

TripAdvisor

Webscraping

Imports

First we start with the imports. We need essentially three (or four) main libraries to work this out; these are:

- requests (to fetch the website)
- lxml (a faster html parser to speed bs4)
- bs4 (a.k.a beautiful soup, a web scraping library)
- pandas (a maths oriented data(set) manipulation library)

Since requests uses urllib3 as a dependency, we can import it first to configure it to suppress the annoying warning about the "insecure" connection (lack of SSL).

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

Request configuration

We need to configure our request, specially in this case, since TripAdvisor won't send us a webpage if we at least not try to emulate a real browser. First we configure our headers, ripping the main headers from our browser, as seen in the Developer tools (F12) in Chromium (we used the new Microsoft Edge).

Then we request the webpage (Hotels in Beja, Portugal) with our headers attached, a timeout to stop if it takes too long (something is wrong), and verification is disabled (SSL). If the status code is OK (200), doesn't print.

After that we create a BeautifulSoup4 scrapable object with the html content of the page, using lxml.

```
In [ ]: 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/96.0.4690.80 Safari/537.36'
        url = "https://www.tripadvisor.pt/Hotels-g189102-Beja_Beja_District_Alentejo-Hotels.html"
        req = requests.get(url,headers=headers,timeout=5,verify=False)
        req.status_code
        bsobj = soup(req.content, 'lxml')
```

Scraping

Now we start scraping. First we start getting hotel names.

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

Then their ratings.

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

The number of reviews (they have a big issue).

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

This number is referring to the TOTAL number of reviews. However, we can only scrape a single language, that's dependent of the domain/locale (.com .pt); there's no query parameter to change either the number here, or the sorting of reviews.

Now we get the prices.

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

Some of these will be empty, since the price is gotten via phone call.

Now, we get to the weird part: Reviews. These reviews are handled by the hotel page in weird ways:

- only four shown per subpage
- each subpage is counted via a multiple of five
- any multiple of five non-existent will not give 404, but redirect to the first subpage
- language selection via scraping is non-existent, no query parameters, only radio buttons with random labels, defaults chosen via domain/locale (.com .pt)

So the strategy found is:

- create a monstrous amount of subpage links (about 200)
- scrape all reviews, even repeated via the redirect to the first subpage
- later use sets, or dictionaries to remove duplicates

That was done with this awful looking but functional code.

```
In [ ]: 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
print(links)
```

Now, we get the smallest length number of the arrays regarding to the hotels, hoping we remove some of the (repeated) sponsorships.

```
In [ ]: # bad code
length = len(price)
if length > len(reviews):
    length = len(reviews)
if length > len(hotel):
    length = len(hotel)
if length > len(ratings):
    length = len(ratings)
```

And then create the ID table of the hotels with the most basic information in a pandas DataFrame, and export that one to a .csv file that we can use in Excel, PowerBI, ML libraries like Keras, Tensorflow, SciKitLearn can use.

```
In [ ]: d1 = {'Hotel':hotel[:length], 'Estrelas':ratings[:length], 'Avaliações':reviews[:length], 'Preço':price[:length]}
df = pd.DataFrame.from_dict(d1)
print(df)
df.to_csv('listtable.csv')
```

Now the most horrible of codes presents you with the creations of various, separated .csv files with the scraped reviews, that we can use.

It iterates all the links and since there's 200 links per hotel, every 200 we use some list comprehension magic with sets to remove duplicates and export the DataFrame to a useful .csv file.

```
In [ ]: 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({'Avaliações':allreviews})
        print(dfr)
        dfr.to_csv('hotel' + str(count2) + '.csv')
        allreviews = []
        count = 0
        count2 += 1
```