

## 大数据实验2—倒排索引 实验报告

陈越琦

(121160005 Yueqichen.0x0@gmail.com)

刘威

(131220085 liuwei13cs@smail.nju.edu.cn)

杨杰才

(131220115 mark\_grove@qq.com)

周子博

(121250229 441842096@qq.com)

**摘要:** 本次实验我们小组通过课堂上介绍的“带词频属性的文档倒排算法”，统计了词语的倒排索引并输出，按照要求输出了词语的平均出现次数，使用给定的小说数据集在集群上完成了调试和测试。在此基础上，完成了两个选做任务：基于平均出现次数对词语的全局排序与计算出每位作家每个词语的TF-IDF并输出。实验分别在本地进行了简单测试并提交到集群运行获得最终运行结果。在得到实验结果后，我们将输出结果下载到本地，然后对倒排索引的结果进行了简要的分析，得到了一些比较有趣的结论。

**关键词:** Hadoop、倒排索引、全局排序、TF-IDF

### §1. 引言

倒排索引是文档检索系统中最常用的数据结构，被广泛的应用于全文搜索引擎。它主要用来存储某个单词（或词组），在一个文档或一组文档中的存储位置的映射，即提供了一种根据内容来查找文档的方式，由于不是根据文档来确定文档所包含的内容，而是进行了相反的操作，因而被称为倒排索引。在本次实验中，我们实现了(1)带词频属性的文档倒排算法。除了实验任务外，我们还设计了两个MapReduce Job完成选做内容：(2)根据每个词语的平均出现次数进行全局排序；(3)为每位作家、计算每个词语的TF-IDF。

实验报告的第2节简要介绍了实验环境和完成情况。第3节中将详细介绍实验各个部分的设计。测试与运行的结果留在第4节中展示。在第5节中总结实验内容和团队合作。

\*

---

\*陈越琦：121160005 完成代码实现并调试测试 刘威：131220085 完成实验报告  
杨杰才：121160005 帮助刘威完成实验报告 周子博：121250229 帮助陈越琦实现集群测试

## 目录

§1. 引言	1
§2. 实验环境与概述	3
§3. 实验具体设计	3
3.1 带词频属性的文档倒排算法 . . . . .	3
3.1.1 设计思路 . . . . .	3
3.1.2 相关伪代码 . . . . .	4
3.2 基于平均出现次数对词语进行全局排序 . . . . .	6
3.2.1 设计思路 . . . . .	6
3.2.2 相关伪代码 . . . . .	6
3.3 为每位作家、计算每个词语的TF-IDF . . . . .	7
3.3.1 设计思路 . . . . .	7
3.3.2 相关伪代码 . . . . .	8
§4. 实验测试与运行结果	10
4.1 JAR包执行方式说明 . . . . .	10
4.2 实验运行结果展示 . . . . .	10
4.3 一些有趣的发现 . . . . .	18
§5. 实验总结	19
5.1 实验内容总结 . . . . .	19
5.2 团队合作总结 . . . . .	19

## §2. 实验环境与概述

本次实验的本地开发与测试环境如下：

软件	版本
OS	Ubuntu-15.04
JDK	JDK 7u80
Hadoop	0.20.205.0 & 2.7.1

图 1 开发与测试环境

本次实验，我们完成了三个MapReduce任务的编写：

1. 带词频属性的文档倒排算法。
2. 基于平均出现次数对词语进行全局排序。
3. 为每位作家、计算每个词语的TF-IDF。

## §3. 实验具体设计

### 3.1 带词频属性的文档倒排算法

#### 3.1.1 设计思路

为完成InvertedIndex任务我们分别实现了五个类： InvertedIndex, InvertedIndexMapper, InvertedIndexReducer, NewPartitioner, SumCombiner.

1. InvertedIndex是主类，负责配置并启动作业。
2. InvertedIndexMapper是Map类，输入为<Object ,Text>键值对，输出为<Text, IntWritable>键值对。从输入的split中获得文本的行，切割行中每一词语，并加上小说名作为键。
3. SumCombiner是Combine类，对中间结果中的词频进行累加，减少中间数据的传输。
4. NewPartitioner中重新定义getPartitioner方法覆盖父类中同名方法。根据词语进行分区而不是Map结果中的词语小说名组合。避免后者可能导致的错误：同一个词语的键值对可能被分到不同Reducer。
5. InvertedIndexReducer是Reduce类，根据不同词语统计出最终结果并计算输出平均出现次数和相应文档信息。

MapReduce各阶段的K,V类型见下表：

Map			Combiner	Reduce
输入	key	当前行的offset	词语+文档id	combiner的结果key
	value	行中内容	词频 (1)	一个Inputspli中某词语出现的次数汇总
输出	key	词语+文档id	词语+文档id	词语
	value	词频 (1)	一个Inputspli中某词语出现的次数汇总	文档id+词频

图 2 作业各阶段的键值

Map			Combiner	Reduce
输入	key	当前行的offset	浩浩#金庸05射雕英雄传	浩浩#金庸05射雕英雄传
	value	钱塘江 浩浩 江水 .....	1	14
输出	key	浩浩#金庸05射雕英雄传	浩浩#金庸05射雕英雄传	浩浩
	value	1	14	金庸05射雕英雄传:14; 古龙04 .....

图 3 作业各阶段的键值例子

### 3.1.2 相关伪代码

#### Mapper:

---

##### Class 1 class Mapper

---

**Input :** 文本文件的行

**Output :** 键值对 (词语#文档名, 词频)

```

1: procedure MAP(Object key, Text line)
2:   获得文件名为fileName
3:   for all term t in line l do
4:     Emit(t#fileName, one)
5:   end for
6: end procedure

```

---

#### Combiner:

---

##### Class 2 class Combiner

---

**Input :** 键值对 (词语#文档名, 词频)

**Output :** 键值对 (词语#文档名, 词频)

```

1: procedure COMBINE(Text key, Iterable<IntWritable> value)
2:   Sum = 0
3:   for all val in value do do
4:     Sum ← Sum + val
5:   end for
6:   Emit(key, Sum)
7: end procedure

```

---

**NewPartitioner:****Class 3 class NewPartitioner**


---

```

1: procedure GETPARTITION(Text key, IntWritable value, int NumReduceTasks)
2:   term  $\leftarrow$  key.split("#")[0]
3:   return super.getPartition(term, value, NumReduceTasks)
4: end procedure

```

---

**Reducer:****Class 4 class Reducer**


---

**Input :** Combiner的结果<Text key, Iterable<IntWritable> >

**Output :** 词语 平均出现次数 [小说名: 词频]

```

1: procedure SETUP( )
2:    $t_{prev} \leftarrow \emptyset$ 
3:    $P \leftarrow newPostingsList$ 
4: end procedure
5: procedure REDUCE(tuple< $t, n$ >,  $tf[f]$ )
6:   if  $t \neq t_{prev}$  &&  $t_{prev} \neq \emptyset$  then
7:     计算平均出现次数并插入到 $P$ 的头部
8:     Emit( $t_{prev}, P$ )
9:      $P.Reset()$ 
10:     $P.Add(<n, f>)$ 
11:     $t_{prev} \leftarrow t$ 
12: end procedure
13: procedure CLOSE( )
14:   计算平均出现次数并插入到 $P$ 的头部
15:   Emit( $t, P$ )
16: end procedure

