IPBeja

INSTITUTO POLITÉCNICO DE BEJA

Sistema de apoio à promoção do turismo rural
Fase de Webscraping

Gonçalo Amaro – 17440,
Pedro Tomás – 18962,
Vítor Abreu – 18966

15 de Dezembro, 2021

Conteúdo

Lista de Figuras

Lista de Tabelas

1 Introdução

O objectivo desta fase de trabalho é a recolha da informação que será utilizada no decorrer do projecto, isto é, o "web-scrapping".

Simplificando, o processo de "web-scrapping" é simplesmente a recolha de dados de uma forma automatizada, no nosso caso foi utilizado a linguagem Python juntamente com algumas bibliotecas como a "BeautifulSoup4".

Todas as informações recolhidas foram apenas dentro da localidade de Beja e entre todos os resultados alguns até podem ser comuns, umas vez que todos nós recolhemos por exemplo as "reviews" dos hotéis, atracções e restaurantes.

2 Planeamento

2.1 Divisão de Tarefas

Para a realização desta fase de trabalho o grupo decidiu dividir as tarefas e organizá-las a partir da plataforma "Trello" (https://trello.com/b/PIApdmTA/pi-2021-22), uma plataforma de gestão de projecto estilo *board* do qual podemos aplicar Kanban, Scrum ou outra AGILE *management framework*.

A divisão de tarefas decidida foi a distribuição de cada um dos três "websites" por cada um dos elementos do grupo, pela ordem de dificuldade em congruência linear com o tempo extra-curricular disponível de cada elemento.

Assim sendo, o site "TripAdvisor" foi realizado pelo aluno Gonçalo Amaro, o "Booking" pelo Pedro Tomás e o "Zomato" pelo Vítor Abreu. No final verificou-se que todos conseguiram aceder aos seus devidos "websites" e adquirir as informações possíveis via *scraping*.

2.2 Tecnologias Usadas

Na realização do *web-scrapping* foi desenvolvido um ambiente virtual de Python 3 para realizar os *scripts* que iriam recolher as informações.

Como forma de organizar todos os pacotes e possíveis actualizações de bibliotecas dentro do código também foi gerado um ficheiro .txt denominado "requirements" que actualizávamos e usávamos sempre que um dos elementos do grupo iria realizar o seu trabalho.

A linguagem optada para a construção dos *scripts* foi o Python já que é uma das mais acessíveis linguagens de programação disponíveis devido à sua simples *syntax* e também pela vasta quantidade de bibliotecas disponibilizadas, das quais temos uma dúzia que são bastante úteis para a realização deste projecto.

Para finalizar, todos os ficheiros foram guardados em formato .csv uma vez que, como explicado no relatório de progresso anterior, é um formato que é aceite e nos facilita a manipulação de dados (ETL), a alimentação dos dados ao algoritmo de *machine learning*, e pode ser usado com ferramentas de analise de dos e geradores de tabelas (como o PowerBI).

2.2.1 Ambientes Virtuais de Python

Neste projecto usamos Ambientes Virtuais de Python. Um ambiente virtual é uma forma de ter várias instâncias paralelas do interpretador de Python, cada uma com diferentes conjuntos de pacotes e diferentes configurações.

Cada ambiente virtual contém uma cópia discreta do interpretador de Python, incluindo cópias dos seus utilitários de suporte como o pip. Estes contêm também uma zona para instalação de pacotes/bibliotecas localmente (dentro do ambiente virtual), sendo esta a razão principal pela qual foi decido usá-los.

Tendo introduzido a razão, consegue-se perceber o óbvio: sendo este um trabalho de grupo e que posteriormente poderá ser testado pelos docentes ou futuros alunos, ao usar ambientes virtuais podemos fazer "pip freeze" para um ficheiro de texto do qual facilita a portabilidade e transmissão de requerimentos do projecto.

Para a criação destes ambientes virtuais foi instalado o virtualenvwrapper o qual traz o virtualenv como dependência e um *set* de extensões para o mesmo, tal como sempre recomendado pelo Professor José Jasnau Caeiro, todos os anos na disciplina de Linguagens de Programação.

2.2.2 Bibliotecas de Python usadas

Como dito previamente, na explicação pelo qual o uso de Python, foi referida a grande quantidade de bibliotecas que nos são facilmente fornecidas pelo pip.

Dentro deste repositório existe (perto de) uma dúzia de bibliotecas que nos permitem facilmente completar as nossas tarefas deste projecto. Dessa dúzia, para esta etapa, foram usadas:

- BeautifulSoup4, uma biblioteca que facilita a extracção de informações de páginas da web, fornecendo expressões para iterar, pesquisar e modificar a árvore de análise;
- lxml, uma biblioteca Python que permite fácil manuseio de arquivos XML e HTML;
- requests, uma biblioteca HTTP elegante e simples para Python, construída de raiz para ser fácil de usar;
- pandas, uma ferramenta de manipulação e análise de dados de código aberto rápida, poderosa, flexível e fácil de usar;
- jupyter, um meta-pacote o qual traz (como dependências) o sistema Jupyter (em especial os cadernos), o *kernel* IPython e outros.

Com estes pacotes temos um mapa de actuação para esta etapa de projecto (de webscraping): abrir um ambiente virtual (e instalar bibliotecas), abrir um caderno de Jupyter, importar as bibliotecas das quais usamos "requests" para ir buscar a nossa página, fazer *parsing* da pagina via "lxml", criar um objecto "Soup" com o conteúdo *parsed*, fazer *scraping* e iterar pelos *scrapes* dos quis criamos *dataframes* de "pandas" e exportamos os mesmos em .csv para uso futuro.

3 TripAdvisor

O TripAdvisor é uma empresa americana de viagens *online* que opera *web* e *mobile apps* com conteúdo *user generated* e um "website" de comparação de preços, dos quais se pode fazer com hotéis, locais atractivos (como monumentos, parques, museus, etc..) e restauração.

Este como sendo um produto/serviço que oferece acesso a três categorias distintas (hotéis, restaurantes e atracções), foi divido em três partes que representam as três categorias.

Este "website" é conhecido pelas suas tentativas de dificultar os processos de *scraping*, o qual foi observado, mas resolvido a custo de tempo. Felizmente encontramos um "website" chamado *"Worth Web Scraping"* o qual nos mostrou como fazer na página de hotéis do TripAdvisor o *scraping* da tabela de referência[1] e dos *reviews*[2].

