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

In [2]:
```
df = pd.read_csv('twitter_training.csv', header=None)
```

In [3]:
```
df.head(5)
```

Out[3]:

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

In [4]:
```
df.shape
```

Out[4]:
```
(74682, 4)
```

In [5]:
```
df[0].value_counts()
```

Out[5]:
```
0
2401    6
6164    6
6141    6
6142    6
6143    6
       ..
4678    6
4679    6
4680    6
4681    6
9200    6
Name: count, Length: 12447, dtype: int64
```

In [6]:
```
# prompt: i want to rename the headers in the sdf

import pandas as pd

# Assuming your dataframe is named 'df'
# Rename the headers
new_headers = {
    0: 'id',
    1: 'game',
    2: 'sentiment',
    3: 'tweet'
}
df = df.rename(columns=new_headers)

# Display the first few rows to verify the change
df.head()
```

Out[6]:

	id	game	sentiment	tweet
0	2401	Borderlands	Positive	im getting on borderlands and i will murder yo...
1	2401	Borderlands	Positive	I am coming to the borders and I will kill you...
2	2401	Borderlands	Positive	im getting on borderlands and i will kill you ...
3	2401	Borderlands	Positive	im coming on borderlands and i will murder you...
4	2401	Borderlands	Positive	im getting on borderlands 2 and i will murder ...

In [7]:

```
# Assuming your dataframe is named 'df' and the column you want to label is 'sen
def label_sentiment(sentiment):
    if sentiment == 'Positive':
        return 1
    elif sentiment == 'Negative':
        return 0
    elif sentiment == 'Neutral':
        return 2
    elif sentiment == 'Irrelevant':
        return 3
df['sentiment_label'] = df['sentiment'].apply(label_sentiment)

# Display the first few rows to verify the change
print(df.head())
```

```
     id         game sentiment  \
0  2401  Borderlands  Positive
1  2401  Borderlands  Positive
2  2401  Borderlands  Positive
3  2401  Borderlands  Positive
4  2401  Borderlands  Positive

                                               tweet  sentiment_label
0  im getting on borderlands and i will murder yo...                1
1  I am coming to the borders and I will kill you...                1
2  im getting on borderlands and i will kill you ...                1
3  im coming on borderlands and i will murder you...                1
4  im getting on borderlands 2 and i will murder ...                1
```

In [8]:

```
df['sentiment'].value_counts()
```

Out[8]:

```
sentiment
Negative      22542
Positive      20832
Neutral       18318
Irrelevant    12990
Name: count, dtype: int64
```

In [9]:

```
df['sentiment_label'].value_counts()
```

Out[9]:

```
sentiment_label
0    22542
1    20832
2    18318
3    12990
Name: count, dtype: int64
```

In [10]:

```
df_train = df[['tweet','sentiment_label']]
```

In [11]:

```
df_train.head()
```

Out[11]:

	tweet	sentiment_label
0	im getting on borderlands and i will murder yo...	1
1	I am coming to the borders and I will kill you...	1
2	im getting on borderlands and i will kill you ...	1
3	im coming on borderlands and i will murder you...	1
4	im getting on borderlands 2 and i will murder ...	1

In [12]:

```
# prompt: from the df_train dataframe, format the tweet column and treat it for

import re
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords

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

def format_tweet(tweet):
    tweet = re.sub(r'http\S+', '', str(tweet)) # remove URLs
    tweet = re.sub(r'@[A-Za-z0-9]+', '', tweet) # remove mentions
    tweet = re.sub(r'#', '', tweet) # remove hashtags
    tweet = re.sub(r'[^\w\s]', '', tweet) #remove punctuation

    tweet = ' '.join(word for word in tweet.split() if word.lower() not in stop_
    tweet = tweet.strip().lower() #strip and lowercase
    return tweet


df_train['tweet'] = df_train['tweet'].apply(format_tweet)
df_train.head()
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\pcz\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
C:\Users\pcz\AppData\Local\Temp\ipykernel_720\1716538142.py:21: SettingWithCopyWa
rning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_train['tweet'] = df_train['tweet'].apply(format_tweet)
```