```

---

### 3.2 基于平均出现次数对词语进行全局排序

#### 3.2.1 设计思路

对词语平均出现次数进行全局排序是基于带词频属性的倒排索引的输出结果来操作的。为完成这个任务，我们分别实现了ResultSort与ResultSortMapper两个类。

1. ResultSort是主类，负责配置并启动作业。在其中设置Reducer数量为1，从而控制作业的最终输出文件数为1，使得部分排序变为全局排序。
2. ResultSortMapper是Map类，输入为输入为<Object ,Text>键值对，输出为<DoubleWritable, Text>键值对。从倒排文件中获得输入并切割得到每个词语的平均出现次数，以此为键，输入的剩下内容为值。

MapReduce各阶段的K,V类型见下表(因为使用默认的IdentityReducer类，所以忽视Reduce阶段的键值对)：

Map		
输入	key	当前行的offset
	value	行中内容
输出	key	平均出现次数
	value	行中内容

图 4 作业各阶段的键值

Map		
输入	key	当前行的offset
	value	海瑞 455.0 李凉09 .....
输出	key	455
	value	海瑞 455.0 李凉09 .....

图 5 作业各阶段的键值例子

#### 3.2.2 相关伪代码

Mapper的伪代码如下：

---

Class 5 class Mapper

---

Input : 倒排索引记录

Output : 键值对 (平均出现次数, 倒排索引记录)

```

1: procedure MAP(InvertedIndexRecord record)
2:   average ← Word frequency in record
3:   Emit(average, record)
4: end procedure

```

---

### 3.3 为每位作家、计算每个词语的TF-IDF

#### 3.3.1 设计思路

TF-IDF计算在一定程度上与InvertedIndex的设计思想相似。

为完成TFIDF任务我们分别实现了五个类： TFIDF, TFIDFMapper, TFIDFReducer, NewPartitioner, SumCombiner.

1. TFIDF是主类，负责配置并启动作业。
2. TFIDFMapper是Map类，输入为<Object ,Text>键值对，输出为<Text, IntWritable>键值对。从输入的split中获得文本的行，切割行中每一词语，并加上作者名和小说名作为键。
3. SumCombiner是Combine类，对中间结果中的词频进行累加，减少中间数据的传输。
4. NewPartitioner中重新定义getPartitioner方法覆盖父类中同名方法。根据词语作者名的组合进行分区而不是Map结果中的词语作者名小说名组合。避免后者可能导致的错误：同一个词语作者名的键值对可能被分到不同Reducer。也不是单一的根据词语进行分区，避免同一个词语同一个作者不是连续出现。
5. TFIDFReducer是Reduce类，根据不同词语作者名统计出最终结果并计算输出TF和IDF。

MapReduce各阶段的K,V类型见下表：

	Map		Combiner	Reduce
输入	key	当前行的offset	词语+作者+小说名	combiner的结果key
	value	行中具体内容	词频 (1)	一个Inputsplit中某词语出现的次数汇总
输出	key	词语+作者+小说名	词语+作者+小说名	作者
	value	词频 (1)	一个Inputsplit中某词语出现的次数汇总	词语+TF值+IDF值

图 6 作业各阶段的键值

	Map		Combiner	Reduce
输入	key	当前行的offset	浩浩#金庸#射雕英雄传	浩浩#金庸#射雕英雄传
	value	钱塘江 浩浩 江水 .....	1	23
输出	key	浩浩#金庸#射雕英雄传	浩浩#金庸#射雕英雄传	金庸
	value	1	23	浩浩, 12, 0.5108

图 7 作业各阶段的键值例子

### 3.3.2 相关伪代码

**Mapper:**

---

**Class 6** *class Mapper*

---

**Input :** 文本文件的行

**Output :** 键值对 (词语#作者名#小说名, 词频)

```

1: procedure MAP(Object key, Text value)
2:   author  $\leftarrow$  value.split()
3:   fileName  $\leftarrow$  value.split()
4:   for all term t in line l do
5:     Emit(t#author#fileName, one)
6:   end for
7: end procedure

```

---

**Combiner:**

---

**Class 7** *class Combiner*

---

**Input :** 键值对 (词语#作者名#文档名, 词频)

**Output :** 键值对 (词语#作者名#文档名, 词频)

```

1: procedure COMBINE(Text key, Iterable<IntWritable> value)
2:   Sum = 0
3:   for all val in value do do
4:     Sum  $\leftarrow$  Sum + val
5:   end for
6:   Emit(key, Sum)
7: end procedure

```

---

**NewPartitioner:**

---

**Class 8** *class NewPartitioner*

---

```

1: procedure GETPARTITION(Text key, IntWritable value, int NumReduceTasks)
2:   word  $\leftarrow$  key.split("#")[0]
3:   author  $\leftarrow$  key.split("#")[1]
4:   term  $\leftarrow$  word + author
5:   return super.getPartition(term, value, NumReduceTasks)
6: end procedure

```

---

**Reducer:****Class 9 class Reducer****Input :** Combiner的结果<Text key, Iterable<IntWritable>>**Output :** 词语 平均出现次数 [小说名: 词频]

```
1: procedure SETUP( )
2:    $t_{prev} \leftarrow \emptyset$ 
3:    $P \leftarrow newPostingsList$ 
4: end procedure
5: procedure REDUCE(tuple< $t, n >, tf[f]$ )
6:   if  $t \neq t_{prev}$  &&  $t_{prev} \neq \emptyset$  then
7:      $author \leftarrow t_{prev}.split()$ 
8:      $word \leftarrow t_{prev}.split()$ 
9:     计算TF, TDF
10:    Emit( $author, < word, TF, IDF >$ )
11:     $P.Reset()$ 
12:     $P.Add(< n, f >)$ 
13:     $t_{prev} \leftarrow t$ 
14: end procedure
15: procedure CLOSE( )
16:    $author \leftarrow t_{prev}.split()$ 
17:    $word \leftarrow t_{prev}.split()$ 
18:   计算TF, TDF
19:   Emit( $author, < word, TF, IDF >$ )
20: end procedure
```

---

## §4. 实验测试与运行结果

### 4.1 JAR包执行方式说明

1. 本地编译源代码并打包

2. 将jar包上传到集群

3. 使用命令hadoop jar InvertedIndex.jar InvertedIndex

```
hdfs://master01:9000/data/wuxia_novels ./Lab2/InvertedIndexoutput  
执行倒排表作业
```