3.1 Estratégia

Após um *scouting* inicial às páginas das três categorias, foi observado as seguintes peculiaridades:

- as páginas das três categorias são diferentes no seu *layout* e organização;
- os nomes das classes nos "span", "div" e outros elementos são *random generated* e mudam de acordo com a sessão aberta ou cookie;
- existem representações repetidas, estes são os *posts sponsored* pelo próprio "website";
- as "subpáginas" que nos retornam os *reviews*, são de comprimentos diferentes de acordo com a categoria de *listing*;
- as "subpáginas" que nos retornam os *reviews*, usam múltiplos de cinco ou dez na *query*;
- as "subpáginas" que nos retornam os *reviews* mostram por defeito os que estão na linguagem referente ao dominio (.pt, .com, etc..) sem *query parameter* para alterar,
- *links* com *query parameters* que representem uma "subpágina" não existente não dão erro 404 (Page not found), mas redireccionam para a primeira;
- quando tentamos extrair o total de *reviews* apenas conseguimos o total dos totais e não o total por linguagem, impedindo assim de fazer uma conta para saber qual o múltiplo de cinco ou dez que seria a ultima "subpágina".

Assim sendo, a estratégia que foi usada, embora extremamente má em termos de tempo despendido e extracções redundantes, era a única que assegurava que se conseguia extrair todos os *reviews*. Essa estratégia foi:

- criar uma lista de *links* para 200 ou 400 "subpáginas" (de acordo com o *listing* daquela categoria com mais *reviews* em português);
- extrair incluindo os repetidos para um *array/list/arraylist*;
- usar compreensão de listas através de *sets*/dicionários/*tuples* que possam ser ordenados para remover repetidos e não perder ordem;
- transpor esses dados para um *dataframe* de "pandas" e exportá-lo para .csv para uso futuro.

3.2 Desenvolvimento

Aqui iremos detalhar o processo longo do *webscraping* da plataforma TripAdvisor e as suas três principais categorias.

3.2.1 Hotéis

Para o desenvolvimento do *webscraping* dos Hotéis de Beja, foi aberto um caderno de Jupyter no qual começamos por fazer o *import* das bibliotecas e desactivar o aviso da falta de certificado SSL (após a introdução do trabalho em Inglês).

```
import urllib3
urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)
import requests
from bs4 import BeautifulSoup as soup
import pandas as pd
```

Seguidamente, foi feita a configuração do *request* onde qual fazemos download da página *web* pretendida. Estes *headers* foram extraídos do *browser* do computador usado, Microsoft Edge (Chromium).

Após fazer *request* e verificar o status code (vazio ou 200 para OK), foi criado um objecto "Soup" com o parsing (via "lxml") da página *requested*.

```
headers = {
  'Access-Control-Allow-Origin': '*',
  'Access-Control-Allow-Methods': 'GET',
  'Access-Control-Allow-Headers': 'Content-Type',
  'accept': '*/*',
  'accept-encoding': 'gzip, deflate, br',
  'accept-language': 'en-GB,en;q=0.9,en-US;q=0.8',
  'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
  ↪ (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36 Edg/96.0.1054.29'}
url = "https://www.tripadvisor.pt/Hotels-g189102-Beja_Beja_District_Alentejo-Hot
↪ els.html"
req = requests.get(url,headers=headers,timeout=5,verify=False)
req.status_code
bsobj = soup(req.content, 'lxml')
```

Para a criação da tabela de referência dos hotéis fazemos um grupo de ciclos que nos vão fazer *scrape* aos nomes, *ratings*, número total de *reviews* e preços. Sendo que este número de *reviews* não nos vale de muito tal como previamente referido.

```
hotel = []
for name in bsobj.findAll('div',{'class':'listing_title'}):
  hotel.append(name.text.strip())
print(len(hotel))
ratings = []

for rating in bsobj.findAll('a',{'class':'ui_bubble_rating'}):
  ratings.append(rating['alt'])
print(len(ratings))
reviews = []

for review in bsobj.findAll('a',{'class':'review_count'}):
  reviews.append(review.text.strip())
print(len(reviews))
price = []

for p in bsobj.findAll('div',{'class':'price-wrap'}):
  price.append(p.text.replace('euros','').strip())
print(len(price))
```

Sendo que agora podemos simplesmente através destes *arrays* criados fazer um *dataframe* der "pandas" via um dicionário de Python com os variados *pandas* referidos. Seguidamente exportamos o *dataframe* para um ficheiro .csv.

```
d1 = {'Hotel':hotel[:length],'Estrelas':ratings[:length],'Avaliacoes':reviews[:l
↪  ength],'Preco':price[:length]}
df = pd.DataFrame.from_dict(d1)
print(df)
df.to_csv('listtable.csv')
```

O qual gerou uma tabela de hotéis como referência, a qual é exemplificada aqui.

	Hotel	Estrelas	Preço
0	Pousada Convento Beja	4,5 de 5 bolhas	100
1	Vila Galé Clube de Campo	4,5 de 5 bolhas	99
2	Herdade dos Grous	4,5 de 5 bolhas	130
3	Hotel Bejense	4 de 5 bolhas	63
4	Herdade do Vau	4,5 de 5 bolhas	85
5	Herdade Da Diabroria	4 de 5 bolhas	76
6	Hotel Melius	4 de 5 bolhas	75
7	BejaParque Hotel	3,5 de 5 bolhas	80
8	Hotel São Domingos	4 de 5 bolhas	55
9	Maria‘s Guesthouse	5 de 5 bolhas	85 81
10	Hotel Santa Bárbara	4 de 5 bolhas	59
11	Beja Hostel	3,5 de 5 bolhas	50
12	Império romano guest house	4,5 de 5 bolhas	67
13	Guest House Stories	5 de 5 bolhas	50
14	Monte Das Beatas - Alojamento Local	5 de 5 bolhas	50
15	Hospedaria Santa Maria	3 de 5 bolhas	36
16	Aljana Guest House	4,5 de 5 bolhas	99
17	Sesmarias Turismo Rural & SPA	5 de 5 bolhas	90
18	Monte da Floresta B&B	3,5 de 5 bolhas	80 76
19	Casa de Pedrogao	4 de 5 bolhas	54 51
20	Hotel Santa Clara	5 de 5 bolhas	58
21	Herdade das Barradas da Serra	5 de 5 bolhas	125
22	Paradise In Portugal	5 de 5 bolhas	79
23	Villa Extramuros	5 de 5 bolhas	
24	Albergaria Do Calvario	5 de 5 bolhas	

Tabela 1: Tabela de hotéis referência do TripAdvisor da zona de Beja

Mesmo que o numero total de *reviews* não nos seja relevante o "HTML tag" onde é retirado contem um "href" com uma parte do *link* que nos possibilita (criar o *link* inteiro e) visitar a pagina de *reviews*.

