Implementing a Collaborative Filtering Recommender System using Map-Reduce

Pratima Kshetry(pkshetry@umd.edu)
University of Maryland, College Park

December 08, 2015

Abstract:

Recommender systems apply data mining and knowledge discovery techniques to the problem of making personalized recommendations to users. The user interaction data available from different products provide the major chunk of information available to build a recommender system. However, this huge availability of data and customer information available also poses key challenges for recommender system. Some of these are performing accurate recommendations to millions of users per second along with dealing with the data sparsity issues when building recommender systems.

In this study, I have implemented an item-item collaborative filtering recommendation system using Hadoop, MapReduce and Java technologies. This recommendation engine utilizes correlation metric to detect the degree of similarity between two users, and thus recommend items with higher correlation values to users. By doing this study, I have actually tried to solve a Big Data problem using the Amazon dataset. The Amazon dataset used to develop this recommendation system can be found at http://snap.stanford.edu/data/amazon-meta.html. The dataset contains metadata of 548,552 different products of Amazon with their review information

The few screenshots of how this system works can be found at the Appendix Section.

Keywords: Recommendation Engine, Collaborative Filtering, Java, MapReduce, Hadoop, Big Data

1. Introduction

Big data has become an important trend in the present field of data management. Today, data is being generated at unprecedented rate and volume (petabytes of data) from various sources such as social media interactions, web streams, electronic sensors and devices etc. Every individual, at every second with their every single action that ranges from swiping of credit cards to simple pushing of the text in their mobile phones generate bytes of data. According to the research conducted by MGI (McKinsey Global Institute, May 2011), this availability of data is bound to increase in the future and will surely make a foundation for market competition, reinforcing new waves of productive growth and consumer market. As organizations store more data in digital form such as online sales or user shopping preferences in their website, they can collect more precise and thorough key information on everything from product inventories to customer preferences' patterns. This learning of consumer behavior from the available big data will help organizations discover more efficient and effective ways to improve existing product lines and services. This will also help them to gain more competitive advantage as big data allows a narrower segmentation of customers. As a result, organizations will be able to deliver a much more specifically personalized products and services to their customers.

As the world today is becoming more and more virtual, recommendation engines are getting more popular and it has become increasingly important for businesses to develop a system that gives a more specific personalized recommendations to customers. According to third quarter 2015 retail e-commerce sales report produced by US Census Bureau, retail e-commerce sales for the third quarter of 2015 totaled $81.1 billion, an increase of 2.9 percent (±0.9%) from the second quarter of 2015 [1]. Also, the sales report of 2015 [2] shows that shoppers opted to buy online rather than fight the crowd in retail stores during November holiday season. As online shopping is getting more popular, it has become increasingly important for businesses to offer accurate recommendations to its customers. This is where the recommendations come to play a significant role in suggesting products to customers that are more relevant to their interests.

This paper as mentioned earlier, presents an implementation of a collaborative filtering algorithm by developing a recommendation engine. The collaborative filtering algorithm works by filtering information for a user based on a collection of user profiles generated. For a user, an information can be filtered in/out with respect to the behaviors of his or her similar users. This system considers users' ratings as a basis for computing the degree of similarity between two users.

2. Methodology:

The methodology consists of following modules.

1. Parse the amazon dataset.
2. Generating Customer Profiles.
3. Generating Product Profiles.
4. Clustering Data Points.
5. Correlation computation between two products from cluster points generated from step 3.
6. Ranking of the products according to correlation value thus obtained from step 5.
7. Generate a Recommend List of products to users based on step 6.

Since this paper addresses a data mining problem over a vast expanse of data, I chose to use the MapReduce framework of Hadoop to undertake such heavy computation. MapReduce scales well to many thousands of nodes and can handle petabytes of data. For recommendations where we have to find the similar products to a product a user may be interested at, we must calculate how similar pairs of items are [3]. For instance, if someone buys a book "The color of Magic", the recommender would suggest the book "Harry Potter" and the like. So we need to compute the similarity between two products (here books). There are many techniques to compute similarity between two items such as pearson correlation, cosine similarity, jaccard similarity etc. The system as developed during this study, utilizes person correlation measure to compute similarity between two products. This system also implements combinatorics approach to generate data points for the clustering phase which will ultimately be used to compute correlation between the product pair. For example, if the system has 50 products, we need to compute 2500 different combination pairs for the system. As a result, resulting dataset will be very large and sparse. And we have also to deal with temporal aspect since the user interest in products changes with time, so we need the correlation calculation done periodically so that the results are up to date. For this reason the best way to handle this scenario and problem is the divide and conquer pattern, and MapReduce is a powerful framework and can be used to implement data mining algorithms [3].

