Cwen Fernandes

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] (https://www.airlinequality.com%5D) 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] (https://www.airlinequality.com/airline-reviews/british-airways%5D) 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.

In [ ]:
```
import requests
from bs4 import BeautifulSoup
import pandas as pd
```

In [ ]:

```
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
```

In [ ]:

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

Out[3]:

	reviews
0	Not Verified | British Airways is always late,...
1	✅ Trip Verified | Flew from Amman to London on...
2	✅ Trip Verified | This is the worst experience...
3	✅ Trip Verified | Flying LHR T5 to CPT Novemb...
4	Not Verified | Worst experience ever. Outbound...

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.

In [ ]:
```
df
```

Out[4]:

	reviews
0	Not Verified | British Airways is always late,...
1	✅ Trip Verified | Flew from Amman to London on...
2	✅ Trip Verified | This is the worst experience...
3	✅ Trip Verified | Flying LHR T5 to CPT Novemb...
4	Not Verified | Worst experience ever. Outbound...
...	...
995	✅ Trip Verified | Very full flight on G-BNLP/B...
996	✅ Trip Verified | Warsaw to London. WAW is not...
997	✅ Trip Verified | I booked my flight with Cat...
998	✅ Trip Verified | Flew British Airways from Li...
999	✅ Trip Verified | Singapore to Heathrow. I ski...

1000 rows × 1 columns

Removing the parts before | in the reviews column

In [ ]:
```
df.reviews= df.reviews.str.split('|',expand=True)[1]
```

In [ ]:
```
df
```

Out[6]:

	reviews
0	British Airways is always late, their website...
1	Flew from Amman to London on Nov. 14 2022. No...
2	This is the worst experience I have ever had ...
3	Flying LHR T5 to CPT November 2022: BA app ...
4	Worst experience ever. Outbound flight was ca...
...	...
995	Very full flight on G-BNLP/B747 flying from M...
996	Warsaw to London. WAW is not a pleasant airpo...
997	I booked my flight with Cathay Pacific, the...
998	Flew British Airways from Lisbon to London He...
999	Singapore to Heathrow. I skipped my meal on b...

1000 rows × 1 columns

Rule-based approach

This is a practical approach to analyzing text without training or using machine learning models. The result of this approach is a set of rules based on which the text is labeled as positive/negative/neutral. These rules are also known as lexicons. Hence, the Rule-based approach is called Lexicon based approach.

Widely used lexicon-based approaches are TextBlob, VADER, SentiWordNet.

Data preprocessing steps:

Cleaning the text

Tokenization

Enrichment – POS tagging

Stopwords removal

Obtaining the stem words

Step 1: Cleaning the text

In [ ]:
```
import re

# Define a function to clean the text
def clean(text):
# Removes all special characters and numericals leaving the alphabets
    text = re.sub('[^A-Za-z]+', ' ', str(text))
    return text

# Cleaning the text in the review column
df['Cleaned Reviews'] = df['reviews'].apply(clean)
df.head()
```

Out[7]:

	reviews	Cleaned Reviews
0	British Airways is always late, their website...	British Airways is always late their website ...
1	Flew from Amman to London on Nov. 14 2022. No...	Flew from Amman to London on Nov Not sure wha...
2	This is the worst experience I have ever had ...	This is the worst experience I have ever had ...
3	Flying LHR T5 to CPT November 2022: BA app ...	Flying LHR T to CPT November BA app and websi...
4	Worst experience ever. Outbound flight was ca...	Worst experience ever Outbound flight was can...

Step 2: Tokenization

Tokenization is the process of breaking the text into smaller pieces called Tokens. It can be performed at sentences(sentence tokenization) or word level(word tokenization).

Step 3: Enrichment – POS tagging

Parts of Speech (POS) tagging is a process of converting each token into a tuple having the form (word, tag). POS tagging essential to preserve the context of the word and is essential for Lemmatization.

