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 supress the annoying warning about the "insecure" connection (lack of SSL).

In [ ]:

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

Request configuration

We need to configure our request, specially in this case, since TripAdvisor wont 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 (Attractions 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 [ ]:

```python
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/96.0.4
}
url = (
    "https://www.tripadvisor.pt/Attractions-g189102-Activities-Beja_Beja_District_Alentejo.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 attraction names.

In [ ]:

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

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 attraction page in weird ways:

- only ten shown per subpage
- each subpage is counted via a multiple of ten
- 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 monstruos amount of subpage links (about 400)

- 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", {"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
# print(links)
```

And then create the ID table of the attractions 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 [ ]:

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

Now the most terrible 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 400 links per attraction, every 400 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("span", {"class": "NejBf"}): # as of 7Dez, because in 6Dez it was "class": "cSoN
            for rev in r:
                try:
                    rv = rev.text
                    if "desde" not in rv and "€" 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({"Avaliações": allreviews})
        print(dfr)
        dfr.to_csv("place" + str(count2) + ".csv")
        allreviews = []
        count = 0
        count2 += 1
```