Amazon US customer review and product analysis

Li Yuan and Mattew Flaherty

DS5460-Big Data Scaling
Vanderbilt University

Table of Contents

Introduction

Amazon is the largest e-commerce company in the United States controlling 41% of retail e-commerce sales. 150.6 million people use Amazon mobile as of September 2019 with $125.6 billion in revenue. Amazon also sells more than 12 million products. With an abundance of users and products, Amazon must have a recommender system that keeps customers on the site or app and makes their time there efficient. Without this efficiency, the customers lose interest in using Amazon and use another source. Users also do not want to have to search through millions of products if they cannot think of the name of their product for which to search. The easiest way to shop would be to get an email or other form of alert for items in which you might be interested. One way to do this would be to create a recommendation system that can recommend a given number of items to customers in the database by previous purchases.

We have built a recommendation system with an ALS algorithm, performed sentiment analysis on customers' reviews with nlptown/bert-base-multilingual-uncased-sentiment, and used K-means to cluster Amazon product titles and descriptions into 12 product categories. This topic is interesting because a recommendation system can increase Amazon's revenue by introducing users to items that are similar to ones they have previously purchased which would increase their probability of buying the recommended item. On top of that, recommendation systems can improve customer satisfaction as they are recommended products and do not have to search for them in the Amazon app or website. It is also an interesting project because we are able to use pySpark's Alternating Least Squares (ALS) matrix factorization to make the recommendations. While Amazon already has a recommendation system, similar things have been completed by Netflix as well. At Amazon, Leino et al. (2007) looked at strategies that users use to select an item. They found that reviews and ratings are important to the selection process. Takács, Gábor, et al. (2007) explored the Netflix Prize dataset and compared their solution to the solution of the person who had received the lowest root mean squared error (RMSE), 0.8743, as of November 2007. Sentiment analysis on the reviews is another interesting topic because we can compare the customers' ratings to their sentiment scores. This will give us a better understanding of whether customers or choosing a number based on how they feel about the product or choosing a number to move on to the next step. The fine-tuned sentiment model can also be used to predict rating stars when there are only reviews without rating stars. Clustering is also interesting because this can aid recommendations as we could recommend items that fall into the same cluster that customers previously purchased items that belong to.

We will go over the problem, methods, results, and conclusion. For the problem, we will discuss the purpose of our project and how Amazon will benefit. The methods section will cover how we got our results. The results section will show what we discovered and we will provide suggestions and the next steps in the conclusion section.

Data Description

From 1995 to 2015, Amazon has collected all the customers' reviews, star ratings, and customers' purchase history through these two decades. This collection of datasets has 54 product category datasets, such as Apparel, Automotive, Baby, Watches, and so on. The total size is 54.41 GB. The below table is all the columns each dataset share:

customer_id	**review_id**	**product_id**	**product_parent**
product_title	**product_category**	**star_rating**	**helpful_votes**
total_votes	**vine**	**verified_purchase**	**review_headline**
review_body	**review_date**	**marketplace**	

We chose 12 product categories (12 dataset) of size 21.78 GB > 20 GB.
The below table is the 12 product categories we chose and the number of rows of each table.

Sports **(4,849,945 rows)**	**Baby** **(1,752,598 rows)**	**Apparel** **(5,902,724 rows)**	**Grocery** **(2,400,612 rows)**
Electronics **(3,093,705 rows)**	**Automotive** **(3,514,816 rows)**	**Books** **(3,102,417 rows)**	**Music** **(4,749,744 rows)**
Furniture **(792,035 rows)**	**Personal Care Appliances** **(85,976 rows)**	**Camera** **(1,801,821 rows)**	**Beauty** **(5,113,668 rows)**

Problem

We will be solving multiple problems. We would like to build a recommendation system, predict ratings from customers' reviews, and cluster items based on product titles and descriptions. The recommendation system will improve customer experience on Amazon's website as they will be able to shop more efficiently and could lead to more purchases. Predicting ratings using reviews will allow Amazon to understand if their customers are leaving accurate ratings that match their sentiment and help to fill empty star ratings given only customer reviews after fine-tuning the BERT model on this dataset we chose. The third problem to solve is clustering as this will allow us to potentially make recommendations for items that share the same category as well as see how similar products are to other products in the true category.

Methods

We used ALS, bert-base-multilingual-uncased-sentiment, k-means clustering, and Locality-Sensitive Hashing (LSH).

