SRINIVAS MILESTONE 2
srinivasacademics@gmail.com

Key Objectives

The goal of this milestone is to perform sentiment analysis on raw tweet data using parallel processing. The workflow involves:
- Reading and preprocessing raw tweet data.

- Performing sentiment analysis (Positive, Neutral, Negative) using TextBlob.

- Applying multiprocessing to speed up sentiment computation.

- Storing results (with new columns) into SQLite3 database.

- Sending an email summary report of sentiment distribution and sample tweets..

Dataset
Source: tweets dataset in csv format from kaggle

Attributes
New Column added **content_for_sentiment**, **sentiment**

Code

Data Loading and Preprocessing

The dataset is loaded from the SQLite3 database (`tweets.db`) into a Pandas DataFrame.
Preprocessing steps include:
- Converting text to lowercase
- Removing emojis and special characters
- Retaining raw content for sentiment analysis in a new column named content_for_sentiment

```
import sqlite3
import pandas as pd
import re

conn = sqlite3.connect("tweets.db")
df = pd.read_sql_query("SELECT * FROM tweets", conn)

# Preprocess text
def clean_text(text):
    text = text.lower()
    text = re.sub(r'[^\w\s]', '', text)    # Remove
punctuation
    text = re.sub(r'[^\x00-\x7F]+', '', text)  # Remove
emojis and non-ASCII
    return text

df['content_for_sentiment'] =
df['content'].apply(clean_text)
print(df[['content', 'content_for_sentiment']].head())
```

CountVectorizer & TF-IDF

CountVectorizer:
- Initialized with stop_words='english' and min_df=0.01 to filter out common English words and rare terms.
- The top 10 most frequent terms are identified and plotted.

TF-IDF Vectorizer:
- Initialized with stop_words='english', min_df=0.01, and max_df=0.8 to focus on terms that are significant but not overly common.
- The top 10 terms with the highest TF-IDF weights are identified and plotted.

Count Vectorizer

```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer,
TfidfVectorizer
import matplotlib.pyplot as plt
from collections import Counter
# Default Count Vectorizer
count_vectorizer_default = CountVectorizer()
X_count_default =
count_vectorizer_default.fit_transform(df['content_for_senti
ment'])

# CountVectorizer with updated parameters
count_vectorizer = CountVectorizer(
    stop_words='english',  # remove common stop words
    min_df=0.01             # include terms in at least 10%
of documents
)
X_count =
count_vectorizer.fit_transform(df['content_for_sentiment'])

# Convert to DataFrame
count_df = pd.DataFrame(X_count.toarray(),
columns=count_vectorizer.get_feature_names_out())

# Top 10 most common terms
```

```
term_sums =
count_df.sum(axis=0).sort_values(ascending=False)
top_10_terms = term_sums.head(10)
print("Top 10 most common terms (Count Vectorizer):")
print(top_10_terms)

# Top 10 least common terms that appear in at least 10% of
docs
least_10_terms = term_sums.tail(10)
print("\nTop 10 least common terms (min_df=10%)")
print(least_10_terms)

# Horizontal bar chart for top 10 most common terms
plt.figure(figsize=(10,6))
top_10_terms.sort_values().plot(kind='barh',
color='skyblue')
plt.title("Top 10 Most Common Terms (Count Vectorizer)")
plt.xlabel("Frequency")
plt.show()
```

