**CODE DOCUMENTATION**

1. Data Enhancement and Data Cleaning:

We used following jobs to enhance the number of features by combining results returned from Amazon **Product Advertising API** and **OMDB API**.

AmazonToOmdbCrawl.java

It maps the title from Amazon Product advertising API and fetches IMDB data for corresponding title from OMDB API.

public class AmazonToOmdbCrawl {

public static class Map extends Mapper<LongWritable, Text, Text, Text> {

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

throws IOException, InterruptedException {

/\*

\* Here the input is a line containing ID Title Score we are

\* canonicalizing the title field and sending it as key to reducer

\*/

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

String line = null;

try {

line = id\_title[1].trim();

/\*line = line.replaceAll("\\<.\*?>", "");

line = line.replaceAll("\\(.\*?\\) ?", "").trim();

line = line.replaceAll("\\[.\*?\\] ?", "").trim();

line = line.replaceAll("^[\\//\\-\\+\\.\\,\\\*]\*", "").trim();\*/

//line = line.replaceAll("^\\s+", "");

if (!line.isEmpty())

context.write(new Text(line), value);

} catch (Exception e) {

}

}

}

public static class Reduce extends Reducer<Text, Text, Text, NullWritable> {

/\*

\* Here the title recieved is sent to OMDB, which results a JSON

\* response the response is finally stored in a file, thus we get part-r

\* files of omdb responses.

\*/

@SuppressWarnings("unchecked")

public void reduce(Text key, Iterable<Text> values, Context context)

throws IOException, InterruptedException {

String title = key.toString().trim();

String amazonTitle = title.replace("\"", "").trim();

//title = title.replaceAll("^[\\-\\+\\.\\,]", " ").trim();

title = title.replaceAll("[VHS]", " ").trim();

//title = title.replace("\"", "");

title = title.replaceAll("%", "%25").trim();

title = title.replaceAll("/ +/g"," ").trim();

JSONParser jsonParser = new JSONParser();

JSONObject jsonObject = null;

String valueStr[] = null;

for (Text value : values) {

valueStr = value.toString().split("\t");

break;

}

for (String l : title.split("/", 2)) {

System.out.println("request............."+l);

String r = getResponse(l.trim());

System.out.println("response............"+r);

try {

if (r != null)

jsonObject = (JSONObject) jsonParser

.parse(new String(r));

} catch (Exception e) {

e.printStackTrace();

}

if (jsonObject != null

&& !jsonObject.get("Response").equals("False")) {

jsonObject.put("AmazonTitle", amazonTitle);

if (valueStr != null && valueStr.length > 1) {

jsonObject.put("AmazonID", valueStr[0]);

jsonObject.put("AmazonAvgScore", valueStr[2]);

} else {

jsonObject.put("AmazonID", "");

jsonObject.put("AmazonAvgScore", "");

}

//System.out.println("json......"+jsonObject);

context.write(new Text(jsonObject.toJSONString()),

NullWritable.get());}

}

}

/\*

\* This function takes care of handling the http connection and

\* resturning Json response from the server.

\*/

public static String getResponse(String name)

throws InterruptedException, IOException {

boolean notSuccess = true;

URL url = null;

BufferedReader rd = null;

while (notSuccess) {

try {

url = new URL("http://www.omdbapi.com/?t="+ name.replaceAll(" ", "+")+ "&y=&plot=short&r=json");

rd = new BufferedReader(new InputStreamReader(url.openConnection().getInputStream()));

notSuccess = false;

} catch (Exception e) {

System.out.println(e);

Thread.sleep(1000);

}

}

String result = rd.readLine();

rd.close();

return result;

}

}

public static void main(String args[]) throws IOException,

ClassNotFoundException, InterruptedException {

Configuration conf = new Configuration();

@SuppressWarnings("deprecation")

Job job = new Job(conf, "Movie Filter");

job.setJarByClass(AmazonToOmdbCrawl.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

job.setNumReduceTasks(10);

job.setMapperClass(Map.class);

job.setReducerClass(Reduce.class);

job.setInputFormatClass(TextInputFormat.class);

job.setOutputFormatClass(TextOutputFormat.class);

//job.setOutputFormatClass(TextOutputFormat.class);

FileInputFormat.addInputPath(job,new Path(

"C:\\Users\\inigam\\Documents\\Semester\\Amazon-Movie-Review-Analysis-master\\output\\movieOmdbInput.txt"));

FileOutputFormat.setOutputPath(job, new Path(args[0]));

job.waitForCompletion(true);

}

}

AmazonTitleAPICrawl.java

We extracted the productid field from Amazon dataset and Extracted title from Amazon Product Advertizing API

public class AmazonTitleAPICrawl {

/\*

\* Here we are extracting the productid field and review score field and sending it as

\* key value pair to reducer.

\*

\*/

public static class MapperJob extends Mapper<LongWritable, Text, Text, DoubleWritable> {

private static final String MATCHING\_PRODUCT="product/productId";

private static final String MATCHING\_SCORE="review/score";

private static Text word = new Text();

private static DoubleWritable one = new DoubleWritable();

protected void map(LongWritable fileOffset, Text lineContents, Context context) throws IOException, InterruptedException {

String line = lineContents.toString();

if(line.toLowerCase().contains(MATCHING\_PRODUCT.toLowerCase())){

String[] contentsOfString = line.split(":");

word.set(contentsOfString[1].trim());

}

if(line.toLowerCase().contains(MATCHING\_SCORE.toLowerCase())){

String[] contents = line.split(":");

one.set(Double.parseDouble(contents[1].trim()));

}

context.write(word, one);

}

}

public static class ReviewReducer extends Reducer<Text, DoubleWritable, Text, Text> {

static int count=0;

protected void reduce(Text MovieId, Iterable<DoubleWritable> scores,

Context context) throws IOException, InterruptedException {

double totalScore = 0;

int countForAverage =0;

for (DoubleWritable score : scores) {

countForAverage++;

totalScore += score.get();

}

ProductSearch p= new ProductSearch();

double averageScore=totalScore/countForAverage;

String title = null;

System.out.println("count.................."+count);

if(count<1){

try {

title = p.getMovieDetails(MovieId.toString());

} catch (SAXException e) {

//TODO Auto-generated catch block

e.printStackTrace();

} catch (ParserConfigurationException e) {

//TODO Auto-generated catch block

e.printStackTrace();

} catch (ParseException e) {

//TODO Auto-generated catch block

e.printStackTrace();

} catch (InterruptedException e) {

//TODO Auto-generated catch block

e.printStackTrace();

}

if(title!=null){

String outputString = (String) title +"\t"+ averageScore;

context.write(MovieId,new Text(outputString))

}

count++;

}else{

count=0;

Thread.sleep(1000);

}

}

}

public static void main(String args[]) throws IOException, ClassNotFoundException, InterruptedException{

Configuration conf = new Configuration();

GenericOptionsParser parser = new GenericOptionsParser(conf, args);

args = parser.getRemainingArgs();

Job job = new Job(conf, "Title Crawling");

job.setJarByClass(AmazonTitleAPICrawl.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(DoubleWritable.class);

job.setNumReduceTasks(10);

job.setMapperClass(MapperJob.class);

job.setReducerClass(ReviewReducer.class);

job.setInputFormatClass(TextInputFormat.class);

job.setOutputFormatClass(TextOutputFormat.class);

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

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

job.waitForCompletion(true);

}

}

AmazonInputRecordFetcher

public class AmazonInputRecordFetcher {

public static void main(String arg[]) {

//TODO Auto-generated constructor stub

try {

FileReader fileReader = new FileReader("C://Users//inigam//Documents//Semester//Amazon-Movie-Review-Analysis-master//input//movies.txt");

BufferedReader buf =new BufferedReader(fileReader);

String line;

int count=0;

while(( line=buf.readLine())!=null){

count++;

if(line.contains("B0001XLXGQ")){

System.out.println("got it.........."+line+" "+count);

break;

}

}

} catch (FileNotFoundException e) {

//TODO Auto-generated catch block

e.printStackTrace();

} catch (IOException e) {

//TODO Auto-generated catch block

e.printStackTrace();

}

System.out.println("completed");

}

}

ProductSearch.java

public class ProductSearch {

/\*

\* Your AWS Access Key ID, as taken from the AWS Your Account page.