Recommender System

1. Describe the algorithm or approaches you designed

To create the recommendation system we used ALS. This system takes a large matrix and factors it into smaller representations of the original matrix. Inputs include user identification, product identification, and user ratings on products. The system outputs the predicted rating on a product by the customer and product category. Moving forward, we can then find K recommendations for each customer where K is the number of recommendations for each customer.

2. The tools and software used to implement the algorithm

The platform we used is Google Cloud and the Hadoop file system on Google Cloud. The tools and software we used to implement this algorithm include PySpark and MLlib utilizing the ALS function within the package. With MLlib, we can choose the number of blocks used to parallelize computation, the number of latent factors, iterations, cold start strategy which controls the behavior when the ALS encounters new items or users in the test set that don't present in the training set, and the regularization parameter. While PySpark does a good job with large datasets, our 20GB+ dataset still took a long time to run.

3. What problems did you face? How did you solve them?

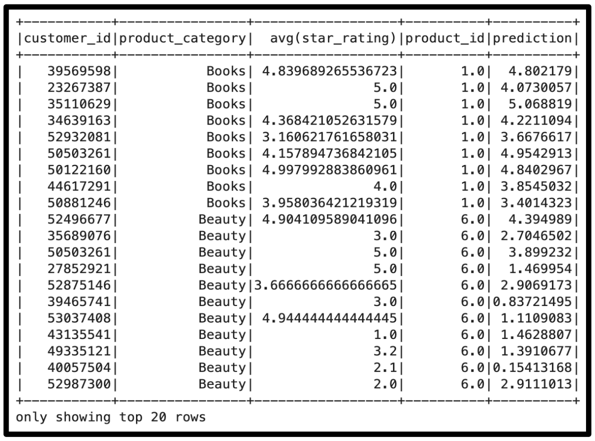
The problem we ran into was that firstly we only selected the top 20 customers who posted the most reviews on the Amazon website and ran the ALS algorithm to fit it. However, after our presentation, the instructor informed us of the potential bias created by this selection because the top 20 of the most active customers are more positive to the Amazon products and their service as s result of overwhelming positive (5-star ratings) reviews. Then, we selected the top 20 customers who posted the most reviews within each star rating interval. We calculated the average star ratings by each product category of each customer. We treated this average rating as item scores rather than different granular products within each product category as item scores due to millions of products. After this calculation, we selected top 20 customers whose average ratings are within [0.8, 1.2] for star one; top 20 customers [1.8, 2.2] for star two; top 20 customer [2.8, 3.2] for three stars; top 20 customers [3.8, 4.2] for hour stars, top 20 customers [4.8, +infinity] for 5 stars. In total, we scaled up our only 20 customers to 20*12=240 customers.

4. What were your raw results? How do you evaluate your results?

Then, we split our data into 80% training and 20% test for the evaluation of our model. Finally, we trained our training data and evaluated our ALS on the test set to have a 3.56 root mean square error which is not bad. From the below table, we can observe that our recommender system can predict well from 1 star to 5 stars because of our well-defined balanced sample selection to make sure the training dataset has the balance rating stars from 1 to 5.

