Book Recommender System

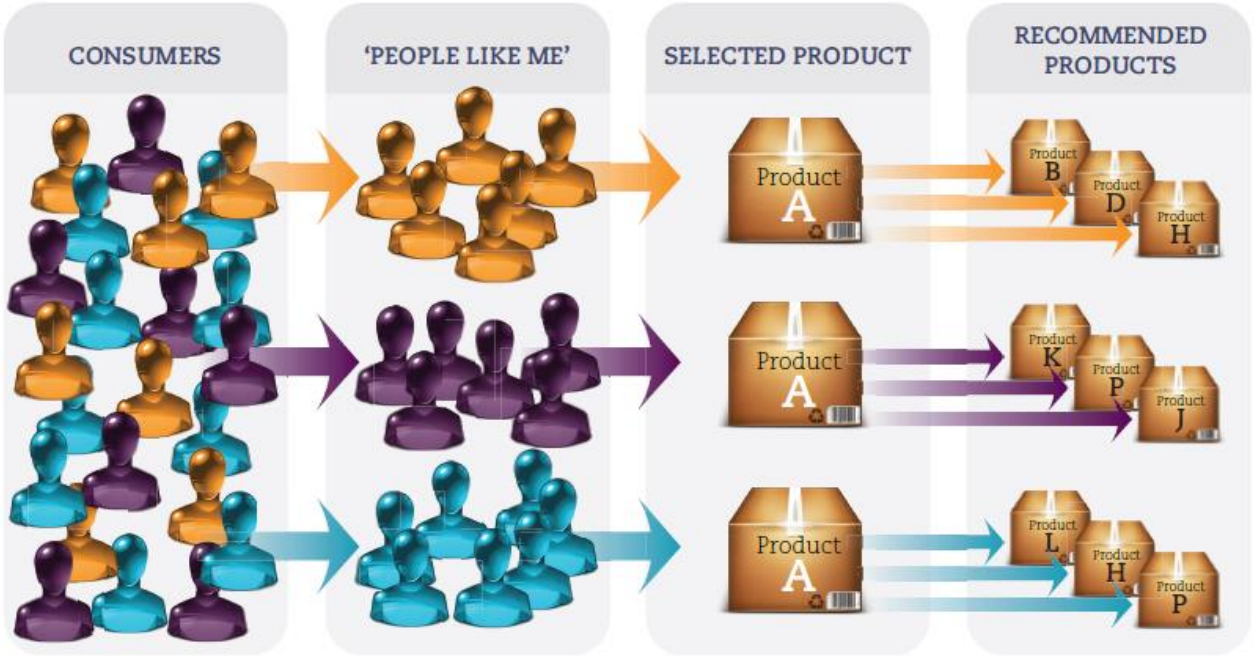
Introduction:

Both the online entertainment and e-commerce companies are trying to retain their customers by taking their access to the website to more personalized manner. So, provide additional recommendations based on users past activity. Our project would be one of such system that recommends additional books that belongs to similar genre, author or publisher. Such systems result in increase in rate of purchase, these may also include unplanned purchases driven by surprise factor from the recommendations made.

Collaborative Filtering Systems:

It is a common approach that provide recommendations of items based on patterns of sales of a product of other brands along which the one the user has searched for. This technique does not rely much on the information about items or users but make recommendations based on the user ratings. The proposed system collects the ratings from the user to predict the interest and analyses the item to find the features using collaborative filtering method. A correlation function is used to calculate the similarity between the movies in the dataset. Once the user selects a book, the books that have a cosine similarity closer to the selected book are recommended. In this process, we don't need to have the knowledge about the item specifications. The approaches taken are Item-Item based filtering, User-User based filtering and Alternating Least Squares Algorithm.

The Item-Item based filtering identifies the similarity between the items and uses this information to recommend books to the users based the previous ratings which the user has provided. The User-User based filtering identifies the similarity between the users and uses this information to recommend books to the users.



Dataset:

http://www2.informatik.uni-freiburg.de/~cziegler/BX/
Dataset consists three files.
BX-Users.csv: Contains the users. User IDs (`User-ID`) and Demographic data (`Location`, `Age`)
BX-Books.csv: Contains Books data (`Book-Title`, `Book-Author`, `Year-Of-Publication`, `Publisher`)
BX-Book-Ratings.csv: Contains the book rating information. Ratings (`Book-Rating`) are either explicit, expressed on a scale from 1-10 (higher values denoting higher appreciation)

Working and Implementation:

In many applications, Recommender systems usually provide the user with a list of recommendations that they might prefer, or supply predictions on how much the user might prefer each item.

