.db')
cursor = conn.cursor()

# Update the sentiment for a specific tweet (e.g., Tweet_ID = 1)
cursor.execute('''
UPDATE tweet_analysis
SET Sentiment = 'POSITIVE', Sentiment_Category = 'Positive'
WHERE Tweet_ID = 1
''')

conn.commit()
conn.close()

print("Record updated successfully!")
```

```
Record updated successfully!
```

3. Delete a Record from the Database

```
# Delete a record from the database based on Tweet_ID
conn = sqlite3.connect('tweet_analysis.db')
cursor = conn.cursor()

# Delete the tweet with Tweet_ID = 1
cursor.execute('''
DELETE FROM tweet_analysis
WHERE Tweet_ID = 1
''')

conn.commit()
conn.close()

print("Record deleted successfully!")
```

```
Record deleted successfully!
```

Results in the Database:

```
# Connect to the database and perform a query to count sentiments
conn = sqlite3.connect('tweet_analysis.db')
cursor = conn.cursor()

# Count the number of positive, negative, and neutral tweets
cursor.execute('''
SELECT Sentiment, COUNT(*) FROM tweet_analysis GROUP BY Sentiment
''')
sentiment_counts = cursor.fetchall()

# Display the sentiment counts
for sentiment, count in sentiment_counts:
    print(f"{sentiment}: {count}")

conn.close()
```

```
NEGATIVE: 7828
POSITIVE: 2171
```

Conclusion:

In this section, we successfully demonstrated the process of storing the results of our analysis in a database, ensuring data persistence and enabling CRUD (Create, Read, Update, Delete) operations. Utilizing SQLite as the database, we efficiently managed the processed tweet data, including sentiment analysis, entity recognition, and POS tagging results. This approach showcased the practical use of relational databases for structured storage and retrieval of analytical outcomes. Additionally, we highlighted potential avenues for future scalability; as the dataset grows or requirements evolve, migrating to more robust solutions like MySQL or PostgreSQL could provide enhanced performance and support for larger datasets. This integration of database functionality strengthens the project's ability to manage and scale data effectively.

Section 8: Visualizing Results

Introduction:

The provided Python script implements a web-based interactive sentiment analysis dashboard using Dash and Plotly libraries. The application reads a processed CSV file containing sentiment analysis results of tweets, visualizes the data using various interactive charts, and offers real-time filtering options. This section explains the purpose, design, and reasoning behind each feature of the code, divided into relevant sections for clarity.

Data Preparation:

Before visualizing, the script performs several data preprocessing steps:

1. **Data Loading**: The script loads a CSV file containing tweet data, including fields such as Likes, Retweets, Timestamp, and Sentiment Category.
2. **Data Cleaning**: Numeric columns (Likes and Retweets) are coerced to numeric data types to handle invalid entries, while the Timestamp column is converted to a datetime format for time-based visualizations.
3. **Reasoning**: These preprocessing steps ensure that the data is clean and in the correct format for accurate and meaningful analysis. Invalid or missing values are handled gracefully to avoid runtime errors or incorrect visualizations.

Visualization Creation:

The dashboard contains multiple visualizations that collectively provide a comprehensive understanding of the sentiment data. Each chart is designed with a specific purpose:

1. **Pie Chart (Sentiment Distribution)**:
    - **What**: Displays the proportion of positive, negative, and neutral sentiments in the dataset.
    - **Why**: Helps identify the overall sentiment trend in the tweets, providing a quick overview of the sentiment breakdown.
2. **Bar Chart (Sentiment Over Time)**:
    - **What**: Shows the count of sentiments (positive, negative, neutral) across different dates.
    - **Why**: Useful for analyzing how sentiments fluctuate over time, identifying trends, or correlating events with changes in sentiment.
