|  |
| --- |
| MapReduce大数据课程设计之金庸的江湖 实验报告 |

|  |
| --- |
| 第四小组，李书城151220053 李一鸣151220056 刘鎏 151220063 刘洋 151220064 |

[MapReduce大数据课程设计之金庸的江湖 实验报告 2](#_Toc518815201)

[**1** 任务描述 2](#_Toc518815202)

[**2** 设计思路与实现 2](#_Toc518815203)

[**2.1** 设计思路 2](#_Toc518815204)

[**2.2** 数据预处理 3](#_Toc518815205)

[2.2.1 分词工具介绍 3](#_Toc518815206)

[2.2.2 同现算法介绍 3](#_Toc518815207)

[2.2.3 实现细节 3](#_Toc518815208)

[**2.3** PageRank 4](#_Toc518815209)

[2.3.1 算法介绍 4](#_Toc518815210)

[2.3.2 实现细节 4](#_Toc518815211)

[**2.4** 标签传播算法 5](#_Toc518815212)

[2.4.1 算法简介 5](#_Toc518815213)

[2.4.2 实现细节 5](#_Toc518815214)

[**2.5** 可视化 5](#_Toc518815215)

[2.5.1 可视化工具Gephi介绍 5](#_Toc518815216)

[**3** 实验结果及其分析 6](#_Toc518815217)

[**3.1** 数据预处理结果 6](#_Toc518815218)

[3.1.1 结果分析 7](#_Toc518815219)

[**3.2** PageRank结果 7](#_Toc518815220)

[3.2.1 结果分析 8](#_Toc518815221)

[**3.3** 标签传播算法结果 8](#_Toc518815222)

[3.3.1 结果分析 10](#_Toc518815223)

[**3.4** 可视化结果 10](#_Toc518815224)

[3.4.1 结果分析 11](#_Toc518815225)

[**4** 性能分析 12](#_Toc518815226)

[**5** 运行截图 12](#_Toc518815227)

[**5.1** 分词、邻接统计、同现概率统计 12](#_Toc518815228)

[**5.2** PageRank 12](#_Toc518815229)

[5.2.1 预处理 13](#_Toc518815230)

[5.2.2 PageRank迭代计算 13](#_Toc518815231)

[5.2.3 排序输出结果 13](#_Toc518815232)

[**5.3** 标签传播算法 14](#_Toc518815233)

[**5.4** 提交次数 14](#_Toc518815234)

[**6** 小组分工 14](#_Toc518815235)

[**7** 目录说明以及测试方法 14](#_Toc518815236)

[**7.1** 运行环境 14](#_Toc518815237)

[**7.2** 运行方法 14](#_Toc518815238)

[**7.3** 源代码编译方法 15](#_Toc518815239)

[**7.4** 文档结构 15](#_Toc518815240)

[附中文参考文献: 16](#_Toc518815241)

MapReduce大数据课程设计之金庸的江湖 实验报告

第4组，李书城151220053 李一鸣151220056 刘 鎏 151220063 刘 洋 151220064

(南京大学 计算机科学与技术系, 南京 210046)

# 任务描述

通过金庸武侠小说中的人物关系挖掘，来学习和掌握MapReduce程序设计，熟悉和掌握以下MapReduce编程技能：

1. 在Hadoop中使用第三方的Jar包来辅助分析；

2. 掌握简单的MapReduce算法设计：

a) 单词同现算法；

b) 数据整理与归一化算法；

c) 数据排序；

3. 掌握带有迭代特性的MapReduce算法设计：

a) PageRank算法；

b) 标签传播（Label Propagation）算法。

具体的任务要求分为以下六点：

1. 数据预处理，从原始的金庸小说文本中，抽取出与人物互动相关的数据，而屏蔽掉与人物关系无关的文本内容，为后面的基于人物共现的分析做准备。
2. 人物同现统计，完成基于单词同现算法的人物同现统计。
3. 人物关系图构建与特征归一化，根据共现关系构建人物关系邻接表，并计算各邻接点的共现概率。
4. 基于人物关系图的PageRank计算，通过计算PageRank，我们就可以定量地金庸武侠江湖中的“主角”们是哪些。
5. 在人物关系图上的标签传播，为图上的顶点打标签，进行图顶点的聚类分析，从而在一张类似社交网络图中完成社区发现。
6. 分析结果整理，对PageRank的结果进行排序。

# 设计思路与实现

## 设计思路

根据任务要求，任务分工为4个大部分：

1. 对小说的处理：数据预处理、同现统计、构建人物关系图、计算共现概率

这部分主要任务是对段落进行分词，对分词后的结果进行单词同现算法，找出人物共现关系对并统计频次后，计算概率进行归一化处理，从而得到归一化的人物共现关系图的邻接表。

1. PageRank算法

这部分主要任务是基于人物共现关系图运行PageRank算法，并对最后的结果排序输出。

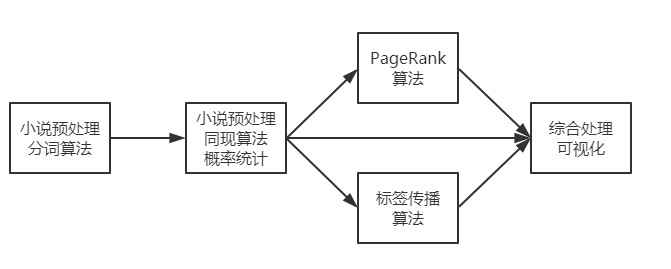
1. 标签传播算法

这部分主要任务是依据定义好的初始标签，使用标签传播算法对归一化的人物共现关系图进行处理，得到各个人物相应的标签，对得到的数据做可视化处理，使其特征能直观地显现出来。

