Prodigy_DS_Task_04🌐 🔍 📝

Task 4. Analyze and visualize sentiment patterns in social media data to understand public opinion and attitudes towards specific topics or brands..

INTRODUCTION :

Sentiment analysis is the computational task of automatically determining what feelings a writer is expressing in text. Sentiment is often framed as a binary distinction (positive vs. negative), but it can also be a more fine-grained, like identifying the specific emotion an author is expressing (like fear, joy or anger).

Sentiment analysis is used for many applications, especially in business intelligence. Some examples of applications for sentiment analysis include:

Analyzing the social media discussion around a certain topic

Evaluating survey responses

Determining whether product reviews are positive or negative

Sentiment analysis is not perfect, and as with any automatic analysis of language, you will have errors in your results. It also cannot tell you why a writer is feeling a certain way. However, it can be useful to quickly summarize some qualities of text, especially if you have so much text that a human reader cannot analyze all of it.

Data Description 🔍 📊 💡

This is an entity-level sentiment analysis dataset of twitter. Given a message and an entity, the task is to judge the sentiment of the message about the entity. There are three classes in this dataset: Positive, Negative and Neutral. We regard messages that are not relevant to the entity (i.e. Irrelevant) as Neutral.

Import Libraries 📚

In [16]:
```
import warnings
warnings.filterwarnings("ignore")

import numpy as np
import pandas as pd
import re

from wordcloud import WordCloud
import matplotlib.pyplot as plt
from plotly.subplots import make_subplots
import plotly.express as px
import plotly.graph_objects as go
import plotly.figure_factory as ff

from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.model_selection import GridSearchCV
```

Loading data Set and Preprocessing Of Data

In [17]:
```
train_data = pd.read_csv("C:\\Users\\Ujjaval Raj\\Downloads\\twitter_training.csv")
validation_data = pd.read_csv("C:\\Users\\Ujjaval Raj\\Downloads\\twitter_validatio

train_head = train_data.head()
validation_head = validation_data.head()

print(train_head)
print(validation_head)
```

```
   2401  Borderlands  Positive  \
0  2401  Borderlands  Positive
1  2401  Borderlands  Positive
2  2401  Borderlands  Positive
3  2401  Borderlands  Positive
4  2401  Borderlands  Positive

  im getting on borderlands and i will murder you all ,
0  I am coming to the borders and I will kill you...
1  im getting on borderlands and i will kill you ...
2  im coming on borderlands and i will murder you...
3  im getting on borderlands 2 and i will murder ...
4  im getting into borderlands and i can murder y...
   3364   Facebook Irrelevant  \
0   352     Amazon    Neutral
1  8312  Microsoft   Negative
2  4371      CS-GO   Negative
3  4433     Google    Neutral
4  6273       FIFA   Negative

  I mentioned on Facebook that I was struggling for motivation to go for a run the
other day, which has been translated by Tom’s great auntie as ‘Hayley can’t get out
of bed’ and told to his grandma, who now thinks I’m a lazy, terrible person 🤣
0  BBC News - Amazon boss Jeff Bezos rejects clai...
1  @Microsoft Why do I pay for WORD when it funct...
2  CSGO matchmaking is so full of closet hacking,...
3  Now the President is slapping Americans in the...
4  Hi @EAHelp I’ve had Madeleine McCann in my cel...
```

Checking for null values and duplicates

In [18]:

```
missing_train = train_data.isnull().sum()
missing_validation = validation_data.isnull().sum()

duplicates_train = train_data.duplicated().sum()
duplicates_validation = validation_data.duplicated().sum()

print(missing_train)
print(missing_validation)
print(duplicates_train)
print(duplicates_validation)
```

```
2401                                                        0
Borderlands                                                 0
Positive                                                    0
im getting on borderlands and i will murder you all ,     686
dtype: int64
3364
0
Facebook
0
Irrelevant
0
I mentioned on Facebook that I was struggling for motivation to go for a run the ot
her day, which has been translated by Tom’s great auntie as ‘Hayley can’t get out o
f bed’ and told to his grandma, who now thinks I’m a lazy, terrible person 🤣      0
dtype: int64
2700
0
```

