Noé Flores

Introduction:
Our data consists of a zip file (hotelCustsFA2017.zip) containing 26 json.text files, each containing TripAdvisor reviews for a particular hotel and some additional information as well, such as hotel name, and hotel id. Ultimately, we want to gather some basic statistics on the sum of the hotel review data, as well as explore the comments made by hotel visitors in an effort to extract any hot-words and examine the frequency of certain terms used in the reviews. Ideally, we want to extract these files in a loop in order to assist us in processing all the files more efficiently. After the forward loop is complete, we can proceed to create our desired pandas data frames and extract our desired information.

The data sets were initially housed in a zip file which was later transferred to our local directory. We created a file list which we can use in our forward loop to extract the specific files.

Part 1::
Create a 'file_list' from the directory and verify that our json files are located in the new 'file_list'.

```
###get list of files in current directory####
file list = os.listdir('.')
file list
Out[17]:
['100506.json',
 '1217974.json',
 '150849.json',
 '214680.json',
 '240124.json',
 '2515575.json',
 '287670.json',
 '550994.json',
 '655424.json',
 '677703.json',
 '72572.json',
 '72579.json',
 '72586.json',
 '72598.json',
 '73393.json',
 '73644.json',
 '73706.json',
 '73712.json',
 '73718.json',
 '73727.json',
 '73739.json',
 '73743.json',
 '73751.json',
 '73757.json',
 '73760.json',
 '73768.json',
```

We want to create a frame list that will extract the desired columns from the json files. Within that frame_list, which we have labeled final_data, we will house the forward loop within that structure.

```
###Frame list for json files
final data=pd.DataFrame()
###Create forward loop
for file in file_list:
    if file.endswith('.json'):
        with open(file) as input_file:
            jsondat = json.load(input_file)
```

The frame_list now contains the loop through which our json files will be extracted. In lieu of having to pull each and every file from the local directory, we will loop around and open a "file" within the file_list if it ends with '.json' .

We want to examine the file keys in order to know where we will be looking to extract the data for our new dataframes. The jsondat keys will provide us with the dataframes located within each file..

```
###Examine file keys
jsondat.keys()

jsondat.keys()
Out[19]: dict_keys(['HotelInfo', 'Reviews'])
```

Our files have two main dictionary keys, "HotelInfo" and "Reviews".
We can now examine the specific keys as they relate to "HotelInfo" and "Reviews".

```
###Create Dataframe for Hotel Info from first json file
hotel_info = jsondat['HotelInfo']
hotel_info.keys()

hotel_info.keys()
Out[20]: dict_keys(['ImgURL', 'Name', 'HotelID', 'HotelURL', 'Price', 'Address'])
```

HotelInfo has a total of six different dictionary keys. A similar process tells is taken with respect to "Reviews".

```
review_info = jsondat['Reviews']
review_info[0].keys()

Out[23]: dict_keys(['ReviewID', 'Ratings', 'Content', 'Author', 'AuthorLocation', 'Date', 'Title'])
```

What we realized is that for every hotel in our file, the reviews are not uniform. What we will do is create a frame_list that is inclusive of all the fields, and extract our desired information once it's been created and concatenated. The code below is what we used to create the frame_list.

```
import json
import os
import pandas as pd
from pandas.io.json import json_normalize
import numpy as np
###get list of files in current directory####
file_list = os.listdir('.')
file list

final data=pd.DataFrame()
###Create forward loop
for file in file_list:
    if file.endswith('.json'):
        with open(file) as input_file:
            jsondat = json.load(input file)

            ###Create Dataframe for Hotel Info from first json file
            hotel_info = jsondat['HotelInfo']
            ###Create Dataframe for Review Info from first json file
            review_info = jsondat['Reviews']
            ###Normalize Review Info
            review df = json normalize(review info)


            hotel_rating=review_df
            hotel_df = json_normalize(hotel_info)
            if (hotel_df.columns.str.contains('Name')).sum()>0:
                hotel df = hotel df[["HotelID","Name"]]
                hotel rating['HotelID']=hotel df['HotelID'].iloc[0]
                hotel_rating['Name']=hotel_df['Name'].iloc[0]
            else:
                hotel_df = hotel_df[["HotelID"]]
                hotel_rating['HotelID']=hotel_df['HotelID'].iloc[0]

            hotel df = hotel df


            final_data=final_data.append(hotel_rating)
```