Essas páginas têm determinadas restrições faladas nas secções anteriores e a sua solução. A qual aqui em baixo representada, cria uma enormidade de *links* por local.

```
links = []
for review in bsobj.findAll('a',{'class':'review_count'}):
  try:
    a = review['href']
    a = 'https://www.tripadvisor.pt'+ a
    c = a[:(a.find('Reviews')+7)] + '' + a[(a.find('Reviews')+7):]
    links.append(c)
    for i in range(5,1000,5):
        b = a[:(a.find('Reviews')+7)] + '-or' + str(i) +
        ↪  a[(a.find('Reviews')+7):]
        links.append(b)
  except:
    pass
```

Dos quais *links* agora serão *scraped* (incluindo os *reviews* repetidos) e seguidamente tratados (remoção de repetidos) indo seguidamente para um (dicionário e transformado num) *dataframe* de "pandas", o qual é imediatamente exportado com o número do hotel referente na tabela de referência.

```
count = 0
count2 = 0
allreviews = []
for link in links:
    try:
        html2 = requests.get(link,headers=headers)
        bsobj2 = soup(html2.content,'lxml')
        for r in bsobj2.findAll('q'):
            try:
                rev = r.span.text.strip()
                allreviews.append(rev + '\n')
            except:
                pass
    except:
        pass
    count += 1
    if count == 200:
        seen = set()
        allreviews = [item for item in allreviews if not(tuple(item) in seen or
        ↪  seen.add(tuple(item)))]
        dfr = pd.DataFrame.from_dict({'Avaliacoes':allreviews})
        print(dfr)
        dfr.to_csv('hotel' + str(count2) + '.csv')
        allreviews = []
        count = 0
        count2 += 1
```

O resultado pode ser visto numa secção seguinte que mostrará resultados de todos os *scrapes* do TripAdvisor.

3.2.2 Atracções

Para o desenvolvimento do *webscraping* das Atracções de Beja, foi aberto um caderno de Jupyter no qual começamos por fazer o *import* das bibliotecas e desactivar o aviso da falta de certificado SSL (após a introdução do trabalho em Inglês).

```
import urllib3
import requests
import re
from bs4 import BeautifulSoup as soup
import pandas as pd
urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)
```

Seguidamente, foi feita a configuração do *request* onde qual fazemos download da página *web* pretendida. Estes *headers* foram extraídos do *browser* do computador usado, Microsoft Edge (Chromium).

Após fazer *request* e verificar o status code (vazio ou 200 para OK), foi criado um objecto "Soup" com o parsing (via "lxml") da página *requested*.

```
headers = {
  "Access-Control-Allow-Origin": "*",
  "Access-Control-Allow-Methods": "GET",
  "Access-Control-Allow-Headers": "Content-Type",
  "accept": "*/*",
  "accept-encoding": "gzip, deflate, br",
  "accept-language": "en-GB,en;q=0.9,en-US;q=0.8",
  "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
  ↪  (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36 Edg/96.0.1054.29",
}
url = (
    "https://www.tripadvisor.pt/Attractions-g189102-Activities-Beja_Beja_Distric
    ↪  t_Alentejo.html"
)
req = requests.get(url, headers=headers, timeout=5, verify=False)
req.status_code
bsobj = soup(req.content, "lxml")
```

Para a criação da tabela de referência das atracções fazemos um ciclo que nos vão fazer *scrape* aos nomes.

```
place = []
for name in bsobj.findAll("span", {"name": "title"}):
    place.append(re.sub(r"\b\d+\b", "", name.text.strip())[2:])
print(place)
```

Sendo que agora podemos simplesmente através destes *arrays* criados fazer um *dataframe* der "pandas" via um dicionário de Python com os variados *pandas* referidos. Seguidamente exportamos o *dataframe* para um ficheiro .csv.

```
length = len(place)
d1 = {
    "Attraction": place[:length]
}
df = pd.DataFrame.from_dict(d1)
print(df)
df.to_csv("listtable.csv")
```

O qual gerou uma tabela de atracções como referência, a qual é exemplificada aqui.

	Attraction
0	Castelo de Beja
1	Museu Regional de Beja (Museu Rainha D. Leonor)
2	Nucleo Museologico
3	Casa de Santa Vitória
4	Igreja de Nossa Senhora Dos Prazeres E Museu Episcopal
5	Ruínas Romanas de Pisões
6	Museu Visigotico–Igreja de Santo Amaro
7	Jardim Gago Coutinho e Sacadura Cabral
8	Sé Catedral de Beja / Igreja de São Tiago
9	Museu Jorge Vieira/Casa Das Artes
10	Porta de Évora - Arco romano de Beja
11	Pelourinho de Beja
12	Igreja de Santa Maria da Feira
13	Igreja do Salvador
14	Igreja da Misericórdia
15	Estátua da Rainha Dona Leonor
16	Igreja do Carmo
17	Ermida de Santo André
18	Igreja de Nossa Senhora do Pé da Cruz
19	Ermida de Santo Estêvão
20	Bairro da Mouraria
21	Janela Manuelina
22	Arcadas da Praça da República
23	Arco das portas de Avis
24	Monumento ao Prisioneiro Político Desconhecido
25	Palácio dos Maldonados
26	Convento de Santo António em Beja
27	Colégio dos Jesuítas de Beja
28	Piscina Descoberta Municipal de Beja
29	Passo da Rua da Ancha

Tabela 2: Tabela de atracções referência do TripAdvisor da zona de Beja

Agora um ciclo que retira os “HTML tag” onde contem um “href” com uma parte do *link* que nos possibilita (criar o *link* inteiro e) visitar a pagina de *reviews*.

