Data loading

Subtask:

Load the twitter_training.csv file into a pandas DataFrame.

Reasoning: Load the dataframe and display the first few rows.

```
In [1]: import pandas as pd

        try:
            df = pd.read_csv('twitter_training.csv')
            display(df.head())
        except FileNotFoundError:
            print("Error: 'twitter_training.csv' not found.")
        except pd.errors.ParserError:
            print("Error: Could not parse the CSV file. Check the file format.")
        except Exception as e:
            print(f"An unexpected error occurred: {e}")
```

	2401	Borderlands	Positive	im getting on borderlands and i will murder you all ,
0	2401	Borderlands	Positive	I am coming to the borders and I will kill you...
1	2401	Borderlands	Positive	im getting on borderlands and i will kill you ...
2	2401	Borderlands	Positive	im coming on borderlands and i will murder you...
3	2401	Borderlands	Positive	im getting on borderlands 2 and i will murder ...
4	2401	Borderlands	Positive	im getting into borderlands and i can murder y...

Data exploration

Subtask:

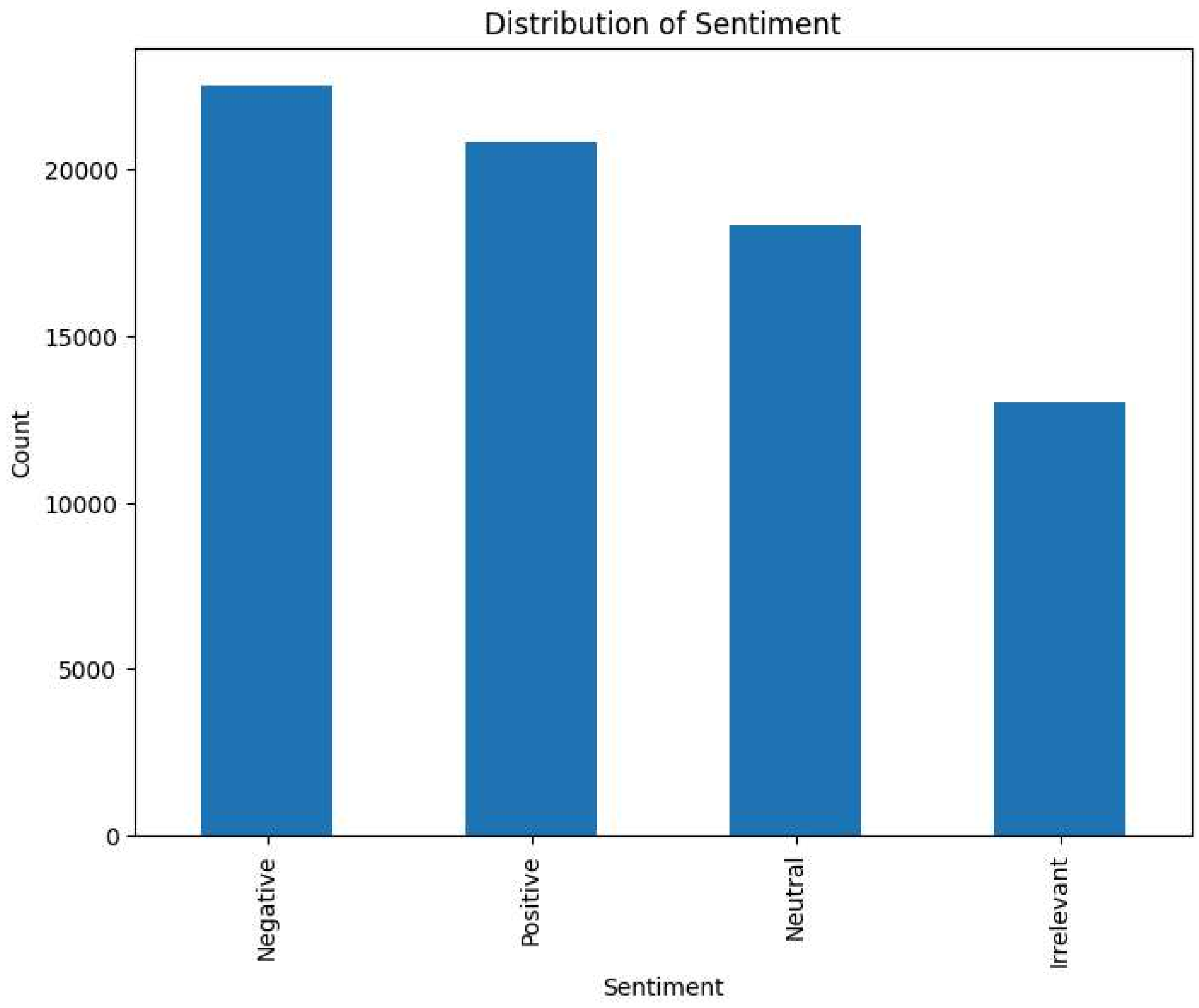
Explore the loaded data to understand its structure, including column names, data types, and the presence of missing values. Determine the distribution of the target variable ("Sentiment").

