DPA – Lab 8

//LineCountMapOnly.java

public class LineCountMapOnly {

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

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

context.getCounter("Custom", "LinesCounted").increment(1);

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

}

}

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

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "line count map-only");

job.setJarByClass(LineCountMapOnly.class);

job.setMapperClass(LineMapper.class);

job.setNumReduceTasks(0);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(NullWritable.class);

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

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

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

}

}

//WordCountWithCombiner.java

public class WordCountWithCombiner {

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

private final static IntWritable one = new IntWritable(1);

private Text word = new Text();

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

context.getCounter("Custom", "LinesProcessed").increment(1);

StringTokenizer itr = new StringTokenizer(value.toString());

while (itr.hasMoreTokens()) {

word.set(itr.nextToken());

context.write(word, one);

}

}

}

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

public void reduce(Text key, Iterable<IntWritable> values, Context context)

throws IOException, InterruptedException {

int sum = 0;

for (IntWritable val : values) {

sum += val.get();

}

context.write(key, new IntWritable(sum));

}

}

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

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "word count with combiner");

job.setJarByClass(WordCountWithCombiner.class);

job.setMapperClass(TokenizerMapper.class);

job.setCombinerClass(IntSumReducer.class);

job.setReducerClass(IntSumReducer.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);

}

}

//ImageTypeCounter.java

public class ImageTypeCounter {

public static class LogMapper extends Mapper<LongWritable, Text, NullWritable, NullWritable> {

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

String line = value.toString().toLowerCase();

if (line.contains(".gif")) {

context.getCounter("ImageCounter", "gif").increment(1);

} else if (line.contains(".jpeg") || line.contains(".jpg")) {

context.getCounter("ImageCounter", "jpeg").increment(1);

} else if (line.contains(".png")) {

context.getCounter("ImageCounter", "png").increment(1);

} else {

context.getCounter("ImageCounter", "other").increment(1);

}

}

}

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

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "image type counter");

job.setJarByClass(ImageTypeCounter.class);

job.setMapperClass(LogMapper.class);

job.setNumReduceTasks(0); // Map-only job

job.setOutputKeyClass(NullWritable.class);

job.setOutputValueClass(NullWritable.class);

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

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

boolean success = job.waitForCompletion(true);

if (success) {

System.out.println("GIF Count: " + job.getCounters().findCounter("ImageCounter", "gif").getValue());

System.out.println("JPEG Count: " + job.getCounters().findCounter("ImageCounter", "jpeg").getValue());

System.out.println("PNG Count: " + job.getCounters().findCounter("ImageCounter", "png").getValue());

System.out.println("Other Count: " + job.getCounters().findCounter("ImageCounter", "other").getValue());

}

System.exit(success ? 0 : 1);

}

}