Advanced Python Project: Price Tracker and Comparison Model

Tinatin Nikvashvili tn709 · Paula Kiatkamolwong kk4158 · Khasi-Marc Jamieson kmj426 · Will Egan wve204 · Nathan Griffin nlg297

Anu-Ujin Gerelt-Od ago265

May 4, 2020

1 Introduction

It has become increasingly difficult to shop as a rational consumer and buy products from retailers offering the lowest prices. There are differences in prices in the market; retailers offer the same product for different prices and external factors create price fluctuations over time. The price information is abundant, and it is time consuming to sift through it in its entirety and manually compare prices from different retailers over time. Because no one has the time to find the lowest price, buyers end up buying products at a higher markup. In this project, we build a price tracker and comparison tool that will help buyers find better deals. We do this by enabling them to track prices of items over time, across retailers, and save money by buying items from the retailer with the cheapest price. Also, users could buy items when the price drops to a desired price they have in mind.

In our project we tracked prices across three major e-commerce companies: Amazon, Target and Walmart. We enable users to track baskets of items over time by providing specific URLs of items from Amazon. We call these buyers informed buyers. If buyers prefer to only check the price of items at a given time instead of keeping track over time and not provide specific URLs, they can do that by providing a general description of items, such as an 'air fryer' or 'keyboard'. We call these buyers windows shoppers. Once the informed buyer provides a basket of items to track, our model starts scraping and collecting information on prices of these items across the three retailers every 12 hours over time until the user stops wanting to collect information on a given basket. Users can explore price data and check how prices change over time or which retailer sells a given item for the cheapest price at a given time by exploring graph of the prices. We used different optimization techniques learned in class to make our scraper more efficient. To test our scraper, we scraped prices on 7 baskets of items (68 items total) for two weeks. The result and details of our approach and implementations are described below.

2 Approach

Our project is comprised of two types of price checks: those that only need to be checked on the spot and those that need to be followed over time. We call these functionalities the Window Shopper and Informed buyer, respectively. Both of these functionalities require a list of Amazon URLs as input. That list, which we refer to as a basket, can be generated in two ways: explicitly by the user or implicitly by a function, "Find Top Items", within our program.

2.1 Find Top Items

This function allows a user to proclaim a desire for a generic product within a price range and review level, and returns a list of items that satisfy these parameters. For instance, a user could ask for a listing of 10 keyboards in the price range \$60-\$100, with a minimum of 4 star average review rating, and at least 100 reviews. Our tool would return 10 Amazon product listings of keyboards that meet the criteria described above. The number of items returned, price range, review metrics and product string will all be variable inputs for the user. This is achieved through the scraping package for python called Selenium, which we designed to search through Amazon to find the appropriate products.

2.2 Window Shopper

The Window Shopper portion of the code takes in a basket from the user, either defined explicitly or returned from the Find Top Items function, and provides a snapshot of prices at that current moment. This is just for a user who is curious about prices across retailers at the current moment and may or may not buy depending on the output. To perform this cross-website search, the Amazon ID for each product (ASIN) is scraped and run through a website called synccentric that acquires the UPC (Universal Product Code) number for that product, which should be present on any authentic, online sale of that product. That code is used to search for the same product on Target and Walmart's websites.