Essas páginas têm determinadas restrições faladas nas secções anteriores e a sua solução. A qual aqui em baixo representada, cria uma enormidade de *links* por local.

```
links = []
for review in bsobj.findAll("a", {"href": re.compile(r'#REVIEWS')}):
    try:
        a = review["href"]
        a = "https://www.tripadvisor.pt" + a
        c = a[: (a.find("Reviews") + 7)] + "" + a[(a.find("Reviews") + 7):]
        links.append(c)
        for i in range(10, 4000, 10):
            b = (
                a[: (a.find("Reviews") + 7)]
                + "-or"
                + str(i)
                + a[(a.find("Reviews") + 7):]
            )
            links.append(b)
    except:
        pass
```

Dos quais *links* agora serão *scraped* (incluindo os *reviews* repetidos e excepto os que contem "desde" e "euros") e seguidamente tratados (remoção de repetidos) indo seguidamente para um (dicionário e transformado num) *dataframe* de "pandas", o qual é imediatamente exportado com o número do atracção referente na tabela de referência.

```
count = 0
count2 = 0
allreviews = []
for link in links:
    try:
        html2 = requests.get(link, headers=headers)
        bsobj2 = soup(html2.content, "lxml")
        for r in bsobj2.findAll("span", {"class": "NejBf"}): # as of 7Dez,
        ↪ because in 6Dez it was "class": "cSoNT"; tripadvisor, i hope you go
        ↪ bankrupt
            for rev in r:
                try:
                    rv = rev.text
                    if "desde" not in rv and "euros" not in rv:
                        allreviews.append(rv + "\n")
                except:
                    pass
    except:
        pass
    count += 1
    if count == 400:
        seen = set()
        allreviews = [
            item
            for item in allreviews
            if not (tuple(item) in seen or seen.add(tuple(item)))
        ]
        dfr = pd.DataFrame.from_dict({"Avaliacoes": allreviews})
        print(dfr)
        dfr.to_csv("place" + str(count2) + ".csv")
        allreviews = []
        count = 0
        count2 += 1
```

O resultado pode ser visto numa secção seguinte que mostrará resultados de todos os *scrapes* do TripAdvisor.

3.2.3 Restaurantes

Para o desenvolvimento do *webscraping* dos Restaurantes de Beja, foi aberto um caderno de Jupyter no qual começamos por fazer o *import* das bibliotecas e desactivar o aviso da falta de certificado SSL (após a introdução do trabalho em Inglês).

```
import urllib3
import requests
import re
from bs4 import BeautifulSoup as soup
import pandas as pd
urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)
```

Seguidamente, foi feita a configuração do *request* onde qual fazemos download da página *web* pretendida. Estes *headers* foram extraídos do *browser* do computador usado, Microsoft Edge (Chromium).

Após fazer *request* e verificar o status code (vazio ou 200 para OK), foi criado um objecto "Soup" com o parsing (via "lxml") da página *requested*.

```
headers = {
  "Access-Control-Allow-Origin": "*",
  "Access-Control-Allow-Methods": "GET",
  "Access-Control-Allow-Headers": "Content-Type",
  "accept": "*/*",
  "accept-encoding": "gzip, deflate, br",
  "accept-language": "en-GB,en;q=0.9,en-US;q=0.8",
  "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
  ↪  (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36 Edg/96.0.1054.29",
}
url = "https://www.tripadvisor.pt/Restaurants-g189102-Beja_Beja_District_Alentej
↪  o.html"
req = requests.get(url, headers=headers, timeout=5, verify=False)
req.status_code
bsobj = soup(req.content, "lxml")
```

Para a criação da tabela de referência das atracções fazemos um ciclo que nos vão fazer *scrape* aos nomes e as partes de *href* contidas nos "href" para a criação dos links dos *reviews*.

```
place = []
prelinks = []
for name in bsobj.findAll("div", {"class": "OhCyu"}):
    place.append(re.sub(r"\b\d+\b", "", name.span.text.strip())[2:])
    prelinks.append(name.span.a["href"])
```

Sendo que agora podemos simplesmente através destes *arrays* criados fazer um *dataframe* der "pandas" via um dicionário de Python com os variados *pandas* referidos. Seguidamente exportamos o *dataframe* para um ficheiro .csv.

```
length = len(place)
d1 = {"Restaurant": place[:length]}
df = pd.DataFrame.from_dict(d1)
print(df)
df.to_csv("listtable.csv")
```

O qual gerou uma tabela de restaurantes como referência, a qual é exemplificada aqui.

	Restaurant
0	Íntimo restaurante
1	Restaurante Dom Dinis
2	Herdade dos Grous Restaurante
3	Adega Tipica Restaurante
4	Bifanas do Márinho
5	Pulo Do Lobo
6	Toi Farois
7	Restaurante Sabores Do Monte
8	Pizzaria Milano
9	Pizaria e Restaurante Mediterrâneo Dona Maria
10	Frango à Guia
11	Casa de Pasto - Tem Avondo
12	Restaurante Espelho D'Água
13	Hamburgueria da Avenida
14	O Arbitro
15	Adega do Castelo - Museu do Vinho
16	Pinguinhas - Tapas e Petiscos
17	Taberna A Pipa
18	Luiz Da Rocha
19	Restaurante Pousada São Francisco
20	Malhadinha Restaurant Wine & Gourmet
21	A Ilha Do Peixe
22	Sabores do Campo
23	Art Deco
24	Os Bolos da Marisa
25	Restaurante Típico O Arcada
26	A Merenda Snack Bar Restaurante
27	A Pracinha
28	Restaurante Alcoforado
29	Restaurante A Lareira

Tabela 3: Tabela de restaurantes referência do TripAdvisor da zona de Beja

Agora um ciclo que vai buscar os as partes de *links* onde do ciclo anterior que nos possibilita (criar o *link* inteiro e) visitar a pagina de *reviews*.