Checking the distribution of sentiment labels and the number of unique entities

In [19]:
```
train_sentiment_distribution = train_data.iloc[:, 2].value_counts()
validation_sentiment_distribution = validation_data.iloc[:, 2].value_counts()


unique_entities_train = train_data.iloc[:, 1].nunique()

print(train_sentiment_distribution)
print(validation_sentiment_distribution)
print(unique_entities_train)
```

```
Positive
Negative      22542
Positive      20831
Neutral       18318
Irrelevant    12990
Name: count, dtype: int64
Irrelevant
Neutral       285
Positive      277
Negative      266
Irrelevant    171
Name: count, dtype: int64
32
```

In [20]:
```
train_data.info()
validation_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 74681 entries, 0 to 74680
Data columns (total 4 columns):
 #   Column                                                  Non-Null Count  Dtype 
---  ------                                                  --------------  ----- 
 0   2401                                                    74681 non-null  int64 
 1   Borderlands                                             74681 non-null  object
 2   Positive                                                74681 non-null  object
 3   im getting on borderlands and i will murder you all ,  73995 non-null  object
dtypes: int64(1), object(3)
memory usage: 2.3+ MB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 999 entries, 0 to 998
Data columns (total 4 columns):
 #   Column
Non-Null Count  Dtype 
---  ------
--------------  ----- 
 0   3364
999 non-null    int64 
 1   Facebook
999 non-null    object
 2   Irrelevant
999 non-null    object
 3   I mentioned on Facebook that I was struggling for motivation to go for a run t
he other day, which has been translated by Tom’s great auntie as ‘Hayley can’t get
out of bed’ and told to his grandma, who now thinks I’m a lazy, terrible person 🤣
999 non-null    object
dtypes: int64(1), object(3)
memory usage: 31.3+ KB
```

Remove duplicate rows from the training set

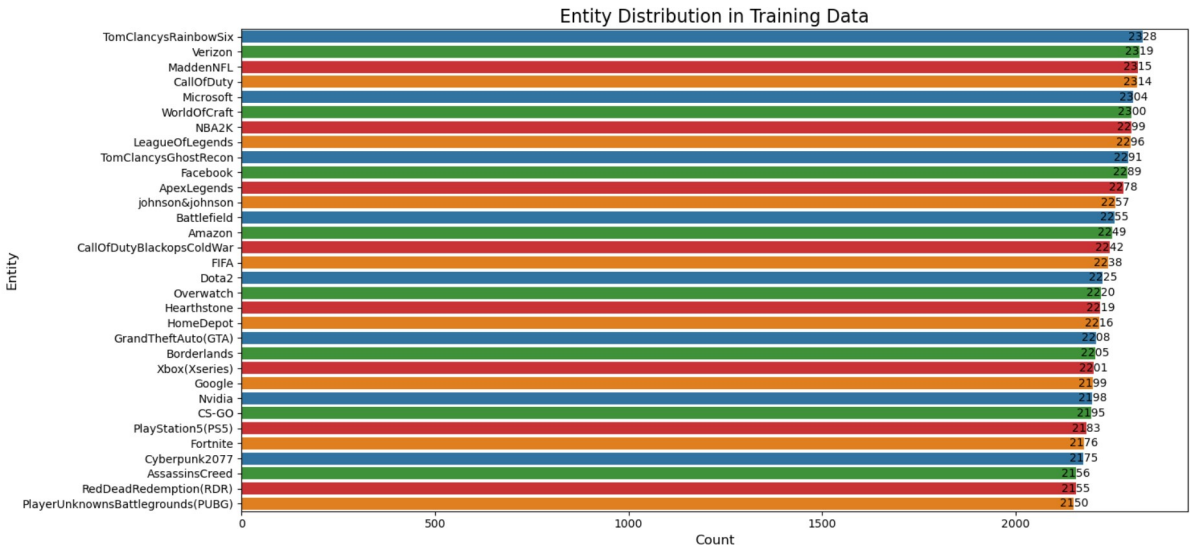
In [21]:
```
train_data_cleaned = train_data.drop_duplicates()
train_data_cleaned = train_data_cleaned.dropna(subset=[train_data.columns[3]])

remaining_duplicates_train = train_data_cleaned.duplicated().sum()
remaining_missing_train = train_data_cleaned.isnull().sum()

remaining_duplicates_train
remaining_missing_train
```

