Ian Schwartz

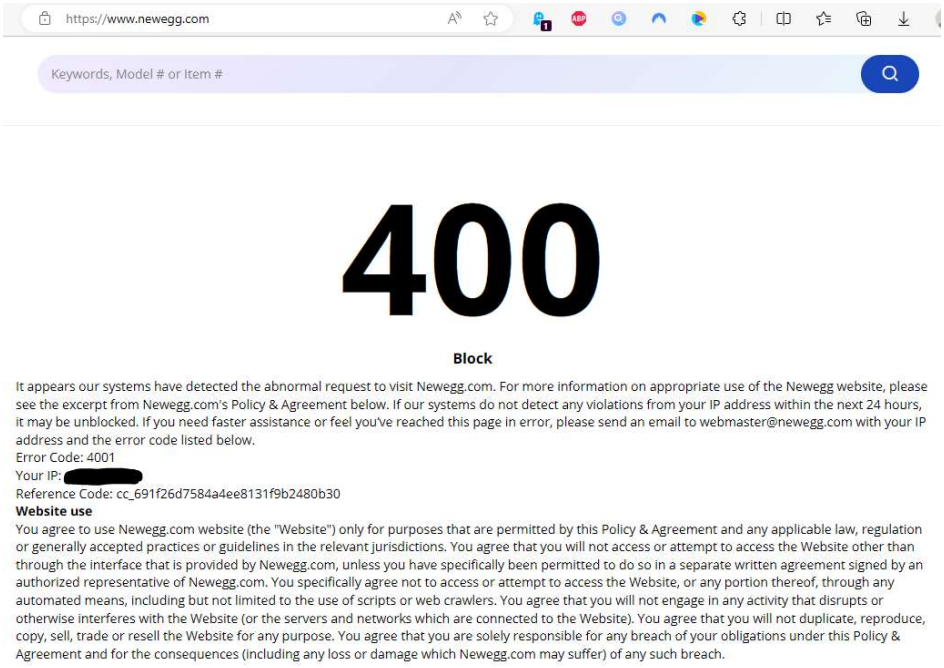
07/28/2023

COMPSCI 767 – Assignment 3 Discussion

Current Issues:

There are a couple issues I've run across while trying to complete this part of the project that have caused me to take much longer than I would have hoped:

- **Rewriting Code for Different Data:** First and foremost, the data I had before consisted of numeric data that isn't good for matching. This includes things like price, rating, # of ratings, etc. which vary from website to website for the same product. As such, I had to rewrite my code to actually access product page URLs which was a nightmare to get working.
- **IP Blocked by WAF:** Once I did get this working, I ran into the issue of getting my IP blocked. I thought it was happening before to some degree, but it was very clear this time because I went to access one of the websites and it gave me a message that I had been blocked (I'll include it below.)
    - As such, I was only able to look at ~8 search pages of products per day which doesn't provide me nearly as much data as I had before.
    - I attempted to figure out a proxy setup in order to circumvent this in the meantime, but could not get that to work. I honestly don't fully understand where I am going wrong with this, but I know that the free proxies I am attempting to use are like part of the issue.



In order to match records from two data sources, Amazon and Newegg, I employed both direct comparison and fuzzy matching techniques. The purpose was to identify products that are similar or identical across both platforms.

Algorithms Used for Matching:

Fuzzy Matching:

Recognizing that data might not always be represented identically across sources, we used the fuzzywuzzy library, which is based on the Levenshtein Distance (or edit distance) to measure similarities between sequences.

The main algorithmic function used was fuzz.ratio() from the fuzzywuzzy library. This function computes the similarity as a ratio between the two strings provided, returning a score between 0 (completely different) to 100 (completely identical).

For instance, if we have the names "Galaxy S21" in one table and "Galaxy S-21" in another, they might not match in a direct comparison, but fuzzy matching would recognize their similarity.

Direct Comparison:

After employing advanced algorithms, I used a simple direct comparison. If two records had the exact same value in a given field, they were considered a match on that field. I'm glorifying this a bit too much, as I literally just checked with my eyes whether or not the Fuzzy matching was relatively accurate by looking up the IDs in both tables and comparing them next to each other.

Problems Encountered:

Data Inconsistencies: Data wasn't always formatted consistently. For example, a product's name might include additional specifications on one platform but not on another, making direct comparisons unreliable. Encoding was an issue as well, I no longer have the exact example, but it was something similar to "Aez" or something that would prefix every single individual value when I ran one of my programs.