TF-IDF Vectorizer

```
# Default TF-IDF Vectorizer
tfidf_vectorizer_default = TfidfVectorizer()
X_tfidf_default =
tfidf_vectorizer_default.fit_transform(df['content_for_senti
ment'])

# TF-IDF Vectorizer with updated parameters
tfidf_vectorizer = TfidfVectorizer(
    stop_words='english',  # remove stop words
    min_df=0.01,            # appear in at least 10% of
documents
    max_df=0.8             # appear in at most 50% of
documents
)
X_tfidf =
tfidf_vectorizer.fit_transform(df['content_for_sentiment'])

# Convert to DataFrame
```

```
tfidf_df = pd.DataFrame(X_tfidf.toarray(),
columns=tfidf_vectorizer.get_feature_names_out())

# Top 10 most highly weighted terms
tfidf_sums =
tfidf_df.sum(axis=0).sort_values(ascending=False)
top_10_tfidf = tfidf_sums.head(10)
print("\nTop 10 terms by TF-IDF weight:")
print(top_10_tfidf)

# Horizontal bar chart for TF-IDF top terms
plt.figure(figsize=(10,6))
top_10_tfidf.sort_values().plot(kind='barh', color='salmon')
plt.title("Top 10 Terms by TF-IDF Weight")
plt.xlabel("TF-IDF Score")
plt.show()
```

Sentiment Analysis using TextBlob with Multiprocessing

Sentiment analysis is performed using TextBlob to determine the polarity of each tweet:

- Polarity > 0 → Positive
- Polarity < 0 → Negative
- Polarity = 0 → Neutral

To speed up the process, Python's multiprocessing Pool is used to process tweets in parallel.

```
from textblob import TextBlob
from multiprocessing import Pool, cpu_count

def analyze_sentiment(text):
    try:
        polarity = TextBlob(text).sentiment.polarity
        if polarity > 0:
            return 'positive'
        elif polarity < 0:
            return 'negative'
        else:
            return 'neutral'
    except:
        return 'neutral'

if __name__ == "__main__":
    texts = df['content_for_sentiment'].tolist()
    num_cores = cpu_count()
    with Pool(num_cores) as pool:
        sentiments = pool.map(analyze_sentiment, texts)

    df['sentiment'] = sentiments
```

Store Results in Database

The analyzed data is stored in table Column named sentiment, without deleting the old data.

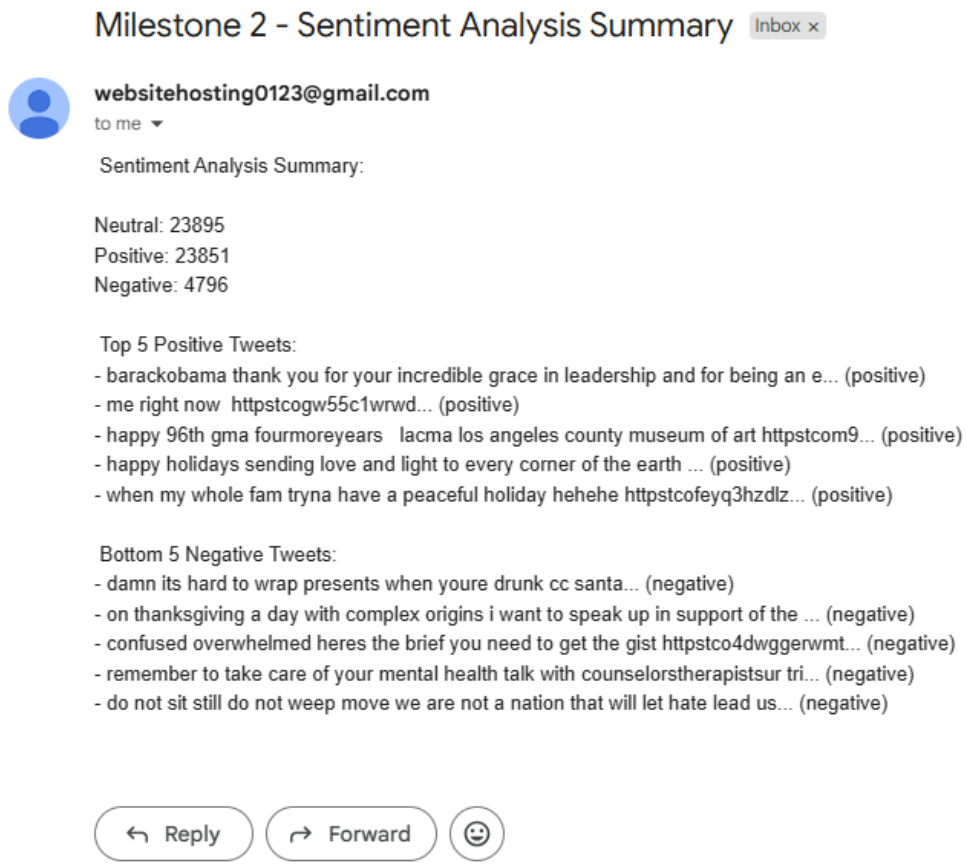
```
for idx, row in df.iterrows():
    cursor.execute(
        "UPDATE tweets SET content_for_sentiment = ? WHERE rowid = ?",
        (row['content_for_sentiment'], idx+1)  # rowid in SQLite starts from 1
    )
# Update each row in the database
for idx, row in df.iterrows():
    cursor.execute(
        "UPDATE tweets SET sentiment = ? WHERE rowid = ?",
        (row['sentiment'], idx+1)  # rowid in SQLite starts from 1
    )

conn.commit()
```

