Evaluating and Enhancing Efficiency of Recommendation System using Big Data Analytics

Archit Verma[1], Dharmendra Kumar[2]

[1]*M.Tech Student, Computer Science and Engineering at United College of Engineering and Research, Allahabad, Uttar Pradesh, India*
[2] *Associate Professor, Computer Science Department at United College of Engineering and Research ,Allahabad, Uttar Pradesh, India*

---***---

Abstract - *Recommendation System helps purchaser in finding out the most favorable item he should purchase out of a large number of items, by predicting rating of items that are not yet rated by him. It helps user by showing list of items that the user may like to buy, based on user's past purchases. Large amount of information on the Internet, e-commerce sites like Amazon , ebay and social media like twitter, linked etc ,are the cause for making recommendation system, as it filters useful information from huge data sets. Recommender System is of two types, Content based that recommends items on basis of the features of item, and Collaborative Filtering based that recommends items based on the user's social environment. In this paper we will discuss collaborative filtering which is further classified 1.) User's based, that recommends items on basis of user similarity,2.)Item based, that recommends items on basis of item similarity, and 3.)Matrix factorization based (Alternating least squares (ALS)).Due to large amount of number of users and items, recommendation system experiences scalability problem. To handle this scalability problem we use Big Data Analytics. Big data refers to datasets that are not only big with high volume, but also high in variety and velocity, known as (3V) of big data, which makes them difficult to handle using traditional tools and techniques. Hadoop is one of the answers for this problem of Big Data. Apache Mahout is a machine learning tool that provides scalable machine learning algorithm for collaborative filtering on Hadoop environment, it also provides non-hadoop implementation. Mahout provides non-hadoop implementation of user based and item based collaborative filtering and hadoop implementation of item based collaborative filtering. Apache Spark is a fast and general purpose cluster computing technology, designed for fast computation. MLlib is Apache Spark's scalable machine learning library. MLlib provides implementation of Matrix factorization based on ALS algorithm for collaborative filtering. HBase is a distributed column-oriented NoSQL database built on top of the Hadoop file system,we are using HBase in storing rating data inputted by user by web forms in JSP(Java Server Pages). Apache Phoenix is the fastest way to access HBase data,it provides SQL interface to this NoSQL database. For analysis purpose we will use movieLens dataset. We will evaluate recommendation algorithms of Mahout on different similarity measures along with ALS of spark MLlib based on MAE, RMSE, Precision and Time taken, finally we use the best algorithms based on our analysis in terms of accuracy and time taken in implementing recommendation systems.*

Key Words: Recommendation System, Collaborative Filtering, User Based ,Item Based, Matrix Factorization, Alternating Least Squares, BigData, MAE, RMSE, Precision,Recall,F-Score, Hadoop, Mahout , Spark , MLlib, HBase, Apache Phoenix ,Pig, and Crontab.

1. INTRODUCTION

Recommender systems is an information filtering system that predicts the 'rating' that user would give to an item or social element they had not yet considered or rated, using a model built from the characteristics of an item known as content-based approaches or the user's social environment known is collaborative filtering approaches.
Recommendation systems are used by e-commerce (like Amazon, eBay), social media (like Facebook, Twitter, LinkedIn) and (Pendora Radio). Amazon utilizes item-based collaborative filtering approach in recommendation. These systems use collaborative filtering for predicting rating of items that are not purchased by a particular user from large amounts of data and provides narrow suggestions to that user. LinkedIn makes substantial use of item-based collaborative filtering. As an example, each member's profile on LinkedIn has a "People Who Viewed This Profile Also Viewed" recommendation module.
Due to this large data sets recommendation system experiences scalability problem so there is a big need of using Big Data analytics.
Big data is a term that refers to data sets or combinations of data sets whose volume, variability and velocity make them difficult to be captured, managed, processed or analyzed by conventional technologies and tools, such as relational databases and desktop statistics or visualization packages, within the time necessary to make them useful.[3]
V's of Big Data[2],[3]
Volume: Volume refers to the vast amount of data generated every second. We are not talking terabytes, but zettabytes or brontobytes of data.
Velocity: The term velocity refers to the speed of generation of data or how fast the data is generated and processed.
Variety: Variety refers to the different types of data we can now use, structured (SQL), semi-structured (CSV, XML, JSON,

NoSQL), and unstructured (videos, photos, audio files, webpages)
Veracity: Veracity refers to the messiness or trustworthiness of the data.
Value: Value refers to our ability turn our data into value.
Variability: Variability refers to data whose meaning is constantly changing.
Big data analytics is the process of examining large data sets in order to uncover hidden patterns, unknown correlations, market trends, customer preferences and other useful business information. [11]
In this present paper we will focus on Recommendation system using collaborative filtering algorithms with tools such as Apache mahout and Apache Spark on Apache Hadoop file system. We will also design an efficient recommendation system after evaluating them based on MAE, RMSE, Precision, and Time.

2. CONTENT BASED RECOMMENDATION SYSTEM

Content based recommendation System recommends items to users' based on features of items that the user had purchased in the past. Systems implementing content-based recommendation approach analyze a set of documents and/or descriptions of items previously considered by a user, and build a model or profile of user interests based on the features of the objects considered by that user. The profile is representation of user interests in a structured manner, adopted to recommend new interesting Items. The recommendation process basically consists in matching up the attributes of the user profile against the attributes of a content object.

3. COLLABORATIVE FILTERING BASED RECOMMENDATION SYSTEMS

Collaborative filtering algorithms are generally based on calculating similarities or dissimilarities between user profiles. It finds nearest neighbor based on similarity measures. The similarity calculation is based on the rating that users gave to different items. The users who have more or less the same behavior in rating item are nearest neighbors, this behavior is measured by similarity formula. So, collaborative filtering predictions and recommendations are based on the ratings or behavior of other users in the system. The fundamental assumption behind this method is that other user's likes in form rating can be selected and aggregated in such a way as to find a reasonable prediction value of the item the user under consideration has not seen. Intuitively, If a group of users likes the same things as Mr.X, then Mr.X is likely to like the things that group like which he hasn't yet seen. In other simple example we say if Mr.A likes items P, Q, R and S. and Mr.B likes items Q, R, S and T. then Mr. A should be recommended item T and Mr.B should be recommended item P.

