

Task 1

Web scraping and analysis

This Jupyter notebook includes some code to get you started with web scraping. We will use a package called `BeautifulSoup` to collect the data from the web. Once you've collected your data and saved it into a local `.csv` file you should start with your analysis.

Scraping data from Skytrax

If you visit <https://www.airlinequality.com> you can see that there is a lot of data there. For this task, we are only interested in reviews related to British Airways and the Airline itself.

If you navigate to this link: <https://www.airlinequality.com/airline-reviews/british-airways> you will see this data. Now, we can use Python and `BeautifulSoup` to collect all the links to the reviews and then to collect the text data on each of the individual review links.

```
import os
import requests
from bs4 import BeautifulSoup
import pandas as pd

base_url = "https://www.airlinequality.com/airline-reviews/british-airways"
pages = 10
page_size = 100

reviews = []

# for i in range(1, pages + 1):
for i in range(1, pages + 1):

    print(f"Scraping page {i}")

    # Create URL to collect links from paginated data
    url = f"{base_url}/page/{i}?sortBy=post_date%3ADesc&pagesize={page_size}"

    # Collect HTML data from this page
    response = requests.get(url)

    # Parse content
    content = response.content
    parsed_content = BeautifulSoup(content, 'html.parser')
    for para in parsed_content.find_all("div", {"class": "text_content"}):
        reviews.append(para.get_text())

    print(f"----> {len(reviews)} total reviews")
```

```
Scraping page 1
----> 100 total reviews
Scraping page 2
----> 200 total reviews
Scraping page 3
----> 300 total reviews
Scraping page 4
----> 400 total reviews
Scraping page 5
----> 500 total reviews
Scraping page 6
----> 600 total reviews
Scraping page 7
----> 700 total reviews
Scraping page 8
----> 800 total reviews
Scraping page 9
----> 900 total reviews
Scraping page 10
----> 1000 total reviews
```

```
df = pd.DataFrame()
df["reviews"] = reviews
df.head()
```



	reviews
0	✔ Trip Verified Flight mainly let down by ...
1	✔ Trip Verified Another awful experience b...
2	✔ Trip Verified The service was rude, full...
3	✔ Trip Verified This flight was a joke. Th...
4	✔ Trip Verified This time British Airways ...

```
if not os.path.exists(os.path.join(os.getcwd(), "data")):
    os.mkdir("data")

df.to_csv("data/BA_reviews.csv")
```

Congratulations! Now you have your dataset for this task! The loops above collected 1000 reviews by iterating through the paginated pages on the website. However, if you want to collect more data, try increasing the number of pages!

The next thing that you should do is clean this data to remove any unnecessary text from each of the rows. For example, "✔ Trip Verified" can be removed from each row if it exists, as it's not relevant to what we want to investigate.

```
import pandas as pd

# Load the dataset
file_path = "data/BA_reviews.csv"
df = pd.read_csv(file_path)

# Display the first few rows to understand its structure
df.head()
```



	Unnamed: 0	reviews
0	0	✔ Trip Verified Flight mainly let down by ...
1	1	✔ Trip Verified Another awful experience b...
2	2	✔ Trip Verified The service was rude, full...
3	3	✔ Trip Verified This flight was a joke. Th...
4	4	✔ Trip Verified This time British Airways ...


```
# Function to clean the reviews
def clean_review(review):
    # Remove "✔ Trip Verified | " prefix
    if review.startswith("✔ Trip Verified | "):
        review = review.replace("✔ Trip Verified | ", "")
    # Remove leading and trailing whitespace
    review = review.strip()
    return review

# Apply the cleaning function to the 'reviews' column
df['reviews'] = df['reviews'].apply(clean_review)

# Save the cleaned dataset to a new CSV file
df.to_csv('BA_reviews_cleaned.csv', index=False)

print("Dataset cleaned and saved as 'BA_reviews_cleaned.csv'")
df.head()
```

Dataset cleaned and saved as 'BA_reviews_cleaned.csv'

Unnamed: 0	reviews
0	0 Flight mainly let down by a disagreeable fligh...
1	1 Another awful experience by British Airways. T...
2	2 The service was rude, full of attitude to me, ...
3	3 This flight was a joke. There was four people ...
4	4  Trip Verified This time British Airways ...

