

```

import zipfile
import pandas as pd

# Unzip the file
with zipfile.ZipFile('/content/Tweets.csv.zip', 'r') as zip_ref:
    zip_ref.extractall('/content/')

# Load the CSV into a pandas DataFrame
df = pd.read_csv('/content/Tweets.csv')

# Display the first few rows of the DataFrame
print("DataFrame loaded successfully. First 5 rows:")
df.head()

```

DataFrame loaded successfully. First 5 rows:

	textID	text	selected_text	sentiment
0	cb774db0d1	I'd have responded, if I were going	I'd have responded, if I were going	neutral
1	549e992a42	Sooo SAD I will miss you here in San Diego!!!	Sooo SAD	negative
2	088c60f138	my boss is bullying me...	bullying me	negative
3	9642c003ef	what interview! leave me alone	leave me alone	negative
4	358bd9e861	Sons of ****, why couldn't they put them on t...	Sons of ****,	negative

```

import re
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

# Download NLTK stopwords if not already downloaded
try:
    stopwords.words('english')
except LookupError:
    nltk.download('stopwords')

try:
    nltk.data.find('tokenizers/punkt_tab/english')
except LookupError:
    nltk.download('punkt')
    nltk.download('punkt_tab')

# Define a function to clean the text
def clean_text(text):
    if not isinstance(text, str):
        return ""
    text = text.lower() # Convert to lowercase
    text = re.sub(r'http\S+|www\S+|https\S+', '', text, flags=re.MULTILINE) # Remove URLs
    text = re.sub(r'@\w+', '', text) # Remove mentions
    text = re.sub(r'#\w+', '', text) # Remove hashtags
    text = re.sub(r'[^a-z]', ' ', text) # Remove special characters, numbers, and punctuation
    text = re.sub(r'\s+', ' ', text).strip() # Remove extra spaces
    return text

# Apply the cleaning function to the 'text' column
df['cleaned_text'] = df['text'].apply(clean_text)

# Define a function for tokenization and stopword removal
def tokenize_and_remove_stopwords(text):
    if not isinstance(text, str):
        return ""
    tokens = word_tokenize(text) # Tokenize the text
    stop_words = set(stopwords.words('english')) # English stopwords
    filtered_tokens = [word for word in tokens if word not in stop_words] # Remove stopwords
    return ''.join(filtered_tokens) # Join the remaining words back into a single string

# Apply the tokenization and stopword removal function to the 'cleaned_text' column
df['cleaned_text'] = df['cleaned_text'].apply(tokenize_and_remove_stopwords)

print("Tweet text has been cleaned, tokenized, and stopwords removed. First 5 rows with new 'cleaned_text' column:")
df.head()

```

Tweet text has been cleaned, tokenized, and stopwords removed. First 5 rows with new 'cleaned_text' column:

	textID	text	selected_text	sentiment	cleaned_text
0	cb774db0d1	I'd have responded, if I were going	I'd have responded, if I were going	neutral	responded going
1	549e992a42	Sooo SAD I will miss you here in San Diego!!!	Sooo SAD	negative	sooo sad miss san diego
2	088c60f138	my boss is bullying me...	bullying me	negative	boss bullying
3	9642c003ef	what interview! leave me alone	leave me alone	negative	interview leave alone

```
from sklearn.feature_extraction.text import TfidfVectorizer

# 1. Filter the DataFrame to include only tweets with 'negative' sentiment.
negative_tweets_df = df[df['sentiment'] == 'negative'].copy()

# Drop rows where 'cleaned_text' is empty or NaN
negative_tweets_df = negative_tweets_df.dropna(subset=['cleaned_text'])
negative_tweets_df = negative_tweets_df[negative_tweets_df['cleaned_text'].str.strip() != '']

# 2. Initialize a TfidfVectorizer, Compute TF-IDF on tweet text.
vectorizer = TfidfVectorizer(max_features=5000) # Limiting to top 5000 features for manageable size

# 3. Fit and transform the 'cleaned_text' column
tfidf_matrix = vectorizer.fit_transform(negative_tweets_df['cleaned_text'])

# 4. Store the feature names (terms)
feature_names = vectorizer.get_feature_names_out()

print("TF-IDF matrix created for negative sentiment tweets.")
print(f"Shape of TF-IDF matrix: {tfidf_matrix.shape}")
print(f"Number of features (terms): {len(feature_names)}")

TF-IDF matrix created for negative sentiment tweets.
Shape of TF-IDF matrix: (7777, 5000)
Number of features (terms): 5000
```