4. 使用命令hadoop jar ResultSort.jar ResultSort

```
./Lab2/InvertedIndexoutput/part-r-00000 ./Lab2/ResultSortoutput  
执行排序作业
```

5. 使用命令hadoop jar TFIDF.jar TFIDF

```
hdfs://master01:9000/data/wuxia_novels ./Lab2/TFIDFoutput  
执行TFIDF计算作业
```

### 4.2 实验运行结果展示

在集群上运行的结果如下：

- 带词频属性的文档倒排算法InvertedIndex任务的运行结果：该任务的输出结果在HDFS上的存放路径为：hdfs://master01:9000/user/2016st21/Lab2/InvertedIndexoutput  
该任务在集群上对全部数据集运行的结果的部分截图如下(输出格式为[词语] TAB 平均出现次数， 小说1:词频； 小说2:词频； 小说3:词频； ...； 小说N:词频)：

```

X - □ 终端 文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
0      1.6875, 卧龙生07飞燕惊龙:1; 卧龙生37天涯情侣:1; 卧龙生42新仙鹤神针:1; 卧
龙生45燕子传奇:1; 李凉07赌棍小狂侠:5; 李凉15江湖一担皮:1; 李凉21六宝江湖行:1; 李
凉23妙贼丁小勾:3; 李凉27奇神杨小邪:1; 李凉38笑笑江湖:4; 梁羽生01白发魔女传:1; 梁
羽生11广陵剑:1; 梁羽生12瀚海雄风:1; 梁羽生25牧野流星:3; 梁羽生34武当一剑:1; 金庸
07鹿鼎记:1;
007    1.0, 李凉12活宝小淘气:1;
01     1.0, 卧龙生45燕子传奇:1; 李凉23妙贼丁小勾:1;
01章   1.0, 古龙60神君别传:1;
02章   1.0, 古龙60神君别传:1;
03章   1.0, 古龙60神君别传:1;
04     1.0, 李凉26奇神扬小邪续集:1;
04章   1.0, 古龙60神君别传:1;
05     1.0, 李凉23妙贼丁小勾:1;
05章   1.0, 古龙60神君别传:1;
06     1.0, 李凉23妙贼丁小勾:1;
06章   1.0, 古龙60神君别传:1;
07章   1.0, 古龙60神君别传:1;
08张   1.0, 卧龙生47一代天骄:1;
08章   1.0, 古龙60神君别传:1;
09章   1.0, 古龙60神君别传:1;
0—     1.5, 卧龙生46摇花放鹰传:1; 李凉34天下第一当:2;
0年     2.5, 梁羽生01白发魔女传:4; 梁羽生33随笔集 :三剑楼随笔:1;
@      1.0, 古龙60神君别传:1;

X - □ 终端 文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
1      3.3114754098360657, 卧龙生03翠袖玉环:1; 卧龙生04地狱门:1; 卧龙生16金笔点
龙记:2; 卧龙生24七绝剑:2; 卧龙生32天鹤谱:1; 卧龙生33天剑绝刀:1; 卧龙生42新仙鹤神
针:3; 卧龙生47一代天骄:1; 卧龙生50玉钗盟:1; 古龙06彩环曲:4; 古龙18大旗英雄传:1;
古龙19大人物:2; 古龙24护花铃:4; 古龙26浣花洗剑录:1; 古龙30剑客行:1; 古龙39陆小凤
02绣花大盗:1; 古龙48飘香剑雨:1; 古龙57七种武器07拳头:1; 古龙64湘妃剑:1; 李凉02霸
枪艳血:2; 李凉06超级邪侠:77; 李凉09红顶记:1; 李凉10滑头傻小子:1; 李凉13江湖急救>
站:2; 李凉14江湖双响炮:2; 李凉15江湖一担皮:8; 李凉16江湖一品郎:1; 李凉20狂侠南宫
鹰:2; 李凉21六宝江湖行:6; 李凉22矛盾大师:4; 李凉24妙贼丁小勾续集:1; 李凉25魔手邪
怪:1; 李凉26奇神扬小邪续集:3; 李凉29神偷小千:1; 李凉30淘气世家:4; 李凉33天齐大帝
:2; 李凉34天下第一当:1; 李凉40新蜀山剑侠传续:1; 梁羽生02冰川天女传:3; 梁羽生03冰
河洗剑录:3; 梁羽生05草莽龙蛇传:5; 梁羽生06大唐游侠传:1; 梁羽生07弹指惊雷:2; 梁羽
生09风雷震九州:1; 梁羽生12瀚海雄风:1; 梁羽生14幻剑灵旗:3; 梁羽生15慧剑心魔:2; 梁
羽生16剑网尘丝:1; 梁羽生18绝塞传烽录:1; 梁羽生19狂侠天娇魔女:8; 梁羽生20联剑风云
录:1; 梁羽生22凤宝钗缘:2; 梁羽生24鸣镝风云录:6; 梁羽生31随笔集 :笔不花:1; 梁羽
生32随笔集 :笔花六照:1; 梁羽生34武当一剑:1; 梁羽生35武林天骄:1; 梁羽生38云海玉弓
缘:1; 金庸04天龙八部:2; 金庸05射雕英雄传:1; 金庸07鹿鼎记:4;
1.     1.0, 梁羽生20联剑风云录:1;
10     2.0, 李凉06超级邪侠:1; 李凉26奇神扬小邪续集:2; 梁羽生31随笔集 :笔不花:4;
梁羽生38云海玉弓缘:1;
1000万钱   1.0, 梁羽生31随笔集 :笔不花:1;
100间   1.0, 梁羽生31随笔集 :笔不花:1;
102个   1.0, 梁羽生31随笔集 :笔不花:1;
107    1.0, 梁羽生31随笔集 :笔不花:1;

