This is the project report submitted for Term Project CIS 731- 2021 FALL

Developing a Collaborative Recommendation System for Amazon Kindle Store Data

Krish Weragalaarachchi
Kansas State University

Abstract:

These days, whether we are regular customers or new customers we get recommendations of products to buy on Amazon as well as any other online market. This is all thanks to machine learning algorithms and recommender systems. Indeed, high-quality recommender systems are the main drivers for success in the online marketplace these days. This paper is mainly focusing on developing a collaborative recommender system for Amazon product data.

Introduction:

The goal of a recommendation system is to analyze how users interact with an existing system and make recommendations of what they can buy, listen or watch in the future. Recommender systems are broadly classified into two types based on the data being used to make inferences. They are content-based filtering which uses attributes of users and items and collaborative filtering which uses user interactions. This project specifically focuses on Amazon Kindle store products data and, the data is used to develop a collaborative recommendation system that recommends a list of products to a given customer based on the ratings of other similar customers on the website. These recommendations are further ordered based on the product's average sentiment score calculated using customer reviews and the top 5 were selected as the best recommendations.

Related Work:

There are so many examples and articles online on building recommendation systems for movies, books as well as amazon products. However, I have not seen any project specifically using the same dataset that I selected from Julian McAuley Lab at the University of California San Diego. Zuzanna Klyszejko has a blog post featuring “Recommendation systems with Apache Spark” in which she uses amazon Patio, Lawn, and Garden data and her work is the inspiration to my project.

Data Set:

I first tried using the 2018 Amazon Kindle store data set which is publicly available online from Julian McAuley lab at the University of California, San Diego. It is 4GB in size and too big to run in Google Colaboratory. Therefore, I switched to the 2014 Amazon Kindle store data set which has the same format but is smaller in size. That can be downloaded from Julian McAuley lab at the University of California, San Diego.They have data for different Amazon product categories and I used the 2014 Amazon Kindle store 5-core data set. These data have been reduced to extract the 5-core, such that each of the remaining users and items has 5 reviews each. The data set is 827MB in size and has 982,619 entries. It has information on product id, customer name, customer id, review text, overall rating, review summary, and review time.

Methodology:

The selected data is checked for null, missing, or nan values. To get a less sparse customer-product matrix, I got a dense subset from the dataset so that the remaining customers have at least 30 ratings. It could have been better if I could get a subset such that the remaining customers and products have 30 ratings. I used two window functions to do that but, it was unsuccessful as fulfilling one constraint failed the other. I was not able to find a way to fulfill both constraints at the same time. However, the sparseness of the product-user matrix could be reduced from 99.98% to 99.67% by doing that. To put into context, the filtered dataset contains 196190 ratings, 12291 products, and 4867 customers. Table 01 shows the number of counts for each rating type in the selected data set.

Rating	Count
1	2631
2	5591
3	20477
4	57308
5	110183

Table 01: Number of counts of each rating type in the selected subset of Amazon Kindle Store data set

As can be seen in figure 01 the distribution of ratings is highly skewed to the right where the maximum rating of 5 stars. This is very common in any rating system as customers tend to rate things that they buy, 4 or 5 stars unless they are highly disappointed and unhappy with the product. As can be seen from figure 02, the number of customers who gave an average of 5 stars is also higher. Therefore, the median value of a rating system is usually higher. In this dataset median rating is 5.0 and the mean rating is 4.36.

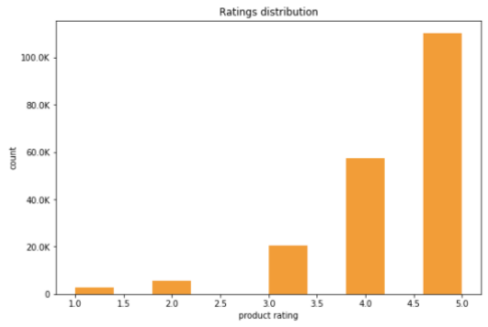


Figure 01: A histogram showing the distribution of ratings in the Amazon Kindle Store data set

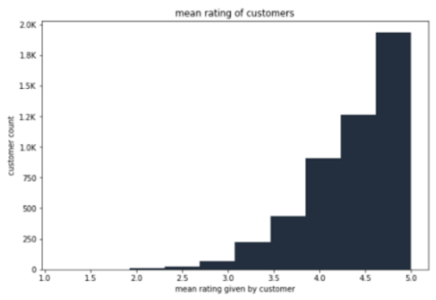


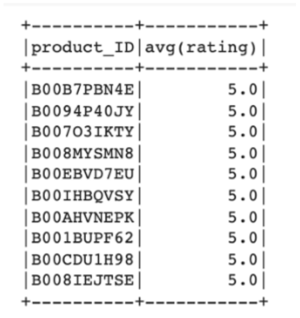
Figure 02: A histogram showing the distribution of the average rating of customers.

Then, I implemented a few recommendation systems to find the best model for this data. They are as follows.

- Basic recommendations based on median and mean
- Popularity based recommendations
- Collaborative recommendation based on
  - Alternative Least Square algorithm
  - K- Nearest Neighbor Basic
  - K- Nearest Neighbor With Means
  - Singular Value Decomposition

Basic recommendations, popularity-based recommendations, and collaborative recommendations based on the Alternative Least Square algorithm (ALS) were implemented using PySpark.

Usually, the most basic recommendation would be the median or mean. The popularity-based recommendation system recommends products to customers based on the overall popularity of the products in the market regardless of the customer preferences. This type of recommendation is important when recommending products to new customers as the system does not need any historical data of the customer for its recommendations. On the other hand, they are not so effective as they lack personalization. The dataset was split into 80-20 train-test split and the average rating for each product was calculated. The testing set was used for model evaluation. Figure 03 shows the best 10 recommendations based on the popularity-based recommendation system.



```
+----------+-----------+
|product_ID|avg(rating)|
+----------+-----------+
|B00B7PBN4E|        5.0|
|B0094P40JY|        5.0|
|B007O3IKTY|        5.0|
|B008MYSMN8|        5.0|
|B00EBVD7EU|        5.0|
|B00IHBQVSY|        5.0|
|B00AHVNEPK|        5.0|
|B001BUPF62|        5.0|
|B00CDU1H98|        5.0|
|B008IEJTSE|        5.0|
+----------+-----------+
```

Figure 03: 10 recommendations to any customer based on popularity-based recommendations.

8 models were tested with ALS algorithm using ParamGridBuilder(); a builder for a param grid used in grid search-based model selection in PySpark and assessed the effectiveness of the model with 5 fold cross-validation. The parameters I chose are 5 and 10 for the rank of the factorization (rank), 5 and 10 for the maximum number of iterations(maxIter) and, 0.01 and 0.1 for the regularization parameter (regParam).

K-Nearest Neighbor Basic (KNNBasic), K-Nearest Neighbor with user mean (KNNWithMeans) and Singular Value Decomposition (SVD) algorithms were implemented using python surprise library. The parameters chosen are depicted in table 02 below. These models were also assessed for effectiveness with 5-fold cross-validation.

Algorithm	Parameters tested
K-Nearest Neighbor (KNN-With Means)	name : msd,cosine,pearson_baseline min_support : 3, 4, 5 user_based : True k : 5 ,10 Verbose : True ,Fasle
K-Nearest Neighbor (KNN-Basics)	name : msd,cosine,pearson_baseline min_support : 3, 4, 5 user_based : True k : 5 ,10 Verbose : True ,Fasle

Singular Value Decomposition (SVD)	n_epochs= 20, 25
	lr_all=0.007, 0.009
	0.01 reg_all= 0.4 , 0.6

Table 02 : selected parameters for KNN and SVD

In addition to the ratings given by customers, the dataset contains reviews given by them for each purchase as well. Customer reviews include what customers truly feel and they can be more powerful than ratings at times. Next, the review data were preprocessed by removing non - ASCII characters, fixing abbreviations, removing hyperlinks, removing mentions, removing numerals, and removing non-alphanumeric characters and words less than 1 in length. Then the compound scores for each review were calculated using VADER sentiment analyzer. The extensive cleaning, stemming, lemmatization, or stop word removal of text data is not necessary when using the VADER sentiment analyzer as it can handle raw data very well. The reviews were categorized as positive (sentiment score >0.1), negative (sentiment score <-0.1), and neutral (-0.1< sentiment score <0.1) based on sentiment score, and the sentiment distribution is depicted in figure 04. Then, the average sentiment score of reviews for each product is calculated. This average sentiment scores of reviews are used to filter out final recommendations to the customer.

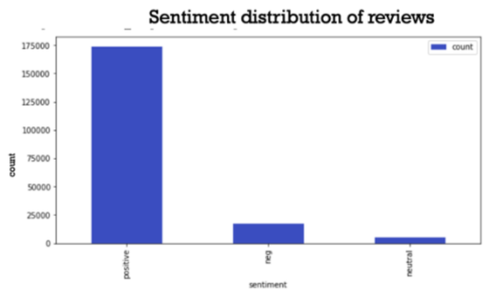


Figure 04: Sentiment distribution of customer reviews.

Evaluation:

All the models I implemented are evaluated based on the most popular metric for evaluating recommender systems. The root mean squared error (RMSE) is a prediction accuracy metric that tries to answer the question of how close the predictions are to actual ratings. It uses squared deviations and so larger errors tend to get amplified. I evaluated all the models based on RMSE.

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(p_i - a_i)^2}{n}} \qquad \begin{array}{l} p_i = \text{predicted rating} \\ a_i = \text{ actual rating} \end{array}$$

RMSE values for each model I tested are depicted in table 03. I selected the baseline as the most basic recommendation. The RMSE of the median-based model is 1.087 which is higher compared to the RMSE of the mean-based model (RMSE = 1.087). Hence, the mean-based model with RMSE of 0.878 was selected as the baseline model.

Model	**RMSE**
median based	1.087
mean based	0.878
Popularity based	0.852
Alternative Least Square	0.787
KNN (Basic){'name': 'pearson_baseline', 'min_support': 5, 'user_based': True, 'k': 5, 'verbose': True}	0.902
KNN(With Mean) {'name': 'msd', 'min_support': 3, 'user_based': True, 'k': 5, 'verbose': True}	0.76
SVD {'n_epochs': 25, 'lr_all': 0.01, 'reg_all': 0.4}	0.713

Table 03: Root mean squared errors of implemented models

Results:

Among the models that I tested, the Single Value Decomposition algorithm with n_epochs= 25, lr_all=0.01, reg_all= 0.4 has the lowest RMSE (0.713) compared to our baseline (0.878) and, we can safely select this as our collaborative recommender system. As you can see from figure 05, SVD and KNN (With Mean) are the best ML algorithms to recommend products for users from this data set.

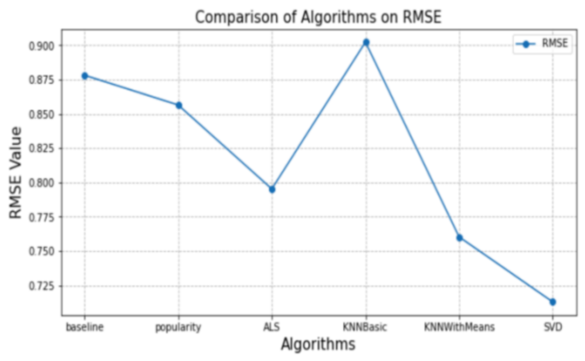
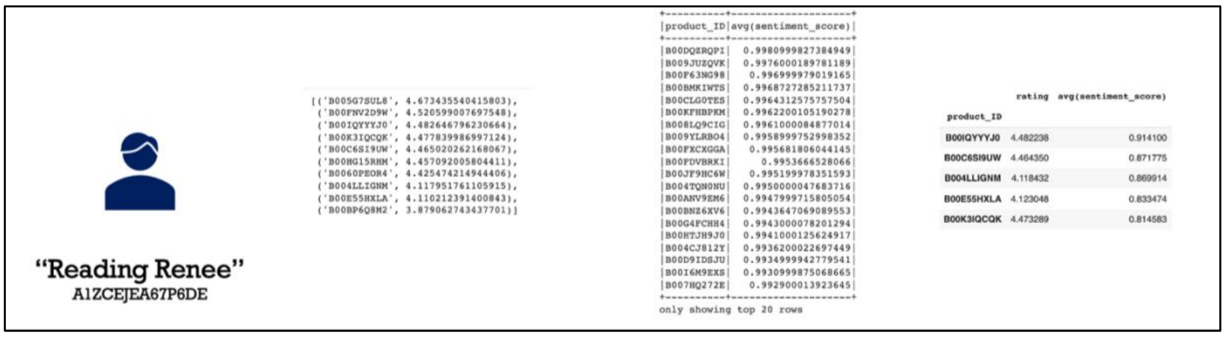


Figure 05 : Comparison of Root mean squared errors of implemented models

The selected model can successfully recommend Kindle Store products to customers based on their previous purchase history and other customers with similar likes and dislikes. These recommendations are further ordered by their average sentiment score of customer reviews and the top 5 products are recommended to the customer as the best recommendations. Figure 06 shows an example case. The collaborative recommender system recommends 10 products to the customer named “Reading Renee” and those 10 products are ordered according to their average sentiment score of customer reviews. Finally, the top 5 products from this ordered list are recommended to “Reading Renee” as the best recommendations. Note that the products ‘B00FNV2D9W’ has an estimated rating of 4.52 from SVD based collaborative recommender system but, it is not recommended to ‘Reading Rennee’ as the average sentiment score of that product is not high enough to be filtered to the final top 5 list.

Conclusion:

The root mean square error of the developed SVD model-based collaborative recommendation system is smaller compared to all the models that I implemented including the baseline model based on the mean value of rating. Usually, as the training set is large and sparse,



```
[('B005G7SUL8', 4.673435540415803),
 ('B00FNV2D9W', 4.520599007697548),
 ('B00IQYYYJ0', 4.482646796230664),
 ('B00K3IQCQK', 4.477839986997124),
 ('B00C6SI9UW', 4.465020262168067),
 ('B00HG15RHM', 4.457092005804411),
 ('B0060PEOR4', 4.425474214944406),
 ('B004LLIGNM', 4.117951761105915),
 ('B00E55HXLA', 4.110212391400843),
 ('B00BP6Q8M2', 3.879062743437701)]
```

```
+----------+--------------------+
|product_ID|avg(sentiment_score)|
+----------+--------------------+
|B00DQZRQPI|  0.9980999827384949|
|B009JUZQVK|  0.9976000189781189|
|B00F63NG98|   0.996999979019165|
|B00BMKIWTS|  0.9968727285211737|
|B00CLG0TES|  0.9964312575757504|
|B00KFHBPKM|  0.9962200105190278|
|B008LQ9CIG|  0.9961000084877014|
|B009YLRBO4|  0.9958999752998352|
|B00FXCXGGA|   0.995681806044145|
|B00FDVBRKI|     0.9953666528066|
|B00JF9HC6W|   0.995199978351593|
|B004TQN0NU|  0.9950000047683716|
|B00ANV9EM6|  0.9947999715805054|
|B00BNZ6XV6|  0.9943647069089553|
|B00G4FCHH4|  0.9943000078201294|
|B00HTJH9J0|  0.9941000125624917|
|B004CJ812Y|  0.9936200022697449|
|B00D9IDSJU|  0.9934999942779541|
|B00I6M9EXS|  0.9930999875068665|
|B007HQ272E|   0.992900013923645|
+----------+--------------------+
only showing top 20 rows
```

	rating	avg(sentiment_score)
product_ID		
B00IQYYYJ0	4.482238	0.914100
B00C6SI9UW	4.464350	0.871775
B004LLIGNM	4.118432	0.869914
B00E55HXLA	4.123048	0.833474
B00K3IQCQK	4.473289	0.814583

Figure 06: Recommendations for a customer from developed recommender system pipeline

ALS provides a more computationally cost-efficient method. Interestingly, ALS performed poorer compared to SVD for our data. It should be noted that ALS is only efficient in certain cases.

In all, this project is challenging and rewarding. The use of PySpark for data cleaning and model building was faster compared to the use of python. However, K-Nearest Neighbor Algorithm and Singular Value Decomposition algorithm are not yet implemented in PySpark and I had to use python surprise library for model building.

Less experience in handling big data tools such as AWS and MongoDB paid a huge disadvantage as I ended up using smaller data set instead of the planned 2018 - 4GB data set in Google Colaboratory. Nevertheless, as both 2018 and 2014 data sets have the same format, the developed pipeline for recommender systems can be easily adopted with a bigger dataset in any big data tools in the future. Also, the tested parameter range is limited due to extended computational time in Google Colaboratory. Extending the parameter search in a wide range would have helped me to find a better model with lower RMSE. Moreover, the 2018 Kindle Store dataset has a metadata file associated with it but, the 2014 data set does not. The metadata contains features of products such as price, brand name, etc. This data could have been used to develop a content-based filtering system and combine it with this collaborative-based system to get the best of both methods and improve RMSE of prediction.

References :

1. Image-based recommendations on styles and substitutes J. McAuley, C. Targett, J. Shi, A. van den Hengel SIGIR, 2015
2. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering R. He, J. McAuley, WWW, 2016
3. Zuzanna Klyszejko 2017, Recommendation systems with Apache Spark, https://zuzannna.github.io/Recomenderation-System-with-Spark/
4. Maverick 2020,Amazon Recommender System, Kaggle, https://www.kaggle.com/vipulbahl/amazon-recommender-system
5. Old Monk, 2021 Recommendation System on Amazon electronic data, Kaggle, https://www.kaggle.com/saurabhbagchi/recommendation-system-on-amazon-electronic-data
6. Sasmita Panigrahi, Rakesh Ku. Lenka, Ananya Stitipragyan, A Hybrid Distributed Collaborative Filtering Recommender Engine Using Apache Spark, Procedia Computer Science, Volume 83, 2016, Pages 1000-1006, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2016.04.214.
7. Apache Spark, 2018, VectorIndexer, https://spark.apache.org/docs/3.0.1/ml-features.html#vectorindexer
8. Saket Garodia, 2020, Building a recommendation engine to recommend books in Spark Towards Data Science , https://towardsdatascience.com/building-a-recommendation-engine-to-recommend-books-in-spark-f09334d47d67

9. Wenqiang Feng , 2017, text mining, Learning Apache Spark with Python, https://runawayhorse001.github.io/LearningApacheSpark/textmining.html#sentiment-analysis
10. Nicolas Hug, 2015, Revision fa745588, Tune algorithm parameters with GridSearchCV, surprise, https://surprise.readthedocs.io/en/stable/getting_started.html#tune-algorithm-parameters-with-gridsearchcv
11. Mohammad Murtaza Hashmi, 2020, Solving complex big data problems using combinations of window functions, deep dive in PySpark , Analytics Vidhya, https://medium.com/analytics-vidhya/solving-complex-big-data-problems-using-combinations-of-window-functions-deep-dive-in-pyspark-b1830eb00b7d