**Product Recommendation Engine using Amazon Dataset**

**Objective-**

Recommendation engines are nothing but an automated form of a “shop counter guy”. You ask him for a product. Not only he shows that product, but also the related ones which you could buy. They are well trained in cross selling and up selling. So, does our recommendation engines. Also did some Ad-hoc Analysis using Map Reduce, Hbase and Neo4J in Hadoop environment.

**Process –**

Amazon Electronic products dataset contains the record of electronics with the reviews of different users in JSON format. After getting data from online website, I converted JSON dataset to CSV format to do analyses. I used this converted CSV format records to find out the recommendation of products for users to buy the products along with it. I analyzed the products with favorite rating got it by reviewers with the total count. I also used the filters and secondary sorting to sort the records as per rating and total number of reviews. I used filters and sorting techniques to analyze the favorite rating, total purchase count of individual product.

*Dataset link -* “<http://snap.stanford.edu/data/amazon/productGraph/categoryFiles/reviews_Electronics_5.json.gz>”

**Sample Review-**

{"reviewerID": "AO94DHGC771SJ", "asin": "0528881469", "reviewerName": "amazdnu", "helpful": [0, 0], "reviewText": "We got this GPS for my husband who is an (OTR) over the road trucker. Very Impressed with the shipping time, it arrived a few days earlier than expected... within a week of use however it started freezing up... could of just been a glitch in that unit. Worked great when it worked! Will work great for the normal person as well but does have the \"trucker\" option. (the big truck routes - tells you when a scale is coming up ect...) Love the bigger screen, the ease of use, the ease of putting addresses into memory. Nothing really bad to say about the unit with the exception of it freezing which is probably one in a million and that's just my luck. I contacted the seller and within minutes of my email I received a email back with instructions for an exchange! VERY impressed all the way around!", "overall": 5.0, "summary": "Gotta have GPS!", "unixReviewTime": 1370131200, "reviewTime": "06 2, 2013"}

where

* reviewerID - ID of the reviewer, e.g. AO94DHGC771SJ
* asin - ID of the product, e.g. 0528881469
* reviewerName - name of the reviewer
* helpful - helpfulness rating of the review, e.g. 2/3
* reviewText - text of the review
* overall - rating of the product
* summary - summary of the review
* unixReviewTime - time of the review (unix time)
* reviewTime - time of the review (raw)

**Steps -**

**1) Converted to CSV format**

I wrote the hava program to convert he JSON format to CSV format with limited no of columns.

I took the following columns from dataset- reviewerID, asin – ID, helpful, reviewText, overall, reviewTime.

The CSV format sample is -

AO94DHGC771SJ,528881469,5,6,2,2013,0,0,We got this GPS for my husband who is an (OTR) ove,

* uerId- AO94DHGC771SJ
* PrudtcID - 528881469
* rating – 5
* DateMM – 6
* DateDD – 2
* DateYY – 2013
* HelpA – 0
* HelpB – 0
* Review -We got this GPS for my husband who is an (OTR) ove

Note - the code to convert JSON to CSV is- find in Appendix

2)Wrote Mapreduce to find out the products with favorite rating by count of reviewrs with total count of the reviewers.

O/P -

ProductID FavRating favRatingByNoofUsers TotalReviewers

0528881469 1 2 5

0594451647 5 3 5

0594481813 5 3 8

0972683275 5 143 219

1400501466 5 27 43

Created another MapReduce to group by rating and sort by total reviewers rating.

3) wrote the filter to filter out the records as per the rating and year

o/p-

AO94DHGC771SJ,528881469,5,6,2,2013,0,0,We got this GPS for my husband who is an (OTR) ove,

AO94DHGC771SJ,528881469,5,6,2,2013,0,0,We got this GPS for my husband who is an (OTR) ove,

AO94DHGC771SJ,528881469,5,6,2,2013,0,0,We got this GPS for my husband who is an (OTR) ove,

4)

top 50 reviewed Products using chaining and setSortComparatorClass

o/p

B007WTAJTO 4915

B003ES5ZUU 4143

B00DR0PDNE 3798

B0019EHU8G 3435

B002WE6D44 2813

B003ELYQGG 2652

B0002L5R78 2599

B009SYZ8OC 2542

B00BGGDVOO 2104

B002V88HFE 2082

B0074BW614 2069

B000LRMS66 1960

B006GWO5WK 1917

B000QUUFRW 1890

B005FYNSPK 1884

B005HMKKH4 1866

B0041Q38NU 1812

B00622AG6S 1710

B004QK7HI8 1581

B005DKZTMG 1564

B000VX6XL6 1556

B001XURP7W 1485

B004XC6GJ0 1456

B004G6002M 1424

B0052SCU8U 1419

B000S5Q9CA 1393

B000BQ7GW8 1388

B002MAPRYU 1374

B005CT56F8 1365

B006W8U2MU 1339

B00316263Y 1332

B008OHNZI0 1304

B0012S4APK 1295

B00007E7JU 1279

B00E3W15P0 1270

B00004ZCJE 1258

B0015DYMVO 1223

B007I5JT4S 1210

B004GF8TIK 1204

B005CLPP84 1182

B001TH7GUU 1154

B00B46XUQU 1150

B0027VT6V4 1122

B001TH7GSW 1120

B002QEBMAK 1096

B002SZEOLG 1086

B0044YU60M 1082

B007R5YDYA 1077

B0043T7FXE 1050

B000I68BD4 1018

5) Wrote the MapReduce job to find the recommend 5 product with each product. Also used Mahout libraries to find out the recommended product for each product.

MapReduce Solution

We implement CWBTIAB with two iterations of MapReduce:

Phase 1: generate lists of all items bought by the same user. Grouping is handled by

the Hadoop framework, where both the mapper and the reducer perform an identity

function.

Phase 2: solve the co-occurrences problem on list items. We use the Stripes design

pattern and emit only the five most common co-occurrences.

O/p for MapReduce first phase is-

UserID with Items he reviewed[item1, item2, item3]

A000715434M800HLCENK9 B00HMZG3YS,B003ES5ZUU,B003AFONFU,B000UYYZ0M,B001EHAI6Y

A00101847G3FJTWYGNQA B005F778JO,B006T9B6R2,B00C7NSIO8,B009NB8WR0,B00IBPLI48,B00B19L8LO

A00166281YWM98A3SVD55 B005HSG3BA,B0074BW614,B005HGBF9W,B008BB7XVE,B007B5S8BU

A0046696382DWIPVIWO0K B002TTKC2G,B0025ZUF8K,B00CH643A8,B00HMXIKCS,B00E055H5O

A00472881KT6WR48K907X B00GYL9KK0,B009O7YUF6,B00CNZTPGA,B0000AZJZT,B007UIX1M4,B003LTRLPY,B0053OLY9O

A00473363TJ8YSZ3YAGG9 B008CXTX7S,B0039BYJ4A,B003NR57BY,B003MX4D0U,B002LB42NM,B00AGABISW,B00D2OLUZ0,B00CTUIT20,B00EOGGDXO

A005721627VX5W2COKKK2 B003ELYQGG,B0031MGUBC,B002WAZ0B0,B005QX7KYU,B005IQRMN4,B007HBLX8I,B001T9N0SU

A00700212KB3K0MVESPIY

O/p of second phase MapReduce job-

item recommended Item

0528881469 B005OM2AD4

0528881469 B00DSTUVHW

0528881469 B0036UU40C

0528881469 B002Y27P6O

0528881469 B00005AXHW

0594451647 B004NY9UV4

0594451647 B00AWKC0EC

0594451647 B006JPGM48

0594451647 B005B47AIU

6)Recommendation using Mahout libraries-

Mahout program need different input file which contains the data in number format.