2.1 Parse the Amazon dataset.

The dataset as downloaded from the SNAP followed the following format:

```
Id:   15
ASIN: 1559362022
  title: Wake Up and Smell the Coffee
  group: Book
  salesrank: 518927
  similar: 5  1559360968  1559361247  1559360828  1559361018  0743214552
  categories: 3
   |Books[283155]|Subjects[1000]|Literature & Fiction[17]|Drama[2159]|United States[2160]
   |Books[283155]|Subjects[1000]|Arts & Photography[1]|Performing Arts[521000]|Theater[21
   |Books[283155]|Subjects[1000]|Literature & Fiction[17]|Authors, A-Z[70021]|( B )[70023
  reviews: total: 8  downloaded: 8  avg rating: 4
    2002-5-13  cutomer: A2IGOA66Y6O8TQ  rating: 5  votes:   3  helpful:   2
    2002-6-17  cutomer: A2OIN4AUH84KNE  rating: 5  votes:   2  helpful:   1
    2003-1-2  cutomer: A2HN382JNT1CIU  rating: 1  votes:   6  helpful:   1
    2003-6-7  cutomer: A2FDJ79LDU4O18  rating: 4  votes:   1  helpful:   1
    2003-6-27  cutomer: A39QMV9ZKRJXO5  rating: 4  votes:   1  helpful:   1
    2004-2-17  cutomer:  AUUVMSTQ1TXDI  rating: 1  votes:   2  helpful:   0
    2004-2-24  cutomer: A2C5K0QTLL9UAT  rating: 5  votes:   2  helpful:   2
    2004-10-13  cutomer:  A5XYF0Z3UH4HB  rating: 5  votes:   1  helpful:   1
```

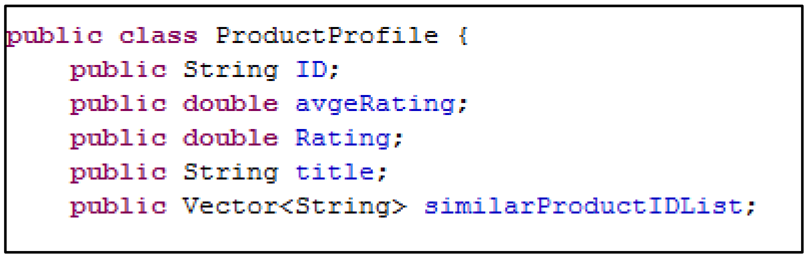
As a result, I needed to parse the dataset in order to able to generate customer and product profiles.

2.2 Generating Customer Profiles.

The parsed dataset was processed to generate the Customer Profiles. The structure of the customer profile is:

```java
public class CustomerDataReader extends RecordReader<Text, Text> {
    private  BufferedReader reader=null;
    private  String inputLine=null;
    private  ProductProfile currentProduct=new ProductProfile();
    //private  Vector<CustomerProfile> currentCustomerProfiles= new Vector<CustomerProfile>();
    private  Map<String,CustomerProfile> currentCustomerProfileMap= new HashMap<String,CustomerProfile>();
    private  CustomerProfile currentCustomer=null;
    private  int currentRecord=0; //currentRecord points to current index for the key,val in the currentCustomerProfiles
    private  boolean testMode=false;
    private int count = 0;
    private boolean endofLine=false;
```

2.3 Generating Product Profiles.
The parsed dataset was also processed to generate the Product/Item Profiles. The structure of the item profile is:



```
public class ProductProfile {
    public String ID;
    public double avgeRating;
    public double Rating;
    public String title;
    public Vector<String> similarProductIDList;
```

2.4 Clustering Data Points.
This module generates clusterPoints (R1, R2) rating pair for two products (P1, P2) for every two products rated by a given customer. In order to be considered for clustering, the product pair must be rated by at least two common customer. For example, if P1 and P2 are any two products rated by a customer C as (R1, R2), the data Point is (R1, R2) for keys (P1, P2). Since the data points to process and generate are too large, this system generates one output split for 6000 lines of customer profile being processed and grouped by (P1, P2) or if the clusterPoints being generated for a single customer reaches more than 1000000 points. The output split will have following information:

```
B000006O86          -     B00001U0DR : 2-2, 2-4
B000002OQH          -     B00005KARO : 5-5
```

In above example product B000006O86 and product B00001U0DR are rated by two common customers and their ratings were (2,2) (as rated by one customer) and (2,4) (as rated by another customer) respectively.

In order to generate all these data points, any data point from the customer who has rated more than 10,000 products are ignored. Since it is unlikely that one customer may have rated more than 10,000 product, inclusion of data points from such customer may lead to inaccurate results and a probable inclusion of bias in the output. Hence, such data points are ignored and have been flagged as scammers. Besides if such data points are considered, the data point combination will be huge with useless results. For instance, if these data flagged as scammers are considered, the data points produced by only considering these data points will be 49995000. [Note: For n products rated there will be nC2 pair of data points nC2= n! / (n-2)!2!.For n=10000; nC2=49995000].

