Sentimental analysis of British Airways

Now, we are in the second phase of the project. In this phase our job is to collect the data we scraped and perform sentimental analysis of the reviews gathered using NLP techniques.

In [1]:
```
#Importing the dataset

import pandas as pd
bacr=pd.read_csv("C:\\Users\\sujoydutta\\Downloads\\british_airways_reviews.csv")
bacr.head()
```

Out[1]:

	review_text
0	✅Trip Verified| British Airways at its best....
1	✅Trip Verified| An excellent flight! Despite t...
2	✅Trip Verified| I recently traveled with Briti...
3	✅Trip Verified| My family and I were booked ...
4	Not Verified| We had to change from AA to BA ...

In [2]:
```
# Identifying rows with the ✅ symbol
bacr['Verified'] = bacr['review_text'].apply(lambda x: '✅' in x if '✅' in x else 'Fals


bacr.head()
```

Out[2]:

	review_text	Verified
0	✅Trip Verified| British Airways at its best....	True
1	✅Trip Verified| An excellent flight! Despite t...	True
2	✅Trip Verified| I recently traveled with Briti...	True
3	✅Trip Verified| My family and I were booked ...	True
4	Not Verified| We had to change from AA to BA ...	False

In [3]:
```
# Assign weights based on the 'Verified' column
import numpy as np

bacr['weightage'] = np.where(bacr['Verified'] == True, 1.5, 1.0)

bacr.head()
```

Out[3]:

	review_text	Verified	weightage
0	✅Trip Verified| British Airways at its best....	True	1.5
1	✅Trip Verified| An excellent flight! Despite t...	True	1.5
2	✅Trip Verified| I recently traveled with Briti...	True	1.5
3	✅Trip Verified| My family and I were booked ...	True	1.5
4	Not Verified| We had to change from AA to BA ...	False	1.0

In [4]:
```
#getting nlp packages
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
```

In [5]:
```
# Downloading VADER lexicon
nltk.download('vader_lexicon')
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\sujoydutta\AppData\Roaming\nltk_data...
```

Out[5]:
```
True
```

In [6]:
```
# Initializing the VADER sentiment analyzer
sia = SentimentIntensityAnalyzer()
```

In [7]:
```
# Function to calculate sentiment score
def calculate_sentiment(review):
    score = sia.polarity_scores(review)['compound']
    return score
```

In [9]:
```
# Applying function to get sentiment score
bacr['sentiment_score'] = bacr['review_text'].apply(calculate_sentiment)
bacr.head()
```

Out[9]:

	review_text	Verified	weightage	sentiment_score
0	✅Trip Verified| British Airways at its best....	True	1.5	0.9787
1	✅Trip Verified| An excellent flight! Despite t...	True	1.5	0.9975
2	✅Trip Verified| I recently traveled with Briti...	True	1.5	-0.8223
3	✅Trip Verified| My family and I were booked ...	True	1.5	-0.9041
4	Not Verified| We had to change from AA to BA ...	False	1.0	-0.5915

In [26]:
```
#getting consolidated score
bacr['sentiment_con_score']=bacr['sentiment_score']*bacr['weightage']
bacr.head()
```

Out[26]:

	review_text	Verified	weightage	sentiment_score	sentiment_category	sentiment_con_score
0	✅Trip Verified| British Airways at its best....	True	1.5	0.9787	Positive	1.46805
1	✅Trip Verified| An excellent flight! Despite t...	True	1.5	0.9975	Positive	1.49625
2	✅Trip Verified| I recently traveled with Briti...	True	1.5	-0.8223	Negative	-1.23345
3	✅Trip Verified| My family and I were booked ...	True	1.5	-0.9041	Negative	-1.35615
4	Not Verified| We had to change from AA to BA ...	False	1.0	-0.5915	Negative	-0.59150

In [36]:
```
#Seeing mean score value
bacr.sentiment_con_score.mean()
```

Out[36]:
```
0.080910475
```

In [33]:
```
# Function to categorize sentiment
def categorize_sentiment(score):
```

```
    if score > 0.8:
        return 'Positive'
    elif score < -0.8:
        return 'Negative'
    else:
        return 'Neutral'
```

In [34]:
```
# Applying the function to the sentiment_score column
bacr['sentiment_category'] = bacr['sentiment_con_score'].apply(categorize_sentiment)

bacr.head()
```

