Scraping Tool Report - Jonathon Panzera

February 20, 2020

This project folder contains two driver scripts, each specifically catered to scrape from the web article structure of their respective organizations: Infatuation Los Angeles and the New York Times. Each script is named **scrape.py** and has a corresponding JSON output file, which is overwritten each time the scripts are run. They are run independently of one another with unrelated results.

DEPENDENCIES:

This project requires use of the following software modules, and they require internet access to execute.

- **Python v3**
- **bs4 (python library)**
- **re (python library)**
- **urllink (python library)**

DESIGN CHOICES:

My selection of tools for this application was ultimately decided by the scale of the task at hand, and the limitations to the available time to complete it in. The objective was to scrape specific data fields from two static pages for one-time use, and to complete the project in under five hours.

As I had never built a web scraper tool from the "ground up" before, I chose the Beautiful Soup open-source parser framework for its ease of use and for its versatile, robust selectors. Beautiful Soup is popular with a wide community of documentation available, and is used in many projects, but it cannot make URL requests, so I included the **urllink** library for this purpose and made use of the **re** library for more complex expressions.

These choices were specific to this instance. Others considered that are worth mentioning are Scrapy, Spacy, and UiPath. Scrapy is a highly extensible, portable framework that would serve better for performance constraints and for raw data volume. Spacy is a leading Natural Language Processing project that would lend assistance in Named Entity Recognition where the desired values are not explicitly tagged in the DOM. Finally, UiPath is a robotic processing automation solution with an enormous array of user-friendly abilities designed to emulate a human and make flexible choices through text scraping or image recognition. I elected to write Python scripts to demonstrate adaptation to an unfamiliar task and ruled out UiPath as it would have been too easy for the scope of this project. Conversely, I did not have the suitable time to familiarize myself with or make use of Scrapy or Spacy.

ASSUMPTIONS:

- My project is largely free of consideration for data storage. All output is sent to a file loosely conforming to the JSON format. However, a single standard file output handle is used for all writes, so the output can easily be manipulated to conform to existing storage/database practices.
- My project is free of any consideration for version control or deployment platforms, aside from my own personal backup conventions.
- My project is aimed at demonstrating a knowledge of DOM navigation and adaptation to a specific unfamiliar task, and so does not bear intense scrutiny for concerns of the enterprise applications of data scraping. There are considerable optimizations to be made depending on the direction of a given data import endeavor I might be placed on.

LOGIC SUMMARY:

Both scripts accept a URL and pass the URL to a url-request mechanism and a Beautiful Soup HTML parser. The parser downloads the entire HTML page of the target URL, and then containers are initialized selected as closely as possible to the scope where the requested data fields reside. Each script iterates through every container, which loosely match to the individual restaurant/location reviews of each author. The requested headers and data are extracted, parsed, formatted, and organized in the output file, then the scripts terminate upon closing the output stream.

The scripts acquired these values:

Infatuation Los Angeles

Group Subtitle, Name, Related Article, Cost Index, Cuisine Type, Neighborhood, Image URL, Rating

New York Times

Entry Names, Relevant Time, Relevant Places, Place URLs, Place costs

RESULTS AND FINAL REMARKS:

I was pleased with Beautiful Soup for this task though I did encounter inconveniences resulting from not being able to use an XML and an HTML parser concurrently. Under more flexible circumstances, Scrapy would make for a more efficient light-weight enterprise solution, as it can run asynchronously with less resource consumption than other frameworks. I would make use next time of an NLP library like Spacy, especially in the New York Times article.

The effort with this article was identifying the locations of focus in each entry, as they are not all explicitly tagged or tied to a link. Because of this, I was not able to find all place or image values in this article, and I would use NER to approach such nebulously identified values. UiPath also would have handled the task with ease, though it is a much more involved solution and significantly more resource-intensive. Similarly, I was not able to extract the addresses of each restaurant in the Infatuation Los Angeles article under the given time constraints, as Beautiful Soup requires more elegant solutions to extract data from reactive Java elements.

Overall a significant portion of data was extracted successfully from both articles and the scripts are available for review. The implications of automating this kind of data inflow are considerable and there are excellent tools available to optimize efforts tangential to this one with. I am eager with multiple areas of enhancement should I be dedicated to a cause!