The Occidental Computer Science Comprehensive Project: Tutorial Report

Xintai Aaron Ao
xao@oxy.edu
Occidental College

Abstract

This report documents the tutorial completed as part of my *Occidental College Computer Science Comprehensive Project*. This report has four components: methods, evaluation, discussion, and software documentation. This report details the tutorial: Beginner's Guide to Scraping Data (Uber Eats Example), which reviews the key concepts and learned information relevant to this comprehensive project.

1 Methods

This tutorial utilizes Python to web scrape the Uber Eats in order to pull data regarding restaurants and their respective ratings, estimated delivery time and fees. This is because Python has been the leading web scraping language for the better part of a decade, according to this tutorial. The steps for this tutorial are as follows:

1. The Python *BeautifulSoup4* library will be an important tool for this project in order to web scrape the Uber Eats website. To install it, assuming the user has pip installed on their machine, run:

   ```
   pip install beautifulsoup4
   ```

2. The next step will be to import the library into your program:

   ```
   from bs4 import BeautifulSoup
   ```

3. Next, import the following at the top of your program:

   ```
   from urllib.request import Request, urlopen
   import json
   import ssl
   ```

4. In doing this, our downloaded libraries are now sorted. In this tutorial, the author uses the Burlington, Ontario Canadian Uber Eats website to reference from. Since my project revolves around Eagle Rock, Los Angeles, the area surrounding Occidental College, I used this link:
   `https://www.UberEats.com/neighborhood/eagle-rock-los-angeles-ca`

5. Now, we will retrieve the webpage contents; the following lines of code allow us to do so:

   ```
   url = "https://www.Uber Eats.com/neighborhood
       /eagle-rock-los-angeles-ca" # url for Los
        Angeles area

   # send a request to the page, using the
       Mozilla 5.0 browser header
   req = Request(url, headers={'User-Agent' : '
       Mozilla/5.0'})

   # open the page using our urlopen library
   page = urlopen(req).read()

   # use BeautifulSoup to parse the webpage
   soup = BeautifulSoup(page, 'html.parser')
   ```

   The lines above basically tells the program where to find and request a specific web page whilst mimicking the actions of a user using Mozilla 5.0. After the page is opened, it is finally parsed through by using BeautifulSoup, which does most of the difficult work for us. The next step is to grab the data we want to use for the project. Just like with solving a puzzle, in web scraping you start with your end piece of data that you want to extract from a website. In the case of my web application project, I want to find all the restaurants in Eagle Rock, Los Angeles that are on Uber Eats. Thus the next steps are:

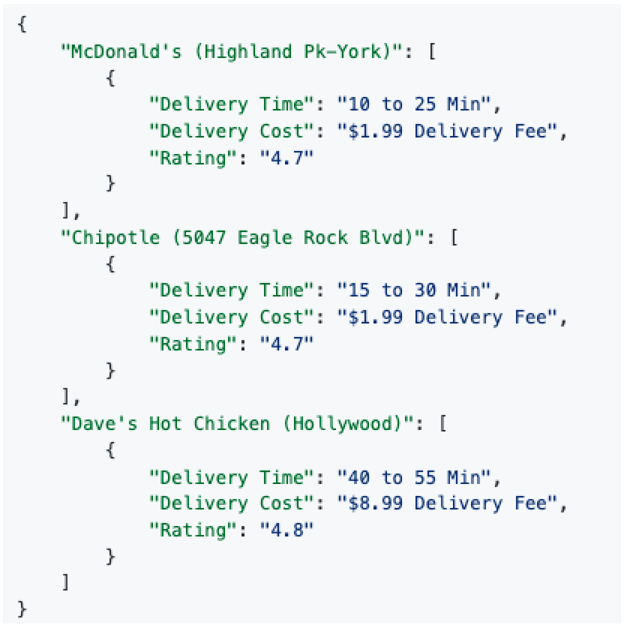
6. Assuming that you are working on an intuitive browser, right click on the name of any restaurant and hit *"Inspect"*; the front-end code should pop up allowing you to see the tags of each element. During my run through of this tutorial, I right clicked on "McDonald's (Highland Pk-York)" and clicked *"Inspect"*, which then took me to this line:

   ```
   <h3 class="h3 c4 c5 ai">McDonald's (Highland
       Pk-York)</h3>
   ```

   What this line of code means: "<h3>" is the tag used by Uber Eats to identify the specific names of the restaurant on the page.

