Amazonics Report

Max Marcussen, Nicholas Nowicki, Liam Puknys

June 10, 2019

1 Dataset and Hypothesis

1.1 Dataset

Our dataset was a subset of 82 million unique Amazon product reviews provided by UCSD Professor Julian McAuley.* The compressed version of the entire dataset was 18.2 GB, which was approximately 72.3 GB after decompression. We decided to start by using category-specific subsets, namely video games (231,780 reviews), health products (346,335 reviews), electronics (1,689,188 reviews), home and kitchen (551,682 reviews), and sports (296,337 reviews).

1.2 Hypothesis

We wanted to see if we could discover relationships in the interest in different products, using number of reviews as a metric for interest. We were particularly interested in uncovering products who may drive each other's purchase or who complement each other, either immediately or after a period of time. For example, if a spike in interest in soft drinks show up years later as a spike in interest in isulin medication.

To test every combination of products is quite computationally intensive, as we need to check on the order of n^2 relationships, where n is the number of products, which is well in the millions, so at least a trillion such relationships.

1.3 Dataset as it Relates to Hypothesis

One of the difficulties we encountered with our data was that it is very sparse. Even limiting ourselves to products that had at least 5 reviews, the vast majority of products did not have enough data to draw any sort of statistically significant conclusion from and making it likely that we miss products that complement each other well. To address this, stored the number of reviews by month and year in a list and smoothed each review so that if a review was left in a given month, we put a 1 there, and then smoothed our data by adding a 0.5 to both sides and 0.2 to both spots two spaces away. This accounts for not only the

*http://jmcauley.ucsd.edu/data/amazon/links.html

somewhat arbitrary nature of splitting by month but also that what we really care about is interest in a product, not actually the number of reviews. If someone let a review in a given month, presumably interest in that product was high in the few months around it. These corrections are not perfect, at the end of the day nothing makes up for having sparse data, but they did help smooth our data to make finding true products of interest easier.

2 Algorithm

Due to the statistical irrelevance of this problem (we were unable to find much literature on it because finding spurious correlations is not really something of interest in statistical theory or applications) we needed to come up with our own algorithm for this.

We want to find things that show similar review patterns across time periods, that is, we want to find things that move and spike in similar ways, even if they spike in different time periods. A spike in 50 Shades of Grey books sold followed by a spike in diapers nine months later is of interest, and we want to detect that. To account for this, we loop forward and backwards through both lists of data, lining them up in every possible way (until one of the lists becomes all zeroes).

Once we line up the data, we look at the covariance between the lined up lists. This is extremely fast relative to other calculations and is among the simplest ways we could have measured correlation (correlation itself being derived from covariance), but the simplicity of the measurement gives it elegance, especially since covariance is an effective way to find how interests in two products vary together over time, especially given the sparseness of our data set.

The reason that we would normally favor correlation over covariance is that correlation normalizes the relationship of two vectors relative to their own internal variances.

$$corr(x, y) = \frac{cov(x, y)}{\sqrt{var(x) \cdot var(y)}}$$

We do not want to do that for two reasons. The first is that we want to give weight to products we have more data for, if we were to use correlation, our metric would look something like

$$corr(x, y) \cdot weight(N, M)$$

where N and M are the number of reviews for x and y respectively. While we do not have much prior knowledge of the distribution of reviews for a product, it is clear that the variance generally increases as the number of reviews does, since the variance of a list x is

$$\frac{1}{len(x)} \sum_{i=1}^{len(x)} (x_i - mean(x))^2$$

The more data we have for a product x, the larger x_i and $mean(x)$ will be, so a larger variance necessitates a larger number of reviews. Instead of multiplying by correlation a weight, we can just decline to divide correlation by variance, which will implicitly weight our analysis towards data we have many reviews for.