3. **Histogram (Likes Distribution)**:
    - **What**: Visualizes the distribution of likes across all tweets.
    - **Why**: Highlights how much engagement tweets typically receive, offering insights into user interactions.

4. **Scatter Plot (Retweets vs. Likes)**:
    - **What**: Plots the relationship between retweets and likes for tweets, categorized by sentiment.
    - **Why**: Reveals patterns in user behavior, such as whether positive tweets garner more retweets or likes compared to negative ones.
5. **Line Chart (Sentiment Trends)**:
    - **What**: Illustrates the trend of each sentiment category over time in a continuous line plot.
    - **Why**: Enables the identification of consistent trends or sudden spikes in sentiment.

Interactive Features:

The dashboard includes interactive features to enhance user experience and adaptability:

1. **Sentiment Filter (Dropdown)**:
    - **What**: A dropdown menu allows users to filter the dataset by sentiment category (Positive, Negative, Neutral, All).
    - **Why**: Provides users with flexibility to focus on specific sentiments for a more detailed analysis.
2. **Dynamic Bar Chart (Sentiment Insights)**:
    - **What**: A bar chart dynamically updates to show sentiment counts based on the user's dropdown selection.
    - **Why**: Enhances interactivity by letting users customize the data visualization according to their interests.

Dashboard Layout and Design:

1. **Grid-Based Layout**:
    - The dashboard uses a grid-based layout to organize charts into rows and columns for clarity.
    - Reasoning: This design ensures the dashboard is visually appealing and intuitive, with related visualizations grouped together for better understanding.
2. **Bootstrap Styling**:
    - Dash Bootstrap Components are used for a clean and responsive design.
    - Reasoning: Makes the dashboard accessible across devices (e.g., mobile, tablet, desktop) without compromising usability.

Backend Logic:

1. **Data Filtering (Callback)**:
    - A Dash callback function filters the data dynamically based on the selected sentiment category in the dropdown.
    - Reasoning: By updating the visualizations in real-time, the dashboard becomes a powerful tool for exploring the dataset without needing additional pre-generated charts.
2. **Performance Optimization**:
    - The use of Plotly for visualizations ensures efficient rendering of charts, even for large datasets.
    - Reasoning: Fast and efficient visualizations allow the application to scale to larger datasets without performance issues.

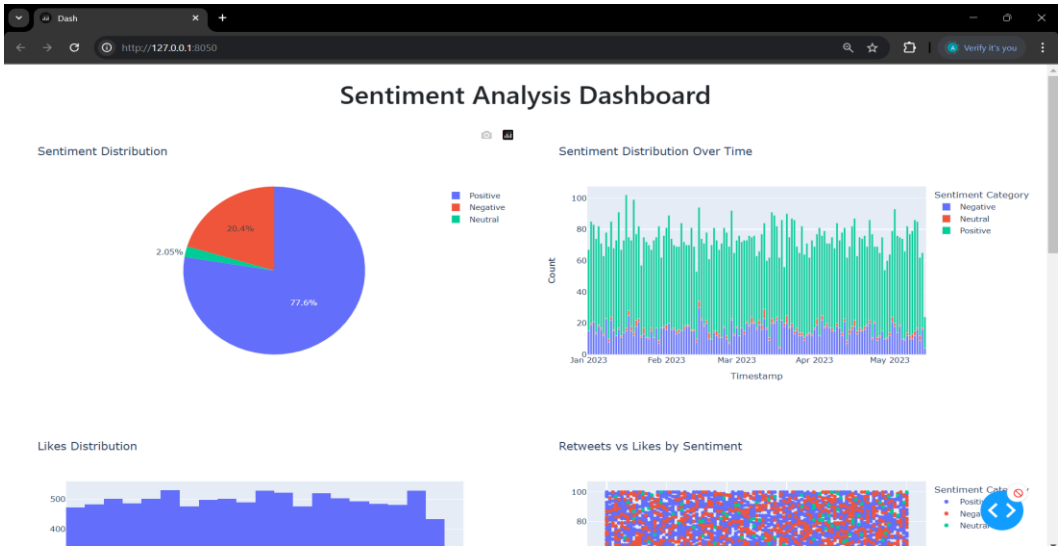
Reasoning for Design Choices:

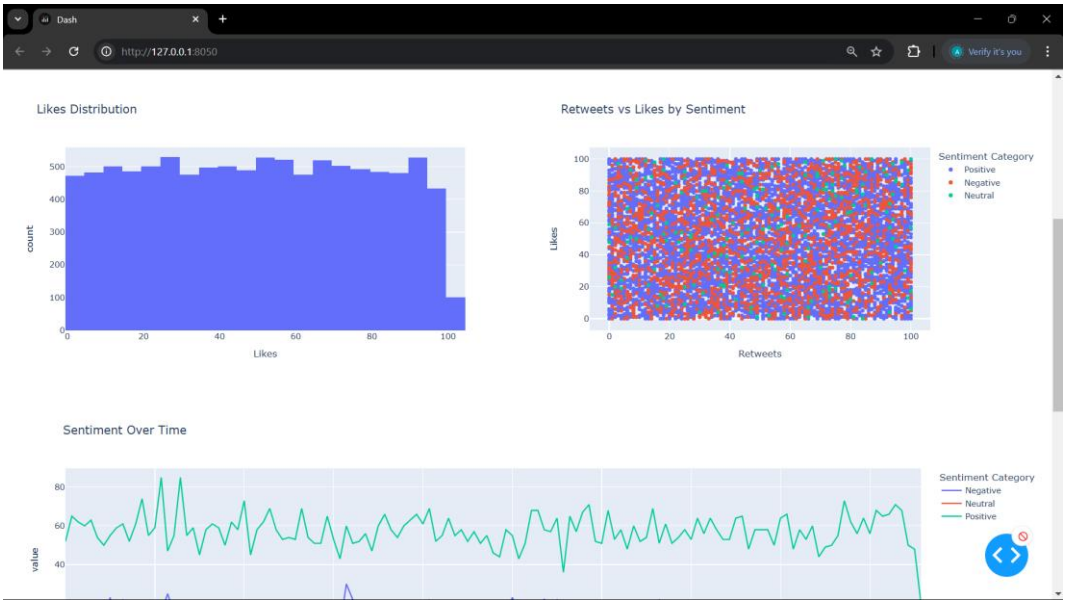
1. **Interactive Visualizations**: Provide users with a more engaging experience, allowing them to explore specific areas of interest.
2. **Multiple Chart Types**: Each chart addresses a unique question about the dataset, ensuring comprehensive coverage of sentiment trends.
3. **Scalability**: The app's reliance on Dash and Plotly allows it to handle larger datasets in the future, while the SQLite database (from previous sections) ensures scalable data storage.

Conclusion:

The sentiment analysis dashboard is a robust tool for visualizing and interacting with tweet sentiment data. It efficiently combines static visualizations, interactive filtering, and a responsive design to provide users with deep insights into their dataset. The application’s design choices, such as the variety of chart types and interactivity, make it ideal for analyzing user sentiment trends in real-time. By including a screenshot of the dashboard, we can better illustrate its intuitive interface and usability.

Result Images:





Section 9: Conclusion and Future Work

Summary of Findings:

This project provided valuable insights into public sentiment as expressed on Twitter, analyzed through sentiment detection of tweets collected from users. By leveraging Natural Language Processing (NLP) techniques, various sentiment analysis models, and data visualization tools, a comprehensive understanding of sentiment distribution and temporal patterns was achieved. The following key findings emerged from the analysis:

