Tuhina Das

ISM II

Research Assessment #4

Date: September 16, 2024

Subject: Full-Stack Software Development

Assessment: Research Assessment #4 - Web Scraping Basics

At this point in my investigation into using software to mitigate mental health, I have two product ideas in mind. One of these potential ideas involves web scraping, or a process involving code to extract data from a website. I have heard of this process before, but I decided to further investigate how it worked by learning how to do it myself with Alex Freberg's beginner tutorial on web scraping using Python. While there are sites that can automatically handle web scraping for the user with little code involvement, I wanted to try using Python for this process. After all, I know web scraping can be optimized with machine learning algorithms written in Python, which could be a future branch of research for me in the near future.

Going into this tutorial, I knew that the library BeautifulSoup was a popular dependency used in the process of web scraping. However, I knew little about how it worked, which is what Freberg's tutorial mainly taught me. In this tutorial, which covered web scraping with a published domain such as Wikipedia, Freberg used Python's Requests library to make an HTTP GET request given the specific URL for a page on Wikipedia. The request returns various pieces of data regarding the provided URL, including the webpage's HTML content and the request's status code after retrieving the webpage's content – this is content I am familiar with from my previous research into full-stack web development. BeautifulSoup then takes the text from the webpage's content and parses it to locate specific objects or keywords tied with data that can be scraped from the webpage.

All of this can actually be done in fewer than ten lines of code, which I found convenient. As Freburg explains, however, the real challenge with web scraping comes from how a programmer can effectively sift through the data provided without including too much "fluff material", or extraneous HTML. For example, if a programmer is attempting to scrape data from a particular table on a page on Wikipedia, they may encounter the issue of having multiple HTML table elements on one page, skewing their results. Freburg goes on to explain that using unique Cascading Style Sheet (CSS) selectors present in the HTML tags of a particular element can help with narrowing the results of a BeautifulSoup search down.

As I watched this part of the tutorial however, I had another question: Could different pages on the same website be scraped if they appeared to follow the same format? Freburg's tutorial scraped a Wikipedia page over the largest companies by revenue in the world; I, alternatively, decided to try his method of scraping with another Wikipedia page over the largest GDPs by country in the world. Both pages had tables formatted with data; both tables, when inspected, had similar CSS class names. However, calling on the class names that Freburg used in his tutorial (which matched the class names of my Wikipedia page's table) failed to yield the same result. So, this has now led me to wonder if there is a way to scrape multiple websites without having to write a unique program for each website. This may delve into the realm of artificial intelligence. And as this has been a recurring theme in my research, this may indicate a new area of technical expertise I should consider exploring.

APA citation(s):

Freberg, A. (2023, July 11). *Scraping Data from a Real Website | Web Scraping in Python*. YouTube. Retrieved September 16, 2024, from https://youtu.be/8dTpNajxaH0?si=eNskFQc7aM4TDccb