package actualreq8;

import java.io.IOException;

import java.util.HashMap;

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

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

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;

import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

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

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

import org.apache.hadoop.fs.Path;

public class Req8 {

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

@Override

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

String line = value.toString();

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

String id = segments[0] + " " + segments[1];

String review = segments[2];

String words[] = review.replaceAll("[\\p{P}&&[^]]]", "").toLowerCase().split(" ");

for (String s : words) {

context.write(new Text(s), new Text(id));

}

}

}

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

@Override

public void reduce(Text key, Iterable<Text> values, Context context) throws IOException, InterruptedException {

HashMap map = new HashMap();

int count = 0;

for (Text t : values) {

String str = t.toString();

if (map != null && map.get(str) != null) {

count = (int)map.get(str);

map.put(str, ++count);

} else {

map.put(str, 1);

}

}

context.write(key, new Text(map.toString()));

}

}

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

// TODO Auto-generated method stub

Configuration conf= new Configuration();

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

job.setJarByClass(Req8.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

job.setMapperClass(Map.class);

job.setReducerClass(Reduce.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

job.setInputFormatClass(TextInputFormat.class);

job.setOutputFormatClass(TextOutputFormat.class);

Path outputPath = new Path(args[1]);

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

FileOutputFormat.setOutputPath(job, outputPath);

outputPath.getFileSystem(conf).delete(outputPath);

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

}

}