1. 分析结果整理

综合共现关系图邻接表、PageRank输出以及标签传播算法的结果，生成边和点集合，使用Gephi工具进行可视化操作。

程序的运行流程如下图所示：



## 数据预处理

### 分词工具介绍

Ansj\_seg支持对中文文本进行分词，并且可以添加用户自定义的词典，这样它可以准确识别金庸武侠小说中的人名。实验中使用了用户自定义词典，以及用户自定义词典优先的分词。

### 同现算法介绍

对于任意一个词A，和不同于A的任意一个词B，如果A和B出现在了同一行，则认为A和B有同现关系，记为<A,B>，每一行仅记录一次同现关系。对于一整篇小说而言，出现的同现关系次数越多，则认为A和B有关系的概率越大，归一化同现频率就是B和A有关系的频次占与A有关系的所有人的频次之和的比例。

### 实现细节

**ReadNovelMapper类**

在此类的setup方法中，按行读取用cacheFile的方式共享的文件（people\_name\_list.txt），然后将其导入Ansj\_seg工具的自定义的字典中，并归类为WuxiaNames。

在此类的map方法中，使用DicAnalysis方法，即用户词典优先的方法进行分词，保存进List中，然后对于List中的所有人名，使用HashMap过滤重复人名。然后进行人物同现统计，使用迭代器遍历HashMap，用双重循环进行一一比对，如果两者名字不相同，则作为Map的输出。

**ReadNovelReducer类**

在此类的reduce方法中，首先对匹配对进行整合。使用HashMap结构保存所有到该节点的Map的输出，如果HashMap中没有出现过这个有关系的人，那么将这个人加入HashMap中，value置为1，否则value加1。

然后计算总权重，用一个Set遍历HashMap，将上一步的所有value相加，就得到了总权重。

然后计算概率值，用一个Set遍历HashMap，对于每一个value，和总权重相除就是归一化同现概率，然后按照格式进行输出，就得到了该人物的全部邻接表和其同现概率输出。

## PageRank

### 算法介绍

PageRank是一种在搜索引擎中根据网页之间相互的链接关系计算网页排名的技术。PageRank是Google用来标识网页的等级或重要性的一种方法。其级别从1到10级，PR值越高说明该网页越受欢迎（越重要）。

因为在人物关系图中，人物的关系是双向的所以不会有单向边，因此不会存在因为没有出度导致的Page link问题和因为没有入度导致的Page sink问题。所以使用PageRank的简化模型即可。

任意一个人物Pi的PR值可表示为上述表达式，Bi表示链接到Pi的人物集合，Lj表示人物Pj的对外链接。在这次实验中改为

### 实现细节

1. CreateGraph类

这个类是为了创建邻接图和赋予初始PR值1.0。

CreateGraphMapper逐行分析原始数据，以每个人的人名作为key值，PageRank的初始值和其共现关系人物表一起作为value，输出<name,(PR\_init,links)>键值对。格式如下

Name**[\t]**PR**[\t]**name1:w1|name2:w2|name3:w3|…….

2. PageRankIter类

这个类主要是用来迭代计算PageRank值，直到满足运算的结束条件：迭代次数等于10。

PRIterMapper对上阶段的<name,(PR\_init,links)>产生两种<key, value>对：

① For each u in links, 输出 <u,name cur\_rank\*w|>

其中u代表当前name所链接到人物u，并作为key；

name表示当前人物，cur\_rank为当前name的PageRank值， w为u在当前name中的权重， ,

name cur\_rank/|link\_list|作为value。

② 同时为了完成迭代过程，需要传递每个网页的链接信息

<name, |links>

在迭代过程中，必须保留网页的局部链出信息，以维护图的结构，添加 "," 供区分

PRIterReducer从Mapper处得到的原始链接信息继续传递下去，而得到的多个<u,name cur\_rank\*radio>，把所有的cur\_rank\*radio值相加即可得到现在的PR值,从而输出键值对<u,(pr,links)>。

3.PageRankViewer类

这个类首先先将上一步获得的<u,(pr,links)>提取生成<pr,u>键值对以供排序，Hadoop默认对DoubleWritable按升序排序，所以实现一个DoubleWritableDecreasingComparator类，并指定使用这个自定义的Comparator类，对输出结果中的Key即PageRank值进行排序，DoubleWritableDecreasingComparator：这里重载了compare(WritableComparable a, WritableComparable b),和compare(byte[] b1,int s1,int l1,byte[] b2,int s2,int l2)。