Essas páginas têm determinadas restrições faladas nas secções anteriores e a sua solução. A qual aqui em baixo representada, cria uma enormidade de *links* por local.

```
links = []
for pre in prelinks:
    try:
        a = "https://www.tripadvisor.pt"
        c = a + "" + pre
        d = c[: (c.find("-Reviews-") + len("-Reviews-") - 1)]
        e = c[(c.find("-Reviews-") + len("-Reviews-") - 1) :]
        links.append(c)
        for i in range(10, 4000, 10):
            b = d + "-or" + str(i) + e
            links.append(b)
    except:
        pass
```

Dos quais *links* agora serão *scraped* (incluindo os *reviews* repetidos) e seguidamente tratados (remoção de repetidos) indo seguidamente para um (dicionário e transformado num) *dataframe* de "pandas", o qual é imediatamente exportado com o número do restaurante referente na tabela de referência.

```
count = 0
count2 = 0
allreviews = []
for link in links:
    try:
        html2 = requests.get(link, headers=headers)
        bsobj2 = soup(html2.content, "lxml")
        for r in bsobj2.findAll("p", {"class": "partial_entry"}):
            for rev in r:
                try:
                    rv = rev.text.strip()
                    allreviews.append(rv + "\n")
                except:
                    pass
    except:
        pass
    count += 1
    if count == 400:
        seen = set()
        allreviews = [
            item
            for item in allreviews
            if not (tuple(item) in seen or seen.add(tuple(item)))
        ]
        dfr = pd.DataFrame.from_dict({"Avaliacoes": allreviews})
        dfr.to_csv("restaurant" + str(count2) + ".csv")
        print(dfr)
        allreviews = []
        count = 0
        count2 += 1
```

O resultado pode ser visto numa secção seguinte que mostrará resultados de todos os *scrapes* do TripAdvisor.

3.3 Resultados

Aqui apresenta-se os resultados do *webscrape* do TripAdvisor. Os quais representam um exemplar dos dez primeiros *reviews* do primeiro hotel, atracção e restaurante, respectivamente.

	Avaliações
0	Excelente hotel . Pessoal da recepção e serviços de quarto muito atenciosos e prestativos - o frigobar. . .
1	Património histórico ao seu melhor nível de recuperação, renovação e utilização. Magnificas salas,como. . .
2	Chegámos depois da meia-noite e fomos recebidos com extrema simpatia! Passámos o nosso aniversário de. . .
3	Gostei muito do alojamento e da estadia. Destaco a beleza, qualidade e localização da pousada, a simpatia de. . .
4	Na reserva tinha a descrição de um tipo de quarto e foi atribuído outro. Parque infantil insuficiente e fechado.. . .
5	Boas instalações, uma óptima piscina, bons acessos e estacionamento. Um inexcedível acolhimento, simpatia e. . .
6	A Pousada de Beja é um sitio onde me sinto muito bem. Aliás não é a primeira vez que fico nesta Pousada e que por. . .
7	Meia hora pro check in. Pousada em obras , serviço de pequeno almoço terror. Marcamos cedo para não ser. . .
9	Muito boa localização. Excelente qualidade do serviço. Muito agradável a piscina e os jardins envolventes. . .
. . .	. . .

Tabela 4: Tabela representativa dos dez primeiros *reviews* (cortados) do hotel 0: “hotel0.csv”

	Avaliações
0	Muito bom.
1	Castelo bem conservado, com torre de menagem altiva e exuberante, pena fechar a horas proibitivas. Pode-se dar a. . .
3	Castelo bonito numa zona central de Beja. Não se paga para entrar nem para andar pelas muralhas. A vista é bonita. . . .
4	
5	No entanto, apesar de existirem pessoas dedicadas, não se vende qualquer tipo de recordações.
6	Também não é possível subir à torre o que foi uma desilusão.
7	A rever.
8	Bem preservado
9	É central e tem uma torre bem alta!
. . .	. . .

Tabela 5: Tabela representativa dos dez primeiros *reviews* (cortados) da atracção 0: “attarction0.csv”

	Avaliações
0	Num bairro de Beja encontra-se este restaurante com esplanada e sala mediana. Carta com muitas sugestões de entradas. . .
1	Não se deixem intimidar pelo aspecto do restaurante. Comida ao nível de uma estrela Michelin. Bem confecionada. . .
2	duas pessoas, embora sendo individuais. Serviço simpático e pronto. O espaço não é condizente com a delícia da comida. . .
3	Mais
4	Cozinhar divinamente, sem dúvida é uma arte! Nível de estrela Michelin e preço normal! Só retiraria a TV da sala. . .
5	Restaurante com um ementa vasta e recheada de produtos de qualidade, as vieiras, o biqueirão são exemplos de. . .
6	com uma agradável surpresa, nesta época tem um crunble de marmelo top
7	Fomos atendidos por Marcelo, grande conhecedor dos pontos turísticos e históricos de Beja. Os pratos tem ótima. . . .
8	Simplesmente top.... comida excelente... funcionários muito simpáticos.... e local muito agradável para um bom. . .
9	Comida e quantidade excelente. Preço adequado ao que é oferecido. Só é pena a demora mas é compensada pelo. . .
. . .	. . .

Tabela 6: Tabela representativa dos dez primeiros *reviews* (cortados) do restaurante 0: “restaurant0.csv”

4 Booking

4.1 Estratégia

Para a realização do "web-scrapping" no "website" da Booking.com, inicialmente foi necessário a filtragem pelos hotéis apenas na localidade de Beja, uma vez ser o local que o grupo em conjunto decidiu optar para realizar todas as pesquisas num sítio em comum. Após ter o Booking a apresentar todos os resultados para os hotéis de Beja, foi recolhido o link que redirecciona especificamente para esses resultados. Para aceder ás informações específicas de cada elemento da página e mais tarde aceder aos mesmos para retirar a informação pretendida, foi usado a ferramenta de "inspeccionar a página" e assim descobrir os nomes das classes e todos os outros elementos que continham conteúdo importante para o projecto[3], como o nome dos hotéis, preço, classificação, número de comentários e alguns outros detalhes que pudessem ser úteis.

Em seguida foi necessário realizar o "web-scrapping" das *reviews* de cada hotel, a realização desta parte foi um pouco mais difícil uma vez que para as *reviews* serem bem recolhidas era fulcral que o "web-scrapping" fosse realizado usando outro link, ou seja, foi retirado do site o prefixo de um novo link que seria o "https://www.booking.com/reviews/pt/hotel/" e baseando nos hotéis já retirados foi colocado o nome de cada um á frente do mesmo, criando assim um novo link que seria usado na realização do "web-scrapping" após a criação de um novo link para cada hotel, os processos foram semelhantes aos anteriormente feitos.