### Step1: Create DataModel Object

a data model object, which holds a file that contains the Users, Items, and Preferences details of a product(Rating of product). Here is the sample data model file:

user, item, rating

1,1,5.0

2,1,1.0

3,1,3.0

4,1,2.0

5,1,1.0

6,2,5.0

7,2,2.0

### Step2: Create UserSimilarity Object and UserNeighborhood object

DataModel model = new FileDataModel(new File("/home/mayur/Documents/ADBMS/Project/ElectronicsMahotInput.csv")); //args[0]---"/home/mayur/Documents/ADBMS/Project/ElectronicsMahotInput.csv"---"/home/mayur/Desktop/TestReco.csv"

// this for test

//DataModel model = new FileDataModel(new File("/home/mayur/Desktop/TestReco.csv"));

UserSimilarity similarity = new PearsonCorrelationSimilarity(model);

UserNeighborhood neighborhood = new NearestNUserNeighborhood(2, similarity, model);//home/mayur/Desktop/TestReco.csv

UserBasedRecommender recommender = new GenericUserBasedRecommender(model, neighborhood, similarity);

NearestNUserNeighborhood - This class computes a neighborhood consisting of the nearest *n* users to a given user. "Nearest" is defined by the given UserSimilarity.

O/P of this-

0972683275,B00HFRWWAM,5.0,4.409356725146199

0972683275,B00E12FJK4,4.0,4.2

7799813393,B005JACJ50,5.0,4.52972972972973

9966694544,B0016BVKAQ,4.0,4.147368421052631

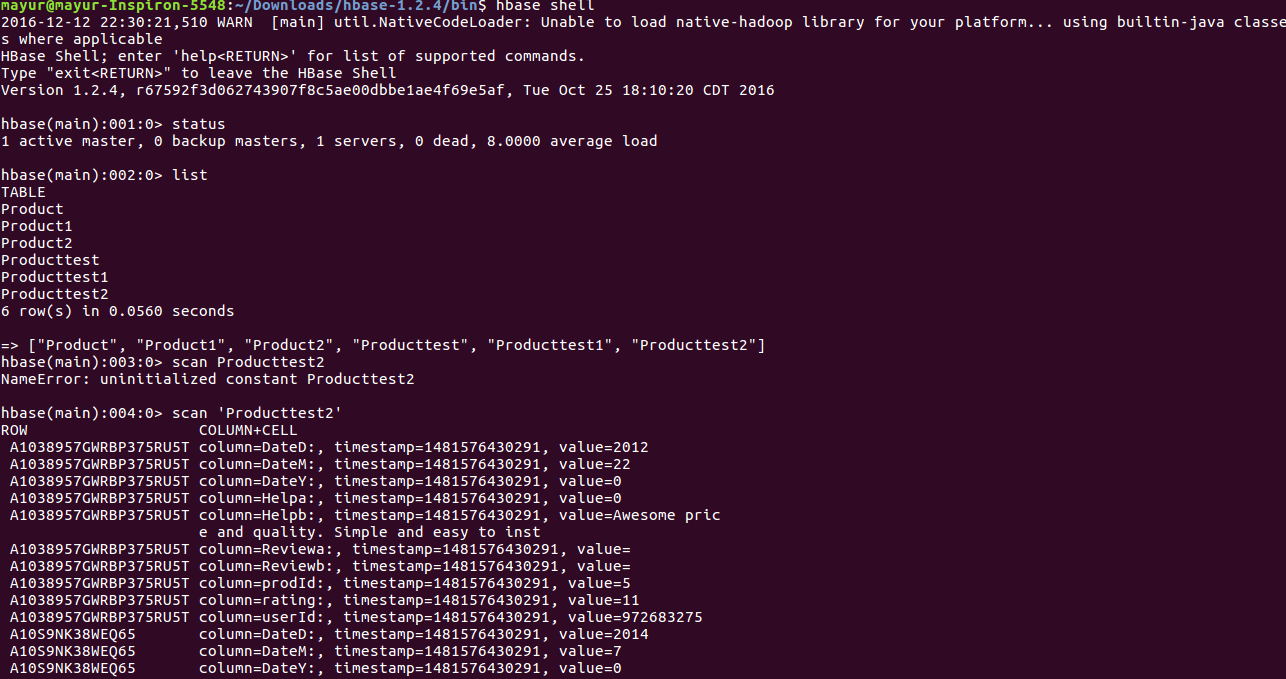
B000001OM5,B007WTAJTO,4.0,4.587589013224822

B00000IGBF,B0088CJT4U,4.0,4.060075093867334

7) Using Hbase Filter the data-

Created table using following command to filter the data.

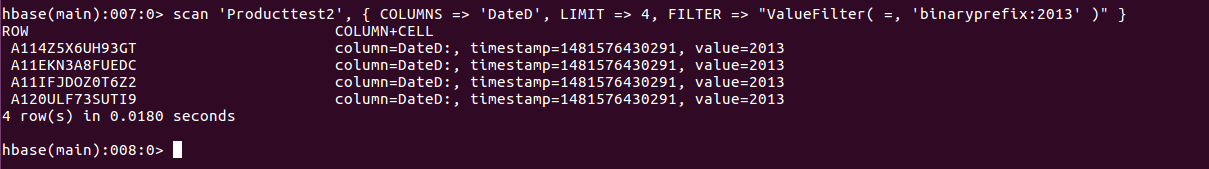
create 'Producttest1','userId','prodId','rating','DateM','DateD','DateY','Helpa','Helpb','Reviewa','Reviewb'

****

To find out the rows using filters on column-

The query to find rows is-

scan 'Producttest2', { COLUMNS => 'DateD', LIMIT => 4, FILTER => "ValueFilter( =, 'binaryprefix:2013' )" }

****

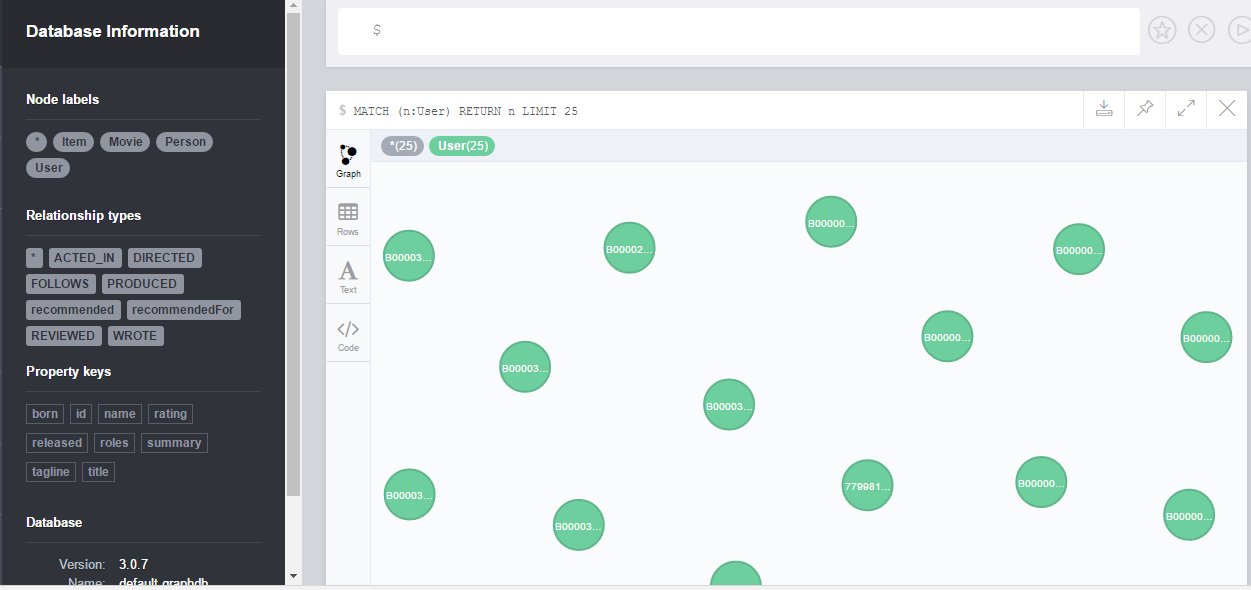
**Neo4j Database to display the connectivity the connectivity –**

LOAD CSV WITH HEADERS FROM "file:///ElectronicsRecoOutput.csv" AS line WITH line.USER\_ID as User, line.ITEM\_ID as Item LIMIT 1 RETURN User