The other issue we want to address is seasonal variation. Interest in certain product, for example ski equipment and swim suits, spikes in certain seasons. These seasonal variations are easy to predict just from knowing what the products are and are not particularly interesting. The insight that interest in skis and swim suits seem to move together offset by six months is not particularly interesting. Using correlation places more weight on products whose demand is less predictable and spikes. This is precisely what we want, wince two products spiking in similar patterns is much more interesting than two products having similar but steady minorly fluctuating levels of interest.

The problem with using covariance as a measure is that if two products both have one huge spike in interest, since we compare them across all time periods, their spikes will eventually line up and return a massive covariance for some time period. The other downside is that if that line up occurs very early for one product and very late for the other, $len(x)$ will be lower, so the covariance between them will be higher. The trade off for this is that we do not give undue weight to older products relative to younger products for which we have similar amounts of data in the time periods they are both active, which is good, since we don't want to penalize young but popular products.

3 Big Data

The data was pretty big.

In the end, most of our analysis focused on our five category-specific subsets. Even if we just focused on one subset, our project would fall within the "big data" category. For each product, we tested the monthly review frequency against every other product. This meant that, if our product CSV had N lines, we would have to run N squared tests. Home and Kitchen, for example, had 28,237 tests.

To avoid duplicate test results (if product A is tested against every product and so is product B, both correlations A*B and B*A will be given), we implemented a function to determine whether a product ID was lexicographically greater than the product ID it was tested against. This removed all duplicate correlations, cutting run time by half.

3.1 Mapper

For our mapper, we needed to compare every product against every other product. We tried a few different approaches.

3.1.1 Approach 1: Two Mappers

Pros: ideally faster than any other approach. Cons: we were only ever able to get it to compare every line against itself. Even if this problem had been avoided, mappers only pass certain packets of lines to each core. Each line would not have been compared with each other line, but rather only with the packet of lines it belongs to. Run time on CSIL machines: $0.0007*N^2$.

3.1.2 Approach 2: Reading from CSV

Pros: allows us to test every line against every other line. Cons: prohibitively slow. Holding a CSV open with csvreader slowed run time by 16x. Run time on CSIL machines: $0.011*N^2$.

3.1.3 Approach 3: List-based approach

Pros: compares many lines against each other. Much faster than CSV approach, allowing for more correlations to be run. Cons: does not compare every line against every other line. Since each core was passed a certain group of lines, and we didn't pass the CSV around, we lost out on many correlations. As the mapper passes lines to more cores, the fraction of correlations run will shrink.

As an example, suppose our mapper splits our dataset into three chunks: a, b, and c. All lines run against all lines would give us the lines in a correlated with the lines in b, b correlated with c, and all nine possible combinations. But if each set of lines only gets to run against itself, we get a*a, b*b, and c*c, three possible combinations. This holds true with more mappers as well: we always get

$$\frac{1}{chunks}$$

combinations. So, with only one mapper, we get all correlations run, but with two, we get half of correlations run. Three mappers will have 1/3 of correlations run.

This limitation means that efficiency scales with accuracy. If we wanted all correlations, we would either need to use our CSV approach or just one mapper, both prohibitively slow. We can only gain increased speed by sacrificing some potentially interesting correlations. However, speed is so much greater under the list-based approach (30 times faster than the CSV based approach with small datasets) that, given a fixed amount of time, even if we sacrifice half of our correlations, we can run far more data through our algorithm. Of course, a great increase in speed would mean a massive sacrifice in amount of correlations - but for the level of this assignment, the fact that we can run so much more data means this implementation gives us far more correlations than we would otherwise get. There's likely a sweet spot of efficiency where we have enough splitting of the data set to accomplish the correlation task quickly, but not so much splitting where we lose a great deal of our correlations. Run time on CSIL machines was: $0.00037*N^2$. Run time on Google Cloud would of course

depend on number of workers used and number of cores the mapper split the lines between.

3.2 Number of correlations run

Assuming our mapper passes data to 16 cores, for each category, this is the number of correlations we get. This also assumes each mapper is passed an equal number of lines.