4. SIMILARITY MEASURES USED BY IN USER BASED AND ITEM BASED RECOMMENDATION SYSTEM

Similarity measures are used to calculate similarity between users or items in collaborative filtering algorithms.
Pearson coefficient: The Pearson correlation is a ratio of the covariance of the two data sets to their standard deviations.[15]

$$\rho_{x,y} = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i-\bar{y})^2}} \qquad (1)$$

Spearman Correlation: The formula for Spearman's rank correlation coefficient involves the differences d_i between two sets of n rankings.

$$\rho = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n(n^2-1)}, \text{ where } d_i = x_i - y_i \qquad (2)$$

Euclidian Distance:
Euclidean distance measurement states that it is the square root of the sum of squared differences between corresponding elements of two vectors.

$$s=\frac{1}{1+d} \text{ where } d=\sqrt{(x_1-y_1)^2+(x_2-y_2)^2+\dots\dots+(x_n-y_n)^2} \qquad (3)$$

Cosine:
Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them.[17]

$$\cos(\theta) = \frac{\sum_{i=1}^{n}(x_i)(y_i)}{\sqrt{\sum_{i=1}^{n}(x_i)^2}\sqrt{\sum_{i=1}^{n}(y_i)^2}} \qquad (4)$$

Tanimoto:
The Tanimoto coefficient also called Jaccard coefficient measures similarity between finite sample sets, and is defined as the size of the intersection divided by the size of the union of the sample sets.[16]

$$J(A,B) = \frac{|A\cap B|}{|A\cup B|} \qquad (5)$$

LogLikelihood:
It is an expression of how unlikely are will users have so much overlap, given the total number of items and the number of items each user has a preference for. The resulting value can be interpreted as a probability that an overlap isn't just due to chance. [12],[36]

$$D=-2\ln\left(\frac{likelihood\,of\,null\,model}{likelihood\,of\,alternative\,model}\right) \qquad (6)$$

Note that 'null model' is model with few parameters and 'alternative model' is model with more parameters
City Block Distance:
The City Block Distance also called taxicab distance, d_1, between two vectors p, q in an n-dimensional real vector space with fixed Cartesian coordinate system, is the sum of the lengths of the projections of the line segment between the points onto the coordinate axes.[18]

$$s=\frac{1}{1+d_1} \text{ where } d_1 = \sum_{i=1}^{n}|p_i - q_i| \qquad (7)$$

Co-occurrence:
Instead of calculating similarity between every pair of items, co-occurrence computes the number of times each pair of items occurs together in some user's list of preferences, so as to fill out item-item similarity matrix. For instance, if there are 5 users who express some preference for both items I_1 and I_2, then I_1 and I_2 co-occur 5 times. Two items that never appear together in any user's preferences have a co-occurrence of 0.It is generally used in distributed item based recommendations.[13]

5. USER BASED COLLABORATIVE FILTERING ALGORITHMS

It is also known as nearest neighbor based collaborative filtering. It uses user-item database to predict rating of items that are not purchased by user. This algorithm makes use of similarity measures for finding the nearest neighbors of the user in question .The similarity computation makes use of rows of user-item matrix.
After nearest-neighbors have being found out, the algorithm uses similarity value and rating of item rated by neighbors to predict rating of an item that is not rated by user under consideration, according to the formula given below ,and hence output top N recommendation for a given user. The advantage of user-based approach is that it does not requires information about features of items .The problem with this approach is cold start problem in case of new user and new item ,sparsity problem in which the number of items sold on an e-commerce sites is extremely large. The most active users will only have rated a small subset of the overall database

Example of Pearson Similarity measure and rating prediction formula , w(u,v) represents similarity between user u and user v.

$$w(u,v)=\frac{\sum_{i\in I_u\cap I_v}(r_{u,i}-\bar{r}_u)(r_{v,i}-\bar{r}_v)}{\sqrt{\sum_{i\in I_u\cap I_v}(r_{u,i}-\bar{r}_u)^2}\sqrt{\sum_{i\in I_u\cap I_v}(r_{v,i}-\bar{r}_v)^2}} \quad (8)$$

$$\hat{r}_{u,i}=\bar{r}_u+\frac{\sum_{v\in N(u)} w(u,v)*(r_{v,i}-\bar{r}_v)}{\sum_{v\in N(u)} w(u,v)} \quad (9)$$

Item User	I1	I2	I3	I4	I5
U1	3.5	2	4	3	2
U2	2	1	2.5	5	
U3		2	1	2	1
U4	3	3	3.5	3	3.5
U5	2	5			1
U6		4	4.5		

Fig-1. Showing similarity between "u4" and "u6"

6. ITEM BASED COLLABORATIVE FILTERING ALGORITHM

Item-based collaborative filtering algorithm finds the similarity between items to make a prediction of rating. The item-based similarities are computed using columns of user-item matrix.

[6]Similarities between items are computed using columns of user-item matrix and put into an item-to-item similarity matrix say W where W[i,j] contains the similarity value between the items in row i and the item in column j.
The algorithms select items that are most similar to the already rated item. Now the unrated item's neighbors, are selected ,this is done either by threshold-based selection or the top- n selection.
The advantage of item-based approach is that it does not requires information about features of items ,it has better scalability as similarity has to be found out among items that are limited in number than users and, it reduces sparsity problem as well. The problem with this approach is cold start problem in case of new user and new item.
As an example the similarity measure formula and unknown rating prediction formula using item-based approach for Pearson is[14]:
Note that w(i,j) denotes similarity between item i and item j.

$$w(i,j)=\frac{\sum_{u\in U_i\cap U_j}(r_{u,i}-\bar{r}_i)(r_{u,j}-\bar{r}_j)}{\sqrt{\sum_{u\in U_i\cap U_j}(r_{u,i}-\bar{r}_i)^2}\sqrt{\sum_{u\in U_i\cap U_j}(r_{u,j}-\bar{r}_j)^2}} \quad (10)$$

$$\hat{r}_{u,i}=\frac{\sum_{j\in similar\ items\ in I_u} w(i,j)*r_{u,j}}{\sum_{j\in similar\ items\ in I_u}|w(i,j)|} \quad (11)$$

Item User	I1	I2	I3	I4	I5
U1	3.5	2	4	3	2
U2	2	1	2.5	5	
U3		2	1	2	1
U4	3	3	3.5	3	3.5
U5	2	5			1
U6		4	4.5		

Fig-2. Showing similarity between item I_3 and I_1

7. MATRIX FACTORIZATION BASED COLLABORATIVE FILTERING

Matrix Factorization[21]
Consider a rating matrix $R\in R^{m\times n}$ of m users and n items. Applying matrix factorization technique on R will end up factorizing R into two matrices $U\in R^{m\times k}$ and $P\in R^{n\times k}$ such that $R\approx U\times P^T$(their multiplication approximates R).This algorithm introduces a new quantity, k, that appears in both U`s and P`s dimensions. This is the rank of the factorization or number of latent factors. Formally, each $R_{i,j}$ is factorized into the dot product of $u_i.p_j$, having $u_i.p_j\in R^k$. This model assumes every rating in R is affected by k latent factors. Moreover, it represents both users and items in U and P accordingly, in terms of those k latent factors.

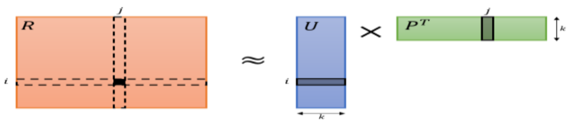


Fig-3 Diagram showing factors of user-item matrix factorization[21]

MF is a form of optimization process that aims to approximate the original matrix R with the two matrices U and P, such that it minimizes the following cost function:

$$J = ||R - U \times P^T||^2 + \lambda \left(||U||^2 + ||P||^2\right) \quad (12)$$

The first term in this cost function is the Mean Square Error (MSE) distance measure between the original rating matrix R and its approximation U×P^T. The second term is called a regularization term and is added to govern a generalized solution.