Step 4: Stopwords removal

Stopwords in English are words that carry very little useful information. We need to remove them as part of text preprocessing. nltk has a list of stopwords of every language.

Step 5: Obtaining the stem words

A stem is a part of a word responsible for its lexical meaning. The two popular techniques of obtaining the root/stem words are Stemming and Lemmatization.

The key difference is Stemming often gives some meaningless root words as it simply chops off some characters in the end. Lemmatization gives meaningful root words, however, it requires POS tags of the words

```
NLTK is a leading platform for building Python programs to work with human language data.

It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along
with a suite of text processing libraries for classification, tokenization, stemming, tagging,
parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries
```

In [ ]:

```
import nltk

"""This punkt tokenizer divides a text into a list of sentences by using an unsupervised algorithm to bui
collocations, and words that start sentences. """

nltk.download('punkt')
from nltk.tokenize import word_tokenize
from nltk import pos_tag
nltk.download('stopwords')
from nltk.corpus import stopwords
nltk.download('wordnet')
from nltk.corpus import wordnet
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
```

In [ ]:

```
#The nltk.corpus package defines a collection of corpus reader classes, which can be used to access the c


nltk.download('omw-1.4')
nltk.download('averaged_perceptron_tagger')

# POS tagger dictionary
pos_dict = {'J':wordnet.ADJ, 'V':wordnet.VERB, 'N':wordnet.NOUN, 'R':wordnet.ADV}
def token_stop_pos(text):
    tags = pos_tag(word_tokenize(text))
    #print(tags)
    newlist = []
    for word, tag in tags:
        if word.lower() not in set(stopwords.words('english')):
          newlist.append(tuple([word, pos_dict.get(tag[0])]))
          #print(tag[0])
          #print(pos_dict.get(tag[0]))
    return newlist

df['POS tagged'] = df['Cleaned Reviews'].apply(token_stop_pos)
df.head()
```

```
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
```

Out[10]:

	reviews	Cleaned Reviews	POS tagged
0	British Airways is always late, their website...	British Airways is always late their website ...	[(British, a), (Airways, n), (always, r), (lat...
1	Flew from Amman to London on Nov. 14 2022. No...	Flew from Amman to London on Nov Not sure wha...	[(Flew, n), (Amman, n), (London, n), (Nov, n),...
2	This is the worst experience I have ever had ...	This is the worst experience I have ever had ...	[(worst, a), (experience, n), (ever, r), (airl...
3	Flying LHR T5 to CPT November 2022: BA app ...	Flying LHR T to CPT November BA app and websi...	[(Flying, v), (LHR, n), (CPT, n), (November, n...
4	Worst experience ever. Outbound flight was ca...	Worst experience ever Outbound flight was can...	[(Worst, n), (experience, n), (ever, r), (Outb...

In [ ]:

```
# Obtaining the stem words - Lemmatization

from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer()
def lemmatize(pos_data):
    lemma_rew = " "
    for word, pos in pos_data:
     if not pos:
        lemma = word
        lemma_rew = lemma_rew + " " + lemma
     else:
        lemma = wordnet_lemmatizer.lemmatize(word, pos=pos)
        lemma_rew = lemma_rew + " " + lemma
    return lemma_rew

df['Lemma'] = df['POS tagged'].apply(lemmatize)
df.head()
```

Out[11]:

	reviews	Cleaned Reviews	POS tagged	Lemma
0	British Airways is always late, their website...	British Airways is always late their website ...	[(British, a), (Airways, n), (always, r), (lat...	British Airways always late website atrociou...
1	Flew from Amman to London on Nov. 14 2022. No...	Flew from Amman to London on Nov Not sure wha...	[(Flew, n), (Amman, n), (London, n), (Nov, n),...	Flew Amman London Nov sure type aircraft Tic...
2	This is the worst experience I have ever had ...	This is the worst experience I have ever had ...	[(worst, a), (experience, n), (ever, r), (airl...	bad experience ever airline fly British Airl...
3	Flying LHR T5 to CPT November 2022: BA app ...	Flying LHR T to CPT November BA app and websi...	[(Flying, v), (LHR, n), (CPT, n), (November, n...	Flying LHR CPT November BA app website work ...
4	Worst experience ever. Outbound flight was ca...	Worst experience ever Outbound flight was can...	[(Worst, n), (experience, n), (ever, r), (Outb...	Worst experience ever Outbound flight cancel...