3.2.1 Health

18534 products. $18534^2/16 = 21{,}469{,}322$ comparisons.

3.2.2 Electronics

63001 products. $63001^2/16 = 248{,}070{,}372$ comparisons.

3.2.3 Home and Kitchen

23238 products. $23238^2/16 = 33{,}750{,}290$ comparisons.

3.2.4 Video games

10762 products. $10762^2/16 = 7{,}238{,}790$ comparisons.

3.2.5 Total

Total of 310,528,774 comparisons.

Each list we compared was length 240. Running sys.getsize() on a list of length 240 tells me a list of length 240 is 1984 bytes. So one comparison is of size $1984^2 = 3.936$ million bytes, or 3.936 megabytes. All 310 million comparisons would therefore give a size of 1.222 quadrillion bytes, or 1.222 petabytes. With over a petabyte's worth of comparisons, it's safe to say our project qualifies as a big data project.

3.3 Issues encountered

Beyond the mapper issues already discussed, we encountered a number of problems with deploying our algorithm to Google Cloud.

3.3.1 403 errors

Permission Denied errors. Came about for two reasons: we didn't properly set our permissions and we hadn't properly installed packages we felt we needed. Our first approach was to try to create a compfile, but we created an implementation of our correlation function to not use numpy.

3.3.2 503 errors

Caused by either a server-side error or loss of internet connectivity. Internet connectivity in Campus North was an issue. We had to forgo running a dataset because connection kept crashing.

3.3.3 CSV size

Some group members' VMs were bad at handling csv files of this size. For example, Max couldn't open the electronics product csv at all. This limited the speed at which we could run tests.

4 Results

4.1 Health and Personal Care

One defining characteristic of the health products group was that among the product pairs with the highest correlation scores, only one of the fifty pairs of products had a time shift of 4 or more months. Among the top fifty, the mean time shift was 1.02 months, with a variance of 0.69. This tendency of time shifts to be small means that, for any given product pair in health, the highest magnitude covariance was produced by applying only a small time shift when comparing the two frequency distributions. This pattern in the results could suggest that the highest scoring products in our results were complementary goods. If we use review interest as a proxy for demand, then we could say that since review interest for a given product pair lines up at similar points of the year, that demand for the first good within the pair might create demand for the second good, or vice versa.

Many of the product pairs were complementary in their functions. For example, one recurring pair was razor blades (from various different brands) and health supplements, specifically weight-loss and testosterone supplements (also from various brands). Another popular set of pairings was weight loss supplements with testosterone supplements, and pre-workout powder paired with these supplements. It seems plausible that men (and likely older men) who are on a fitness regimen would want to shop for these supplements in a bundle.

4.2 Electronics

The top results from the electronics group consisted of product pairs that may have some economic relationship, as well as some unexpected matchings. We found that streaming devices such as Google Chromecast, Roku, and Tivo dominated the results, and were frequently paired with each other, or accessory devices for smartphones. When streaming devices were paired with other streaming devices, we found that the time shifts was very small. At first, it might seem counterintuitive that competing streaming devices were "purchased

together," but one possible explanation for this phenomenon could be that electronics as a general category has a dramatic spike in interest during the months of November and December, when Black Friday and Cyber Monday sales occur on Amazon.

4.3 Home and Kitchen

Home and kitchen goods had the highest average time shift, 2.28 months, and the highest variance: 2.68 months. Among the products with time shifts of 3 months or more, we found one surprising pairing that showed up frequently: a neck pillow paired with many different kitchen items. There seems to be no intuitive economic relationship between these two items because their functions are completely different. However, the large time shift might make sense, as consumers tend to become interested in leisure items at a different time than when most shop for kitchen supplies.

Another result that is more easily explainable were the pairing of different competing kitchen items like knife sets, and vegetable slicers, with a small time shift. Although one consumer may not be buying both of these substitutable products for herself, similar to Black Friday and Cyber Monday for electronics, this may indicate that certain points of the year are popular to make kitchen purchases.