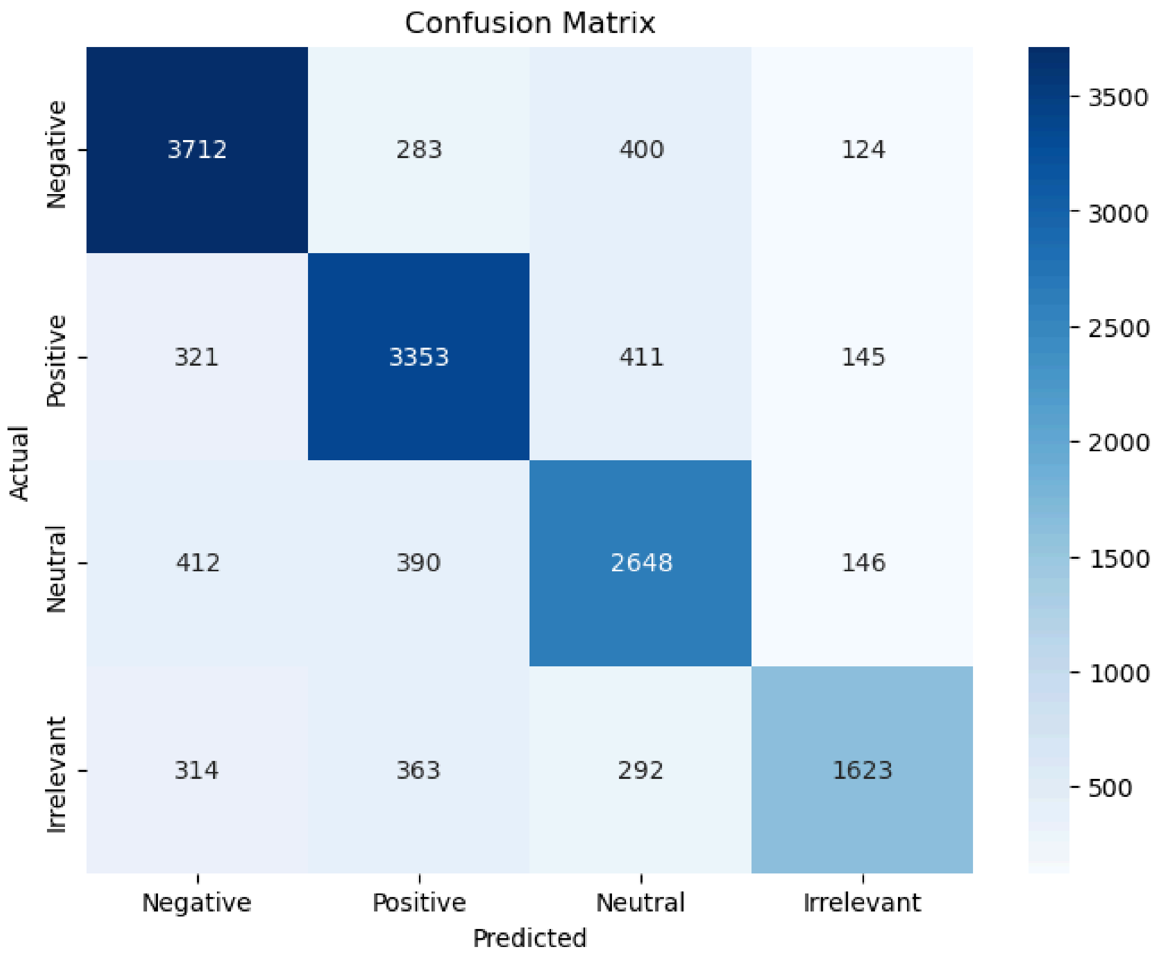
Out[12]:

	tweet	sentiment_label
0	im getting borderlands murder	1
1	coming borders kill	1
2	im getting borderlands kill	1
3	im coming borderlands murder	1
4	im getting borderlands 2 murder	1