Out[21]:
```
2401                                                    0
Borderlands                                             0
Positive                                                0
im getting on borderlands and i will murder you all ,   0
dtype: int64
```

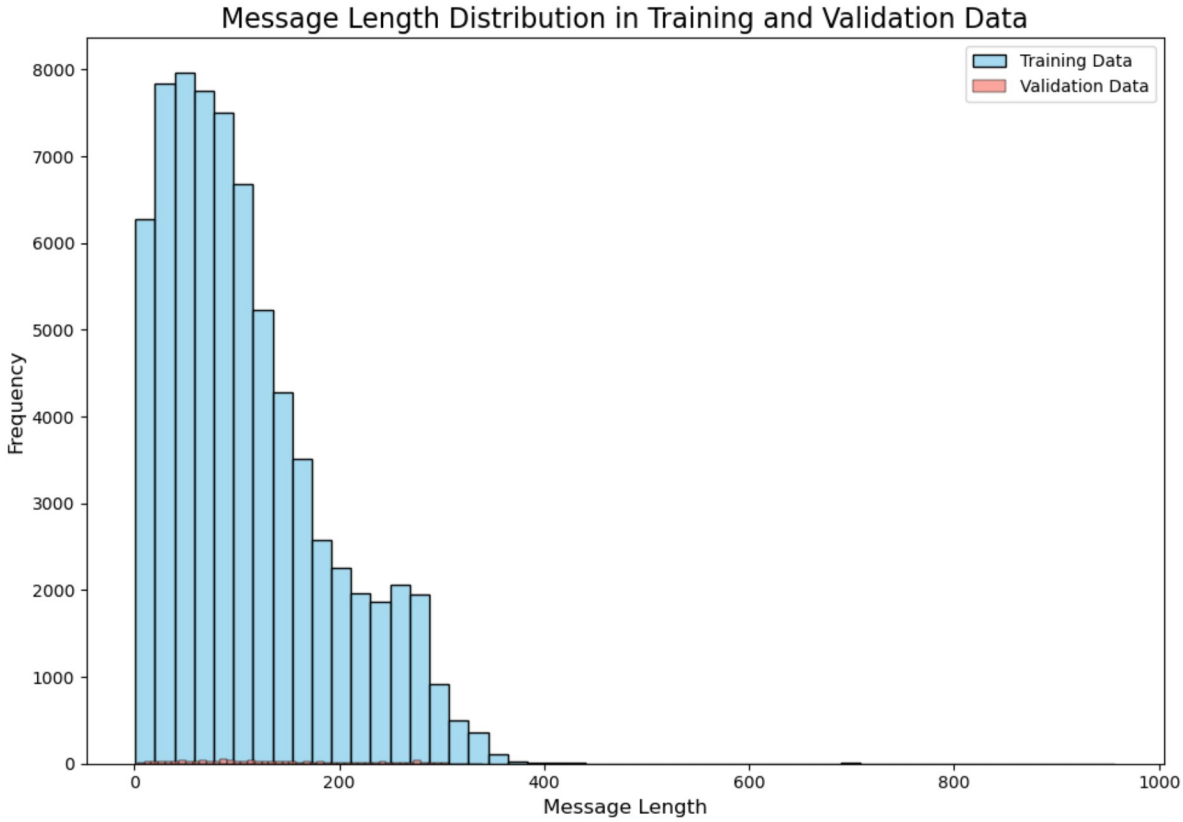
Exploratory Data Analysis

In [34]:
```
colors = sns.color_palette("pastel")

fig, ax = plt.subplots(1, 2, figsize=(15, 5), gridspec_kw={'width_ratios': [1, 1]})

sns.countplot(data=train_data_cleaned, x=train_data_cleaned.columns[2],
              order=['Positive', 'Negative', 'Neutral', 'Irrelevant'],
              palette=colors, ax=ax[0])
ax[0].set_title('Sentiment Distribution in Training Data')
ax[0].set_ylabel('Count')
ax[0].set_xlabel('Sentiment')

sns.countplot(data=validation_data, x=validation_data.columns[2],
              order=['Positive', 'Negative', 'Neutral', 'Irrelevant'],
              palette=colors, ax=ax[1])
ax[1].set_title('Sentiment Distribution in Validation Data')
ax[1].set_ylabel('Count')
ax[1].set_xlabel('Sentiment')

plt.tight_layout(pad=3)

plt.suptitle('Sentiment Distribution in Training and Validation Data', fontsize=16)
plt.show()
```