```

- 对词语的平均出现次数进行排序的ResultSort任务的运行结果：该任务的输出结果在HDFS上的存放路径为：hdfs://master01:9000/user/2016st21/Lab2/ResultSortoutput  
该任务在集群上对全部数据集运行的结果的部分截图如下（输出格式为平均出现次数 TAB 原纪录）：

```

x -> 终端 文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
1.0 因材 1.0, 梁羽生05草莽龙蛇传:1;
1.0 因时制宜 1.0, 卧龙生12剑气洞彻九重天:1; 卧龙生15绎雪玄霜:1; 卧龙>
生36天香飘:1; 卧龙生49幽灵四艳:1; 卧龙生50玉钗盟:1; 梁羽生32随笔集 : 笔花六照:1;
金庸13碧血剑:1;
1.0 因日 1.0, 古龙29剑毒梅香:1;
1.0 因人制宜 1.0, 卧龙生30素手劫:1;
1.0 因循 1.0, 卧龙生09风雨燕归来:1; 卧龙生22女捕头:1; 卧龙生26情剑无刃:1;
卧龙生30素手劫:1; 梁羽生29塞外奇侠传:1; 金庸07鹿鼎记:1;
1.0 因式 1.0, 梁羽生23龙虎斗京华:1;
1.0 因地制宜 1.0, 卧龙生50玉钗盟:1; 李凉07赌棍小狂侠:1; 梁羽生32随笔>
集 : 笔花六照:1;
1.0 因势利导 1.0, 卧龙生07飞燕惊龙:1; 卧龙生12剑气洞彻九重天:1; 卧龙>
生51玉手点将录:1; 梁羽生38云海玉弓缘:1;
1.0 因利乘便 1.0, 梁羽生14幻剑灵旗:1; 梁羽生22龙凤宝钗缘:1; 梁羽生37>
游剑江湖:1;
1.0 因公殉职 1.0, 李凉15江湖一担皮:1; 李凉28忍者龟:1; 金庸07鹿鼎记:1;
1.0 08张 1.0, 卧龙生47一代天骄:1;
1.0 08章 1.0, 古龙60神君别传:1;
1.0 09章 1.0, 古龙60神君别传:1;
1.0 突升 1.0, 卧龙生19惊鸿一剑震江湖:1;
1.0 突刺 1.0, 梁羽生03冰河洗剑录:1; 金庸09书剑恩仇录:1;
1.0 突击队 1.0, 李凉04本尊分身:1;
"~/桌面/part-r-00000" 134881L, 112096626C 1,1 顶端

```

```

x -> Vim 文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
1.9473684210526316 磨折 1.9473684210526316, 卧龙生07飞燕惊龙:2; 卧龙生08
风尘侠隐:1; 卧龙生33天剑绝刀:1; 卧龙生42新仙鹤神针:1; 卧龙生52袁紫烟:1; 梁羽生03
冰河洗剑录:1; 梁羽生07弹指惊雷:3; 梁羽生10风云雷电:4; 梁羽生11广陵剑:1; 梁羽生12
瀚海雄风:3; 梁羽生13还剑奇情录:3; 梁羽生14幻剑灵旗:2; 梁羽生15慧剑心魔:1; 梁羽生
19狂侠天娇魔女:4; 梁羽生22龙凤宝钗缘:3; 梁羽生25牧野流星:2; 梁羽生30散花女侠:3;
梁羽生36侠骨丹心:1; 梁羽生37游剑江湖:2;
1.9473684210526316 一隔 1.9473684210526316, 卧龙生29双凤旗:1; 卧龙生39铁
苗神剑:2; 卧龙生46摇花放鹰传:1; 古龙32剑玄录:1; 古龙64湘妃剑:1; 李凉27奇神杨小邪
:1; 梁羽生02冰川天女传:3; 梁羽生09风雷震九洲:2; 梁羽生17江湖三女侠:1; 梁羽生19狂
侠天娇魔女:1; 梁羽生20联剑风云录:2; 梁羽生22龙凤宝钗缘:2; 梁羽生27萍踪侠影录:7;
梁羽生28七剑下天山:1; 梁羽生30散花女侠:1; 金庸05射雕英雄传:2; 金庸09书剑恩仇录:4
; 金庸10神雕侠侣:3; 金庸13碧血剑:1;
1.9473684210526316 萋畜 1.9473684210526316, 卧龙生08风尘侠隐:1; 卧龙生14
剑仙列传:5; 卧龙生19惊鸿一剑震江湖:1; 卧龙生21妙绝天香:3; 卧龙生33天剑绝刀:1; 卧
龙生39铁苗神剑:1; 李凉07赌棍小狂侠:1; 李凉12活宝小淘气:5; 李凉19酒狂任小赌:1; 梁
羽生03冰河洗剑录:4; 梁羽生06大唐游侠传:3; 梁羽生09风雷震九洲:1; 梁羽生10风云雷电
:1; 梁羽生12瀚海雄风:2; 梁羽生17江湖三女侠:2; 梁羽生19狂侠天娇魔女:1; 梁羽生26女
帝奇英传:2; 梁羽生38云海玉弓缘:1; 金庸10神雕侠侣:1;
@ @ @ @ @
90003,1 66%

```

- 为每位作家、每个词语计算TF-IDF值的任务的运行结果：该任务的输出结果在HDFS上的存放路径为：`hdfs://master01:9000/user/2016st21/Lab2/TFIDFoutput`。  
该任务在集群上对全部数据集运行的结果的部分截图如下（输出格式为：作家名TAB 词语，TF值，IDF值）：

× ━ □ 终端 文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)	
李凉	跳走, 2, 2.614959778036198
卧龙生	跳起, 37, 0.9932517730102834
古龙	跳起, 61, 0.8472978603872037
李凉	跳起, 136, 0.3813675565291038
梁羽生	跳起, 219, 0.17185025692665923
金庸	跳起, 50, 0.22314355131420976
卧龙生	跳起来, 41, 0.9444616088408515
古龙	跳起来, 439, 0.5849335959197126
李凉	跳起来, 148, 0.31237468504215243
梁羽生	跳起来, 264, 0.1112256351102244
金庸	跳起来, 13, 0.6286086594223741
古龙	跳越, 2, 3.1498829533812494
金庸	跳越, 1, 2.0149030205422647
卧龙生	跳跃, 54, 0.8534898306351247
古龙	跳跃, 56, 1.0296194171811581
李凉	跳跃, 55, 0.578077850775158
梁羽生	跳跃, 176, 0.17185025692665923
金庸	跳跃, 54, 0.31015492830383945
卧龙生	跳跳, 7, 2.043073897508961
古龙	跳跳, 1, 3.5553480614894135
李凉	跳跳, 46, 1.0055218656020977
梁羽生	跳跳, 10, 1.33500106673234
金庸	跳跳, 5, 0.9162907318741551