Reasoning: Analyze the data types, missing values, and the distribution of the target variable.

```
In [2]: # Display data types of each column
        print(df.dtypes)

        # Check for missing values
```

```python
print(df.isnull().sum())

# Analyze the distribution of the target variable
print(df['Positive'].value_counts())

import matplotlib.pyplot as plt
plt.figure(figsize=(8, 6))
df['Positive'].value_counts().plot(kind='bar')
plt.title('Distribution of Sentiment')
plt.xlabel('Sentiment')
plt.ylabel('Count')
plt.show()
```

```
2401                                                       int64
Borderlands                                               object
Positive                                                  object
im getting on borderlands and i will murder you all ,    object
dtype: object
2401                                                        0
Borderlands                                                 0
Positive                                                    0
im getting on borderlands and i will murder you all ,    686
dtype: int64
Positive
Negative      22542
Positive      20831
Neutral       18318
Irrelevant    12990
Name: count, dtype: int64
```



Data cleaning

Subtask:

Clean the data by handling missing values and removing duplicate rows.

Reasoning: Clean the data by handling missing values and removing duplicate rows as per the instructions.

In [3]:
```
# Address missing values in the tweet text column.
# Removing rows with missing tweet text as they are not useful for sentiment
original_rows = len(df)
df.dropna(subset=['im getting on borderlands and i will murder you all ,'],
dropped_rows = original_rows - len(df)
print(f"Dropped {dropped_rows} rows with missing tweet text.")

# Check for and remove duplicate rows.
duplicate_rows = df.duplicated().sum()
df.drop_duplicates(inplace=True)
print(f"Removed {duplicate_rows} duplicate rows.")
```

```
# Display first 5 rows of the cleaned DataFrame.
display(df.head())
```

```
Dropped 686 rows with missing tweet text.
Removed 2340 duplicate rows.
```

	2401	Borderlands	Positive	im getting on borderlands and i will murder you all ,
0	2401	Borderlands	Positive	I am coming to the borders and I will kill you...
1	2401	Borderlands	Positive	im getting on borderlands and i will kill you ...
2	2401	Borderlands	Positive	im coming on borderlands and i will murder you...
3	2401	Borderlands	Positive	im getting on borderlands 2 and i will murder ...
4	2401	Borderlands	Positive	im getting into borderlands and i can murder y...

Data preparation

Subtask:

Prepare the text data for analysis.

Reasoning: Clean the text data by converting to lowercase, removing punctuation, stop words, and performing stemming.

In [5]:
```
import nltk
nltk.download('punkt_tab')

import re
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer

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

def clean_text(text):
    text = str(text).lower()
    text = re.sub(r'[^\w\s]', '', text)  # Remove punctuation
    words = nltk.word_tokenize(text)
    words = [w for w in words if not w in stop_words]  # Remove stop words
    words = [stemmer.stem(w) for w in words]  # Stemming
    return ' '.join(words)

df['cleaned_text'] = df['im getting on borderlands and i will murder you all
display(df.head())
```

```
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
```

	2401	Borderlands	Positive	im getting on borderlands and i will murder you all ,	cleaned_text
0	2401	Borderlands	Positive	I am coming to the borders and I will kill you...	come border kill
1	2401	Borderlands	Positive	im getting on borderlands and i will kill you ...	im get borderland kill
2	2401	Borderlands	Positive	im coming on borderlands and i will murder you...	im come borderland murder
3	2401	Borderlands	Positive	im getting on borderlands 2 and i will murder ...	im get borderland 2 murder
4	2401	Borderlands	Positive	im getting into borderlands and i can murder y...	im get borderland murder

Feature engineering

Subtask:

Create TF-IDF features from the cleaned text data.

Reasoning: Create TF-IDF features from the cleaned text data using TfidfVectorizer.

In [6]:
```python
from sklearn.feature_extraction.text import TfidfVectorizer

# Initialize TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer(max_features=5000, max_df=0.95, min_df=2)

# Fit and transform the cleaned text
tfidf_matrix = tfidf_vectorizer.fit_transform(df['cleaned_text'])

# Convert the sparse matrix to a dense array
tfidf_matrix = tfidf_matrix.toarray()
```

Data splitting

Subtask:

Split the data into training and testing sets.

Reasoning: Split the data into training and testing sets using train_test_split.

In [7]:
```python
from sklearn.model_selection import train_test_split

# Assuming 'Positive' is the target variable and 'tfidf_matrix' is already c
```

```
X_train, X_test, y_train, y_test = train_test_split(
    tfidf_matrix, df['Positive'], test_size=0.2, random_state=42, stratify=d
)
```

Model training

Subtask:

Train a Logistic Regression model.

Reasoning: Train a Logistic Regression model using the training data.