2.5 Correlation computation between two products
All the data points as generated from 2.4 (Step 4) will be combined and grouped. The correlation between product pair is then computed for each product using the following formulae:

$$r = \frac{n(\sum xy) - (\sum x)(\sum y)}{\sqrt{[n(\sum x^2) - (\sum x)^2]\,[n(\sum y^2) - (\sum y)^2]}}$$

The product pair for correlation computation for this system is considered only if both the products have been rated by at least 4 customers. The correlation is computed as follows:

- ✓ For every item A and item B, find all users who have rated both item A and B.
- ✓ Use the ratings to form item A vector and item B vector respectively.
- ✓ Compute correlation between vector A and B.

The Output of the reducer will be of following format:

```
0001054600  -  0316769533   |5|   0.18463723
0001054600  -  0345339681   |7|   0.3363364
```

Where, 0001054600-0316769533 represent product pair, 5 is the number of customers who rated both the products, and 0.18463723 is correlation factor between the two products.

2.6. Ranking of the products

This module clusters products on the basis of their correlation value as computed from Step 2.5. For a product, all the other products correlated with it is sorted (using **Insertion Sort** technique) according to correlation factor obtained. The most highly correlated products are ranked first and so forth. Hence, the output of the reducer thus produced will be the basis of collaborative filtering for recommending the products to a user. The output of the reducer will be of following format:

```
0007154615:0007141076,1.0||0060098899,1.0||1402524994,0.9432422||140
2511787,0.875||0786247924,0.62931675||1559277807,0.0||0060199652,0.0
||0312422156,0.0||0694525596,-0.3227486||0375726403,-0.3227486
```

The output as shown above gives the list of correlated products for product 0007154615 as (x,y) pair where x is another productID and y is correlation factor between two product. For example, in the above example, product 0007141076 is correlated to product 0007154615 with correlation factor 1.0, product 0060098899 is also correlated to product 0007154615 with correlation factor 1.0. Thus, this output is used as a model for recommending customers on the basis of product they have purchased.

2.7 Generate a Recommend List

The recommended list for a customer consists of items that are most correlated to each of the products purchased by the customer. For instance, for a customer A10AR58JCYQ4CF who purchased product 086140324X which has ProductTitle: "Colour of Magic (Discworld Novels (Hardcover))", the format of the recommendation list is as follows:

ProductID	Correlation	Title
61020710	1	"The Color of Magic"
B00003CXI0	1	"Harry Potter and the Sorcerer's Stone"
B00003CXI1	1	"Harry Potter and the Sorcerer's Stone (Special Widescreen Edition)"
B00006IRHA	1	"Harry Potter and the Sorcerer's Stone Gift Set With Fluffy Collectible"
B00006IRHB	1	"Harry Potter and the Sorcerer's Stone Gift Set With Fluffy Collectible"

Thus, the recommendation list will list highly correlated products (the system is designed to consider products with correlation above 0.4) for each product the customer has purchased.

4. Evaluation:

For the evaluation purpose, 100 random customers with product purchased greater or equal to 20 and less than 30 are selected while the data points for clustering are generated. For each such customer, only half of the products rated by them are fed into system. These products are exported to include.txt file which serves the purpose of Training data. For eg: a single entry in include.txt contain data in following format:

```
customerID: NumofProductRated - productId, rating || productId, rating
```

For example, one entry of the include.txt file looks like:

```
A109VVAISTEUKY:25-B00008AJCH,3  ||  0671695347,3  ||  1587240068,3||
0743520319,3|| 0553280406,2 || 0833508024,5
```

The above example shows data for one customer A109VVAISTEUKY and the [ProductID, Rating] data points as considered for the system.

For the same customer, the other half of the products rated by the same customer are not fed into system and hence they serve the purpose of Test data. These data are exported in exclude.txt. The format is similar to include.txt file.

Now, as the recommendation list consists of universal set of all recommended products for each of the product purchased by the customer. From this list, I have only considered products that are in include.txt for customer. So for testing purpose, I have filtered out recommended products for only those products that are considered in include.txt, and then this recommended products are checked to see if they also fall in the exclude.txt file for the same customer. This way, we can check to see how many of the recommended item are purchased by the customer.

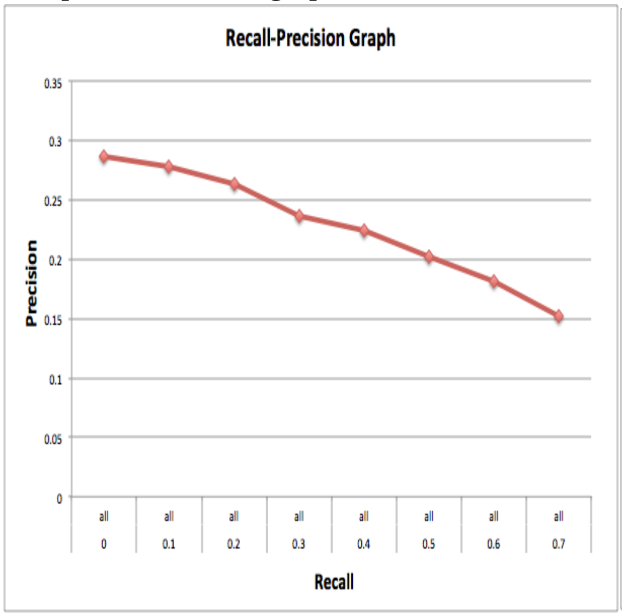
If the subset considering the above example for customer A109VVAISTEUKY is considered, following is an example of a subset of universal set for customer A109VVAISTEUKY;

