

# 作业 6 MapReduce之倒排索引

191870105 刘婷

题目：在作业5的数据集基础上完成莎士比亚文集单词的倒排索引，输出按字典序对单词进行排序，单词的索引按照单词在该文档中出现的次数从大到小排序。单词忽略大小写，忽略标点符号（punctuation.txt），忽略停词（stop-word-list.txt），忽略数字，单词长度 $\geq 3$ 。输出格式如下：

单词1: 文档i#频次a, 文档j#频次b...

单词2: 文档m#频次x, 文档n#频次y...

...

【注】作业提交方式：git仓库地址或者相关文件的zip包

## 一、代码思路：InvertedIndexer.class

该代码以原有的倒排索引为基础，进行简单的修改完成作业6的任务

### 1.FileNameInputFormat

InputFormat类的作用有：

- (1)验证作业数据的输入形式和格式
- (2)将输入数据分割若干个逻辑意义上的InputSplit
- (3)提供一个RecordReader用于将Mapper的输入处理转化为若干输入记录。

FileNameInputFormat类是一个InputFormat类，该类的作用就是将原来FileInputFormat提供的RecordReader重写为自定义FileNameRecordReader类。FileNameRecordReader定义如下所示。

### 2.FileNameRecordReader:

自定义FileNameRecordReader类，将原来的LineRecordReader中的函数重写，对于原有的LineRecordReader来说，它以偏移值为key，以一行为value。FileNameRecordReader中，以文件名为key，一行为value，这样在传给Mapper时，Mapper可以知道该行数据所在的文件名，并把文件名纳入输出的<key,value>的key中，供reduce使用。

### 3.InvertedIndexMapper

该Class和作业5中的class基本一致，包括过滤停词、替换符号、忽略数字和字符长度大于等于3。另外，作业5中，我在map过程中读取当前文件的文件名写入key中，由于该代码已经通过FileNameInputFormat类将文件名传入key中，所以该步骤在作业6中也删去。

## 4.SumCombiner

使用Combiner将Mapper的输出结果中value部分的词频进行统计，减少reduce节点的任务。

## 5.NewPartitioner

自定义HashPartitioner，保证 <word, filename>格式的key值按照word分发给Reducer，这样才能保证同一个word的键值对发送给同一个reduce节点。以便按照后续思路完成任务。

## 6.InvertedIndexReducer:

思路如注释所示。

```
//变量
private Text word1 = new Text();    // 存储当前key中的单词
private Text word2 = new Text();
// 存储当前单词在文件中出现次数 + 该文件名
String temp = new String(); // 存储文件名
static Text CurrentItem = new Text(" "); //用于判断是否当前单词已经全部统计完，跳到第二次单词进行新的统计。
static List<String> postingList = new ArrayList<String>(); //用于存放单词所曾出现过的所有文件名及词频，便于写入文件。
```

//步骤

STEP1：对当前key:word#filename进行词频统计，获得该单词在该文件中出现词频。

STEP2：判断是否前一个单词已经全部统计完：即由于NewPartitioner让key中word相同的放于同一个reduce节点，所以在单个reduce节点上，<key,value>的顺序为<word1#file1,1><word1#file1,1><word1#file2,1>;<word1#file2,1>;<word1#file3,1>;<word2#file1>;<word2#file2>;<word3#file1,1> 因此只有word1全部统计完毕，才会统计到word2。所以可以用该方法进行判断。

STEP3：如果仍在word1中，则将“filename#词频”不断add进postingList。如果已经到word2，则将之前的postingList 按照词频从大到小排序,这里使用冒泡排序法，代码如下所示，接着全部读出进行context.write操作并清空postingList，进行新一轮对于word2的操作。

