INFO7390: Advances Data Sci/Architecture

Naman Gupta
(NUID: 002729751)
Northeastern University
Boston Campus
gupta.naman@northeastern.edu

Introduction

Web scraping is a powerful technology that allows individuals and companies to collect useful information from websites for a variety of reasons such as data analysis, research, and application development. BeautifulSoup, a Python module, makes web scraping easier by offering a user-friendly interface for parsing and navigating HTML content. This research delves into the complexities of online scraping with BeautifulSoup, with an emphasis on data extraction from the prominent website AutoTrader. AutoTrader, among America's leading online markets for auto buyers and sellers, provides a plethora of information about vehicles for sale in a variety of categories, including Sedans, SUVs, Trucks, and Luxury cars. I will walk you through the process of sending HTTP queries, processing HTML content, and overcoming data extraction obstacles.

Methodology

The project is implemented using Python version 3.10.12, leveraging several essential libraries:

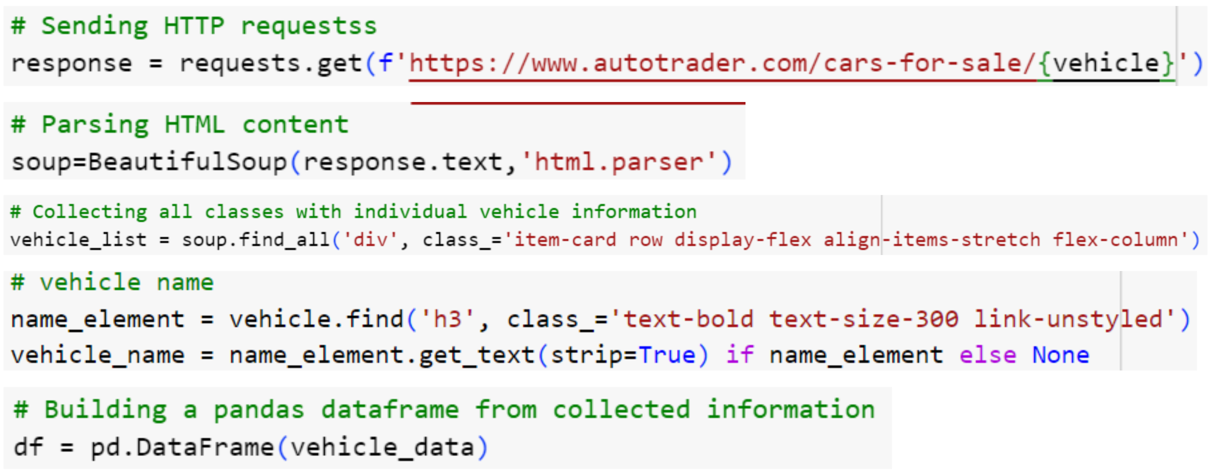
- BeautifulSoup: Used to parse HTML content from webpages.
- Requests: Employed to send HTTP requests to retrieve web data.
- Pandas: Facilitated the creation of structured data frames for scraped information.

Solution

The solution development begins with building URLs for each vehicle type's sale web pages. Then, looping over each URL to send an HTTP request for fetching HTML content from the web page. The HTML content from the webpage is parsed using BeautifulSoup. Further, the HTML content is separated into a list with each element containing the information of a particular vehicle from the web page.

This extract list is processed more to extract each element that is relevant information for the vehicle, for example, vehicle name, type, description, miles, price, etc. This processing involves storing them to be column names of the dataframe as keys of a dictionary and appending column values in the respective keys. During this processing, if any of the expected information cannot be found/unavailable in the HTML content, a null (None) value is passed to maintain the consistency of the data.

After ensuring that data for all the vehicle types is extracted, the dictionary is transformed into a pandas dataframe. I finally have our desired web-scraped data set.



```
# Sending HTTP requestss
response = requests.get(f'https://www.autotrader.com/cars-for-sale/{vehicle}')
```

```
# Parsing HTML content
soup=BeautifulSoup(response.text,'html.parser')
```

```
# Collecting all classes with individual vehicle information
vehicle_list = soup.find_all('div', class_='item-card row display-flex align-items-stretch flex-column')
```

```
# vehicle name
name_element = vehicle.find('h3', class_='text-bold text-size-300 link-unstyled')
vehicle_name = name_element.get_text(strip=True) if name_element else None
```

```
# Building a pandas dataframe from collected information
df = pd.DataFrame(vehicle_data)
```

Challenges and Outlook

Our web scraping journey encountered notable challenges:

- **Class Name Ambiguity:** In the process of extracting data, I faced the challenge of dealing with similar class names within the HTML content. To mitigate this, I adopted a systematic approach by identifying potential class names and finalizing them based on the expected content.
- **Data Structure Variability:** Across different vehicle types' web pages, I observed variations in the HTML content structure. To address this, I adapted our scraping logic to accommodate these variances dynamically.

Future Enhancements

The solution can be used for future price prediction, analyzing features of a vehicle, making recommendations for a buyer, featuring the best/beneficial deals, etc. The solution can be enhanced by increasing the size of dataset, which is possible by involving selenium in it. Selenium is a Python library that is used to automate web browser interactions from Python code. Hence, assisting BeautifulSoup in extracting many more features of every vehicle by opening web pages for each vehicle and scraping data, as well as navigating to multiple pages for every vehicle type to scrape more data.

References

- https://tedboy.github.io/bs4_doc
- https://medium.com/ymedialabs-innovation/web-scraping-using-beautiful-soup-and-selenium-for-dynamic-page-2f8ad15efe25