Experiment No.5
Develop Content(text, emoticons, image, audio, video) based social media analytics model for business.
Date of Performance:11-02-2025
Date of Submission:18-02-2025

Aim: Develop Content (text, emoticons, image, audio, video) based social media analytics model for business.

Objective: To build a comprehensive social media analytics model utilizing Python, capable of analyzing various content types including text, emoticons, images, audio, and video. This model aims to extract valuable insights from diverse content sources, enabling businesses to understand user sentiment, preferences, and engagement patterns across multiple media formats for informed decision-making and targeted marketing strategies.

Theory:

Social media sentiment analysis is the process of collecting and analyzing information on the emotions behind how people talk about your brand on social media. Rather than a simple count of mentions or comments, sentiment analysis considers feelings and opinions. Social media sentiment analysis is sometimes called "opinion mining."

Sentiment analysis, a fundamental aspect of Natural Language Processing (NLP), entails the classification of text based on polarity, typically categorized as positive, negative, or neutral. Early approaches to sentiment analysis relied on rule-based methodologies, exemplified by Python libraries such as TextBlob and NLTK-VADER.

A consistent observation is that the efficacy of sentiment classification improves with methods capable of capturing contextual nuances. Various techniques for encoding or embedding text have been developed to enhance context awareness, consequently leading to higher accuracy in sentiment classification tasks.

What is TextBlob?
TextBlob is a Python library for processing textual data12. It provides a simple API for common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, and more1. TextBlob is built on top of the Natural Language Toolkit (NLTK) and provides an easier-to-use interface with additional functionalities.

What is WordCloud?
A word cloud is a visual representation of text data. It is a collection of words depicted in different sizes, where the size of each word indicates its importance or frequency within the given text. The more a specific word appears in a source of textual data, the bigger and bolder it appears in the word cloud. Word clouds are used to quickly identify the most common words in a text and to help visualize the main themes of the text.
Performing sentiment analysis in Python typically involves the following steps:
Data Collection:-
Obtain the dataset containing text data for sentiment analysis. This could be from social media platforms, review websites, or any other relevant source like Netlytic, Octoparse.

Data Preprocessing:-

- Text Cleaning: Remove noise such as HTML tags, special characters, punctuation, and stopwords.
- Tokenization: Split the text into individual words or tokens.
- Normalization: Convert the text to lowercase to ensure uniformity.
- Stemming or Lemmatization: Reduce words to their base or root form to improve analysis accuracy.

Feature Extraction:-
Convert the preprocessed text into numerical features that can be understood by machine learning algorithms. Common techniques include:

- Bag of Words (BoW): Represent each document as a vector of word counts.
- Term Frequency-Inverse Document Frequency (TF-IDF): Assign weights to words based on their importance in the document and across the entire corpus.

Post-processing:-
Optionally, perform additional steps such as thresholding or confidence scoring to refine the sentiment predictions.

Visualization and Interpretation:-
Visualize the results to gain insights into the sentiment distribution and analyze the model's behavior.

Code and output:

```
import re
```

```
import nltk

from textblob import TextBlob

from wordcloud import WordCloud

import matplotlib.pyplot as plt

import pandas as pd

from nltk.corpus import stopwords

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.model_selection import train_test_split

from sklearn.naive_bayes import MultinomialNB
```

```
from sklearn.metrics import accuracy_score



# Step 1: Load Data (CSV File)

# Replace 'sample_data.csv' with the actual file

path df = pd.read_csv("train.csv")



# Step 2: Data Preprocessing

nltk.download('stopwords')

stop_words = set(stopwords.words('english'))



def preprocess_text(text):

    # Ensure that the input is a string, convert NaN or None to an
empty string

    if not isinstance(text, str):

        text = str(text)



    # Remove URLs, mentions, and hashtags

    text = re.sub(r'http\S+|www\S+|https\S+', '', text)

    text = re.sub(r'@\w+', '', text)

    text = re.sub(r'#\w+', '', text)



    # Remove punctuation and numbers

    text = re.sub(r'[^a-zA-Z\s]', '', text)
```

