THEORY ASSIGNMENT 3

Name: Aditya Shivram Mahajan
Roll no: A-10
Reg no: 2017BCS001

Design a MapReduce Application to process the following data for extracting meaningful information from it. Computing Average rating of movies with movie title (Used movies.csv and ratings.csv)

STEP 1: In the first step we will Create a Mapper Class code. The things will be implemented as follows.

MovieNameRatingMapper.java :-

```
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
public class MovieNameRatingMapper extends Mapper<LongWritable, Text, IntWritable,
Text> {
	@Override
	public void map(LongWritable key, Text value, Context context) throws IOException,
	InterruptedException {
		// Skip the header row of csv
		if (key.get() == 0 && value.toString().contains("userId")) {
		return;
	}
	String line = value.toString();
	String[] ratingPieces = line.split(",");
		if (ratingPieces.length >= 2) {
			// ratingPieces is: [userId, movieId, rating, timestamp]
			int movieId = Integer.parseInt(ratingPieces[1]);
			String rating = ratingPieces[2];
			// Convert to Hadoop types
			IntWritable mapKey = new IntWritable(movieId);
			Text mapValue = new Text("rating\t" + rating);// Output intermediate
			key,value pair
			context.write(mapKey, mapValue);
		}
	}
}
```

MovieRatingMapper.java :-

```
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
public class MovieRatingMapper extends Mapper<LongWritable, Text, IntWritable,
DoubleWritable> {
	@Override
	public void map(LongWritable key, Text value, Context context) throws IOException,
	InterruptedException {
		// Skip the header row of csv
		if (key.get() == 0 && value.toString().contains("userId")) {
		return;
	}
	String line = value.toString();
	String[] ratingPieces = line.split(",");
	if (ratingPieces.length >= 2) {
				// ratingPieces is: [userId, movieId, rating, timestamp]
				int movieId = Integer.parseInt(ratingPieces[1]);
				double rating = Double.parseDouble(ratingPieces[2]);// Convert to
				Hadoop types
				IntWritable mapKey = new IntWritable(movieId);
				DoubleWritable mapValue = new DoubleWritable(rating);
				// Output intermediate key,value pair
				context.write(mapKey, mapValue);
		}
	}
}
```

MovieNameMapper.java :-

```
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import java.io.IOException;
public class MovieNameMapper extends Mapper<LongWritable, Text, IntWritable, Text> {
	@Override
	public void map(LongWritable key, Text value, Context context) throws IOException,
```

```
        InterruptedException {
                // Skip the header row of csv
                if (key.get() == 0 && value.toString().contains("movieId")) {
                return;
        }
        String line = value.toString();
        String[] moviePieces = line.split(",");
                if (moviePieces.length >= 2) {
                        // ratingPieces is: [movieId,title,genres], want (movieId, title)
                        int movieId = Integer.parseInt(moviePieces[0]);
                        String title = moviePieces[1];// Convert to Hadoop types (mark this as
                        the "title" value)
                        IntWritable mapKey = new IntWritable(movieId);
                        Text mapValue = new Text("title\t" + title);
                        // Output intermediate key,value pair
                        context.write(mapKey, mapValue);
                }
        }
}
```

STEP 2: Now in the second step we will Create a Reducer Class code. The imlementation is as following.

MovieNameRatingReducer.java :-

```
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
public class MovieNameRatingReducer extends Reducer<IntWritable, Text, IntWritable, Text>
{
        @Override
        public void reduce(IntWritable key, Iterable<Text> values, Context context)
        throws IOException, InterruptedException{
                        double avgRating = 0;
                        int numValues = 0;
                        String title = "";
                        // Add up all the ratings
                        for (Text value: values) {
                        // Separate the value from the tag
                        String parts[] = value.toString().split("\t");if (parts[0].equals("title")) {
                        // Get the title
                        title = parts[1];
                } else {
                        // Get a rating for this title
```

```
                        double ratingValue = Double.parseDouble(parts[1]);
                        avgRating += ratingValue;
                        numValues++;
                }
        }
        // Divide by the number of ratings to get the average for this movie
        avgRating /= numValues;
        String outputValue = title + "\n" + avgRating;
        Text output = new Text(outputValue);
        // Output movieId, title and avg rating
        context.write(key, output);
        }
}
```

MovieRatingReducer.java :-

```
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.mapreduce.Reducer;public class MovieRatingReducer extends
Reducer<IntWritable, DoubleWritable, IntWritable,
DoubleWritable> {
        @Override
        public void reduce(IntWritable key, Iterable<DoubleWritable> values, Context context)
        throws IOException, InterruptedException{
                        double avgRating = 0;
                        int numValues = 0;
                        // Add up all the ratings
                        for (DoubleWritable value: values) {
                        avgRating += value.get();
                        numValues++;
                }
                // Divide by the number of ratings to get the average for this movie
                avgRating /= numValues;
                DoubleWritable average = new DoubleWritable(avgRating);
                // Output the average rating for this movie
                context.write(key, average);
        }
}
```

STEP 3: Now we Create Driver code

MovieNamesRatings.java :-

```
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.TextInputFormat;import
org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.input.MultipleInputs;
public class MovieNamesRatings {
	public static void main(String[] args) throws Exception {
		// Check that the input and output path have been provided
		if (args.length != 3) {
			System.err.println("Syntax: MovieNamesRatings <ratings input path>
			<titles input path>
			<output path>");
			System.exit(-1);
		}
		// Create an instance of the MapReduce job
		Job job = new Job();
		job.setJarByClass(MovieNamesRatings.class);
		job.setJobName("Average movie rating by movie title");
		// Set input and output locations
		MultipleInputs.addInputPath(job, new Path(args[0]), TextInputFormat.class,
		MovieNameRatingMapper.class);
		MultipleInputs.addInputPath(job, new Path(args[1]), TextInputFormat.class,
		MovieNameMapper.class);
		FileOutputFormat.setOutputPath(job, new Path(args[2]));
		// Set reducer classes
		job.setReducerClass(MovieNameRatingReducer.class);
		// Set output key/value
		job.setOutputKeyClass(IntWritable.class);
		job.setOutputValueClass(Text.class);
		// Run the job and then exit the
		programSystem.exit(job.waitForCompletion(true) ? 0 : 1);
	}
}
```

STEP 4: Compile all JAVA files using -classpath option in javac
javac -cp
$HADOOP_INSTALL/share/hadoop/mapreduce/hadoop-mapreduce-client-core-2.9.2.jar:$HADOOP_INSTALL/share/hadoop/common/hadoop-common-2.9.2.jar -d <OUTPUT_PATH_FOR_COMPILED_FILES> <INPUT_JAVA_FILES_PATH>/*.java

==========

```
skystone@skystone-HP-Notebook:/mnt/E4687B61687B3182/map-reduce$ javac -cp
/usr/skystone/hadoop/hadoop-2.9.2/ share/hadoop/mapreduce/hadoop-mapreduce-
client-core-2.9.2.jar:/usr/local/hadoop/hadoop-2.9.2/share/hadoop/common/hadoop-common-2.9.2.jar -d
/mnt/E4687B61687B3182/map-reduce/src *.java

Note: Some input files use or override a deprecated API.
Note: Recompile with -Xlint :deprecation for details.
skystone@skystone-HP-Notebook:/mnt/E4687B61687B3182/map-reduce$
```

==========

STEP 5: Create JAR file
jar -vcf <PATH_TO_INITIALIZED_NEWLY_CREATED_JAR_FILE>/mr.jar <INPUT_PATH_OF_ALL_COMPILED_CLASS_FILES>/*.class

==========

```
skystone@skystone-HP-Notebook:/mnt/E4687B61687B3182/map-reduce$ jar -vcf mr.jar *.class
added manifest
adding: movieNamemapper.class(in = 2272) (out= 958) (deflated 57%)
adding: movieNameRatingmapper.class(in = 2284) (out= 961) (deflated 57%)
adding: movieNameRatingReducer.class(in = 2472) (out= 1857) (deflated 57%)
adding: movieNamesRatings.class(in = 1638) (out= 878) (deflated 46%)
adding: movieRatingmapper.class(in = 1957) (out= 819) (deflated 58%)
adding: movieRatingReducer.class(in = 1785) (out= 781) (deflated 58%)
adding: movieRatings.class(in = 1414) (out= 885) (deflated 43%)
skystone@skystone-HP-Notebook:/mnt/E4687B61687B3182/map-reduce$
```

==========

STEP 6: Run the JAR file
$HADOOP_INSTALL/bin/hadoop jar <PATH_OF_JAR_FILE/mr.jar <PATH_OF_DRIVER_CLASS>/MovieNamesRatings <INPUT_PATH_TO_RATING.CSV_FILE_OR_DATA>/ratings.csv <INPUT_PATH_TO_MOVIES>CSV_FILE>/movies.csv <OUTPUT_PATH_WHERE_OUTPUT_IS_CREATED>

==========

skystone@skystone-HP-Notebook:/mnt/E4687B61687B3182/map-reduce$
/usr/local/hadoop/hadoop-2.9.2/bin/hadoop jar mr.jar MovieNamesRatings
/mnt/E4687B61687B3182/map-reduce/data/ratings.csv
/mnt/E4687B61687B3182/map-reduce/data/movies.csv /mnt/E4687B61687B3182/map-reduce/OUTPUTS

```
06/12/12 12:11:58 INFO mapreduce. Job: counters: 38
        File System Counters
                FILE: Number of bytes read-7886222
                FILE: Number of bytes written-5352124
                FILE: Number of read operations=ø
                FILE: Number of large read operations=ø
                FILE: Number of write operations=ø
        map-Reduce Framework
                map input records-188837
                map output records=1ßß836
                map output bytes-1218832
                map output materialized bytes-1411718
                Input split bytes-133
                Combine input records=ß
                Combine output records=ß
                Reduce input groups-9724
                Reduce shuffle bytes-1411718
                Reduce input records-188836
                Reduce output records-9724
                Spilled Records-281672
                Shuffled maps
                Failed Shuffles=ß
                merged map outputs-I
                GC time elapsed
                Total committed heap usage (bytes) =488247888
        Shuffle Errors
                BAD ID=ø
                CONNECTION=ø
                10 ERROR=ø
                WRONG LENGTH=ø
                WRONG
                WRONG REDUCE=ø
        File Input Format Counters
                Bytes Read-2483723
        File Output Format Counters
                Bytes Written-137486
```

STEP 7: Display Output File
$HADOOP_INSTALL/bin/hdfs dfs -cat <PATH_OF_OUTPUT_FILE>/part-r-00000

==========

skystone@skystone-HP-Notebook:/mnt/E4687B61687B3182/map-reduce$
/usr/local/hadoop/hadoop-2.9.2/bin/hdfs dfs -cat
/mnt/E4687B61687B3182/map-reduce/OUTPUT9/part-r-00000

==========

8593 Juice (1992)
4.0
8596
Revenge of the Pink Panther (1978)
3.25
8600
Angels with Dirty Faces (1938)
3.75
8601
Zero de conduite (Zero for Conduct) (Zéro de conduite: Jeunes diables au collège) (1933)
4.0
8604
Taxi (1998)
3.2
8605
Taxi 3 (2003)
3.25
8607
Tokyo Godfathers (2003)
3.9
8609
Our Hospitality (1923)
3.25
8610
All of Me (1984)
2.5
8611 "Farmer's Daughter
3.5
8614

Overboard (1987)
3.0
8617
Butterfield 8 (1960)
3.0
8620 "Exterminating Angel
1.5
8622
Fahrenheit 9/11 (2004)3.4864864864864864
8623
Roxanne (1987)
3.272727272727273
8626
Dr. Terror's House of Horrors (1965)
1.5
8632
Secret Society (2002)
0.5
8633 "Last Starfighter
3.5714285714285716
8636
Spider-Man 2 (2004)
3.8037974683544302
8638
Before Sunset (2004)
3.7
8640
King Arthur (2004)
2.9615384615384617
8641
Anchorman: The Legend of Ron Burgundy (2004)
3.7719298245614037
8643 "Cinderella Story
3.2777777777777777
8644 "I
3.4918032786885247
8645 "Maria Full of Grace (Maria
3.9375
8650
Long Day's Journey Into Night (1962)
3.5
8656 "Short Film About Killing

3.75
8665 "Bourne Supremacy
3.7866666666666666
8666
Catwoman (2004)
1.3333333333333333
8667
A Home at the End of the World (2004)3.5
8670 "Testament of Dr. Mabuse
4.0
8677
Flash Gordon Conquers the Universe (1940)
1.25
8684 "Man Escaped
3.5
8685 "Miracle of Marcelino
3.5
8690
Slaughterhouse-Five (1972)
4.0
8695 "Bachelor and the Bobby-Soxer
3.0