Isaac Gentz
Paul Barstad
Thomas Stevens
CS435 Big Data - TP2

Amazon Product Review Analysis

Ever been curious as to how Amazon sorts the reviews on products? Amazon gives you the option to view reviews based on those created most recently, or the “top reviews”. Customers can vote whether or not a review was helpful, the more helpful a review is it is regarded towards the top of all available reviews per product. We analyzed data to see what may cause a review to be deemed most helpful to browsing customers.

We are aiming to answer some specific questions such as:

- Which product category has the most helpful reviews?
- Does product rating correlate with review length?
- Does the length of the review correlate with the helpfulness of the review?

The goal of our project was to look for correlations among reviews, based on different aspects of each review. This would give Amazon and other online shopping sites some insight on how to possibly better display review information to their customers. By answering specific questions like the ones above, a company like Amazon could highlight specific reviews to give customers a better shopping experience. This could lead to higher quicker checkouts, an easier time finding useful information, and an overall more satisfying shopping experience leading to repeat customers in the future.

This analysis is incredibly important as so much online shopping is done due to the convenience. While shopping online, specifically at Amazon, you know what kind of product you want, but how do you decide on the brand or model? Typically by checking reviews on several similar products. If Amazon can show the best reviews to their customers, customer satisfaction will be maximized if they can easily narrow which product to purchase.

Our project is an interesting Big Data problem because there is an abundance of review data online, due to the massive online shopping market. We were able to download 16 gigabytes of reviews from 2016 Amazon sales. The results to our problem can be applied to Amazon, or any other site looking to maximize how efficient customers can be at choosing products to buy. If customers are pleased with the competency of information, or ease of finding it, online shoppers will return when looking to purchase something else.

Our software was written in MapReduce as it is what we have become most familiar with throughout this semester. Fortunately, our dataset is compiled of JSONs allowing easy parsing. Google has an API called GSON which allows you to take strings containing a JSON, and immediately assign the values you are interested in to another class, our Review class. The Review class only has three elements, an array for helpfulness score, a float for the overall review score,

and a string containing the review text. Once these elements were populated, we were able to begin choosing what to map dependent on what analysis was occuring in that job.

To compare the overall score (scaling from 1 to 5 stars) and how helpful a review was to other customers, we mapped the reviews by a key of overall score, with values of the ratio of helpfulness. In simpler terms, the JSON gave us the number of people who voted a review was helpful, and how many people had voted on that review in total. This became our first hiccup which we will discuss later. Our reducer was very similar to the Word Count assignment we created earlier this semester. Counting the total number of reviews and the sum of all the ratios inside the reducer is how we chose to analyze the two aspects of Amazon reviews. With the sum of ratios and total number of reviews we calculated the average ratio representing what percentage of online shoppers found reviews helpful for each of the five possible overall scores a product can be given.

With regards to the specific question of ‘which category had the most helpful reviews’, most of the work was done in the mapper, with a final sum and average being calculated in the reducer. This approach was taken because the question being asked is one of a numerical design pattern. This means that the data can be passed through once in the mapper to extract the useful bits of information. In this case, the useful information was the review helpfulness score, which came in as a JSON array of [x,y] (x out of y people found this helpful). We then simply wrote this out with a dummy key of the total number of reviews (this key could not be relied upon for the reducer’s calculation however, since there may be multiple mappers). The reducer then had a very easy task of counting the total number of inputs, summing each rating (as a percentage), and then dividing by the total number of inputs to get the average review score.

When looking to answer the question of does review length correlate to what the customers rate the product, the mapper is yet again where most of the work was done, with the reducer being again similar to that in our Word Count assignment. The mapping was done with a dummy key and the value being the score, which was pulled in as mentioned before using the very simple GSON method. The newest part of our mapper that this question introduced was the need for custom bins to separate the scores, unlike earlier where the scorers were the bins. We achieved a relatively uniform bin size through simple trial and error, first with only a small portion of the data, and then, again with all of it. It was very interesting to see that as the review length went up, we had to increase the character difference of the bins by more and more, with the last difference being a total of 350 characters. The reducer was simple, counting the amount of values for each key, summing the values, and then dividing by the count to get an average score for that grouping. An extra context write in our reducer was the method used for trial and error to find bin sizes we were happy with, by simply outputting the key and the total number of values associated with that key. This question started out as a slightly different one, being does review length correspond to how helpful the review is, but because of so much of the data being dumped due to the problem of no votes, we decided to switch it to its current iteration, which is able to show just as interesting of an answer.