In [ ]:
```
df[['reviews','Lemma']]
```

Out[12]:

	reviews	Lemma
0	British Airways is always late, their website...	British Airways always late website atrociou...
1	Flew from Amman to London on Nov. 14 2022. No...	Flew Amman London Nov sure type aircraft Tic...
2	This is the worst experience I have ever had ...	bad experience ever airline fly British Airl...
3	Flying LHR T5 to CPT November 2022: BA app ...	Flying LHR CPT November BA app website work ...
4	Worst experience ever. Outbound flight was ca...	Worst experience ever Outbound flight cancel...
...	...	...
995	Very full flight on G-BNLP/B747 flying from M...	full flight G BNLP B fly Miami London Accord...
996	Warsaw to London. WAW is not a pleasant airpo...	Warsaw London WAW pleasant airport one loung...
997	I booked my flight with Cathay Pacific, the...	book flight Cathay Pacific route Stockholm L...
998	Flew British Airways from Lisbon to London He...	Flew British Airways Lisbon London Heathrow ...
999	Singapore to Heathrow. I skipped my meal on b...	Singapore Heathrow skip meal board request s...

1000 rows × 2 columns

Sentiment Analysis using VADER

VADER stands for Valence Aware Dictionary and Sentiment Reasoner.

Vader sentiment not only tells if the statement is positive or negative along with the intensity of emotion.

In [ ]:
```
!pip install vaderSentiment
```

```
Looking in indexes: https://pypi.org/simple, (https://pypi.org/simple,) https://us-python.pkg.dev/colab-wheels/public/simple/ (https://us-python.pkg.dev/colab-wheels/public/simple/)
Collecting vaderSentiment
  Downloading vaderSentiment-3.3.2-py2.py3-none-any.whl (125 kB)
     |████████████████████████████████| 125 kB 4.9 MB/s eta 0:00:01
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from vaderSentiment) (2.23.0)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests->vaderSentiment) (3.0.4)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests->vaderSentiment) (1.24.3)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests->vaderSentiment) (2.10)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests->vaderSentiment) (2022.9.24)
Installing collected packages: vaderSentiment
Successfully installed vaderSentiment-3.3.2
```

In [ ]:

```
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
analyzer = SentimentIntensityAnalyzer()


# function to calculate vader sentiment
def vadersentimentanalysis(review):
    vs = analyzer.polarity_scores(review)
    return vs['compound']

df['Sentiment'] = df['Lemma'].apply(vadersentimentanalysis)

# function to analyse
def vader_analysis(compound):
    if compound >= 0.5:
        return 'Positive'
    elif compound < 0 :
        return 'Negative'
    else:
        return 'Neutral'
df['Analysis'] = df['Sentiment'].apply(vader_analysis)
df.head()
```

Out[14]:

	reviews	Cleaned Reviews	POS tagged	Lemma	Sentiment	Analysis
0	British Airways is always late, their website...	British Airways is always late their website ...	[(British, a), (Airways, n), (always, r), (lat...	British Airways always late website atrociou...	-0.2960	Negative
1	Flew from Amman to London on Nov. 14 2022. No...	Flew from Amman to London on Nov Not sure wha...	[(Flew, n), (Amman, n), (London, n), (Nov, n),...	Flew Amman London Nov sure type aircraft Tic...	0.6419	Positive
2	This is the worst experience I have ever had ...	This is the worst experience I have ever had ...	[(worst, a), (experience, n), (ever, r), (airl...	bad experience ever airline fly British Airl...	-0.2960	Negative
3	Flying LHR T5 to CPT November 2022: BA app ...	Flying LHR T to CPT November BA app and websi...	[(Flying, v), (LHR, n), (CPT, n), (November, n...	Flying LHR CPT November BA app website work ...	0.7096	Positive
4	Worst experience ever. Outbound flight was ca...	Worst experience ever Outbound flight was can...	[(Worst, n), (experience, n), (ever, r), (Outb...	Worst experience ever Outbound flight cancel...	-0.8591	Negative