There are two main phases in the process of Data Mining namely learning phase and use phase. In **learning phase**, the data mining system analyses the data and builds a model of consumer behaviour. This method is time-consuming. Automatic recommender systems are specialized to recommend products in commerce applications. Some recommenders have an offline phase during which they learn a model of customer behaviour.

In **use phase**, the model can be rapidly applied to consumer situations. In this phase the model is applied to the real-time data.

Functional evaluation:

Definitely: Our system will be capable enough of recommending books based on user ratings with the help of collaborative filtering technique in pySpark.

Likely: Implement a non-similarity method like Alternating Least Square Method (ALS).

Ideally: We will try to make recommendations mostly location specific using the demographic data available. Like - which books are highly rated in your location.

Execution Process:

Collaborative Method:

- Clean and pre-process the data into a required format.
- Split the data into 2 parts, 80% of data is used for training and remaining 20% is used for testing.
- Calculate cosine similarity for each book in the dataset w.r.t user rating.
- Generate prediction for missing items by Computing item-item similarity using weighted sum
- Fetching top-K of the generated ratings and recommend to user.
- Filter the data and sort relevant books based on ratings provided by the user.
- Return the list of recommended books based on the highest cosine similarity.
- Compare the recommendations made from the training data and compare it with the test data.

Collaborative based movie recommendation system consists of six steps:

Let the Users be U1, U2, U3, U4

Books be B1, B2, B3, B4

The Users have rated the books which they read, let the data be
U1 – B1 – 6
U1 – B2 – 4
U1 – B3 – 7
U2 – B2 – 2
U2 – B4 – 9
U3 – B1 – 5
U3 – B3 – 8

U3 – B4 – 2
U4 – B1 – 2
U4 – B4 – 8

Step 1: User-Book Rating

In in first step the ratings.csv file is taken as input to create a format such that it consists of tuples of (User, (Book, Rating)) and Map the result based on key – "User".

U1, [(B1,6), (B2,4), (B3,7)]
U2, [(B2, 2), (B4, 9)]
U3, [(B1, 5), (B3, 8), (B4, 2)]
U4, [(B1, 2), (B4, 8)]

Step 2: Book-Book Pairs

For each user find the combinations of books and generate tuples with Key – (Book A, Book B); Value – (Rating of A, Rating of B)

[((B1, B2), (6, 4)), ((B2, B3), (4, 7)), ((B1, B3), (6, 7))]
[(B2, B4), (2, 9)]
[((B1, B3), (5, 8)), ((B3, B4), (8, 2)), ((B1, B4), (5, 2))]
[(B1, B4), (2, 8)]

Step 3: Combining Rating and Similarities

Map the data with key as the tuple of books

[(B1, B2), [(6, 4)]]
[(B1, B3), [(6, 7), (5, 8)]]
[(B1, B4), [(5, 2), (2, 8)]]
[(B2, B3), [(4, 7)]]
[(B2, B4), [(2, 9)]]
[(B3, B4), [(8, 2)]]

Step 4: Cosine Similarity

Find the Cosine Similarity for each Book Pair with the help of available list. And return the tuple with key as book pair and value as (CosineSimilarity, number of pairs)

[((B1, B2), (1, 1))]
[((B1, B3), (1.04, 2))]
[((B1, B4), (0.58, 2))]
[((B2, B3), (1, 1))]
[((B2, B4), (1, 1))]
[((B3, B4), (1, 1))]

Step 5: Split the Book Pair tuples so that it derives relation between each Other

[(B1, (B2, (1, 1)))]
[(B2, (B1, (1, 1)))]
[(B1, (B3, (1.04, 2)))]
[(B3, (B1, (1.04, 2)))]
[(B1, (B4, (0.58, 2)))]
[(B4, (B1, (0.58, 2)))]
[(B2, (B3, (1, 1)))]
[(B3, (B2, (1, 1)))]
[(B2, (B4, (1, 1)))]

[(B4, (B2, (1, 1)))]
[(B3, (B4, (1, 1)))]
[(B4, (B3, (1, 1)))]

Step 6: Generate a dictionary and group all these tuples by key value

[B1, [(B2, (1, 1)), (B3, (1.04, 2)), (B4, (0.58, 2))]]
[B2, [(B1, (1, 1)), (B3, (1, 1)), (B4, (1, 1))]]
[B3, [(B1, (1.04, 2)), (B2, (1, 1)), (B4, (1, 1))]]
[B4, [(B1, (0.58, 2)), (B2, (1, 1)), (B3, (1, 1))]]

Step 7: Generate user specific recommendations.