- “江湖”、“风雪”两个词语的输出结果如下：

江湖 116.06481481481481, 卧龙生01镖旗:275; 卧龙生02春秋笔:329; 卧龙生03翠袖玉环:402; 卧龙生04地狱门:105; 卧龙生05飞花逐月:298; 卧龙生06飞铃:244; 卧龙生07飞燕惊龙:269; 卧龙生08风尘侠隐:228; 卧龙生09风雨燕归来:198; 卧龙生10黑白剑:223; 卧龙生11黑白双娇:117; 卧龙生12剑气洞彻九重天:299; 卧龙生13剑无痕:261; 卧龙生14剑仙列传:25; 卧龙生15绝雪玄霜:274; 卧龙生16金笔点龙记:318; 卧龙生17金凤剪:326; 卧龙生18金剑雕翎:397; 卧龙生19惊鸿一剑震江湖:346; 卧龙生20梦幻之刀:106; 卧龙生21妙绝天香:28; 卧龙生22女捕头:317; 卧龙生23飘花令:263; 卧龙生24七绝剑:130; 卧龙生25七绝剑II还情剑:144; 卧龙生26情剑无刃:7; 卧龙生27琼楼十二曲:181; 卧龙生28神州豪侠:261; 卧龙>生29双凤旗:261; 卧龙生30素手劫:374; 卧龙生31桃花血令:131; 卧龙生32天鹤谱:106; 卧龙生33天剑绝刀:321; 卧龙生34天龙甲:293; 卧龙生35>天马霜衣:278; 卧龙生36天香飘:246; 卧龙生37天涯情侣:113; 卧龙生38铁剑玉佩:71; 卧龙生39铁苗神剑:358; 卧龙生40无名箫:136; 卧龙生41无形剑:256; 卧龙生42新仙鹤神针:188; 卧龙生43血剑丹心:350; 卧龙生44烟锁江湖:238; 卧龙生45燕子传奇:132; 卧龙生46插花放鹰传:446; 卧龙>生47一代天骄:353; 卧龙生48银月飞霜:132; 卧龙生49幽灵四艳:221; 卧龙生50玉钗盟:320; 卧龙生51玉手点将录:150; 卧龙生52袁紫烟:30; 卧龙生53岳小钗:308; 卧龙生54指剑为媒:57; 古龙01白玉老虎:111; 古龙02白玉雕龙:24; 古龙03碧血洗银枪:81; 古龙04边城刀声:50; 古龙05边城浪>子:62; 古龙06彩环曲:76; 古龙07残金缺玉:82; 古龙08苍穹神剑:131; 古龙09楚留香01血海飘香:47; 古龙10楚留香02大沙漠:23; 古龙11楚留香03画眉鸟:96; 古龙12楚留香04僵尸还魂(鬼恋传奇):41; 古龙13楚留香05蝙蝠传奇:80; 古龙14楚留香06桃花传奇:8; 古龙15楚留香07新月传奇:20; >古龙16楚留香08午夜兰花:68; 古龙17大地飞鹰:56; 古龙18大旗英雄传:151; 古龙19大人物:37; 古龙20多情剑客无情剑:150; 古龙21飞刀, 又见飞刀:30; 古龙22风铃中的刀声:76; 古龙23孤星传:175; 古龙24护花铃:193; 古龙25欢乐英雄:64; 古龙26浣花洗剑录:285; 古龙27血鹏鹉:11; 古龙28剑·花·烟雨·江南:18; 古龙29剑毒梅香:110; 古龙30剑客行:187; 古龙31剑气书香:27; 古龙32剑亥录:160; 古龙33九月鹰飞:50; 古龙34绝不低头:3; 古龙35绝代双骄:249; 古龙36猎鹰·赌局:107; 古龙37流星·蝴蝶·剑:34; 古龙38陆小凤01陆小凤前传(金鹏王朝):15; 古龙39陆小凤02绣花大盗:23; 古龙40陆小凤03决战前后:26; 古龙41陆小凤04银钩赌坊:10; 古龙42陆小凤05幽灵山庄:40; 古龙43陆小凤06凤舞九天:42; 古龙44陆小凤07剑神一笑:47; 古龙45名剑风流:201; 古龙46那一剑的风情:32; 古龙47怒剑狂花:102; 古龙48飘香剑雨:67; 古龙49七杀手:20; 古龙50七星龙王:55; 古龙51七种武器01长生剑:23; 古龙52七种武器02碧玉刀:22; 古龙53七种武器03孔雀翎:10; 古龙54七种武器04多情环:8; 古龙55七种武器05霸王枪:54; 古龙56七种武器06离别钩:39; 古龙57七种武器07拳头:16; 古龙58情人箭:197; 古龙59三少爷的剑:86; 古龙60神君别传:12; 古龙61失魂引:96; 古龙62天涯·明月·刀:69; 古龙63武林外史:179; 古龙64湘妃剑:169; 古龙65萧十一郎:56; 古龙66火并萧十一郎:89; 古龙67英雄无泪:54; 古龙68游侠录:103; 古龙69圆月弯刀:155; 古龙70月异星邪:102; 李凉01暗器高手:53; 李凉02霹雳艳血:34; 李凉03百败小赢家:79; 李凉04本尊分身:97; 李凉05超霸的男人:7; 李凉06超级邪侠:209; 李凉07赌棍小狂侠:31; 李凉08公孙小刀:69; 李凉09红顶记:48; 李凉10滑头傻小子:43; 李凉11会醉才会赢:189; 李凉12活宝小淘气:121; 李凉13江湖急救站:118; 李凉14江湖双响炮:291; 李凉15江湖一担皮:197; 李凉16江湖一品郎:186; 李>凉17惊神关小刀:38; 李凉18酒赌小浪子:17; 李凉19酒狂任小猪:167; 李凉20狂侠南宫鹰:15; 李凉21六宝江湖行:72; 李凉22矛盾天师:59; 李凉23妙贼丁小勾:16; 李凉24妙贼丁小勾续集:13; 李凉25魔手邪怪:6; 李凉26奇神扬小邪续集:98; 李凉27奇神杨小邪:102; 李凉28忍者龟:77; 李凉29>神偷小千:17; 李凉30淘气世家:81; 李凉31天才混混:89; 李凉32天才混混外集:27; 李凉33天齐大帝:62; 李凉34天下第一当:52; 李凉35武林暗游>记:180; 李凉36小鬼大赢家:29; 李凉37小鱼吃大鱼:42; 李凉38笑笑江湖:69; 李凉39新蜀山剑侠传:20; 李凉40新蜀山剑侠传续:1; 李凉41杨小邪>发威:34; 梁羽生01白发魔女传:63; 梁羽生02冰川天女传:65; 梁羽生03冰河洗剑录:65; 梁羽生04冰魄寒光剑:4; 梁羽生05草莽龙蛇传:143; 梁羽>生06大唐游侠传:97; 梁羽生07弹指惊雷:77; 梁羽生08飞凤潜龙:4; 梁羽生09风雷震九洲:135; 梁羽生10风云雷电:139; 梁羽生11广陵剑:149; 梁>羽生12瀚海雄风:69; 梁羽生13还剑奇情录:8; 梁羽生14幻剑灵旗:27; 梁羽生15慧剑心魔:102; 梁羽生16剑网尘丝:127; 梁羽生17江湖三女侠:252; 梁羽生18绝塞传烽录:40; 梁羽生19狂侠天娇魔女:180; 梁羽生20联剑风云录:126; 梁羽生21梁羽生传奇:74; 梁羽生22龙凤宝钗缘:109; 梁羽生23 80883, 4-3 59%