This cost function introduces two parameters: k and λ. Along with minimizing this cost function for a given k and λ, we have to determine the optimal value of these parameters as well

Alternating Least Squares:[21]

From MF's cost function, it appears that we aim at learning two types of variables those of U and those of P, and the two types are tied in the multiplication of U×P^T . Recall that the actual cost function is the sum

$$||R - U \times P^T||^2 = \sum_{i,j} \left(R_{i,j} - u_i \times p_j\right)^2 \quad (13)$$

plus regularization term. The fact that both U's and V's values are unknown variables makes this cost function non-convex. If we fix P and optimize for U alone, the problem is simply reduced to the problem of linear regression. In linear regression we solve by minimizing the squared error $||y - X\beta||_2$ given X and y. The solution is ultimately given by the Ordinary Least Squares(OLS) formula $\beta=(X^TX)^{-1}X^Ty$. (14)

Alternating least squares does just that. It is a two-step iterative optimization process. In every iteration it first fixes P and solves for U, and following that it fixes U and solves for P. Since OLS solution is unique and guarantees a minimal MSE, in each step the cost function can either decrease or stay unchanged, but never increase. Alternating between the two steps guarantees reduction of the cost function, until convergence. Since the actual cost function includes a regularization term, it is slightly longer. According to the two-step process, the cost function can be broken down into two cost functions:

$$\forall u_i : J(u_i) = ||R_i - u_i \times P^T||^2 + \lambda \cdot ||u_i||^2 \quad (15)$$

$$\forall p_j : J(p_j) = ||R_j - U \times p_j^T||^2 + \lambda \cdot ||p_j||^2 \quad (16)$$

And the matching solutions for u_i and p_j are:

$$u_i = \left(P^T \times P + \lambda I\right)^{-1} \times P^T \times R_i \quad (17)$$

$$p_j = \left(U^T \times U + \lambda I\right)^{-1} \times U^T \times R_j \quad (18)$$

And since each u_i doesn't depend on other $u_{j\neq i}$ vectors, each step can potentially be introduced to massive parallelization.

8. EVALUATION METRICES

In order to evaluate a recommendation system we split user's rating data sets into two data sets called training data(that is generally 80% of total) and evaluation data(that is generally 20% of total), by using this training data set, the recommendation system tries to compute a user's preference for an item. After the recommendation system calculates this preference, it uses actual user preference data from the evaluation data set in order to determine how accurate the computed preference is based on the following metrics.[4]

1.) Mean Absolute Error (MAE)

2.) Root Mean Square Error (RMSE)

MAE and RMSE are known as predictive accuracy or statistical accuracy metrics because they represent how accurately a recommendation system has predicted a user's preference for an item. Note that p_i is the predicted rating, and r_i is the actual ratings.[4]

$$\text{MAE} = \frac{\sum_{i=1}^{n} |p_i - r_i|}{n} \quad (19)$$

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n} (p_i - r_i)^2}{n}} \quad (20)$$

Information Retrieval (IR) Metrics for evaluation (Precision , Recall ,F-Score)

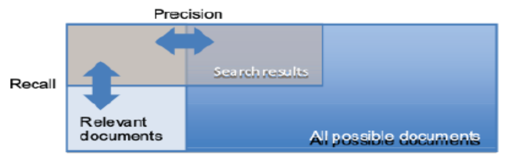


Fig-4. Precision and Recall[13]

1.)**Precision** : is a measure of exactness, determines the fraction of relevant items retrieved out of all items retrieved.[25]

$$\text{Precision} = \frac{tp}{(tp+fp)} \quad (21)$$

2.)**Recall**: is a measure of completeness, determines the fraction of relevant items retrieved out of all relevant items. [25]

$$\text{Recall} = \frac{tp}{(tp+fn)} \quad (22)$$

[4]t_p is the interesting item is recommended to the user, f_p is the uninteresting item is recommended to the user ,and f_n is the interesting item is not recommended to the user.

3.)**F-score** :In recommendations domain, it is considered an single value obtained combining both the precision and recall measures and indicates an overall utility of the recommendation list.

$$\text{F-Score} = 2\frac{(precision * recall)}{(precision + recall)} \quad (23)$$

It is a weighted average of the precision and recall, where the best F-score has its value at 1 and worst score at the value 0.[4]

Time: Time taken in recommendation process, smaller the time the more efficient our Recommendation will be.

9. APACHE HADOOP

[10]Hadoop is an open-source framework that allows storing and performing computations on big data in a distributed environment across clusters of computers using simple map-reducing programming model.

It is designed to scale up from single servers to thousands of machines, each offering local computation and storage.
Hadoop runs applications using the MapReduce algorithm, where the data is processed in parallel on different CPU nodes. Hadoop framework is capable enough to develop applications capable of running on clusters of computers and they could perform complete statistical analysis for a huge amounts of data.
Hadoop Architecture Hadoop framework includes following four modules:
1.)Hadoop Common: These are Java libraries and utilities required by other Hadoop modules. These libraries provides filesystem and OS level abstractions and contains the necessary Java files and scripts required to start Hadoop.
2.)Hadoop YARN: This is a framework for job scheduling and cluster resource management.
3.)Hadoop Distributed File System (HDFS™): A distributed file system that provides high-throughput access to application data.
4.)Hadoop MapReduce: This is YARN-based system for parallel processing of large data sets.

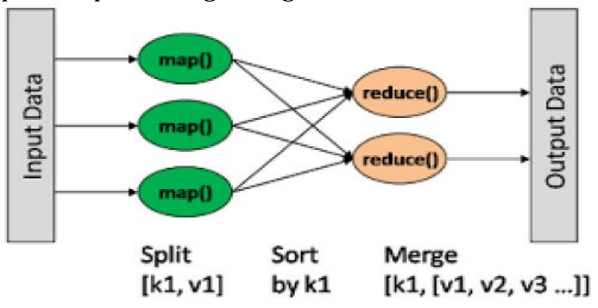


Fig-5. Map Reduce Operation in Hadoop[33]

Mapper Function: It divides the problem into smaller sub-problems. A master node distributes the work to worker nodes. The worker node just does one thing and returns the work back to the master node.
Reducer Function: Once the master gets the work from the worker nodes, the reduce function takes over and combines all the work. By combining the work some answer and ultimately output is produced.

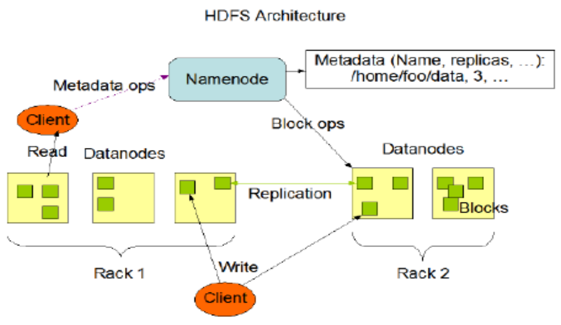


Fig-6. Hadoop File System Architecture[3]

HDFS is a file system based on the master slave architecture. It breaks the large files into default 64 MB blocks and store them into large cluster in a very efficient way.In this architecture there are three basic nodes in a Hadoop cluster.
1.)name node
2.)data node
3.)secondary name node.
Name Node: is the master node which controls all the data nodes and it contains Meta data. It manages all the file operations like read, write etc.
Data nodes :are the slave nodes present in Hadoop cluster. All the file operations performed on these nodes and data is actually stored on these nodes as decided by name nodes.
Secondary name node: is the back up of name node. As name node is the master node, it becomes very important to take its backup. If the name node fails, secondary name node will be used as name node.
Apache Hadoop can be operated in three modes:Local/Stand alone mode,pseudo distributed mode and fully distributed modes.

