prog_1

April 25, 2017

In [ ]:
```
"""
Mapping Amman (project with professor Anderson)
Freja Mickos, Lida Karadimou

first part of notebook:
Scraper for search results of restaurants
getting property title cuisine tags, number of reviews, ranking, prices, ur
the tripadvisor_restaurant file)
future plans would include finding most popular cuisine categories accordi

second part of notebook: tripadvisor_restaurant.py
takes as an input urls, returns more information on each individual restaur
future plans will include getting also location coordinates, mapping them a
"""
```

In [1]:
```
import requests
from bs4 import BeautifulSoup
import time
import pandas as pd
import random

#https://www.tripadvisor.com/Restaurants-g293986-Amman_Amman_Governorate.ht
"""
soup.find_all('a', {'class':'property_title'})
soup.find_all('span', {'class':'reviewCount'})
soup.find_all('div', {'class':'popIndex rebrand popIndexDefault'}) ranking
soup.find_all('span', {'class':'item price'})
soup.find_all('a', {'class':'item cuisine'})

"""

property_titles = []
review_count = []
rankings = []
prices = []
cuisines = []
url2 = []
```

```
for offset in range(0, 19):
    url = 'https://www.tripadvisor.com/Restaurants-g293986-oa' + str(offset

    r = requests.get(url)
#     soup = BeautifulSoup(r.text, "html.parser")
    time.sleep(0.05)
    soup = BeautifulSoup(r.text,"lxml")
#     li = soup.find_all('div', {'id':'EATERY_SEARCH_RESULTS'}) # all restau

#     for restaurant in soup.find_all('')
    for link in soup.find_all('a', {'class':'property_title'}):
        title = link.text
        property_titles.append(title.strip('\n'))

    for link in soup.find_all('span', {'class':'reviewCount'}):
        count = link.text
        review_count.append(count.strip('\n'))

    for link in soup.find_all('div', {'class':'popIndex rebrand popIndexDef
        rank = link.text
        rankings.append(rank.strip('\n'))

    # some restaurants don't have a price, have to figure this out
    for link in soup.find_all('span', {'class':'item price'}):
        price = link.text
        prices.append(price.strip('\n'))

    # this isn't really working as of now -- it collects all cuisine tags,
    #for link in soup.find_all('a', {'class':'item cuisine'}):
        #cuisine = link.text
        #print(cuisine)
        #cuisines.append(cuisine.strip('\n'))


    #for this I am not sure what you were thinking about it, but what if we
    #for each restaurant as a restaurant may be in more than one category a
    #clustering or anything ??


    t = []
    for link in soup.find_all('div', {'class':'cuisines'}):
        for k in link.find_all('a',{'class':'item cuisine'}):
            t.append(k.text)
        cuisines.append(t)
        t = []

    #added url so that we can use these links to move through all the pages
```

```
            # "https://tripadvisor.com/ + url_of_rest
            for link in soup.find_all('a',{'class':'property_title'}):
                url_of_rest = link['href']
                url2.append(url_of_rest.strip('\n'))

        #columns = {'property_title':property_titles, 'review_count':review_count,
        columns = {'property_title':property_titles, 'review_count':review_count, '
        df = pd.DataFrame(columns)