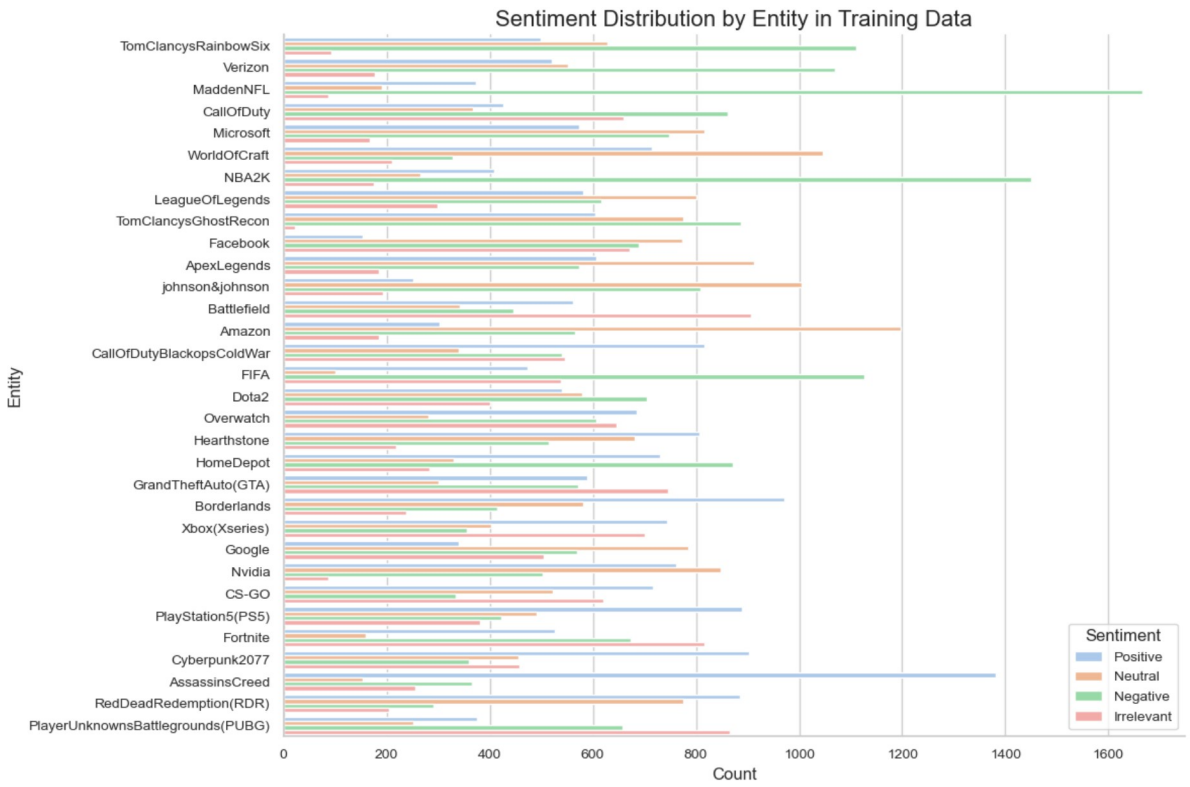
In [23]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Define a custom color palette with four distinct colors
custom_palette = ["#1f78b4", "#33a02c", "#e31a1c", "#ff7f00"]

plt.figure(figsize=(15, 7))

# Plot entity distribution for training data
sns.countplot(data=train_data_cleaned, y=train_data_cleaned.columns[1],
              order=train_data_cleaned[train_data_cleaned.columns[1]].value_counts(
              palette=custom_palette)

plt.title('Entity Distribution in Training Data', fontsize=16)
plt.xlabel('Count', fontsize=12)
plt.ylabel('Entity', fontsize=12)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)