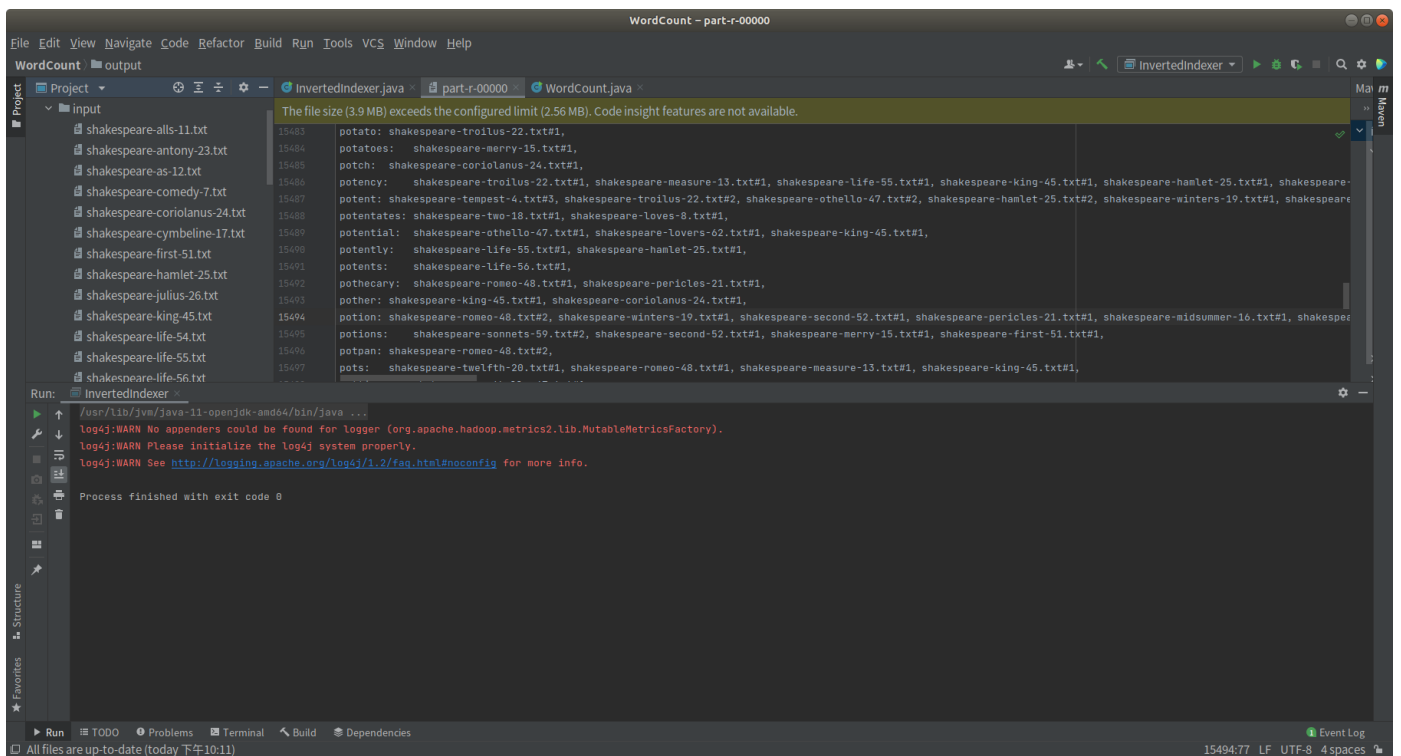
```

for (int i = 0; i < str_length-1; i++){
    for (int j = 0; j < str_length-1-i; j++){
        String temp = "";
        int sum_j =
Integer.valueOf(postingList.get(j).substring(postingList.get(j).indexOf("<")+1,
postingList.get(j).indexOf(", ")));
        int sum_k =
Integer.valueOf(postingList.get(j+1).substring(postingList.get(j+1).indexOf("<")+1,
postingList.get(j+1).indexOf(", ")));
        if (sum_j < sum_k) {
            temp = postingList.get(j);
            postingList.set(j, postingList.get(j+1));
            postingList.set(j+1, temp);
        }
    }
}