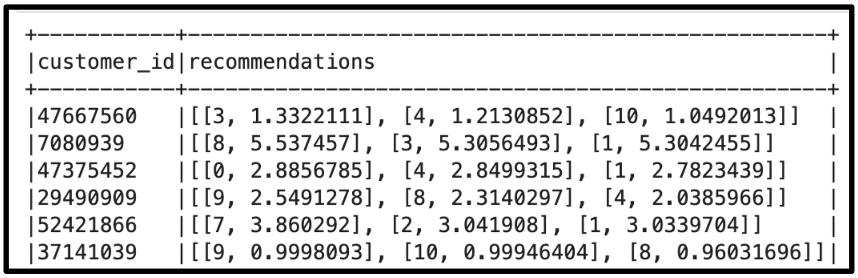
5. Chart/graph your results

The below first chart is the raw prediction rating stars on the test set.



```
+-----------+----------------+------------------+----------+----------+
|customer_id|product_category|  avg(star_rating)|product_id|prediction|
+-----------+----------------+------------------+----------+----------+
|   39569598|           Books| 4.839689265536723|       1.0|  4.802179|
|   23267387|           Books|               5.0|       1.0| 4.0730057|
|   35110629|           Books|               5.0|       1.0|  5.068819|
|   34639163|           Books| 4.368421052631579|       1.0| 4.2211094|
|   52932081|           Books| 3.160621761658031|       1.0| 3.6676617|
|   50503261|           Books| 4.157894736842105|       1.0| 4.9542913|
|   50122160|           Books| 4.997992883860961|       1.0| 4.8402967|
|   44617291|           Books|               4.0|       1.0| 3.8545032|
|   50881246|           Books| 3.958036421219319|       1.0| 3.4014323|
|   52496677|          Beauty| 4.904109589041096|       6.0|  4.394989|
|   35689076|          Beauty|               3.0|       6.0| 2.7046502|
|   50503261|          Beauty|               5.0|       6.0|  3.899232|
|   27852921|          Beauty|               5.0|       6.0|  1.469954|
|   52875146|          Beauty|3.6666666666666665|       6.0| 2.9069173|
|   39465741|          Beauty|               3.0|       6.0|0.83721495|
|   53037408|          Beauty| 4.944444444444445|       6.0| 1.1109083|
|   43135541|          Beauty|               1.0|       6.0| 1.4628807|
|   49335121|          Beauty|               3.2|       6.0| 1.3910677|
|   40057504|          Beauty|               2.1|       6.0|0.15413168|
|   52987300|          Beauty|               2.0|       6.0| 2.9111013|
+-----------+----------------+------------------+----------+----------+
only showing top 20 rows
```

The below table is once we fitted and trained the ALS, we made the top three recommendation movies for each customer sorted by prediction ratings.



```
+-----------+---------------------------------------------------+
|customer_id|recommendations                                    |
+-----------+---------------------------------------------------+
|47667560   |[[3, 1.3322111], [4, 1.2130852], [10, 1.0492013]]  |
|7080939    |[[8, 5.537457], [3, 5.3056493], [1, 5.3042455]]    |
|47375452   |[[0, 2.8856785], [4, 2.8499315], [1, 2.7823439]]   |
|29490909   |[[9, 2.5491278], [8, 2.3140297], [4, 2.0385966]]   |
|52421866   |[[7, 3.860292], [2, 3.041908], [1, 3.0339704]]     |
|37141039   |[[9, 0.9998093], [10, 0.99946404], [8, 0.96031696]]|
```

6. How do your results solve the initial problem? What other conclusions can you draw?

As we included and scaled up the number of customers in our recommender system, the RMSE didn't increase a lot and it only increased about 1. This showed our new balance method is capable of predicting the rating stars for different levels across 12 product categories. We concluded that our ALS recommender system didn't perform all the time. There are a few cases where the true rating is 4 or 5 but the model predicts 1 or 2, this gave some big gaps between them, however, the number of odd cases is less than before.

Sentiment Analysis (NLP-fine tune)

1. Describe the algorithm or approaches you designed

The bert-base-multilingual-uncased model can take sentences as inputs and produce probabilities of rating scores where the highest probability score can be used as the predicted rating. The bert-base-multilingual-uncased model was pre-trained for sentiment analysis on product reviews in six languages: English, Dutch, German, French, Spanish, and Italian. We fine-tuned this pre-trained model on our selected subset dataset

2. The tools and software used to implement the algorithm

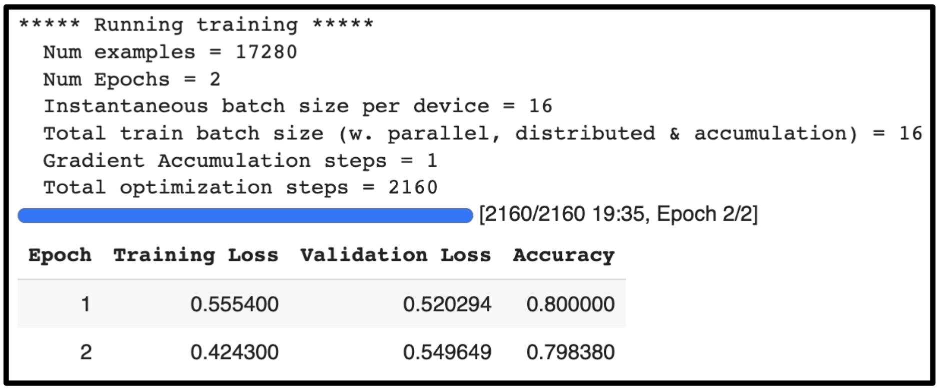
We used transformer, datasets packages provided by Hugging Face, and used Cloud Computing Platform, Google Colab, given the GPU hardware Tesla V100-SXM2-16GB and 54.8 GB virtual memory. Numpy, Pandas, Sklearn, and other common data science packages were used.