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

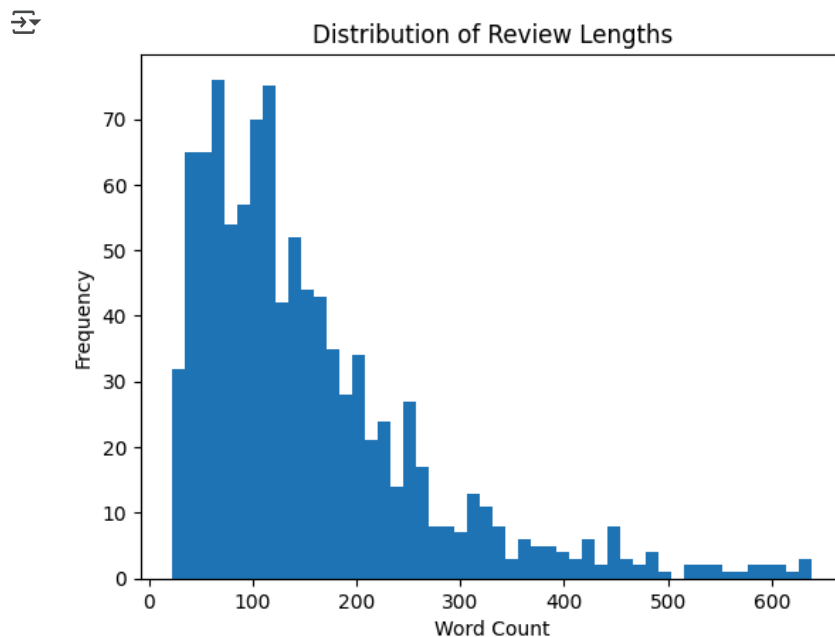
```
# Check the number of reviews
print(f"Total reviews: {len(df)}")
```

```
# Check for missing values
print(f"Missing reviews: {df['reviews'].isnull().sum()}")
```

```
# Analyze review lengths
df['word_count'] = df['reviews'].apply(lambda x: len(x.split()))
print(df['word_count'].describe())
```

```
Total reviews: 1000
Missing reviews: 0
count    1000.000000
mean      155.682000
std       112.331559
min        23.000000
25%       75.750000
50%      123.000000
75%      198.500000
max       639.000000
Name: word_count, dtype: float64
```

```
# Plot a histogram of review lengths
plt.hist(df['word_count'], bins=50)
plt.title("Distribution of Review Lengths")
plt.xlabel("Word Count")
plt.ylabel("Frequency")
plt.show()
```



```
# Generate a word cloud
wordcloud = WordCloud(width=800, height=400, background_color='white').generate(' '.join(df['reviews']))
```