# Add count annotations on the bars
for p in plt.gca().patches:
    plt.gca().annotate(f'{int(p.get_width())}',
                       (p.get_width() + 0.1, p.get_y() + p.get_height() / 2),
                       ha='center', va='center', fontsize=10, color='black')

plt.tight_layout()
plt.show()
```



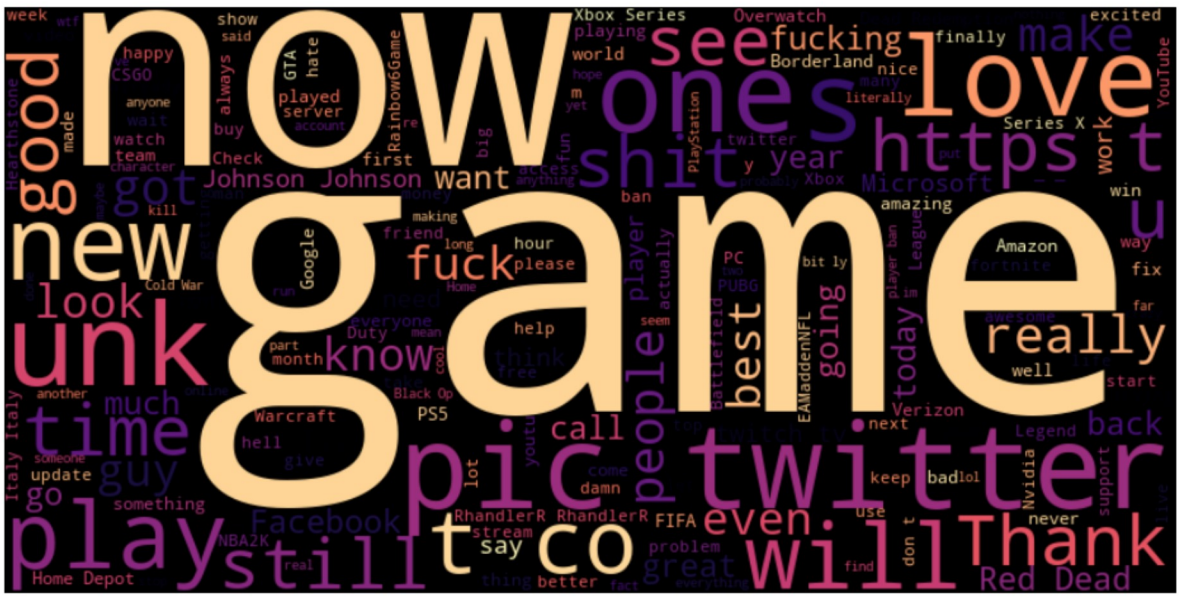
In [24]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Calculate the length of each message
train_data_cleaned['message_length'] = train_data_cleaned[train_data_cleaned.column
validation_data['message_length'] = validation_data[validation_data.columns[3]].app

# Set up the plot
plt.figure(figsize=(10, 7))

# Plot message length distribution for training data
sns.histplot(train_data_cleaned['message_length'], bins=50, color='skyblue', label=

# Plot message length distribution for validation data on top of the first plot
sns.histplot(validation_data['message_length'], bins=50, color='salmon', label='Val

plt.title('Message Length Distribution in Training and Validation Data', fontsize=1
plt.xlabel('Message Length', fontsize=12)
plt.ylabel('Frequency', fontsize=12)
plt.legend()

plt.tight_layout()
plt.show()
```



In [25]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Set Seaborn style to whitegrid
sns.set(style="whitegrid")

# Set up the plot
plt.figure(figsize=(12, 8))

# Basic count plot with a pleasing color palette
sns.countplot(data=train_data_cleaned, y=train_data_cleaned.columns[1],
              hue=train_data_cleaned.columns[2],
              order=train_data_cleaned[train_data_cleaned.columns[1]].value_counts(
              palette="pastel")

plt.title('Sentiment Distribution by Entity in Training Data', fontsize=16)
plt.xlabel('Count', fontsize=12)
plt.ylabel('Entity', fontsize=12)
plt.legend(title='Sentiment', fontsize=10)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)

