Assignment - 5.1

Task1: Write a Map Reduce program to find the number of unique listeners in the data set.

Answer1:

Java code screen shots are as below.(Mapper code + Reducer code + Driver code)

UniqueListenersMapper.java | UniqueListenersReducer.java | UniqueListeners.java

```
1  package UniqueListenersPackage;
2
3  import java.io.IOException;
12
13 public class UniqueListenersMapper extends Mapper<LongWritable, Text, IntWritable, IntWritable>{
14
15     IntWritable trackId = new IntWritable();
16     IntWritable userId = new IntWritable();
17
18     public void map(LongWritable key, Text value, Context context)
19               throws IOException, InterruptedException {
20
21         String record = value.toString();
22         String parts[] = record.split("\\|");
23
24         trackId.set(Integer.parseInt(parts[1]));
25         userId.set(Integer.parseInt(parts[0]));
26
27         if (parts.length == 5) {
28             context.write(trackId, userId);
29         }
30     }
31
32 }
```

UniqueListenersMapper.java | UniqueListenersReducer.java | UniqueListeners.java

```
1  package UniqueListenersPackage;
2
3  import java.io.IOException;
9
10 public class UniqueListenersReducer extends Reducer<IntWritable, IntWritable, IntWritable, IntWritable> {
11
12       public void reduce(
13               IntWritable trackId,
14               Iterable<IntWritable> userIds,
15               Reducer<IntWritable, IntWritable, IntWritable, IntWritable>.Context context)
16               throws IOException, InterruptedException {
17
18           Set<Integer> userIdSet = new HashSet<Integer>();
19             for (IntWritable userId : userIds) {
20                     userIdSet.add(userId.get());
21             }
22             IntWritable size = new IntWritable(userIdSet.size());
23             context.write(trackId, size);
24       }
25 }
26
```

UniqueListenersMapper.java | UniqueListenersReducer.java | UniqueListeners.java

```java
package UniqueListenersPackage;

import java.io.IOException;

public class UniqueListeners {

    public static void main(String[] args) throws ClassNotFoundException, IOException, InterruptedException {
        // TODO Auto-generated method stub
        if (args.length != 2) {
            System.err.println("Usage: UniqueListeners <input path> <output path>");
            System.exit(-1);
          }

        //Job Related Configurations
        Configuration conf = new Configuration();
        Job job = new Job(conf, "No. of Unique Listeners");
        job.setJarByClass(UniqueListeners.class);

        // Specify the number of reducer to 1
        job.setNumReduceTasks(1);

        //Provide paths to pick the input file for the job
        FileInputFormat.setInputPaths(job, new Path(args[0]));

        //Provide paths to pick the output file for the job, and delete it if already present
        Path outputPath = new Path(args[1]);
        FileOutputFormat.setOutputPath(job, outputPath);
        outputPath.getFileSystem(conf).delete(outputPath, true);

          //To set the mapper and reducer of this job
          job.setMapperClass(UniqueListenersPackage.UniqueListenersMapper.class);
          job.setReducerClass(UniqueListenersPackage.UniqueListenersReducer.class);

          //set the input and output format class
          job.setInputFormatClass(TextInputFormat.class);
          job.setOutputFormatClass(TextOutputFormat.class);

          //set up the output key and value classes
          job.setOutputKeyClass(IntWritable.class);
          job.setOutputValueClass(IntWritable.class);

          //execute the job
          System.exit(job.waitForCompletion(true) ? 0 : 1);
      }

}
```

Map Reduce execution in HDFS:

Input Command:

hadoop jar UniqueListeners.jar /musicdata.txt /UniqueOut

```
[acadgild@localhost ~]$ hadoop jar UniqueListeners.jar /musicdata.txt /UniqueOut
18/03/18 15:36:35 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
18/03/18 15:36:36 INFO client.RMProxy: Connecting to ResourceManager at localhost/127.0.0.1:8032
18/03/18 15:36:37 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
18/03/18 15:36:38 INFO input.FileInputFormat: Total input paths to process : 1
18/03/18 15:36:38 INFO mapreduce.JobSubmitter: number of splits:1
18/03/18 15:36:38 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1521362588389_0004
18/03/18 15:36:38 INFO impl.YarnClientImpl: Submitted application application_1521362588389_0004
18/03/18 15:36:38 INFO mapreduce.Job: The url to track the job: http://localhost:8088/proxy/application_1521362588389_0004/
18/03/18 15:36:38 INFO mapreduce.Job: Running job: job_1521362588389_0004
```

Output: Since it is a single reducer job, only one output file will be received as mentioned below which will have the unique listeners for each of the tracks.

```
[acadgild@localhost ~]$ hadoop fs -ls /UniqueOut
18/03/18 15:37:21 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes whe
re applicable
Found 2 items
-rw-r--r--   1 acadgild supergroup          0 2018-03-18 15:37 /UniqueOut/_SUCCESS
-rw-r--r--   1 acadgild supergroup         18 2018-03-18 15:37 /UniqueOut/part-r-00000
[acadgild@localhost ~]$ hadoop fs -cat /UniqueOut/part-r-00000
18/03/18 15:37:38 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes whe
re applicable
222     1
223     1
225     2
```

Task2: Write a Map Reduce program to get are the number of times a song was heard fully.

Answer2:

Java code screen shots are as below.(Mapper code + Reducer code + Driver code)

HeardFullyMapper.java | HeardFullyReducer.java | HeardFully.java

```
1  package HeardFullyPackage;
2
3  import java.io.IOException;
4
5  import org.apache.hadoop.io.IntWritable;
6  import org.apache.hadoop.io.LongWritable;
7  import org.apache.hadoop.io.Text;
8  import org.apache.hadoop.mapreduce.Mapper;
9  import org.apache.hadoop.mapreduce.Mapper.Context;
10
11 import java.util.*;
12
13 public class HeardFullyMapper extends Mapper<LongWritable, Text, IntWritable, IntWritable>{
14
15     IntWritable trackId = new IntWritable();
16     IntWritable heardFull = new IntWritable();
17
18     public void map(LongWritable key, Text value, Context context)
19               throws IOException, InterruptedException {
20
21         String record = value.toString();
22         String parts[] = record.split("\\|");
23
24         trackId.set(Integer.parseInt(parts[1]));
25         heardFull.set(Integer.parseInt(parts[4]));
26
27         if (parts.length == 5) {
28             context.write(trackId, heardFull);
29         }
30     }
31
32 }
```

HeardFullyMapper.java | HeardFullyReducer.java | HeardFully.java

```java
package HeardFullyPackage;

import java.io.IOException;
import java.util.*;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.Reducer.Context;

public class HeardFullyReducer extends Reducer<IntWritable, IntWritable, IntWritable, IntWritable> {

	public void reduce(
			IntWritable trackId,
			Iterable<IntWritable> heardFull,
			Reducer<IntWritable, IntWritable, IntWritable, IntWritable>.Context context)
			throws IOException, InterruptedException {

		List<Integer> heardFullyList = new ArrayList<Integer>();

		for(IntWritable songsHeard: heardFull) {

			if(songsHeard.get() == 1) {
				heardFullyList.add(songsHeard.get());
			}
		}

		IntWritable size = new IntWritable(heardFullyList.size());
		context.write(trackId, size);

	}
}
```