Para finalizar, os resultados foram todos guardados em ficheiros .csv para uma mais fácil visualização.

4.2 Desenvolvimento

Aqui detalha-se o processo de desenvolvimento do "webscraping" do "website" Booking.

4.2.1 Hotéis

Inicialmente foi feita a filtragem de apenas os hotéis de Beja.

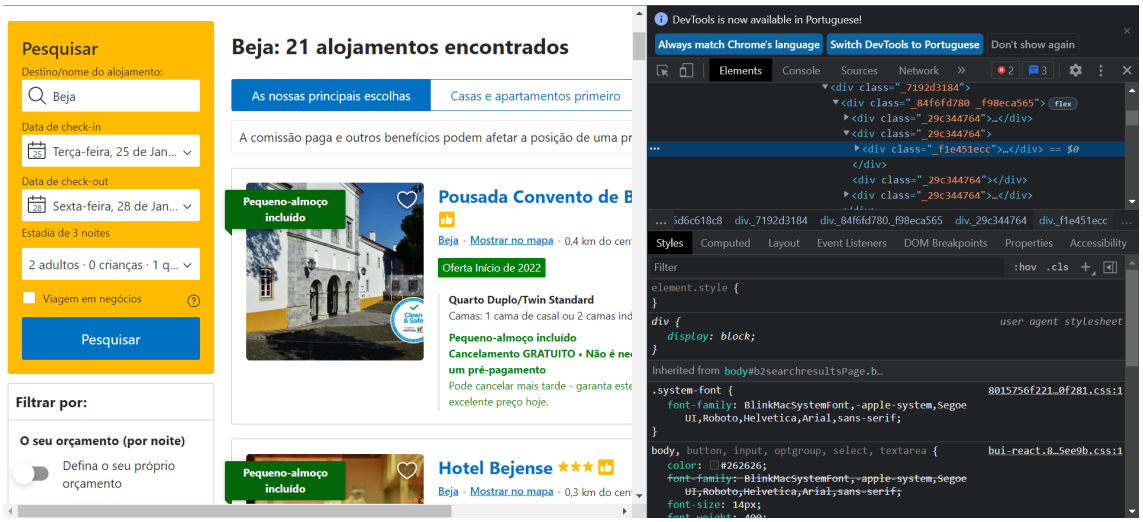

Figura 1: Site Booking.com ao usar a ferramenta inspecionar

No código foi implementado as bibliotecas BeautifulSoup para facilitar a tarefa de realizar o "web-scraping".

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

A partir do "website" ao inspeccionar a página era possível retirar os *headers* que eram valores necessários na realização do "web-scrapping". Também é realizado o pedido HTTP e juntou-se a informação com a biblioteca "BeautifulSoup".

```
headers = {
  'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit...'}
url = "https://www.booking.com/searchresults.pt-pt.html?..."
response = requests.get(url, headers=headers)
soup = BeautifulSoup(response.content, 'lxml')
```

Foram criados diferentes *arrays* para receber as informações e posteriormente colocada a respectiva informação em cada um deles.

```
hotel = []
badge = []
title = []
reviews = []
price = []
for item in soup.select('.fb3c4512b4'):
    try:
        hotel.append(item.select('.fde444d7ef')[0].get_text().strip())
        badge.append(item.select('._9c5f726ff')[0].get_text().strip())
        title.append(item.select('._192b3a196')[0].get_text().strip())
        reviews.append(item.select('._1e6021d2f')[0].get_text().strip())
        price.append(item.select('._e885fdc12')[0].get_text().strip())
    except Exception as e:
        print('')
```

Devido a alguns "arrays" conterem mais informação, possivelmente devido a algum tipo de informação adicional que possa estar em algum hotel especificamente, para prevenir erros, foram reduzidos ao tamanho do *array* mais curto.

Por fim todos os resultados contidos nos "arrays" foram guardados num ficheiro .csv denominado "listtable.csv".

```
d1 = {'Hotel': hotel[:length], 'Classificacao': badge[:length],
    'Suma': title[:length], 'Avaliacoes': reviews[:length], 'Preco':
    ↪ price[:length]}
df = pd.DataFrame.from_dict(d1)
print(df)
df.to_csv('listtable\text{.csv}')
```

Construção dos links para realizar o "web-scraping" das "reviews" de cada hotel.

```
reviews_links = []
for link in soup.findAll('a', {'class': 'fb01724e5b'}):
    a = link['href']
    hotel = a.split('/')[5].split('?')[0]
    a = 'https://www.booking.com/reviews/pt/hotel/' + hotel
    reviews_links.append(a)
```

Foi realizado o pedido “HTTP” e juntado á biblioteca “BeautifulSoup” para aceder ás “reviews” de cada site e todos os valores foram salvos no formato .csv.

```
count = 0
allreviews = []
for link in reviews_links:
    try:
        response2 = requests.get(link, headers=headers)
        soup2 = BeautifulSoup(response2.content, 'lxml')
        for r in soup2.findAll('span', {'itemprop': 'reviewBody'}):
            try:
                rev = r.text
                allreviews.append(rev + '\n')
            except:
                pass
    except:
        pass
    count += 1
    if allreviews != []:
        seen = set()
        allreviews = [item for item in allreviews if not(
            tuple(item) in seen or seen.add(tuple(item)))]
        dfr = pd.DataFrame.from_dict({'Avaliacoes': allreviews})
        print(dfr)
        dfr.to_csv('hotel' + str(count) + '\text{.csv}')
        allreviews = []
```

No final, temos esta tabela representativa do hotéis *scraped* ordenada e representativa dos *scrapes* “hotelXX.csv”.