In [ ]:
```
vader_counts = df['Analysis'].value_counts()
vader_counts
```

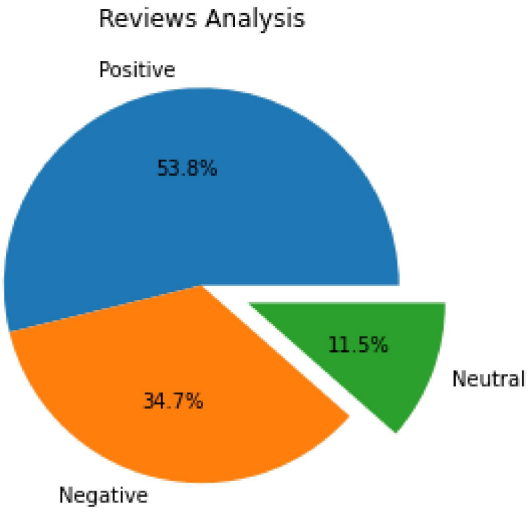
Out[15]:
```
Positive    538
Negative    347
Neutral     115
Name: Analysis, dtype: int64
```

Visual Representation

In [ ]:
```
import matplotlib.pyplot as plt
%matplotlib inline
plt.figure(figsize=(15,7))

plt.subplot(1,3,2)
plt.title("Reviews Analysis")
plt.pie(vader_counts.values, labels = vader_counts.index, explode = (0, 0, 0.25), autopct='%1.1f%%', shad
```

Out[16]:
```
([<matplotlib.patches.Wedge at 0x7f290dab5350>,
  <matplotlib.patches.Wedge at 0x7f290dab5ad0>,
  <matplotlib.patches.Wedge at 0x7f290dac5390>],
 [Text(-0.13100683356568732, 1.0921708701293458, 'Positive'),
  Text(-0.26350539923933375, -1.0679723332426359, 'Negative'),
  Text(1.2628494161539447, -0.4771911064968008, 'Neutral')],
 [Text(-0.07145827285401125, 0.5957295655250977, '53.8%'),
  Text(-0.1437302177669093, -0.5825303635868921, '34.7%'),
  Text(0.7951274101710023, -0.3004536596461338, '11.5%')])
```



In [ ]:
```
df.to_csv("BA_reviews.csv")
```

Wordcloud

Word Cloud or Tag Clouds is a visualization technique for texts that are natively used for visualizing the tags or keywords from the websites

In [ ]:

```
from wordcloud import WordCloud, STOPWORDS
stopwords = set(STOPWORDS)

def show_wordcloud(data):
    wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        max_words=100,
        max_font_size=30,
        scale=3,
        random_state=1)

    wordcloud=wordcloud.generate(str(data))

    fig = plt.figure(1, figsize=(12, 12))
    plt.axis('off')

    plt.imshow(wordcloud)
    plt.show()

show_wordcloud(df.Lemma)
```


pleasant
request
board
G
always
book
airline
London
type
Cathay
CPT
airport
Accord
Worst
loung
Airl
B
atrociou
November
Miami
Amman
Lemma
Flew
Flying
route
flight
dtype
work
one
Warsaw
WAW
experiencefly
Lisbon
object
bad
Singapore
Nov
cancel
L
British
meal
LHR
Pacific
s
full
aircraft
sure
Heathrow
Length
BA
Stockholm
Outbound
Tic
website
late
BNLP
skip
Airways
app
Name