“江湖”的输出结果

```

风雪      4. 5333333333333333, 卧龙生01镖旗:3; 卧龙生07飞燕惊龙:16; 卧龙生08风尘侠隐:1; 卧龙生09风雨燕归来:1; 卧龙生12剑气洞彻九
36; 卧龙生15绛雪玄霜:4; 卧龙生18金剑雕翎:9; 卧龙生19惊鸿一剑震江湖:2; 卧龙生22女捕头:4; 卧龙生25七绝剑II还情剑:2; 卧龙生27琼
二曲:1; 卧龙生31桃花血令:2; 卧龙生36天香飘:2; 卧龙生38铁剑玉佩:7; 卧龙生39铁苗神剑:7; 卧龙生42新仙鹤神针:15; 卧龙生47一代天骄
卧龙生49幽灵四艳:1; 卧龙生52袁紫烟:6; 古龙07残金缺玉:7; 古龙13楚留香05蝙蝠传奇:1; 古龙14楚留香06桃花传奇:1; 古龙20多情剑客无
4; 古龙23孤星传:2; 古龙24护花铃:1; 古龙25欢乐英雄:3; 古龙29剑毒梅香:2; 古龙31剑气书香:3; 古龙42陆小凤05幽灵山庄:1; 古龙46那
风情:3; 古龙47怒剑狂花:2; 古龙56七种武器06离别钩:1; 古龙61失魂引:14; 古龙63武林外史:24; 古龙64湘妃剑:5; 古龙67英雄无泪:8; 李
暗器高手:1; 李凉02霸枪艳血:5; 李凉03百败小赢家:1; 李凉04本尊分身:6; 李凉09红顶记:1; 李凉11会醉才会赢:1; 李凉14江湖双响炮:4;
6江湖一品郎:1; 李凉19酒狂任小赌:32; 李凉31天下才混混:4; 李凉34天下第一当:1; 李凉36小鬼大赢家:1; 李凉39新蜀山剑侠传:1; 李凉40新
剑侠传续:12; 梁羽生01白发魔女传:4; 梁羽生02冰川天女传:5; 梁羽生03冰河洗剑录:1; 梁羽生07弹指惊雷:1; 梁羽生09风雷震九州:1; 梁羽
风云雷电:3; 梁羽生12瀚海雄风:1; 梁羽生14幻剑灵旗:1; 梁羽生20联剑风云录:1; 梁羽生21梁羽生传奇:1; 梁羽生22龙凤宝钗缘:5; 梁羽生
野流星:5; 梁羽生26女帝奇英传:3; 梁羽生27萍踪侠影录:3; 梁羽生30散花女侠:1; 梁羽生37游剑江湖:1; 梁羽生38云海玉弓缘:1; 金庸01飞
传:3; 金庸03连城诀:1; 金庸04天龙八部:1; 金庸05射雕英雄传:7; 金庸06白马啸西风:11; 金庸07鹿鼎记:1; 金庸10神雕侠侣:2; 金庸12倚天
记:1;

```

### “风雪”的输出结果

从上面的结果可以看到，不出意料地江湖基本出现在了所有的武侠小说之中，正所谓“人在江湖，身不由己”。

本实验在集群上执行MapReduce Job后获得的执行报告如下：

- 在集群All Application (<http://114.212.190.91:8088/>) 的WebUI页面中查看Job的执行状态的截图如下：

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI
application_1461411805941_0067	2016st21	InvertedIndexTable	MAPREDUCE	default	Tue Apr 26 19:47:45 +0800 2016	Tue Apr 26 19:48:33 +0800 2016	FINISHED	SUCCEEDED		<a href="#">History</a>
application_1461411805941_0064	2016st21	InvertedIndexTable	MAPREDUCE	default	Tue Apr 26 19:31:50 +0800 2016	Tue Apr 26 19:32:40 +0800 2016	FINISHED	SUCCEEDED		<a href="#">History</a>
application_1461411805941_0063	2016st21	InvertedIndexTable	MAPREDUCE	default	Tue Apr 26 19:27:53 +0800 2016	Tue Apr 26 19:28:57 +0800 2016	FINISHED	SUCCEEDED		<a href="#">History</a>
application_1461411805941_0060	2016st21	InvertedIndexTable	MAPREDUCE	default	Tue Apr 26 19:17:36 +0800 2016	Tue Apr 26 19:18:26 +0800 2016	FINISHED	SUCCEEDED		<a href="#">History</a>
application_1461411805941_0058	2016st21	InvertedIndexTable	MAPREDUCE	default	Tue Apr 26 19:12:39 +0800 2016	Tue Apr 26 19:13:17 +0800 2016	FINISHED	SUCCEEDED		<a href="#">History</a>
application_1461411805941_0042	2016st21	InvertedIndexTable	MAPREDUCE	default	Tue Apr 26 15:30:46 +0800 2016	Tue Apr 26 15:31:32 +0800 2016	FINISHED	SUCCEEDED		<a href="#">History</a>
application_1461411805941_0041	2016st21	InvertedIndexTable	MAPREDUCE	default	Tue Apr 26 15:16:18 +0800 2016	Tue Apr 26 15:17:08 +0800 2016	FINISHED	SUCCEEDED		<a href="#">History</a>
application_1461411805941_0025	2016st21	QuasiMonteCarlo	MAPREDUCE	default	Mon Apr 25 23:58:09 +0800 2016	Mon Apr 25 23:59:07 +0800 2016	FINISHED	SUCCEEDED		<a href="#">History</a>
application_1461411805941_0070	2016st21	Result Sort	MAPREDUCE	default	Tue Apr 26 19:51:37 +0800 2016	Tue Apr 26 19:52:01 +0800 2016	FINISHED	SUCCEEDED		<a href="#">History</a>
application_1461411805941_0053	2016st21	InvertedIndexTable	MAPREDUCE	default	Tue Apr 26 18:23:09 +0800 2016	Tue Apr 26 18:23:43 +0800 2016	FINISHED	FAILED		<a href="#">History</a>
application_1461411805941_0179	2016st21	TFIDF	MAPREDUCE	default	Thu Apr 28 21:27:57 +0800 2016	Thu Apr 28 21:28:44 +0800 2016	FINISHED	SUCCEEDED		<a href="#">History</a>
application_1461411805941_0177	2016st21	TFIDF	MAPREDUCE	default	Thu Apr 28 21:14:53 +0800 2016	Thu Apr 28 21:15:48 +0800 2016	FINISHED	SUCCEEDED		<a href="#">History</a>
application_1461411805941_0175	2016st21	TFIDF	MAPREDUCE	default	Thu Apr 28 21:07:11 +0800 2016	Thu Apr 28 21:08:13 +0800 2016	FINISHED	SUCCEEDED		<a href="#">History</a>
application_1461411805941_0174	2016st21	TFIDF	MAPREDUCE	default	Thu Apr 28 21:01:20 +0800 2016	Thu Apr 28 21:02:40 +0800 2016	FINISHED	FAILED		<a href="#">History</a>