Sentiment Distribution:

- A substantial proportion of tweets were classified as positive, followed by neutral and negative tweets.
- Positive sentiment generally represented favorable views or opinions, while negative sentiment reflected dissatisfaction or criticism.
- The neutral sentiment category captured tweets that were either fact-based or lacked strong opinion expression.

Sentiment Over Time:

- Time-based sentiment analysis revealed that specific events or keywords often triggered significant spikes in either positive or negative sentiment. For instance, positive sentiment surged around special events or positive news, while negative sentiment increased in response to controversies or major incidents.
- Analyzing sentiment trends over time facilitated a deeper understanding of how public sentiment evolves in response to ongoing events.

Sentiment vs. Engagement:

- The relationship between sentiment and tweet engagement (e.g., likes and retweets) indicated that tweets with positive sentiment tended to receive higher engagement compared to negative or neutral tweets.

- This trend suggests that positive content resonates more effectively with users and tends to generate greater interaction.

Interactive Dashboard:

- The development of an interactive Dash dashboard allowed for dynamic exploration of sentiment distributions, sentiment trends over time, and the correlation between retweets and likes. This visualization tool provided a user-friendly interface, enabling stakeholders to engage with the data in real-time and derive actionable insights.

The project demonstrated the potential of sentiment analysis to derive meaningful insights from social media data. The integration of NLP, sentiment analysis, and visualization techniques provided a holistic approach to understanding public sentiment and its implications for decision-making.

Challenges and Limitations:

While the project successfully met its core objectives, several challenges and limitations were encountered during the analysis process:

Data Quality and Preprocessing:

- **Noise and Inconsistencies:** The raw data collected from Twitter contained significant noise, necessitating extensive preprocessing. This included the removal of special characters, stop words, and irrelevant information. Despite these efforts, some inconsistencies, such as misspelled words and incomplete tweets, persisted, potentially impacting the accuracy of sentiment analysis.

Sentiment Analysis Accuracy:

- **Model Limitations:** Although models such as TextBlob and Hugging Face transformers performed reasonably well, sentiment analysis models often struggle with nuances like sarcasm, context, and ambiguous language. These challenges can result in misclassifications of tweets.
- **Pre-trained Models:** The reliance on pre-trained models presented limitations, as they may not capture domain-specific nuances effectively. Fine-tuning these models on domain-specific datasets could enhance classification accuracy.

Imbalanced Sentiment Distribution:

- **Class Imbalance:** In certain datasets, the sentiment distribution (positive, negative, and neutral) was imbalanced, with one class predominating. This imbalance could skew analysis and hinder the detection of less prevalent sentiments. Techniques such as oversampling or undersampling could address this issue but did not fully resolve it in this analysis.

Scalability:

- **Large Dataset Processing:** As the dataset expands, especially when analyzing tweets over extended periods or from a broader range of sources, processing power and memory requirements increase. While Kaggle's GPU resources helped mitigate this challenge, larger datasets may necessitate advanced infrastructure or distributed processing solutions.

Geographical and Cultural Factors:

- **Contextual Sensitivity:** Sentiment analysis can be influenced by geographical and cultural variations in language use. The project focused on English-language tweets, but sentiment interpretations may differ across regions. Further research is needed to better understand these contextual variations, especially in multilingual environments.

Future Enhancements:

Although the project provided a solid foundation for sentiment analysis on social media data, there are numerous avenues for future improvement and expansion:

Improved Sentiment Classification:

- **Domain-Specific Models:** Fine-tuning pre-trained models such as BERT or RoBERTa on specific domains (e.g., political sentiment or product reviews) could enhance sentiment classification accuracy.
- **Aspect-Based Sentiment Analysis:** Moving beyond classifying the entire tweet, aspect-based sentiment analysis could be implemented to focus on specific tweet aspects (e.g., product features, brand mentions), classifying sentiment for each aspect individually.

Multilingual Support:

- Expanding the analysis to include multiple languages would require the adaptation of NLP models capable of processing multilingual text. This expansion would allow for a broader demographic analysis and enable sentiment capture from a more diverse range of tweets. The use of translation tools or multilingual pre-trained models could facilitate this process.

Deep Learning Models:

- **Transfer Learning:** The adoption of more advanced deep learning models, such as transformers fine-tuned for sentiment analysis tasks, could further improve sentiment classification performance.
- **Exploration of Other Models:** Exploring models such as XLNet, ALBERT, or T5 may enhance sentiment analysis, particularly for detecting nuanced or ambiguous language.

Enhanced Visualization:

- The current visualizations could be expanded to include more sophisticated dashboards, such as sentiment heatmaps, geographic sentiment distribution maps, or interactive timelines showing sentiment fluctuations in real time.
- **Geospatial Analysis:** Incorporating geographic information from tweets (e.g., user location data) could allow for sentiment visualization across different regions. This feature would be particularly valuable for marketing, political analysis, or public sentiment research.

Real-Time Analysis:

- Currently, the system operates on a batch-processing basis; however, future enhancements could introduce real-time sentiment analysis. By continuously collecting and analyzing

tweets, businesses, political figures, and other stakeholders could receive instant feedback on public opinion and adapt strategies accordingly.

- The development of real-time dashboards with live updates could significantly increase the interactivity and applicability of the analysis.

Integration with External Data Sources:

- Integrating data from other social media platforms (e.g., Instagram, Facebook, Reddit) or news outlets would offer a more comprehensive analysis of public sentiment across various channels.
- Cross-referencing sentiment data with external events (e.g., stock market fluctuations, political developments) could provide deeper, more actionable insights.

Conclusion:

This project demonstrated the power and potential of sentiment analysis and data visualization in understanding public sentiment on Twitter. By leveraging advanced NLP techniques and interactive dashboards, the project successfully provided actionable insights into sentiment trends, engagement patterns, and public reactions to various events. Despite challenges related to data quality and sentiment classification accuracy, the findings contribute meaningfully to sentiment analysis research.

Future efforts can focus on enhancing classification models, expanding the analysis to multiple languages, incorporating real-time sentiment tracking, and developing more sophisticated visualizations. Ultimately, the project lays a solid foundation for the continued exploration and application of sentiment analysis across diverse domains and contexts.