\*/

private static final String AWS\_ACCESS\_KEY\_ID = "AKIAIUC2TDK4O7F6IPZA";

/\*

\* Your AWS Secret Key corresponding to the above ID, as taken from the AWS

\* Your Account page.

\*/

private static final String AWS\_SECRET\_KEY = "DRk9BoIlStxz3DxSu7zTjG1Zn7lI0UDIssrH9/Bi";

/\*

\* Use the end-point according to the region you are interested in.

\*/

private static final String ENDPOINT = "webservices.amazon.com";

SignedRequestHelper helper;

public ProductSearch(){

/\*

\* Set up the signed requests helper.

\*/

try {

helper = SignedRequestHelper.getInstance(ENDPOINT, AWS\_ACCESS\_KEY\_ID, AWS\_SECRET\_KEY);

} catch (Exception e) {

e.printStackTrace();

}

}

public String getMovieDetails(String prodids) throws IOException, SAXException, ParserConfigurationException, ParseException, InterruptedException {

String requestUrl = null;

Map<String, String> params = new HashMap<String, String>();

params.put("Service", "AWSECommerceService");

params.put("Operation", "ItemLookup");

params.put("AWSAccessKeyId", "AKIAIUC2TDK4O7F6IPZA");

params.put("AssociateTag", "amazonreviewt-20");

params.put("ItemId", prodids);

params.put("IdType", "ASIN");

params.put("ResponseGroup", "ItemAttributes");

requestUrl = helper.sign(params);

/\* System.out.println("Signed URL: \"" + requestUrl + "\"");

\*/

DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();

DocumentBuilder db = dbf.newDocumentBuilder();

Document doc = null;

boolean notSuccess=true;

while(notSuccess){

try{

doc = db.parse(new URL(requestUrl).openStream());

System.out.println(doc);

notSuccess=false;

break;

}

catch(Exception e){

System.out.println(e);

}

}

NodeList node=null;

try{

node = doc.getElementsByTagName("Title");

return node.item(0).getTextContent();

}

catch(Exception e){

}

return null;

}

}

SignedRequestHelper.java

public class SignedRequestHelper {

/\*\*

\* All strings are handled as UTF-8

\*/

private static final String UTF8\_CHARSET = "UTF-8";

/\*\*

\* The HMAC algorithm required by Amazon

\*/

private static final String HMAC\_SHA256\_ALGORITHM = "HmacSHA256";

/\*\*

\* This is the URI for the service, don't change unless you really know

\* what you're doing.

\*/

private static final String REQUEST\_URI = "/onca/xml";

/\*\*

\* The sample uses HTTP GET to fetch the response. If you changed the sample

\* to use HTTP POST instead, change the value below to POST.

\*/

private static final String REQUEST\_METHOD = "GET";

private String endpoint = null;

private String awsAccessKeyId = null;

private String awsSecretKey = null;

private SecretKeySpec secretKeySpec = null;

private Mac mac = null;

/\*\*

\* You must provide the three values below to initialize the helper.

\*

\* @param endpoint Destination for the requests.

\* @param awsAccessKeyId Your AWS Access Key ID

\* @param awsSecretKey Your AWS Secret Key

\*/

public static SignedRequestHelper getInstance(

String endpoint,

String awsAccessKeyId,

String awsSecretKey

) throws IllegalArgumentException, UnsupportedEncodingException, NoSuchAlgorithmException, InvalidKeyException

{

if (null == endpoint || endpoint.length() == 0)

{ throw new IllegalArgumentException("endpoint is null or empty"); }

if (null == awsAccessKeyId || awsAccessKeyId.length() == 0)

{ throw new IllegalArgumentException("awsAccessKeyId is null or empty"); }

if (null == awsSecretKey || awsSecretKey.length() == 0)

{ throw new IllegalArgumentException("awsSecretKey is null or empty"); }

SignedRequestHelper instance = new SignedRequestHelper();

instance.endpoint = endpoint.toLowerCase();

instance.awsAccessKeyId = awsAccessKeyId;

instance.awsSecretKey = awsSecretKey;

byte[] secretyKeyBytes = instance.awsSecretKey.getBytes(UTF8\_CHARSET);

instance.secretKeySpec = new SecretKeySpec(secretyKeyBytes, HMAC\_SHA256\_ALGORITHM);

instance.mac = Mac.getInstance(HMAC\_SHA256\_ALGORITHM);

instance.mac.init(instance.secretKeySpec);

return instance;

}

/\*\*

\* The construct is private since we'd rather use getInstance()

\*/

private SignedRequestHelper() {}

/\*\*

\* This method signs requests in hashmap form. It returns a URL that should

\* be used to fetch the response. The URL returned should not be modified in

\* any way, doing so will invalidate the signature and Amazon will reject

\* the request.

\*/

public String sign(Map<String, String> params) {

//Let's add the AWSAccessKeyId and Timestamp parameters to the request.

params.put("AWSAccessKeyId", this.awsAccessKeyId);

params.put("Timestamp", this.timestamp());

//The parameters need to be processed in lexicographical order, so we'll

//use a TreeMap implementation for that.

SortedMap<String, String> sortedParamMap = new TreeMap<String, String>(params);

//get the canonical form the query string

String canonicalQS = this.canonicalize(sortedParamMap);

//create the string upon which the signature is calculated

String toSign =

REQUEST\_METHOD + "\n"

+ this.endpoint + "\n"

+ REQUEST\_URI + "\n"

+ canonicalQS;

//get the signature

String hmac = this.hmac(toSign);

String sig = this.percentEncodeRfc3986(hmac);

//construct the URL

String url =

"http://" + this.endpoint + REQUEST\_URI + "?" + canonicalQS + "&Signature=" + sig;

return url;

}

/\*\*

\* This method signs requests in query-string form. It returns a URL that

\* should be used to fetch the response. The URL returned should not be

\* modified in any way, doing so will invalidate the signature and Amazon

\* will reject the request.

\*/

public String sign(String queryString) {

//let's break the query string into it's constituent name-value pairs

Map<String, String> params = this.createParameterMap(queryString);

//then we can sign the request as before

return this.sign(params);

}

/\*\*

\* Compute the HMAC.

\*

\* @param stringToSign String to compute the HMAC over.

\* @return base64-encoded hmac value.

\*/

private String hmac(String stringToSign) {

String signature = null;

byte[] data;

byte[] rawHmac;

try {

data = stringToSign.getBytes(UTF8\_CHARSET);

rawHmac = mac.doFinal(data);

Base64 encoder = new Base64();

signature = new String(encoder.encode(rawHmac));

} catch (UnsupportedEncodingException e) {

throw new RuntimeException(UTF8\_CHARSET + " is unsupported!", e);

}

return signature;

}

/\*\*

\* Generate a ISO-8601 format timestamp as required by Amazon.

\*

\* @return ISO-8601 format timestamp.

\*/

private String timestamp() {

String timestamp = null;

Calendar cal = Calendar.getInstance();

DateFormat dfm = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss'Z'");

dfm.setTimeZone(TimeZone.getTimeZone("GMT"));

timestamp = dfm.format(cal.getTime());

return timestamp;

}

/\*\*

\* Canonicalize the query string as required by Amazon.

\*

\* @param sortedParamMap Parameter name-value pairs in lexicographical order.

\* @return Canonical form of query string.

\*/

private String canonicalize(SortedMap<String, String> sortedParamMap) {

if (sortedParamMap.isEmpty()) {

return "";

}

StringBuffer buffer = new StringBuffer();

Iterator<Map.Entry<String, String>> iter = sortedParamMap.entrySet().iterator();

while (iter.hasNext()) {

Map.Entry<String, String> kvpair = iter.next();

buffer.append(percentEncodeRfc3986(kvpair.getKey()));

buffer.append("=");

buffer.append(percentEncodeRfc3986(kvpair.getValue()));

if (iter.hasNext()) {

buffer.append("&");

}

}

String cannoical = buffer.toString();

return cannoical;

}

/\*\*

\* Percent-encode values according the RFC 3986. The built-in Java

\* URLEncoder does not encode according to the RFC, so we make the

\* extra replacements.

\*

\* @param s decoded string

\* @return encoded string per RFC 3986

\*/

private String percentEncodeRfc3986(String s) {

String out;

try {

out = URLEncoder.encode(s, UTF8\_CHARSET)

.replace("+", "%20")

.replace("\*", "%2A")

.replace("%7E", "~");

} catch (UnsupportedEncodingException e) {

out = s;

}

return out;

}

/\*\*

\* Takes a query string, separates the constituent name-value pairs

\* and stores them in a hashmap.

\*

\* @param queryString

\* @return

\*/

private Map<String, String> createParameterMap(String queryString) {

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

String[] pairs = queryString.split("&");

for (String pair: pairs) {

if (pair.length() < 1) {

continue;

}

String[] tokens = pair.split("=",2);

for(int j=0; j<tokens.length; j++)

{

try {

tokens[j] = URLDecoder.decode(tokens[j], UTF8\_CHARSET);

} catch (UnsupportedEncodingException e) {

}

}

switch (tokens.length) {

case 1: {

if (pair.charAt(0) == '=') {

map.put("", tokens[0]);

} else {

map.put(tokens[0], "");

}

break;

}

case 2: {

map.put(tokens[0], tokens[1]);

break;

}

}

}

return map;

}

}

AmazonOutputJSONToCSV.java

public class AmazonOutputJSONToCSV {

public static void main(String args[]) {

//TODO Auto-generated method stub

PrintWriter writer = null;

try (BufferedReader br = new BufferedReader(new FileReader("C:\\Users\\inigam\\Documents\\Semester\\Amazon-Movie-Review-Analysis-master\\out\\disk1out.txt"))) {

writer = new PrintWriter("C:\\Users\\inigam\\Documents\\Semester\\Amazon-Movie-Review-Analysis-master\\out\\disk1out.csv");

String line;

JSONParser parser = new JSONParser();

int count =0;

//

/\*\*

\* Released":"31 Jul 2009",

"Metascore":"N\/A",

"imdbID":"tt1286871","

Plot":"A documentary about the first woman to join the Indian Police Service.",

"+":"Megan Doneman","

Title":"Kiran Bedi: Yes Madam, Sir",

"Actors":"Kiran Bedi, Helen Mirren",

"imdbRating":"8.9","

imdbVotes":"85",

"Response":"True",

"AmazonAvgScore":"2.925925925925926",

"Runtime":"95 min",

"Type":"movie"

,"Awards":"2 wins.","

Year":"2008",

"Language":"Hindi,English",

"Rated":"N\/A",

"Poster":"http:\/\/ia.media-imdb.com\/images\/M\/MV5BMTUwODAyMzk3MV5BMl5BanBnXkFtZTgwNjA4MjA2MDE@.\_V1\_SX300.jpg",

"Country":"Australia, India",

"AmazonTitle":"Yes, Madam!",

"AmazonID":"B000009HII","

Genre":"Documentary",

"Writer":"Megan Doneman, Jeneffa Soldatic (story)"

}

\*/

Pattern p = Pattern.compile("-?\\d+");

Pattern oscarRegEx = Pattern.compile("(^['Nominated']+\\s+['for']+)+\\s\*\\d+\\s\*['Oscar']+");

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

JSONObject obj = (JSONObject) parser.parse(line);

long imdbVote= 0;

try{

imdbVote=Long.parseLong(((String) obj.get("imdbVotes")).replaceAll(",",""));

}catch(NumberFormatException ex){

imdbVote= 0;

}

Matcher m = p.matcher((String) obj.get("Runtime"));

long runtime=0;

while(m.find()){

runtime= Long.parseLong(m.group());

break;

}

String oscarAwards = (String) obj.get("Awards");

int oscarNominated=0;

int oscarWinner=0;

int otherAwards=0;

String beforeOscar=null;

String afterOscar=null;

try{

if((oscarAwards.toLowerCase()).contains("oscar")){

beforeOscar= (oscarAwards.toLowerCase()).split("oscar")[0];

afterOscar= (oscarAwards.toLowerCase()).split("oscar")[1];

}else{

afterOscar=oscarAwards;

}

if(beforeOscar!=null){

if(beforeOscar.toLowerCase().contains("nominat")){

beforeOscar=beforeOscar.replaceAll("[^0-9]+"," ");

oscarNominated=Integer.parseInt(beforeOscar.trim());

}else{

oscarWinner=Integer.parseInt(beforeOscar.replaceAll("[^0-9]+"," ").trim());

}

}

if(afterOscar!=null){

afterOscar = afterOscar.replaceAll("[^0-9]+"," ");

String[] intArr= afterOscar.trim().split(" ");

int sum =0;

for(String val:intArr){

sum =sum+Integer.parseInt(val);

}

otherAwards=sum;

sum=0;

//System.out.println(afterOscar.trim().split(" ")[0]+afterOscar.trim().split(" ")[1]);

}

}catch(Exception e){

}

String result = ((String) obj.get("Released")).replaceAll(",","\t")+","+((String) obj.get("imdbID")).replaceAll(",", "\t")+","+((String) obj.get("Director")).replaceAll(",","\t")+","+((String) obj.get("Title")).replaceAll(",","\t")+","+((String) obj.get("Actors")).replaceAll(",","\t")+","+((String) obj.get("imdbRating")).replaceAll(",","\t")+","+(imdbVote)+","+((String) obj.get("AmazonAvgScore")).replaceAll(",","\t")+","+runtime+","+((String) obj.get("Awards")).replaceAll(",","\t")+","+((String) obj.get("Year")).replaceAll(",","\t")+","+((String) obj.get("Language")).replaceAll(",","\t")+","+((String) obj.get("Country")).replaceAll(",","\t")+","+((String) obj.get("AmazonTitle")).replaceAll(",","\t")+","+((String) obj.get("AmazonID")).replaceAll(",","\t")+","+((String) obj.get("Genre")).replaceAll(",","\t")+","+((String) obj.get("Writer")).replaceAll(",","\t")+","+((String) obj.get("Poster")).replaceAll(",","")+","+oscarWinner+","+oscarNominated+","+otherAwards;

writer.println(result);

}

} catch (FileNotFoundException e) {

//TODO Auto-generated catch block

e.printStackTrace();

} catch (IOException e) {

//TODO Auto-generated catch block

e.printStackTrace();

} catch (ParseException e) {

//TODO Auto-generated catch block

e.printStackTrace();

}finally{

writer.close();

}}}

1. Data Analysis using PIG (PIG Scripts):

Load Data into HDFS:

movies = LOAD 'hdfs://Hadoop-Master:9000/user/hduser/mergedinput.txt' USING PigStorage(',') AS (Released:chararray, imdbID:chararray, Director:chararray, Title:chararray, Actors:chararray, imdbRating:float, imdbVote:int, AmazonAvgScore:float, runTime:int, Awards:chararray, Year:int, Language:chararray, Country:chararray, AmazonTitle:chararray, AmazonID:chararray, Genre:chararray, Writer:chararray, Poster:chararray, OscarWinner:int, OscarNominated:int, OtherAwards:int);

Top 5 movies by genre:

-- Comedy

filterComedy = filter movies by (Genre matches '.\*(comedy|Comedy|COMEDY).\*');

**-- Won oscars**

FilterWonOscarByComedy = filter filterComedy by (Awards matches '.\*(won|WON|Won).\*' and Awards matches '.\*(oscar|Oscars|oscars|Oscar|OSCAR|OSCARS).\*');

**-- Removing columns not required**

removingColumns = foreach FilterWonOscarByComedy generate Released,imdbID,Director,Title,Actors,imdbRating,imdbVote,runTime,Awards,Year,Language,Genre,Writer,OscarWinner,OscarNominated,OtherAwards;

**--Fetching Distinct records**

distinctComedyMovies = Distinct removingColumns;

**-- Order it by IMDB rating**

orderComedyMoviesImdbRating = order distinctComedyMovies by imdbRating desc;

**-- Top 10 comedy movies by IMDB rating**

TopTenComedyMovies = LIMIT orderComedyMoviesImdbRating 10;

-- Horror

filterHorror = filter movies by (Genre matches '.\*(horror|Horror|HORROR).\*');

**-- Won oscars**

FilterWonOscarByHorror = filter filterHorror by (Awards matches '.\*(won|WON|Won).\*' and Awards matches '.\*(oscar|Oscars|oscars|Oscar|OSCAR|OSCARS).\*');

**-- Removing columns not required**

removingColumns = foreach FilterWonOscarByHorror generate Released,imdbID,Director,Title,Actors,imdbRating,imdbVote,runTime,Awards,Year,Language,Genre,Writer,OscarWinner,OscarNominated,OtherAwards;

**--Fetching Distinct records**

distinctHorrorMovies = Distinct removingColumns;

**-- Order it by IMDB rating**

orderHorrorMoviesImdbRating = order distinctHorrorMovies by imdbRating desc;

**-- Top 10 Horror movies by IMDB rating**

TopTenHorrorMovies = LIMIT orderHorrorMoviesImdbRating 10;

**--storing data:**

STORE TopTenHorrorMovies into 'TopTenHorrorMoviesNewData' using PigStorage(',');

-- Romance

filterRomance = filter movies by (Genre matches '.\*(romance|Romance|ROMANCE).\*');

**-- Won oscars**

FilterWonOscarByRomance = filter filterRomance by (Awards matches '.\*(won|WON|Won).\*' and Awards matches '.\*(oscar|Oscars|oscars|Oscar|OSCAR|OSCARS).\*');

**-- Removing columns not required**

removingColumns = foreach FilterWonOscarByRomance generate Released,imdbID,Director,Title,Actors,imdbRating,imdbVote,runTime,Awards,Year,Language,Genre,Writer,OscarWinner,OscarNominated,OtherAwards;

**--Fetching Distinct records**

distinctRomanticMovies = Distinct removingColumns;

**-- Order it by IMDB rating**

orderRomanticMoviesImdbRating = order distinctRomanticMovies by imdbRating desc;

**-- Top 10 Romantic movies by IMDB rating**

TopTenRomanticMovies = LIMIT orderRomanticMoviesImdbRating 10;

**--storing data:**

STORE TopTenRomanticMovies into 'TopTenRomanticMoviesNewData' using PigStorage(',');

-- Thriller Mystery

filterMysThrill = filter movies by (Genre matches '.\*(Thriller|thriller|THRILLER|Mystery|mystery|MYSTERY).\*');

**-- Won oscars**

FilterWonOscarByMysThrill = filter filterMysThrill by (Awards matches '.\*(won|WON|Won).\*' and Awards matches '.\*(oscar|Oscars|oscars|Oscar|OSCAR|OSCARS).\*');

**-- Removing columns not required**

removingColumns = foreach FilterWonOscarByMysThrill generate Released,imdbID,Director,Title,Actors,imdbRating,imdbVote,runTime,Awards,Year,Language,Genre,Writer,OscarWinner,OscarNominated,OtherAwards;

**--Fetching Distinct records**

distinctMysThrillMovies = Distinct removingColumns;

**-- Order it by IMDB rating**

orderMysThrillMoviesImdbRating = order distinctMysThrillMovies by imdbRating desc;

**-- Top 10 MysThrill movies by IMDB rating**

TopTenMysThrillMovies = LIMIT orderMysThrillMoviesImdbRating 10;

**--storing data:**

STORE TopTenMysThrillMovies into 'TopTenMysThrillMoviesNewData' using PigStorage(',');

-- Sci-Fi

filterSciFi = filter movies by (Genre matches '.\*(Sci|sci|Sci-Fi).\*');

**-- Won oscars**

FilterWonOscarBySciFi = filter filterSciFi by (Awards matches '.\*(won|WON|Won).\*' and Awards matches '.\*(oscar|Oscars|oscars|Oscar|OSCAR|OSCARS).\*');

**-- Removing columns not required**

removingColumns = foreach FilterWonOscarBySciFi generate Released,imdbID,Director,Title,Actors,imdbRating,imdbVote,runTime,Awards,Year,Language,Genre,Writer,OscarWinner,OscarNominated,OtherAwards;

**--Fetching Distinct records**

distinctSciFiMovies = Distinct removingColumns;

**-- Order it by IMDB rating**

orderSciFiMoviesImdbRating = order distinctSciFiMovies by imdbRating desc;

**-- Top 10 SciFi movies by IMDB rating**

TopTenSciFiMovies = LIMIT orderSciFiMoviesImdbRating 10;

**--storing data:**

STORE TopTenSciFiMovies into 'TopTenSciFiMoviesNewData' using PigStorage(',');

Analyzing the frequency of movies and average duration of movies in a gap of 10 years:

-(2010-Present):

filterMovies\_2010 = filter movies by (Year>2010 and Year<=2020);

groupMovies\_2010 = GROUP filterMovies\_2010 ALL;

averageDuration\_2010 = FOREACH groupMovies\_2010 GENERATE AVG(filterMovies\_2010.runTime);

totalCount\_2010 = FOREACH groupMovies\_2010 GENERATE COUNT(filterMovies\_2010);

dump averageDuration\_2010;

dump totalCount\_2010;

-(2000-2010):

filterMovies\_2000 = filter movies by (Year>2000 and Year<=2010);

groupMovies\_2000 = GROUP filterMovies\_2000 ALL;

averageDuration\_2000 = FOREACH groupMovies\_2000 GENERATE AVG(filterMovies\_2000.runTime);

totalCount\_2000 = FOREACH groupMovies\_2000 GENERATE COUNT(filterMovies\_2000);

dump averageDuration\_2000;

dump totalCount\_2000;

-(1990-2000):

filterMovies\_1990 = filter movies by (Year>1990 and Year<=2000);

groupMovies\_1990 = GROUP filterMovies\_1990 ALL;

averageDuration\_1990 = FOREACH groupMovies\_1990 GENERATE AVG(filterMovies\_1990.runTime);

totalCount\_1990 = FOREACH groupMovies\_1990 GENERATE COUNT(filterMovies\_1990);

dump averageDuration\_1990;

dump totalCount\_1990;

-(1980-1990):

filterMovies\_1980 = filter movies by (Year>1980 and Year<=1990);

groupMovies\_1980 = GROUP filterMovies\_1980 ALL;

averageDuration\_1980 = FOREACH groupMovies\_1980 GENERATE AVG(filterMovies\_1980.runTime);

totalCount\_1980 = FOREACH groupMovies\_1980 GENERATE COUNT(filterMovies\_1980);

dump averageDuration\_1980;

dump totalCount\_1980;

-(1970-1980):

filterMovies\_1970 = filter movies by (Year>1970 and Year<=1980);

groupMovies\_1970 = GROUP filterMovies\_1970 ALL;

averageDuration\_1970 = FOREACH groupMovies\_1970 GENERATE AVG(filterMovies\_1970.runTime);

totalCount\_1970 = FOREACH groupMovies\_1970 GENERATE COUNT(filterMovies\_1970);

dump averageDuration\_1970;

dump totalCount\_1970;

-(1960-1970):

filterMovies\_1960 = filter movies by (Year>1960 and Year<=1970);

groupMovies\_1960 = GROUP filterMovies\_1960 ALL;

averageDuration\_1960 = FOREACH groupMovies\_1960 GENERATE AVG(filterMovies\_1960.runTime);

totalCount\_1960 = FOREACH groupMovies\_1960 GENERATE COUNT(filterMovies\_1960);

dump averageDuration\_1960;

dump totalCount\_1960;

-(1950-1960):

filterMovies\_1950 = filter movies by (Year>1950 and Year<=1960);

groupMovies\_1950 = GROUP filterMovies\_1950 ALL;

averageDuration\_1950 = FOREACH groupMovies\_1950 GENERATE AVG(filterMovies\_1950.runTime);

totalCount\_1950 = FOREACH groupMovies\_1950 GENERATE COUNT(filterMovies\_1950);

dump averageDuration\_1950;

dump totalCount\_1950;

-(1940-1950):

filterMovies\_1940 = filter movies by (Year>1940 and Year<=1950);

groupMovies\_1940 = GROUP filterMovies\_1940 ALL;

averageDuration\_1940 = FOREACH groupMovies\_1940 GENERATE AVG(filterMovies\_1940.runTime);

totalCount\_1940 = FOREACH groupMovies\_1940 GENERATE COUNT(filterMovies\_1940);

dump averageDuration\_1940;

dump totalCount\_1940;

-(1930-1940):

filterMovies\_1930 = filter movies by (Year>1930 and Year<=1940);

groupMovies\_1930 = GROUP filterMovies\_1930 ALL;

averageDuration\_1930 = FOREACH groupMovies\_1930 GENERATE AVG(filterMovies\_1930.runTime);

totalCount\_1930 = FOREACH groupMovies\_1930 GENERATE COUNT(filterMovies\_1930);

dump averageDuration\_1930;

dump totalCount\_1930;

-(1920-1930):

filterMovies\_1920 = filter movies by (Year>1920 and Year<=1930);

groupMovies\_1920 = GROUP filterMovies\_1920 ALL;

averageDuration\_1920 = FOREACH groupMovies\_1920 GENERATE AVG(filterMovies\_1920.runTime);

totalCount\_1920 = FOREACH groupMovies\_1920 GENERATE COUNT(filterMovies\_1920);

dump averageDuration\_1920;

dump totalCount\_1920;

Distribution of ratings for various Genres:

-- Comedy

filterComedy = filter movies by (Genre matches '.\*(comedy|Comedy|COMEDY).\*');

groupfilterComedy = GROUP filterComedy ALL;

maxRatingOfComedy = FOREACH groupfilterComedy GENERATE MAX(filterComedy.imdbRating);

dump maxRatingOfComedy; 9.3

minRatingOfComedy = FOREACH groupfilterComedy GENERATE MIN(filterComedy.imdbRating);

dump minRatingOfComedy; 1.5

-- Actionexit

filterAction = filter movies by (Genre matches '.\*(Action|action|ACTION).\*');

groupfilterAction = GROUP filterAction ALL;

maxRatingOfAction = FOREACH groupfilterAction GENERATE MAX(filterAction.imdbRating);

dump maxRatingOfAction; 9.0

minRatingOfAction = FOREACH groupfilterAction GENERATE MIN(filterAction.imdbRating);

dump minRatingOfAction; 1.5

-- Animation

filterAnimation = filter movies by (Genre matches '.\*(Animation|animation|ANIMATION).\*');

groupfilterAnimation = GROUP filterAnimation ALL;

maxRatingOfAnimation = FOREACH groupfilterAnimation GENERATE MAX(filterAnimation.imdbRating);

dump maxRatingOfAnimation; 9.0

minRatingOfAnimation = FOREACH groupfilterAnimation GENERATE MIN(filterAnimation.imdbRating);

dump minRatingOfAnimation; 4.4

-- Drama

filterDrama = filter movies by (Genre matches '.\*(Drama|drama|DRAMA).\*');

groupfilterDrama = GROUP filterDrama ALL;

maxRatingOfDrama = FOREACH groupfilterDrama GENERATE MAX(filterDrama.imdbRating);

dump maxRatingOfDrama; 9.4

minRatingOfDrama = FOREACH groupfilterDrama GENERATE MIN(filterDrama.imdbRating);

dump minRatingOfDrama; 1.6

-- Documentary

filterDocumentary = filter movies by (Genre matches '.\*(Documentary|documentary|DOCUMENTARY).\*');

groupfilterDocumentary = GROUP filterDocumentary ALL;

maxRatingOfDocumentary = FOREACH groupfilterDocumentary GENERATE MAX(filterDocumentary.imdbRating);

dump maxRatingOfDocumentary; 9.3

minRatingOfDocumentary = FOREACH groupfilterDocumentary GENERATE MIN(filterDocumentary.imdbRating);

dump minRatingOfDocumentary; 2.1

-- Romance

filterRomance = filter movies by (Genre matches '.\*(Romance|romance|ROMANCE).\*');

groupfilterRomance = GROUP filterRomance ALL;

maxRatingOfRomance = FOREACH groupfilterRomance GENERATE MAX(filterRomance.imdbRating);

dump maxRatingOfRomance; 9.0

minRatingOfRomance = FOREACH groupfilterRomance GENERATE MIN(filterRomance.imdbRating);

dump minRatingOfRomance; 2.4

-- Horror

filterHorror = filter movies by (Genre matches '.\*(horror|Horror|HORROR).\*');

groupfilterHorror = GROUP filterHorror ALL;

maxRatingOfHorror = FOREACH groupfilterHorror GENERATE MAX(filterHorror.imdbRating);

dump maxRatingOfHorror; 9.0

minRatingOfHorror = FOREACH groupfilterHorror GENERATE MIN(filterHorror.imdbRating);

dump minRatingOfHorror;

-- Sci-Fi

filterSciFi = filter movies by (Genre matches '.\*(Sci|sci|Sci-Fi|sci-fi).\*');

groupfilterSciFi = GROUP filterSciFi ALL;

maxRatingOfSciFi = FOREACH groupfilterSciFi GENERATE MAX(filterSciFi.imdbRating);

dump maxRatingOfSciFi;

minRatingOfSciFi = FOREACH groupfilterSciFi GENERATE MIN(filterSciFi.imdbRating);

dump minRatingOfSciFi;

MOVIE RECOMMENDATION SYSTEM:

1. **Initial Job: Filter out the users ratings data which match the data of all movies data.**
2. **Job0:** **For every user parse the records from Amazon all movies reviews and get data in following format:**

**Output: user1 movie1; rating; total records for this movie**

TrainingDataPreprocessDriver.java

**public class TrainingDataPreprocessDriver extends Configured implements Tool** {

private static Logger THE\_LOGGER = Logger

.getLogger(TrainingDataPreprocessDriver.class);

@Override

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

Job job = new Job(getConf());

job.setJarByClass(TrainingDataPreprocessDriver.class);

job.setJobName("TrainingDataPreprocess");

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

job.setMapperClass(TrainingDataPreprocessMapper.class);

job.setReducerClass(TrainingDataPreprocessReducer.class);

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

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

boolean status = job.waitForCompletion(true);

THE\_LOGGER.info("run(): status=" + status);

return status ? 0 : 1;

}

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

//Make sure there are exactly 2 parameters

if (args.length < 2) {

THE\_LOGGER

.warn("usage TrainingDataPreprocessDriver <input> <output>");

System.exit(1);

}

THE\_LOGGER.info("inputDir=" + args[0]);

THE\_LOGGER.info("outputDir=" + args[1]);

int returnStatus = ToolRunner.run(new TrainingDataPreprocessDriver(),

args);

System.exit(returnStatus);

}

}

TrainingDataPreprocessMapper.java

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

private Text K2 = new Text();

private Text V2 = new Text();

@Override

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

throws IOException, InterruptedException {

String valueAsString = value.toString().trim();

String[] tokens = valueAsString.split(",");

if (tokens.length != 3) {

return;

}

String movieId = tokens[0].trim();

String userId = tokens[1].trim();

String rating = tokens[2].trim();

K2.set(movieId);

V2.set(userId + "," + rating);

context.write(K2, V2);

}

}

TrainingDataPreprocessReducer.java

public class TrainingDataPreprocessReducer extends

Reducer<Text, Text, Text, Text> {

private Text K2 = new Text();

private Text V2 = new Text();

@Override

public void reduce(Text key, Iterable<Text> values, Context context)

throws IOException, InterruptedException {

int numRatings = 0;

long sumRatings = 0;

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

for (Text i : values) {

/\* tokens = [UserId, Rating] \*/

String[] tokens = i.toString().split(",");

numRatings++;

sumRatings += (int) Double.parseDouble(tokens[1].trim());

users.add(i.toString());

}

for (String i : users) {

/\* tokens = [UserId, Rating] \*/

String[] tokens = i.split(",");

K2.set(tokens[0].trim()+":"); // UserId

V2.set(key.toString().trim() + "," + tokens[1].trim() + ";" + numRatings + ";"

+ sumRatings); // MovieId,Rating;numRatings;sumRatings

context.write(K2, V2);

}

}

}

1. **Job1: For every user make the cross product of the movies in this format:**

**Output: movie1, movie2; rating1, rating2; total reviews1, total reviews2;**

Job1Mapper.java

public class Job1Mapper extends Mapper<Object, Text, Text, Text> {

@Override

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

throws IOException, InterruptedException {

String line = value.toString();

String[] contents = line.split(":");

Text k = new Text(contents[0].trim());

Text V = new Text(contents[1].trim());

context.write(k,V);

}

}

Job1Reducer.java

public class Job1Reducer extends Reducer<Text, Text, Text, Text> {

private Text K2 = new Text();

private Text V2 = new Text();

static final Comparator<String> MOVIEID\_ORDER = new Comparator<String>() {

public int compare(String s1, String s2) {

return s1.compareTo(s2);

}

};

@Override

public void reduce(Text key, Iterable<Text> values, Context context)

throws IOException, InterruptedException {

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

for (Text value : values) {

moviesList.add(value.toString().trim());

}

/\* sorting is necessary here to avoid duplicate \*/

Collections.sort(moviesList, MOVIEID\_ORDER);

/\* cross product \*/

for (int i = 0; i < moviesList.size(); i++) {

for (int j = i + 1; j < moviesList.size(); j++) {

String[] movie1Tokens = moviesList.get(i).split(",");

String[] movie2Tokens = moviesList.get(j).split(",");

K2.set(movie1Tokens[0].trim() + "," + movie2Tokens[0].trim()+":");

V2.set(movie1Tokens[1].trim() + "," + movie2Tokens[1].trim());

//System.out.println("keys are "+ K2.toString());

if(!movie1Tokens[0].trim().equalsIgnoreCase(movie2Tokens[0].trim())) context.write(K2, V2);

}

}

}

}

Job1Driver.java

public class Job1Driver extends Configured implements Tool {

private static Logger THE\_LOGGER = Logger.getLogger(Job1Driver.class);

@Override

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

Job job = new Job(getConf());

job.setJarByClass(Job1Driver.class);

job.setJobName("Job1");

//job.setInputFormatClass(FileInputFormat.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

/\* compress the final output \*/

/\*

FileOutputFormat.setCompressOutput(job, true);

FileOutputFormat.setOutputCompressorClass(job, GzipCodec.class);

\*/

job.setMapperClass(Job1Mapper.class);

job.setReducerClass(Job1Reducer.class);

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

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

boolean status = job.waitForCompletion(true);

THE\_LOGGER.info("run(): status=" + status);

return status ? 0 : 1;

}

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

//Make sure there are exactly 2 parameters

if (args.length < 2) {

THE\_LOGGER.warn("usage Job1Driver <input> <output>");

System.exit(1);

}

THE\_LOGGER.info("inputDir=" + args[0]);

THE\_LOGGER.info("outputDir=" + args[1]);

int returnStatus = ToolRunner.run(new Job1Driver(), args);

System.exit(returnStatus);

}

}

1. **Similarity Calculation job: Job2: Calculates the movies similarity using numerical distance calculation: Output: movie1, Movie2: similarity Score;**

Job2Reducer.java

public class Job2Reducer extends Reducer<Text, Text, Text, Text> {

@Override

public void reduce(Text key, Iterable<Text> values, Context context)

throws IOException, InterruptedException {

double NdotProd = 0;

double Nrating1squaredSum = 0;

double Nrating2squaredSum = 0;

double mean1 = -1;

double mean2 = -1;

double similarity = 0;

int n = 0;

for (Text value : values) {

n++;

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

String[] ratings1 = tokens[0].split(";"); /\*

\* ratings =

\* rating,numRatings

\* ,sumRatings

\*/

String[] ratings2 = tokens[1].split(";");

double rating1 = Double.parseDouble(ratings1[0].trim());

double rating2 = Double.parseDouble(ratings2[0].trim());

if (mean1 == -1 || mean2 == -1) {

double num1 = Double.parseDouble(ratings1[1].trim());

double sum1 = Double.parseDouble(ratings1[2].trim());

double num2 = Double.parseDouble(ratings2[1].trim());

double sum2 = Double.parseDouble(ratings2[2].trim());

mean1 = sum1 / num1;

mean2 = sum2 / num2;

}

NdotProd += (rating1 - mean1) \* (rating2 - mean2);

Nrating1squaredSum += Math.pow(rating1 - mean1, 2);

Nrating2squaredSum += Math.pow(rating2 - mean2, 2);

}

if (Nrating1squaredSum == 0 || Nrating2squaredSum == 0 || n == 1) {

similarity=0;

}

similarity = NdotProd

/(Math.sqrt(Nrating1squaredSum) \* Math

.sqrt(Nrating2squaredSum));

similarity=Math.round(similarity\*100.0)/100.0;

if(similarity>0.0d){

context.write( key, new Text(": "+String.valueOf(similarity)+",x"));}

}

}

Job2Driver.java

public class Job2Driver extends Configured implements Tool {

private static Logger THE\_LOGGER = Logger.getLogger(Job2Driver.class);

@Override

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

Job job = new Job(getConf());

job.setJarByClass(Job2Driver.class);

job.setJobName("Job2");

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

job.setMapperClass(Job1Mapper.class);

job.setReducerClass(Job2Reducer.class);

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

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

boolean status = job.waitForCompletion(true);

THE\_LOGGER.info("run(): status=" + status);

return status ? 0 : 1;

}

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

//Make sure there are exactly 2 parameters

if (args.length < 2) {

THE\_LOGGER.warn("usage Job1Driver <input> <output>");

System.exit(1);

}

THE\_LOGGER.info("inputDir=" + args[0]);

THE\_LOGGER.info("outputDir=" + args[1]);

int returnStatus = ToolRunner.run(new Job2Driver(), args);

System.exit(returnStatus);

}

}

1. **Movies Attributes Based Similarity Calculation** Job3: Takes the input of all the movies and make the cross product of all the movies in this format:

Output: movie1,movie2 ; m1.attrib1, m2.attrib1; m1.attrib2, m2.attrib2; m1.attrib3, m2.attrib3;

Job3Mapper.java

public class Job3Mapper extends Mapper<Object, Text, Text, Text> {

@Override

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

throws IOException, InterruptedException {

String line = value.toString();

context.write(new Text("1"),value);

}

}

Job3Reducer.java

public class Job3Reducer extends Reducer<Text, Text, Text, Text> {

private Text K2 = new Text();

private Text V2 = new Text();

static final Comparator<String> MOVIEID\_ORDER = new Comparator<String>() {

public int compare(String s1, String s2) {

return s1.compareTo(s2);

}

};

@Override

public void reduce(Text key, Iterable<Text> values, Context context)

throws IOException, InterruptedException {

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

for (Text value : values) {

moviesList.add(value.toString().trim());

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

}

/\* sorting is necessary here to avoid duplicate \*/

Collections.sort(moviesList, MOVIEID\_ORDER);

/\* cross product \*/

for (int i = 0; i < moviesList.size(); i++) {

for (int j = i + 1; j < moviesList.size(); j++) {

String[] m\_arr1 = moviesList.get(i).split(",", 2);

String[] m\_arr2 = moviesList.get(j).split(",", 2);

K2.set(m\_arr1[0] + ":" +m\_arr1[1]+";");

V2.set( m\_arr2[0]+ ":" + m\_arr2[1]);

context.write(K2, V2);

}}}}

Job3Driver.java

public class Job3Driver extends Configured implements Tool {

private static Logger THE\_LOGGER = Logger.getLogger(Job3Driver.class);

@Override

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

Job job = new Job(getConf());

job.setJarByClass(Job3Driver.class);

job.setJobName("Job1");

//job.setInputFormatClass(FileInputFormat.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

/\* compress the final output \*/

/\*

FileOutputFormat.setCompressOutput(job, true);

FileOutputFormat.setOutputCompressorClass(job, GzipCodec.class);

\*/

job.setMapperClass(Job3Mapper.class);

job.setReducerClass(Job3Reducer.class);

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

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

boolean status = job.waitForCompletion(true);

THE\_LOGGER.info("run(): status=" + status);

return status ? 0 : 1;

}

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

//Make sure there are exactly 2 parameters

if (args.length < 2) {

THE\_LOGGER.warn("usage Job1Driver <input> <output>");

System.exit(1);

}

THE\_LOGGER.info("inputDir=" + args[0]);

THE\_LOGGER.info("outputDir=" + args[1]);

int returnStatus = ToolRunner.run(new Job3Driver(), args);

System.exit(returnStatus);

}

1. **Job4: For every key-value pair find the difference between the attributes and provide the output :**

**Output: movie1, movie2 :similarityScore;**

Job4Mapper.java

public class Job4Mapper extends Mapper<Object, Text, Text, Text> {

@Override

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

throws IOException, InterruptedException {

String line = value.toString();

String [] moviedata = line.split(";");

context.write(new Text(moviedata[0]),new Text(moviedata[1]));

}

}

Job4Reducer.java

public class Job4Reducer extends Reducer<Text, Text, Text, Text> {

@Override

public void reduce(Text key, Iterable<Text> values, Context context)

throws IOException, InterruptedException {

try{

String [] m1\_det = key.toString().split(":");

String m1 = m1\_det[0];

String[] features\_m1 = m1\_det[1].split(",");

String director1 = features\_m1[0].trim();