```

## 二、运行截图

### 1.单机运行



### 2.分布式运行

```
holmes@holmes-Lenovo-XiaoXin-Air-14IKBR: /usr/local/hadoop
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

Total time spent by all reduce tasks (ms)=49312
Total vcore-milliseconds taken by all map tasks=307549
Total vcore-milliseconds taken by all reduce tasks=49312
Total megabyte-milliseconds taken by all map tasks=314930176
Total megabyte-milliseconds taken by all reduce tasks=50495488

Map-Reduce Framework
  Map input records=158963
  Map output records=422310
  Map output bytes=15326055
  Map output materialized bytes=4797447
  Input split bytes=5187
  Combine input records=422310
  Combine output records=122919
  Reduce input groups=122919
  Reduce shuffle bytes=4797447
  Reduce input records=122919
  Reduce output records=23596
  Spilled Records=245838
  Shuffled Maps =40
  Failed Shuffles=0
  Merged Map outputs=40
  GC time elapsed (ms)=4871
  CPU time spent (ms)=118630
  Physical memory (bytes) snapshot=12357054464
  Virtual memory (bytes) snapshot=115589529600
  Total committed heap usage (bytes)=8550088704
  Peak Map Physical memory (bytes)=348221440
  Peak Map Virtual memory (bytes)=2834673664
  Peak Reduce Physical memory (bytes)=243417088
  Peak Reduce Virtual memory (bytes)=2836393984

Shuffle Errors
  BAD_ID=0
  CONNECTION=0
  IO_ERROR=0
  WRONG_LENGTH=0
  WRONG_MAP=0
  WRONG_REDUCE=0

File Input Format Counters
  Bytes Read=5020327
File Output Format Counters
  Bytes Written=3901434

holmes@holmes-Lenovo-XiaoXin-Air-14IKBR: /usr/local/hadoop$
```




### 3.9870端口文件结构

Your endpoint × java.net.Conn × (38条消息) ja × LineRedcord × (38条消息) Li × cslabcms.nju × 作业 × 191870105-刘婷 × Browsing HDFS × All Applications ×






localhost:9870/explorer.html#/user/holmes

Hadoop Overview Datanodes Datanode Volume Failures Snapshot Startup Progress Utilities

### Browse Directory

/user/holmes Go!   

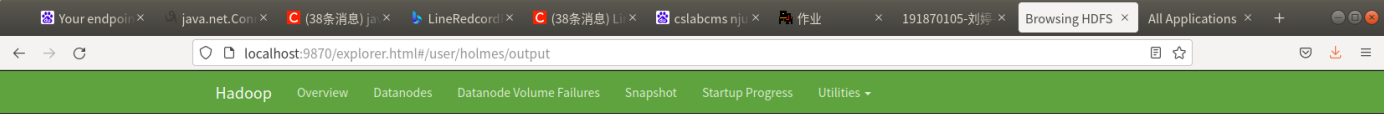
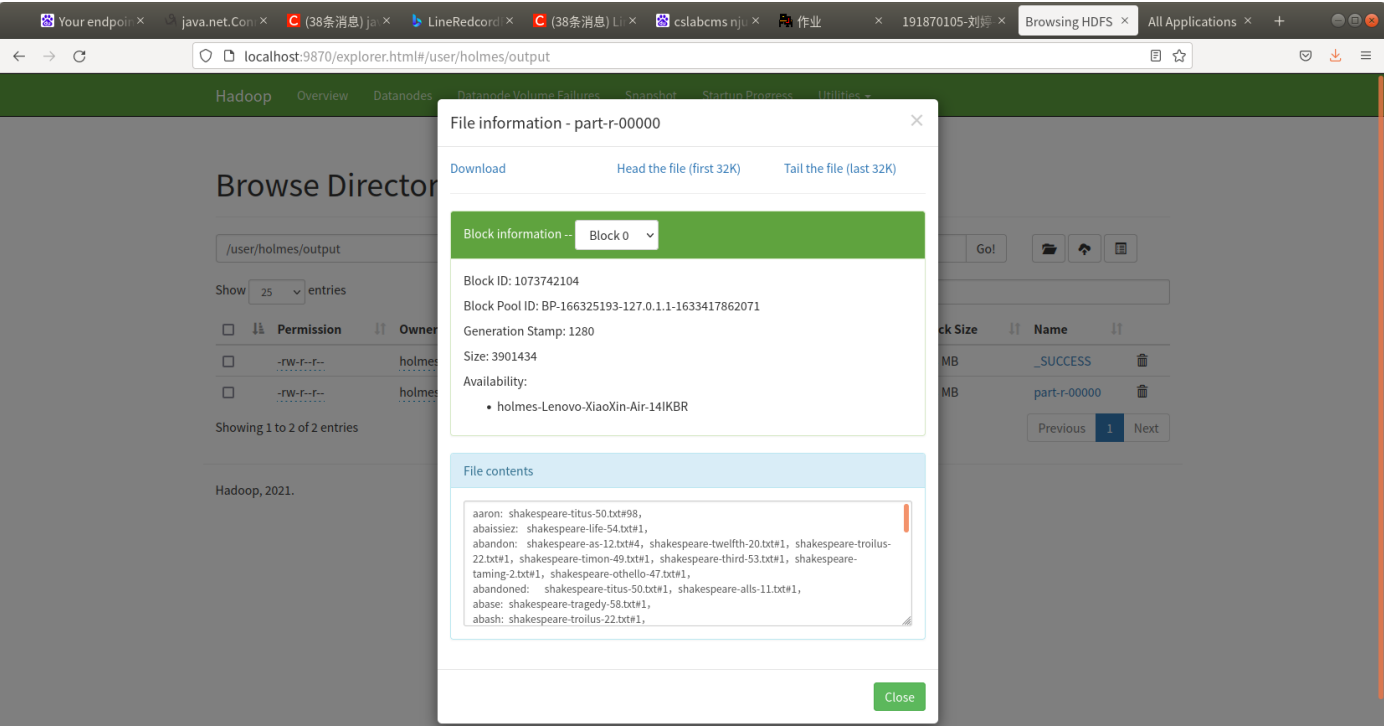
Show 25 entries Search:

