Product Recommender Engine

Teams:
Lakshmi Udupa 800956319
Shreyas Subramanya Bhat 800958406

Introduction:

Our project aims at recommending products to users based on the ratings of other similar products given by them. In our project we have implemented Alternating Least Square method which is a collaborative filtering technique used to build a recommender engine. Collaborative filtering techniques aim to fill in the missing entries of a user-item association matrix. We have used Spark, Python in this project.

Motivation:

Modern day e-commerce websites are heavily reliant on recommendation systems to sell the right product to the right consumer. It helps improve customer satisfaction and needs. A recommendation system is said to be a success if the recommended product is purchased by the customer. We were extremely intrigued by this concept and the potential applications beyond shopping needs. Having advanced distributed cloud computing technologies such as Apache Spark, we wanted to take up the challenge of creating a recommender system for a real company - Amazon and using their data collection. We also wanted to investigate the power of Spark's machine learning library and compare its performance against a recommender engine created by us based on a slightly different approach.

Approach:

Collaborative filtering analyzes relationships between users and interdependencies among products to identify new user-item associations. Interesting feature of the collaborative filtering analysis is that it is domain free, yet it can address data aspects that are often elusive and difficult to profile using content filtering. The two primary areas of collaborative filtering are the neighborhood methods and latent factor models.

In this project we have implemented two collaborating filtering algorithms: **Cosine similarity** and **alternating least squares method**. We have devised a recommender engine which recommends items to a user based on items previously rated by other similar users.

User-based Cosine similarity algorithm is a neighborhood method of collaborative filtering in which user is predicted ratings and products based on the cosine score.
Figure 1 shows the collaborative filtering process. Figure 2 shows the isolation of the co-rated items and similarity computation. Similarity between items i and j, denoted by sim(i, j) is given by Eq(1) :

$$sim(i,j) = cos(\vec{i}, \vec{j}) = \frac{\vec{i} \cdot \vec{j}}{||\vec{i}||_2 * ||\vec{j}||_2}$$

<u>*Eq(1)*</u>

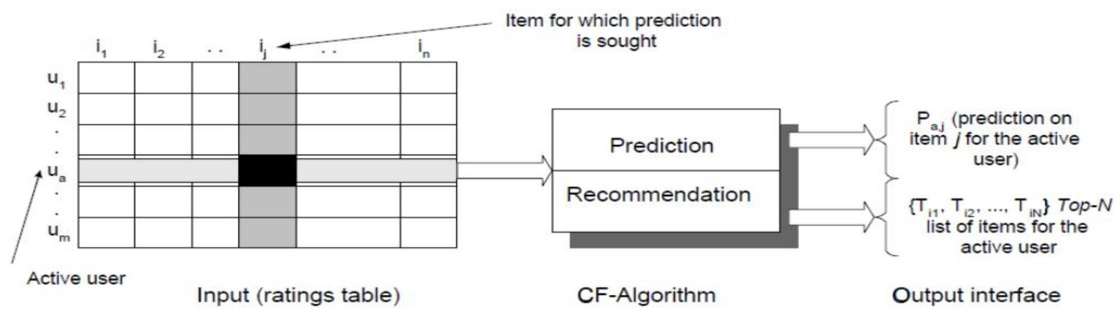


Figure 1: The collaborative filtering process.

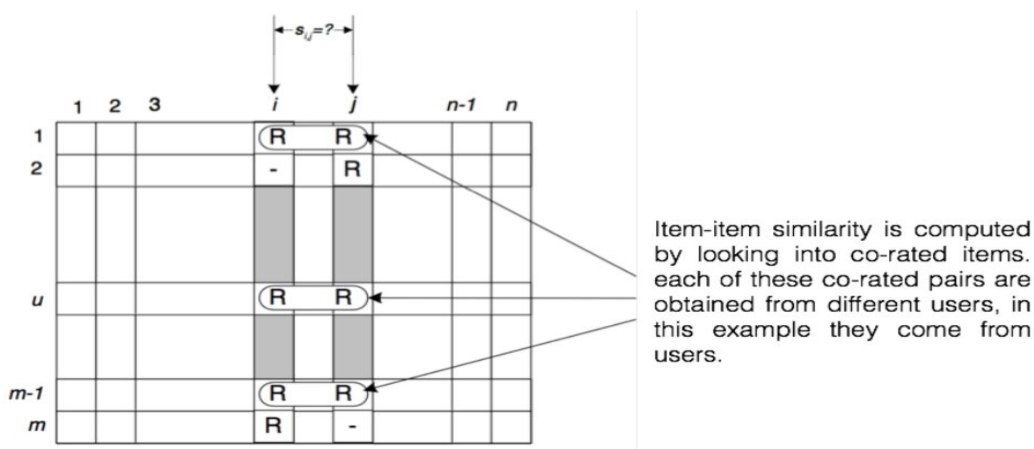
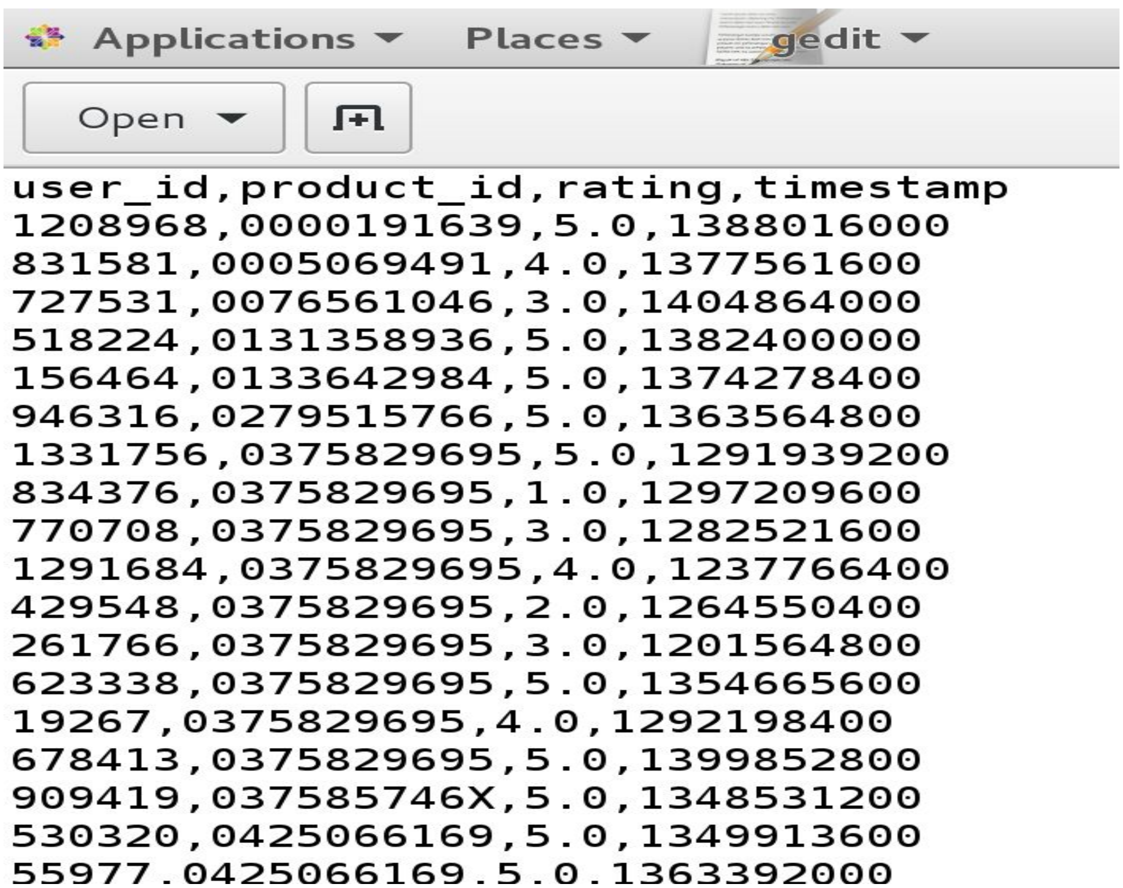


Figure 2: Isolation of the co-rated items and similarity computation.

Alternating least squares algorithm is a latent factor model, we have implemented this using Spark. Spark supports model-based collaborative filtering through *spark.mllib*, in which users and products are described by a small set of latent factors that can be used to predict missing entries. In the recommender engine using ALS we have implemented code to display customized recommendations based on products that have not been rated by the user(assumption being they have not yet purchased those products).

Datasets involved:

We have used amazon product data (http://jmcauley.ucsd.edu/data/amazon/links.html) for this recommender engine.
For Cosine similarity implementation we have used a ratings_only dataset. This dataset is a pre-cleaned, csv file which contains only user_id , product_id , ratings and unix timestamp. This dataset contains alphanumeric user_id, product_id and ratings in decimal with lowest 0.0 and highest 5.0. The user_id is encoded by our cosin_data_preprocessor.py, which also adds headers to the dataset. Fig:3 shows the processed dataset. ratings2.csv is the name of the file used.




```
user_id,product_id,rating,timestamp
1208968,0000191639,5.0,1388016000
831581,0005069491,4.0,1377561600
727531,0076561046,3.0,1404864000
518224,0131358936,5.0,1382400000
156464,0133642984,5.0,1374278400
946316,0279515766,5.0,1363564800
1331756,0375829695,5.0,1291939200
834376,0375829695,1.0,1297209600
770708,0375829695,3.0,1282521600
1291684,0375829695,4.0,1237766400
429548,0375829695,2.0,1264550400
261766,0375829695,3.0,1201564800
623338,0375829695,5.0,1354665600
19267,0375829695,4.0,1292198400
678413,0375829695,5.0,1399852800
909419,037585746X,5.0,1348531200
530320,0425066169,5.0,1349913600
55977,0425066169,5.0,1363392000
```

Fig:3

For Alternating least square method, we have used two datasets.

1) Ratings_only dataset which contains user_id(string), product_id(string), ratings(0.0-5.0, float) and unixtimestamp. This ratings_only file is then encoded by als_data_preprocessor.py. Fig:4 shows the cleaned and encoded ratings only data. The file name is ratings_als.csv.

```
1208968|0|5.0|1388016000
831581|1|4.0|1377561600
727531|2|3.0|1404864000
518224|3|5.0|1382400000
156464|4|5.0|1374278400
946316|5|5.0|1363564800
1331756|6|5.0|1291939200
834376|6|1.0|1297209600
770708|6|3.0|1282521600
1291684|6|4.0|1237766400
429548|6|2.0|1264550400
261766|6|3.0|1201564800
623338|6|5.0|1354665600
19267|6|4.0|1292198400
678413|6|5.0|1399852800
```

Fig:4

2) Metadata.json.gz which contains product_id, product_title, price, imageURL, relatedProducts, salesRank, brand and category. This metadata.json.gz file is converted to a .csv file by our metadata_preprocessor.py and only product_id, product_title are kept. In this metadata.csv file, product_id is encoded by metadata_encoder.py. Fig:5 shows cleaned and encoded metadata. The file name is metadata.csv.

```
Open ▾                                                                 meta_data.c
                                                                       ~/Downloads/amaz
0#Dr. Suess 19163 Dr. Seuss Puzzle 3 Pack Bundle#[['Toys & Games', 'Puzzles', 'Jigsaw Puzzles']]
1#Nursery Rhymes Felt Book#[['Toys & Games']]
2#Fraction Decimal Percent Card Deck#[['Toys & Games', 'Learning & Education', 'Flash Cards']]
3##[['Toys & Games', 'Learning & Education', 'Mathematics & Counting']]
4#Algebra 2 California Teacher Center#[['Toys & Games', 'Learning & Education', 'Mathematics & Counting']]
5#Vintage 1982 Strawberry Shortcake Doll#[['Toys & Games', 'Dolls & Accessories', 'Dolls']]
6#Dr. Seuss Jigsaw Puzzle Book: With Six 48-Piece Puzzles#[['Toys & Games', 'Puzzles', 'Jigsaw Puzzles']]
7#Blues Clues On the Go with Blue Color &amp; Activity Book#[['Toys & Games', 'Arts & Crafts', 'Drawing & Pain
8#Sanctuary: The Thieves' World Boardgame [BOX SET]#[['Toys & Games', 'Games', 'Board Games']]
9#Clifford The Big Red Dog: 10-Piece Counting &amp; Alphabet Nesting and Stacking Blocks#[['Toys & Games', 'Le
10#3D Puzzle Buster#[['Toys & Games', 'Puzzles', '3-D Puzzles']]
11#Mrs. Green: The Teacher From the Black Lagoon Hand Puppet#[['Toys & Games', 'Stuffed Animals & Plush', 'Pup
12#TEACHERS FRIEND NOTE FROM THE TEACHER SCHOOL TO HOME NOTES#[['Toys & Games', 'Learning & Education', 'Flash
13#Scholastic International Flags Borders with Corners (TF2939)#[['Toys & Games', 'Party Supplies']]
```

Fig:5

3) Single_user_rating.txt is a file which contains user_id, product_id, rating, timestamp and product_title. Which has encoded user_id, product_id. Fig:6 shows the single user rating file.

```
Open ▾
0|525|5.0|1400000000|Duck Duck Safari
0|625|3.0|1400000000|Capture the Gag
0|744|2.5|1400000000|Superman Vs. Doomsday Collector Set
0|1219|4.5|1400000000|Britannia
```

Fig:6

Implementation:

1) **Algorithm:**

Cosine-similarity method:

a. Input: We take three arguments from the user which are ratings file, single user_id and product_id for which recommendations are required.
b. Output:
    i. Top ten recommended products based on input product.
    ii. Ten most similar users based on input user.
    iii. List of products based on other user ratings for the input user.
    iv. Evaluation metric: Root mean square(RMSE) error value.

c. Procedure:
    i. We take a subset of input records roughly 10000 due to computational reasons.
    ii. We create a pivot table, with user_id as index, product_id as columns and ratings as values. This is our factor matrix containing a vector representation of products for each user.
    iii. We then normalize the ratings in order to standardize values for users with one rating or where user has rated the same for all the products.
    iv. We then drop rating values which are zero.
    v. We now create a sparse matrices to perform further computations.
    vi. We compute the cosine similarity for the items using the sparse matrix and transpose of this matrix to compute cosine similarity for the users.
    vii. We display top products based on cosine similarity for the input product.
    viii. We display top users based on cosine similarity for the input user_id.
    ix. Now we predict rating by taking similar users and their cosine similarity value. We go through each of these users and find the ratings for each product that they have rated. Wherever there exists a rating we multiply it with the similarity value and store this value. We then return sum(rating list)/ sum(weight list) which is the similarity. Where Rating list is product of rating and similarity and Weight list is similarity.
    x. To get the RMSE value for the prediction we find the rating for the actual value and the value returned by the predict rating functions. These errors are stored as a list and then RMSE is computed.

<u>Alternating least squares:</u>

a. Inputs: rating file, metadata file, single user rating file.
b. Outputs:
    i. RMSE values for various training parameters for the model.
    ii. Baseline improvement percentage.
    iii. List of top 50 products recommended for the given single_user_rating file data.
c. Procedure:
    i. The rating file is loaded and parsed to retrieve last digit of timestamp as key and user_id, product_id and rating as value tuple.
    ii. We similarly parse metadata file to form a product_id as key and title as value pair.
    iii. We split the ratings data set in a ratio of 6:2:2 for training, validation and testing respectively.

iv. We use the spark.mllib and ALS algorithm to implement an explicit training model.
v. Once we train this model and determine the best parameters to obtain an RMSE as close as possible to 1. We save this model for further predictions.
vi. We use validation and test sets against this model to determine model performance against naive baseline.
vii. Finally we determine a set of products that have not been rated by the user to make personalized recommendations.

2) Frameworks and Packages:

- Apache Spark - Pyspark
- Additional python packages used:
  - pandas : Python package providing fast, flexible, and expressive data structures designed to make working with structured (tabular, multidimensional, potentially heterogeneous) and time series data both easy and intuitive.
  - scipy : Python-based ecosystem of open-source software for mathematics, science, and engineering. We use it here to check for sparse matrix.
  - sklearn : An open source Python library that implements a range of preprocessing, cross-validation and visualization algorithms.

Challenges faced during implementation:

1. Obtaining the right data for the right task is very difficult. We originally intended to use a dataset from yahoo to make similar user interest predictions based on search data. The approval to obtain the dataset took quite some time and also the dataset itself came as a surprise having completely anonymized data that was very difficult to interpret with limited documentation. Hence we decided to use amazon product rating dataset.
2. In Spite of having a well structured dataset collected from amazon. We were further challenged with encoding categorical data to numeric data for computational purposes and performing further cleaning and data alignment operations.
3. With the cosine-similarity user based collaborative filtering approach we were challenged with the scale of users which made computation difficult without parallelization. This is the case with user based collaborative filtering because

the number of products that are sold are much lesser than the total number of users that can be shopping.

Conclusion:

The *performance metric* which we used to compare predictions against the actual value was the RMSE. The RMSE values for the two implementations are as follows:

Cosine-Similarity method:
RMSE = 2.0

Alternating least squares method:
The best model was trained with rank = 12 and lambda = 0.1, and numIterations = 20, and its RMSE on the test set was 1.534974.

We observed that ALS implementation performs better and since it is model based it is easy to make quick recommendations.

We have accomplished the following:

- traditional collaborative filtering algorithm which represents a customer as an N-dimensional vector of interests.
- unsupervised clustering to find if user/ customers can be clustered based on their common interests (rating).

What we could not accomplish:

- Incorporate geographic location into the model to help better narrow the users and their interests since the dataset did not contain this information.
- Implement a web crawler that can take the user interests to fetch websites that sell a particular product. This could help us identify potential competition or threats to market a product.

Future Scope:

- Provide interactive front-end and database in the backend to store the recommendations.
- Incorporate geographic location into the model to help better narrow the users and their interests.

Work Division:

Task	Lakshmi	Shreyas
Research	X	X
Programming	X	X
Dataset handling	-	X
Dataset cleaning	X	-
Documentation	X	X

Results screenshots:

Cosine-similarity method:

```
Similar products to 0439893577 include:
No. 1: B005FRMDIU
No. 2: B005FDYCBK
No. 3: B005FG34QQ
No. 4: B005FH3XDO
No. 5: B005FHJPAO
No. 6: B005FIFJW6
No. 7: B005FLN1SC
No. 8: B005FLN1TG
No. 9: B005FLUJ1O
No. 10: B005FOTPQ6


Most Similar Users:
User #120, Similarity value: 0.58
User #9753, Similarity value: 0.50
User #6711, Similarity value: 0.00
User #6699, Similarity value: 0.00
User #6692, Similarity value: 0.00
User #6682, Similarity value: 0.00
User #6679, Similarity value: 0.00
User #6665, Similarity value: 0.00
User #6654, Similarity value: 0.00
User #6653, Similarity value: 0.00


List of products based on other user ratigns for the current user:
[('B0000Y70QE', 1), ('B0057MGKO6', 1), ('B003BNZOYU', 1), ('B007HZLT4W', 1), ('B00000JGWY', 1)]


RMSE is:  2.0
```

Fig:7

Alternating least squares method:

```
Got 2252771 ratings from 1342911 users on 327698 products.
Training: 1348206, validation: 448943, test: 455626
RMSE (validation) = 2.576039 for the model trained with rank = 12, lambda = 0.0, and numIter = 20.
RMSE (validation) = 1.546958 for the model trained with rank = 12, lambda = 0.1, and numIter = 20.
The best model was trained with rank = 12 and lambda = 0.1, and numIter = 20, and its RMSE on the test set is 1.534974.
The best model improves the baseline by -20.45%.
Products recommended for you:
 1: Dora with Stars Backpack with Lunch Case
 2: Paddington Bear Luncheon Napkin
 3: LEGO Star Wars: Droid Fighter (7111)
 4: Winning Moves Games Rubik's You Can Do It
 5: HOT WHEELS 2010 NEW MODELS 22 OF 44 WHITE '08 VIPER SRT10 ACR
 6: Magnetic Colors Book
 7: Purple-kids Washable Ink Stamp Pad - Childsafe
 8: Family Guy - Stewie Head Football
 9: Phineas And Ferb Lenticular Puzzle - 150 Pieces
10: Happy Birthday Foil Banner - PINK - 2.7m
11: Breyer Run-In Barn - Wood
12: Disney Celebrations Vinylmation 'Happy Birthday' - 3&quot;
13: Yu-Gi-Oh! - Granmarg the Mega Monarch (SHSP-EN041) - Shadow Specters - 1st Edition - Secret Rare
14: Star Wars 84866 Dexter Jettster Coruscant Informant Action Figure - Attack of the Clones
15: The Mummy - Legionnaire O'Connell with Officer's Uniform!
16: 16 Inch Raggedy Ann Classic Doll
17: Kidoozie jr Series Rocket Zoomer Novelty
18: LiL' Bratz Wallet
19: Heads &amp; Tails Lion
20: WrestleMania XV Signature Series 3: &quot;Jacqueline&quot;
21: Maiden Mary
22: OFNA Racing Pull Start Unit w/Rope, Force .21, .26, .28
23: Yu-Gi-Oh! - Hysteric Party (SD8-EN027) - Structure Deck 8: Lord of the Storm - 1st Edition - Common
24: Kidrobot Happy Labbit Mini Plush Figures
25: Bendon Spider Man and Friends Board Books (Set of 4)
26: 50 Pieces Micro Timber by Maxim Enterprise (53051)
27: 'NSync Collectible Marionette - Justin Timberlake Doll
28: Combat Pure Gear Youth Baseball Bat -12
29: Kyosho EP 2WD Sand Master Combo Kit (1/10 Scale), Color Type 1 Red
30: Marvel The Amazing Spider-Man, Stealth Venom (BLACK) Action Figure
31: LittleBigPlanet 24&quot; Sackboy
32: Learning Resources Cash Bash
33: Love Me Playing Cards
34: Pride Eagle Plus Toy Ready to Stuff
35: Alhambra: Big Box
36: White Mountain Puzzles Our Friends - 1000 Piece Jigsaw Puzzle
37: Hollywood Squares by Parker Brothers
38: Winnie the Pooh Sandbox and Umbrella Set
39: Mini Soccer Ball and Players Erasers 42 Pcs
40: 1963 Chevy Impala SS Hard Top 1/18 White
41: Hand Puppet - Pig
42: Peanuts Charlie Brown Christmas Playset - Charlie the Play Director
43: LeapFrog Learn And Groove Xylophone Zoo
44: LeapFrog My Own Story Time Pad
45: Daisy Rock WildWood Acoustic Short Scale Guitar, Bleach Blonde
46: Full Metal Alchemist Snake Necklace with Bracelet
47: Aurora Plush 16 inches Mama and Baby Lynx
48: Disney Infinity Exclusive Rare 5 Pcs Set - Included: Power Disc 3 Pcs (Mike's Car, Tron User Control, Zurgs Wrath) + Toys
49: Little Tikes Sport Coupe
50: Care Bears Brave Heart Lion 8&quot; Plush
```

Fig:8

References:

[1]:
https://cseweb.ucsd.edu/~jmcauley/cse255/reports/wi15/Guanwen%20Yao_Lifeng_Cai.pdf
[2]:
http://infolab.stanford.edu/~ullman/mmds/ch9.pdf
[3]:
http://spark.apache.org/documentation.html
[4]:
https://docs.python.org/3/
[5]:
https://datajobs.com/data-science-repo/Recommender-Systems-%5BNetflix%5D.pdf
[6]:
Professor Srinivas Akella, ITCS-6190/8190: Cloud Computing for Data Analysis Spring 2017 Class - lecture notes: UNC Charlotte.
[7]:
http://jmcauley.ucsd.edu/data/amazon/links.html