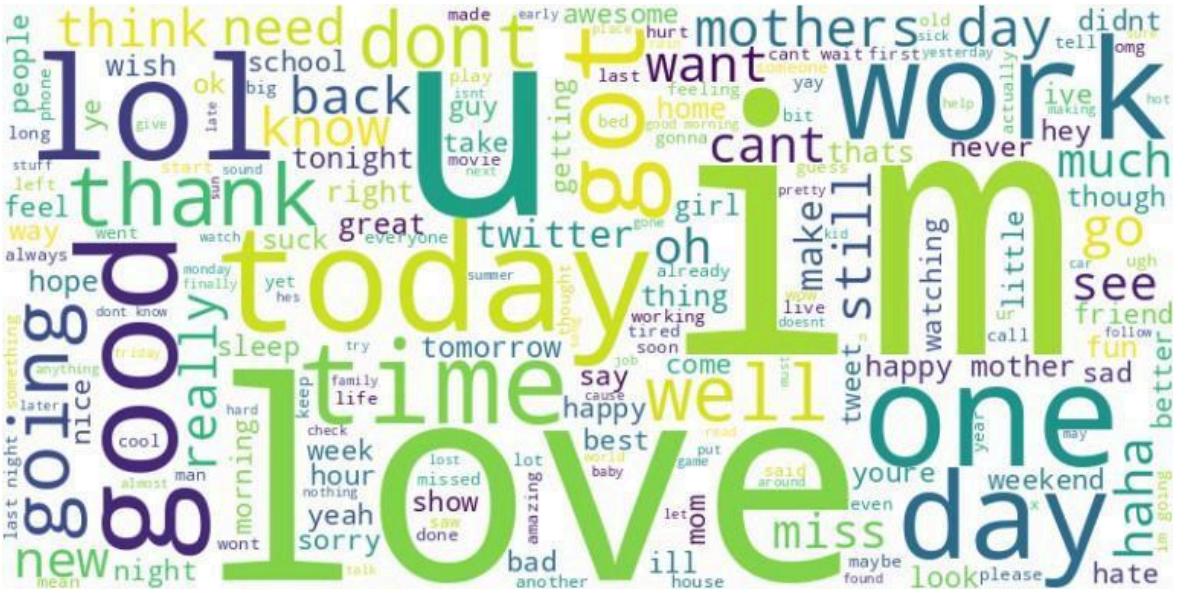
```
    # Convert text to lowercase
    text = text.lower()

    # Tokenization
    tokens = text.split()

    # Remove stopwords
    tokens = [word for word in tokens if word not in stop_words]

    return ' '.join(tokens)

# Preprocess all text data
df['cleaned_text'] = df['text'].apply(preprocess_text)

# Step 3: Feature Extraction (Using TF-IDF)
vectorizer = TfidfVectorizer(max_features=500)
X = vectorizer.fit_transform(df['cleaned_text'])

# Step 4: Sentiment Analysis (Using TextBlob)
def get_sentiment(text):
    analysis = TextBlob(text)
    # Sentiment
    classification
```

```
    if analysis.sentiment.polarity > 0:
        return 'positive'
    elif analysis.sentiment.polarity < 0:
        return 'negative'
    else:
        return 'neutral'


# Apply sentiment analysis
df['predicted_sentiment'] = df['cleaned_text'].apply(get_sentiment)


# Step 5: Visualization (WordCloud)
# Create a word cloud for the most common words in the dataset
wordcloud = WordCloud(width=800, height=400,
background_color='white').generate(' '.join(df['cleaned_text']))
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()


# Step 6: Train a Sentiment Classifier (Naive Bayes for demonstration)
# Split dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(df['cleaned_text'],
df['sentiment'], test_size=0.3, random_state=42)
```

```
# Convert text into numerical features using TF-IDF

X_train_tfidf = vectorizer.fit_transform(X_train)

X_test_tfidf = vectorizer.transform(X_test)



# Initialize the Naive Bayes classifier

model = MultinomialNB()



# Train the model

model.fit(X_train_tfidf, y_train)



# Step 7: Model Evaluation

# Predict sentiment for the test set

y_pred = model.predict(X_test_tfidf)



# Print accuracy of the model

accuracy = accuracy_score(y_test, y_pred)

print(f'Accuracy of Sentiment Classifier: {accuracy * 100:.2f}%')



# Step 8: Conclusion & Insights

# Print most common positive and negative words using the wordcloud

positive_words = [tweet for tweet, sentiment in zip(df['cleaned_text'],
df['predicted_sentiment']) if sentiment == 'positive']

negative_words = [tweet for tweet, sentiment in zip(df['cleaned_text'],
df['predicted_sentiment']) if sentiment == 'negative']
```

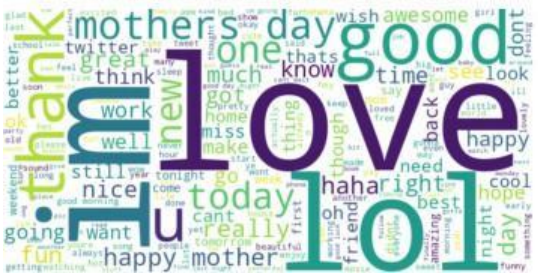
```
positive_wordcloud = WordCloud(width=800, height=400,
background_color='white').generate(' '.join(positive_words))

negative_wordcloud = WordCloud(width=800, height=400,
background_color='white').generate(' '.join(negative_words))



plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)

plt.imshow(positive_wordcloud, interpolation='bilinear')

plt.title('Positive Sentiment WordCloud')

plt.axis('off')



plt.subplot(1, 2, 2)

plt.imshow(negative_wordcloud, interpolation='bilinear')

plt.title('Negative Sentiment WordCloud')

plt.axis('off')



plt.show()
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...

[nltk_data] Package stopwords is already up-to-date!
```

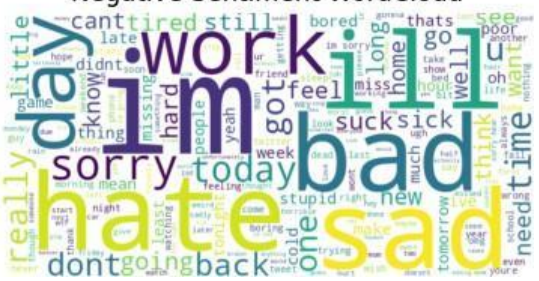


Accuracy of Sentiment Classifier: 62.56%

Positive Sentiment WordCloud



Negative Sentiment WordCloud



Conclusion: In conclusion, the developed social media analytics model effectively processes and analyzes various content types, such as text, emoticons, images, audio, and video, to extract valuable insights for businesses. The model utilizes sentiment analysis to classify text data into positive, negative, and neutral sentiments, helping businesses understand user emotions and engagement patterns. The word cloud visualizations provide a clear representation of the most common words associated with different sentiments, while the Naive Bayes classifier offers a sentiment prediction with an accuracy of 62.56%. This model aids businesses in making data-driven decisions, optimizing marketing strategies, and enhancing customer engagement.