Out[34]:

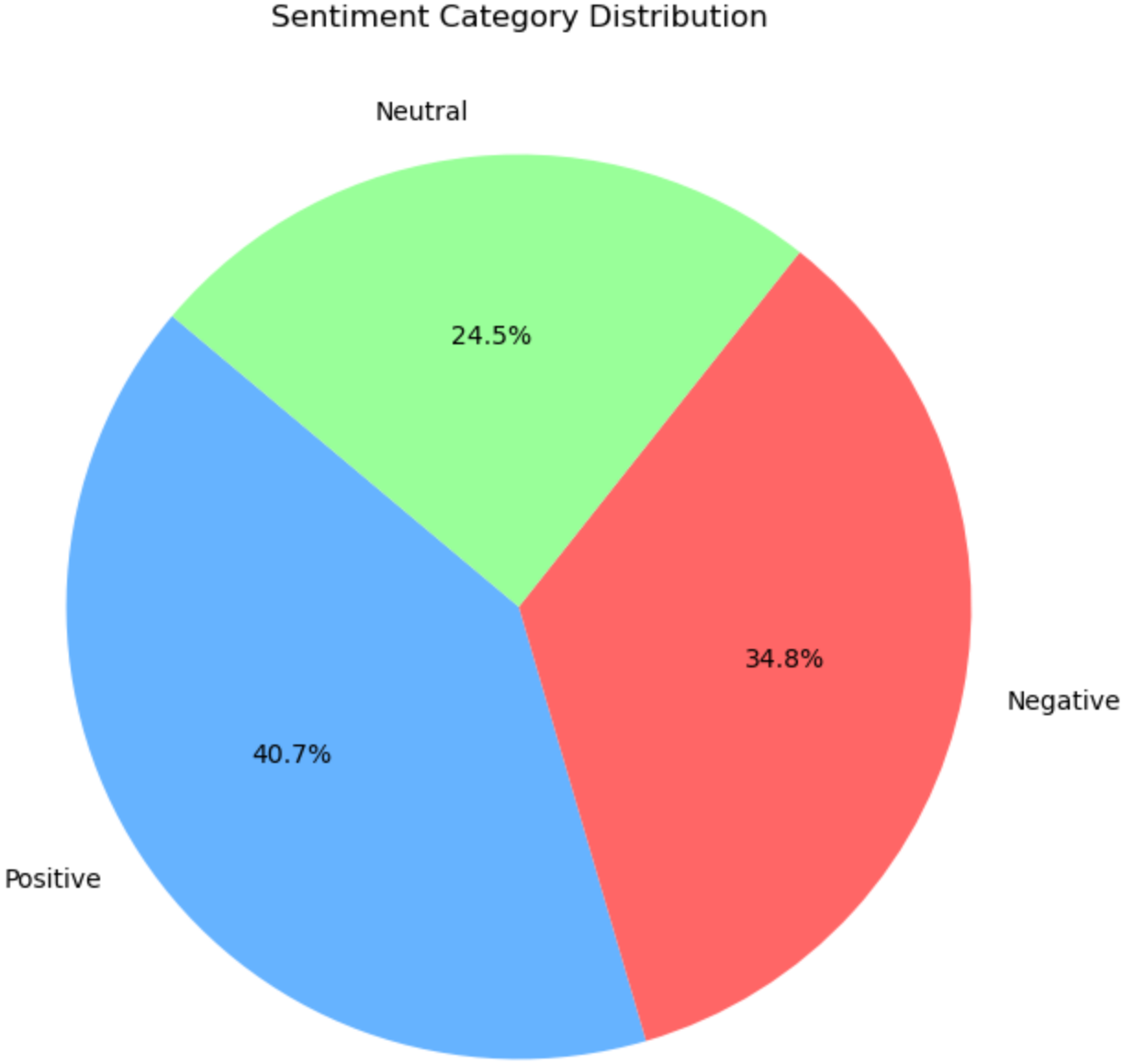
	review_text	Verified	weightage	sentiment_score	sentiment_category	sentiment_con_score
0	✅Trip Verified| British Airways at its best....	True	1.5	0.9787	Positive	1.46805
1	✅Trip Verified| An excellent flight! Despite t...	True	1.5	0.9975	Positive	1.49625
2	✅Trip Verified| I recently traveled with Briti...	True	1.5	-0.8223	Negative	-1.23345
3	✅Trip Verified| My family and I were booked ...	True	1.5	-0.9041	Negative	-1.35615
4	Not Verified| We had to change from AA to BA ...	False	1.0	-0.5915	Neutral	-0.59150