3. What problems did you face? How did you solve them?

Before the presentation, we only applied this pre-trained model to predict the rating stars given the subset sample amazon dataset. However, after the presentation, we want to improve the prediction accuracy and F1 scores. We fine-tuned the bert-base-multilingual-uncased model on our selected random subset. In order to include all 12 product category data, we randomly sampled 2,000 rows of each category. Then we split the into training, validation, and test set. We replaced its head classification with our customer review title concatenated with the review body to fine-tune it on 17,280 rows of training set while validating it on 4,320 rows of dev set. Finally, we evaluated our model performance on a held-out test set: 2,400 rows. During the fine-tuning process, we encounter the error that the star 1 from 5 couldn't be trained because the model only recognizes the label from 0, so we deducted 1 from all rating stars and converted them into integer types. Then we tokenized all reviews and defined metrics used by the model to evaluate the training and validation performance.

4. What were your raw results?

The below screenshot is what I got from the fine-tuning of the pre-trained transformers model on our selected random subset dataset. We chose two epochs because we found that the third and more epochs didn't improve the validation accuracy significantly and more epochs took more time and resources to train. Only after the first epoch, we found that validation accuracy reached 80% accuracy and in the second epoch the validation accuracy dropped a little, then we stopped training.



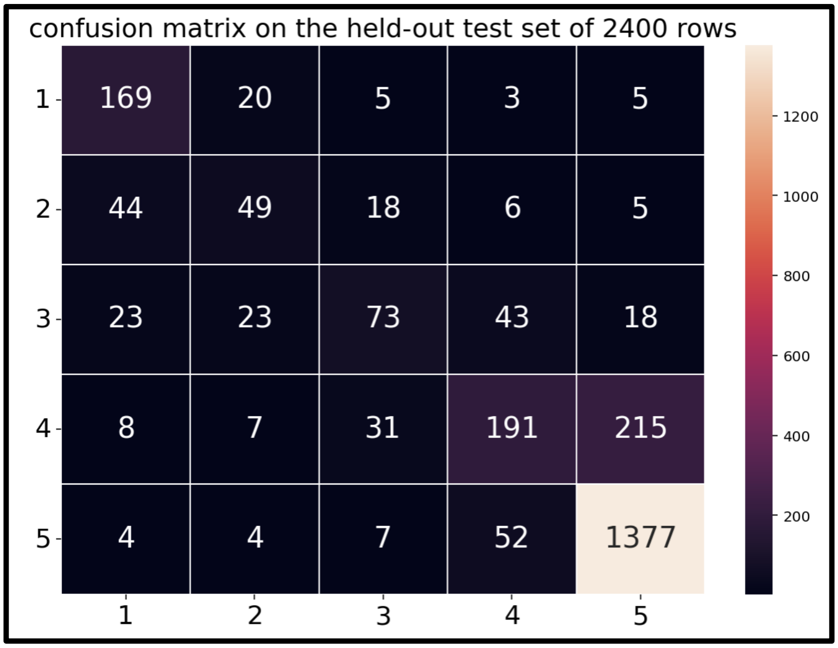
```
***** Running training *****
  Num examples = 17280
  Num Epochs = 2
  Instantaneous batch size per device = 16
  Total train batch size (w. parallel, distributed & accumulation) = 16
  Gradient Accumulation steps = 1
  Total optimization steps = 2160
```

[2160/2160 19:35, Epoch 2/2]

Epoch	Training Loss	Validation Loss	Accuracy
1	0.555400	0.520294	0.800000
2	0.424300	0.549649	0.798380

5. How do you evaluate your results? Chart/graph your results

We evaluated the model performance on the test held-out set of 2,400 samples. We got 77.8% accuracy and a 75.7% F1 score on it. We visualized the test performance by the below confusion matrix.