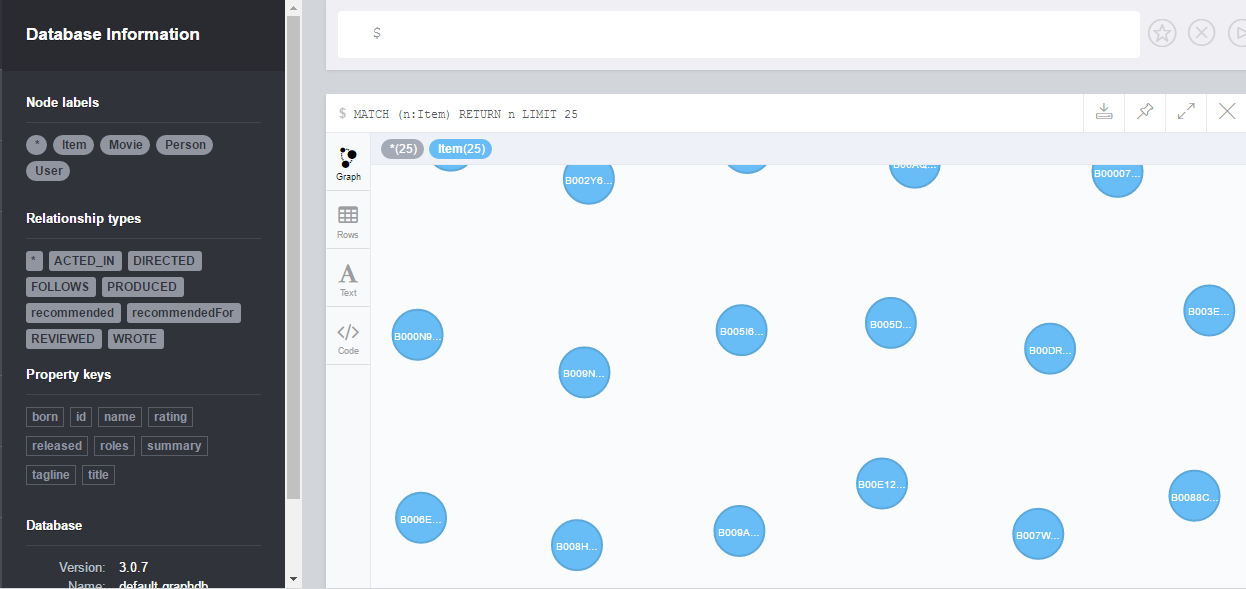
CREATE CONSTRAINT ON (u:User) ASSERT u.id IS UNIQUE;

CREATE CONSTRAINT ON (i:Item) ASSERT i.id IS UNIQUE;

MATCH (n:User) RETURN n LIMIT 25



MATCH (n:Item) RETURN n LIMIT 25



USING PERIODIC COMMIT

LOAD CSV WITH HEADERS FROM "file:///ElectronicsRecoOutput.csv" As line

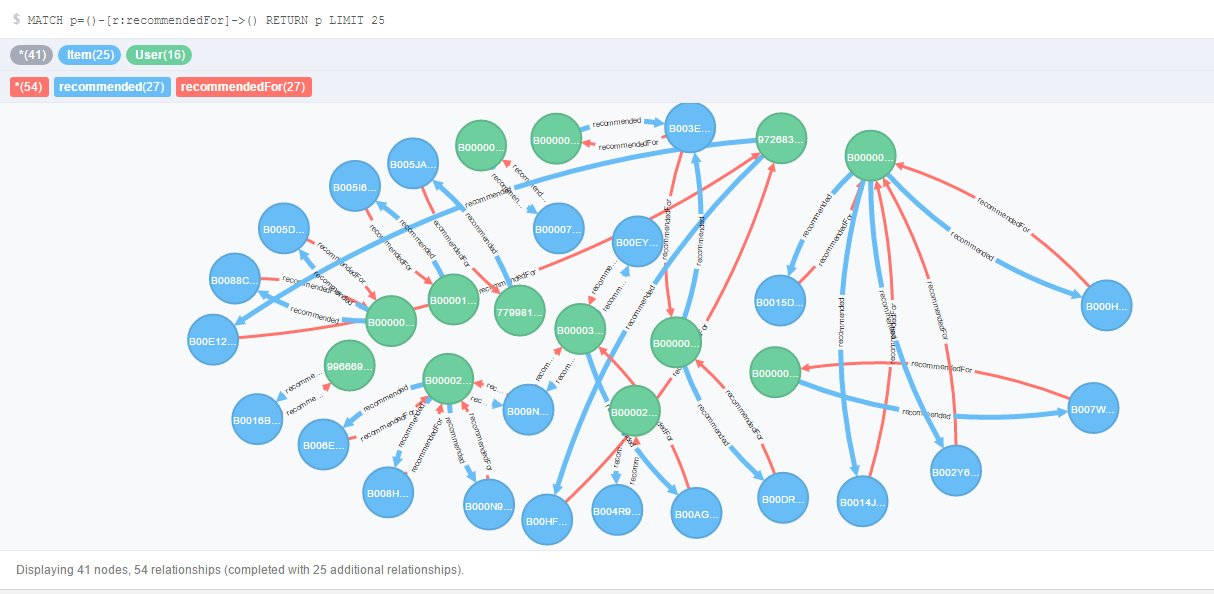
WITH line.USER\_ID as uid, line.ITEM\_ID as iid

MERGE (x:User {id:uid})

MERGE (y:Item {id:iid})

MERGE (y)-[:recommendedFor]->(x)

MERGE (x)-[:recommended]->(y)



Useful commands for project-

to convert json to CSV-

/home/mayur/NetBeansProjects/JSONtoCSV/dist/JSONtoCSV.jar

To run the ProductReview-

hadoop jar /home/mayur/NetBeansProjects/ProductReview/dist/ProductReview.jar /mayur/ProductReview/InputFiles/ElectronicsShort.csv /mayur/ProductReview/OutputFiles/ProductReviewData

partition by rating and secondary sort by total count -

hadoop jar /home/mayur/NetBeansProjects/ProductReviewSortByTtlCnt/dist/ProductReviewSortByTtlCnt.jar /mayur/ProductReview/OutputFiles/OpProductReviewData/part-r-00000 /mayur/ProductReview/OutputFiles/opSortedProductByTtlReview

hadoop jar /home/mayur/Desktop/HadoopJar/RatingFilter.jar /mayur/ProductReview/OutputFiles/OpTopRatedProduct12/part-r-00000 /mayur/ProductReview/OutputFiles/opFilteredRating6

top rated 50 product-

/home/mayur/NetBeansProjects/TopRated50Products/src/toprated50products/TopRated50Products.java

Product recommandation by MR--

--to create a list of items for one user

hadoop jar /home/mayur/NetBeansProjects/ProductRecoMR/dist/ProductRecoMR.jar /mayur/ProductReview/InputFiles/ElectronicsShortTemp.csv /mayur/ProductReview/OutputFiles/opMR1Reco14.csv

--recommandation of 5 items with each item

hadoop jar /home/mayur/NetBeansProjects/ProductRecoMR1/dist/ProductRecoMR1.jar /mayur/ProductReview/OutputFiles/opMR1Reco14.Csv /mayur/ProductReview/OutputFiles/FinalReco8