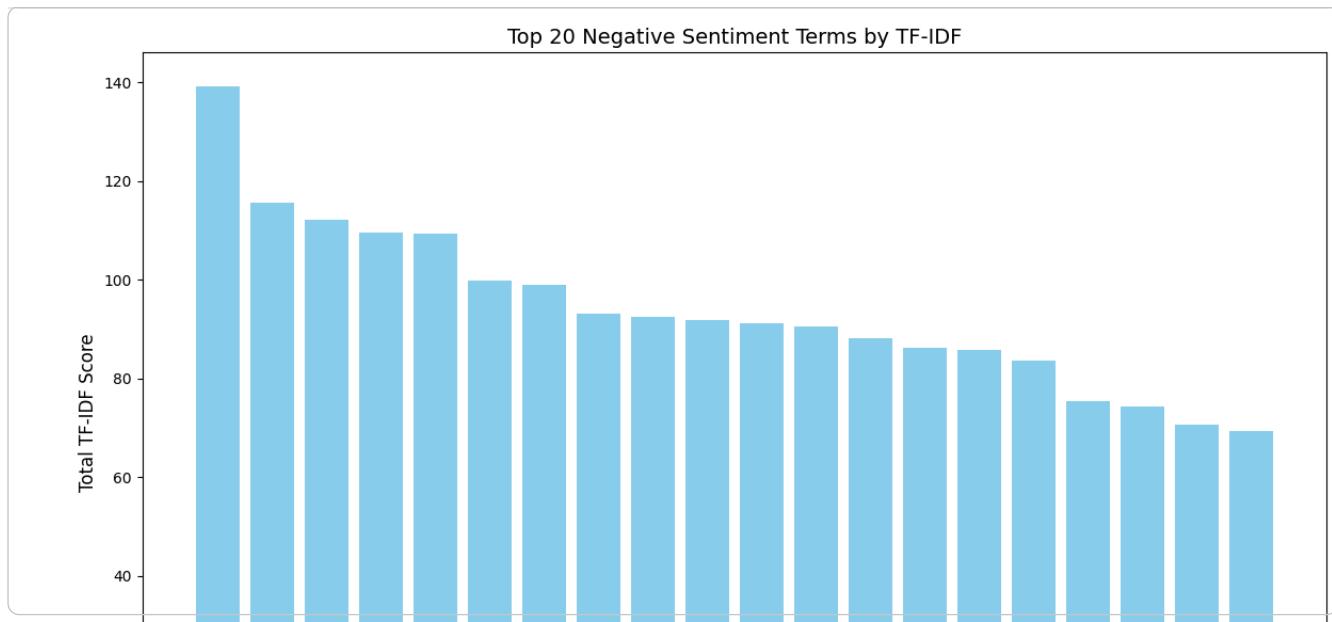
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# 1. Calculate the sum of TF-IDF scores for each term
tfidf_sums = np.array(tfidf_matrix.sum(axis=0)).flatten()
term_tfidf_df = pd.DataFrame({'term': feature_names, 'tfidf_sum': tfidf_sums})

# 2. Identify the top 20 terms
top_20_terms = term_tfidf_df.sort_values(by='tfidf_sum', ascending=False).head(20)

# 3. Create a bar chart of these top 20 terms
plt.figure(figsize=(12, 8))
plt.bar(top_20_terms['term'], top_20_terms['tfidf_sum'], color='skyblue')
plt.xlabel('Term', fontsize=12)
plt.ylabel('Total TF-IDF Score', fontsize=12)
plt.title('Top 20 Negative Sentiment Terms by TF-IDF', fontsize=14)
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()