In [13]:

```
# prompt: train a sentiment analysis model on the df-train dataframe with target

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(df_train['tweet'].values.ast

# Create TF-IDF vectors
vectorizer = TfidfVectorizer()
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)

# Train a Logistic Regression model
model = LogisticRegression()
model.fit(X_train_vec, y_train)

# Predict on the test set
y_pred = model.predict(X_test_vec)

# Create and display the confusion matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Negative', 'Positive', 'Neutral', 'Irrelevant'],
            yticklabels=['Negative', 'Positive', 'Neutral', 'Irrelevant'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

```
C:\Users\pcz\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py:469: C
onvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```



In [14]:

```python
# prompt: evaluate the performance of the above created sentiment analysis model

from sklearn.metrics import classification_report, accuracy_score

# ... (Your existing code) ...

# Predict on the test set
y_pred = model.predict(X_test_vec)

# Evaluate the model
print(classification_report(y_test, y_pred, target_names=['Negative', 'Positive'
print(f"Accuracy: {accuracy_score(y_test, y_pred)}")
```

```
              precision    recall  f1-score   support

    Negative       0.78      0.82      0.80      4519
    Positive       0.76      0.79      0.78      4230
     Neutral       0.71      0.74      0.72      3596
  Irrelevant       0.80      0.63      0.70      2592

    accuracy                           0.76     14937
   macro avg       0.76      0.74      0.75     14937
weighted avg       0.76      0.76      0.76     14937

Accuracy: 0.7589208006962577
```

In [15]:

```python
df_val = pd.read_csv('twitter_validation.csv', header=None)
```

In [16]:

```python
df_val_test = df_val[3]
```

In [17]:
```
# prompt: use the above created model to identify sentiments on the df_val_test

# Assuming df_val_test contains the tweets you want to analyze

# Preprocess the tweets in df_val_test
df_val_test = df_val_test.apply(format_tweet)

# Transform the preprocessed tweets into TF-IDF vectors
df_val_test_vec = vectorizer.transform(df_val_test)

# Predict sentiments using the trained model
predicted_sentiments = model.predict(df_val_test_vec)

# Add the predicted sentiments to the dataframe (optional)
df_val['predicted_sentiment'] = predicted_sentiments

#Example to map back the numbers to sentiments
def sentiment_to_label(label):
    if label == 1:
        return 'Positive'
    elif label == 0:
        return 'Negative'
    elif label == 2:
        return 'Neutral'
    elif label == 3:
        return 'Irrelevant'
    else:
        return 'Unknown'
df_val['predicted_sentiment_label'] = df_val['predicted_sentiment'].apply(sentim
print(df_val.head())
```

```
      0          1           2  \
0  3364   Facebook  Irrelevant
1   352     Amazon     Neutral
2  8312  Microsoft    Negative
3  4371      CS-GO    Negative
4  4433     Google     Neutral

                                                   3 predicted_sentiment  \
0  I mentioned on Facebook that I was struggling ...                   3
1  BBC News - Amazon boss Jeff Bezos rejects clai...                   2
2  @Microsoft Why do I pay for WORD when it funct...                   0
3  CSGO matchmaking is so full of closet hacking,...                   0
4  Now the President is slapping Americans in the...                   2

  predicted_sentiment_label
0                   Unknown
1                   Unknown
2                   Unknown
3                   Unknown
4                   Unknown
```

In [18]:
```
df_pred = df_val['predicted_sentiment']
```

In [19]:
```
# prompt: export df_pred as a csv file

df_pred.to_csv('predicted_sentiments.csv', index=False)
```

In [21]:

```
predicted_sentiments = pd.read_csv('predicted_sentiments.csv', header=None)
predicted_sentiments.head(5)
```

Out[21]:

```
                     0
0  predicted_sentiment
1                    3
2                    2
3                    0
4                    0
```

In [ ]: