2. TopN

Driver-TopN.class

package samples.topn;  
  
import java.io.IOException;  
import java.util.StringTokenizer;  
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.lib.input.FileInputFormat;  
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;  
import org.apache.hadoop.util.GenericOptionsParser;  
  
public class TopN {  
 public static void main(String[] args) throws Exception {  
 Configuration conf = new Configuration();  
 String[] otherArgs = (new GenericOptionsParser(conf, args)).getRemainingArgs();  
 if (otherArgs.length != 2) {  
 System.err.println("Usage: TopN <in> <out>");  
 System.exit(2);  
 }   
 Job job = Job.getInstance(conf);  
 job.setJobName("Top N");  
 job.setJarByClass(TopN.class);  
 job.setMapperClass(TopNMapper.class);  
 job.setReducerClass(TopNReducer.class);  
 job.setOutputKeyClass(Text.class);  
 job.setOutputValueClass(IntWritable.class);  
 FileInputFormat.addInputPath(job, new Path(otherArgs[0]));  
 FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));  
 System.exit(job.waitForCompletion(true) ? 0 : 1);  
 }  
   
 public static class TopNMapper extends Mapper<Object, Text, Text, IntWritable> {  
 private static final IntWritable one = new IntWritable(1);  
   
 private Text word = new Text();  
   
 private String tokens = "[\_|$#<>\\^=\\[\\]\\\*/\\\\,;,.\\-:()?!\"']";  
   
 public void map(Object key, Text value, Mapper<Object, Text, Text, IntWritable>.Context context) throws IOException, InterruptedException {  
 String cleanLine = value.toString().toLowerCase().replaceAll(this.tokens, " ");  
 StringTokenizer itr = new StringTokenizer(cleanLine);  
 while (itr.hasMoreTokens()) {  
 this.word.set(itr.nextToken().trim());  
 context.write(this.word, one);  
 }   
 }  
 }  
}

TopNCombiner.class

package samples.topn;  
  
import java.io.IOException;  
import org.apache.hadoop.io.IntWritable;  
import org.apache.hadoop.io.Text;  
import org.apache.hadoop.mapreduce.Reducer;  
  
public class TopNCombiner extends Reducer<Text, IntWritable, Text, IntWritable> {  
 public void reduce(Text key, Iterable<IntWritable> values, Reducer<Text, IntWritable, Text, IntWritable>.Context context) throws IOException, InterruptedException {  
 int sum = 0;  
 for (IntWritable val : values)  
 sum += val.get();   
 context.write(key, new IntWritable(sum));  
 }  
}

TopNMapper.class

package samples.topn;  
  
import java.io.IOException;  
import java.util.StringTokenizer;  
import org.apache.hadoop.io.IntWritable;  
import org.apache.hadoop.io.Text;  
import org.apache.hadoop.mapreduce.Mapper;  
  
public class TopNMapper extends Mapper<Object, Text, Text, IntWritable> {  
 private static final IntWritable one = new IntWritable(1);  
   
 private Text word = new Text();  
   
 private String tokens = "[\_|$#<>\\^=\\[\\]\\\*/\\\\,;,.\\-:()?!\"']";  
   
 public void map(Object key, Text value, Mapper<Object, Text, Text, IntWritable>.Context context) throws IOException, InterruptedException {  
 String cleanLine = value.toString().toLowerCase().replaceAll(this.tokens, " ");  
 StringTokenizer itr = new StringTokenizer(cleanLine);  
 while (itr.hasMoreTokens()) {  
 this.word.set(itr.nextToken().trim());  
 context.write(this.word, one);  
 }   
 }  
}

TopNReducer.class

package samples.topn;  
  
import java.io.IOException;  
import java.util.HashMap;  
import java.util.Map;  
import org.apache.hadoop.io.IntWritable;  
import org.apache.hadoop.io.Text;  
import org.apache.hadoop.mapreduce.Reducer;  
import utils.MiscUtils;  
  
public class TopNReducer extends Reducer<Text, IntWritable, Text, IntWritable> {  
 private Map<Text, IntWritable> countMap = new HashMap<>();  
   
 public void reduce(Text key, Iterable<IntWritable> values, Reducer<Text, IntWritable, Text, IntWritable>.Context context) throws IOException, InterruptedException {  
 int sum = 0;  
 for (IntWritable val : values)  
 sum += val.get();   
 this.countMap.put(new Text(key), new IntWritable(sum));  
 }  
   
 protected void cleanup(Reducer<Text, IntWritable, Text, IntWritable>.Context context) throws IOException, InterruptedException {  
 Map<Text, IntWritable> sortedMap = MiscUtils.sortByValues(this.countMap);  
 int counter = 0;  
 for (Text key : sortedMap.keySet()) {  
 if (counter++ == 20)  
 break;   
 context.write(key, sortedMap.get(key));  
 }   
 }  
}  
  
4.MeanMax

MeanMaxDriver.class

package meanmax;  
  
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.lib.input.FileInputFormat;  
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;  
  
public class MeanMaxDriver {  
 public static void main(String[] args) throws Exception {  
 if (args.length != 2) {  
 System.err.println("Please Enter the input and output parameters");  
 System.exit(-1);  
 }   
 Job job = new Job();  
 job.setJarByClass(MeanMaxDriver.class);  
 job.setJobName("Max temperature");  
 FileInputFormat.addInputPath(job, new Path(args[0]));  
 FileOutputFormat.setOutputPath(job, new Path(args[1]));  
 job.setMapperClass(MeanMaxMapper.class);  
 job.setReducerClass(MeanMaxReducer.class);  
 job.setOutputKeyClass(Text.class);  
 job.setOutputValueClass(IntWritable.class);  
 System.exit(job.waitForCompletion(true) ? 0 : 1);  
 }  
}  
  
MeanMaxMapper.class

package meanmax;  
  
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;  
  
public class MeanMaxMapper extends Mapper<LongWritable, Text, Text, IntWritable> {  
 public static final int MISSING = 9999;  
   
 public void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text, IntWritable>.Context context) throws IOException, InterruptedException {  
 int temperature;  
 String line = value.toString();  
 String month = line.substring(19, 21);  
 if (line.charAt(87) == '+') {  
 temperature = Integer.parseInt(line.substring(88, 92));  
 } else {  
 temperature = Integer.parseInt(line.substring(87, 92));  
 }   
 String quality = line.substring(92, 93);  
 if (temperature != 9999 && quality.matches("[01459]"))  
 context.write(new Text(month), new IntWritable(temperature));   
 }  
}  
  
MeanMaxReducer.class

package meanmax;  
  
import java.io.IOException;  
import org.apache.hadoop.io.IntWritable;  
import org.apache.hadoop.io.Text;  
import org.apache.hadoop.mapreduce.Reducer;  
  
public class MeanMaxReducer extends Reducer<Text, IntWritable, Text, IntWritable> {  
 public void reduce(Text key, Iterable<IntWritable> values, Reducer<Text, IntWritable, Text, IntWritable>.Context context) throws IOException, InterruptedException {  
 int max\_temp = 0;  
 int total\_temp = 0;  
 int count = 0;  
 int days = 0;  
 for (IntWritable value : values) {  
 int temp = value.get();  
 if (temp > max\_temp)  
 max\_temp = temp;   
 count++;  
 if (count == 3) {  
 total\_temp += max\_temp;  
 max\_temp = 0;  
 count = 0;  
 days++;  
 }   
 }   
 context.write(key, new IntWritable(total\_temp / days));  
 }  
}