- 在WebUI页面 (<http://114.212.190.91:19888/jobhistory>) 找到对应的job如下

### Retired Jobs

Search:											
Submit Time	Start Time	Finish Time	Job ID	Name	User	Queue	State	Maps Total	Maps Completed	Reduces Total	Reduces Completed
2016.04.26 19:17:36 CST	2016.04.26 19:17:41 CST	2016.04.26 19:18:25 CST	<a href="#">job_1461411805941_0060</a>	InvertedIndex Table	2016st21	default	SUCCEEDED	218	218	1	1
2016.04.26 19:27:53 CST	2016.04.26 19:27:57 CST	2016.04.26 19:28:56 CST	<a href="#">job_1461411805941_0063</a>	InvertedIndex Table	2016st21	default	SUCCEEDED	218	218	1	1
2016.04.26 19:31:50 CST	2016.04.26 19:31:55 CST	2016.04.26 19:32:40 CST	<a href="#">job_1461411805941_0064</a>	InvertedIndex Table	2016st21	default	SUCCEEDED	218	218	1	1
2016.04.26 19:47:45 CST	2016.04.26 19:47:49 CST	2016.04.26 19:48:33 CST	<a href="#">job_1461411805941_0067</a>	InvertedIndex Table	2016st21	default	SUCCEEDED	218	218	1	1
2016.04.26 19:51:37 CST	2016.04.26 19:51:41 CST	2016.04.26 19:52:01 CST	<a href="#">job_1461411805941_0070</a>	Result Sort	2016st21	default	SUCCEEDED	1	1	1	1
...	...	...									
2016.04.28 21:27:57 CST	2016.04.28 21:28:02 CST	2016.04.28 21:28:44 CST	<a href="#">job_1461411805941_0179</a>	TFIDF	2016st21	default	SUCCEEDED	218	218	1	1
2016.04.28 21:14:53 CST	2016.04.28 21:14:58 CST	2016.04.28 21:15:48 CST	<a href="#">job_1461411805941_0177</a>	TFIDF	2016st21	default	SUCCEEDED	218	218	1	1
2016.04.28 21:07:17 CST	2016.04.28 21:07:21 CST	2016.04.28 21:08:12 CST	<a href="#">job_1461411805941_0175</a>	TFIDF	2016st21	default	SUCCEEDED	218	218	1	1

- 根据Job ID链接进入Job详细页面，几个job的详细信息如下所示。

- InvertedIndexTable:

Logged in as: dr.wmo

**loop** **Counters for job\_1461411805941\_0067**

Counter Group	Name	Map	Reduce	Total
File System Counters	FILE: Number of bytes read	0	148,774,946	148,774,946
	FILE: Number of bytes written	173,963,852	148,890,429	322,854,281
	FILE: Number of large read operations	0	0	0
	FILE: Number of read operations	0	0	0
	FILE: Number of write operations	0	0	0
	HDFS: Number of bytes read	268,312,252	0	268,312,252
	HDFS: Number of bytes written	0	110,648,830	110,648,830
	HDFS: Number of large read operations	0	0	0
	HDFS: Number of read operations	654	3	657
	HDFS: Number of write operations	0	2	2
Job Counters	Data-local map tasks	0	0	215
	Launched map tasks	0	0	218
	Launched reduce tasks	0	0	1
	Rack-local map tasks	0	0	3
	Total megabyte-seconds taken by all map tasks	0	0	1,653,874,688
	Total megabyte-seconds taken by all reduce tasks	0	0	31,096,832
	Total time spent by all map tasks (ms)	0	0	1,615,112
	Total time spent by all maps in occupied slots (ms)	0	0	1,615,112
	Total time spent by all reduce tasks (ms)	0	0	30,368
	Total time spent by all reduces in occupied slots (ms)	0	0	30,368
	Total vcore-seconds taken by all map tasks	0	0	1,615,112
	Total vcore-seconds taken by all reduce tasks	0	0	30,368
Map-Reduce Framework	Name	Map	Reduce	Total
	Combine input records	45,567,096	0	45,567,096
	Combine output records	3,977,262	0	3,977,262
	CPU time spent (ms)	991,960	32,260	1,024,220
	Failed Shuffles	0	0	0
	GC time elapsed (ms)	28,291	925	29,216
	Input split bytes	32,349	0	32,349
	Map input records	1,954,746	0	1,954,746
	Map output bytes	1,529,780,369	0	1,529,780,369
	Map output materialized bytes	148,776,242	0	148,776,242
	Map output records	45,567,096	0	45,567,096
	Merged Map outputs	0	218	218
	Physical memory (bytes) snapshot	58,413,002,752	291,491,840	58,704,494,592
	Reduce input groups	0	3,977,262	3,977,262
	Reduce input records	0	3,977,262	3,977,262
	Reduce output records	0	134,881	134,881
	Reduce shuffle bytes	0	148,776,242	148,776,242
	Shuffled Maps	0	218	218
	Spilled Records	3,977,262	3,977,262	7,954,524
	Total committed heap usage (bytes)	43,146,805,248	209,190,912	43,355,996,160
	Virtual memory (bytes) snapshot	357,850,693,632	1,654,018,048	359,504,711,680
Shuffle Errors	Name	Map	Reduce	Total
	BAD_ID	0	0	0
	CONNECTION	0	0	0
	IO_ERROR	0	0	0
	WRONG_LENGTH	0	0	0
	WRONG_MAP	0	0	0
File Input Format Counters	Name	Map	Reduce	Total
	Bytes Read	268,279,903	0	268,279,903
File Output Format Counters	Name	Map	Reduce	Total
	Bytes Written	0	110,648,830	110,648,830

– ResultSort

**loop**

### Counters for job\_1461411805941\_0070

Counter Group	Name	Counters			Total
		Map	Reduce	Total	
File System Counters	FILE: Number of bytes read	112,297,666	112,297,660	224,595,326	
	FILE: Number of bytes written	224,710,423	112,412,702	337,123,125	
	FILE: Number of large read operations	0	0	0	
	FILE: Number of read operations	0	0	0	
	FILE: Number of write operations	0	0	0	
	HDFS: Number of bytes read	110,648,967	0	110,648,967	
	HDFS: Number of bytes written	0	112,096,626	112,096,626	
	HDFS: Number of large read operations	0	0	0	
	HDFS: Number of read operations	3	3	6	
Job Counters	HDFS: Number of write operations	0	2	2	
	Name	Map	Reduce	Total	
	Launched map tasks	0	0	1	
	Launched reduce tasks	0	0	1	
	Rack-local map tasks	0	0	1	
	Total megabyte-seconds taken by all map tasks	0	0	8,190,976	
	Total megabyte-seconds taken by all reduce tasks	0	0	6,990,848	
	Total time spent by all map tasks (ms)	0	0	7,999	
	Total time spent by all maps in occupied slots (ms)	0	0	7,999	
	Total time spent by all reduce tasks (ms)	0	0	6,827	
	Total time spent by all reduces in occupied slots (ms)	0	0	6,827	
	Total vcore-seconds taken by all map tasks	0	0	7,999	
	Total vcore-seconds taken by all reduce tasks	0	0	6,827	
Map-Reduce Framework	Name	Map	Reduce	Total	
	Combine input records	0	0	0	
	Combine output records	0	0	0	
	CPU time spent (ms)	7,710	5,860	13,570	
	Failed Shuffles	0	0	0	
	GC time elapsed (ms)	129	85	214	
	Input split bytes	137	0	137	
	Map input records	134,881	0	134,881	
	Map output bytes	111,877,679	0	111,877,679	
	Map output materialized bytes	112,297,660	0	112,297,660	
	Map output records	134,881	0	134,881	
	Merged Map outputs	0	1	1	
	Physical memory (bytes) snapshot	267,386,880	169,914,368	437,301,248	
	Reduce input groups	0	19,569	19,569	
	Reduce input records	0	134,881	134,881	
	Reduce output records	0	134,881	134,881	
	Reduce shuffle bytes	0	112,297,660	112,297,660	
	Shuffled Maps	0	1	1	
	Spilled Records	269,762	134,881	404,643	
	Total committed heap usage (bytes)	199,229,440	190,316,544	389,545,984	
	Virtual memory (bytes) snapshot	1,645,809,664	1,652,441,088	3,298,250,752	
Shuffle Errors	Name	Map	Reduce	Total	
	BAD_ID	0	0	0	
	CONNECTION	0	0	0	
	IO_ERROR	0	0	0	
	WRONG_LENGTH	0	0	0	
	WRONG_MAP	0	0	0	
File Input Format Counters	Name	Map	Reduce	Total	
	Bytes Read	110,648,830	0	110,648,830	
File Output Format Counters	Name	Map	Reduce	Total	
	Bytes Written	0	112,096,626	112,096,626	