	Hotel	Classificação	Preço
0	Hotel Bejense	“8,4”	189
1	Aljana Guest House Beja	“9,3”	330
2	BejaParque Hotel	“8,1”	255
3	Pousada Convento de Beja	“8,7”	270
4	Guest House Stories	“8,7”	135
5	Hotel Melius	“8,1”	242
6	Casa do Arco - Beja	“9,4”	210
7	Barrote Beja- Alojamento Local	“9,7”	255
8	Maria‘s Guesthouse	“9,4”	226
9	Beja Garden	“8,0”	90
10	HI Beja - Pousada de Juventude	“9,6”	327
11	Casa do Sembrano	“9,5”	330
12	Casa Idalina Villa in Beja’s beautiful countryside	“9,4”	210
13	Império Romano Guest House	“9,0”	150
14	Monte das Beatas - Alojamento Local	“8,7”	306
15	Casa do Jardim	“8,8”	270
16	Casa das Histórias	“8,9”	195
17	Casa Centro Histórico Beja - Castelo	“7,9”	180
18	Quinta do Castelo	“9,0”	214
19	Casa do Avô Zé	“8,1”	174
20	Suite na Praça da República	“8,7”	228
21	Herdade da Diabroria - Agroturismo	“8,8”	285
22	Herdade do Vau	“7,8”	270
23	Monte da Corte Ligeira	“8,3”	150

Tabela 7: Tabela com todos os hotéis de Beja retirados do Booking.com

4.3 Resultados

Aqui exemplifica-se um ficheiro .csv, “hotel18.csv”, que contem os *reviews* do hotel “Quinta do Castelo”.

	Opiniões
0	A localização é excelente assim como as condições do espaço. Local muito bem cuidado e apelativo. Fomos muito bem recebidos . . .
1	Nada digno de registo.
2	A simpatia da Sra Catarina foi fantástica. A casa e as acomodações corresponderam ás expetivas e relação qualidade preço foi perfeita . . .
3	Localização excelente, apartamento espaçoso, e totalmente equipado. O facto de ser uma construção antiga, cria um ambiente muito . . .
4	A localização é impecável mesmo no centro histórico de Beja. Casa Limpa e organizada. Dona super prestável e e simpática.
5	De noite ouve-se o barulho da rua com muita facilidade.
6	A casa está muito perto do castelo e está muito bem decorada e limpa. Muito agradável!
7	O dono não tem culpa mas a zona não é muito bem frequentada á noite
8	A casa é muito gira e funcional!
9	Casa muito agradável e bem situada. Os anfitriães são simpáticos e disponíveis. Cozinha muito bem equipada e casa de banho excelente.
10	Da localização, da relação preço/qualidade que entendo adequada.
11	Localização no centro histórico tem vantagens (centralidade) e desvantagens (dificuldades de estacionamento).
12	Localização, decoração, conforto, organização do espaço e o gosto cuidado na decoração.
13	Não tinha microondas.
14	Localização, estacionamento perto, a decoração do alojamento, ter máquina de lavar roupa deu muito geito.
15	Virado para duas ruas públicas isso condiciona a entrada de luz e de ventilação em casa pela noção de segurança; Existem algumas . . .
16	A casa em si é antiga e possui algumas divisões de formatos, alturas, níveis de pavimento e arcadas distintas, o que lhe dão um ar . . .
17	Poderia ter um microondas.
18	Muito bem localizado, anfitriões muito simpáticos. Muito interessante a manutenção de casa típica alentejana. Gostamos todos muito . . .
19	Gostei de tudo. NÃ£o tenho do q reclamar.
20	O apartamento é uma graça. Super completo, amplo, decorado com muito bom gosto e jovial! Adorei!!!!!
21	Serviço conforto e localização.
22	A limpeza poderia ser mais cuidada. Não dar para ligar a chaleira eléctrica porque o quadro não aguentava. O fogão estar rachado . . .
23	Da receção, da decoração, da temperatura da casa, o ser uma casa acolhedora situada no coração do centro histórico.
. . .	. . .
28	De tudo. Um lugar ótimo para uma família passar uns dias, a casa está bem situada e as pessoas muito simpáticas. Um lugar a repetir.

Tabela 8: Tabela de resultados de algumas “Reviews” para um dos hotéis (hotel18.csv)

5 Zomato

A Zomato é um serviço de busca de restaurantes para quem quer sair para jantar, buscar comida ou pedir em casa. A Zomato possui duas secções: guia de restaurante e blog. Previamente, havia uma secção de eventos, já descontinuada.

O guia de restaurantes Zomato permite ao usuário buscar informações relacionadas a restaurantes, bares, cafés, pubs e casa nocturnas. As informações fornecidas geralmente incluem o nome do estabelecimento, telefones de contacto, endereço, cardápio, fotografias, avaliações e mapas de localização.

5.1 Estratégia

As páginas da *web app* do Zomato usam um "paralax" de *scrolling* infinito (até não haver mais restaurantes) e as classes dos "HTML tags" mudam por sessão e/ou *rendering*, logo aqui a estratégia é literalmente fazer "download" da página web e fazer o *scrape* a partir do *parsing* dessa página.

5.2 Desenvolvimento

Aqui é representada uma aproximação do desenvolvimento deste *scraping*.

5.2.1 Restaurantes

Primeiramente foi feito o *import* das bibliotecas.

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

Depois pegando no código dos colegas como *template*, adaptou-se para usar uma página previamente descarregada.

```
headers = {
    "User-Agent": "Mozilla/5.0 (Linux; Android 6.0; Nexus 5 Build/MRA58N)
    ↪  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.93 Mobile
    ↪  Safari/537.36",
}
soup = BeautifulSoup(open(r"C:\Users\vitor\Desktop\pi2021\projecto\webscrape\scr
↪  apes\zomato\restaurantes\zomato-beja-14-dez.html", encoding="utf8").read(),
↪  'lxml')
```

Fazemos um ciclo de *scraping* dos nomes dos locais de consumo.

```
restaurant = []
for name in soup.findAll('h4',{'class':'sc-1hp8d8a-0'}):
    restaurant.append(name.text.strip())
print(len(restaurant))
```

E agora dois ciclos, um para as classes com nome gerado no *prerender* e outra pós *render*; vamos buscar os tipos/classes de restaurantes.

```
type = []
for name in soup.findAll('p',{'class':'jaKOQh'}):
    print(name)
    type.append(name.text.strip())
for name in soup.findAll('p',{'class':'kegdaG'}):
    print(name)
    type.append(name.text.strip())
print(len(type))
```

E outros dois ciclos do mesmo motivo, para ir buscar os preços.

```
price = []
for p in soup.findAll('p',{'class':'ftdqla'}):
  price.append(p.text.replace('euros para dois','').strip())
for p in soup.findAll('p',{'class':'kOONhy'}):
  price.append(p.text.replace('euros para dois','').strip())
print(len(price))
```

