Project :- Patent Citation Data Manipulation .

Data Source: <http://www.nber.org/patents/>

**Pairwise citations data-** [Cite75\_99.txt](http://www.nber.org/patents/Cite75_99.txt)

Data size : 82 Mb

This project is divided into two pases

Phase 1: Our first program will take the patent citation data and *invert* it. For each patent,

we want to find and group the patents that cite it.

**Map reduce program for this is:**

package patient;

import java.io.IOException;

import java.util.Iterator;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.conf.Configured;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.FileInputFormat;

import org.apache.hadoop.mapred.FileOutputFormat;

import org.apache.hadoop.mapred.JobClient;

import org.apache.hadoop.mapred.JobConf;

import org.apache.hadoop.mapred.KeyValueTextInputFormat;

import org.apache.hadoop.mapred.MapReduceBase;

import org.apache.hadoop.mapred.Mapper;

import org.apache.hadoop.mapred.OutputCollector;

import org.apache.hadoop.mapred.OutputFormat;

import org.apache.hadoop.mapred.Reducer;

import org.apache.hadoop.mapred.Reporter;

import org.apache.hadoop.mapred.TextInputFormat;

import org.apache.hadoop.mapred.TextOutputFormat;

import org.apache.hadoop.util.Tool;

import org.apache.hadoop.util.ToolRunner;

public class MyJob extends Configured implements Tool {

public static class MapClass extends MapReduceBase

implements Mapper<Text, Text, Text, Text> {

public void map(Text key, Text value,

OutputCollector<Text, Text> output,

Reporter reporter) throws IOException {

output.collect(value, key);

}

}

public static class Reduce extends MapReduceBase

implements Reducer<Text, Text, Text, Text> {

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

OutputCollector<Text, Text> output,

Reporter reporter) throws IOException {

String csv = "";

while (values.hasNext()) {

if (csv.length() > 0) csv += ",";

csv += values.next().toString();

}

output.collect(key, new Text(csv));

}

}

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

Configuration conf = getConf();

JobConf job = new JobConf(conf, MyJob.class);

Path in = new Path("input/cite75\_99.txt");

Path out = new Path("out");

FileInputFormat.setInputPaths(job, in);

FileOutputFormat.setOutputPath(job, out);

job.setJobName("MyJob");

job.setMapperClass(MapClass.class);

job.setReducerClass(Reduce.class);

job.setInputFormat(KeyValueTextInputFormat.class);

job.setOutputFormat(TextOutputFormat.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

job.set("key.value.separator.in.input.line", ",");

JobClient.runJob(job);

return 0;

}

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

int res = ToolRunner.run(new Configuration(), new MyJob(), args);

System.exit(res);

}}

Generated Output is like :

"CITED" "CITING"

1 3964859,4647229

10000 4539112

100000 5031388

1000006 4714284

1000007 4766693

1000011 5033339

1000017 3908629

1000026 4043055

1000033 4190903,4975983

1000043 4091523

1000044 4082383,4055371

1000045 4290571

1000046 5918892,5525001

1000049 5996916

1000051 4541310

1000054 4946631

1000065 4748968

1000067 5312208,4944640,5071294

1000070 4928425,5009029

1000073 4107819,5474494

1000076 4867716,5845593

1000083 5566726,5322091

1000084 4683770,4182197

**Phase 2: Output file from 1st phase will act as input file for this phase.**

We will find the number of citations a patent has received. This too is counting. This is same as word count applied to this file again.

**package** patient;

**import** java.io.IOException;

**import** java.util.\*;

**import** org.apache.hadoop.fs.Path;

**import** org.apache.hadoop.conf.\*;

**import** org.apache.hadoop.io.\*;

**import** org.apache.hadoop.mapred.\*;

**import** org.apache.hadoop.util.\*;

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

**public** **static** **class** MapClass **extends** MapReduceBase

**implements** Mapper<Text, Text, IntWritable, IntWritable> {

**private** **final** **static** IntWritable *uno* = **new** IntWritable(1);

**private** IntWritable citationCount = **new** IntWritable();

**public** **void** map(Text key, Text value,

OutputCollector<IntWritable, IntWritable> output,

Reporter reporter) **throws** IOException {

citationCount.set(Integer.*parseInt*(value.toString()));

output.collect(citationCount, *uno*);

}

}

**public** **static** **class** Reduce **extends** MapReduceBase

**implements** Reducer<IntWritable,IntWritable,IntWritable,IntWritable>

{

**public** **void** reduce(IntWritable key, Iterator<IntWritable> values,

OutputCollector<IntWritable, IntWritable>output,

Reporter reporter) **throws** IOException {

**int** count = 0;

**while** (values.hasNext()) {

count += values.next().get();