- TFIDF

Logged in as: dr.who

The screenshot shows a table titled "Counters for job\_1461411805941\_0179". The table has columns for Counter Group, Name, Map, Reduce, and Total. The data is categorized into several groups:

- File System Counters:**
  - FILE: Number of bytes read: 142,095,547
  - FILE: Number of bytes written: 309,486,504
  - FILE: Number of large read operations: 0
  - FILE: Number of read operations: 0
  - FILE: Number of write operations: 0
  - HDFS: Number of bytes read: 268,312,252
  - HDFS: Number of bytes written: 16,968,160
  - HDFS: Number of large read operations: 0
  - HDFS: Number of read operations: 654
  - HDFS: Number of write operations: 2
- Job Counters:**
  - Data-local map tasks: 0
  - Launched map tasks: 0
  - Launched reduce tasks: 0
  - Rack-local map tasks: 0
  - Total megabyte-seconds taken by all map tasks: 1,671,572,480
  - Total megabyte-seconds taken by all reduce tasks: 29,027,328
  - Total time spent by all map tasks (ms): 1,632,395
  - Total time spent by all map tasks in occupied slots (ms): 1,632,395
  - Total time spent by all reduce tasks (ms): 28,347
  - Total time spent by all reduces in occupied slots (ms): 28,347
  - Total vcore-seconds taken by all map tasks: 1,632,395
  - Total vcore-seconds taken by all reduce tasks: 28,347
- Map-Reduce Framework:**
  - Combine input records: 45,567,096
  - Combine output records: 3,977,262
  - CPU time spent (ms): 1,056,930
  - Failed Shuffles: 0
  - GC time elapsed (ms): 28,641
  - Input split bytes: 32,349
  - Map input records: 1,954,746
  - Map output bytes: 1,464,482,550
  - Map output materialized bytes: 142,096,843
  - Map output records: 45,567,096
  - Merged Map outputs: 0
  - Physical memory (bytes) snapshot: 58,423,013,376
  - Reduce input groups: 0
  - Reduce input records: 0
  - Reduce output records: 0
  - Reduce shuffle bytes: 0
  - Shuffled Maps: 0
  - Spilled Records: 3,977,262
  - Total committed heap usage (bytes): 43,093,852,160
  - Virtual memory (bytes) snapshot: 356,644,625,920
- 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: 268,279,903
- File Output Format Counters:**
  - Bytes Written: 16,968,160

### 4.3 一些有趣的发现

- 年代，地名，人名等词语因为小说的不同具有独占性，对于搜索和定位来说是较优的关键字。
- 出现了一些英文字母和单词，按常理来说中国的武侠小说是不应该含西文的。不过考虑到这些小说是网上下载的资源，因为大部分应该都是下载的链接信息或者出处信息。
- 统计结果中出现的四字成语、古文俗语较多，频率大，这个不难看出小说行文的文采与流畅之处，毕竟都是大师作品，不是一些烂俗的网络小说。
- 的、地、得等助词出现的频率最高，符合汉语的表达习惯。

- 在统计结果中我们发现“道”这个词的平均出现次数很高，看来只有得道之人才能修得上乘武功。
- 根据TF-IDF理论，我们从结果中检索了几个小说主人公的名字，比如韦小宝，虚竹，发现他们的TF×IDF的值和其他常用词语比起来惊人的高，验证了TF-IDF方法在文本挖掘中的作用[2]。

## §5. 实验总结

### 5.1 实验内容总结

本次实验是大数据处理综合实验的第二次实验，在这次实验中我们完成了三个MapReduce任务的编写：(1)带词频属性的文档倒排算法。(2)根据每个词语的平均出现次数进行全局排序。(3)为每位作家、计算每个词语的TF-IDF。通过本次实验，加深了对MapReduce算法设计与编程的了解，同时进一步巩固了对Hadoop MapReduce基本构架的理解。

本次实验各个任务中最主要的问题是确定不同阶段的键值对。为确定Map与Reduce的设计与各个过程中键值对的类型转换过程，我们根据教材中内容举一反三完成了选做内容。此外，在具体实现时，我们在java零基础的情况下学习了字符串相关的操作方法，灵活切割输入文本与文件名获得有关信息，例如作者名与小说名。

本次实验是我们第一次较为深入地了解Hadoop MapReduce的算法设计与程序的编写和运行，整体完成较为顺利，为后面的实验打下了较好的基础。

### 5.2 团队合作总结

在本次实验中，我们小组采取的是“代码实现+实验报告”的分工模式，即两位同学合作实现代码并完成测试，另外两位同学合作完成实验报告。但是在合作过程中，我们发现这并不是最佳最高效的合作模式。主要表现在完成实验报告的同学仅仅通过与代码实现的同学交流并不能准确理解设计思路，最终还是要依赖阅读代码。而负责代码实现的同学最终还要修改实验报告使得报告中的表达更加准确。

为了进一步提高效率，我们打算在下一次实验中尝试另外一种分工模式：“包产到户”。即在动手实验之前，通过集体交流确定实验内容和内容之间的依赖关系，然后将实验内容划分成不同部分，分别交由几个人完成。每位同学在完成自己的代码部分的同时完成对应的实验报告部分。这一分工模式强烈依赖先期交流的高效性和彻底性以及讨论结果的一致性，但是能让每位同学的参与度大大提高。同时也能让每位同学掌握更多技能，成为多面手。

## 参 考 文 献

- [1] 黄宣华. 深入理解大数据 大数据处理与编程实践[M]. 北京: 机械工业出版社, 2014.7.
- [2] <https://en.wikipedia.org/wiki/Tf>