## 标签传播算法

### 算法简介

标签传播算法的基本思路是是通过已知节点的标签信息来预标记与其相邻或连接相对紧密的节点，每个待标记的节点都是根据其周围节点的标签选择与自己相关性最大的来决定自己的结果，最基础的标签传播算法是根据周围邻居的某种标签的数目多少来决定，总是选择数量最多的标签；而如果不同邻居的关系紧密程度不一致的话（通常体现在点的权重或边的权重），就形成了带权重的标签传播算法，与基础算法类似，这种算法统计其周围某种标签所占的权重和，选择权重最大的标签进行标记。

在本次实验中由于第三步的输出结果就是不同人物之间的紧密程度，所以可以直接利用这一信息设计带边权重的标签传播算法初始化的信息可以根据十四部作品中的主角来决定（详见3.3图1），对于其他次要角色都初始化为0，他们的标签就是通过由每部作品主角所引伸出的关系网决定。考虑到大部分角色都是只出现在一部作品中，即使有部分出现在多部作品中的角色他们在不同作品中的关系网权重也是差别极大的，所以一旦认为我们的迭代结果覆盖了所有结点（这实际上只需要2-3次迭代就能达到），然后再进行一次迭代精确结果即可。最后的结果就是每个角色都有了一个标签，可以根据这一标签进行可视化实现。

### 实现细节

因为具体实现的代码量较为简洁，所以这次可以直接写入一个LPA.java文件中。

main方法是用于实现标签传播算法的迭代过程，可以根据输入的信息决定迭代次数。

LPAMapper类是用于mapreduce的实现，其中setup方法是用来将原来初始化的标签信息读入内存中，map方法则是实际的迭代计算过程，其中参数my\_key表示文本行号，my\_value表示一行文本的信息。对每个节点都用一个数组储存周围所有节点的权重，分别求和后选择权重最大的非0标签设为自己新的标签； cleanup是用于每次迭代计算后将新的结果储存到原来的文件中。

另外一个文件RawTagCreator.java是用来初始化RawTag.txt文件信息，根据每个主角的信息分别给予他们1-14的标签，其他角色则定为0，运行时只需要一次初始化计算即可。

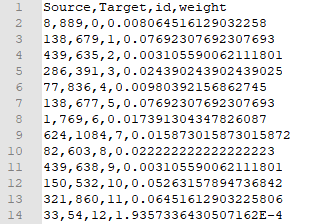
## 可视化

### 可视化工具Gephi介绍

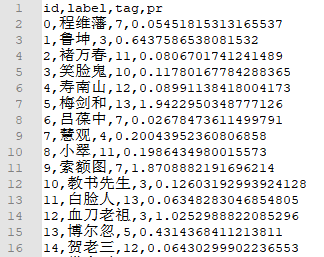
Gephi is an open-source network analysis and visualization software package written in Java on the NetBeans platform. Gephi是一款开源免费跨平台基于JVM的复杂网络分析软件,，其主要用于各种网络和复杂系统，动态和分层图的交互可视化与探测开源工具。