10. APACHE MAHOUT

[20]Apache Mahout is a project of the Apache Software Foundation to produce free implementations of distributed or otherwise scalable machine learning algorithms.
Many of the implementations use the Apache Hadoop platform. It has implementations of a wide range of machine learning and data mining algorithms:
Clustering, classification, collaborative filtering etc.
Recommendation algorithms analyzed by us in this paper are:
User-based Collaborative Filtering with different similarity measures, Item-based Collaborative Filtering with different similarity measures.
Input to Mahout is Raw data CSV file with format userid,itemid,rating
The output of mahout is preferences estimation.
Step 1:Mapping raw data into a DataModel Mahout-compliant
Step 2 :Tuning recommender components Similarity measure, neighborhood, etc.
Step 3:Computing rating estimations
Step 4:Evaluating recommendation
Mahout key abstractions are implemented through Java interfaces :
DataModel interface:Methods for mapping raw data to a Mahout-compliant form
UserSimilarity interface: Methods to calculate the degree of correlation between two users
ItemSimilarity interface:Methods to calculate the degree of correlation between two items
UserNeighborhood interface:Methods to define the concept of 'neighborhood'
Recommender interface:Methods to implement the recommendation step itself.
Recommender Evaluation
Metrics: RMSE, MAE,Precision,Recall,F1-Score
Implementations of RecommenderEvaluator interface,
AverageAbsoluteDifferenceRecommenderEvaluator,
RMSRecommenderEvaluator,
GenericRecommenderIRStatsEvaluator
Use RandomUtils.useTestSeed()to ensure

the consistency among different evaluation runs
Invoke the evaluate() method
Parameters
RecommenderBuilder: recommender instance
DataModelBuilder: specific criterion for training
Split Training-Test: double value (e.g. 0.8 for 80%)
Amount of data to use in the evaluation: double value (e.g 1.0 for 100%)

Running Mahout Recommendation System for Big Data in Hadoop File System[34]

```
mahout recommenditembased
-s  SIMILARITY_LOGLIKELIHOOD
-i  /path_to_csv_file_in_HDFS
-o  /path_to_desired_output_folder_in_HDFS
--numRecommendations 10
```

11. APACHE SPARK

[24]Apache Spark is a lightning-fast cluster computing technology, designed for fast computation. It is based on Hadoop MapReduce and it extends the MapReduce model to use it efficiently. The main feature of Spark is its in-memory cluster computing that increases the processing speed of an application.

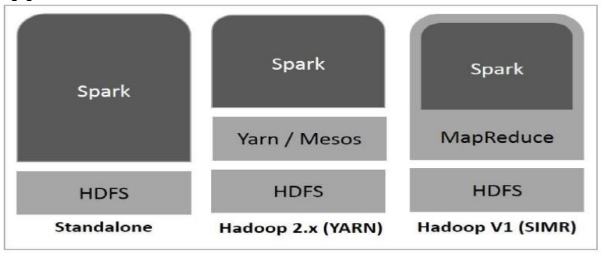


Fig-7.Spark[24]

There are three ways of Spark deployment
Standalone, Hadoop Yarn, Spark in MapReduce (SIMR)
MLlib(Machine Learning Library)is a distributed machine learning framework above Spark because of the distributed memory-based Spark architecture .It is, according to benchmarks, done by the MLlib developers against the Alternating Least Squares (ALS) implementations. Spark MLlib is nine times as fast as the Hadoop disk-based version of Apache Mahout.
Resilient Distributed Datasets Resilient Distributed Datasets (RDD) is a fundamental data structure of Spark.It is an immutable distributed collection of objects. Each dataset in RDD is divided into logical partitions, which may be computed on different nodes of the cluster. RDDs can contain any type of Python, Java , or Scala objects, including user-defined classes. An RDD is a read-only, partitioned collection of records. RDDs can be created in two ways; one is by referencing datasets in external storage systems and second is by applying transformations (e.g. map, filter, reducer, join) on existing RDDs.The Spark RDD API introduces few Transformations and few Actions to manipulate RDD.RDD transformations returns pointer to new RDD and allows you to create dependencies between RDDs. Each RDD in dependency chain (String of Dependencies) has a function for calculating its data and has a pointer (dependency) to its parent RDD. Spark is lazy, so nothing will be executed unless you call some transformation or action that will trigger job creation and execution. Therefore, RDD transformation is not a set of data but is a step in a program (might be the only step) telling Spark how to get data and what to do with it.
Transformations on RDD to be used by us in scala recommendation program
map(func): Returns a new distributed dataset, formed by passing each element of the source through a function func.
filter(func): Returns a new dataset formed by selecting those elements of the source on which func returns true.
flatMap(func): Similar to map, but each input item can be mapped to 0 or more output items (so func should return a Seq rather than a single item).
distinct([numTasks]): Returns a new dataset that contains the distinct elements of the source dataset.
join(otherDataset, [numTasks]): When called on datasets of type (K, V) and (K, W),returns a dataset of (K, (V, W)) pairs with all pairs of elements for each key. Outer joins are supported through leftOuterJoin, rightOuterJoin, and fullOuterJoin.
Actions on RDD to be used by us in scala recommendation program
reduce(func): Aggregate the elements of the dataset using a function func (which takes two arguments and returns one). The function should be commutative and
associative so that it can be computed correctly in parallel.
collect(): Returns all the elements of the dataset as an array at the driver program. This is usually useful after a filter or other operation that returns a sufficiently small subset of the data.
count(): Returns the number of elements in the dataset.
saveAsTextFile(path): Writes the elements of the dataset as a text file (or set of text files) in a given directory in the local filesystem, HDFS or any other Hadoop-supported file system. Spark calls toString on each element to convert it to a line of text in the file.
foreach(func): Runs a function func on each element of the dataset. This is usually,done for side effects such as updating an Accumulator or interacting with external storage systems.
API used to implement Spark Recommendation in Scala Programming Language [22]
org.apache.spark.mllib.recommendation.{ALS, Rating}
Function used is ALS.train()
Example:

```
val model=ALS.train(ratings, rank, numIterations, lambda)
//Recommend 11 product to user with userid 7
val pred=model.recommendProducts(7,11)
```

[23]**API to evaluate ALS Recommendation System in scala programming language**
For Evaluation of ALS we used the following class:
org.apache.spark.mllib.evaluation.RankingMetrics

org.apache.spark.mllib.evaluation.RegressionMetrics.
Ranking Metrics is used in evaluation precision at some value k .
RegressionMetics is used in evaluating MAE and RMSE
For evaluation our parameter for ALS will be rank and lambda.