# Remove top and right spines for cleaner look
sns.despine()

plt.tight_layout()
plt.show()
```



In [26]:
```
from wordcloud import WordCloud
import matplotlib.pyplot as plt

# Generate the word cloud for the entire training dataset
all_text = " ".join(tweet for tweet in train_data_cleaned[train_data_cleaned.column

# Define a custom color scheme for dark theme
color_map_dark = plt.cm.magma_r

# Generate the WordCloud with custom parameters for dark theme
wordcloud_all_dark = WordCloud(
    background_color='black',
    width=800,
    height=400,
    max_words=200,
    colormap=color_map_dark,
    contour_color='white',
    contour_width=1,
    random_state=42
).generate(all_text)

# Plot the Word Cloud with a dark theme
plt.figure(figsize=(12, 6))
plt.imshow(wordcloud_all_dark, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud for All Tweets in Training Data (Dark Theme)', fontsize=16, c
plt.show()
```



In [27]:
```
# Initialize sentiment categories
sentiments = ['Positive', 'Negative', 'Neutral', 'Irrelevant']

# Set up the plots
fig, axs = plt.subplots(2, 2, figsize=(15, 10))

# Generate and plot word clouds for each sentiment
for sentiment, ax in zip(sentiments, axs.ravel()):
    sentiment_text = " ".join(tweet for tweet in train_data_cleaned[train_data_clea
    wordcloud_sentiment = WordCloud(background_color='black', width=400, height=200

    ax.imshow(wordcloud_sentiment, interpolation='bilinear')
    ax.axis('off')
    ax.set_title(f'Word Cloud for {sentiment} Sentiment')

plt.tight_layout()
plt.show()
```

In [28]:
```
from sklearn.feature_extraction.text import CountVectorizer

# Alternative function to preprocess text without lemmatization or NLTK stopwords
def preprocess_text_simplified(text):
    # Convert to lowercase
    text = text.lower()
    # Simple tokenization using split (without relying on NLTK)
    tokens = text.split()
    # Remove special characters and numbers
    tokens = [token for token in tokens if token.isalpha()]
    return " ".join(tokens)

# Apply simplified preprocessing to training data
train_data_cleaned['processed_message_simplified'] = train_data_cleaned[train_data_

# Extract most frequent terms using CountVectorizer with simplified preprocessing
vectorizer_simplified = CountVectorizer(max_features=20)
X_simplified = vectorizer_simplified.fit_transform(train_data_cleaned['processed_me
frequent_terms_simplified = vectorizer_simplified.get_feature_names_out()

frequent_terms_simplified
```

Out[28]:
```
array(['and', 'but', 'for', 'game', 'have', 'in', 'is', 'it', 'just',
       'my', 'not', 'of', 'on', 'so', 'that', 'the', 'this', 'to', 'with',
       'you'], dtype=object)
```

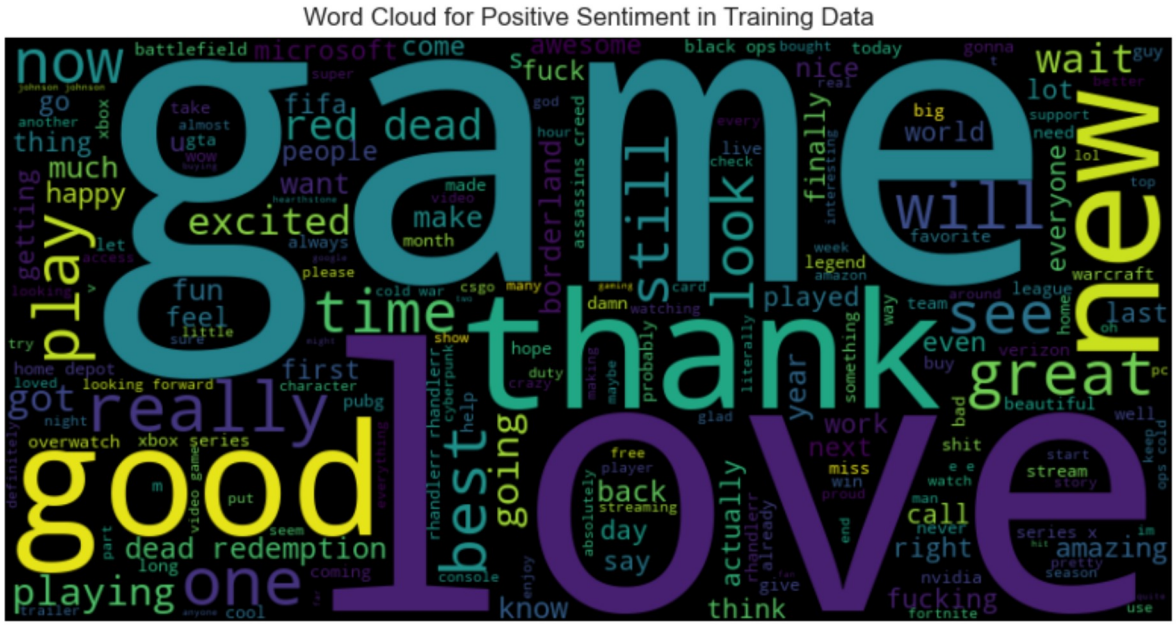
In [29]:
```
# Adjusting the labels: Convert "Irrelevant" labels to "Neutral"
train_data_cleaned[train_data_cleaned.columns[2]] = train_data_cleaned[train_data_c
validation_data[validation_data.columns[2]] = validation_data[validation_data.colum