该软件支持使用csv数据格式导入节点列表和边列表；同时使用节点属性进行渲染，数据格式如下：

边列表：



点列表：



# 实验结果及其分析

## 数据预处理结果

下图为数据预处理、同现统计、构建人物关系图、计算共现概率后得到的人物关系邻接表以及同现概率统计的部分结果截图：

注1，文件位置：**ReadNovelOutput/part-r-00000**

注2，每行的格式：人名**\t**关系人名**:**共现概率**|**关系人名:共现概率**|…|**关系人名**:**共现概率



### 结果分析

对于每行，每行包含了一个人物所有的邻接信息。

对于每行的人物，他的各条邻接关系的权重总和为1（浮点数计算会损失少量精度）,权重越高的说明同现次数越多，关系越紧密。

## PageRank结果

部分结果如下图所示：

注1：每次迭代都会得到新的结果，以下展示的是第一次迭代后、第十次迭代后及最终统计的结果

注2：每一行的格式：PageRank值**\t**人名

注3：图1、图2的位置：PRData[1|10]/part-r-00000；图3的位置：FinalRank/ part-r-00000

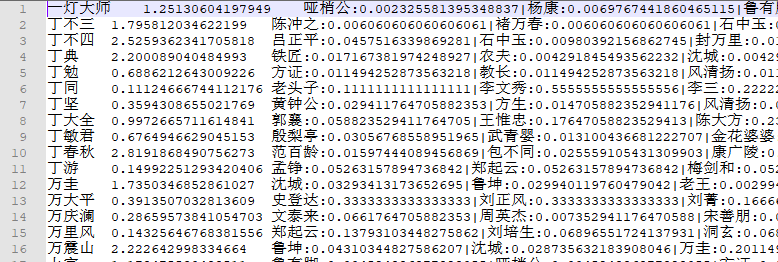


图1. 迭代1次后的输出结果

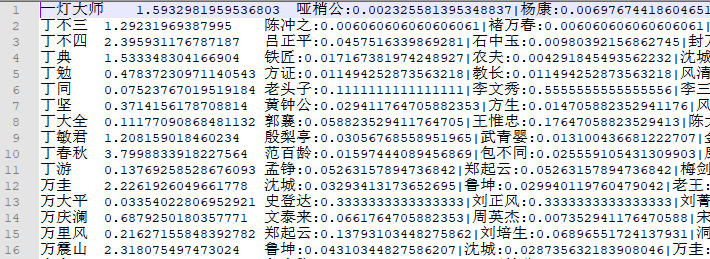


图2. 迭代10次后的输出结果

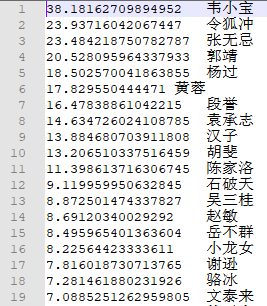


图3. 迭代完成后的输出结果

### 结果分析

上述图3的结果是进行过排序的PageRank值，越高的PageRank值意味着和其他人的交互越多，且和他有关系的人是主角的概率也越大，因此是主角的概率也越大。

韦小宝、令狐冲、张无忌、郭靖、杨过等人都有非常高的PageRank值，这个结果和现实的他们都是金庸小说中的主角的结果一致，符合客观事实。

## 标签传播算法结果

部分结果如下图所示：

注1：图1表示各小说的主角表，每本小说提供了三个主角

注2：图2表示最开始生成的标签，其中极少数主角所含的标签是根据图0加的

注3：图3表示迭代三次后的标签结果。