12 .RESULT OF EVALUATION

Evaluation of User based similarity in Mahout

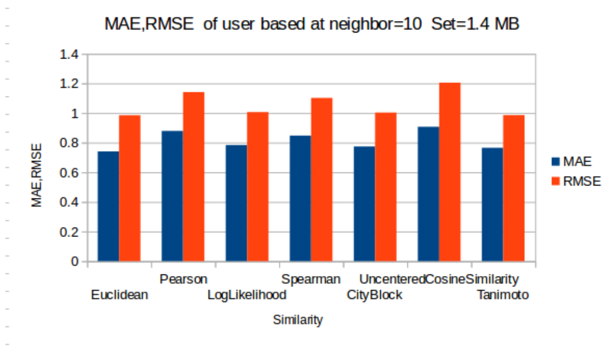


Chart-1

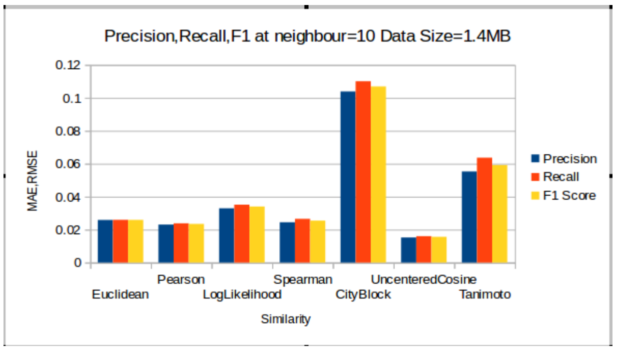


Chart-2

Evaluation of Item based similarity in Mahout

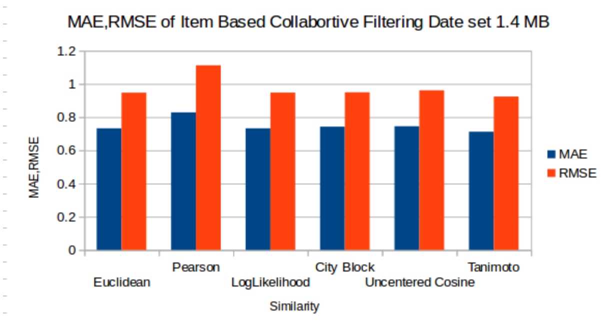


Chart-3

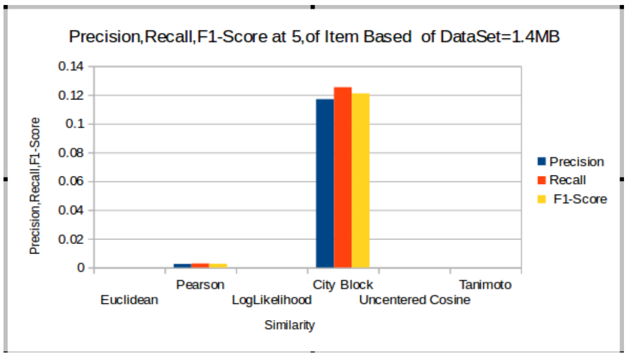


Chart-4

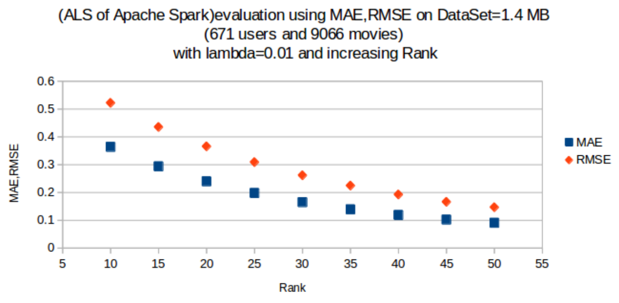


Chart-5

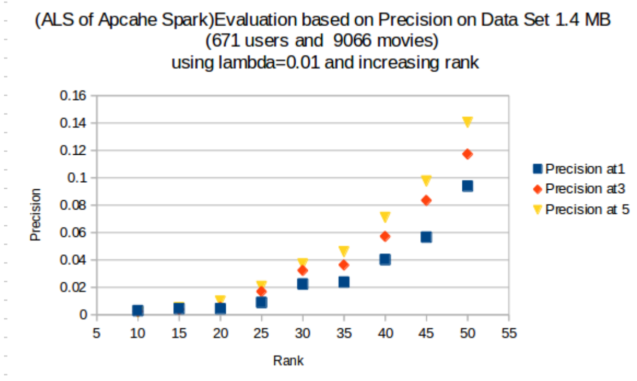


Chart-6

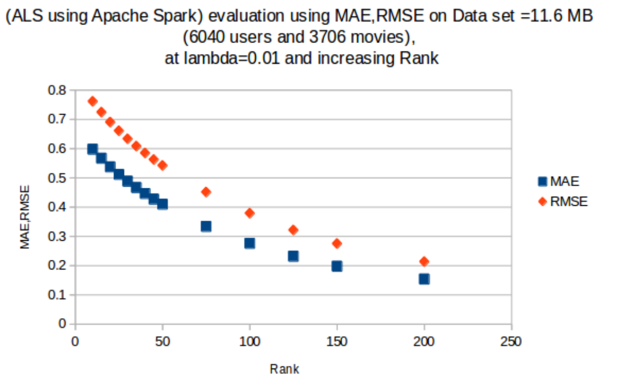


Chart-7

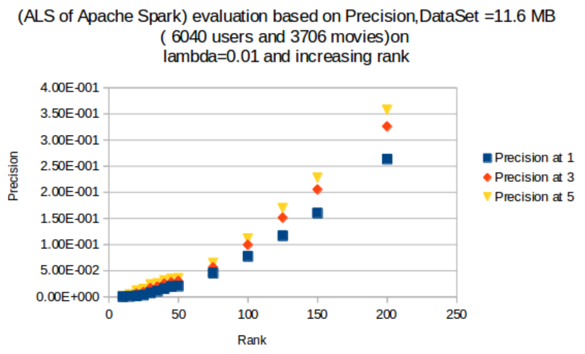


Chart-8

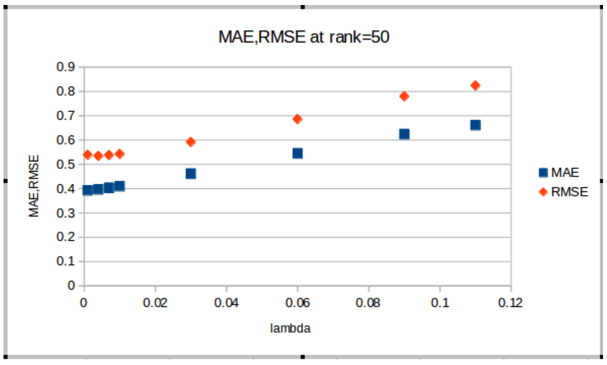


Chart-9

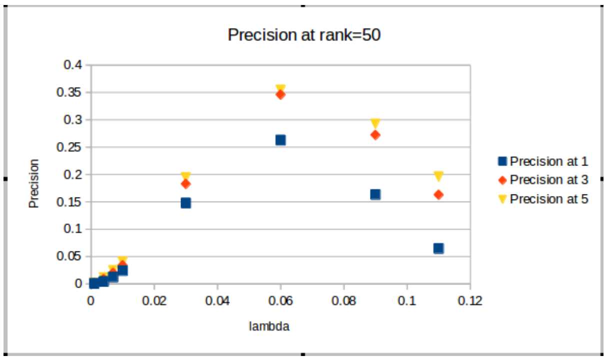


Chart-10

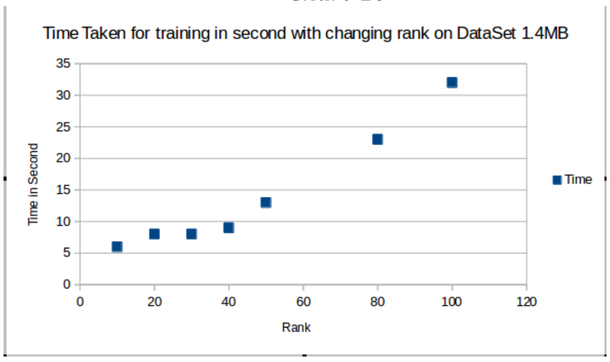


Chart-11

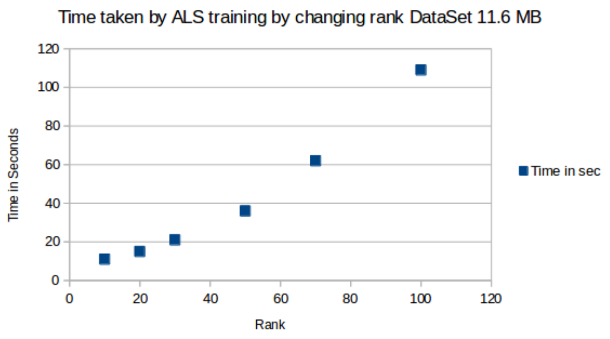


Chart-12

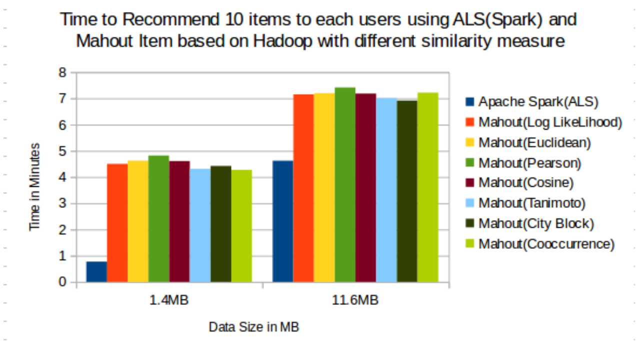


Chart-13

13. IMPORTANT POINTS FROM EVALUATION

1.)From our analysis of Mahout user Based on MovieLens Data set of 1.4MB we see that minimum MAE,RMSE out all similarity measure is of Euclidean with MAE=0.74072954757139 and RMSE=0.984777673459325 in Chart-1
2.)From our analysis of Mahout Item Based Data set of 1.4MB we see that minimum MAE,RMSE out all similarity measure is Tanimoto with =0.712088209 and RMSE is=0.924050483. in Chart-3
3.)From our analysis of Precision,Recall and F1 Score of userbased highest score of city block similarity:Precision at 5=0.1039007092,Recall at 5=0.1100768322,F1-Score at 5=0.1068996387 Chart-2
4.)From our analysis of Precision,Recall and F1 Score of Itembased highest score of city block similarity: Precision at 5=0.1170212766,Recall at 5= 0.1253250591, F1-Score at 5=0.121030907 Chart-4
5.)From our analysis of ALS on 1.4 MB dataset the MAE and RMSE decreases, and Precision increases with increase in rank, even at rank=10 MAE is 0.363812698 much less than userbased and item based of mahout. Chart-5,6
6.)MAE,RMSE,Precision on Data set 11.6 MB gives the same behaviour. Chart-7,8
7.)From further analysis we see that MAE,RMSE increases with increase in lambda as in Chart-9

8.)From our analysis on data set 11.6 MB, if we fix rank and increase lambda then precision first increases and then decreases, that is there is peak at lambda=0.06. Chart-10
9.)From our analysis on graph of Chart-11,12 we see that training time increases with rank.
10.)From graph Chart-13 of running time we see that ALS takes least time compared to other similarity measures.

14. EXTRA TOOLS USED IN IMPLEMENTATION

a. HBase

