

Session 4: MR -INTRODUCTION

Assignment 4

Problem Statement

We have a dataset of sales of different TV sets across different locations.

Records look like:

Samsung|Optima|14|Madhya Pradesh|132401|14200

The fields are arranged like:

Company Name|Product Name|Size in inches|State|Pin Code|Price

There are some invalid records which contain 'NA' in either Company Name or Product Name.

Task 1:

- Write a Map Reduce program to filter out the invalid records. Map only job will fit for this context.

Code to execute the task:

```
package task1_Assignment5;

import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class FilterInvalidRecords {

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

```

Configuration conf = new Configuration();

Job job = new Job(conf, "FilterInvalidRecords");

job.setJarByClass(FilterInvalidRecords.class);

job.setMapperClass(FilterInvalidRecordsMapper.class);

job.setOutputKeyClass(NullWritable.class);

job.setOutputValueClass(Text.class);

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

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

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

public static class FilterInvalidRecordsMapper extends
    Mapper<Object, Text, NullWritable, Text> {

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

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

        Text company = new Text(parts[0]);

        Text product = new Text(parts[1]);

        if (parts.length == 6 && !(company.toString().matches("NA") ||
        product.toString().matches("NA"))) {

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

        }

    }

}

```

Output:

There were 2 invalid records out of 18 total records in the given dataset, which can be seen in the job output as shown below:

```

Map-Reduce Framework
  Map input records=18
  Map output records=16
  Map output bytes=646
  Map output materialized bytes=684
  Input split bytes=122
  Combine input records=0
  Combine output records=0
  Reduce input groups=1
  Reduce shuffle bytes=684
  Reduce input records=16
  Reduce output records=16
  Spilled Records=32
  Shuffled Maps =1
  Failed Shuffles=0
  Merged Map outputs=1

```

```

You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$ hadoop fs -cat /user/acadgild/rt-r-00000
18/12/14 23:47:13 WARN util.NativeCodeLoader: Unable to load native library libhadoop: you may need to install the native library (see http://mr.apache.org/doc/FAQ#compile-runtime-errors-for-hadoop) or use the fallback-nativelibrary property to disable this warning
ry for your platform... using builtin-java classes where applicable
Samsung|Super|14|Maharashtra|619082|9200
Samsung|Super|14|Maharashtra|619082|9200
Samsung|Super|14|Maharashtra|619082|9200
Lava|Attention|20|Assam|454601|24200
Samsung|Decent|16|Kerala|922401|12200
Samsung|Optima|14|Madhya Pradesh|132401|14200
Zen|Super|14|Maharashtra|619082|9200
Lava|Attention|20|Assam|454601|24200
Onida|Decent|14|Uttar Pradesh|232401|16200
Onida|Lucid|18|Uttar Pradesh|232401|16200
Samsung|Optima|14|Madhya Pradesh|132401|14200
Zen|Super|14|Maharashtra|619082|9200
Lava|Attention|20|Assam|454601|24200
Akai|Decent|16|Kerala|922401|12200
Onida|Lucid|18|Uttar Pradesh|232401|16200
Samsung|Optima|14|Madhya Pradesh|132401|14200
[acadgild@localhost ~]$ █

```

Task 2:

- Write a Map Reduce program to calculate the total units sold for each Company.

Code written to execute the task:

```

package task1_Assignment5;

import java.io.IOException;
import java.util.HashSet;
import java.util.Set;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class UnitsPerCompany {

    private enum COUNTERS {
        INVALID_RECORD_COUNT
    }

    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();

        Job job = new Job(conf, "Units sold per company");
        job.setJarByClass(UnitsPerCompany.class);
    }
}

```

```

job.setMapperClass(UnitsPerCompanyMapper.class);
job.setReducerClass(UnitsPerCompanyReducer.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));
System.exit(job.waitForCompletion(true) ? 0 : 1);
org.apache.hadoop.mapreduce.Counters counters = job.getCounters();
System.out.println("No. of Invalid Records :")
    + counters.findCounter(COUNTERS.INVALID_RECORD_COUNT)
        .getValue();
}

```