# Check the updated sentiment distribution in the training and validation data
updated_train_sentiment_distribution = train_data_cleaned[train_data_cleaned.column
updated_validation_sentiment_distribution = validation_data[validation_data.columns

updated_train_sentiment_distribution, updated_validation_sentiment_distribution
```

Out[29]:
```
(Positive
 Neutral     30245
 Negative    21698
 Positive    19712
 Name: count, dtype: int64,
 Irrelevant
 Neutral     456
 Positive    277
 Negative    266
 Name: count, dtype: int64)
```

In [30]:
```
from sklearn.feature_extraction.text import TfidfVectorizer

# Initialize the TF-IDF vectorizer
tfidf_vectorizer = TfidfVectorizer(max_features=5000)  # Limiting to 5000 features

# Fit and transform the preprocessed text from the training data
X_train_tfidf = tfidf_vectorizer.fit_transform(train_data_cleaned['processed_messag

# Apply simplified preprocessing to the validation data
validation_data['processed_message_simplified'] = validation_data[validation_data.c

# Transform the preprocessed text from the validation data
X_validation_tfidf = tfidf_vectorizer.transform(validation_data['processed_message_

# Extract target labels for training and validation
y_train = train_data_cleaned[train_data_cleaned.columns[2]]
y_validation = validation_data[validation_data.columns[2]]

X_train_tfidf.shape, X_validation_tfidf.shape
```

Out[30]:
```
((71655, 5000), (999, 5000))
```

In [32]:
```
# Extract text for Positive sentiment from the training dataset
positive_text = " ".join(tweet for tweet in train_data_cleaned[train_data_cleaned[t

# Generate word cloud for Positive sentiment
wordcloud_positive = WordCloud(background_color='black', width=800, height=400).gen

# Plot the word cloud
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud_positive, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud for Positive Sentiment in Training Data')
plt.show()
```



In [33]:
```
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression

# Define the target variable for training and validation again
y_train = train_data_cleaned[train_data_cleaned.columns[2]]
y_validation = validation_data[validation_data.columns[2]]

# Create the ML pipeline with TF-IDF vectorization and Logistic Regression
pipeline = Pipeline([
    ('tfidf', TfidfVectorizer(max_features=5000)),
    ('classifier', LogisticRegression(solver='sag', multi_class='auto', max_iter=10
])

# Train the pipeline model using the training data
pipeline.fit(train_data_cleaned['processed_message_simplified'], y_train)

# Validate the model's performance on the validation dataset
validation_accuracy = pipeline.score(validation_data['processed_message_simplified'

validation_accuracy
```

Out[33]:
```
0.8048048048048048
```

Conclusion:

We have sucessfully Built a sentiment analysis of twitter data set and validated our model with 0.80 accuracy.

Thank You !: