International Journal on Future Revolution in Computer Science & Communication Engineering
Volume: 4 Issue: 4

ISSN: 2454-4248
260 – 263

Product Recommendation using Hadoop

Prof. Deepali Patil
Department of Information Technology
Shree L. R. Tiwari College of Engineering
Mumbai, India
e-mail: deep.patil1987@gmail.com

Ujwal Ahir
Department of Information Technology
Shree L. R. Tiwari College of Engineering
Mumbai, India
e-mail: ujwal.ahir@gmail.com

Dhruv Bindoria
Department of Information Technology
Shree L. R. Tiwari College of Engineering
Mumbai, India
e-mail: dhruv.bindoria@gmail.com

Shankarlal Bhati
Department of Information Technology
Shree L. R. Tiwari College of Engineering
Mumbai, India
e-mail: bhatishankar96@gmail.com

Raj Mehta
Department of Information Technology
Shree L. R. Tiwari College of Engineering
Mumbai, India
e-mail: rraajjmehta@gmail.com

***Abstract*—** Recommendation systems are used widely to provide personalized recommendations to users. Such systems are used by e-commerce and social networking websites to increase their business and user engagement. Day-to-day growth of customers and products pose a challenge for generating high quality recommendations. Moreover, they are even needed to perform many recommendations per second, for millions of customers and products. In such scenarios, implementing a recommendation algorithm sequentially has large performance issues. To address such issues, we propose a parallel algorithm to generate recommendations by using Hadoop map-reduce framework. In this implementation, we will focus on item-based collaborative filtering technique based on user's browsing history, which is a well-known technique to generate recommendations.

Keywords- *Hadoop, HDFS, Content based filtering, Collaborative filtering, Map-Reduce, E-commerce, Similarity Calculation.*

*****

I. INTRODUCTION

A. Recommendation System :

Recommendation systems [3] are part of information filtering system which aims at predicting the preference the user would give to an item. Recommendation systems have changed the way websites interact with their users. They eliminate the static experience, in which users search for static information for potentially buying products, by increasing interaction among users to provide rich user experience dynamically. Recommendation systems derive recommendations for each individual user based on their past purchases, searches and on other users' purchase and search behaviours. Recommendation systems personalize the experience of each user by randomizing content that is particularly relevant to their experienced interests, instead of providing a static experience to every user. Recommendation systems can be extremely effective on large scale if they are implemented correctly. Many of the world's top used websites, such as Facebook, Twitter, LinkedIn, Amazon etc. use recommendation systems to engage their users with relevant content.

B. Collaborative Filtering:

Collaborative filtering is the most successful recommendation system technique till date. It is used in many of the most successful recommendation systems on the web. Collaborative

Filtering systems recommend products to a target user based on the behaviours of other users. Collaborative filtering systems uses statistical techniques to and a set of users known as

Neighbours, that have a similar experiences as that of the target user i.e. either they have rated different products similarly or they tend to buy similar set of products or either they have watch same set of movies or listened to similar kind of music. Once a neighbourhood of users is formed, the system uses several algorithms to derive recommendations. In order to understand the algorithm and the recommendation process, its good to introduce basic terms and then get familiar with approaches, methods, tasks, challenges and evaluation of recommendation systems. Items and users are the two important entities engaged in every recommendation system. An item refers to any product such as a music song, a movie, a product, an article that recommendation system is to recommend. User is a person ready to accept recommendations when providing opinions about various items. The goal of collaborative filtering algorithms is to either make suggestions about new items or to make prediction about the acceptance of a certain item for recommendations when providing opinions about various items. In addition, it even aims to either make suggestions of new items or to make prediction about the acceptance of a certain item for a particular user based on users past experiences and similarity with others users. Prediction is a numeric value expressing the affinity of an item for the active user. Recommendation is a list of items that the active user will like the most. This is also known as top N recommendations where the list contains top N liked items [3].

C. Hadoop Map-Reduce Framework:

 ISSN: 2454-4248
260 – 263

Map-Reduce [3] is a programming framework designed for processing large amounts of data, in parallel, by dividing a complete task into a set of independent tasks where each independent task performs the similar computation. The data in the map-reduce framework is not shared across the nodes. Instead, the data elements in the map-reduce are immutable i.e. the data once written cannot be written twice and it can only be read many times. All the data used as an input and the resultant data post processing is stored in HDFS (Hadoop

Distributed File System). The data read from the input les, stored in HDFS are processed and converted into intermediate results and are further processed to generate final results.

II. LITERATURE SURVEY

The paper [1] proposes an algorithm for generating recommendations based on typicality.

To model this behavior, following traditional collaborative filtering assumptions, we define the following two assumptions based on user taste:
1. If people with similar taste to ua like ur, ua will like ur;
2. If people with similar taste to ur like ua, ur will like ua

Since both assumptions lead to the same predicted selections, therefore, ur should be recommended to ua when ur likes people with similar attractiveness to ua and ua likes people with similar attractiveness to ur, or equivalently, when people with similar taste to ur like ua and people with similar taste to ua like ur.

More formally, for a predicted successful interaction between ua and ur: denoted ua _! ur, there are two conditions to be fulfilled: 5. The attractiveness of the recommended user should match the taste of the active user, which will facilitate initiation of the interaction from the active user to the recommended user

The paper [2] proposes an algorithm based on three phases:

1) Data Partitioning Phase
2) Map Phase
3) Reduce Phase

1. Data Partitioning Phase: Here, it separates the UserID into different files, in these files each row stores a UserID. These files are as the I/P to the map phase.

2. Map Phase: The Hadoop platform, initialize a new mapper if the Datanode has enough response to initialize a mapper. The mapper's setup builds the rating matrix between user and item which are already filtered by local filter. The mapper reads the UserID file by line no. Take the line no. as the i/p key and contents of the line as the values. The local filter of Bloom filter randomly selects 50% users by the random function. In the next step, it computes the similarity between this user and other users. Finally, it identifies the user's nearest neighbor (by similarity values) and accordingly with equation 2 to calculate his predict rating on items. The Global filter of the Bloom filter works for the accuracy. It compares the two rating matrices and use e.g. threshold value to select the users from them. The algorithm sort the predict rating and store them in recommendation list. The UserID and its corresponding recommendation list as the intermediate key/value, output them to the reduce phase.

3. Reduce Phase: The Hadoop platform would generate some reducers implicitly. The reducers collect the UserID and its corresponding recommendation list, sort them to UserID and then o/p them to the HDFS.

III. PROPOSED SYSTEM

Following figure clearly shows how the system is being accessed. First user creates an account in the website and browses some products and purchase as per his/her needs. Similarly more than one users browses the products and its actions are stored in the database. Now, the database records are converted into required csv/txt format and transferred to the Hadoop framework.

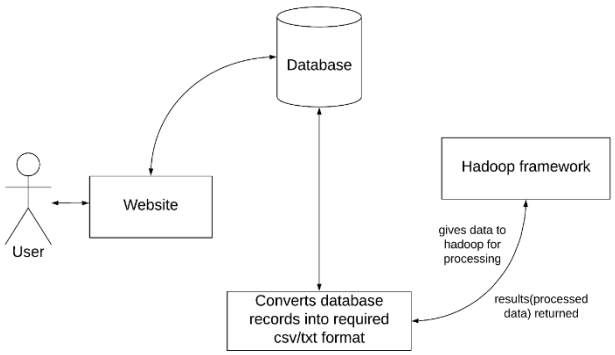

Fig.1 System Architecture

Hadoop framework applies collaborative filtering algorithm to the datasets and send the result back to the website database after pre-processing. Now from the updated database, recommended products for each product is shown on the website.

IV. SIMILARITY CALCULATION

First the datasets from website is preprocessed and converted into required csv/txt format for calculation. Each browsing action of user is converted in the form of "userid, productid, 1/0" where, rating 1 represents user have browsed and purchased the product and rating 0 represents user have only browsed the product and have now purchased it. Now, these data is given to Hadoop framework which performs several map reduce jobs and generates similarity.

Following figure represents how hadoop framework is used for computing similarity among itesms:

 ISSN: 2454-4248
260 – 263

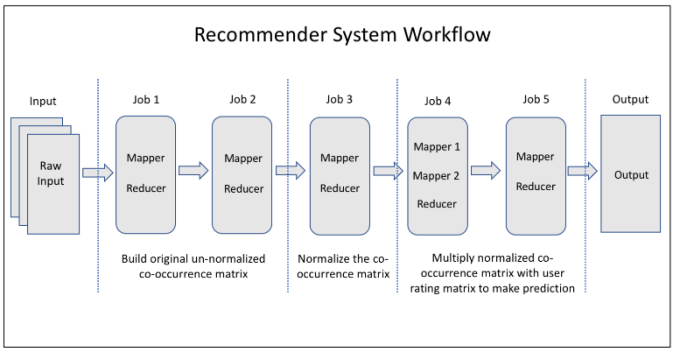

Fig.2 Hadoop map reduce job workflow

Job1: This job divides ratings from input file according to each user.
- Mapper: read raw input information
  - input: userID, productID, rating
  - output: < key=userID, value=productID: rating >
- Reducer: merge the output from Mapper according to unique userID
  - input: < key=userID, value=< product1: rating1, product2: rating2, ... > >
  - output: < key=userID, value="product1: rating1, product2: rating2, ..." >

Job2: This job is used to Build the original co-occurrence matrix.
- Mapper: read output from MapReduce job 1
  - input: userID \t "product1: rating1, product2: rating2, ..."
  - output: < key="product_A: product_B", value=1 >
- Reducer: merge the output from Mapper according unique productA: productB
  - input: < key="product_A: product_B", value=1, 1, 1,...>
  - output: < key="product_A: product_B", value=count >

Job3: This job is used to normalize the co-occurrence matrix.
- Mapper: read output from MapReduce job 2 and split
  - input: product_A: product_B \t count
  - output: < key=product_A, value="product_B=count" >
- Reducer: calculate the normalized co-occurrence matrix value
  - input: < key=product_A, value=< product_B=count, product_C=count, ... > >
  -output: < key=product_B, value="productA=count/total" >

Job4: This job is used to calculate the average purchase made by each user.
- Mapper: read the original user rating information
  - input: userID, productID, rating
  - output: < key=userID, value=rating >
- Reducer: merge the output from Mapper according to unique userID
  - input: < key=userID, value=< rating1, rating2, ... > >
  - output: < key=userID, value=< average rating > >

Job5: This job is used to replace the value of products which are not browsed by the user with the average purchase value.
- Mapper 1: read co-occurrence matrix from MapReduce Job 3
  - input: product_B \t product_A=ratio
  - output: < key=product_B, value="product_A=ratio" >
- Mapper 2: read the original user rating information to build the rating matrix
  - input: userID, productID, rating
  - output: < key=product_B, value="userID: rating" >
- Reducer:
  - input: < key=product_B, value="product_A=ratio1, product_C=ratio2, ..., user1: rating1, user2: rating2, ..." >
  - output: < key="userID: productID", value=ratio* rating>
  - setup: read the user average rating

Job6: This job is used to multiply the co-occurrence matrix with the rating matrix to generate similarities.
- Mapper: read the output from MapReduce Job 5
  - input: userID: productID \t ratio * rating
  - output: < key="userID: productID", value=ratio*rating>
- Reducer:
  - input: < key="userID: productID", value=< subrating1, subrating2, ... > >
  - output:< key="userID: productID", value=rating prediction >

V. EXAMPLE

Say we have browsing history of 5 users and 7 items as follows:

Item/User	U1	U2	U3	U4	U5
I1	1	1	0	1	1
I2	1	1	-	-	0
I3	1	1	-	0	0
I4	-	1	1	1	1
I5	-	-	0	-	0
I6	-	-	-	1	1
I7	-	-	0	-	-

Fig.3 Sample Input

In above table, rating 1 represents user has browsed and purchased that item and rating 0 represents user has only browsed that item. User id is from 1 to 5 and item id is from 10001 to 10007.

Now, after the map reduce job 1 calculation, we get the browsing as per each user i.e. from above input, we get

1 10001:1,10002:1,10003:1
2 10001:1,10002:1,10003:1,10004:1
3 10007:0,10005:0,10004:1,10001:0
4 10006:1,10003:0,10001:1,10004:1
5 10004:1,10005:0,10006:1,10001:1,10002:0,10003:0

Now, after the map reduce job 2 calculation, we get co-occurrence matrix as follows:

Item/Item	I1	I2	I3	I4	I5	I6	I7
I1	5	3	4	4	2	2	1
I2	3	3	3	2	1	1	0
I3	4	3	4	3	1	2	0
I4	4	2	3	4	2	2	1
I5	2	1	1	2	2	1	1
I6	2	1	2	2	1	2	0
I7	1	0	0	1	1	0	1

Fig.4 Co-occurrence Matrix

Above values represents the number of users that have browsed both the items. For example, I1 and I3 value is 4

 ISSN: 2454-4248
260 – 263

which means there are 4 users (U1, U2, U4 and U5 from fig. 3) which have browsed both I1 and I3.

