Goods Notes

Darek Johnson

Feb 24, 2017

2/24/2017

The goal of today was to think about and come up with a way to best store which items on Amazon and also how to get an initial database with the origins of a reasonable number of goods.

Each product on Amazon marketplace has a unique identifies called an ASIN. In our own database, we should use these as our own unique identifiers. If you consider any Amazon url it is of the following form

https://www.amazon.com/Russell-Athletic-T-Shirt-Heather-4X-Large/dp/B00719ZZWQ/...

Notice the .../dp/B00719ZZWQ/... the B00719ZZWQ is the ASIN for this product. It always comes after the /dp/ in the URL. This is at least true for the 30 random goods I checked today. In the past dp has been changed with other things like /product or /gp, but as of today it is /dp. Also because of the uniqueness of ASIN the following url is equivalent to the one above amazon.com/dp/B00719ZZWQ from a product standpoint (only minor differences).

In other news I started an Amazon Associates account that is free for 180 days, and this gives us access to Amazon Product API. This API is compatible with PHP, Java, ROR directly, but there are third party packages for Python. Java is probably best since third party usually just has worse documentation. If you want to see an example of what kind of information is provided from Amazon API, see test.xml and test2.xml in this folder. One important thing to notice is that the "origin" field that sometimes occurs in the product description area, is not returned in the Amazon API. This makes things a little more difficult, but things of note are the following

- Manufacturer
- Barcode (UPC, EAN)
- ASIN ID of a parent good (related good)
- Brand
- Model Number

To get our database started we can get a list of manufacturers we know are in the USA and use the API to find some of these and add the ASIN to the database. Next things to do are to start writing our own API in java to do these searches and set up our database manager in java. I'll ask Eli what database we want to store these things in.

2/28/2017

Since last we have recovered all the XMLs for Amazons subcategories. This is the hierarchy for Amazon products. This has given us a lot of the frequently searched keywords. What I have done so far is stored these XML files, then finding a node with no children, use the name of that node and append it with the root ancestor, for example Home & Goods Chairs, uses this as a keyword search Amazon, then from the 20 pages of results, HTML scrapes to get each ASINs from those pages. We store the ASINs in asin˙list.csv. Regardless of what we do with Amazon, getting a list of ASINs will be very important.

Things to do soon,

- Set up multiple rotating proxies to scrape all of these ASINs without getting caught
- Use ASIN to get the following information: price + company + country of origin + name of product + classification of product
- Amazon API only allows 2000 requests an hour, so how do we get around this? What should we use Amazon API for and how. If this is limiting, what can we scrape instead.
- Rewrite parse.py to run requests concurrently to optimize using grequests library
- Use try and catch statements around anything HTTP requests or things that could break. Log what's broken somewhere not in console

Overall structure, we should have the following python files

- amazon.py - Will do all the calls to the Amazon Associates API
- html˙scraper.py - Will do all the html scraping from Amazon's Website
- data˙manager.py - Will have the more complicated functionality for writing to and from CSVs and the database. Simple stuff can be within other classes
- main.py - This isn't made yet, but will eventually just import the other classes and use them. Eventually this should be run on our server every night or something to update our databse

The data we have to work with right now is a database of 20k ASIN numbers, and the entire Amazon Organizational Tree downloaded in the form of XML files. Next steps regarding ASIN is to get the 5 previously mentioned data points for each ASIN (maybe some more too)

3/2/2107

Made the mobile app to take a photo and save it with other information to the database. Created a new react-native project and bundled the APK.

3/3/2017

Goals

1. Make index.html so in plain text so that we can submit on the browser
2. Get the new git onto Heroku
3. Write the shell for the addon
4. Set up our proxies for scraping

3/4/2017

Goals

1. Structure the project as discussed with Kenny and Ben (ask for their feedback when done)
2. Go to Target and see what's easy to find, then check if you can use that information to find the product on Amazon
3. Write very basic versions of the AmazonScraper, AmazonProcessor, AmazonWriter
4. Write a description on how Amazon's HTML is structured with screenshots on which div id tags have what information.

Things that were easy to see from Target

- Bar Code
- Most products actually had the Made in blank
- Product Name
- At least one of company / brand / manufacturer

The big question is "how much can we" determine from a bar code. I think this will be the best way for us to uniquely identify products moving forward. We can get these for every good from Amazon API. With just the product name and company some products will be easy to find, but not all. One that I had a lot of trouble finding was happy birthday gift cards. They're very hard to distinguish easily. But products with a clear model number or name like appliances are pretty easy to find.

For reference, visiting 6300 connections on Amazon sites is about 800MB, which got us about 2600 scraped websites. Each GB of proxy data costs 50cents through proxy bananza.

3/5/2017

Goals

- Add barcode scanning to our application so that the UPC is provided on submission
- Add an admin tool page the website so that Eli and Darek can view submissions easily and then add the functionality to accept or reject.
- Debug the user submission as necessary
- Add rows in database for the use submissions for confirmed and for the UPC (barcode)
- Comment the code as Ben + Kenny advise
- Organize the code as they advise
- Try to determine cheaper proxy services
- Set up the python file to use Amazon's API to write information to our database. (Ask Kenny + Ben advice on how to structure this)
- Check the blanks in the Chairs data set. See if you can improve your scraping