注4：文件位置：图1：主角列表.txt；图2：RawTagPre.txt；图3：RawTag.txt

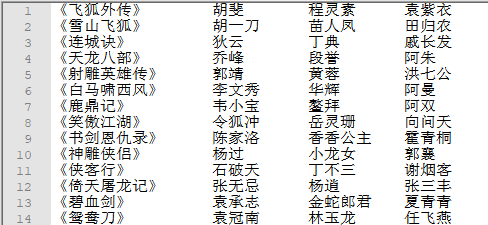


图1. 主角列表

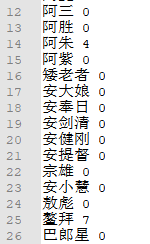


图2. 迭代开始前的标签表



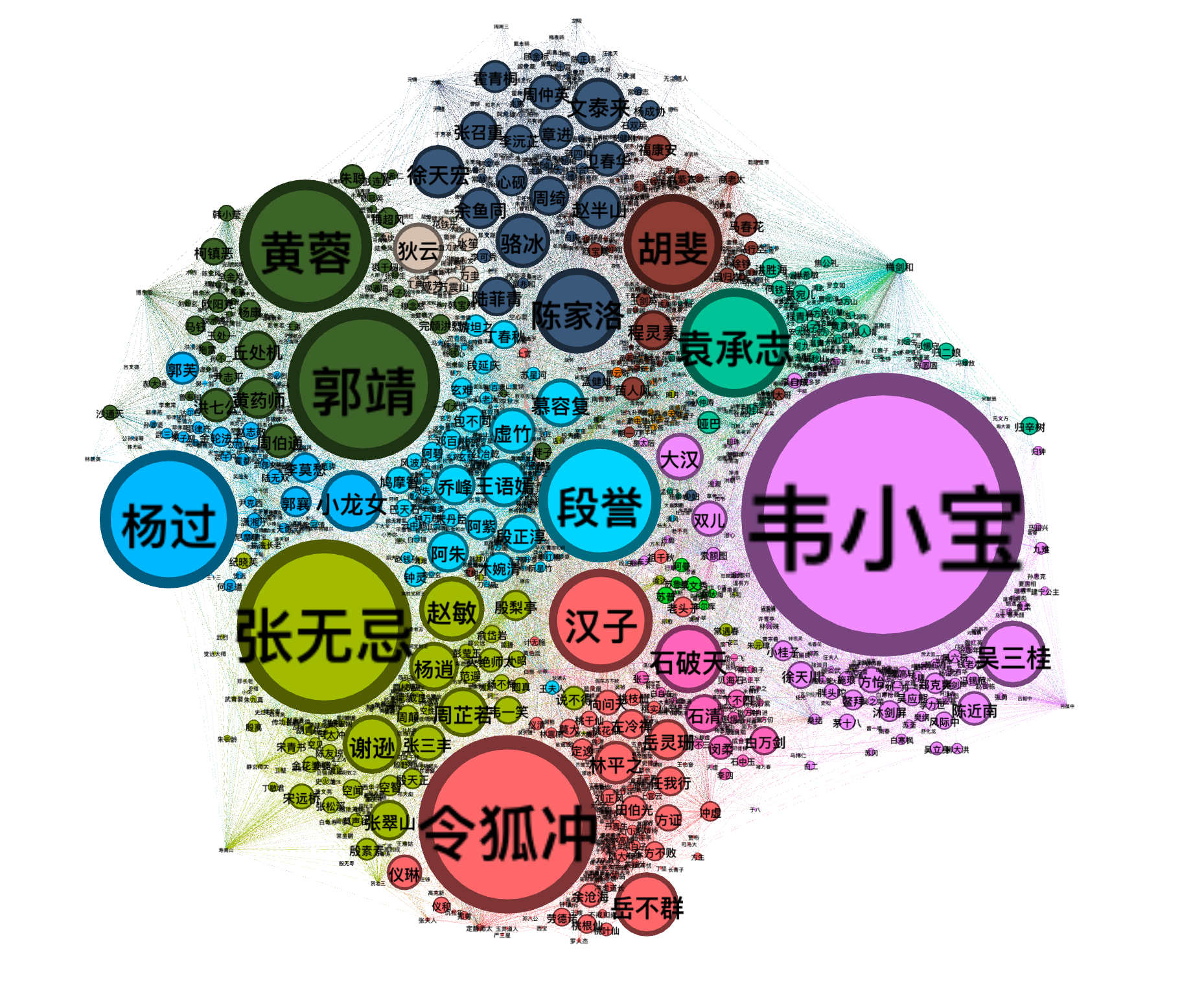
图3. 迭代3次后的标签表

### 结果分析

标签传播算法，第一遍结果有很多“0”，第二遍有所减少，第三轮完全没有“0”，所以只需三轮迭代，结果也非常合理。

## 可视化结果

全局图：



局部图：



### 结果分析

通过观察可以清楚地看到，原图中主要有11个不同颜色的聚集体，其他三个tag由于节点数量太少，PageRank值较小，较难发现；

而韦小宝、张无忌、令狐冲、郭靖、黄蓉、杨过以及段誉这些主角节点明显突出，其pr值也较大；

在这些主角节点周围环绕着一些小说中出现的其他人物节点，这些节点较小。可参考上面的张无忌局部图；

由于金庸的武侠小说是成体系的，一些人物可能在多本小说中出现，并联系不同小说中的人物，这些人物的颜色一般取决于在小说中的出现次数，比如人物“郭靖”即在《神雕侠侣》中出现，也在《射雕英雄传》中出现，只是由于他在《射雕英雄传》中出现次数更多，因此颜色和《射雕英雄传》tag相同.

整体来看，比较独立的几个“极”为：天龙八部独占中央；鹿鼎记位于右下角；笑傲江湖和侠客行位于下方；而被称为射雕三部曲的射雕英雄传、神雕侠侣、倚天屠龙记为左侧且联系紧密；上方有连城诀、书剑恩仇录、雪山飞狐、飞狐外传和碧血剑。

# 性能分析

所有任务在MapReduce上时间开销为272s(仅含Map和Reduce的计算时间)。

其中：分词、邻接同现统计耗时17s，PageRank预处理20s，PageRank每次迭代平均18s，PageRank最终综合整理14s，标签传播每次迭代平均13s。

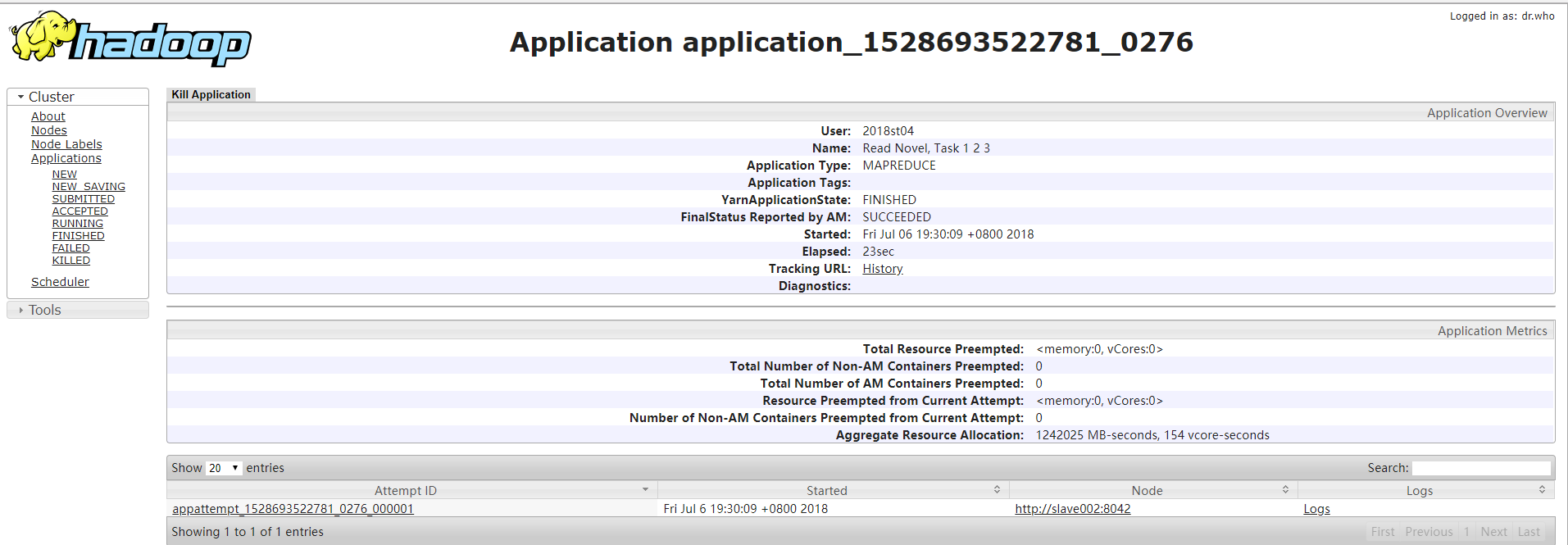
从图中可以看出整个程序运行过程中大量的时间消耗在PageRank 的迭代过程，原因是有10次的迭代过程。