String actor1 = features\_m1[1].trim();

double imdbRating1 = Double.parseDouble(features\_m1[2].trim());

int imdbVote1 = Integer.parseInt(features\_m1[3].trim());

double amazonAvgScore1 = Double.parseDouble(features\_m1[4].trim());

int year1 = Integer.parseInt(features\_m1[5].trim());

String language1 = features\_m1[6].trim();

String country1 = features\_m1[7].trim();

String genre1 = features\_m1[8].trim();

int oscarWinner1 = Integer.parseInt(features\_m1[9].trim());

int oscarNominated1 = Integer.parseInt(features\_m1[10].trim());

int otherAwards1 = Integer.parseInt(features\_m1[11].trim());

for (Text value : values) {

try{

String[] m2\_det = value.toString().split(":");

String m2 = m2\_det[0];

String[] features\_m2 = m2\_det[1].split(",");

String director2 = features\_m2[0].trim();

String actor2 = features\_m2[1].trim();

double imdbRating2 = Double.parseDouble(features\_m2[2].trim());

int imdbVote2 = Integer.parseInt(features\_m2[3].trim());

double amazonAvgScore2 = Double.parseDouble(features\_m2[4].trim());

int year2 = Integer.parseInt(features\_m2[5].trim());

String language2 = features\_m2[6].trim();

String country2 = features\_m2[7].trim();

String genre2 = features\_m2[8].trim();

int oscarWinner2 = Integer.parseInt(features\_m2[9].trim());

int oscarNominated2 = Integer.parseInt(features\_m2[10].trim());

int otherAwards2 = Integer.parseInt(features\_m2[11].trim());

String [] actors2 = actor2.split(" ");

String [] languages2 = language2.split(" ");

double sumOfMatchingAttributes =0;

//all categorical features first

if(director2.equals(director1))

{sumOfMatchingAttributes +=1;

}

for(String s: actors2)

{

if(actor1.matches(".\*\\b"+s+"\\b.\*")){

sumOfMatchingAttributes +=1;

break;

}

}

for(String s: languages2)

{

if(language1.matches(".\*\\b"+s+"\\b.\*")){

sumOfMatchingAttributes +=1;

break;

}

}

if(country2.equals(country1))

{sumOfMatchingAttributes +=1; }

if(genre2.equals(genre1))

{sumOfMatchingAttributes +=1; }

if(year2==year1)

{sumOfMatchingAttributes +=1; }

double diff=0;

//now all numerical

if(imdbRating1>imdbRating2)

diff = imdbRating1-imdbRating2;

else diff = imdbRating2-imdbRating1;

sumOfMatchingAttributes += (1-(diff/10d))\*0.6d;

diff=0;

if(imdbVote1>imdbVote2)

diff = imdbVote1-imdbVote2;

else diff = imdbVote2-imdbVote1;

sumOfMatchingAttributes += (1-(diff/1000000d))\*0.6d;

diff=0;

if(amazonAvgScore1>amazonAvgScore2)

diff = amazonAvgScore1-amazonAvgScore2;

else diff = amazonAvgScore2-amazonAvgScore1;

sumOfMatchingAttributes += (1-(diff/5d))\*0.6d;

diff=0;

if(oscarNominated1>oscarNominated2)

diff = oscarNominated1-oscarNominated2;

else diff = oscarNominated2-oscarNominated1;

sumOfMatchingAttributes += (1-(diff/11d))\*0.6d;

diff=0;

if(oscarWinner1>oscarWinner2)

diff = oscarWinner1-oscarWinner2;

else diff = oscarWinner2-oscarWinner1;

sumOfMatchingAttributes += (1-(diff/11d))\*0.6d;

diff=0;

if(otherAwards1>otherAwards2)

diff = otherAwards1-otherAwards2;

else diff = otherAwards2-otherAwards1;

sumOfMatchingAttributes += (1-(diff/480d))\*0.4d;

String keyEmit = m1+","+m2+":";

double similarityScore = (sumOfMatchingAttributes/12d);

similarityScore=Math.round(similarityScore\*100.0)/100.0;

if(similarityScore>0.0d){

context.write( new Text(keyEmit), new Text(String.valueOf(similarityScore)+",y"));

}

}catch(ArrayIndexOutOfBoundsException e){

System.out.println("The values for it is gone "+ value);

e.printStackTrace();

}}}

catch(ArrayIndexOutOfBoundsException e){

System.out.println("The keys for it is crazy "+ key.toString());

e.printStackTrace();

}

}}

Job4Driver.java

public class Job4Driver extends Configured implements Tool {

private static Logger THE\_LOGGER = Logger.getLogger(Job4Driver.class);

@Override

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

Job job = new Job(getConf());

job.setJarByClass(Job4Driver.class);

job.setJobName("Job1");

//job.setInputFormatClass(FileInputFormat.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

/\* compress the final output \*/

/\*

FileOutputFormat.setCompressOutput(job, true);

FileOutputFormat.setOutputCompressorClass(job, GzipCodec.class);

\*/

job.setMapperClass(Job4Mapper.class);

job.setReducerClass(Job4Reducer.class);

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

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

boolean status = job.waitForCompletion(true);

THE\_LOGGER.info("run(): status=" + status);

return status ? 0 : 1;

}

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

//Make sure there are exactly 2 parameters

if (args.length < 2) {

THE\_LOGGER.warn("usage Job1Driver <input> <output>");

System.exit(1);

}

THE\_LOGGER.info("inputDir=" + args[0]);

THE\_LOGGER.info("outputDir=" + args[1]);

int returnStatus = ToolRunner.run(new Job4Driver(), args);

System.exit(returnStatus);

}

}

1. **Merge Similarity job: Job5: Calculates the average of similarity from previous two jobs output Output: movie1, movie2 : similarityScore;**

Job5Mapper.java

public class Job5Mapper extends Mapper<Object, Text, Text, Text> {

@Override

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

throws IOException, InterruptedException {

String line = value.toString();

String [] moviedata = line.split(":");

String [] movies = moviedata[0].split(",");

String k = movies[0].trim()+","+movies[1].trim();

context.write(new Text(k),new Text(moviedata[1].trim()));

}

}

Job5Reducer.java

public class Job5Reducer extends Reducer<Text, Text, Text, Text> {

@Override

public void reduce(Text key, Iterable<Text> values, Context context)

throws IOException, InterruptedException {

try{

String [] movies = key.toString().split(",");

String m1 = movies[0];

String m2 = movies[1];

String present = null;

double count =0;

double sum=0;

double finalscore=0;

for (Text value : values) {

try{

count++;

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

//present +=sc[1]+",";

double score = Double.parseDouble(sc[0].toString().trim());

sum+=score;

}catch(ArrayIndexOutOfBoundsException e){

System.out.println("The values for it is gone "+ value);

e.printStackTrace();

}}

//if(!present.matches(".\*\\by\\b.\*")){

//return;

//}

finalscore=Math.round((sum/count)\*100.0)/100.0;

context.write( key, new Text(": "+String.valueOf(finalscore)));

}catch(Exception e){

e.printStackTrace();

}

}}

Job5Driver.java

public class Job5Driver extends Configured implements Tool {

private static Logger THE\_LOGGER = Logger.getLogger(Job5Driver.class);

@Override

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

Job job = new Job(getConf());

job.setJarByClass(Job5Driver.class);

job.setJobName("Job1");

//job.setInputFormatClass(FileInputFormat.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

/\* compress the final output \*/

/\*

FileOutputFormat.setCompressOutput(job, true);

FileOutputFormat.setOutputCompressorClass(job, GzipCodec.class);

\*/

job.setMapperClass(Job5Mapper.class);

job.setReducerClass(Job5Reducer.class);

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

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

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

boolean status = job.waitForCompletion(true);

THE\_LOGGER.info("run(): status=" + status);

return status ? 0 : 1;

}

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

//Make sure there are exactly 2 parameters

if (args.length < 3) {

THE\_LOGGER.warn("usage Job1Driver <input1> <input2> <output>");

System.exit(1);

}

THE\_LOGGER.info("inputDir=" + args[0]);

THE\_LOGGER.info("outputDir=" + args[1]);

int returnStatus = ToolRunner.run(new Job5Driver(), args);

System.exit(returnStatus);

}

}

1. **Find Similar Movies: Job6: Finds top 10 movies with highest similarity score for every movie. Output: movie# movie1:similarityScore; movie2:similarityScore; movie3:similarityScore;**