Com o código seguinte geramos um *dataframe* que exporta um .csv.

```
d1 = {'Restaurante':
↪  restaurant[:length],'Tipo':type[:length],'Preco':price[:length]}
df = pd.DataFrame.from_dict(d1)
print(df)
df.to_csv('listtable.csv')
```

Que gerou um ficheiro aqui representado nesta tabela:

	Restaurante	Tipo	Preço
0	Adega Típica 25 de Abril	Alentejana, Portuguesa	25
1	Dom Dinis	Bifes, Portuguesa	30
2	Sushi Alentejano	Sushi, Japonesa	35
3	Herdade dos Grous	Contemporânea, Portuguesa	60
4	Pulo do Lobo	Portuguesa	25
5	Bar Parque da Vila Beringel	Snacks, Bebidas	12
6	Figa's	Pizza, Italiana	25
7	Cervejaria Portugalo	Portuguesa, Bebidas, Petiscos	25
8	Entre Arcos	Portuguesa, Grelhados	25
9	Aperitivo	Bebidas, Portuguesa	20
10	O Arbitro	Portuguesa	25
11	Café Central	Snacks, Bebidas, Portuguesa	15
12	Gatus Cervejaria Alentejana	Alentejana, Portuguesa, Marisqueira	25
13	Hamburgueria da Avenida	Hamburgueria	25
14	Casa de Pasto O Forno	Snacks, Bebidas, Portuguesa	12
15	Tennis Courts Club	Portuguesa	25
16	Cervejaria Mira Serra	Marisqueira, Portuguesa	40
17	Espelho d'Água	Portuguesa	25
18	TEM Avondo	Portuguesa, Alentejana	25
19	Menau	rtuguesa	25
20	Toy Faróis	Portuguesa, Grelhados	25
21	Taberna A Pipa	Alentejana, Portuguesa, Petiscos	25
22	Pizzeria Milano	Pizza, Italiana, Portuguesa	25
23	Dona Maria Deck	Portuguesa, Petiscos, Snacks	25
24	O Alemão	Portuguesa, Petiscos, Alentejana	25
25	Deliciosa Alvorada	Portuguesa, Alentejana	25
26	Moments IPDJ	Portuguesa	20
27	Bar Regional	Bebidas, Snacks	6
28	Sushizzaria	Sushi, Hamburgueria, Pizza	30
...	...	...	...
155	Vegetariano	Vegetariana	25

Tabela 9: Tabela dos restaurantes

Agora, pelo mesmo motivo, dois ciclos; que vão buscar os links das páginas dos *reviews*.

```
reviews_links = []
for link in soup.findAll('a', {'class': 'ieKty'}):
    a = link['href']
    reviews_links.append(a.replace('/info', '/reviews'))
for link in soup.findAll('a', {'class': 'jjSAcU'}):
    a = link['href']
    reviews_links.append(a.replace('/info', '/reviews'))
```

O qual extraímos todos os "tags" de parágrafos porque que sempre que se corria o código gerava uma classe nova... Logo, estas extracções desapontantes, vão sofrer ETL.

```
count = 0
allreviews = []
for link in reviews_links:
    try:
        response2 = requests.get(link, headers=headers)
        soup2 = BeautifulSoup(response2.content, 'lxml')
        for r in soup2.findAll('p'): # sempre a mudar a class, vai sofrer ETL
            try:
                rev = r.text
                # print(rev)
                allreviews.append(rev + '\n')
            except:
                pass
    except:
        pass
    count += 1
    if allreviews != []:
        seen = set()
        allreviews = [item for item in allreviews if not(
            tuple(item) in seen or seen.add(tuple(item)))]
        dfr = pd.DataFrame.from_dict({'Avaliacoes': allreviews})
        print(dfr)
        dfr.to_csv('restaurante' + str(count) + '.csv')
        allreviews = []
```

5.3 Resultados

Os resultados deste *scrape* foram desapontantes no minimo devido à infeliz *random generated* nome da classe, que é gerado por cada vez que se usa a página. Estes resultados vão sofrer muito ETL posterior.

	Avaliações
...	...
5	"Matilde Almeida"
6	"Há 4 meses"
7	"Um restaurante de excelente qualidade. A deliciosa comida típica de Beja e muito bem servida... "
8	"0 Votes for helpful, 0 Comentários"
9	"CláudiaSimões"
10	"Há 6 meses"
11	"Desde que venho passar férias a Beja que comecei a vir a este restaurante... "
...	...

Tabela 10: Tabela de resultados do "restaurante1.csv"

6 Próximos Passos

Na seguinte fase de trabalho o grupo irá ter que começar a desenvolver os métodos dos quais criaremos um modelo de *machine learning* para analise de sentimento e extracção de tópicos e *keywords* para associar *reviews* e *listings*.

Para tal é necessário previamente usar processos de ETL *(extract, transform, load)* do trabalho já realizado, uma das tarefas será a adaptação de todo o conteúdo textual já armazenado e reparação de informação corrupta e remoção de informação.

7 Conclusão

Concluindo, este trabalho, de forma análoga e pessoal, serviu para aprender o que realmente é o "web-scraping" e todos os usos que ele pode ter para a recolha de dados estruturados da web; já da forma curricular, serviu do primeiro passo, de acesso à informação necessária para a analise sentimental e textual sobre o turismo rural sul-alentejano.

Para além de aprender o que ele realmente é, também tivemos a oportunidade de trabalhar com ele e usá-lo num caso prático.

Houveram algumas dúvidas principalmente para retirar algumas informações de alguns dos "websites" trabalhados e também na criação do ambiente virtual, porem todas as duvidas foram superadas graças ao trabalho em equipa e pesquisas online, para alem da ajuda da docente responsável pelo projecto.

Por fim, achamos que o balanço desta parte do trabalho tenha sido bastante positiva.

8 Webgrafia

[1] M. Kamal, "How to scrape tripadvisor hotels data using python," Jan 2021. [Online]. Available: https://www.worthwebscraping.com/how-to-scrape-tripadvisor-hotels-data-using-python/

[2] Kamal, "Scrape tripadvisor reviews using python," Jul 2021. [Online]. Available: https://www.worthwebscraping.com/scrape-tripadvisor-reviews-using-python/

[3] M. Ganesan, "Scraping hotel listings from booking.com using python and beautiful soup | proxies api," Sep 2020. [Online]. Available: https://www.youtube.com/watch?v=PRkOFgNAkio