Product Recommandation by Mahout-

/home/mayur/NetBeansProjects/ProductReco/src/productreco/ProductReco.java

HBASE query-

create 'Product','prodId','favRating','favRatingCnt','ttlCnt'

create 'Producttest1','userId','prodId','rating','Date','Help','Review'

create 'Producttest1','userId','prodId','rating','DateM','DateD','DateY','Helpa','Helpb','Reviewa','Reviewb'

hbase> hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -Dimporttsv.separator=, -Dimporttsv.columns="HBASE\_ROW\_KEY,userId,prodId,rating,Date:M,Date:D,Date,Y,Help:a,Help:b,Review:a,Review:b" Producttest1 hdfs://localhost:50070/mayur/ProductReview/InputFiles/ElectronicsShortTemp.csv

hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -Dimporttsv.separator=, -Dimporttsv.columns="HBASE\_ROW\_KEY,userId,prodId,rating,DateM,DateD,DateY,Helpa,Helpb,Reviewa,Reviewb" Producttest2 hdfs://localhost:9000/mayur/ProductReview/InputFiles/ElectronicsShortTemp.csv

scan 'Producttest2', { COLUMNS => 'DateY', LIMIT => 10, FILTER => "ValueFilter( =, 'binaryprefix:<2006>' )" }

scan 'Producttest2', { FILTER => SingleColumnValueFilter.new(Bytes.toBytes('DateY'), CompareFilter::CompareOp.valueOf('EQUAL'), BinaryComparator.new(Bytes.toBytes('2013')))}

scan 'Producttest2' ,{ FILTER => "SingleColumnValueFilter('DateY',=, 'binaryprefix:2013')" }

scan 'Producttest2' ,{ FILTER => " MultipleColumnPrefixFilter('DateY') AND (ValueFilter(=,'binary:2013'))" }

scan 'Producttest2', { COLUMNS => 'DateY', LIMIT => 4, FILTER => "ValueFilter( =, 'binaryprefix:2013' )" }

http://jmcauley.ucsd.edu/data/amazon/amazon\_readme.txt

http://snap.stanford.edu/data/amazon/productGraph/categoryFiles/reviews\_Electronics\_5.json.gz

cloudera for hbase

ElectronicsShort1.csv

APPENDIX

1) JSON to CSV JAVA code-

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package jsontocsv;

/\*\*

\*

\* @author mayur

\*/

import java.io.File;

import java.io.IOException;

import java.util.List;

import java.util.Map;

import org.apache.commons.io.FileUtils;

import org.json.CDL;

//import org.json.JSONArray;

import org.json.JSONException;

//import org.json.JSONObject;

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import java.text.ParseException;

import java.util.HashMap;

import org.json.simple.JSONArray;

import org.json.simple.JSONObject;

import org.json.simple.parser.JSONParser;

//import org.jsontocsv.parser.JSONFlattener;

//import org.jsontocsv.writer.CSVWriter;

public class JSONtoCSV {

public static void main(String[] args) throws IOException, ParseException, org.json.simple.parser.ParseException {

double entryCount = 0;

String COMMA\_DELIMITER = ",";

String NEW\_LINE\_SEPARATOR = "\n";

FileWriter fileWriter = null; // this file writer for JSON to CSV

FileWriter mahoutfileWriter = null; // this file writter for JSON to Mahout input file

FileWriter mahoutKeyValuefileWriter = null;

JSONParser parser = new JSONParser();

int countReviewer = 1;

int countProduct = 1;

//this HashMap is to obtain a unique value for each reviewer and product

HashMap<String, Integer> mapReviewer = new HashMap<String, Integer>();

HashMap<String, Integer> mapProduct = new HashMap<String, Integer>();

try{

BufferedReader br = new BufferedReader(new FileReader("/home/mayur/Documents/ADBMS/Project/Electronics\_5.json"));

BufferedReader br1 = new BufferedReader(new FileReader("/home/mayur/Documents/ADBMS/Project/Electronics\_5.json"));

BufferedReader br2 = new BufferedReader(new FileReader("/home/mayur/Documents/ADBMS/Project/Electronics\_5.json"));

String brLine = null;

// fileWriter = new FileWriter("/home/mayur/Documents/ADBMS/Project/ElectronicsShort.csv");

mahoutfileWriter = new FileWriter("/home/mayur/Documents/ADBMS/Project/ElectronicsMahotInput.csv");

mahoutKeyValuefileWriter = new FileWriter("/home/mayur/Documents/ADBMS/Project/ElectronicsMahotKeyValue.csv");

//this while e add unique values for each reviewerId and ProductId

while ((brLine = br.readLine()) != null) {

JSONObject JsonLine = (JSONObject) parser.parse(brLine);

String revId = null ;

String prdId = null;

if (JsonLine != null) {

try{

revId = (String) JsonLine.get("reviewerID");

prdId = (String) JsonLine.get("asin");

if (revId != null) {

//System.out.println(countReviewer);

if(!mapReviewer.containsKey(revId)){

mapReviewer.put(revId, countReviewer);

countReviewer++;

}

}

if(prdId != null){

if(!mapProduct.containsKey(prdId)){

mapProduct.put(prdId , countProduct);

countProduct++;

}

}

}

catch(Exception e){

System.out.println(e);

System.out.println("ReviewId - " +revId +" Product ID" + prdId);

}

}

}

countReviewer =1;

countProduct = 1;

br.close();

while ((brLine = br1.readLine()) != null){

//System.out.println(brLine);

JSONObject JsonLine = (JSONObject) parser.parse(brLine);

if(JsonLine != null){

String revId = (String) JsonLine.get("reviewerID");

if(revId != null){

//fileWriter.append(String.valueOf(revId));

mahoutKeyValuefileWriter.append(String.valueOf(revId));

mahoutKeyValuefileWriter.append(COMMA\_DELIMITER);

//System.out.println(String.valueOf(revId));

try{

int value = mapReviewer.get(revId);

mahoutfileWriter.append(String.valueOf(value));

mahoutKeyValuefileWriter.append(String.valueOf(value));

//System.out.println(String.valueOf(value));

//mahoutKeyValuefileWriter.append(COMMA\_DELIMITER);

}catch(Exception e){

System.out.println(mapReviewer.get(revId));

}

}

// fileWriter.append(COMMA\_DELIMITER);

mahoutfileWriter.append(COMMA\_DELIMITER);

mahoutKeyValuefileWriter.append(COMMA\_DELIMITER);

String asin = (String) JsonLine.get("asin");

if(asin != null){

//fileWriter.append(String.valueOf(asin));

mahoutKeyValuefileWriter.append(String.valueOf(asin));

mahoutKeyValuefileWriter.append(COMMA\_DELIMITER);

//System.out.println(String.valueOf(asin));

try{

int valueP = mapProduct.get(asin);

mahoutfileWriter.append(String.valueOf(valueP));

mahoutKeyValuefileWriter.append(String.valueOf(valueP));

//System.out.println(String.valueOf(valueP));

// mahoutKeyValuefileWriter.append(COMMA\_DELIMITER);

}catch(Exception e){

System.out.println(mapProduct.get(asin));

}

}

// fileWriter.append(COMMA\_DELIMITER);

mahoutfileWriter.append(COMMA\_DELIMITER);

mahoutKeyValuefileWriter.append(COMMA\_DELIMITER);

String overall = JsonLine.get("overall").toString();

if(overall != null){

//fileWriter.append(String.valueOf(overall).replaceAll(",", ""));

mahoutfileWriter.append(String.valueOf(overall).replaceAll(",", ""));

mahoutKeyValuefileWriter.append(String.valueOf(overall).replaceAll(",", ""));

//System.out.println(String.valueOf(overall).replaceAll(",", ""));

}

// fileWriter.append(COMMA\_DELIMITER);

String time = (String) JsonLine.get("reviewTime");

if(time != null){

String temp = time.replaceAll(" ", ",");

// fileWriter.append(String.valueOf(temp).replaceAll(",,", ","));

//fileWriter.append(COMMA\_DELIMITER);

}else{

//fileWriter.append(COMMA\_DELIMITER);

//fileWriter.append(COMMA\_DELIMITER);

//fileWriter.append(COMMA\_DELIMITER);

}

JSONArray helpful = (JSONArray) JsonLine.get("helpful");

if(helpful != null){

for(Object c : helpful)

{

//System.out.println(c+"");

//fileWriter.append(String.valueOf(c.toString()));

//fileWriter.append(COMMA\_DELIMITER);

}

}else{

//fileWriter.append(COMMA\_DELIMITER);

//fileWriter.append(COMMA\_DELIMITER);

}

String reviewText = (String) JsonLine.get("reviewText");

String review;

if(reviewText.length() > 50){

review = reviewText.substring(0, 50);

}else{

review = reviewText;

}

if(review != null){

//String temp = review.replaceAll(" ", ",");

//fileWriter.append(review.replaceAll(",",""));//String.valueOf(temp).replaceAll(",,", ","));

//fileWriter.append(COMMA\_DELIMITER);

}else{

//fileWriter.append(COMMA\_DELIMITER);

}

//fileWriter.append(NEW\_LINE\_SEPARATOR);

mahoutfileWriter.append(NEW\_LINE\_SEPARATOR);

mahoutKeyValuefileWriter.append(NEW\_LINE\_SEPARATOR);

}

entryCount+=1;

}

//fileWriter.flush();

//fileWriter.close();

mahoutfileWriter.flush();

mahoutfileWriter.close();

mahoutKeyValuefileWriter.flush();

mahoutKeyValuefileWriter.close();

br1.close();

}catch(Exception e){

System.out.println(e);

}

System.out.println(entryCount);

}

}

2) To create product reviewData- (to find the product id, favorite rating, FavratingUser, totalReviewers )

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package productreview;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.conf.Configured;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.util.Tool;

import org.apache.hadoop.io.DoubleWritable;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import org.apache.hadoop.util.ToolRunner;

/\*\*

\*

\* @author mayur

\*/

public class ProductReview extends Configured implements Tool {

@Override

public int run(String[] args) throws Exception {

//throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

if (args.length != 2) {

System.out.printf("Two parameters are required for ProductReview- <input dir> <output dir>\n");

return -1;

}

Job job = new Job(getConf());

job.setJobName("Product Review");

job.setJarByClass(ProductReview.class);

FileInputFormat.setInputPaths(job, new Path(args[0]));

FileOutputFormat.setOutputPath(job, new Path(args[1]));

job.setMapperClass(PRMapper.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(ReviewWritable.class);

//job.setPartitionerClass(PRPartitioner.class);

//job.setSortComparatorClass(SecondarySortbyReview.class);

//job.setGroupingComparatorClass(PRGroupingByRating.class);

job.setCombinerClass(PRReducer.class);

job.setReducerClass(PRReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(ReviewWritable.class);

//job.setNumReduceTasks(5);

boolean success = job.waitForCompletion(true);

return success ? 0 : 1;

}

public static void main(String[] args) throws Exception {

// TODO code application logic here

int exitCode = ToolRunner.run(new Configuration(),

new ProductReview(), args);

System.exit(exitCode);

}

}

Mapper-

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package productreview;

import java.io.IOException;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

/\*\*

\*

\* @author mayur

\*/

public class PRMapper extends Mapper<LongWritable, Text, Text, ReviewWritable> {

// private ReviewWritable review = new ReviewWritable();

@Override

public void map(LongWritable key, Text value, Context context)

throws IOException, InterruptedException {

if (value.toString().length() > 0) {

String arrReviewAttributes[] = value.toString().split(",");

String productID = arrReviewAttributes[1].trim();

// System.out.println("rating: "+ arrReviewAttributes[2].trim());

Double rating =Double.parseDouble(arrReviewAttributes[2].trim());

//System.out.println("rating in double: "+rating);

int a = (int)Math.floor(rating + 0.5d);

//System.out.println("a in int: "+a);

String year = arrReviewAttributes[5].trim();

context.write(new Text(productID), new ReviewWritable(1, a, 1));

}

}

}

Reducer-

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package productreview;

import java.io.IOException;

import java.util.Map;

import java.util.TreeMap;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

/\*\*

\*

\* @author mayur

\*/

public class PRReducer extends Reducer<Text, ReviewWritable, Text, ReviewWritable > {

private ReviewWritable review = new ReviewWritable();

public void reduce(Text key, Iterable<ReviewWritable> values,

Context context

) throws IOException, InterruptedException {

Map<Integer, Long> FRwithNoOfUser = new TreeMap<>();

long sum = 0;

long ttlCnt = 0;

for(ReviewWritable rv : values){

int rating = rv.getFavoriteRating();

if(FRwithNoOfUser.containsKey(rating)){

long reviewers =(long) FRwithNoOfUser.get(rating);

//sum = (long) rv.getFrByNoOfReviewer() + reviewers;

FRwithNoOfUser.put(rating, (long) rv.getFrByNoOfReviewer() + reviewers);

}else{

//sum = (long) rv.getFrByNoOfReviewer();

FRwithNoOfUser.put(rating, (long) rv.getFrByNoOfReviewer());

}

if(ttlCnt == 0){

ttlCnt = rv.getTotalNoOfReviewer();

}

ttlCnt++;

}

long greatest= 0;

int gRating = 0;

for(Map.Entry<Integer, Long> entry : FRwithNoOfUser.entrySet()){

if(entry.getValue() >= greatest){

greatest = entry.getValue();

gRating = entry.getKey();

}

}

review.setFavoriteRating(gRating);

review.setFrByNoOfReviewer(greatest);

review.setTotalNoOfReviewer(ttlCnt);

context.write(key, review);

}

}

ReviewWritable-

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package productreview;

import java.io.DataInput;

import java.io.DataOutput;

import java.io.IOException;

import org.apache.hadoop.io.Writable;

import org.apache.hadoop.io.WritableComparable;

import org.apache.hadoop.io.WritableUtils;

/\*\*

\*

\* @author mayur

\*/

public class ReviewWritable implements Writable,WritableComparable<ReviewWritable> {

private int favoriteRating;

private long frByNoOfReviewer;

private long totalNoOfReviewer;

//private String percentageSoldperYear;

public ReviewWritable(long ttlReviewer, int favRating, long frByNOU){

this.favoriteRating = favRating;

this.frByNoOfReviewer = frByNOU;

this.totalNoOfReviewer = ttlReviewer;

}

ReviewWritable() {

// throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

}

@Override

public void write(DataOutput d) throws IOException {

//throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

WritableUtils.writeVInt(d, favoriteRating);//.writeString(d, favoriteRating);

WritableUtils.writeVLong(d, frByNoOfReviewer);

WritableUtils.writeVLong(d, totalNoOfReviewer);

}

@Override

public void readFields(DataInput di) throws IOException {

//throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

favoriteRating = WritableUtils.readVInt(di);

frByNoOfReviewer = WritableUtils.readVLong(di);

totalNoOfReviewer = WritableUtils.readVLong(di);

}

@Override

public String toString() {

return (new StringBuilder().append(favoriteRating).append("\t")

.append(frByNoOfReviewer).append("\t").append(totalNoOfReviewer)).toString();

}

public int getFavoriteRating() {

return favoriteRating;

}

public void setFavoriteRating(int favoriteRating) {

this.favoriteRating = favoriteRating;

}

public long getFrByNoOfReviewer() {

return frByNoOfReviewer;

}

public void setFrByNoOfReviewer(long frByNoOfReviewer) {

this.frByNoOfReviewer = frByNoOfReviewer;

}

public long getTotalNoOfReviewer() {

return totalNoOfReviewer;

}

public void setTotalNoOfReviewer(long totalNoOfReviewer) {

this.totalNoOfReviewer = totalNoOfReviewer;

}

@Override

public int compareTo(ReviewWritable o) {

//throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

return favoriteRating > o.favoriteRating ? +1 : favoriteRating < o.favoriteRating ? -1 : 0;

/\*if (0 == result) {

//result = totalNoOfReviewer.compareTo(objKeyPair.totalNoOfReviewer);

result = totalNoOfReviewer > o.totalNoOfReviewer ? +1: totalNoOfReviewer < o.totalNoOfReviewer ? -1 : 0;

}

return result; \*/

}

}

3) partition by rating and secondary sort by total count -

Main-

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package productreviewsortbyttlcnt; //ProductReviewSortByTtlCnt

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.conf.Configured;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import org.apache.hadoop.util.Tool;

import org.apache.hadoop.util.ToolRunner;

/\*\*

\*

\* @author mayur

\*/

public class ProductReviewSortByTtlCnt extends Configured implements Tool {

@Override

public int run(String[] args) throws Exception {

//throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

if (args.length != 2) {

System.out.printf("Two parameters are required for ProductReview- <input dir> <output dir>\n");

return -1;

}

Job job = new Job(getConf());

job.setJobName("Product Review");

job.setJarByClass(ProductReviewSortByTtlCnt.class);

FileInputFormat.setInputPaths(job, new Path(args[0]));

