**Example** : Write a Hadoop MapReduce program for Movie Recommendation System.

**Object**: Using given dataset, find Movie Recommendations using Hadoop MapReduce program.

**Dataset**:

Our example is conducted on the real world MovieLens dataset. The data sets were collected over various periods of time, depending on the size of the set. It contains 100,000 ratings from 1000 users on 1700 movies. Users were selected at random for inclusion. Users are represented by its id and item are also represented by id. Format for the file is as follows.

## Data File Structure

All ratings are contained in the DATA file. Each line of this file represents one rating of one movie by one user, and has the following format:

UserID::MovieID::Rating::Timestamp

The lines within this file are ordered first by UserID, then, within user, by MovieID.

Ratings are made on a 5-star scale, with half-star increments.

Timestamp represent seconds since midnight Coordinated Universal Time (UTC) of January 1, 1970.

Download dataset- URL: <http://www.grouplens.org/node/73>

It contains 3 dataset of different size the MovieLens 100,000 rating dataset, the MovieLens 1,000,000 rating dataset, and the MovieLens 10,000,000 rating dataset. In dataset folder there is a file called DATA file, this file is to be used as input.

**Description:**

What is Recommendation System?

**Recommendation system** are a subclass of [information filtering system](http://en.wikipedia.org/wiki/Information_filtering_system) that seek to predict the 'rating' or 'preference' that user would give to an item. Recommender systems have become extremely common in recent years, and are applied in a variety of applications. The most popular ones are probably movies, music, news, books, research articles, search queries, social tags, and products in general.

Imagine that you own a online movie business, and you want to suggest for your clients movie recommendations. Your system runs a rating system, that is, people can rate movies with 1 to 5 starts.

Our goal is to calculate how similar pairs of movies are, so that we recommend movies similar to movies you liked. Using the correlation we can:

* For every pair of movies A and B, find all the people who rated both A and B.
* Use these ratings to form a Movie X vector and a Movie Y vector.
* Calculate the correlation between those two vectors
* When someone watches a movie, you can recommend the movies most correlated with it

We have given a movie file as input in which each line of the file represents one rating of one movie by one user UserID::MovieID::Rating::Timestamp.

You want to compute how similar pairs of movies are, so that if someone watches the movie The Matrix, you can recommend movies like BladeRunner. So how should you define the similarity between two movies ?

One possibility is to compute their correlation. The basic idea behind it is for every pair of movies X and Y, find all the people who rated both X and Y. Use these ratings to form a Movie X vector and a Movie Y vector. Then, calculate the correlation between these two vectors. Now when someone watches a movie, you can now recommend him the movies most correlated with it.

Our task is to find similarity between pair of item using correlation formula.

Similarity(X,Y) = Correlation(X,Y)

X and Y are items

Steps:

1. For pair of items find the users rated both the items X and Y

2. Form two vector X and Y



3. Calculate correlation between X and Y using following formula



n is number of users rated both the items, x and y are rating values between 1 to 5

Similarity(X,Y)=correlation(X,Y)

Solution:

To solve the problem we are using chaining of two MapReduce job. Output of the first job work as input to the second.

To chain the multiple jobs we have used multiple driver method.

Step-1 In first step dataset DATA file is given as input to the first MapReduce job. After completion it will generate output1 file which will work as input to Mapreduce job-2.

Step-2 MapReduce job-2 will wait for the completion of MapReduce job-1 ,its output-1 will work as input to the MapReduce job-2.MapReduce job-2 will generate final output.



Figure: Chaining of Two MapReduce job

MapReduce Job-1 : Work of the first MapReduce job is to collect all the user rated both the items.

MapReduce Job-2 : second MapReduce job will find the similarity between items using correlation formula.

**Algorithm-1**

Map-1 Emit the user\_id as key and (item and rating) as value

Job-1

Input:-

key-line offset

value- row of input file contains (item\_id,user\_id,rating)

Output:-

Key- user\_id

Value-(item\_id,rating) pair

Require: Input dataset containing User\_id, Item\_id, rating fields

**Procedure:**

· user\_id, item\_id, rating = line.split('\t')

· key=user\_id

· value=item\_id

· value.append(rating)

· emit(key,value)

**Algorithm-2**

Reducer-1 For each user, emit a row containing their "postings"(item , rating pairs)

Input:-

key- user\_id

value- Sequence of (user\_id , rating)

Output:-

Key- user\_id

Value-row contain all posting of user (item\_id , rating)

**Procedure:**

· item\_count = 0

· item\_sum = 0

· final = []

· for item\_id, rating in values

· {

· item\_count += 1

· item\_sum += rating

· final.append((item\_id, rating))

· Key=user\_id

· Value= item\_count, item\_sum, final

·   Emit (key,value)

· }

**Algorithm-3**

Map-2: The output drops the user from the key entirely, instead it emits the pair of items as the key

Job-2: Require Output of job-1 as input to the job-2.

Input:-

key-user\_id

value- row containing all the posting of user (user\_id , rating)

Output:-

Key- (item\_id , item\_id)

Value-(rating , rating)

**Procedure:**

· item\_count , item\_sum, ratings = values

· for item1, item2 in combinations(ratings, 2)

· {

· key=(item1[0], item2[0])

· value=( item1[1], item2[1])

· Emit=(key , value)

· }

**Algorithm-4**

Reduce-2: Sum components of each co rating pair across all users who rated both item x and item y, then calculate correlation similarity.

Input:-

key- (item\_id , item\_id)

value- sequence of rating pair(rating , rating)

Output:-

Key- (item\_id , item\_id)

Value-(similarity , n)

**Procedure**

· sum\_xx, sum\_xy, sum\_yy, sum\_x, sum\_y, n = (0.0, 0.0, 0.0, 0.0, 0.0, 0)

· sum\_x += item\_x

· n += 1

· item\_pair, co\_ratings = pair\_key, lines

· item\_xname, item\_yname = item\_pair

· for item\_x, item\_y in lines:

· {

· sum\_xx += item\_x \* item\_x

· sum\_yy += item\_y \* item\_y

· sum\_xy += item\_x \* item\_y

· sum\_y += item\_y

similarity = normalized\_correlation(n, sum\_xy, sum\_x, sum\_y,sum\_xx, sum\_yy)

· Key=(item\_xname,item\_yname)

· Value= (similarity)

· Emit(key,value)

· }

Example:

MAP-1

17 70 3

35 21 1

49 19 2

49 21 1

49 70 4

87 19 1

87 21 2

98 19 2

17 70,3

35 21,1

49 19,2

49 21,1

49 70,4

87 19,1

87 21,2

98 19,2

17 (70,3)

35 (21,1)

49 (19,2 21,1 70,4)

87 (19,1 21,2)

98 (19,2)

REDUCE-1

17 1,3,(70,3)

35 1,1,(21,1)

49 3,7,(19,2 21,1 70,4)

87 2,3,(19,1 21,2)

98 1,2,(19,2)

17 1,3,(70,3)

35 1,1,(21,1)

49 3,7,(19,2 21,1 70,4)

87 2,3,(19,1 21,2)

98 1,2,(19,2)

MAP-2

19,21 2,1

19,70 2,4

21,70 1,4

19,21 1,2

19,21 (2,1 1,2)

19,70 (2,4)

21,70 (1,4)

19,21 0.4

21,19 0.4

19,70 0.6

70,19 0.6

21,70 0.1

70,21 0.1

REDUCE-2

**Input:**

UserID::MovieID::Rating::Timestamp



Figure- Sample MovieLens DataSet File

**Output**:

Each row of output contain pair of movie and its similarity as follows

(Movie\_id, Movie\_id, similarity)