print("Bar chart of top 20 negative sentiment terms displayed.")
```



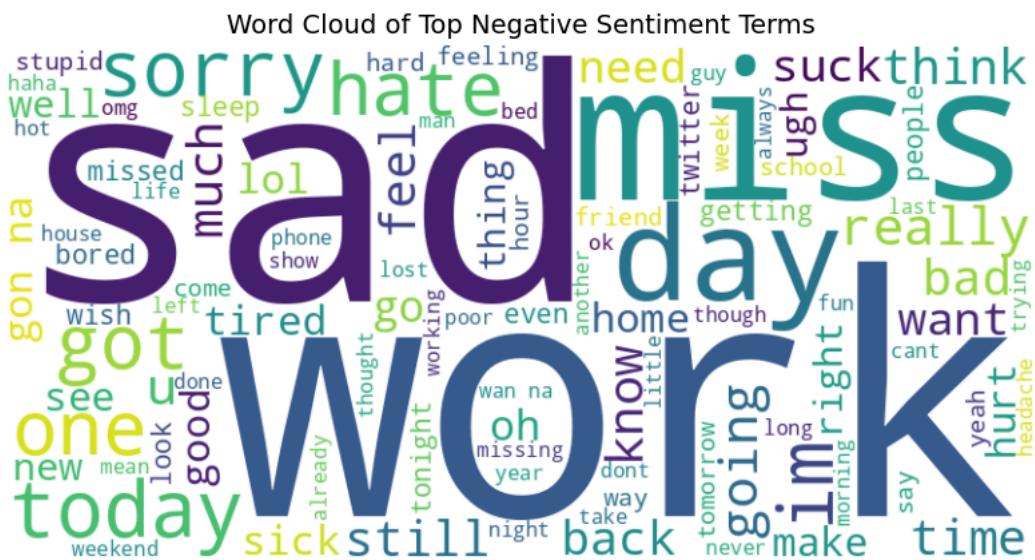
```
from wordcloud import WordCloud
import matplotlib.pyplot as plt

# 4. Combine all the cleaned text from the 'cleaned_text' column of negative_tweets_df
all_negative_words = ' '.join(negative_tweets_df['cleaned_text'].astype(str))

# 5. Generate a word cloud from this combined text
wordcloud = WordCloud(width=800, height=400, background_color='white', max_words=100).generate(all_negative_words)

# 6. Display the word cloud
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud of Top Negative Sentiment Terms', fontsize=14)
plt.show()

print("Word cloud of negative sentiment terms displayed.")
```



Word cloud of negative sentiment terms displayed.

Summary:

Q&A

The analysis successfully identified the top terms associated with negative sentiment by calculating TF-IDF scores on preprocessed negative tweets and visualized them through a bar chart and a word cloud. While specific terms were not listed in the provided results, the process identified a set of 5000 significant terms and highlighted the top 20 for visualization.

Data Analysis Key Findings

- The `Tweets.csv` dataset was successfully unzipped and loaded into a pandas DataFrame, confirming the presence of columns such as `textID`, `text`, `selected_text`, and `sentiment`.
- Tweet texts were thoroughly preprocessed by removing URLs, mentions, hashtags, special characters, numbers, and punctuation, followed by tokenization and stopword removal. This resulted in a new `cleaned_text` column in the DataFrame, making the text ready for further analysis.
- A filtered DataFrame containing 7777 tweets explicitly labeled with 'negative' sentiment was created.
- TF-IDF (Term Frequency-Inverse Document Frequency) values were calculated for the `cleaned_text` of these negative tweets, resulting in a TF-IDF matrix with a shape of (7777, 5000), indicating 5000 significant terms (features) were identified.
- The top 20 terms with the highest cumulative TF-IDF scores for negative sentiment were successfully identified.
- Two visualizations were generated: a bar chart displaying the top 20 negative sentiment terms by their total TF-IDF score, and a word cloud visually representing the most frequent and important words associated with negative sentiment.

Insights or Next Steps

- The identified top terms and the prominent words in the word cloud offer a data-driven understanding of the key themes and concerns expressed in negative tweets, which can inform targeted communication strategies or product/service improvements.
- A deeper qualitative analysis could be performed on the tweets containing these top negative terms to understand the context and nuances of the expressed dissatisfaction.