2.3 Informed Buyer

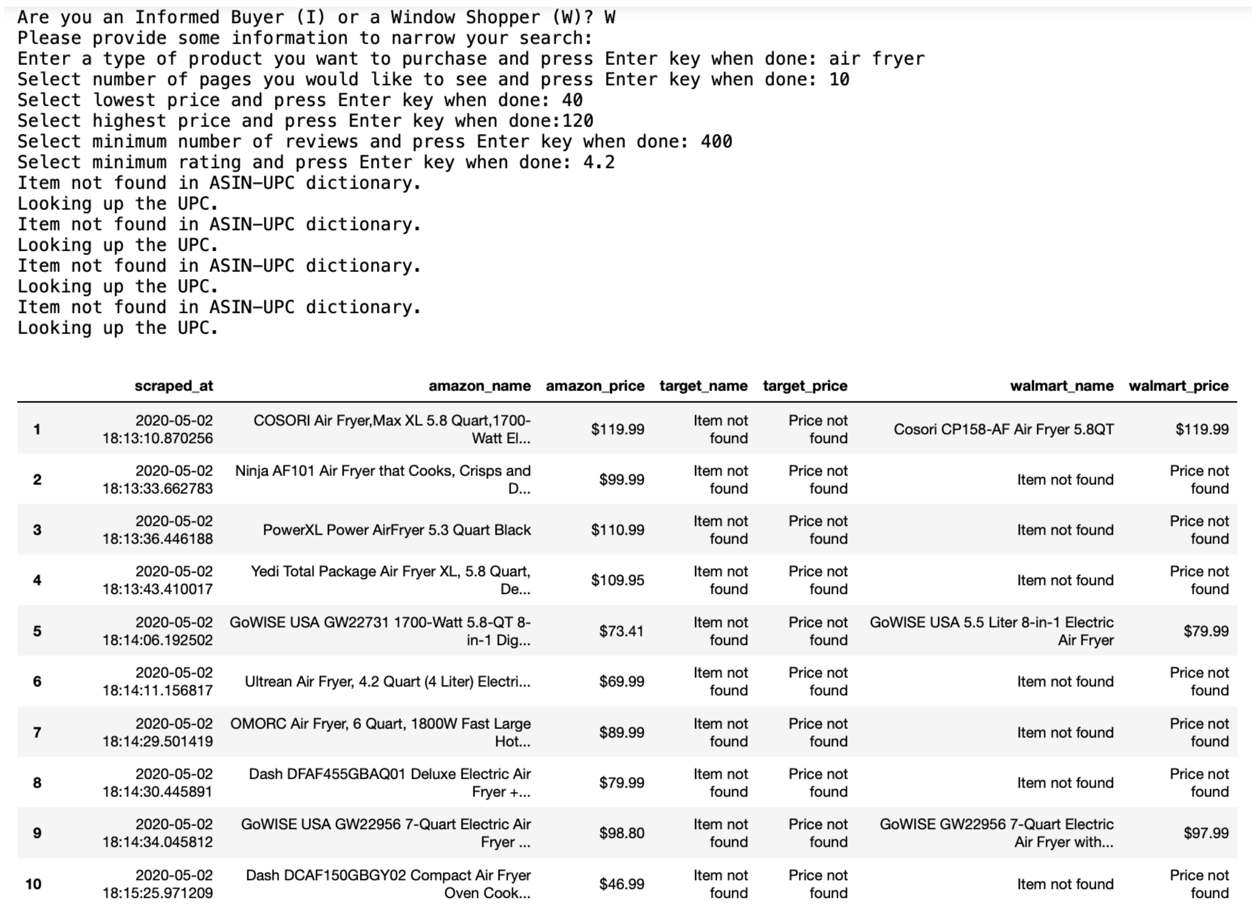
The Informed Buyer portion of the code also takes in a basket of items, and uses Selenium to scrape those products from Amazon, Target, and Walmart. However, in this case, the items are scraped and stored at a specified interval. The prices and time of the scrape are recorded and saved for further investigation and to inform the user of price drops.

There is a barrier to how many queries can be made to syncentric in an hour, so we made sure to store mappings between the ASIN and UPC numbers of products that we want to continually search on in a json file, which will save time on the daily scrape of prices. Additionally, the UPCs of items that the Window Shopper searches for are also saved in the json file, in case they would like to track it over time, thus becoming an Informed Buyer.

3 Experiments in Optimization

The Find Top Items function is used to provide an input for the informed buyer function when a user does not have the URLs of the specific items they would like to compare. The function takes a string input corresponding to the key word used to identify the item of interest and returns a list of URLs to be scraped from and compared. After initial analysis, we discovered a large variability in pricing and user reviews. To control the aforementioned variability, we introduced additional parameters to define the desired number of reviews, ratings and the price range. Figure 1 depicts a sample output of the main function, when it is run for a Window Shopper.