confusion matrix on the held-out test set of 2400 rows

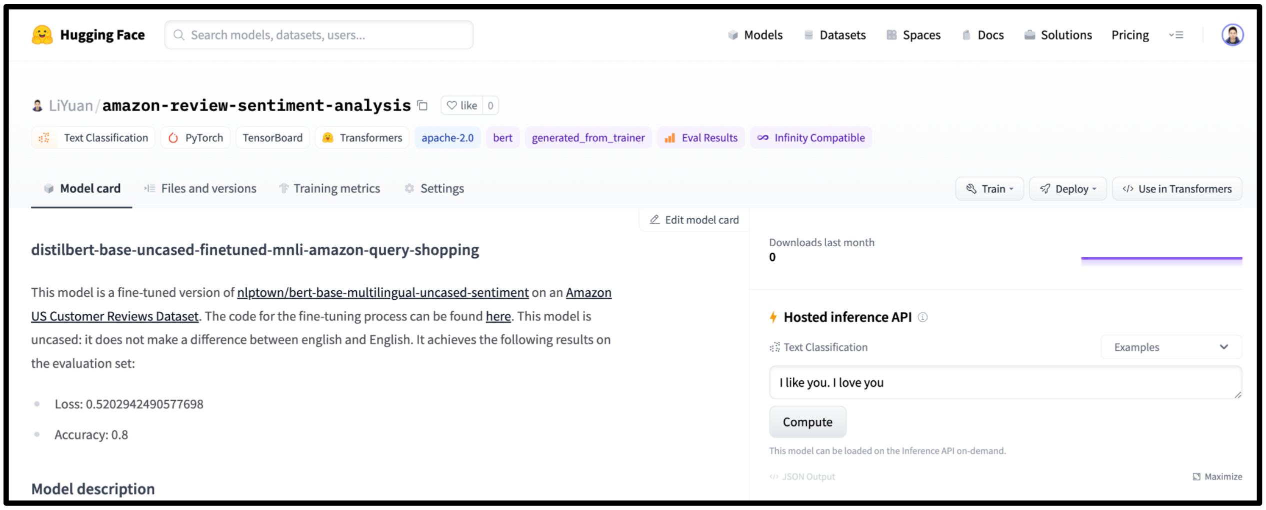
	1	2	3	4	5
1	169	20	5	3	5
2	44	49	18	6	5
3	23	23	73	43	18
4	8	7	31	191	215
5	4	4	7	52	1377

6. How do your results solve the initial problem? What other conclusions can you draw?

Firstly, finding the proper pre-trained model is important otherwise it costs too much training. Fine-tuning the review pre-trained models for multilingual on our balanced dataset improved the accuracy and F1 score a lot because we made the domain adaptation on amazon products and their reviews. We found that this dataset is 5 rating stars overwhelmingly, so, in our confusion matrix, we found that our fine-tuned model can predict most correctly on 5 rating stars.

7. **Publish our model on the Hugging Face community**

Lastly, we published our fine-tuned sequence classification model to the Hugging Face for sharing and letting other people download, test and experiment. Please check out this website to play around it and see more details on the hyperparameter we used to train and the framework version: https://huggingface.co/LiYuan/amazon-review-sentiment-analysis



distilbert-base-uncased-finetuned-mnli-amazon-query-shopping

This model is a fine-tuned version of nlptown/bert-base-multilingual-uncased-sentiment on an Amazon US Customer Reviews Dataset. The code for the fine-tuning process can be found here. This model is uncased: it does not make a difference between english and English. It achieves the following results on the evaluation set:

- Loss: 0.5202942490577698
- Accuracy: 0.8

Model description

I like you. I love you

This model can be loaded on the Inference API on-demand.

Conclusion

Having found recommendations and predicted ratings, we then wanted to cluster the products into categories. Using k-means clustering, we could take the title, map the words to vectors, and get the clustering predictions. We also used LSH to cluster the products because LSH is useful on large datasets. To complete this, we tokenized the columns, removed stop words, took ngrams, used hashing to create term frequency, used minhasing to generate LSH, and calculated Jaccard similarity. However, we found that word2vec model and K-means didn't perform well and were not acceptable on our large dataset because we got 4398708.45 for the within the set sum of squared errors. This result was too large to accept. Our clustering method was not successful because of word2vector, so we advise the use of LSH as this is more suitable for larger datasets. We were able to learn more about Amazon's customers and how they rate items. We also successfully recommended three items per customer. We advise the use of this model as we were able to use customer ratings to recommend items for them. The fine-tuned transformer model was also useful in predicting ratings.

The next steps include obtaining data on Amazon search history to build a more accurate recommendation system. With search history, we can have previous customer purchases as well as items that customers are viewing but have not yet purchased. If we can recommend items that the customers have not yet purchased, then this may intrigue the customers and increase their desire to have the item for which they searched.

References

Leino, Juha, and Kari-Jouko Räihä. "Case amazon: ratings and reviews as part of recommendations." Proceedings of the 2007 ACM conference on Recommender systems. 2007.

Takács, Gábor, et al. "Major components of the gravity recommendation system." Acm Sigkdd Explorations Newsletter 9.2 (2007): 80-83.