Performance Issues: Comparing every record in one table to every record in another was computationally expensive, especially with fuzzy matching. This method scales quadratically with the size of the tables, which can be problematic for large datasets.

Data Cleaning: Before even beginning the matching process, it was evident that data needed cleaning. Some records had non-numeric characters in numeric fields, some had leading or trailing spaces, and others used different units of measurement or had special characters. For weight, some would have 'lbs', some would have forgotten the 's' and just had 'lb', some had 'Pounds', some had 'pounds', etc. This made utilizing the SUBSTITUTE function in excel an absolute nightmare.

Attribute Variation: Attributes were named differently in the two sources. For example, "Display Size" in the Amazon table corresponded to "Screen Size" in the Newegg table. A mapping was necessary to bridge this gap, or physically alter the column titles which probably would have been a much more feasible option, since that doesn't involve manipulation of the data itself.

Threshold Setting: Deciding on a threshold for the fuzzy matching score was challenging. If set too high, I risk missing out on potential matches. If set too low, I risk having false positives. The threshold had to be set probably lower than most would probably consider accurate because of the previous problems I ran across.

Reported Numbers:

Amazon (table A) – 883 tuples/records

Newegg (table B) – *651 tuples/records

Matches (table C) – **23 tuples/records

*again, my IP keeps getting blocked so I'm having to piece this together by extracting around ~130 items every 24 hours every time they unban me, so there will be more for newegg (had over 2000 in my extraction before having to rewrite for more attributes), but amazon seems locked around where it is (tried 100, then 150 pages, no noticeable difference.)

**I can lower the threshold amount and get many more 'matches', but I was being picky and left it to almost exact matches among laptops, which was pretty hard for me to get working. I'll have to play around with the threshold and other attributes, but this should show that the program is working at the very least. The number should rise a bit when I get more tuples into my Newegg extraction.

OTHER:

One thing for me to consider is -- what is a match exactly when it comes to a 'laptop'?

- There are several variations of the same laptop, e.g. a 256GB storage or a 512GB storage, which are considered different by fuzzywuzzy. Even though they are the same brand, CPU, GPU, etc., implementing the storage size into the algorithm will cause the ratio of similarity between them to be less, and might not be picked up depending on what I set the threshold at.
- Small product-specific details might have slipped their way into a category for a few items only, for example:
    - What should be 'Windows10Home' might be:
    - Windows10Home64Bit–Multi-languagesupportsEnglish/Spanish/French
- ^this would throw off fuzzywuzzy I believe, and based on the threshold it might not be considered a match then with so few attributes being implemented.
- I can technically implement the name as the only attribute, in which case I might actually see more results considering it doesn't implement the mentioned numeric values, but then something like these might show up as a match (again, depending on threshold):
    - Dell Latitude 7410 14" Notebook, Intel Core i7-10610U 1.7GHz, 16GB RAM, 256GB SSD, Windows 10 Pro, Renewed
    - Dell Latitude 5414 Rugged Laptop with Intel Core i5-6300u CPU - 16GB RAM - 256GB SSD - Windows 10 Pro
    - ^ the above two are very similar if the threshold is set low, as it picks up the RAM, storage size, operating system, and brand naming (dell latitude), but the specific model is different.
    - This would lead me to use 'model' or 'model number' as an attribute, but values for that attribute are missing across a significant portion of the data I extracted, so it wouldn't help all that much.
- As mentioned before, some columns contain information that should be contained within a separate column (e.g. sometimes the 'Name' was actual 'Name + Model #' likely due to either an error with my script or a different format of information for that specific product. This gave me the idea to set everything up to look at the entire row of data, rather than each individual column at a time.
    - The advantage of concatenating the entire row and comparing it is that you're considering the collective information, and minor discrepancies in one column might be offset by exact matches in other columns.

However, the downside is that very important discrepancies can potentially be masked if other columns match very closely.
    - The other issue with doing this is I can't currently think of a way to account for missing values like I have with checking each column.

BIG TAKEAWAYS:

- Ideally, I want data that is formatted relatively similarly when looking at laptops. This includes data that don't have much room for variation, such as brand and series.
- Numeric values are hit or miss for matching laptops, too much variation between attributes across different websites/products like:
    - **Bad ones:** Storage size, weight, product dimensions, etc.
    - **Exceptions/good ones include:** screen size, resolution
-