Send Summary Mail

An email is automatically sent summarizing the sentiment results.

```
summary = df['sentiment'].value_counts().to_dict()
print("Sentiment Summary:", summary)

import smtplib
from email.mime.text import MIMEText

# Format email content
body = " Sentiment Analysis Summary:\n\n"
for sentiment, count in summary.items():
    body += f"{sentiment.capitalize()}: {count}\n"

# Add Top 5 Positive examples
if 'positive' in df['sentiment'].unique():
    top5 = df[df['sentiment'] ==
'positive'].head(5)[['content_for_sentiment', 'sentiment']]
    body += "\n Top 5 Positive Tweets:\n"
    for i, row in top5.iterrows():
        body += f"- {row['content_for_sentiment'][:80]}...
({row['sentiment']})\n"

# Add Bottom 5 Negative examples
if 'negative' in df['sentiment'].unique():
    bottom5 = df[df['sentiment'] ==
'negative'].head(5)[['content_for_sentiment', 'sentiment']]
    body += "\n Bottom 5 Negative Tweets:\n"
    for i, row in bottom5.iterrows():
        body += f"- {row['content_for_sentiment'][:80]}...
({row['sentiment']})\n"

msg = MIMEText(body)
msg['Subject'] = "Milestone 2 - Sentiment Analysis Summary"
msg['From'] = "websitehosting0123@gmail.com"
msg['To'] = "mobilepersonaluse5@gmail.com"

# Send email (use your credentials or app password)
```

```
with smtplib.SMTP_SSL('smtp.gmail.com', 465) as server:
    server.login("websitehosting0123@gmail.com", "********")
    server.sendmail(msg['From'], [msg['To']],
msg.as_string())

print("Summary email with examples sent!")
```



Milestone 2 - Sentiment Analysis Summary Inbox ×

websitehosting0123@gmail.com
to me

Sentiment Analysis Summary:

Neutral: 23895
Positive: 23851
Negative: 4796

Top 5 Positive Tweets:
- barackobama thank you for your incredible grace in leadership and for being an e... (positive)
- me right now httpstcogw55c1wrwd... (positive)
- happy 96th gma fourmoreyears lacma los angeles county museum of art httpstcom9... (positive)
- happy holidays sending love and light to every corner of the earth ... (positive)
- when my whole fam tryna have a peaceful holiday hehehe httpstcofeyq3hzdlz... (positive)

Bottom 5 Negative Tweets:
- damn its hard to wrap presents when youre drunk cc santa... (negative)
- on thanksgiving a day with complex origins i want to speak up in support of the ... (negative)
- confused overwhelmed heres the brief you need to get the gist httpstco4dwggerwmt... (negative)
- remember to take care of your mental health talk with counselorstherapistsur tri... (negative)
- do not sit still do not weep move we are not a nation that will let hate lead us... (negative)

Reply Forward