```

public static class UnitsPerCompanyReducer extends
    Reducer< Text,IntWritable, Text, IntWritable> {

    public void reduce(
        Text company,
        Iterable<IntWritable> counts,
        Reducer<Text,IntWritable, Text, IntWritable>.Context context)
        throws IOException, InterruptedException {

        if(!(company.toString()).matches("NA")) {
            int i=0;
            for (IntWritable count : counts) {
                i+=count.get();
            }
            IntWritable size = new IntWritable(i);
            context.write(company, size);
        }
    }
}

```

```

public static class UnitsPerCompanyMapper extends
    Mapper<Object, Text,Text, IntWritable> {

    public void map(Object key, Text value,
        Mapper<Object, Text, Text,IntWritable>.Context context)
        throws IOException, InterruptedException {

        String[] parts = value.toString().split("[|]");
        Text company = new Text(parts[0]);
        Text product = new Text(parts[1]);
        IntWritable one = new IntWritable(1);
        if (parts.length == 6 && !(company.toString().matches("NA") ||
product.toString().matches("NA"))) {
            context.write(company,one);
        } else {
            // add counter for invalid records
            context.getCounter(COUNTERS.INVALID_RECORD_COUNT).increment(1L);
        }
    }
}

```

```
}  
}  
  
}
```

Output:

The output shows the units sold per company in different states, which sums to 16 as 2 records are invalid.

```
task1_Assignment5.UnitsPerCompany$COUNTERS  
INVALID_RECORD_COUNT=2  
You have new mail in /var/spool/mail/acadgild  
[acadgild@localhost ~]$ hadoop fs -cat /user/acadgild/hadoop/unitssold11/part-r-  
00000  
18/12/14 23:34:02 WARN util.NativeCodeLoader: Unable to load native-hadoop libra  
ry for your platform... using builtin-java classes where applicable  
Akai      1  
Lava      3  
Onida     3  
Samsung   7  
Zen       2
```

Task 3:

- Write a Map Reduce program to calculate the total units sold in each state for Onida company.

Code written to execute the task:

```
package task1_Assignment5;
```

```
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.io.Text;
```

```
import org.apache.hadoop.mapreduce.Job;
```

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

```
import org.apache.hadoop.mapreduce.Reducer;
```

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

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

```

public class Onidaperstate {

    private enum COUNTERS {
        INVALID_RECORD_COUNT
    }

    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();

        Job job = new Job(conf, "Units sold per company");
        job.setJarByClass(Onidaperstate.class);
        job.setMapperClass(OnidaperstateMapper.class);
        job.setReducerClass(OnidaperstateReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
        org.apache.hadoop.mapreduce.Counters counters = job.getCounters();
        System.out.println("No. of Invalid Records : "
            + counters.findCounter(COUNTERS.INVALID_RECORD_COUNT)
                .getValue());
    }

    public static class OnidaperstateReducer extends
        Reducer< Text,IntWritable, Text, IntWritable> {

        public void reduce(
            Text state,

```

```
    Iterable<IntWritable> units,  
    Reducer<Text,IntWritable, Text, IntWritable>.Context context)  
    throws IOException, InterruptedException {
```

```
    int i=0;  
    for (IntWritable unit : units) {  
        i+=unit.get();  
    }  
    IntWritable size = new IntWritable(i);  
    context.write(state, size);
```

```
}
```

```
}
```

```
public static class OnidaperstateMapper extends  
Mapper<Object, Text,Text, IntWritable> {
```

```
public void map(Object key, Text value,  
    Mapper<Object, Text, Text,IntWritable>.Context context)  
    throws IOException, InterruptedException {
```

```
String[] parts = value.toString().split("[|]");  
Text company = new Text(parts[0]);  
Text product = new Text(parts[1]);  
Text state = new Text(parts[3]);  
IntWritable one = new IntWritable(1);  
if (parts.length == 6  
    && !(company.toString().matches("NA") || product.toString().matches("NA"))  
    && company.toString().matches("Onida")) {
```

```

        context.write(state,one);
    } else {
        // add counter for invalid records
        context.getCounter(COUNTERS.INVALID_RECORD_COUNT).increment(1L);
    }

}

}

}

```

Output:

```

-r-000000
18/12/14 23:30:40 WARN util.NativeCodeLoader: Unable to load native
ry for your platform... using builtin-java classes where applicabl
Uttar Pradesh 3
[acaddgild@localhost ~]$ hadoop fs -ls /user/acaddgild/hadoop/

```