<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	
<input type="checkbox"/>	drwxr-xr-x	holmes	supergroup	0 B	Oct 29 21:50	0	0 B	input	
<input type="checkbox"/>	drwxr-xr-x	holmes	supergroup	0 B	Nov 02 11:42	0	0 B	output	
<input type="checkbox"/>	-rw-r--r--	holmes	supergroup	98 B	Oct 29 21:51	1	128 MB	punctuation.txt	
<input type="checkbox"/>	-rw-r--r--	holmes	supergroup	2.18 KB	Oct 29 21:51	1	128 MB	stop-word-list.txt	

Showing 1 to 4 of 4 entries Previous 1 Next

Hadoop, 2021.

### 4.output结果文件



## 5.8088端口运行结果

hadoop

Cluster

About

Nodes

Node Labels

Applications

NEW

NEW SAVING

SUBMITTED

ACCEPTED

RUNNING

FINISHED

FAILED

KILLED

Scheduler

Tools

All Applications

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Used Resources	Total Resources	Reserved Resources
1	0	0	1	0	<memory:0, vCores:0>	<memory:8192, vCores:8>	<memory:0, vCores:0>

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes
1	0	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory:1024, vCores:1>	<memory:8192, vCores:4>

Show 20 entries

ID	User	Name	Application Type	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB	Allocated GPUs	Reserved CPU VCores
application_1635824333762_0001	holmes	inverted index	MAPREDUCE	default	0	Tue Nov 2 11:40:51 +0800 2021	Tue Nov 2 11:40:52 +0800 2021	Tue Nov 2 11:42:08 +0800 2021	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/A

Showing 1 to 1 of 1 entries

### 三、总结分析

扩展性：

1. 传入文件即可进行统计。
2. 只能传入两个停词文件，且按照一定顺序传入，因为stop文件和punc文件处理方式不同。
3. 没有 -skip参数，也无法正常运行。
4. 自定义FileNameInputFormat，FileNameRecordReader类，在后续修改代码时将更加方便。
5. 按照词频排序时，我采用的方式是将词频写在前面，使用list自身的排序功能完成按照词频排序的功能，但是如果题目要求按照文件名排序，则需要多处修改代码，如果采用自定义的比较类，规定按照哪个部分进行排序，代码的扩展性会更强。

性能：

1. 自定义SumCombiner和NewPartitioner，不仅保证程序运行的正确性，也避免了相同word被发到不同的reduce节点上，造成不同reduce节点之间的通信浪费。
2. 使用FileNameInputFormat，FileNameRecordReader类，无需在map节点获取当前文件地址，读取当前文件名，减少map节点的工作负担。