FileOutputFormat.setOutputPath(job, new Path(args[1]));

job.setMapperClass(sortMapper.class);

job.setMapOutputKeyClass(reviewWritable.class);

job.setMapOutputValueClass(NullWritable.class);

//job.setPartitionerClass(ratingPartitioner.class);

job.setSortComparatorClass(SecondarySortbyTtlCnt.class);

//job.setGroupingComparatorClass(PRGroupingByRating.class);

//job.setCombinerClass(sortReducer.class);

job.setReducerClass(sortReducer.class);

job.setOutputKeyClass(NullWritable.class);

job.setOutputValueClass(reviewWritable.class);

job.setNumReduceTasks(1);

boolean success = job.waitForCompletion(true);

return success ? 0 : 1;

}

public static void main(String[] args) throws Exception {

// TODO code application logic here

int exitCode = ToolRunner.run(new Configuration(),

new ProductReviewSortByTtlCnt(), args);

System.exit(exitCode);

}

}

Mapper-

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package productreviewsortbyttlcnt;

import java.io.IOException;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

/\*\*

\*

\* @author mayur

\*/

public class sortMapper extends Mapper<LongWritable, Text, reviewWritable, NullWritable> {

@Override

public void map(LongWritable key, Text value, Context context)

throws IOException, InterruptedException {

System.out.println(value.toString());

if (value.toString().length() > 0) {

String arrReviewAttributes[] = value.toString().split("\t");

String productID = arrReviewAttributes[0].trim();

// System.out.println("rating: "+ arrReviewAttributes[2].trim());

String rating =arrReviewAttributes[1].trim();

//System.out.println("rating in double: "+rating);

String rateCnt = arrReviewAttributes[2].trim();

String ttlCnt = arrReviewAttributes[3].trim();

context.write( new reviewWritable(productID, rating, rateCnt, ttlCnt), NullWritable.get());

}

}

}

Reducer-

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package productreviewsortbyttlcnt;

import java.io.IOException;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

/\*\*

\*

\* @author mayur

\*/

public class sortReducer extends Reducer<reviewWritable, NullWritable, NullWritable, reviewWritable > {

public void reduce(reviewWritable key, Iterable<NullWritable> values,

Context context

) throws IOException, InterruptedException {

for(NullWritable val : values){

context.write( NullWritable.get(), key);

}

}

}

Grouping-

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package productreviewsortbyttlcnt;

import org.apache.hadoop.io.WritableComparable;

import org.apache.hadoop.io.WritableComparator;

/\*\*

\*

\* @author mayur

\*/

public class PRGroupingByRating extends WritableComparator {

int a;

int b;

protected PRGroupingByRating(){

super(reviewWritable.class, true);

}

@Override

public int compare(WritableComparable w1, WritableComparable w2) {

reviewWritable key1 = (reviewWritable) w1;

reviewWritable key2 = (reviewWritable) w2;

a = Integer.parseInt(key1.getRating());

b = Integer.parseInt(key2.getRating());

return a > b ? +1 : a < b ? -1 : 0;

}

}

SecondarysortByTTlCnt-

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package productreviewsortbyttlcnt;

import org.apache.hadoop.io.WritableComparable;

import org.apache.hadoop.io.WritableComparator;

/\*\*

\*

\* @author mayur

\*/

public class SecondarySortbyTtlCnt extends WritableComparator{

protected SecondarySortbyTtlCnt() {

super(reviewWritable.class, true);

}

@Override

public int compare(WritableComparable w1, WritableComparable w2) {

reviewWritable key1 = (reviewWritable) w1;

reviewWritable key2 = (reviewWritable) w2;

int cmpResult = key1.getRating().compareTo(key2.getRating());

// int b = key2.getTtlCnt();

//int cmpResult = a > b ? +1 : a < b ? -1 : 0;

if (cmpResult == 0)// same deptNo

{

//int c = key1.getFavoriteRating();

//int d = key2.getFavoriteRating();

return key1.getTtlCnt().compareTo(key2.getTtlCnt());

//If the minus is taken out, the values will be in

//ascending order

}

return cmpResult;

}

}

Partitioner-

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package productreviewsortbyttlcnt;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Partitioner;

/\*\*

\*

\* @author mayur

\*/

public class ratingPartitioner extends Partitioner<reviewWritable, NullWritable>{

@Override

public int getPartition(reviewWritable key, NullWritable value, int numReduceTasks) {

//throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

return (Integer.parseInt(key.getRating()) % numReduceTasks);

}

}

reviewWritable-

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package productreviewsortbyttlcnt;

import java.io.DataInput;

import java.io.DataOutput;

import java.io.IOException;

import org.apache.hadoop.io.Writable;

import org.apache.hadoop.io.WritableComparable;

import org.apache.hadoop.io.WritableUtils;

/\*\*

\*

\* @author mayur

\*/

public class reviewWritable implements Writable,WritableComparable<reviewWritable> {

private String productID;

private String rating;

private String rateCnt;

private String ttlCnt;

public reviewWritable(String id, String rate, String rcnt, String ttl){

this.productID = id;

this.rating = rate;

this.rateCnt = rcnt;

this.ttlCnt = ttl;

}

reviewWritable(){}

@Override

public void write(DataOutput d) throws IOException {

//throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

WritableUtils.writeString(d, productID);

WritableUtils.writeString(d, rating);

WritableUtils.writeString(d, rateCnt);

WritableUtils.writeString(d, ttlCnt);

}

@Override

public void readFields(DataInput di) throws IOException {

//throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

productID = WritableUtils.readString(di);

rating = WritableUtils.readString(di);

rateCnt = WritableUtils.readString(di);

ttlCnt = WritableUtils.readString(di);

}

@Override

public String toString() {

return (new StringBuilder().append(productID).append("\t").append(rating).append("\t")

.append(rateCnt).append("\t").append(ttlCnt)).toString();

}

@Override

public int compareTo(reviewWritable o) {

// throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

int result = rating.compareTo(o.rating);

if (0 == result) {

result = ttlCnt.compareTo(o.ttlCnt);

//result = totalNoOfReviewer > o.totalNoOfReviewer ? +1: totalNoOfReviewer < o.totalNoOfReviewer ? -1 : 0;

}

return result;

}

public String getRating() {

return rating;

}

public void setRating(String rating) {

this.rating = rating;

}

public String getRateCnt() {

return rateCnt;

}

public void setRateCnt(String rateCnt) {

this.rateCnt = rateCnt;

}

public String getTtlCnt() {

return ttlCnt;

}

public void setTtlCnt(String ttlCnt) {

this.ttlCnt = ttlCnt;

}

public String getProductID() {

return productID;

}

public void setProductID(String productID) {

this.productID = productID;

}

}

4) top 50 reviewed product using chainin-

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package toprated50products;

import java.io.IOException;

import java.nio.ByteBuffer;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.io.WritableComparator;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

/\*\*

\*

\* @author mayur

\*/

public class TopRated50Products {

/\*\*

\* @param args the command line arguments

\*/

public static class Map1

extends Mapper<LongWritable, Text, Text, IntWritable> {

private final static IntWritable one = new IntWritable(1);

private Text word = new Text();

public void map(LongWritable key, Text value, Context context

) throws IOException, InterruptedException {

String row[] = value.toString().split(",");

String productid = row[1];

word.set(productid);

context.write(word, one);

}

}

public static class Reduce1

extends Reducer<Text, IntWritable, Text, IntWritable> {

private IntWritable result = new IntWritable();

public void reduce(Text key, Iterable<IntWritable> values,

Context context

) throws IOException, InterruptedException {

int sum = 0;

for (IntWritable val : values) {

sum += val.get();

}

result.set(sum);

context.write(key, result);

}

}

public static class Map2

extends Mapper<LongWritable, Text, IntWritable, Text> {

public void map(LongWritable key, Text value, Context context

) throws IOException, InterruptedException {

String[] proctid = (value.toString()).split("\t");

Text prodid = new Text(proctid[0]);

String newCount = proctid[1].trim();

try {

IntWritable count = new IntWritable(Integer.parseInt(newCount));

context.write(count, prodid);