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



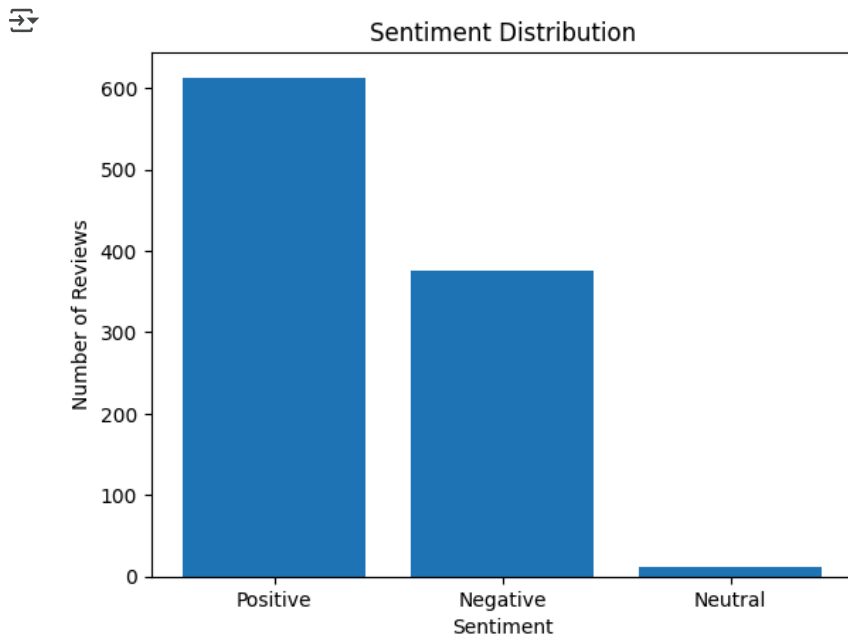
```
from textblob import TextBlob

# Function to get sentiment polarity
def get_sentiment(review):
    return TextBlob(review).sentiment.polarity

# Apply sentiment analysis
df['sentiment'] = df['reviews'].apply(get_sentiment)

# Categorize sentiment
df['sentiment_category'] = df['sentiment'].apply(lambda x: 'Positive' if x > 0 else 'Negative' if x < 0 else 'Neutral')

# Visualize sentiment distribution
sentiment_counts = df['sentiment_category'].value_counts()
plt.bar(sentiment_counts.index, sentiment_counts.values)
plt.title("Sentiment Distribution")
plt.xlabel("Sentiment")
plt.ylabel("Number of Reviews")
plt.show()
```



```

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation
import nltk
from nltk.corpus import stopwords

# Download stopwords
nltk.download('stopwords')
stop_words = list(stopwords.words('english')) # Convert set to list

# Preprocess text
vectorizer = CountVectorizer(max_df=0.95, min_df=2, stop_words=stop_words)
dtm = vectorizer.fit_transform(df['reviews'])

# Apply LDA
lda = LatentDirichletAllocation(n_components=5, random_state=42)
lda.fit(dtm)

# Display topics
for index, topic in enumerate(lda.components_):
    print(f"Topic #{index + 1}")
    print([vectorizer.get_feature_names_out()[i] for i in topic.argsort()[-10:]])
    print()

[🔗] [nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Unzipping corpora/stopwords.zip.
Topic #1
['customer', 'british', 'service', 'would', 'told', 'hours', 'london', 'us', 'ba', 'flight']

Topic #2
['airline', 'check', 'business', 'fly', 'time', 'staff', 'one', 'service', 'flight', 'ba']

Topic #3
['plane', 'time', 'cabin', 'crew', 'seat', 'business', 'seats', 'class', 'ba', 'flight']

Topic #4
['cancelled', 'travel', 'luggage', 'voucher', 'customer', 'ba', 'service', 'flight', 'british', 'airways']

Topic #5
['class', 'time', 'seat', 'cabin', 'service', 'ba', 'crew', 'food', 'good', 'flight']

from sklearn.feature_extraction.text import TfidfVectorizer

# Use TF-IDF to extract keywords
tfidf = TfidfVectorizer(max_features=100, stop_words='english')
tfidf_matrix = tfidf.fit_transform(df['reviews'])

# Get feature names (keywords)
feature_names = tfidf.get_feature_names_out()


# Sum TF-IDF scores for each word
tfidf_scores = tfidf_matrix.sum(axis=0).A1
word_score_pairs = list(zip(feature_names, tfidf_scores))
sorted_word_scores = sorted(word_score_pairs, key=lambda x: x[1], reverse=True)

# Display top 10 keywords
print("Top 10 Keywords:")
for word, score in sorted_word_scores[:10]:
    print(f"{word}: {score}")

[🔗] Top 10 Keywords:
flight: 151.31550543125542
ba: 120.75553357360965
service: 80.75477062948934
time: 66.66243577201229
crew: 66.45170283917075
good: 63.33506909944609
food: 63.15101486515086
london: 62.00135347102991
seat: 61.75022136967446
class: 58.656549884999826

```

```
df.to_csv('BA_reviews_analyzed.csv', index=False)  
print("Analysis results saved as 'BA_reviews_analyzed.csv'")
```

 Analysis results saved as 'BA_reviews_analyzed.csv'