[26],[35]HBase is a distributed column-oriented NoSQL database built on top of the Hadoop file system. It is an open-source project and is horizontally scalable. HBase is a data model that is similar to Google's big table designed to provide quick random access to huge amounts of structured data. It leverages the fault tolerance provided by the Hadoop File System (HDFS).It is a part of the Hadoop ecosystem that provides random real-time read/write access to data in the Hadoop File System.
Data Storage Model
1.)Column-oriented database(column families)
2.)Table consist of rows, each of which has a primary key(row key).
3.)Each row may have any number of columns.
4.)Table schema only defines column families(column family can have any number of columns)
5.)Each cell value has a timestamp.
Hbase has three components:Client Library,Master Server,Region Server
Master Server:
Assigns regions to the region servers and takes the help of Apache Zookeeper for this task.
Regions: Regions are nothing but tables that are split up and spread across the region servers.
Region server:
The region servers have regions that: Communicate with the client and handle data-related operations.
Zookeeper: Zookeeper is an open-source project that provides services like maintaining configuration information, naming, providing distributed synchronization, etc.
Commands that operate in table:
create,list,disable,enable,describe,alter,exists,drop
In this implementation we will use Apache Phoenix as an open source SQL skin for HBase. Phoenix is a relational layer over HBase.
Accessing Hbase data by Phoenix is much easier than direct HBase API.

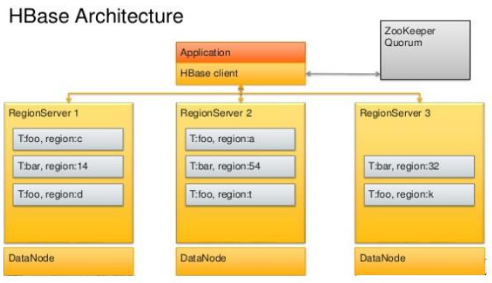


Fig- 8[30]

b. Apache Phoenix

[28],[29],[30],[33]Apache Phoenix is project from Apache Software foundation. It provides SQL interface to HBase. It is developed in Java. Apache Phoenix enables OLTP and OLAP in Hadoop for low latency applications by combining the best of both.
1.)It is delivered as embedded JDBC driver for Hbase data.
2.)It supports composite primary keys.
3.)Allows columns to be modeled as a muti-part row key or key/value cells.
4.)Full query support with predicate push down and optimal scan key formation.
5.)DDL support:CREATE TABLE,DROP TABLE and ALTER TABLE
6.)DML support:Atomic updates by UPSERT VALUES for insertion,SELECT UPSERT for data transfer between different tables,DELETE TABLE for deletion.
7.)Compiles SQL queries into a series of Hbase scans,and runs those scans in parallel to produce regular JDBC result sets.
8.)It can be easily integrate with HBase and Pig as per our need.
9.)CSV Bulk Loader. Bulk load CSV files into HBase either through map-reduce or a client-side script.
10.)Apache Pig Loader . Support for a Pig loader to leverage the performance of Phoenix when processing data through Pig.