Now, after the map reduce job 3 calculation, we get normalized co-occurrence matrix as follows:

Item/Item	I1	I2	I3	I4	I5	I6	I7
I1	0.23	0.14	0.19	0.19	0.095	0.095	0.047
I2	0.23	0.23	0.23	0.15	0.077	0.077	0
I3	0.23	0.18	0.23	0.18	0.059	0.12	0
I4	0.22	0.11	0.17	0.22	0.11	0.11	0.055
I5	0.20	0.10	0.10	0.20	0.20	0.10	0.10
I6	0.20	0.10	0.20	0.20	0.10	0.20	0
I7	0.25	0	0	0.25	0.25	0	0.25

Fig. 5 Normalized Co-occurrence Matrix

Matrix is normalized by dividing each element with the sum of values from that row. For Example, in fig. 5 I1 and I2 value is generated by dividing its value which is 3 (from fig. 4) with the sum of that row i.e. 21 (sum of 1st row in fig.4), so we get 3/21 = 0.14

Now, after the map reduce job 4 calculation, we get the average purchase of each user. In our case we get:

1 1.0
2 1.0
3 0.25
4 0.75
5 0.5

For Example, U5 has value 0.5 since he/she has browsed 6 items out of which only 3 of them is purchased. Therefore, average purchase = 3/6 = 0.5

Now, job 5 and 6 generates final similarities by multiplying the normalized co-occurrence (fig.5) matrix with input matrix (fig.3). Note that first the empty values in input matrix is replaced by corresponding user's average purchase before multiplication is performed. We get final similarity matrix as follows:

Item/User	U1	U2	U3	U4	U5
I1	1	1	0.297	0.738	0.547
I2	1	1	0.288	0.692	0.461
I3	1	1	0.308	0.705	0.529
I4	1	1	0.319	0.763	0.583
I5	1	1	0.275	0.800	0.555
I6	1	1	0.325	0.750	0.601
I7	1	1	0.250	0.875	0.625

Fig. 6 Final User based similarity matrix

Now, for providing recommended items for each user we refer to final similarity matrix and select the items with maximum values. For example, in our case the recommended items for user 4 will be 10007 and 10005 since these have larger similarity values i.e. 0.875 and 0.800 as compared to other items.

We can also find set of items similar to a particular item from co-occurrence matrix (fig. 4). For Example, in our case to find items similar to I1, we check I1 column from co-occurrence matrix and find items with larger similarity value as compared to other items, which are I3 and 14 with similarity value = 4

One drawback of using this approach is that the similarity values are not efficient if user have purchased all the browsed items. For Example in our case for U1 and U2, the similarity value for each item is 1 since the user have purchased all the browsed items and system cannot effectively decide which items to recommend.

VI. CONCLUSION

We discussed about recommendation system algorithms and map-reduce programming model by Hadoop. A parallel algorithm to compute item-based similarity using Hadoop map-reduce framework is explained. The reason behind parallelization of the computation is inspired by word count example using map-reduce. The computation is split into multiple small tasks and each of the small task is handled independently by a map-reduce job.

We showed details of the approach including the inputs and outputs produced at each phase of execution. Different experiments are conducted to show the performance improvement by using a multi-node cluster. Test data for experiments is taken from movie-lens databases. The results show that the performance of the algorithm improves as the number of nodes within a cluster increases with constant dataset size. The Hadoop map-reduce approach saves a lot of resources in computing similarities and generates recommendations in short time.

ACKNOWLEDGMENT

Authors would like to thanks to her guide Prof. Deepali Patil, Head of I.T department, SLRTCE, Thane for their valuable support and help.

REFERENCES

[1] Reshma U. Shinde "Typicality-Based Collaborative Filtering Recommendation System", IJIRCCE, 2016
[2] Nilay Narlawar "A speedy approach: User-based Collaborative Filtering with Mapreduce", IJIRCCE, 2014
[3] Chetan Prakash Somani "Item-Based Recommendation Algorithm Using Hadoop", Auburn University, 2015
[4] Jing Jiang, Jie Lu, Guangquan Zhang, Guodong Long, "Scaling-up Item-based Collaborative Filtering Recommendation Algorithm based on Hadoop", 2011 IEEE World Congress on Services
[5] Jai Prakash Verma, Bankim Patel, Atul Patel, "Big Data Analysis: Recommendation System with Hadoop Framework", 2015 IEEE International Conference on Computational Intelligence & Communication Technology
[6] Sebastian Schelter, Christoph Boden, Volker Markl, "Scalable Similarity-Based Neighborhood Methods with MapReduce", Technische Universität Berlin, Germany
[7] https://grouplens.org/datasets/movielens/
[8] https://hadoop.apache.org/