Our "final_data" frame has 2485 rows of records and 17 total columns.

```
final_data.shape
Out[9]: (2485, 17)
```

We can now proceed to rename the columns of our "final_data" pandas data frame in order to remove the 'ratings' prefix.

```
###Change Column Names
final_data.columns =
["authorName","authorLocation","content","date","hotelID","name","Businessservices","internetaccess",

"fontdesk","cleanliness","location","overall","room","service","sleepquality","value","reviwID","title"]
```

Below is the head() and tail() for "final_data" frame.

```
final data.head(2)
Out[10]:
                 authorName     authorLocation  \
Hotel Seattle  luvsroadtrips      Arlington, WA
Hotel Seattle      estelle e  Vancouver, Canada

                                                  content  \
Hotel Seattle  This place is not even suitable for the homele...
Hotel Seattle  We stayed in downtown hotel Seattle for two ni...

                          date hotelID  Businessservices internetaccess  \
Hotel Seattle    January 3, 2012  100506               NaN            NaN
Hotel Seattle  December 29, 2011  100506               NaN            NaN
               fontdesk  cleanliness  location  overall  room  service  \
Hotel Seattle       NaN          1.0       5.0      1.0   1.0      1.0
Hotel Seattle       NaN          4.0       5.0      4.0   3.0      4.0
               sleepquality  value      reviwID  \
Hotel Seattle           1.0    1.0  UR122476164
Hotel Seattle           5.0    3.0  UR122239883
                                                  title
Hotel Seattle                "You've got to be kidding me??!"
Hotel Seattle  "great service, comfortable rooms, easy stay."

final_data.tail(2)
Out[11]:
                                authorName           authorLocation  \
Days Inn I-17 & Thomas  A TripAdvisor Member  North Richland Hills, Tx
Days Inn I-17 & Thomas  A TripAdvisor Member       Menomonee Falls, WI
                                                              content  \
Days Inn I-17 & Thomas  a few friends and i went to phoenix for spring...
Days Inn I-17 & Thomas  We arrived at the hotel relatively late at nig...
                                date hotelID  Businessservices  \
Days Inn I-17 & Thomas  April 17, 2004   73768              NaN
Days Inn I-17 & Thomas  April 16, 2004   73768              NaN
                       internetaccess  fontdesk  cleanliness  location  \
Days Inn I-17 & Thomas             NaN       NaN          NaN       NaN
Days Inn I-17 & Thomas             NaN       NaN          4.0       NaN
                        overall  room  service  sleepquality  value  \
Days Inn I-17 & Thomas      4.0   NaN      NaN           NaN    NaN
Days Inn I-17 & Thomas      4.0   4.0      4.0           NaN    5.0
                          reviwID  \
Days Inn I-17 & Thomas  UR1789221
Days Inn I-17 & Thomas  UR1789184
                                                             title
Days Inn I-17 & Thomas                    "Could not have been happier"
Days Inn I-17 & Thomas  "Shady neighborhood, but well worth the price"
```

Now, we can pickle our "final_data" dataframe and ensure that we can extract it correctly.

```
final_data.to_pickle("final_data.pickle")
final_data_unpkled_df = pd.read_pickle("final_data.pickle")
final data unpkled df.head(1)
Out[20]:
                 authorName     authorLocation  \
Hotel Seattle  luvsroadtrips      Arlington, WA
                                                  content  \
Hotel Seattle  This place is not even suitable for the homele...
                          date hotelID  Businessservices internetaccess  \
Hotel Seattle    January 3, 2012  100506               NaN            NaN
               fontdesk  cleanliness  location  overall  room  service  \
Hotel Seattle       NaN          1.0       5.0      1.0   1.0      1.0
               sleepquality  value      reviwID  \
Hotel Seattle           1.0    1.0  UR122476164
                                                  title
Hotel Seattle                "You've got to be kidding me??!"
```