        print(df[:10])
```

```
                                        cuisines  \
0                              [American, Barbecue]
1         [Pizza, Arabic, Halal, Gluten Free Options]
2                            [Middle Eastern, Halal]
3  [Lebanese, Mediterranean, Middle Eastern, Hala...
4  [Mediterranean, Middle Eastern, Halal, Gluten ...
5               [Steakhouse, American, International]
6  [Lebanese, Mediterranean, Middle Eastern, Hala...
7  [American, Vegetarian Friendly, Vegan Options,...
8  [Fast Food, Mediterranean, Middle Eastern, Veg...
9  [Fast Food, Mediterranean, Middle Eastern, Veg...

              property_title                          ranking    review_count  \
0                    Brisket   #1 of 545 Restaurants in Amman    268 reviews
1              Pizza Roma Cafe   #2 of 545 Restaurants in Amman    300 reviews
2               Habibah Sweets   #3 of 545 Restaurants in Amman    665 reviews
3      Fakhreldin Restaurant    #4 of 545 Restaurants in Amman    787 reviews
4           Sufra Restaurant    #5 of 545 Restaurants in Amman    701 reviews
5      V Lounge & Restaurant    #6 of 545 Restaurants in Amman     57 reviews
6           Tawaheen al-Hawa    #7 of 545 Restaurants in Amman    713 reviews
7  Chestnut Restaurant & Pub    #8 of 545 Restaurants in Amman    283 reviews
8                      Hashem   #9 of 545 Restaurants in Amman  1,361 reviews
9                  Abu Jbara  #10 of 545 Restaurants in Amman    348 reviews

                                              url2
0  /Restaurant_Review-g293986-d7267482-Reviews-Br...
1  /Restaurant_Review-g293986-d3389330-Reviews-Pi...
2  /Restaurant_Review-g293986-d2084697-Reviews-Ha...
3  /Restaurant_Review-g293986-d1371269-Reviews-Fa...
4  /Restaurant_Review-g293986-d2406112-Reviews-Su...
5  /Restaurant_Review-g293986-d3989474-Reviews-V_...
6  /Restaurant_Review-g293986-d1641029-Reviews-Ta...
7  /Restaurant_Review-g293986-d7761403-Reviews-Ch...
8  /Restaurant_Review-g293986-d1918356-Reviews-Ha...
9  /Restaurant_Review-g293986-d3261923-Reviews-Ab...
```

```
In [5]: print(url2)
```

```
['/Restaurant_Review-g293986-d7267482-Reviews-Brisket-Amman_Amman_Governorate.html'
```

```
In [22]: len(url2)
```

```
Out[22]: 570
```

```
In [8]: import csv
        with open('restaurants.csv', 'w') as csvfile:
            df.to_csv('restaurants.csv', sep='\t', encoding='utf-8')
            #df.to_csv(path_or_buf=None, sep=', ', na_rep='', float_format=None, co
            #fieldnames = ['first_name', 'last_name']
            #writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
        #df.to_csv(path_or_buf=None, sep=', ', na_rep='', float_format=None, columr
```

```
In [30]: with open("links.csv", "w",newline="") as f:
             writer = csv.writer(f)
             for i in url2:
                 writer.writerow([i])
             #writer = csv.writer(f, dialect='excel')
             #writer.writerows(url2)
```

```
In [58]: a = str(review_count[0]).split
```

```
In [63]: a= review_count[0]
```

```
In [67]: a.split(" ")[0]
```

```
Out[67]: '268'
```

```
In [73]: for a in review_count:
             a.split(" ")[0]
             print(a.split(" ")[0])
```

```
268
300
665
787
701
57
713
283
1,361
348
82
197
414
237
116
```