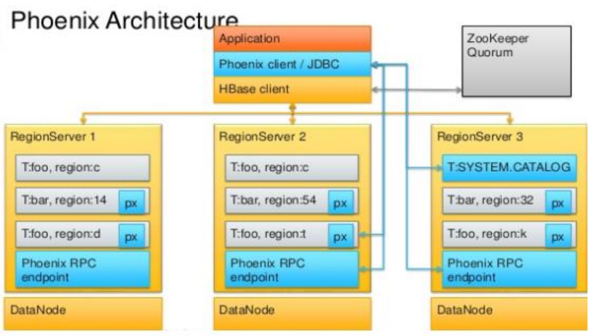


Fig- 9.[30]

Why is Phoenix so fast?[33]

1.)Apache Phoenix breaks up SQL queries into multiple Hbase scans and run them in parallel.
2.) Phoenix performs the computation to the data by using:
2.1.)Co-processors to perform operations on server-side thus minimizing client/server data transfer.
2.2.) Customs filters to prune data as close to the source as possible in addition,to minimize any start up cost.
2.3)Phoenix uses native HBase API rather than map-reduce framework.
3.)Phoenix leverages below HBase customs filters to provide higher performance.
3.1)Essential Column Family filter leads to improved performance.
3.2)Phoenix Skip scans Filters.
Phoenix Java API(JDBC) is used by us in JSP(Java Server Pages):

Connection url used is:jdbc:phoenix:localhost
We have also used CsvBulkLoadTool of Phoenix for loading data from CSV file to HBase table.

c. Apache Pig

[27][29]Apache Pig is an abstraction over MapReduce.It is a tool/platform which is used to analyze larger sets of data representing them as data rows. Pig is generally used with Hadoop; we can perform all the data manipulation operations in Hadoop using Pig. We are using Pig Load Store operation when combined with phoenix:
A = load 'hbase://table/ratings' using
org.apache.phoenix.pig.PhoenixHBaseLoader('localhost');
STORE A INTO 'recomfile' USING PigStorage(',');
In above two operations first operation is used to read data from table ratings in hbase ,and second operation is used to put those data in CSV file in hdfs.

d. Crontab:

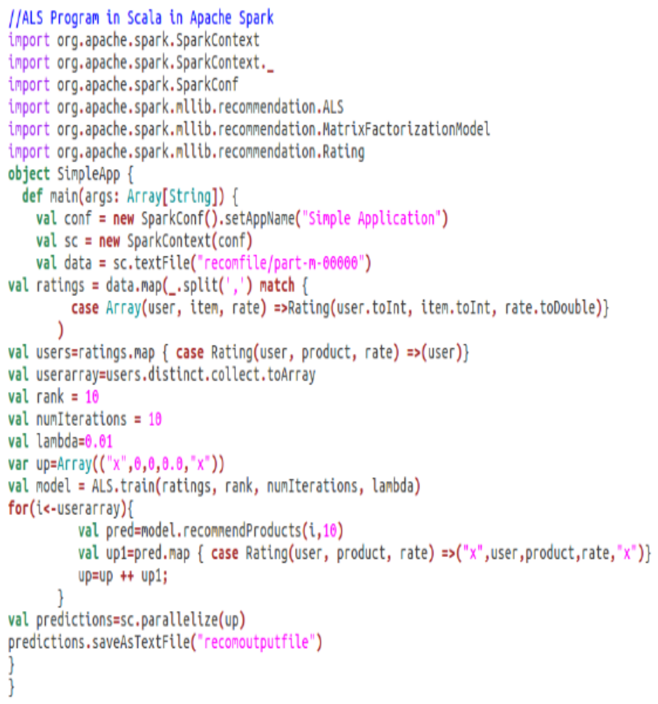
[32]Cron is one of the most powerful tool in a Linux/Unix based operating systems. A cron job is a Linux utility used for scheduling a task to be executed in the specific time according to its schedule at designated time. Cron is a daemon that runs in the background while the schedule of different jobs resides in its configuration file called “crontab” where all the tasks and times are scheduled.

15. Overall Design of Recommendation System

In our recommendation system each user will have a unique user id, each item will also have unique item id. The rating will be in scale of 1,2,3,4,5.

1.)Out of Mahout and Spark(ALS) we used Spark(ALS with explicit feedback)
2.)The database we will use is HBase with SQL interface provided with Apache Phoenix. Ratings data will be stored in HBase table created by Apache Phoenix of name ratings with attributes (uid,itemid,rating)
3.) Input file to ALS program in scala must be in CSV format, uid,itemid,rating so for this, data that is stored in HBase ratings table has to be exported to CSV file. Use Pig to transfer data from HBase table by Phoenix to CSV file.
A = load 'hbase://table/ratings' using
org.apache.phoenix.pig.PhoenixHBaseLoader('localhost');
STORE A INTO 'recomfile' USING PigStorage(',');
4.)Run ALS program of spark ,on that CSV file as input and save the output as CSV file.
5.)Use CsvBulkLoadTool of Apache Phoenix to transfer output CSV file of program to HBase table named recommends with attributes (d1,uid,itemid,rating,d2) where d1 and d2 are dummy columns to store dummy string “(x” and “x)” .
hadoop jar phoenix-4.8.2-HBase-1.2-client.jar
org.apache.phoenix.mapreduce.CsvBulkLoadTool
-table recommends -input
hdfs://localhost:14102/user/hadoop/recomoutputfile/part-00000
6.)The shell script to do these tasks is made and then added to crontab in ubuntu.

//Program in Scala for Recommendation



```
//ALS Program in Scala in Apache Spark
import org.apache.spark.SparkContext
import org.apache.spark.SparkContext._
import org.apache.spark.SparkConf
import org.apache.spark.mllib.recommendation.ALS
import org.apache.spark.mllib.recommendation.MatrixFactorizationModel
import org.apache.spark.mllib.recommendation.Rating
object SimpleApp {
  def main(args: Array[String]) {
    val conf = new SparkConf().setAppName("Simple Application")
    val sc = new SparkContext(conf)
    val data = sc.textFile("recomfile/part-m-00000")
val ratings = data.map(_.split(',') match {
          case Array(user, item, rate) =>Rating(user.toInt, item.toInt, rate.toDouble)}
       )
val users=ratings.map { case Rating(user, product, rate) =>(user)}
val userarray=users.distinct.collect.toArray
val rank = 10
val numIterations = 10
val lambda=0.01
var up=Array(("x",0,0,0.0,"x"))
val model = ALS.train(ratings, rank, numIterations, lambda)
for(i<-userarray){
          val pred=model.recommendProducts(i,10)
          val up1=pred.map { case Rating(user, product, rate) =>("x",user,product,rate,"x")}
          up=up ++ up1;
       }
val predictions=sc.parallelize(up)
predictions.saveAsTextFile("recomoutputfile")
}
}
```

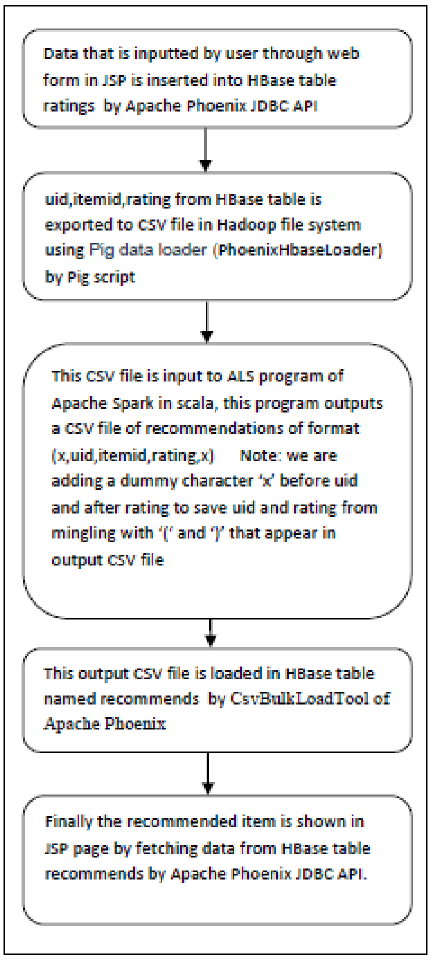

Data that is inputted by user through web form in JSP is inserted into HBase table ratings by Apache Phoenix JDBC API
uid,itemid,rating from HBase table is exported to CSV file in Hadoop file system using Pig data loader (PhoenixHbaseLoader) by Pig script
This CSV file is input to ALS program of Apache Spark in scala, this program outputs a CSV file of recommendations of format (x,uid,itemid,rating,x) Note: we are adding a dummy character 'x' before uid and after rating to save uid and rating from mingling with '(' and ')' that appear in output CSV file
This output CSV file is loaded in HBase table named recommends by CsvBulkLoadTool of Apache Phoenix
Finally the recommended item is shown in JSP page by fetching data from HBase table recommends by Apache Phoenix JDBC API.
Hadoop File System