HeardFullyMapper.java | HeardFullyReducer.java | HeardFully.java

```java
package HeardFullyPackage;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

import UniqueListenersPackage.UniqueListeners;

public class HeardFully {

    public static void main(String[] args) throws ClassNotFoundException, IOException, InterruptedException{
        // TODO Auto-generated method stub
        if (args.length != 2) {
            System.err.println("Usage: HeardFully <input path> <output path>");
            System.exit(-1);
          }

        //Job Related Configurations
        Configuration conf = new Configuration();
        Job job = new Job(conf, "No. of songs heard fully");
        job.setJarByClass(HeardFully.class);

        // Specify the number of reducer to 1
        job.setNumReduceTasks(1);

        //Provide paths to pick the input file for the job
        FileInputFormat.setInputPaths(job, new Path(args[0]));

        //Provide paths to pick the output file for the job, and delete it if already present
        Path outputPath = new Path(args[1]);
        FileOutputFormat.setOutputPath(job, outputPath);
```

```
39          outputPath.getFileSystem(conf).delete(outputPath, true);
40
41          //To set the mapper and reducer of this job
42          job.setMapperClass(HeardFullyPackage.HeardFullyMapper.class);
43          job.setReducerClass(HeardFullyPackage.HeardFullyReducer.class);
44
45          //set the input and output format class
46          job.setInputFormatClass(TextInputFormat.class);
47          job.setOutputFormatClass(TextOutputFormat.class);
48
49          //set up the output key and value classes
50          job.setOutputKeyClass(IntWritable.class);
51          job.setOutputValueClass(IntWritable.class);
52
53          //execute the job
54          System.exit(job.waitForCompletion(true) ? 0 : 1);
55      }
56
57  }
58
```

Input Command:

hadoop jar HeardFully.jar /musicdata.txt /HeardFullyOut

```
[acadgild@localhost ~]$ hadoop jar HeardFully.jar /musicdata.txt /HeardFullyOut
18/03/18 17:06:11 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
18/03/18 17:06:12 INFO client.RMProxy: Connecting to ResourceManager at localhost/127.0.0.1:8032
18/03/18 17:06:13 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
18/03/18 17:06:13 INFO input.FileInputFormat: Total input paths to process : 1
18/03/18 17:06:13 INFO mapreduce.JobSubmitter: number of splits:1
18/03/18 17:06:13 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1521362588389_0006
18/03/18 17:06:14 INFO impl.YarnClientImpl: Submitted application application_1521362588389_0006
18/03/18 17:06:14 INFO mapreduce.Job: The url to track the job: http://localhost:8088/proxy/application_1521362588389_0006/
18/03/18 17:06:14 INFO mapreduce.Job: Running job: job_1521362588389_0006
18/03/18 17:06:24 INFO mapreduce.Job: Job job_1521362588389_0006 running in uber mode : false
18/03/18 17:06:24 INFO mapreduce.Job:  map 0% reduce 0%
18/03/18 17:06:31 INFO mapreduce.Job:  map 100% reduce 0%
18/03/18 17:06:38 INFO mapreduce.Job:  map 100% reduce 100%
18/03/18 17:06:38 INFO mapreduce.Job: Job job_1521362588389_0006 completed successfully
```

Output: The reducer output file will have the details for the no. of times a song was heard fully.

```
[acadgild@localhost ~]$ hadoop fs -cat /HeardFullyOut/part-r-00000
18/03/18 17:09:15 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
222     0
223     1
225     0
```

Task3: Write a Map Reduce program to get number of times a song was shared.

Answer3:

Java code screen shots are as below.(Mapper code + Reducer code + Driver code)

SongSharedMapper.java | SongSharedReducer.java | SongShared.java

```java
package SongSharedPackage;

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;
import org.apache.hadoop.mapreduce.Mapper.Context;

import java.util.*;

public class SongSharedMapper extends Mapper<LongWritable, Text, IntWritable, IntWritable>{

    IntWritable trackId = new IntWritable();
    IntWritable songShared = new IntWritable();

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

        String record = value.toString();
        String parts[] = record.split("\\|");

        trackId.set(Integer.parseInt(parts[1]));
        songShared.set(Integer.parseInt(parts[2]));

        if (parts.length == 5) {
            context.write(trackId, songShared);
        }
    }

}
```