After successful execution of the function we ran the line profiler to get an analysis of general performance of the python code and expose potential bottle necks. We identified that the primary bottle necks came from iterating through web pages and calling the web driver object each time. Our three primary optimization approaches involved using iterators, applying Cython, and Threading. Lists representing page numbers to be iterated through were replaced with iter.counts() module, which netted us a speed up of about 4 seconds. Cython proved ineffective since Selenium objects were not able to be converted to Cython data types. We found no ways to effectively parallelize the function as the stopping condition required knowledge of how many URLs are recorded. Thus threading was not employed. We leave further optimization to the scraping function itself. Lastly we forced input and output types for a speed up of approximately 6 seconds.



```
Are you an Informed Buyer (I) or a Window Shopper (W)? W
Please provide some information to narrow your search:
Enter a type of product you want to purchase and press Enter key when done: air fryer
Select number of pages you would like to see and press Enter key when done: 10
Select lowest price and press Enter key when done: 40
Select highest price and press Enter key when done:120
Select minimum number of reviews and press Enter key when done: 400
Select minimum rating and press Enter key when done: 4.2
Item not found in ASIN-UPC dictionary.
Looking up the UPC.
Item not found in ASIN-UPC dictionary.
Looking up the UPC.
Item not found in ASIN-UPC dictionary.
Looking up the UPC.
Item not found in ASIN-UPC dictionary.
Looking up the UPC.
```

	scraped_at	amazon_name	amazon_price	target_name	target_price	walmart_name	walmart_price
1	2020-05-02 18:13:10.870256	COSORI Air Fryer,Max XL 5.8 Quart,1700-Watt El...	$119.99	Item not found	Price not found	Cosori CP158-AF Air Fryer 5.8QT	$119.99
2	2020-05-02 18:13:33.662783	Ninja AF101 Air Fryer that Cooks, Crisps and D...	$99.99	Item not found	Price not found	Item not found	Price not found
3	2020-05-02 18:13:36.446188	PowerXL Power AirFryer 5.3 Quart Black	$110.99	Item not found	Price not found	Item not found	Price not found
4	2020-05-02 18:13:43.410017	Yedi Total Package Air Fryer XL, 5.8 Quart, De...	$109.95	Item not found	Price not found	Item not found	Price not found
5	2020-05-02 18:14:06.192502	GoWISE USA GW22731 1700-Watt 5.8-QT 8-in-1 Dig...	$73.41	Item not found	Price not found	GoWISE USA 5.5 Liter 8-in-1 Electric Air Fryer	$79.99
6	2020-05-02 18:14:11.156817	Ultrean Air Fryer, 4.2 Quart (4 Liter) Electri...	$69.99	Item not found	Price not found	Item not found	Price not found
7	2020-05-02 18:14:29.501419	OMORC Air Fryer, 6 Quart, 1800W Fast Large Hot...	$89.99	Item not found	Price not found	Item not found	Price not found
8	2020-05-02 18:14:30.445891	Dash DFAF455GBAQ01 Deluxe Electric Air Fryer +...	$79.99	Item not found	Price not found	Item not found	Price not found
9	2020-05-02 18:14:34.045812	GoWISE USA GW22956 7-Quart Electric Air Fryer ...	$98.80	Item not found	Price not found	GoWISE GW22956 7-Quart Electric Air Fryer with...	$97.99
10	2020-05-02 18:15:25.971209	Dash DCAF150GBGY02 Compact Air Fryer Oven Cook...	$46.99	Item not found	Price not found	Item not found	Price not found

Figure 1: Example of an output for a Window Shopper

The Informed Buyer function takes a list, representing a basket of items, consisting of Amazon URLs that the user wants to track. The baseline of our function takes one item at a time from the basket, looks up its ASIN, converts the ASIN to UPC, searches the item on Target and Walmart, and records the prices from three different websites. This process repeats until every item in the basket is looked up and their prices are recorded. As an example, Figure 2 depicts the change in the price of ”Powerbeats Pro Wireless Earphones” over 12 days. What was most surprising was that, contrary to popular belief, Amazon does not always offer the lowest price and Walmart was prone to a lot of fluctuation. However, there is a dip in the price on May 1 at Walmart and Amazon, most likely related to Mother’s Day.

It was obvious to us that, with our basket consisting of nearly 70 items, running this function sequentially would take a very long time and that we could benefit from concurrency and parallelization. Because the process of converting ASIN to UPC and searching for each item’s prices are I/O bound operations and could benefit from multi-threading, we experimented with that, but ran into a problem using threads on Selenium’s webdriver objects. On the other hand, running the function using a pool of processors remarkably decreased the function’s run time from 22 minutes to 6 minutes. Because the function needs to be run every 12 hours in order to track price changes over time, it is vital that the function runs efficiently and for the shortest time possible. Therefore, we implemented the function and optimized it with multi-processing.