141
160
275
56
99
131
342
157
308
61
20
144
29
138
39
268
300
665
787
701
57
713
283
1,361
348
82
197
414
237
116
141
160
275
56
99
131
342
157
308
61
20
144
29
138
39
268
300
665

787
701
57
713
283
1,361
348
82
197
414
237
116
141
160
275
56
99
131
342
157
308
61
20
144
29
138
39
268
300
665
787
701
57
713
283
1,361
348
82
197
414
237
116
141
160
275
56
99
131

342
157
308
61
20
144
29
138
39
268
300
665
787
701
57
713
283
1,361
348
82
197
414
237
116
141
160
275
56
99
131
342
157
308
61
20
144
29
138
39
268
300
665
787
701
57
713
283
1,361

348
82
197
414
237
116
141
160
275
56
99
131
342
157
308
61
20
144
29
138
39
268
300
665
787
701
57
713
283
1,361
348
82
197
414
237
116
141
160
275
56
99
131
342
157
308
61
20
144

29
138
39
268
300
665
787
701
57
713
283
1,361
348
82
197
414
237
116
141
160
275
56
99
131
342
157
308
61
20
144
29
138
39
268
300
665
787
701
57
713
283
1,361
348
82
197
414
237
116

141
160
275
56
99
131
342
157
308
61
20
144
29
138
39
268
300
665
787
701
57
713
283
1,361
348
82
197
414
237
116
141
160
275
56
99
131
342
157
308
61
20
144
29
138
39
268
300
665

787
701
57
713
283
1,361
348
82
197
414
237
116
141
160
275
56
99
131
342
157
308
61
20
144
29
138
39
268
300
665
787
701
57
713
283
1,361
348
82
197
414
237
116
141
160
275
56
99
131

342
157
308
61
20
144
29
138
39
268
300
665
787
701
57
713
283
1,361
348
82
197
414
237
116
141
160
275
56
99
131
342
157
308
61
20
144
29
138
39
268
300
665
787
701
57
713
283
1,361

348
82
197
414
237
116
141
160
275
56
99
131
342
157
308
61
20
144
29
138
39
268
300
665
787
701
57
713
283
1,361
348
82
197
414
237
116
141
160
275
56
99
131
342
157
308
61
20
144

29
138
39
268
300
665
787
701
57
713
283
1,361
348
82
197
414
237
116
141
160
275
56
99
131
342
157
308
61
20
144
29
138
39
268
300
665
787
701
57
713
283
1,361
348
82
197
414
237
116

141
160
275
56
99
131
342
157
308
61
20
144
29
138
39
268
300
665
787
701
57
713
283
1,361
348
82
197
414
237
116
141
160
275
56
99
131
342
157
308
61
20
144
29
138
39
268
300
665

```
787
701
57
713
283
1,361
348
82
197
414
237
116
141
160
275
56
99
131
342
157
308
61
20
144
29
138
39
```

In [ ]:

In [ ]:
```
"""beginning of second file:
tripadvisor_restaurant.ipynb

getting more information on individual restaurants and putting it on a df a
information now include: languages of reviews, types of travelers, rank, av
"""
```

