Project Stage 2

Members:
Phil Martinkus and Abhijeet Kamble

Our Web Sources:

We selected Walmart and Amazon's websites as our two web sources. Specifically, we have decided to extract product data for laptops from each website. We believe that these would make excellent sources because there is plenty of data to extract, each website provides similar data for products, and the pages for each laptop follow relatively the same structure. Each website had more that ten thousand results when searching for all laptops and therefore we knew we would be able to extract the required 3000 tuples for each table. Since the pages for each website follow a relatively similar structure, we knew we would be able to manually write an extractor for each that could store the available data into a csv file.

Both Walmart and Amazon list all of the laptops page by page with 40 laptops per page for Walmart and 24 laptops per page for Amazon. Then on the page for each individual laptop, both Walmart and Amazon have a table that contains the relevant information we are extracting. We knew that we would be able to write a wrapper to extract the information from these tables.

How we Extracted the Data:

We extracted the data with a relatively similar format for both Walmart and Amazon. We started with the page containing results for a search for laptops. Then, we extract the names and links to each laptop on the page (Walmart lists 40 laptops per page and Amazon list 24 per page). Using these links, we load the web page for each individual laptop. On this page, we first extract the price of the laptop from an html tag. For Amazon, we also needed to extract the laptop brand in a similar manner. Then, the rest of the data is located in a single table on the page. We find the table and extract all of the rows. For each row, the attribute name is listed in a <th> tag and the value is listed in a <td> tag. With all of this information, we then create a tuple and print it out to our csv file.

For the Walmart data, we had an extra complication because Walmart does not show user more than 1000 results for a single search even if more than 1000 results are found. To get around this, we split up the search by brand. We knew that no laptop could have more than one brand so we could be certain that this trick would not result in duplicates. After getting results for each brand, we combined all the results into a single table.

The Data:

Our data contains product information for laptops from Walmart and Amazon.

The Attributes:

1. Name (String): The name of the laptop listed on the website.
2. Price (Float): The price of the laptop in dollars.
3. Brand (String): The company who created the laptop.
4. Screen Size (String): The size of the screen. This is a string because it often contains units.

5. RAM (String): The amount of RAM included with the laptop
6. Hard Drive Capacity (String): The sise of the hard drive.
7. Processor Type (String): The name of the processor.
8. Processor Speed (String): The speed of the processor.
9. Operating System (String): The operating system included with the laptop.
10. Battery Life (String): The amount of time the battery generally will last.

The Walmart table contains 3038 tuples and the Amazon table contains 2102 tuples.

The Open-Source Tools Used:

We used three important open source tools to complete this assignment. The first of these is Selenium, which is a tool for loading web pages and retrieving the raw html. We initially tried to use some simpler tools, but we found that many other tools do not completely load the page. Since the specifications tables for each individual laptop are loaded by a script after the initial page is loaded, other tools would not give us the html for these tables. Selenium actually opens up a web browser to load the pages and therefore is is able to completely load a page including information from scripts and other sources. This allowed us to be able to access the specifications tables for each laptop.

The next open source package we used was Beautiful Soup. This package is used to parse the raw html. We used this package to search for tags within each html page. For example, in order to find all of the laptop links on a Walmart page, we used Beautiful Soup to find each <a> tag that contained the class ‘product-title-link’. We used this package to extract all of the information from the raw html we got with Selenium.

Finally the last package we used was Pandas. This package was used mostly to combine each of the individual brand csv files we created for the Walmart data. Since we could only search for up to 1000 laptops at a time, we searched by brand and created a csv file for each brand. Then, using pandas, we loaded each file into a dataframe, concatenated the dataframes, and saved the full table back to a csv file.