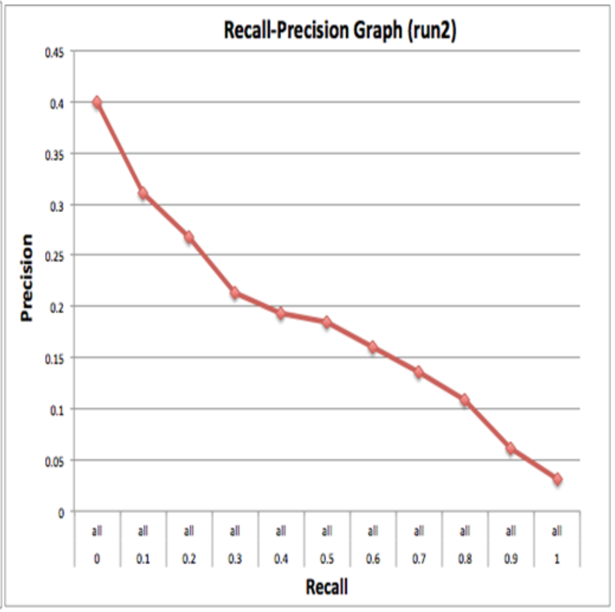
Recommended Product for B00008AJCH	**Recommended Product for 1587240068**
0743520300,1.0	0788761331,1.0
0743520319,1.0	1400047285,1.0
0743225716,1.0	0192723693,1.0
0743233387,1.0	0553527568,1.0
0970309902,0.91287094	1402505396,1.0
0743504232,0.6721591	0743504704,1.0

As we can see, product B00008AJCH and 0743520300 has correlation of 1.0 , B00008AJCH and 0743504232 has correlation of 0.6721591. For the above small sample of subset, three recommended products 0375405445, 0788761331, 0743504232 are in the excluded list which implies that these products were actually bought by customer.

The other evaluation metric for testing the efficiency of this system was the Precision/Recall metric. The Precision measure considers the relevancy of the document while evaluating the efficiency of the system. For my system, however recall (than Precision) would be a better measure as we only have information of what customer already bought than what he will buy. Generally, when the number of recommendations that can be made to the user is pre-determined, the most useful measure for evaluation is Precision at N. In the real world scenario, businesses send survey forms to few selected customers which contain the list of recommended items as generated by the recommender system and customers are asked to check which product are more relevant to their interests. This will help to sort out relevant items from the recommended list and hence help to evaluate precision of the recommender system. However, in our case, we only have information of what the customer has already purchased but we have no way to know what the relevant products could be to that particular customer. As a result, I opted to check how many of the recommended items falls in the already purchased item list (recall) for the customer by following the steps as described earlier in the evaluation phase. For this system, considering a sample size of n=20 customers, the average recall score for this system was computed to be 0.776

The precision-recall graph as obtained for n=20 customers and n=30 customers is shown below:





The above figure shows the P@10 values of all 20 customers of the first run while the second figure (run2) shows precision recall graph for n=30 customers. Overall the P@10 improved by approximately 5.4% when the sample size increased by 10. The results obtained suggests that with a large increase in the sample size, the Precision@10 might also increase. So with a much larger sample of customers we will tend to get higher Precision@10 values. The evaluation results suggest that the algorithm when scaled to a much larger sample of customers will perform consistently in recommending items to a diverse population.

Also, a few visual observation checks of the recommendation output confirmed that the system is working as expected. For example, for customer A109VVAISTEUKY who bought product B00008AJCH. The profile for this product is:

Product Id: B00008AJCH
Title: "Survival Is Not Enough: Zooming, Evolution, and the Future of Your Company"
AvgRating: 3.5
Similar Products: 0786887176 0743227905 1591840414 0684856360 159184021X

The recommendations produce for customer A109VVAISTEUKY based on one of his purchase B00008AJCH consisted of following:

Product Id	**Title**
743520300	Survival is not Enough : Zooming, Evolution, and the Future of Your Company
743520319	Survival is Not Enough : Zooming, Evolution, and the Future of Your Company
743225716	Survival Is Not Enough: Zooming, Evolution, and the Future of Your Company
743233387	Survival Is Not Enough: Why Smart Companies Abandon Worry and Embrace Change
970309902	Unleashing the Ideavirus
743504232	Unleashing The Idea Virus

As seen from the above example, the recommendation system thus designed and developed works as expected.

Problems Encountered:

I had to deal with resource management issues while trying to generate data points for the system (the clustering phase). Our dataset had more than 10 million records. Since I opted to compute the correlation for each item-pair, the total number of combination pairs for the correlation computation was extremely huge. As a result, I had to re-run the clustering module four to five times to generate the final data points after reassigning hard disk space and memory. I was able to run the clustering module only after eliminating the customer data points which were flagged as scammers. As a result, the whole clustering phase took a whole of 1 week to complete.

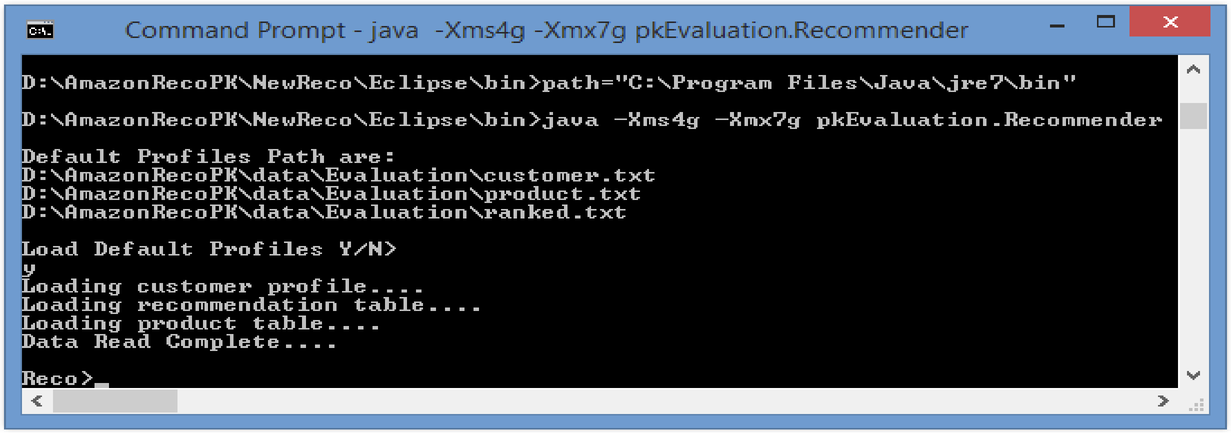
Discussion and Conclusion:

This system considers only one parameter of user behavior i.e rating. Using other variables of user behavior such as number of votes considered useful for a user review, we can assign weightage of certain degree and include it in our recommender system to develop more accurate recommendation to a user.
Also, classification of users as Scammers and Non-Scammers during the clustering phase can also be used for fraud detection.

The source code for this project is made available in github. The code can be reused upon giving credits to the original author. The source code for this project is available at :
https://github.com/pratimaKshetry/CollaborativeFilteringRecommendation

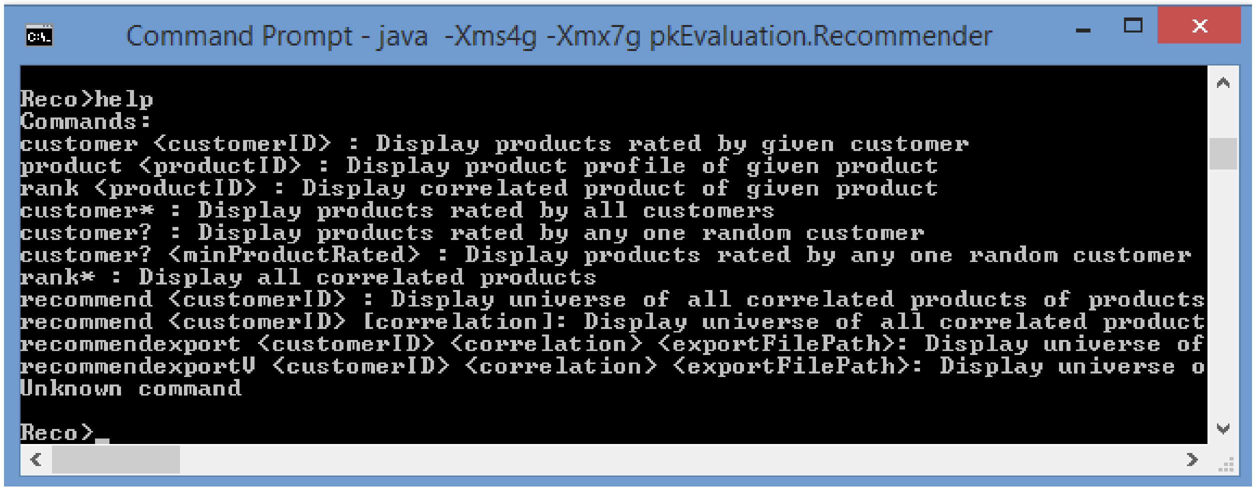
Appendix:

1. First screen after loading the program.


Command Prompt - java -Xms4g -Xmx7g pkEvaluation.Recommender

```
D:\AmazonRecoPK\NewReco\Eclipse\bin>path="C:\Program Files\Java\jre7\bin"

D:\AmazonRecoPK\NewReco\Eclipse\bin>java -Xms4g -Xmx7g pkEvaluation.Recommender

Default Profiles Path are:
D:\AmazonRecoPK\data\Evaluation\customer.txt
D:\AmazonRecoPK\data\Evaluation\product.txt
D:\AmazonRecoPK\data\Evaluation\ranked.txt

Load Default Profiles Y/N>
y
Loading customer profile....
Loading recommendation table....
Loading product table....
Data Read Complete....

Reco>_
```

2. The “Help” menu:

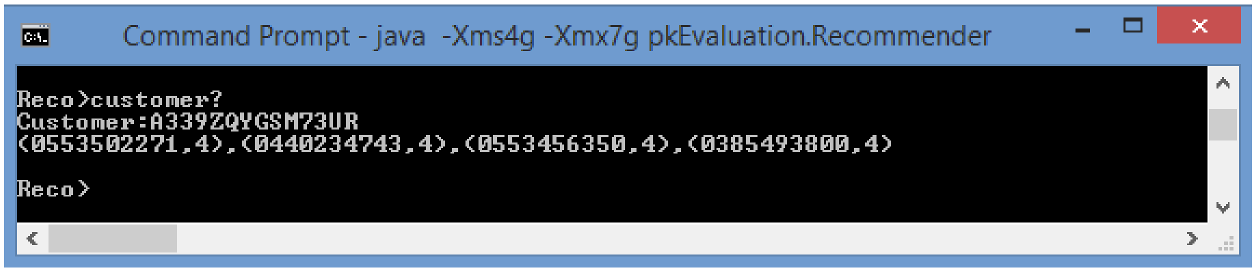

Command Prompt - java -Xms4g -Xmx7g pkEvaluation.Recommender

```
Reco>help
Commands:
customer <customerID> : Display products rated by given customer
product <productID> : Display product profile of given product
rank <productID> : Display correlated product of given product
customer* : Display products rated by all customers
customer? : Display products rated by any one random customer
customer? <minProductRated> : Display products rated by any one random customer
rank* : Display all correlated products
recommend <customerID> : Display universe of all correlated products of products
recommend <customerID> [correlation]: Display universe of all correlated product
recommendexport <customerID> <correlation> <exportFilePath>: Display universe of
recommendexportV <customerID> <correlation> <exportFilePath>: Display universe o
Unknown command

Reco>_
```

3. Customer Profile of any one random customer

The command to generate profile of any random customer is:

Reco>customer?


Command Prompt - java -Xms4g -Xmx7g pkEvaluation.Recommender

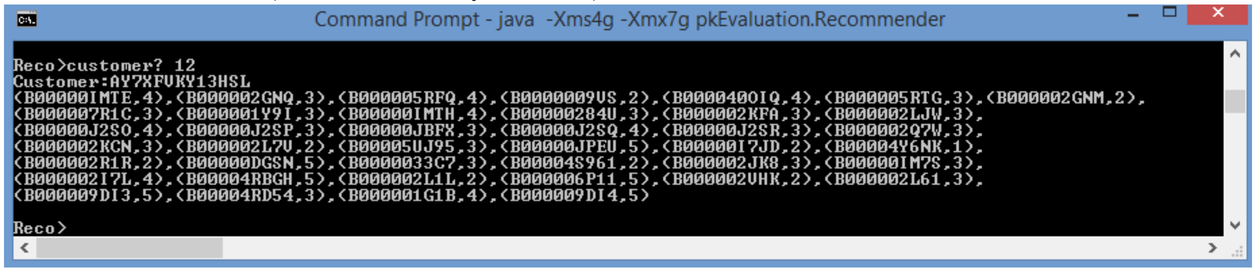
```
Reco>customer?
Customer:A339ZQYGSM73UR
(0553502271,4),(0440234743,4),(0553456350,4),(0385493800,4)

Reco>
```

The format of output is <ProductId, Rating>

4. Customer Profile of any customer who has rated “n” products:

The command to generate customer Profile who has rated n products is:

Reco>Customer? N (where ‘n” is any number)

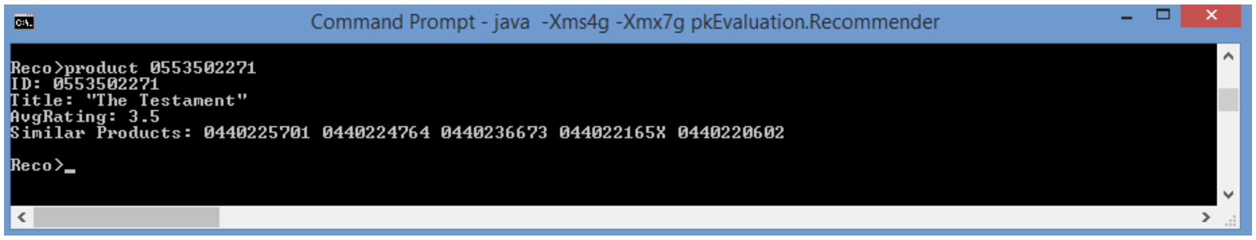


```
Reco>customer? 12
Customer:AY7XFVKY13HSL
<B00000IMTE,4>,<B000002GNQ,3>,<B000005RFQ,4>,<B0000009VS,2>,<B000040OIQ,4>,<B000005RTG,3>,<B000002GNM,2>,
<B000007R1C,3>,<B000001Y9I,3>,<B00000IMTH,4>,<B00000284U,3>,<B000002KFA,3>,<B000002LJW,3>,
<B00000J2SO,4>,<B00000J2SP,3>,<B00000JBFX,3>,<B00000J2SQ,4>,<B00000J2SR,3>,<B000002Q7W,3>,
<B000002KCN,3>,<B000002L7V,2>,<B00005UJ95,3>,<B00000JPEU,5>,<B00000I7JD,2>,<B00004Y6NK,1>,
<B000002R1R,2>,<B00000DGSN,5>,<B0000033C7,3>,<B00004S961,2>,<B000002JK8,3>,<B00000IM7S,3>,
<B000002I7L,4>,<B00004RBGH,5>,<B000002L1L,2>,<B000006P11,5>,<B000002VHK,2>,<B000002L61,3>,
<B000009DI3,5>,<B00004RD54,3>,<B000001G1B,4>,<B000009DI4,5>

Reco>
```

5. Product Profile

The command to generate product Profile is:

Reco>Product ProductID

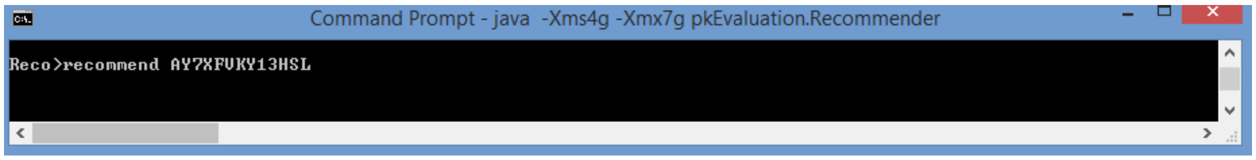


```
Reco>product 0553502271
ID: 0553502271
Title: "The Testament"
AvgRating: 3.5
Similar Products: 0440225701 0440224764 0440236673 044022165X 0440220602

Reco>_
```

6. The “Recommend” Command

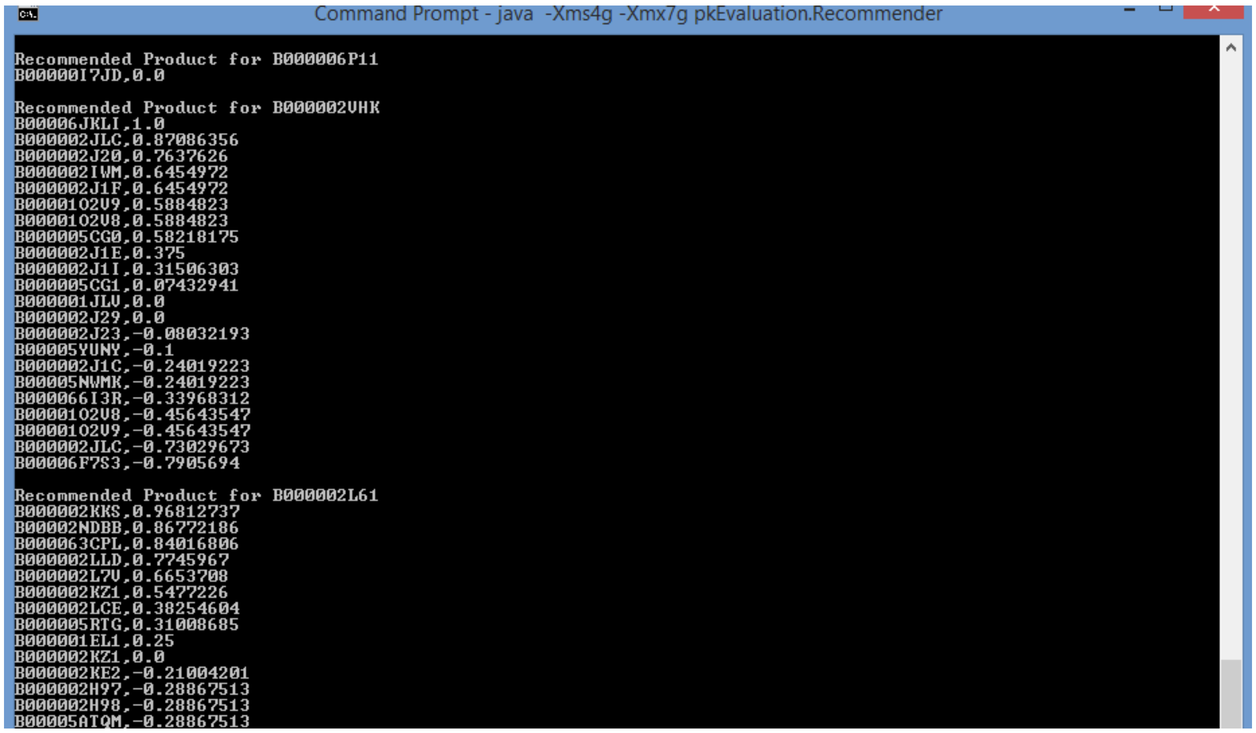
The format of the command is:

Reco> Recommend CustomerID



```
Reco>recommend AY7XFVKY13HSL
```

7. Output of the “Recommend” command



Command Prompt - java -Xms4g -Xmx7g pkEvaluation.Recommender

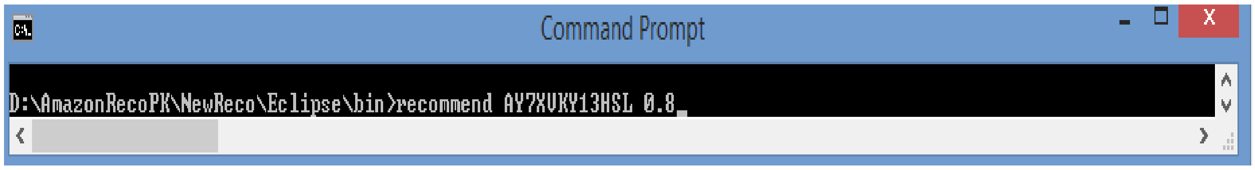
```
Recommended Product for B000006P11
B00000I7JD,0.0

Recommended Product for B000002VHK
B00006JKLI,1.0
B000002JLC,0.87086356
B000002J20,0.7637626
B000002IWM,0.6454972
B000002J1F,0.6454972
B000102V9,0.5884823
B000102V8,0.5884823
B000005CG0,0.58218175
B000002J1E,0.375
B000002J1I,0.31506303
B000005CG1,0.07432941
B000001JLV,0.0
B000002J29,0.0
B000002J23,-0.08032193
B00005YUNY,-0.1
B000002J1C,-0.24019223
B00005NWMK,-0.24019223
B000066I3R,-0.33968312
B000102V8,-0.45643547
B000102V9,-0.45643547
B000002JLC,-0.73029673
B00006F7S3,-0.7905694

Recommended Product for B000002L61
B000002KKS,0.96812737
B00002NDBB,0.86772186
B000063CPL,0.84016806
B000002LLD,0.7745967
B000002L7V,0.6653708
B000002KZ1,0.5477226
B000002LCE,0.38254604
B000005RTG,0.31008685
B000001EL1,0.25
B000002KZ1,0.0
B000002KE2,-0.21004201
B000002H97,-0.28867513
B000002H98,-0.28867513
B00005ATQM,-0.28867513
```

The format of the Output is <Product, Correlation Value>.

8. Recommending products to users above certain correlation value

The command to generate recommended list to users higher than x correlation value is:

Reco>recommend customerID correlationValue

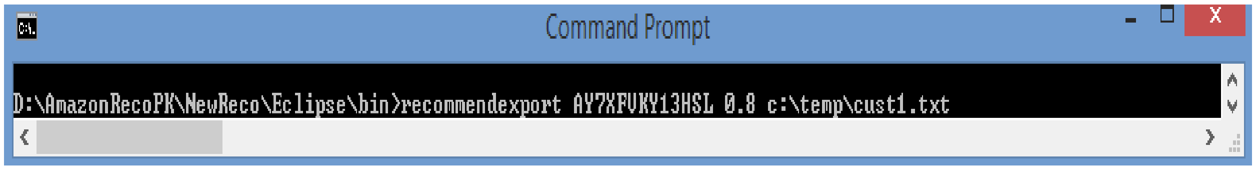


```
D:\AmazonRecoPK\NewReco\Eclipse\bin>recommend AY7XVKY13HSL 0.8_
```

9. Export recommendation list to a file

The command to export recommendation list to a file is:

Reco> Recommenderexport customerID correlationValue filepath



```
D:\AmazonRecoPK\NewReco\Eclipse\bin>recommendexport AY7XFVKY13HSL 0.8 c:\temp\cust1.txt
```

References:

1. Quarterly retail e-commerce sales 3rd quarter 2015. Available at: https://www.census.gov/retail/mrts/www/data/pdf/ec_current.pdf

2. Marcia Kaplan, December 02, 2015. Available at: http://www.practicalecommerce.com/articles/94777-Sales-Report-2015-Thanksgiving-Day-Black-Friday-Cyber-Monday

3. http://aimotion.blogspot.com/2012/08/introduction-to-recommendations-with.html