In [35]:
```
#seeing category distribution
bacr.sentiment_category.value_counts()
```

Out[35]:
```
sentiment_category
Positive    813
Negative    696
Neutral     491
Name: count, dtype: int64
```

In [44]:
```
# Seeing share of each sentiment type
sentiment_counts = bacr['sentiment_category'].value_counts()

plt.figure(figsize=(8, 8))
plt.pie(sentiment_counts, labels=sentiment_counts.index, autopct='%1.1f%%', colors=['#66
plt.title('Sentiment Category Distribution')
plt.show()
```



In [37]:
```
#getting packages for word cloud
import matplotlib.pyplot as plt
from wordcloud import WordCloud
from nltk.corpus import stopwords
```

In [38]:
```
# Getting stopwords corpus
nltk.download('stopwords')
stop_words = set(stopwords.words('english'))
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\sujoydutta\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [39]:
```
# Function to generate and display a word cloud
def generate_word_cloud(text, title):
    wordcloud = WordCloud(width=800, height=400, background_color='white', stopwords=sto

    plt.figure(figsize=(10, 5))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.title(title, fontsize=20)
    plt.axis('off')
    plt.show()
```

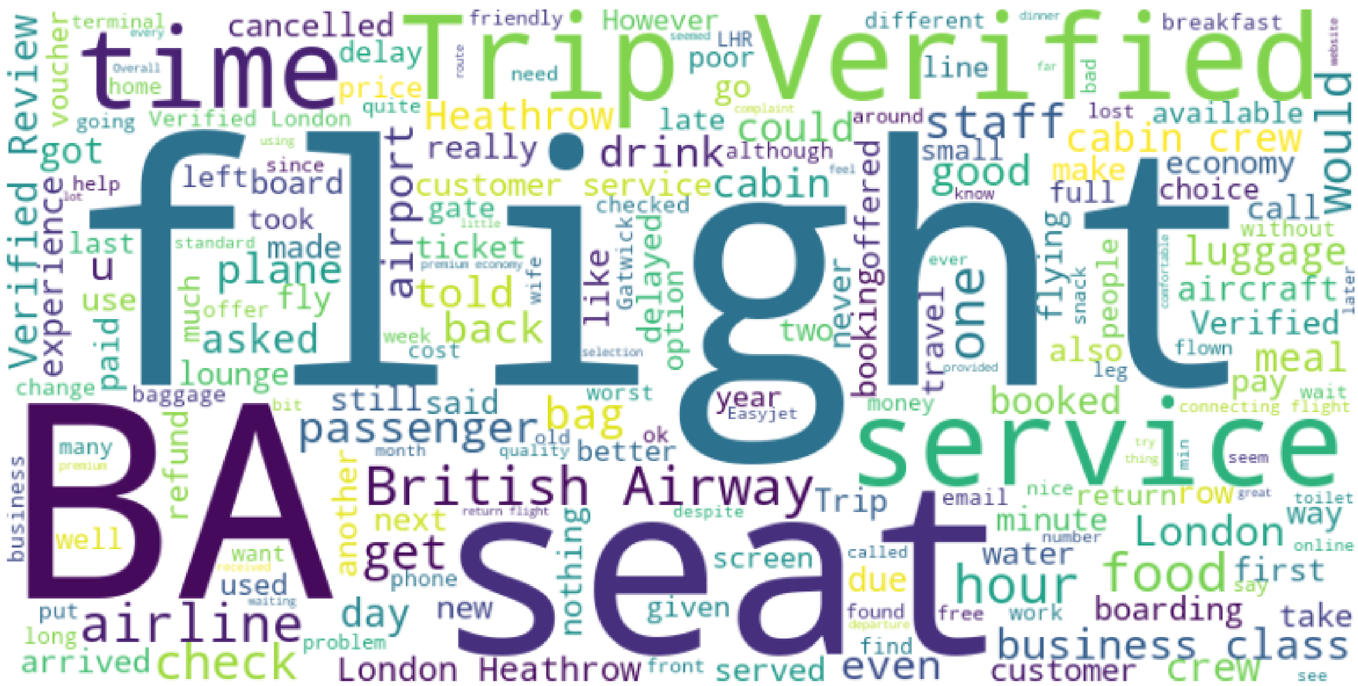
In [40]:
```
# Separating the reviews by sentiment category
positive_reviews = ' '.join(bacr[bacr['sentiment_category'] == 'Positive']['review_text'
neutral_reviews = ' '.join(bacr[bacr['sentiment_category'] == 'Neutral']['review_text'])
negative_reviews = ' '.join(bacr[bacr['sentiment_category'] == 'Negative']['review_text'
```

In [41]:
```
# Generating word clouds for positive reviews
generate_word_cloud(positive_reviews, 'Positive Reviews Word Cloud')
```

In [42]:
```
# Generating word clouds for negative reviews
generate_word_cloud(negative_reviews, 'Negative Reviews Word Cloud')
```

In [43]:
```
# Generating word clouds for neutral reviews
generate_word_cloud(neutral_reviews, 'Neutral Reviews Word Cloud')
```

Neutral Reviews Word Cloud