猜想，如果要对程序的性能进行优化，那么可以考虑完善PageRank的迭代过程和PageRank算法，减少输出。

# 运行截图

## 分词、邻接统计、同现概率统计

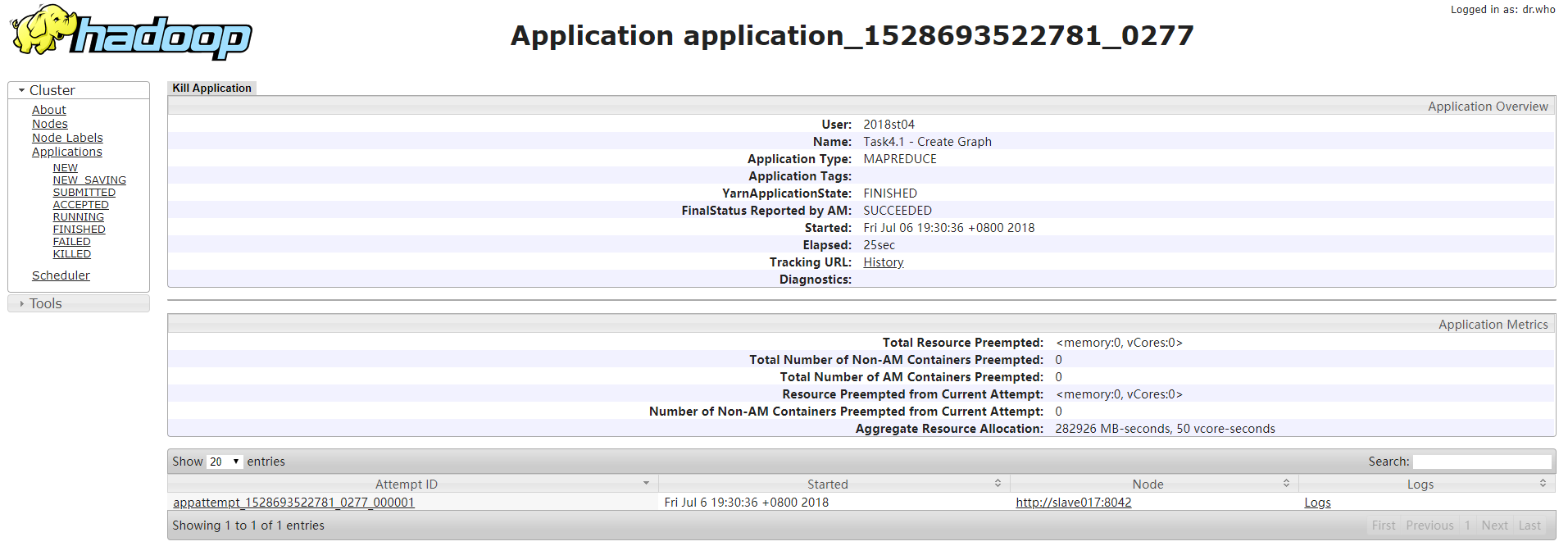
这些步骤放在了同一个MapReduce任务中一次性完成。



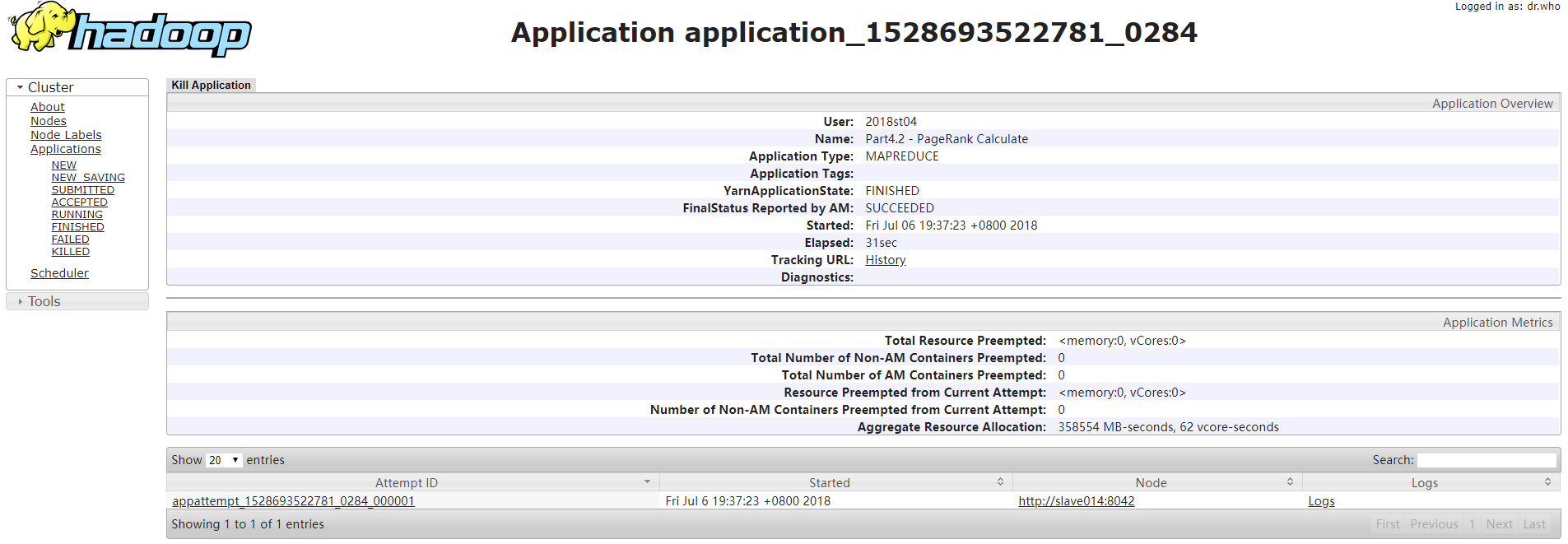
## PageRank

PageRank过程分为预处理、迭代计算PageRank值、排序输出结果三个过程。

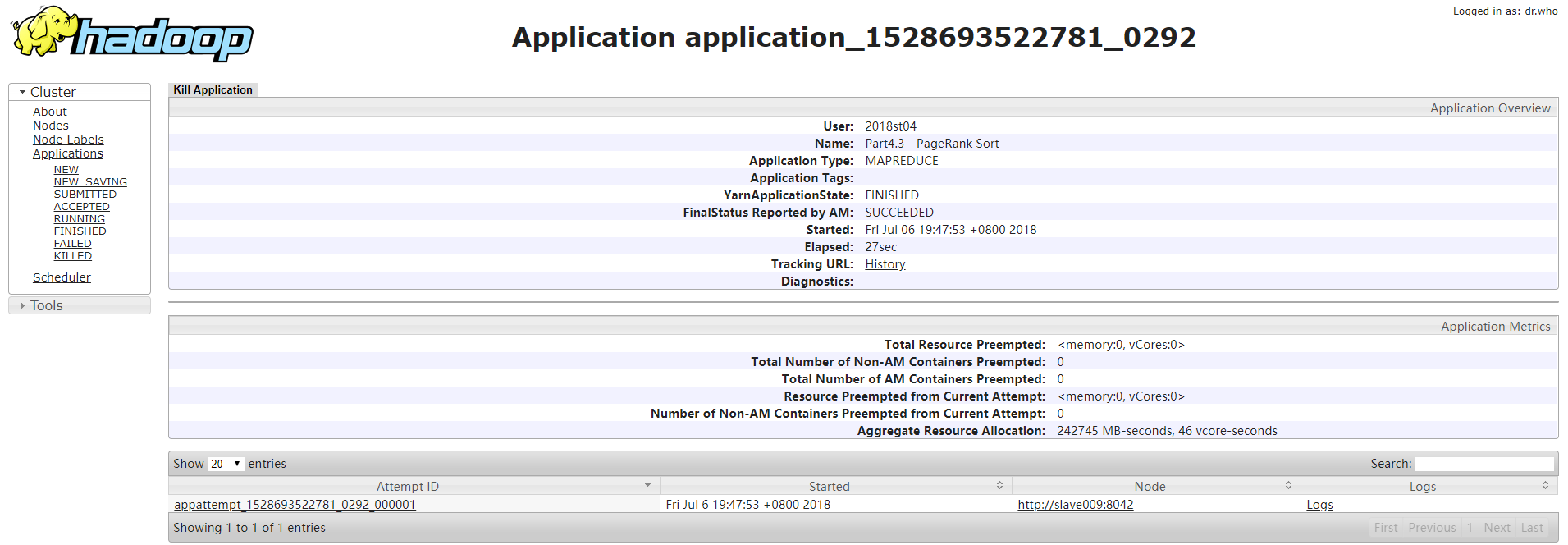
### 预处理



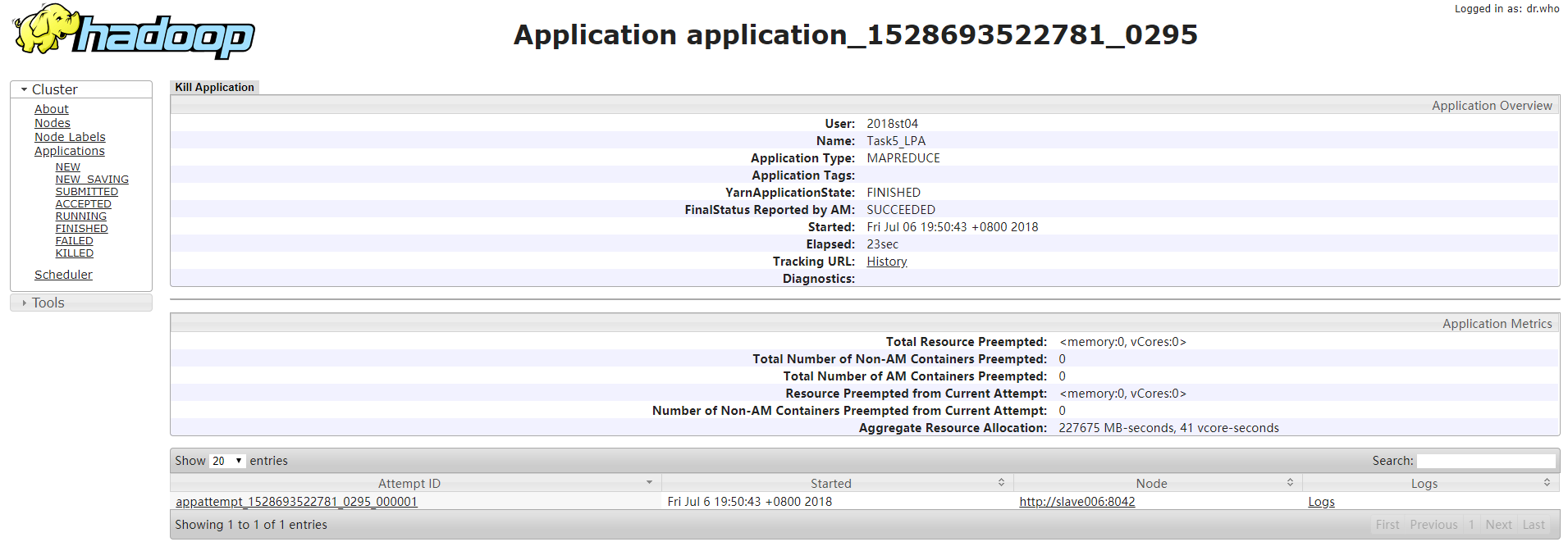
### PageRank迭代计算



### 排序输出结果



## 标签传播算法



## 提交次数

C:\Users\LL\Documents\Tencent Files\386506260\Image\C2C\`OLU2AL(OOI7NN@CQNG%]ER.png

# 小组分工

报告由小组共同完成，项目由Github托管，所有人协同完成。

分工如下：

刘鎏：统计邻接表和同现信息，项目管理与整合

李一鸣：PageRank算法

刘洋：标签传播算法

李书城：综合输出结果，数据可视化

# 目录说明以及测试方法

## 运行环境

Java 1.7、Hadoop 2.7.1

## 运行方法

Job的输入输出目录在程序内已设置好

hdfs dfs -rm -r /user/2018st04/\* %删除输出目录下的文件