} catch (Exception e) {

}

}

}

public static class Reduce2

extends Reducer<IntWritable, Text, Text, IntWritable> {

private IntWritable result = new IntWritable();

private static int count = 1;

public void reduce(IntWritable key, Iterable<Text> value,

Context context

) throws IOException, InterruptedException {

if(count <=50){

for (Text val : value) {

context.write(val, key);

}

}

this.count++;

}

}

public static void main(String[] args) throws Exception {

Configuration conf1 = new Configuration();

Job job1 = Job.getInstance(conf1, "50 top Rated Products 1");

job1.setJarByClass(TopRated50Products.class);

//job1.setSortComparatorClass(DecreasingComparator.class);

job1.setMapperClass(Map1.class);

job1.setMapOutputKeyClass(Text.class);

job1.setMapOutputValueClass(IntWritable.class);

//job1.setCombinerClass(Reduce1.class);

job1.setReducerClass(Reduce1.class);

job1.setOutputKeyClass(Text.class);

job1.setOutputValueClass(IntWritable.class);

FileInputFormat.addInputPath(job1, new Path(args[0]));

FileOutputFormat.setOutputPath(job1, new Path(args[1]));

boolean complete = job1.waitForCompletion(true);

Configuration conf2 = new Configuration();

Job job2 = Job.getInstance(conf2, "50 top Rated Products 2");

if (complete) {

job2.setJarByClass(TopRated50Products.class);

job2.setSortComparatorClass(DescendingIntComparator.class);

job2.setMapperClass(Map2.class);

job2.setMapOutputKeyClass(IntWritable.class);

job2.setMapOutputValueClass(Text.class);

//job2.setCombinerClass(Reduce2.class);

job2.setReducerClass(Reduce2.class);

job2.setOutputKeyClass(Text.class);

job2.setOutputValueClass(IntWritable.class);

FileInputFormat.addInputPath(job2, new Path(args[1]));

FileOutputFormat.setOutputPath(job2, new Path(args[2]));

System.exit(job2.waitForCompletion(true) ? 0 : 1);

}

}

public static class DescendingIntComparator extends WritableComparator {

public DescendingIntComparator() {

super(IntWritable.class);

}

@Override

public int compare(byte[] b1, int s1, int l1, byte[] b2, int s2, int l2) {

Integer v1 = ByteBuffer.wrap(b1, s1, l1).getInt();

Integer v2 = ByteBuffer.wrap(b2, s2, l2).getInt();

return (v1.compareTo(v2)) \* -1;

}

}

}

5) Product recommandation using Mapreduce-

Phase1 -

main-

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package productrecomr;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

/\*\*

\*

\* @author mayur

\*/

public class ProductRecoMR {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) throws Exception {

// TODO code application logic here

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "Product Reco first mapper");

job.setJarByClass(ProductRecoMR.class);

job.setMapperClass(recoMapper.class);

//job.setCombinerClass(recoReducer.class);

job.setNumReduceTasks(1);

job.setReducerClass(recoReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

FileInputFormat.addInputPath(job, new Path(args[0]));

FileOutputFormat.setOutputPath(job, new Path(args[1]));

System.exit(job.waitForCompletion(true) ? 0 : 1);

}

}

Mapper-

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package productrecomr;

import java.io.IOException;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

/\*\*

\*

\* @author mayur

\*/

public class recoMapper extends Mapper<Object, Text, Text, Text> {

@Override

public void map(Object key, Text value, Context context) throws IOException, InterruptedException{

String[] words = value.toString().split(",");

String userId = words[0];

String itemId = words[1];

context.write(new Text(userId), new Text(itemId));

}

}

Reducer-

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package productrecomr;

import java.io.IOException;

import java.text.ParseException;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Map;

import java.util.TreeMap;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

/\*\*

\*

\* @author mayur

\*/

public class recoReducer extends Reducer<Text, Text, Text, Text> {

//static int count = 0;

public void reduce(Text key, Iterable<Text> values,

Context context

) throws IOException, InterruptedException {

StringBuilder sb = new StringBuilder();

String prefix = "";

for(Text t: values){

// count++;

sb.append(prefix);

prefix = ",";

sb.append(t);

}

System.out.println("Key " + key);

System.out.println("items " + sb.toString() );

context.write(key, new Text(sb.toString()));

}

}

RecoWritable object-

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package productrecomr;

import java.io.DataInput;

import java.io.DataOutput;

import java.io.IOException;

import java.util.ArrayList;

import org.apache.hadoop.io.Writable;

/\*\*

\*

\* @author mayur

\*/

public class RecoWritable implements Writable {

private ArrayList<String> itemList = new ArrayList<String>();

@Override

public void write(DataOutput d) throws IOException {

}

@Override

public void readFields(DataInput di) throws IOException {

throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

}

}

Phase2-

Main -

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package productrecomr1;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

/\*\*

\*

\* @author mayur

\*/

public class ProductRecoMR1 {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) throws Exception {

// TODO code application logic here

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "Product Reco Second mapper");

job.setJarByClass(ProductRecoMR1.class);

job.setMapperClass(recoMapper1.class);

//job.setCombinerClass(recoReducer.class);

job.setNumReduceTasks(1);

job.setReducerClass(recoReducer1.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

FileInputFormat.addInputPath(job, new Path(args[0]));

FileOutputFormat.setOutputPath(job, new Path(args[1]));

System.exit(job.waitForCompletion(true) ? 0 : 1);

}

}

Mapper-

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package productrecomr1;

import java.io.IOException;

import java.util.HashMap;

import java.util.Iterator;

import java.util.Map;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.MapWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

/\*\*

\*

\* @author mayur

\*/

public class recoMapper1 extends Mapper<LongWritable, Text, Text, Text>{

public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException{

String[] words = value.toString().split("\\t");

String userId = words[0];

//System.out.println("user"+userId+"%");

//System.out.println("items"+words[1]+"%");

String[] items = words[1].split(",");

for(String item: items){

Map<String, Integer> map = new HashMap<String, Integer>();

for(String i: items){

if(map.containsKey(i.trim())){

map.put(i.trim(), map.get(i)+1);

}else{

map.put(i.trim(), 1);

}

}

StringBuilder sbKeyValue = new StringBuilder();

Iterator it = map.entrySet().iterator();

String prefix="";

while (it.hasNext()) {

Map.Entry pair = (Map.Entry)it.next();

sbKeyValue.append(prefix);

sbKeyValue.append(pair.getKey()).append(",").append(pair.getValue());

prefix = ";";

// System.out.println(pair.getKey() + " = " + pair.getValue());

it.remove(); // avoids a ConcurrentModificationException

}

context.write(new Text(item), new Text(sbKeyValue.toString()));

}

}

}

Reducer-

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package productrecomr1;

import java.io.IOException;

import java.util.Collections;

import java.util.HashMap;

import java.util.Iterator;

import java.util.LinkedHashMap;

import java.util.LinkedList;

import java.util.List;

import java.util.Map;

import org.apache.hadoop.io.BooleanWritable.Comparator;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

/\*\*

\*

\* @author mayur

\*/

public class recoReducer1 extends Reducer<Text, Text, Text , Text>{

public void reduce(Text key, Iterable<Text> values1,

Context context

) throws IOException, InterruptedException {

Map<String, Integer> finalMap = new HashMap<String, Integer>();

for(Text values : values1){

String[] keyValue = values.toString().split(";");

//System.out.println(values.toString());

for(String kv : keyValue){

String[] val = kv.split(",");

String item = val[0].trim();

Integer value = Integer.parseInt(val[1]);

if(finalMap.containsKey(item)){

int cnt = finalMap.get(item) + value;

finalMap.put(item, cnt );

}else{

finalMap.put(item, 1);

}

}

}

Map<String, Integer> sortedMap = sortByValue(finalMap);

Iterator it = sortedMap.entrySet().iterator();

int cnt= 0;

while(it.hasNext() && cnt < 5){

Map.Entry pair = (Map.Entry)it.next();

//System.out.println(pair.getKey().toString()+" "+pair.getValue());

context.write(key, new Text( pair.getKey().toString()));

cnt++;