SongSharedMapper.java | SongSharedReducer.java | SongShared.java

```java
package SongSharedPackage;

import java.io.IOException;
import java.util.*;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.Reducer.Context;

public class SongSharedReducer extends Reducer<IntWritable, IntWritable, IntWritable, IntWritable> {

     public void reduce(
             IntWritable trackId,
             Iterable<IntWritable> songShared,
             Reducer<IntWritable, IntWritable, IntWritable, IntWritable>.Context context)
             throws IOException, InterruptedException {

         List<Integer> songSharedList = new ArrayList<Integer>();

         for(IntWritable shared: songShared) {

             if(shared.get() == 1) {
                 songSharedList.add(shared.get());
             }
         }

         IntWritable size = new IntWritable(songSharedList.size());
         context.write(trackId, size);

     }
}
```

SongSharedMapper.java SongSharedReducer.java SongShared.java

```java
package SongSharedPackage;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

import UniqueListenersPackage.UniqueListeners;

public class SongShared {

    public static void main(String[] args) throws ClassNotFoundException, IOException, InterruptedException{
        // TODO Auto-generated method stub
        if (args.length != 2) {
            System.err.println("Usage: HeardFully <input path> <output path>");
            System.exit(-1);
          }

        //Job Related Configurations
        Configuration conf = new Configuration();
        Job job = new Job(conf, "No. of times a song was shared");
        job.setJarByClass(SongShared.class);

        // Specify the number of reducer to 1
        job.setNumReduceTasks(1);

        //Provide paths to pick the input file for the job
        FileInputFormat.setInputPaths(job, new Path(args[0]));

        //Provide paths to pick the output file for the job, and delete it if already present
        Path outputPath = new Path(args[1]);
        FileOutputFormat.setOutputPath(job, outputPath);
        outputPath.getFileSystem(conf).delete(outputPath, true);

        //To set the mapper and reducer of this job
        job.setMapperClass(SongSharedPackage.SongSharedMapper.class);
        job.setReducerClass(SongSharedPackage.SongSharedReducer.class);

        //set the input and output format class
        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);

        //set up the output key and value classes
        job.setOutputKeyClass(IntWritable.class);
        job.setOutputValueClass(IntWritable.class);

        //execute the job
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }

}
```

Input Command:

hadoop jar SongShared.jar /musicdata.txt /SongSharedOut

```
[acadgild@localhost ~]$ hadoop jar SongShared.jar /musicdata.txt /SongSharedOut
18/03/18 17:10:52 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes whe
re applicable
18/03/18 17:10:53 INFO client.RMProxy: Connecting to ResourceManager at localhost/127.0.0.1:8032
18/03/18 17:10:54 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface a
nd execute your application with ToolRunner to remedy this.
18/03/18 17:10:54 INFO input.FileInputFormat: Total input paths to process : 1
18/03/18 17:10:54 INFO mapreduce.JobSubmitter: number of splits:1
18/03/18 17:10:54 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1521362588389_0007
18/03/18 17:10:55 INFO impl.YarnClientImpl: Submitted application application_1521362588389_0007
18/03/18 17:10:55 INFO mapreduce.Job: The url to track the job: http://localhost:8088/proxy/application_1521362588389_0007/
18/03/18 17:10:55 INFO mapreduce.Job: Running job: job_1521362588389_0007
18/03/18 17:11:03 INFO mapreduce.Job: Job job_1521362588389_0007 running in uber mode : false
18/03/18 17:11:03 INFO mapreduce.Job:  map 0% reduce 0%
18/03/18 17:11:10 INFO mapreduce.Job:  map 100% reduce 0%
18/03/18 17:11:17 INFO mapreduce.Job:  map 100% reduce 100%
18/03/18 17:11:17 INFO mapreduce.Job: Job job_1521362588389_0007 completed successfully
```

Output: The reducer output shares the number of times a song was shared.

```
[acadgild@localhost ~]$ hadoop fs -cat /SongSharedOut/part-r-00000
18/03/18 17:11:34 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes whe
re applicable
222     0
223     0
225     2
```