We want to calculate some basic statistical measures of central tendancy for our rating categories. We will begin by converting the ratings fields to numeric.

```
###Convert ratings fields to numeric for stats
numfields = ["Businessservices",
             "fontdesk",
             "cleanliness",
             "location",
             "overall",
             "room",
             "service",
             "sleepquality",
             "value"]
final_data[numfields] = final_data[numfields].apply(pd.to_numeric, errors = "coerce")
```

Calculate the (Min) value for ratings fields:

```
###Min
pd.DataFrame.min(final data[numfields])
Out[22]:
Businessservices   -1.0
fontdesk           -1.0
cleanliness        -1.0
location           -1.0
overall             1.0
room               -1.0
service            -1.0
sleepquality        1.0
value              -1.0
dtype: float64
```

Calculate the (Max) value for ratings fields:

```
###Max
pd.DataFrame.max(final data[numfields])
Out[23]:
Businessservices    5.0
fontdesk            5.0
cleanliness         5.0
location            5.0
overall             5.0
room                5.0
service             5.0
sleepquality        5.0
value               5.0
dtype: float64
```

Calculate the (Mean) and (Median) value for ratings fields:

```
###Mean                                  ###Median
pd.DataFrame.mean(final_data[numfields])  pd.DataFrame.median(final_data[numfields])
Out[24]:                                  Out[25]:
Businessservices    0.461105              Businessservices   -1.0
fontdesk            2.141753              fontdesk            3.0
cleanliness         3.463675              cleanliness         4.0
location            3.202237              location            4.0
overall             3.728370              overall             4.0
room                3.096119              room                4.0
service             3.415385              service             4.0
sleepquality        3.816149              sleepquality        4.0
value               3.332767              value               4.0
dtype: float64                            dtype: float64
```

Calculate the (Standard Deviation) value for ratings fields:

```
###Standard Deviation
pd.DataFrame.std(final_data[numfields])
Out[26]:
Businessservices    2.206702
fontdesk            2.576034
cleanliness         1.907649
location            2.298497
overall             1.300262
room                1.841401
service             1.918414
sleepquality        1.279765
value               1.925090
dtype: float64
```

Since the mean and median values are close to one another, we can assume that the data is fairly notmally distributed. Typically, values that significantly differ when comparing the Median and the Mean are indicative of possible outlying data points or erroneous entries which affect normality.

Part 2:

In part 2, we want to process the written comments for each hotel, one hotel at a time and extract the number of times each "content" word occurs. To do this, we want to create a "dict" which should have the words as keys, and the counts of the times they occur as the values of the keys.

We start by writing code to extract the data we need into a forward loop.

```
import json
import os
import pandas as pd
from pandas.io.json import json_normalize
import numpy as np

###get list of files in current directory####
file list = os.listdir('.')
file_list

df_content=pd.DataFrame()
contentlist = []
###Create forward loop
for file in file_list:
    if file.endswith('.json'):
        with open(file) as input_file:
            jsondat = json.load(input_file)

            ###Create Dataframe for Hotel Info from first json file
            hotel info = jsondat['HotelInfo']
            ###Create Dataframe for Review Info from first json file
            review info = jsondat['Reviews']
            ###Normalize Review Info
            review_df = json_normalize(review_info)

        for i in jsondat["Reviews"]:
            commentdict = {"Content" : i["Content"]}

        contentlist.append(commentdict)
    df_content = pd.DataFrame(contentlist)
```

We can now verify the contents of this json file and make sure we extracted the contents data.

```
df_content.shape
Out[89]: (113, 1)

df_content.head(3)
Out[90]:
                                              Content
0  This past fall I went with a friend to Seattle...
1  Central location, breakfast every morning, cle...
2  Stayed at this hotel for 1 night having search...
```