Fig-10 Design of our recommendation system

16. CONCLUSION

By using ALS of spark MLlib we can reduce MAE ,RMSE as per our need and increase precision as well. Increasing rank reduces MAE and RMSE and increases Precision. Increasing lambda increased MAE, RMSE it has been also seen increasing lambda increases Precision but to some extent after that precision decreases ,that is there is a peak in graph of precision vs lambda.

We have used HBase to store rating data directly into Hadoop file system entered by Web form in JSP rather than transferring it from local file system to hadoop file system. Records are inserted into HBase by Apache Phoenix that is SQL interface to NoSQL database HBase. Due to random access HBase is faster for real time insertion and retrieval. We use Pig to transfer data from HBase table to CSV file using PhoenixHBaseLoader, as CSV file required by Spark MLlib ALS. The output of ALS is again put into HBase by Phoenix CsvBulkLoadTool with mapreduce. We have also seen that increasing rank increases training time of ALS, so value of rank and lambda should be taken as per our requirement and according to size of data set.

Future Work: Integrate Pig with Apache Spark to reduce time in transferring data from HBase to CSV and from CSV to HBase thus making it more efficient.

REFERENCES

[1] https://en.wikipedia.org/wiki/Recommender_system
[2] V. Srilakshmi , V.Lakshmi Chetana , T.P.Ann Thabitha "A Study on Big Data Technologies" ,International Journal of Innovative Research in Computer and Communication Engineering ,June 2016
[3] Harshawardhan S.Bhosale,Prof.Devendra P. Gadekar "A Review Paper on Big Data and Hadoop" Department of Computer Engineering, JSPM's Imperial College of Engineering & Research, Wagholi, Pune October 2014
[4] Peter Casinelli "Evaluating and Implementing Recommender Systems As Web Services Using Apache Mahout"
[5] Swati Pandey , T. Senthil Kumar "Customization of Recommendation System using Collaborative filtering algorithm on cloud using mahout",International Journal of Research in Engineering and Technology, May 2014
[6] Dr. Senthil Kumar ,Thangavel, Neetha Susan Thampi, Johnpaul C I "Performance Analysis of Various Recommendation Algorithms Using Apache Hadoop and Mahout" ,International Journal of Science and Engineering Research, Volume 4, Issue 12, December-2013.
[7] Chandan Venkatesh, Deekshith Kumar, Ganesh Madhav R, "Increasing Efficiency of Recommendation System using Big Data Analysis", October 2015,International Journal on Recent and Innovation Trends in Computing and Communication
[8] Swati Sharma, Manoj Sethi "Implementing Collaborative Filtering on large scala data using hadoop and mahout" ,July 2015, International Research Journal of Engineering and Technology
[9] Saikat Bagchi "Performance and Quality Assessment of Similarity Measures in Collaborative Filtering Using Mahout" ,Science Direct,2015
[10] https://www.tutorialspoint.com/hadoop/hadoop_mapreduce.htm
[11] http://searchbusinessanalytics.techtarget.com/definition/big-data-analytics
[12] https://en.wikipedia.org/wiki/Likelihood-ratio_test
[13] Mahout in Action, Sean Owen, Robin Anil, Ted Dunning, Ellen Friedman, Part 1,Chapter 2,3
[14] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl "Item-Based Collaborative Filtering Recommendation Algorithm" GroupLens Research Group/Army RPC Research center,Department of Computer Science and Engineering ,University of Minnesota, Minneapolis, MN 55455
[15] https://en.wikipedia.org/wiki/Pearson_correlation_coefficient

[16] https://en.wikipedia.org/wiki/Jaccard_index
[17] https://en.wikipedia.org/wiki/Cosine_similarity
[18] https://en.wikipedia.org/wiki/Taxicab_geometry
[19] https://en.wikipedia.org/wiki/Euclidean_distance
[20] https://www.slideshare.net/Cataldo/apache-mahout-tutorial-recommendation-20132014
[21] https://datasciencemadesimpler.wordpress.com/tag/alternating-least-squares/
[22] http://spark.apache.org/docs/latest/mllib-collaborative-filtering.html#collaborative-filtering
[23] http://spark.apache.org/docs/latest/mllib-evaluation- metrics.html
[24] http://www.tutorialspoint.com/apache_spark/
[25] http://www.ics.uci.edu/~welling/teaching/CS77Bwinter12/presentations/Slides_Recommender_Systems_Introduction_Version_Oct_2011/ppt/Chapter%2007%20-%20Evaluating%20recommender%20systems.ppt
[26] https://www.tutorialspoint.com/hbase/
[27] https://www.tutorialspoint.com/apache_pig/
[28] https://phoenix.apache.org/Phoenix-in-15-minutes-or-less.html
[29] https://phoenix.apache.org/pig_integration.html
[30] https://www.slideshare.net/enissoz/apache-phoenix-past-present-and-future-of-sql-over-hbase
[31] https://engineering.linkedin.com/recommender-systems/browsemap-collaborative-filtering-linkedin
[32] https://vexxhost.com/resources/tutorials/how-to-use-cron-jobs-for-automation-on-ubuntu-14-04/
[33] http://hadooptutorial.info/apache-phoenix-hbase-an-sql-layer-on-hbase/
[34] http://mahout.apache.org/users/recommender/intro-itembased-hadoop.html
[35] Nikita Bhojwani,Asst Prof. Vatsal Shah,B.V.M. "A Survey on Hadoop HBase System" Engineering College, V.V. Nagar, Gujarat, India International Journal of Advance Engineering and Research Development(2016)
[36] http://tuxdna.in/files/notes/mahout.html

BIOGRAPHIES

Mr. Archit Verma

I am M.Tech Computer Science and Engineering Student at United College of Engineering and Research (U.C.E.R.) Allahabad .My area of interest includes Artificial Intelligence, Data Mining, Big Data , Web Technology,J2EE and Distributed Systems .

Mr. Dharmendra Kumar

I am Associate Professor in Computer Science and Engineering of United College of Engineering and Research, Allahabad. I have done M.Tech. From Motilal Nehru National Institute of Technology, Allahabad. Currently, I am pursuing Ph.D. from Dr. APJ Abdul Kalam Technical University, Lucknow. My areas of research interest are Peer to Peer Network, Wireless Ad-Hoc Network and Sensor Network.