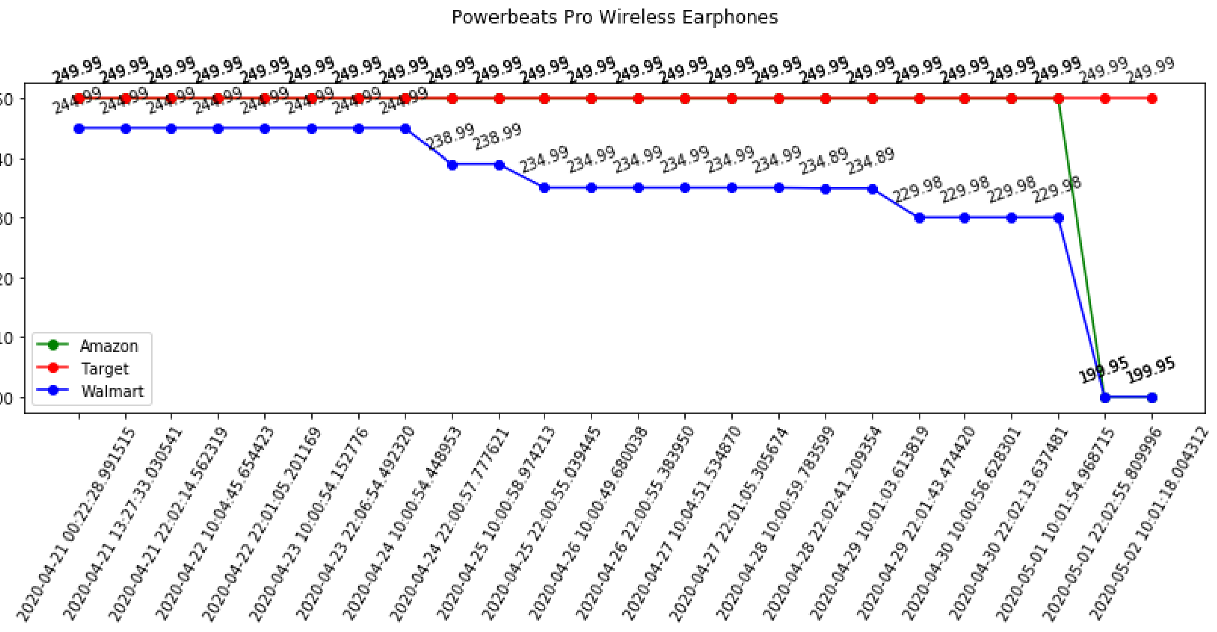


Figure 2: Example of a price tracker graph for an Informed Buyer

4 Discussions of Future Implementations

Our implementation of the price tracking tool described above is a useful ally to have when shopping online. However, there are some areas to extend the project that could provide a more useful front-to-back tool. The team discussed these possibilities but ultimately determined them to be beyond the scope of our current iteration.

First, the reliability of global identifiers from website to website is shaky. Some products have multiple versions of UPCs, therefore resulting in misidentification and empty entries in the dataset. More dependable information is available, but it must be paid for. Our only way of attempting to find global IDs was to have a program search for it on a free website, where we were also limited to a certain number of searches per hour. If this project were to be scaled at all, or if the availability of prices from all 3 websites were paramount to the use case, this would likely be a cost worth undertaking.

On to the point of scalability, we were only tracking prices for the 6 people in our group, but this type of program could be turned into a web page/software application to be employed by a broad range of users. If this is the end goal, we would have to expand our capacity to track users and likely do some work to optimize the data storage piece as the number of users expands. The Find Top Items function would also fit nicely into a UI pipeline since it allows people to track products they are interested in, by a criteria they find useful. The extension here would include some capacity for users to provide a target price or to alert users when a certain price condition is met. This could be a percentage drop, some relative ratio to a moving average, etc.

In conclusion, the tracker we built is a useful tool in a setting with a small number of users and is likely to actually save you some money on a product you do not need right away. As the number of users grows, our code would need to be optimized to handle that increase, and we would need to integrate some more user feedback channels. All in all, we experimented with and successfully implemented many optimization techniques to reduce the run-time of our baseline program by more than three quarters.