We were then able to extend this functionality for each category by simply outputting a combination key of <product category, dummy key> from the mapper. Then in the reducer we could check the product category and have a hashmap of each average instead of a single average. Then

instead of the reducer writing the final output, a cleanup() function could go through the hash map and output the final average for each category.

Dataset

- http://jmcauley.ucsd.edu/data/amazon/
- This dataset was originally from Stanford's Network Analysis Project, we got the updated version at the link above.
- It is compiled of text files which include 82 million reviews on more than 9 million products from Amazon, separated by categories. Each text file is in JSON format, which will allow us easy traversing for the information we want to use for different analyses.
- The different attributes of each review we have access to include the following: reviewerID, asin, reviewerName, helpful, overall, unixReviewTime, reviewTime, summary, and reviewText.
  - We will focus on category, overall product rating, as well as helpfulness, and the length the of text of the review.
  - The helpful variable represents the ratio of customers who found the review helpful in making their decision to purchase a product. We are particularly interested in comparing this amongst Amazon's different categories.
  - Overall from each review is a rating of the product from the customer between 1.0 and 5.0.
  - For the initial analysis we plan to perform we are not concerned with the reviewerID, asin, reviewerName, unixReviewTime, or reviewTime.

Discussion and Analysis

The first analyzing we wanted to do was compare the review's overall score, with how helpful other customers thought the review was. The hypothesis we came up with was that the higher a review overall score is, the more helpful browsing customers would think a review actually is. The contrary we also hypothesized to be true; that the one star reviews would be among the most helpful reviews if a product is just awful. Here are the results from averaging the helpfulness score for each overall score.

Overall Review Rating	Average Percentage Helpfulness of Review
☆	49.0
☆☆	51.7
☆☆☆	58.3
☆☆☆☆	73.9
☆☆☆☆☆	78.2

We were partially correct in our hypotheses. Reviews with a five star overall rating were the most helpful according to other shoppers. One, two, or three star reviews are similar in their average helpfulness rating. However, there is a large jump up to four and five star reviews, which were the most helpful reviews overall.

A challenge we encountered while coding through this part of the project was that many reviews, have zero votes on if a review was helpful or not. If all reviews were mapped then the final average returned was NaN as the computer was trying to divide by zero when calculating the ratio. To remedy this, we did not include the reviews with zero votes concerning helpfulness as there was nothing to analyze. This made us curious as to how much of the reviews truly had a helpfulness score. In order to accomplish this we changed the reducer to write out the total number of reviews (once with all reviews, and once with only non-zero helpfulness reviews). There were 9.28 million total reviews, but only 4.64 million reviews that had a non-zero helpfulness vote. This means by almost exactly 50 percent of the reviews in 2016 had votes whether or not that particular review was helpful. After finding this out we wanted more insight on what overall score most reviews had, so we found the following with another MapReduce job.

Overall Review Rating	Total Number of Reviews	Percentage of Total Reviews
☆	502,530	5.4
☆☆	457,979	4.9
☆☆☆	929,194	10.0
☆☆☆☆	2,028,640	21.8
☆☆☆☆☆	5,370,349	57.9

A massive majority of the reviews are five stars, as may be expected. We were surprised to see that there are more one star reviews than two star reviews however. Having nearly 60% of the reviews be 5 stars may have something to say about what products Amazon allows to be sold on their site.