7. Knowing this, we need to find every "<h3>" tag on our page in order to collect all of the restaurant names. This can be done with the following line of code:

   ```
   # loop through all of the restaurants on this
        page
   for x in soup.findAll(   h3   ):
        print(x.text)
   ```



```
{
    "McDonald's (Highland Pk-York)": [
        {
            "Delivery Time": "10 to 25 Min",
            "Delivery Cost": "$1.99 Delivery Fee",
            "Rating": "4.7"
        }
    ],
    "Chipotle (5047 Eagle Rock Blvd)": [
        {
            "Delivery Time": "15 to 30 Min",
            "Delivery Cost": "$1.99 Delivery Fee",
            "Rating": "4.7"
        }
    ],
    "Dave's Hot Chicken (Hollywood)": [
        {
            "Delivery Time": "40 to 55 Min",
            "Delivery Cost": "$8.99 Delivery Fee",
            "Rating": "4.8"
        }
    ]
}
```

Figure 1. Uber Eats Web Scraping Tutorial Output

8. This simple Python loop just iterates through the webpage content that the BeautifulSoup library already parsed for us. The ”findAll” method creates a Python list of each element in our object ”soup” which contains the ”<h3>” tag. Inside the loop, we’re just printing the object x’s text field. The following output should result in:

```
Charlie’s Tacos (Highland Park)
Brick & Flour (Glendale)
Dave’s Hot Chicken (Glendale)
Lil Ma ana
Tacos on The Fly (4944 York Boulevard)
The Burrito Snob (4944 York Boulevard)
Bruno’s Birria Bar (4944 York Boulevard)
```

This tutorial works not just for Uber Eats but all other major food delivery websites including Postmates, GrubHub, and DoorDash. The only difference is in typing the respective urls for each delivery app’s website during step 5 and the identifying tags used in each website’s front-end code in steps 6-8 (i.e. Uber Eats: <h3>).

2 Evaluation

This tutorial by Ben Minor presents a clear and simplistic approach to web scraping with the given example of using the Uber Eats website. The web scraping guide introduces the use of the BeautifulSoup library to users unfamiliar with web scraping. This library is especially useful to me in the context of my comprehensive project as it gives me the ability to skip the extra steps in having to code own bots in order to scrape data from websites online.

As shown in the figure above, this tutorial is able to generate an output that gives the basic information of restaurants on the Uber Eats website. This web scraped information will later become the foundation of my project, along with the implementation of more detailed scraping; this will include scraping individual restaurant menus in each website for the date on every items including costs and popularity amongst users. The most accurate way to evaluate whether or not the resulting code from this tutorial is able to output accurate real-time date from the Uber Eats website is by checking through the Uber Eats app itself and comparing the results with that of the tutorial’s code. Using the Occidental College address and typing in names of three popular restaurants nearby (McDonald’s, Chipotle, and Dave’s Hot Chicken), I was able to confirm that the results of this code were accurate and sufficient.

3 Discussion

Being as straightforward and concise this tutorial was, I believe that it has become a the perfect benchmark for my project as I continue to work on it in the future. Before coming across it, I did not have any previous experience with web scraping. I thought the process would have tedious and heavily bot-coding based, but this tutorial was able to erase those concerns. One I did come across after completing this tutorial was the subsequent attempt in trying to transfer this process onto other online delivery websites. Though I did state earlier that this tutorial was able to be replicated with the other major online delivery services, the outstanding issue relies in the problems of having to aggregate the process of web scraping multiple restaurant web pages from four different websites and storing all of the subsequently scraped data. However, this problem will be easily solvable with the help of finding another tutorial.

Another problem that I will run into when pursuing this project will be the consideration of whether or not a restaurant has a specialized agreements with the delivery platforms they benefit from. Though one way of finding out if a specific restaurant or chain only has one purveyor of online delivery is by searching the internet or contacting the respective restaurants, I believe this problem can be aggregated to a much more proficient level.

4 Software Documentation

See attached **uber−eats−scraper.py** file for the more in-depth example of this tutorial where I gather more specified restaurant data.