In [ ]:
```
from lxml import html
import requests
from collections import OrderedDict
import json
import argparse
import pandas as pd
import csv

def parse(input_file):
    restaurants = []
```

```
    for line in input_file:
        url = line
        response = requests.get(url).text
        parser = html.fromstring(response)

        XPATH_RATING = '//div[@id="ratingFilter"]//ul//li'

        XPATH_TRAVELER = '//div[@class="col segment extraWidth"]//ul//li'
        XPATH_TIME = '//div[@class="col season extraWidth"]//ul/li'
        XPATH_LANGUAGE = '//div[@class="col language extraWidth"]//ul/li'
#         XPATH_DETAILS = '//div[contains(@class,"details_tab")]//div[conta
        XPATH_NAME = '//h1[@property="name"]//text()'
        XPATH_HOTEL_RATING = '//span[@property="ratingValue"]//@content'
        XPATH_REVIEWS = '//a[@property="reviewCount"]/@content'
        XPATH_RANK = '//div[@class="slim_ranking"]//text()'
        XPATH_OFFICIAL_DESCRIPTION = '//div[contains(@class,"additional_inf
        XPATH_LOCATION = '//div[@class="mapContainer"]//text()'

        ratings = parser.xpath(XPATH_RATING)
        travelers = parser.xpath(XPATH_TRAVELER)
        times = parser.xpath(XPATH_TIME)
        languages = parser.xpath(XPATH_LANGUAGE)

        raw_name = parser.xpath(XPATH_NAME)
        raw_rank = parser.xpath(XPATH_RANK)
        raw_review_count = parser.xpath(XPATH_REVIEWS)
        raw_rating = parser.xpath(XPATH_HOTEL_RATING)
        raw_official_description = parser.xpath(XPATH_OFFICIAL_DESCRIPTION)
#         raw_details = parser.xpath(XPATH_DETAILS)
        raw_location = parser.xpath(XPATH_LOCATION)

        name = ''.join(raw_name).strip() if raw_name else None
        rank = ''.join(raw_rank).strip() if raw_rank else None
        review_count = ''.join(raw_review_count).strip() if raw_review_cour
        hotel_rating = ''.join(raw_rating).strip() if raw_rating else None
        official_description = ' '.join(' '.join(raw_official_description).
#         details = ' '.join(' '.join(raw_details).split()) if raw_details e
        location = ' '.join(' '.join(raw_location).split()) if raw_location

        ratings_dict = OrderedDict()
        for rating in ratings:
            XPATH_RATING_KEY = './/div[@class="row_label"]//text()'
            XPATH_RATING_VALUE = './/span[@class="row_bar"]/following-sibli
            raw_rating_key = rating.xpath(XPATH_RATING_KEY)
            raw_rating_value = rating.xpath(XPATH_RATING_VALUE)
            cleaned_rating_key = ''.join(raw_rating_key).replace('\n','')
            cleaned_rating_value = ''.join(raw_rating_value).replace('\n','
            ratings_dict.update({cleaned_rating_key:cleaned_rating_value})
```

```
        travelers_list = []
        for traveler in travelers:
            XPATH_TRAVELER_TYPE = './/label//text()'
            XPATH_TRAVELER_COUNT = './/span//text()'
            raw_traveler_type = traveler.xpath(XPATH_TRAVELER_TYPE)
            raw_traveler_count = traveler.xpath(XPATH_TRAVELER_COUNT)
            cleaned_traveler_type = ''.join(raw_traveler_type).replace('\n'
            cleaned_traveler_type = cleaned_traveler_type[0]
            cleaned_traveler_count = ''.join(raw_traveler_count).replace('\
            travelers_list.append((cleaned_traveler_type,cleaned_traveler_c

        times_list = []
        for time in times:
            XPATH_TIME_TYPE = './/label//text()'
            XPATH_TIME_COUNT = './/span//text()'
            raw_time_type = time.xpath(XPATH_TIME_TYPE)
            raw_time_count = time.xpath(XPATH_TIME_COUNT)
            cleaned_time_type = ''.join(raw_time_type).replace('\n','').spl
            cleaned_time_type = cleaned_time_type[0]
            cleaned_time_count = ''.join(raw_time_count).replace('\n','')
            times_list.append((cleaned_time_type, cleaned_time_count))

        language_list = []
        for language in languages:
            XPATH_LANGUAGE_TYPE = './/label//text()'
#             XPATH_LANGUAGE_COUNT = './/span//text()'
            raw_language_type = language.xpath(XPATH_LANGUAGE_TYPE)
#             raw_language_count = language.xpath(XPATH_LANGUAGE_COUNT)
            cleaned_language_type = ''.join(raw_language_type).replace('\n'
            cleaned_language_type = cleaned_language_type[0]
#             cleaned_language_count = ''.join(raw_language_count).replace(
#             languages_dict.update({cleaned_language_type:cleaned_language_
            language_list.append(cleaned_language_type)
        language_list = language_list[1:]


        data = {'name':name,
                                    'rank':rank,
                                    'rating':hotel_rating,
                'review_count':review_count,
                'location':location,
                'official_description':official_description,
                  'travelers':travelers_list,
                  'times of year':times_list,
                  'languages':language_list
#                   'details':details
            }
```

```
            restaurants.append(data)


    #    columns = ['property_title', 'rank', 'rating', 'review_count', 'locati
        df = pd.DataFrame(restaurants)
        return df



    input_file = open('restaurants.txt', 'r')
    scraped_data = parse(input_file)
    scraped_data
```

In [ ]:
```
    with open('restaurants_details.csv', 'w') as csvfile:
        scraped_data.to_csv(csvfile, sep='\t', encoding='utf-8')
```