We made some interesting discoveries from the question of which product category had the most helpful reviews. Here are all of the categories, with their review helpfulness as a decimal (x100 for percent)

Category	Helpfulness
Kitchen	0.8204892
Kindle	0.80942357
pet_supplies	0.79767
Tools	0.7852273
Clothing	0.7811705
musical_instruments	0.78106093
Electronics	0.7573361
Patio	0.7550493
Automotive	0.7464452
Sports	0.74116653
office_products	0.7397732
toys_and_games	0.7332471
Baby	0.7114615
Vinyl	0.70895624
Health	0.7084101
Grocery	0.70623416
Beauty	0.70467144
cell_phones	0.7065306
digital_music	0.6930245
Android	0.6806856
movies_and_tv	0.62846977
video_games	0.62281996
instant_video	0.5745876

As you can see, the 3 categories with the most helpful reviews were *kitchen* at 82%, followed by *kindle* at 80%, and thirdly *pet supplies* at 79%. The three categories with the LEAST helpful reviews were *movies and tv* at 62.8%, *video games* at 62.2%, and lastly *instant video at 57%.* It is interesting to note that all three of the lowest categories seem to be related to technology and entertainment, whereas the three top categories seem more utilitarian. It would be interesting to gather further metrics such as the demographics of each type category for review to see if there is a correlation between reviewer ages and helpfulness score. Perhaps young people who are more interested in technology aren't as happy, thus the lower helpfulness scores? Extrapolating this a bit more, we can see that actually the bottom SIX categories are all technology related!

A third thing we wanted to analyze was through comparing the review lengths with the review's score. Our hypothesis for this was that the shortest reviews would have the highest scores, from customers being happy with the product and just putting something down, and that the longest reviews would have the lowest scores, from unhappy customers going on a bit of a rant.

Number of characters in review	Average Score
120 chars or less	4.3344502
150 chars or less	4.3576546
180 chars or less	4.3287826
240 chars or less	4.2918415
300 chars or less	4.252015
380 chars or less	4.216894
500 chars or less	4.177543
700 chars or less	4.137102
950 chars or less	4.110467
1300 chars or less	4.0955963
1300 chars or more	4.095014

The first thing to notice is that our hypothesis was correct, with short reviews having the highest scores and long reviews having the lowest. As touched on earlier, we think this is because happy customers are more likely to put a short review, maybe because they are writing it out of courtesy, while unhappy customers are more likely to write out long winded replies in an attempt to hurt the company by warding off as many potential customers as possible. Something else of note is that while the longest reviews have the lowest average score, their average is still above 4 stars,

which might seem contradictory to why we think they appear in this order, but remember, more than half of all scores are 5 stars which tends to skew the data towards that end of the spectrum.

Project contributions

Paul - Created Review class structure for comparing/analyzing reviews. Analyzed the overall score a review gave a product, and how helpful other shoppers found the review. Found totals for number of reviews, including a helpfulness score or not, which led to more analysis of one to five star reviews.

Isaac - Worked on review class structure so that parsing the JSON using GSON would go smoothly. Initially implemented parsing JSON with GSON. Worked on answering the question “Which product category has the most helpful reviews?”. Set up GitHub repo and Slack channel to facilitate team communication and collaboration.

Thomas - Analyzed the overall score a review gave a product versus the character length of that review. Created similarly sized bins for, with each bin being a reasonable distance from each other. Helped find totals for number of review, which I initially planned to use to split the bins.

Bibliography

J. McAuley and J. Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. RecSys, 2013.

This is the original data host. It has since been moved to the below sites

Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering

R. He, J. McAuley

WWW, 2016

Citation from the site where we downloaded the data set

Image-based recommendations on styles and substitutes

J. McAuley, C. Targett, J. Shi, A. van den Hengel

SIGIR, 2015

Citation from the site where we downloaded the data set

N, Oscar. “Algorithm for Dividing a Range into Ranges and Then Finding Which Range a Number Belongs To.” *Software Engineering Stack Exchange*, 24 Mar. 2013, softwareengineering.stackexchange.com/questions/187680/algorithm-for-dividing-a-range-into-ranges-and-then-finding-which-range-a-number.

Explaining how to use a bucket system with MapReduce