Till now we have generated relativity between every book to each other book. This is not user specific. Now we will generate recommendations for each user based on the users rating for other books, using the training data and the generated dictionary.
For each user from training data, loop over each book rating tuple.
You have the data of relativity of all other books in the dictionary. Create two dictionaries one to sum up the cosine similarities and the other to sum up the similarities multiplied by the user rating.

For User U4:

Loop over training data -> [(B1, 2), (B4, 8)]
    Loop over dictionary -> [(B2, (1, 1)), (B3, (1.04, 2)), (B4, (0.58, 2))]
        dictWithRating[B2] += 1 * 2 -> (cosine value between B1,B2 * Rating by U4 for B1)
        dictWithoutRating[B2] += 1
        dictWithRating[B3] += 1.04 * 2
        dictWithoutRating[B3] += 1.04
        dictWithRating[B4] += 0.58 * 2
        dictWithoutRating[B4] += 0.58
            Loop over dictionary -> [(B1, (0.58, 2)), (B2, (1, 1)), (B3, (1, 1))]
        dictWithRating[B1] += 0.58 * 8 -> (cosine value between B4,B1 * Rating by U4 for B4)
        dictWithoutRating[B1] += 0.58
        dictWithRating[B2] += 1 * 8
        dictWithoutRating[B2] += 1
        dictWithRating[B3] += 1 * 8
        dictWithoutRating[B3] += 1

At the end of this nested loop the values of the dictionary which the user is yet read are:
dictWithRating[B2] = 2 + 8 = 10
dictWithoutRating[B2] = 1 + 1 = 2
dictWithRating[B3] = 2.08 + 8 = 10.08
dictWithoutRating[B3] = 1.04 + 1 = 2.04

Find the recommendation score for each book from the dictionary and return tuple of (book, recommendation score)
Recommendation score for B2 = dictWithRating[B2] / dictWithoutRating[B2] = 5
Recommendation score for B3 = dictWithRating[B3] / dictWithoutRating[B3] = 4.94
Return -> (U4, [(5, B2), (4.94, B3)])

So, from the left-over books B2 is highly recommended. If asked to make multiple recommendations return the required number of books from the list.

Pros & Cons:

Pros:

Collaborative Approach does not require much user profile like content-based recommendation approach.

Cons:

This system suffers cold-start problem where the user/community is very new to the system/environment when there are no enough information and rating for the movies to process and recommend item.

Alternating Least Square Method (ALS):

Alternating Least Squares is a Matrix Factorization approach to implement a recommendation algorithm by decomposing a large user/item matrix into lower dimensional user factors and item factors. We can then estimate the user rating by multiplying those factors.

$$R = PQ^T$$

Where R is a Ratings Matrix, P and Q are features matrices of users and items
Alternating Least Squares can solve for p_u and q_i in alternate iterations each time by assuming the other value is known.
Now, we can estimate the unknow rating values by

$$r_{ui} = p_u q^T_i$$

Steps Involved:

- Data Preparation
- Modeling
- Generating Recommendations
- Evaluation of Results

Challenges:

1. Integer type of User IDs and Book IDs required to use the Rating() function in "org.apache.spark.mllib.recommendation.ALS". So, we have converted the data from Strings to Integers using "hashlib" library.
2. Error faced due to "Container killed by YARN for exceeding memory limits". Increased the Driver and Executor memories to 16GB
3. Error faced due to "GC overhead limit exceed". Executed in AWS to fix this issue.

Technologies Used:

Apache Spark, Python

Task Distribution:

Aditya Viswanadha - Clean and pre-process the data, ALS Method, Documentation
Harish Pendyala - Clean and pre-process the data, ALS Method, Documentation
Navaneeth Reddy Matta - Collaborative Recommendation, Documentation, Webpages

Group Members:

Aditya Viswanadha - 800959537
Harish Pendyala - 800956847
Navaneeth Reddy Matta – 800935534

Relevant Documents:

Proposal – http://webpages.uncc.edu/nmatta1/cloudproject/proposal.html

Report – http://webpages.uncc.edu/nmatta1/cloudproject/report.html

Collaborative Output (Item - Item) - http://webpages.uncc.edu/nmatta1/cloudproject/collaborativeII.out

Collaborative Output (User - User) http://webpages.uncc.edu/nmatta1/cloudproject/collaborativeUU.out

ALS Method Output File – http://webpages.uncc.edu/nmatta1/cloudproject/als.out

References:

1. https://stanford.edu/~rezab/classes/cme323/S15/notes/lec14.pdf
2. https://bugra.github.io/work/notes/2014-04-19/alternating-least-squares-method-for-collaborative-filtering/
3. https://datasciencemadesimpler.wordpress.com/tag/alternating-least-squares/