hdfs dfs -put tags.txt /user/2018st04/RawTag.txt %将初始化的标签表提交到HDFS

hadoop jar JinYongNovelAnalysis-1.0-jar-with-dependencies.jar %运行jar包

注：jar包支持带参数运行，格式为hadoop jar JinYongNovelAnalysis-1.0-jar-with-dependencies.jar args0 args1

如果没有参数，那么默认运行全部任务。

如果args0为task1或者task2或者task3，那么只运行统计邻接表和同现信息任务。

如果args0为task4且args2为pre，那么只运行PageRank预处理。

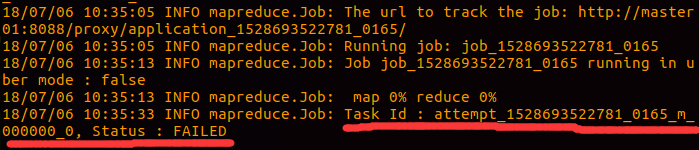
如果args0为task4且args2为loop\*(\*表示0-9)，那么只运行PageRank迭代过程，且从第\*次循环开始。

如果args0为task4且args2为final，那么只运行PageRank最终的排序输出处理。

如果args0为task5且args2为loop\*(\*表示0-2)，那么只运行标签传播的迭代过程，且从第\*次循环开始。

其他参数无效，会直接退出。

如果遇到如下**集群问题**，可以Ctrl+C强制终止，并通过删除当前MapReduce的单个输出目录，然后用带参数执行的方式从当前循环/当前任务开始重新执行即可：



## 源代码编译方法

使用IDEA打开工程，等待maven将依赖项加载完成，使用mvn clean install的方式生成可执行的jar包即可进行测试。