```
In [45]: #Importing packahes to see popular phrases
         from nltk import ngrams
         from collections import Counter
```

```
In [46]: # Ensure necessary nltk packages are downloaded
         nltk.download('punkt')
         nltk.download('stopwords')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\sujoydutta\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\sujoydutta\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
Out[46]: True
```

```
In [50]: #Function to get top phrases
         def get_top_phrases(texts, n=5, top_k=10):
             stop_words = set(nltk.corpus.stopwords.words('english'))


             all_text = ' '.join(texts)


             tokens = nltk.word_tokenize(all_text.lower())


             filtered_tokens = [word for word in tokens if word.isalnum() and word not in stop_wo


             n_grams = list(ngrams(filtered_tokens, n))


             n_gram_counts = Counter(n_grams)


             most_common_phrases = n_gram_counts.most_common(top_k)

             return [' '.join(phrase) for phrase, count in most_common_phrases]
```

In [51]:
```
# Function to get top phrases by sentiment category
def get_phrases_by_sentiment(df, sentiment_category, n=5, top_k=10):

    filtered_reviews = df[df['sentiment_category'] == sentiment_category]['review_text']


    top_phrases = get_top_phrases(filtered_reviews, n, top_k)

    return top_phrases
```

In [52]:
```
# Getting most popular phrases in the positive category
top_positive_phrases = get_phrases_by_sentiment(bacr, 'Positive', n=5, top_k=10)

print("Top Positive Phrases:", top_positive_phrases)
```

```
Top Positive Phrases: ['first time flying british airways', 'avios ba amex companion vou
cher', 'london heathrow new york jfk', 'ba first class check area', 'would definitely co
nsider flying ba', 'flying british airways first time', 'british airways short haul busi
ness', 'airways short haul business class', 'wife used avios ba amex', 'used avios ba am
ex companion']
```

In [54]:
```
# Getting most popular phrases in the neutral category
top_neutral_phrases = get_phrases_by_sentiment(bacr, 'Neutral', n=5, top_k=10)

print("Top Neutral Phrases:", top_neutral_phrases)
```

```
Top Neutral Phrases: ['british airways still forcing people', 'airways still forcing peo
ple fly', 'foreign office advised anything essential', 'office advised anything essentia
l travel', 'passengers already ordered therefore please', 'already ordered therefore ple
ase select', 'ordered therefore please select another', 'therefore please select another
option', 'via london heathrow british airways', 'qatar airways flight british airways']
```

In [57]:
```
# Getting most popular phrases in the negative category
top_negative_phrases = get_phrases_by_sentiment(bacr, 'Negative', n=5, top_k=10)
print("Top Negative Phrases:", top_negative_phrases)
```

```
Top Negative Phrases: ['via london heathrow british airways', 'new york jfk london heath
row', 'ba customer service extremely poor', 'british airways would help us', 'poor custo
mer service british airways', 'delayed least another 30 minutes', 'flight delayed almost
3 hours', 'low cost airline customer service', '48 hr negative covid test', 'cabin crew
spent much time']
```