实验三：倒排索引实验报告

小组成员：MF20330007 陈明远 MF20330003 陈迪 MG20330095 朱志威

- **设计思路：**

本次实验通过输入小说文件来获取词频，其中 Map 部分的输入为 line，输出为((world, bookname),1)，然后通过自己设计的 combiner 将临时文件整合在一起，形成((world, bookname), sum),由于 key 值也是一个键值对，所以要重写 partitioner 函数，使得里面的 hash 函数的 key 只对 world 起作用，接着通过 reduce 函数将相同词的 bookname 和 sum 整合在一起，计算出平均数，然后输入所有 bookname 和出现次数。代码如下：

- InvertedIndex 代码：

```
1. import org.apache.hadoop.conf.Configuration;
2. import org.apache.hadoop.fs.Path;
3. import org.apache.hadoop.io.IntWritable;
4. import org.apache.hadoop.io.Text;
5. import org.apache.hadoop.mapreduce.Job;
6. import org.apache.hadoop.mapreduce.Mapper;
7. import org.apache.hadoop.mapreduce.Reducer;
8. import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
9. import org.apache.hadoop.mapreduce.lib.input.FileSplit;
10. import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
11. import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
12. import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
13. import org.apache.hadoop.mapreduce.lib.partition.HashPartitioner;
14. import org.apache.hadoop.util.GenericOptionsParser;
15.
16. import java.io.IOException;
17. import java.util.ArrayList;
```

```
18. import java.util.List;
19. import java.util.StringTokenizer;
20. 
21. public class InvertedIndex {
22.     public static class InvertedIndexMapper extends Mapper<Object, Text, Tex
    t, IntWritable> {//自定义自己的 mapper，输入的 key 是小说文件，输出一个单词和词
    频
23.         @Override
24.         protected void map(Object key, Text value, Context context)
25.          throws IOException, InterruptedException {
26.             FileSplit fileSplit = (FileSplit) context.getInputSplit();
27.             String filename = fileSplit.getPath().getName(); //获取文件的名
    字
28.             String line = value.toString().toLowerCase();
29.             StringTokenizer itr = new StringTokenizer(line);
30.             while (itr.hasMoreTokens()) {
31.                 Text word = new Text();
32.                 String temp = itr.nextToken();
33.                 if (filename.indexOf(".txt.segmented") > 0) {//因为输出要去
    掉.txt.segmented
34.                     word.set(temp+ "," + filename.substring(0, filename.inde
    xOf(".txt.segmented")));
35.                     context.write(word, new IntWritable(1));
36.                 } else {
37.                     word.set(temp + "," + filename.substring(0, filename.ind
    exOf(".TXT.segmented")));
38.                     context.write(word, new IntWritable(1));//只有写入 contex
    才会生效
39.                 }
40.             }
41.         }
42. 
43.     }
44. 
45.     public static class SumCombiner extends Reducer<Text, IntWritable, Text,
     IntWritable> {//定义自己的 combiner
46.         private IntWritable result = new IntWritable();
47.         @Override
48.         public void reduce(Text key, Iterable<IntWritable> values, Context c
    ontext)
49.                 throws IOException, InterruptedException {
50.             int sum = 0;
51.             for (IntWritable val : values) {
52.                 sum += val.get();
```

```
            }
            result.set(sum);
            context.write(key, result);
        }
    }

    public static class NewPartitioner extends HashPartitioner<Text, IntWritable> {//定义自己的 partitioner，重写哈希函数
        @Override
        public int getPartition(Text key, IntWritable value, int numReduceTasks) {
            String term = new String();
            term = key.toString().split(",")[0];//将哈希函数的 key 变为词，key[0]为词
            return super.getPartition(new Text(term), value, numReduceTasks);
        }
    }

    public static class InvertedIndexReducer extends Reducer<Text, IntWritable, Text, Text> {
        private Text word1 = new Text();
        private Text word2 = new Text();
        String bookname = new String();
        static Text CurrentItem = new Text(" ");
        static List<String> postingList = new ArrayList<String>();

        @Override
        public void reduce(Text key, Iterable<IntWritable> values, Context context)
                throws IOException, InterruptedException {
            int sum = 0;
            word1.set(key.toString().split(",")[0]);
            bookname = key.toString().split(",")[1];
            for (IntWritable val : values) {
                sum += val.get();
            }//将不同的 map 中统计的词频加在一起
            word2.set(bookname + ":" + sum);
            if (!CurrentItem.equals(word1) && !CurrentItem.equals(" ")) {//相同词出现在不同小说中要整合在一起。
                StringBuilder out = new StringBuilder();
                long count = 0;
                for (String p : postingList) {
                    out.append(p);
```

```
90.                     out.append(";");
91.                     count += Long.parseLong(p.substring(p.indexOf(":") + 1))
    ;//将不同小说中词频数加载一起，目的是为了求平均值
92.                 }
93.                 StringBuilder out1 = new StringBuilder(String.format("%.2f",
     ((double) count / postingList.size()))).append("," + out);//求平均值
94.                 if (count > 0) {
95.                     context.write(word1, new Text(out1.substring(0, out1.las
    tIndexOf(";")).toString()));
96.                 }
97.                 postingList = new ArrayList<String>();
98.             }
99.             CurrentItem = new Text(word1);
100.              postingList.add(word2.toString());
101.         }
102.
103.         @Override
104.         protected void cleanup(Context context) throws IOException, Interru
    ptedException {//处理最后一个词
105.             StringBuilder out = new StringBuilder();
106.             long count = 0;
107.             for (String p : postingList) {
108.                 out.append(p);
109.                 out.append(";");
110.                 count += Long.parseLong(p.substring(p.indexOf(":") + 1));
111.             }
112.             StringBuilder out1 = new StringBuilder(String.format("%.2f", ((
    double) count / postingList.size()))).append("," + out);
113.             if (count > 0) {
114.                 context.write(word1, new Text(out1.substring(0, out1.lastIn
    dexOf(";")).toString()));
115.             }
116.         }
117.     }
118.
119.     public static void main(String[] args) throws Exception {
120.         Configuration conf = new Configuration();
121.         String[] otherArgs = new GenericOptionsParser(conf, args).getRemain
    ingArgs();
122.         if (otherArgs.length != 2) {
123.             System.err.println("Usage:  input output");
124.             System.exit(2);
125.         }
126.         Job job = Job.getInstance(conf, "InvertedIndex");//建立任务
```

```
127.         job.setJarByClass(InvertedIndex.class);
128.         job.setMapOutputKeyClass(Text.class);
129.         job.setMapOutputValueClass(IntWritable.class);
130.         job.setInputFormatClass(TextInputFormat.class);
131.         job.setOutputFormatClass(TextOutputFormat.class);
132.         job.setMapperClass(InvertedIndexMapper.class);
133.         job.setCombinerClass(SumCombiner.class);
134.         job.setPartitionerClass(NewPartitioner.class);
135.         job.setReducerClass(InvertedIndexReducer.class);
136.         job.setOutputKeyClass(Text.class);
137.         job.setOutputValueClass(Text.class);
138.         FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
139.         FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
140.         System.exit(job.waitForCompletion(true) ? 0 : 1);
141.     }
142. }
```

最终输出的部分结果和“江湖”和“风雪”两个单词的词频如下：

- 部分结果：

图 1:部分输出结果(1)

图 2:部分输出结果(2)

图 3:“江湖”输出结果(1)

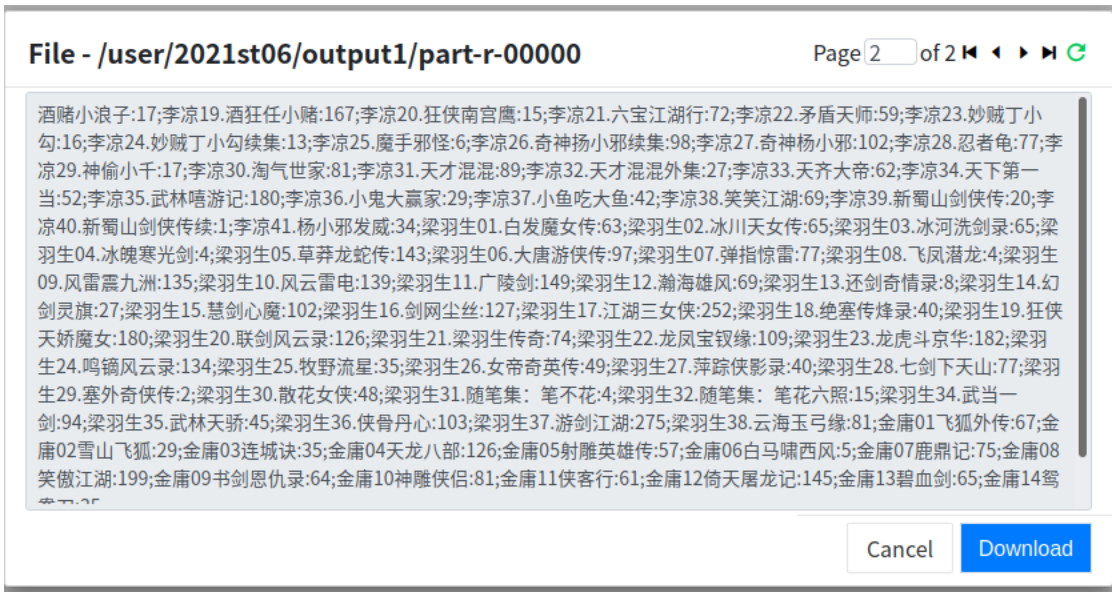

图 4:“江湖”输出结果(2)

图 5:“风雪”输出结果

- 输出结果路径：/user/2021st06
- WebUI 报告：