Job6Mapper.java

public class Job6Mapper extends Mapper<Object, Text, Text, Text> {

@Override

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

throws IOException, InterruptedException {

String line = value.toString().trim();

String [] moviedata = line.split(",");

context.write(new Text(moviedata[0].trim()),new Text(moviedata[1].trim()));

String[] set2 = moviedata[1].split(":");

String newVal = moviedata[0].trim()+":"+set2[1].trim();

context.write(new Text(set2[0].trim()),new Text(newVal));

}

}

Job6Reducer.java

public class Job6Reducer extends Reducer<Text, Text, Text, Text> {

@Override

public void reduce(Text key, Iterable<Text> values, Context context)

throws IOException, InterruptedException {

//HashMap<String, Double> list = new HashMap<String, Double>();

String similarMovies = new String("");

String k = null;

Double v =null;

HashMap<String, Double> map = new HashMap<String, Double>();

ValueComparator bvc = new ValueComparator(map);

TreeMap<String, Double> sorted\_map = new TreeMap<String, Double>(bvc);

for (Text value : values) {

String [] data = value.toString().split(":");

k = data[0];

v = Double.parseDouble(data[1]);

map.put(k, v);

//similarMovies= similarMovies.concat(value+",");

}

sorted\_map.putAll(map);

int count=10;

for (Entry<String, Double> entry : sorted\_map.entrySet()) {

if(count>0)

{ String setval = entry.getKey().toString().trim()+":"+String.valueOf(entry.getValue());

similarMovies= similarMovies.concat(setval+",");

}

count--;

}

context.write( key, new Text("# "+similarMovies));

}

class ValueComparator implements Comparator<String> {

Map<String, Double> base;

public ValueComparator(Map<String, Double> base) {

this.base = base;

}

//Note: this comparator imposes orderings that are inconsistent with

//equals.

public int compare(String a, String b) {

if (base.get(a) >= base.get(b)) {

return -1;

} else {

return 1;

} // returning 0 would merge keys

}

}

}

Job6Driver.java

public class Job6Driver extends Configured implements Tool {

private static Logger THE\_LOGGER = Logger.getLogger(Job6Driver.class);

@Override

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

Job job = new Job(getConf());

job.setJarByClass(Job6Driver.class);

job.setJobName("Job1");

//job.setInputFormatClass(FileInputFormat.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

/\* compress the final output \*/

/\*

FileOutputFormat.setCompressOutput(job, true);

FileOutputFormat.setOutputCompressorClass(job, GzipCodec.class);

\*/

job.setMapperClass(Job6Mapper.class);

job.setReducerClass(Job6Reducer.class);

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

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

boolean status = job.waitForCompletion(true);

THE\_LOGGER.info("run(): status=" + status);

return status ? 0 : 1;

}

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

//Make sure there are exactly 2 parameters

if (args.length < 2) {

THE\_LOGGER.warn("usage Job1Driver <input> <output>");

System.exit(1);

}

THE\_LOGGER.info("inputDir=" + args[0]);

THE\_LOGGER.info("outputDir=" + args[1]);

int returnStatus = ToolRunner.run(new Job6Driver(), args);

System.exit(returnStatus);

}

}

1. **Recommend Movies to user: Job7: Finds top 10 movies with highest similarity score for movies watched by user. Find all movies watched by user**

Job7Mapper.java

public class Job7Mapper extends Mapper<Object, Text, Text, Text> {

private Set<String> moviesWatched = new HashSet<String>();

private String userid="";

@Override

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

throws IOException, InterruptedException {

//System.out.println("data is "+value);

String line = value.toString().trim();

String movie = line.trim().split("#")[0].trim();

for(String s:moviesWatched)

{

if(movie.equalsIgnoreCase(s)){

String val =line.trim().split("#")[1].trim();

//System.out.println("user is "+userid+" and "+val);

context.write(new Text(userid),new Text(val));

break;

}

}

}

@Override

protected void setup(Context context) throws IOException,

InterruptedException {

Configuration conf = context.getConfiguration();

String usersFile = conf.get("usersFile");

userid = conf.get("userid");

File usermoviefile = new File(usersFile);

BufferedReader br = new BufferedReader(new FileReader(usermoviefile));

String line = null;

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

//System.out.println(line);

String [] contents = line.split(":");

String user = contents[0].trim();

if(user.equalsIgnoreCase(userid)){

String movieId = ( contents[1].trim().split(",")[0]).trim();

moviesWatched.add(movieId);

}}

br.close();

}

}

Job7Reducer.java

public class Job7Reducer extends Reducer<Text, Text, Text, Text> {

private Map<String, Double> moviesWatched = new HashMap<String, Double>();

private String userid="";

@Override

public void reduce(Text key, Iterable<Text> values, Context context)

throws IOException, InterruptedException {

String similarMovies = new String("");

String k = null;

Double v =null;

HashMap<String, Double> map = new HashMap<String, Double>();

ValueComparator bvc = new ValueComparator(map);

TreeMap<String, Double> sorted\_map = new TreeMap<String, Double>(bvc);

for (Text value : values) {

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

for(String m:data){

String [] s = m.toString().split(":");

for(Entry<String, Double> entry : moviesWatched.entrySet()){

String watched = entry.getKey();

double rating = entry.getValue();

if(!watched.equalsIgnoreCase(s[0])){

k = s[0];

v =Math.round((Double.parseDouble(s[1])\*rating)\*100.0)/100.0 ;

System.out.println("The movie is "+k+" weighted rate "+v);

if(map.containsKey(k)){

System.out.println("The movie already present "+k);

double val = map.get(k);

if(v>val){

System.out.println("changing value for "+k+" to value "+v);

map.put(k, v);}

}

else

map.put(k, v);

}

}

}

//similarMovies= similarMovies.concat(value+",");

}

sorted\_map.putAll(map);

int count=10;

for (Entry<String, Double> entry : sorted\_map.entrySet()) {

if(count>0)

{ String setval = entry.getKey().toString().trim()+":"+String.valueOf(entry.getValue());

similarMovies= similarMovies.concat(setval+",");

}

count--;

}

context.write( key, new Text("# "+similarMovies));

}

@Override

protected void setup(Context context) throws IOException,

InterruptedException {

Configuration conf = context.getConfiguration();

String usersFile = conf.get("usersFile");

userid = conf.get("userid");

File usermoviefile = new File(usersFile);

BufferedReader br = new BufferedReader(new FileReader(usermoviefile));

String line = null;

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

//System.out.println(line);

String [] contents = line.split(":");

String user = contents[0].trim();

if(user.equalsIgnoreCase(userid)){

String movieId = ( contents[1].trim().split(",")[0]).trim();

double movieRating =Double.parseDouble(( (contents[1].trim().split(",")[1]).split(";")[0]).trim());

moviesWatched.put(movieId,movieRating);

}}

br.close();

}

class ValueComparator implements Comparator<String> {

Map<String, Double> base;

public ValueComparator(Map<String, Double> base) {

this.base = base;

}

//Note: this comparator imposes orderings that are inconsistent with

//equals.

public int compare(String a, String b) {

if (base.get(a) >= base.get(b)) {

return -1;

} else {

return 1;

} // returning 0 would merge keys

}

}

}

Job7Driver.java

public class Job7Driver extends Configured implements Tool {

private static Logger THE\_LOGGER = Logger.getLogger(Job7Driver.class);

@Override

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

Job job = new Job(getConf());

job.setJarByClass(Job7Driver.class);

job.setJobName("Job7");

job.getConfiguration().set("usersFile", args[1]);

job.getConfiguration().set("userid", args[0]);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

job.setMapperClass(Job7Mapper.class);

job.setReducerClass(Job7Reducer.class);

FileInputFormat.addInputPath(job, new Path(args[2]));

FileOutputFormat.setOutputPath(job, new Path(args[3]));

boolean status = job.waitForCompletion(true);

THE\_LOGGER.info("run(): status=" + status);

return status ? 0 : 1;

}

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

//Make sure there are exactly 2 parameters

if (args.length < 3) {

THE\_LOGGER.warn("usage Job7Driver <user Id> <user rating file> <movie similarity file> <output>");

System.exit(1);

}

THE\_LOGGER.info("inputDir=" + args[1]);

THE\_LOGGER.info("outputDir=" + args[3]);

int returnStatus = ToolRunner.run(new Job7Driver(), args);

System.exit(returnStatus);

}

}