In order to process the words count efficiently for the whole hotel, we must concatenate all of the content section into one large section. This will make our word analysis much easier. We perform a .join to consolidate the code in the content section.

```
###Comments
comments=df content
comments = ' '.join(comments['Content'])
```

To build our dictionary of content word counts, we need to exclude stop words, and also any other things like html tags and punctuation marks. We will use the natural language toolkit (nltk) to extract the unnecessary data.

```
###Import Modules
from nltk.tokenize import word_tokenize, sent_tokenize
import nltk
from nltk.corpus import stopwords
from string import punctuation
```

We can use the (nltk) to tekenize the comments in our data, essentially breaking every sentence into a single string.

```
###Sentences
sentences = sent_tokenize(comments) # tokenize comments into a list of sentences...
sentences

Out[99]:
['This past fall I went with a friend to Seattle.',
 'We stayed at the Best Western Loyal Inn and we enjoyed our stay.',
 'The front desk was very helpful and when you call with a request there was follow through.',
 'The room was clean and fresh smelling.',
 'The breakfast was good too.',
```

This process can be taken a step further and we can get a breakdown of every word in our content string.

```
####Words
words = word_tokenize(comments) # tokenize comments into a list of words...
words

Out[100]:
['This',
 'past',
 'fall',
 'I',
 'went',
 'with',
 'a',
```

We need be sure to exclude any stop-words and punctuation from our "contents" analysis as well. This analysis will be used to define key words, and stop-words are words which do not contain important significance when used in Search Queries. You can see from the output below that the words produced carry more substinance.

```
####exclude stop words"...
cust_stop_words = set(stopwords.words('english')+list(punctuation))
filtered_words = [word for sent in sentences for word in word_tokenize(sent) if word.lower() not in
cust_stop_words]
filtered words

Out[102]:
['past',
 'fall',
 'went',
 'friend',
 'Seattle',
 'stayed',
 'Best',
 'Western',
```

Now that we have extracted the key terms from the "contents" data for our hotel, we can also find the frequency of those key terms.

```
###Frequency of words
freq_words = nltk.FreqDist(filtered_words).most_common()  # using nlkt to get word frequencies
freq_words

Out[103]:
[('hotel', 177),
 ('room', 171),
 ('breakfast', 90),
 ('good', 76),
 ('clean', 70),
```

We can use the Collections Counter module in python to count sort the most frequent words as we desire.

```
###Frequency of words
freq words = nltk.FreqDist(filtered words).most common()  # using nlkt to get word frequencies
freq words

Out[103]:
[('hotel', 177),
 ('room', 171),
 ('breakfast', 90),
 ('good', 76),
 ('clean', 70),
```

We also want to find the number of unique content words for each of our hotel_dicts.

```
len(words_72572)
Out[2]: 3563
len(words_72579)
Out[4]: 2494
len(words_72586)
Out[6]: 3177
len(words_72598)
Out[8]: 1301
len(words_73393)
Out[10]: 1875
len(words_73644)
Out[12]: 1351
len(words_73706)
Out[14]: 1031
len(words_73712)
Out[16]: 875
len(words_73718)
Out[18]: 1695
len(words_73727)
Out[20]: 1497
len(words_73739)
Out[22]: 1771
len(words_73743)
Out[24]: 1131
len(words_73751)
Out[26]: 727
len(words_73757)
Out[28]: 977
len(words_73760)
Out[30]: 1242
len(words_73768)
Out[32]: 855
len(words_100506)
Out[34]: 1552
len(words_150849)
Out[36]: 14444
len(words_214680)
Out[38]: 3335
len(words_240124)
Out[40]: 1576
len(words_287670)
Out[42]: 3440
len(words_550994)
Out[44]: 3505
len(words_655424)
Out[46]: 5
len(words_677703)
Out[48]: 47
len(words_1217974)
Out[50]: 62
len(words_2515575)
Out[52]: 2421
```