}

output.collect(key, **new** IntWritable(count));

}

}

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

Configuration conf = getConf();

JobConf job = **new** JobConf(conf, PatientCitationCount.**class**);

FileInputFormat.*setInputPaths*(job, **new** Path("input/part-00000.txt"));

FileOutputFormat.*setOutputPath*(job, **new** Path("output"));

job.setJobName("PatientCitationCount");

job.setMapperClass(MapClass.**class**);

job.setReducerClass(Reduce.**class**);

job.setInputFormat(KeyValueTextInputFormat.**class**);

job.setOutputFormat(TextOutputFormat.**class**);

job.setOutputKeyClass(IntWritable.**class**);

job.setOutputValueClass(IntWritable.**class**);

JobClient.*runJob*(job);

**return** 0;

}

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

{

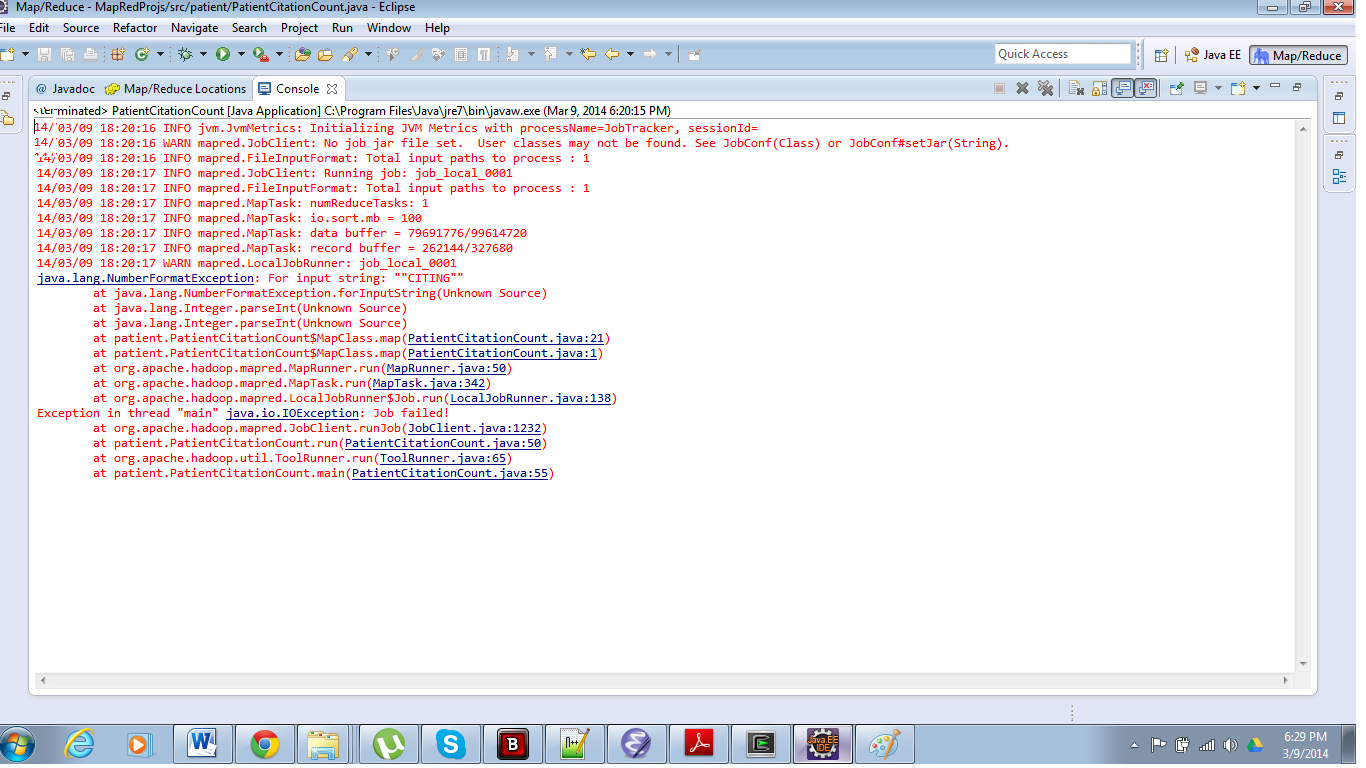
**int** res = ToolRunner.*run*(**new** Configuration(),**new** PatientCitationCount(),args);

System.*exit*(res);

}

}

**Right now I am struggling with format of output generated in phase 1 and then input to phase 2.**

**Error:** ****