}

}

/\*

public static <String, Integer extends Comparable<? super Integer>> Map<String, Integer> sortByValue(Map<String, Integer> unsortMap) {

List<Map.Entry<String, Integer>> list =

new LinkedList<Map.Entry<String, Integer>>(unsortMap.entrySet());

Collections.sort(list, new Comparator<Map.Entry<String, Integer>>() {

public int compare(Map.Entry<String, Integer> o1, Map.Entry<String, Integer> o2) {

return (o1.getValue()).compareTo(o2.getValue());

}

});

Map<String, Integer> result = new LinkedHashMap<String, Integer>();

for (Map.Entry<String, Integer> entry : list) {

result.put(entry.getKey(), entry.getValue());

}

return result;

}\*/

private static Map<String, Integer> sortByValue(Map<String, Integer> unsortMap) {

// 1. Convert Map to List of Map

List<Map.Entry<String, Integer>> list =

new LinkedList<Map.Entry<String, Integer>>(unsortMap.entrySet());

// 2. Sort list with Collections.sort(), provide a custom Comparator

// Try switch the o1 o2 position for a different order

Collections.sort(list, new java.util.Comparator<Map.Entry<String, Integer>>() {

public int compare(Map.Entry<String, Integer> o1,

Map.Entry<String, Integer> o2) {

return (o1.getValue()).compareTo(o2.getValue());

}

});

// 3. Loop the sorted list and put it into a new insertion order Map LinkedHashMap

Map<String, Integer> sortedMap = new LinkedHashMap<String, Integer>();

for (Map.Entry<String, Integer> entry : list) {

sortedMap.put(entry.getKey(), entry.getValue());

}

/\*

//classic iterator example

for (Iterator<Map.Entry<String, Integer>> it = list.iterator(); it.hasNext(); ) {

Map.Entry<String, Integer> entry = it.next();

sortedMap.put(entry.getKey(), entry.getValue());

}\*/

return sortedMap;

}

}

6) Product Recommandation by Mahout-

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package productreco;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import org.apache.mahout.cf.taste.common.TasteException;

import org.apache.mahout.cf.taste.common.NoSuchUserException;

import org.apache.mahout.cf.taste.impl.common.LongPrimitiveIterator;

import org.apache.mahout.cf.taste.impl.model.file.FileDataModel;

import org.apache.mahout.cf.taste.impl.neighborhood.NearestNUserNeighborhood;

import org.apache.mahout.cf.taste.impl.neighborhood.ThresholdUserNeighborhood;

import org.apache.mahout.cf.taste.impl.recommender.GenericItemBasedRecommender;

import org.apache.mahout.cf.taste.impl.recommender.GenericUserBasedRecommender;

import org.apache.mahout.cf.taste.impl.similarity.LogLikelihoodSimilarity;

import org.apache.mahout.cf.taste.impl.similarity.PearsonCorrelationSimilarity;

import org.apache.mahout.cf.taste.model.DataModel;

import org.apache.mahout.cf.taste.neighborhood.UserNeighborhood;

import org.apache.mahout.cf.taste.recommender.RecommendedItem;

import org.apache.mahout.cf.taste.recommender.UserBasedRecommender;

import org.apache.mahout.cf.taste.similarity.ItemSimilarity;

import org.apache.mahout.cf.taste.similarity.UserSimilarity;

/\*\*

\*

\* @author mayur

\*/

public class ProductReco {

static FileWriter recofileWriter = null;

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

//System.out.println(String.valueOf(args[0]));

//System.out.println(String.valueOf(args[1]));

//System.out.println(String.valueOf(args[2]));

try {

BufferedReader br = new BufferedReader(new FileReader("/home/mayur/Documents/ADBMS/Project/ElectronicsMahotKeyValue.csv")); //args[1]--"/home/mayur/Documents/ADBMS/Project/ElectronicsMahotKeyValue.csv"

BufferedReader br1 = new BufferedReader(new FileReader("/home/mayur/Documents/ADBMS/Project/ElectronicsMahotKeyValue.csv"));

String brLine = null;

List<String> rateSet = new ArrayList<String>();

List<String> rateSet1 = new ArrayList<String>();

HashMap<String, String> mapProduct = new HashMap<String, String>();

HashMap<String, List<String>> mapRating = new HashMap<String, List<String>>();

//compute average rating of product based on its occurance in recommendation

String test = "";

while ((brLine = br1.readLine()) != null) {

String[] parts = brLine.split(",");

if (!test.equals(parts[2])) {

rateSet = new ArrayList<>();

}

String productKey = parts[2];

String productRating = parts[4];

rateSet.add(productRating);

mapRating.put(productKey, rateSet);

if (!test.equals(productKey)) {

test = productKey;

// rateSet.clear();

}

}

//build set of product ID and its rating

while ((brLine = br.readLine()) != null) {

String[] parts = brLine.split(",");

String productKey = parts[3];

String productValue = parts[2];

mapProduct.put(productKey, productValue);

}

recofileWriter = new FileWriter("/home/mayur/Documents/ADBMS/Project/ElectronicsRecoOutput.csv"); //args[2]---"/home/mayur/Documents/ADBMS/Project/ElectronicsRecoOutput.csv"

// TODO code application logic here

DataModel model = new FileDataModel(new File("/home/mayur/Documents/ADBMS/Project/ElectronicsMahotInput.csv")); //args[0]---"/home/mayur/Documents/ADBMS/Project/ElectronicsMahotInput.csv"---"/home/mayur/Desktop/TestReco.csv"

// this for test

//DataModel model = new FileDataModel(new File("/home/mayur/Desktop/TestReco.csv"));

UserSimilarity similarity = new PearsonCorrelationSimilarity(model);

UserNeighborhood neighborhood = new NearestNUserNeighborhood(2, similarity, model);//home/mayur/Desktop/TestReco.csv

UserBasedRecommender recommender = new GenericUserBasedRecommender(model, neighborhood, similarity);

//this is for Itembased

//ItemSimilarity similarity = new LogLikelihoodSimilarity(model);

// GenericItemBasedRecommender recommender = new GenericItemBasedRecommender(model, similarity);

int x = 1;

outerloop:

for (LongPrimitiveIterator items = model.getItemIDs(); items.hasNext();) {

long itemId = items.nextLong();

List<RecommendedItem> recommendations = recommender.recommend(itemId, 4);

for (RecommendedItem recommendation : recommendations) {

//System.out.println("itemId - " + itemId + "recomanded ID " + recommendation.getItemID());

String productId = mapProduct.get(String.valueOf(itemId));

recofileWriter.append(productId + ",");

String productId1 = mapProduct.get(String.valueOf(recommendation.getItemID()));

recofileWriter.append(productId1 + ",");

recofileWriter.append(String.valueOf(recommendation.getValue()));

recofileWriter.append(",");

rateSet1 = mapRating.get(String.valueOf(productId1));

double lenght = rateSet1.size();

double sum = 0;

for (String r : rateSet1) {

double num = Double.parseDouble(r);

sum = sum + num;

}

sum = sum / lenght;

recofileWriter.append(String.valueOf(sum) + "\r\n");

// recofileWriter.append(String.valueOf(itemId));

//recofileWriter.append("/t");

// recofileWriter.append(String.valueOf(recommendation.getItemID()));

}

x++;

if (x < 0) {

break outerloop;

}

}

recofileWriter.flush();

recofileWriter.close();

} catch (Exception e) {

System.out.println(e);

}

System.out.println("DONE");

}

}

7) HBASE commands-

create 'Producttest1','userId','prodId','rating','Date’,'Help’,'Review’

hbase> hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -Dimporttsv.separator=, -Dimporttsv.columns="HBASE\_ROW\_KEY,userId,prodId,rating,Date:M,Date:D,Date,Y,Help:a,Help:b,Review:a,Review:b" Producttest1 hdfs://localhost:9000/mayur/ProductReview/InputFiles/ElectronicsShortTemp.csv

scan 'Producttest1', { COLUMNS => 'DateY', LIMIT => 4, FILTER => "ValueFilter( =, 'binaryprefix:2013' )" }