Lab - 8 Checking the negative and positive text/comments/reviews

We have two different datasets named as `Airbnb/positive.txt` and `Airbnb/negative.txt` , which is a list of all the positive and negative words. Based on that, we will compare the negative and positive words from the comments columns which is mainly used for the text sentiment analysis.

In [ ]:
```
import pandas as pd
import numpy as np
import datetime as dt
import seaborn as sns #seaborn is already installed
import matplotlib.pyplot as plt
```

In [ ]:
```
df = pd.read_csv('Airbnb/listings.csv')
print(df.shape)
df.head(3)
```

In [ ]:
```
#viewing every columns properly
count = 1
for i in df.columns:
  print(count, i)
  count += 1
```

In [ ]:
```
#1. Take only the columns that are necessary for analysis
# No of reviews datas was found misleading so it has not been considered here
df = df[['id','neighbourhood','name','room_type','price']]
#df = df[['id','description','zipcode']]
df
```

In [ ]:
```
#Check for any NaN values that affect the dataset
print('Number of rows in each column affected by existence of non-existing values:')
df.isnull().sum()
```

In [ ]:
```
review = pd.read_csv('Airbnb/reviews.csv')
review.head()
```

In [ ]:
```
review = review[['listing_id','comments']]
```

In [ ]:
```
#1. Remove rows that do not contain comments.
reviewNAcomments=review[(review.comments.isnull())]
print(reviewNAcomments.shape)
review=review[~(review.comments.isnull())]
```

In [ ]:
```
#group the dataframe by listing id and then bring all the comments to a particular listing_id to group accord
review_group = review.groupby('listing_id')
print(review_group)
review = review_group.apply(lambda x: list(x['comments']))
```

In [ ]:
```
#Convert from series to dataframe
review = review.to_frame('comments')
review
```

In [ ]:
```
# merging full review + add only specific columns from df
cleaned_df = pd.merge(left=review, right=df, how='left', left_on=review.index, right_on='id')
cleaned_df
```

In [ ]:
```
# cleaned_df.to_csv('processed_airbnb.csv')
```

compare all the words remaining with the positive and negative words that are available based on dictionary file

Positive Comment = positive_word_count > negative_word_count Neutral Comment = positive_word_count == negative_word_count Negative Comment = positive_word_count < negative_word_count

In [ ]:
```
#loading the dataset of positive text and Negative text file:
p_file = open("Airbnb/positive.txt", "r")
positive_list = []
for line in p_file:
    stripped_line = line.strip()
    positive_list.append(stripped_line)
p_file. close()

#loading the dataset for negative text file
n_file = open("Airbnb/negative.txt", "r")
negative_list = []
for line in n_file:
    stripped_line = line.strip()
    negative_list.append(stripped_line)
n_file. close()

def positive_negative_checker(cleanW):
    #print(cleanW)
    #quit()
# comparing whether the comment is negative or neutral or positive
    positive_count = 0;
    negative_count = 0;
    for c in cleanW:
        if c in positive_list:
            positive_count = positive_count + 1

        elif c in negative_list:
            negative_count = negative_count + 1
   # print("Positive Count is: %d \n" %( positive_count))
   # print("Negative Count is: %d \n" %( negative_count))
    #quit()
    if negative_count > positive_count:
        return -1
    elif positive_count > negative_count:
        return 1
    else:
        return 0
```

In [ ]:
```
import nltk
#stopwords = set(STOPWORDS) # STOPWORDS is a list with english common words that you should not count in the
                            # wordcloud, like prepositions and conjuctions
# nltk.download('stopwords')
# nltk.download('punkt')
# from nltk.corpus import stopwords
# stopWords = set(stopwords.words('english'))

#stopwords = set(STOPWORDS) # STOPWORDS is a list with english common words that you should not count in the
                            # wordcloud, like prepositions and conjuctions
```

In [ ]:
```
#We are defining the function getRanks that receive only one parameter: the class to analyse.
#import word_tokenize
import collections,string
from nltk import word_tokenize
def comment_analyser(sentence_list):
    positive_comment_count = 0
    neutral_comment_count = 0
    negative_comment_count = 0
    for i in range(len(sentence_list)):
        words = word_tokenize(sentence_list[i])  #Tokenise all values stored in revtextC
        words= [w.lower() for w in words] #Change to lower
        words = [word for word in words if word.isalnum()]  # Remove conjuctions/punctuation
        #words = [ele for ele in words if ele not in stopWords] #Remove stopwords
        cleanW = [ele for ele in words if ele not in stopwords]

        comment_value = positive_negative_checker(cleanW)
        if comment_value == 1:
          positive_comment_count += 1
        elif comment_value == 0:
          neutral_comment_count += 1
        elif comment_value == -1:
          negative_comment_count += 1

    analyzed_comment = [positive_comment_count, neutral_comment_count, negative_comment_count]
    print(analyzed_comment)
    return analyzed_comment
```

In [ ]:
```
cleaned_df.insert(1,'positive_comment', 0)
cleaned_df.insert(2,'neutral_comment', 0)
cleaned_df.insert(2,'negative_comment', 0)
```

In [ ]:
```
#printing the cleaned dataframe

cleaned_df
```

In [ ]:
```
from wordcloud import WordCloud, STOPWORDS
stopwords = set(STOPWORDS)
count = 0
checking_value = 0
for i in range(len(cleaned_df)):

    checking_value = [] #gets the total positive comments, neutral comments and negative comments count
    checking_value = comment_analyser(cleaned_df.iloc[i,0])
    cleaned_df.iloc[i, 1] = checking_value[0] #put the total_positive_comment into positive_comment column
    cleaned_df.iloc[i, 2] = checking_value[1] #put the total_neutral_comment into neutral_comment column
    cleaned_df.iloc[i, 3] = checking_value[2] #put the total_negative_comment into negative_comment column
    print(i+1) # printing the total rows that have been proceeded
```

In [ ]:
```
#you can save the cleaned data as csv file using .to_csv()
cleaned_df.to_csv('processed_airbnb.csv')
```

In [ ]:
```

```

In [ ]:
```
cleaned_df
```

In [ ]:
```

```