如果要运行RawTagCreator和GraphCsvMaker，只需用普通的Java编译即可。

其中，如果要运行GraphCsvMaker，需要用参数指定目录，五个参数分别为：标签文件路径、邻接表文件路径、PageRank文件路径、节点数据输出路径、边数据输出路径。

## 文档结构

|--

|--result %生成的结果目录

|--ReadNovelOutput %邻接关系及同现概率的结果

|--PRData0 %PageRank预处理结果

|--PRData[1-10] %PageRank迭代结果

|--FinalRank %PageRank最终结果

|--RawTagPre.txt %标签预处理结果

|--RawTag[0-2] %中间输出结果，debug专用

|--RawTag.txt %标签最终结果

|--主角列表.txt %主角列表txt文件

|--edge.csv %边数据的csv文件

|--node.csv %点数据的csv文件

|--src\main\java %源代码目录

|--LPA.java %标签传播算法源代码

|--Main.java %Main类源代码

|--PageRank.java %PageRank算法源代码

|--RawTagCreator.java %读取主角信息生成初始标签表的源代码

|--ReadNovel.java %统计邻接信息和同现概率的源代码

|--GraphCsvMaker.java %综合其他输出，生成点和边数据的源代码

|-pom.xml %Maven配置文件

|-课程设计-4-李书城.docx %实验报告

附中文参考文献:

【1】深入理解大数据 大数据处理与编程实践,机械工业出版,2015年6月,黄宜华