In [8]:
```
from sklearn.linear_model import LogisticRegression

# Initialize and train the Logistic Regression model
model = LogisticRegression(max_iter=1000, solver='saga')
model.fit(X_train, y_train)
```

Out[8]:
```
▾ LogisticRegression
LogisticRegression(max_iter=1000, solver='saga')
```

Model evaluation

Subtask:

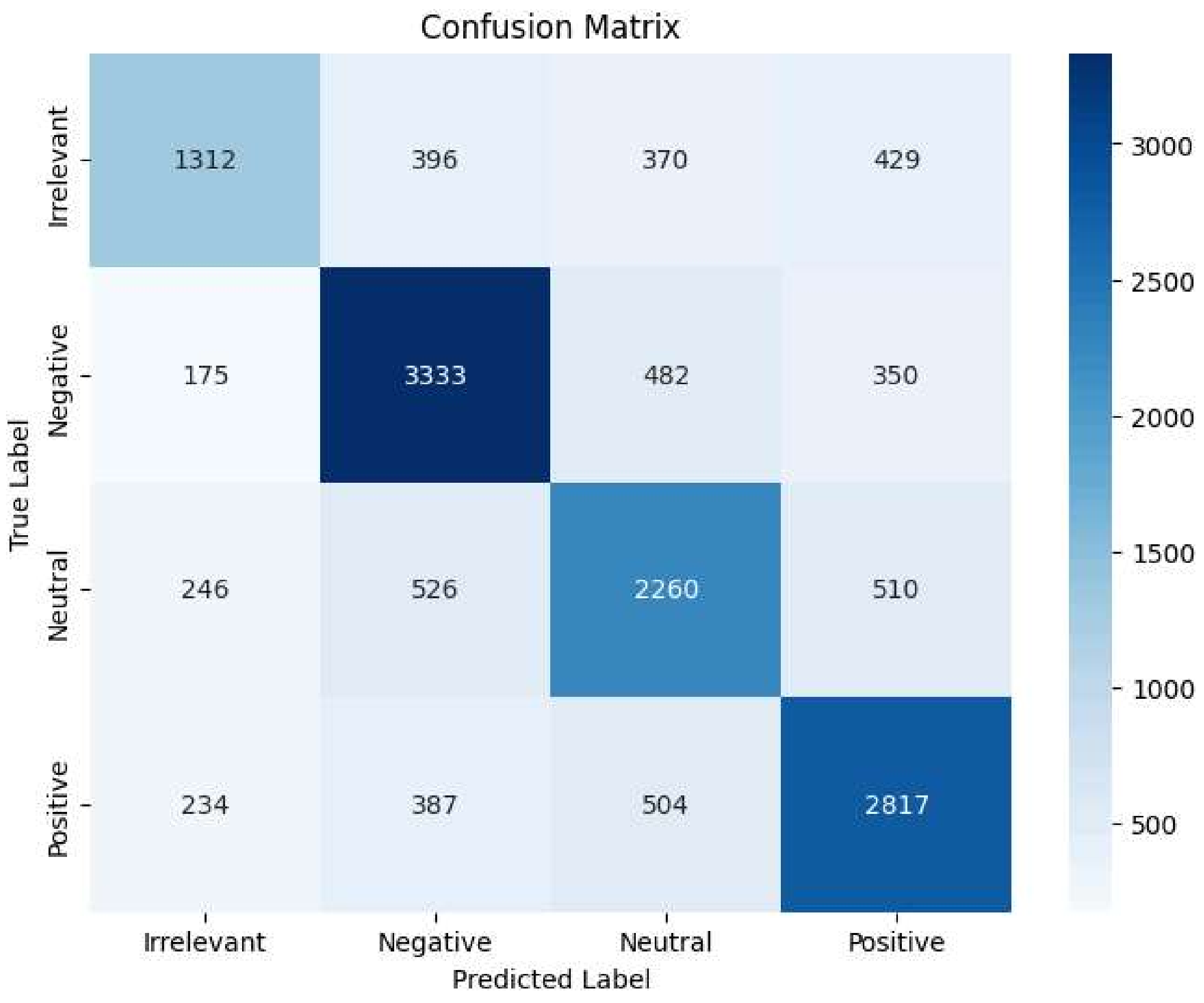
Evaluate the trained Logistic Regression model.

Reasoning: Evaluate the trained model using the test data and visualize the confusion matrix.

In [9]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f
import seaborn as sns
import matplotlib.pyplot as plt

# Predict on the test data
y_pred = model.predict(X_test)

# Calculate evaluation metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted', zero_divisio
recall = recall_score(y_test, y_pred, average='weighted', zero_division=0)
f1 = f1_score(y_test, y_pred, average='weighted', zero_division=0)

print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1-score: {f1:.4f}")
```

```
# Compute the confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Visualize the confusion matrix using a heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
            xticklabels=model.classes_, yticklabels=model.classes_)
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```

```
Accuracy: 0.6784
Precision: 0.6773
Recall: 0.6784
F1-score: 0.6760
```


Confusion Matrix
True Label
Irrelevant
Negative
Neutral
Positive
Irrelevant
Negative
Neutral
Positive
Predicted Label