大数据课程设计　金庸的江湖

王屹　（131220022、1012682755@qq.com）

周博聪　（131220024、bertram_zbc@163.com）

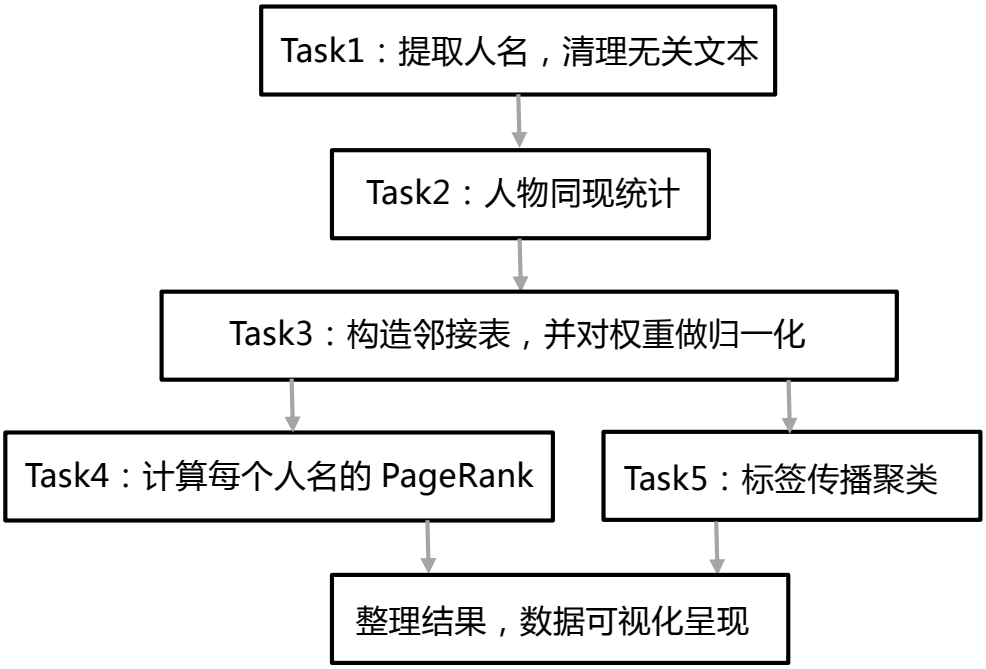
(南京大学 计算机科学与技术系, 南京　210093)

摘　要:　通过一个综合数据分析案例：“金庸的江湖——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，可以体会如何使用 MapReduce 完成一个综合性的数据挖掘任务，包括全流程的数据预处理、数据分析、数据后处理等。该任务主要涉及单词同现算法、数据整理与归一化、数据排序、PageRank、标签传播等算法，还涉及部分数据可视化的内容。

关键词:　MapReduce，Hadoop，数据挖掘，单词同现，PageRank，数据可视化，金庸。

1　项目综述

项目的目标是从金庸的 15 本小说中挖掘出一些有用的信息，比如谁是金庸整个江湖的主角，那些人是一个群体，他们之间联系比较紧密。我们的所有输入只有 15 本金庸小说和一个人名表，为了完成这个任务，我们的处理流程如下：



首先对数据做预处理，我们只需要人与人之间的关系，因此其他的文本就没有那么重要了，可以直接删去。然后我们对每一段中的人名做一个同现的统计，有了这样一个同现统计，就可以建立一个邻接表，用于

描述金庸小说中这些人物两两之间的关系有多密切，共现次数就是用来描述这个密切程度的参数。接下来为了方便后面的分析工作，我们将这个共现次数做归一化。再就是两个分析，通过 PageRank 可以找到整个江湖中 Rank 值较高的一些人，也就是主角；标签传播则可以帮我们给所有的人物归类。最后用 Gephi 将这些分析结果用直观的方式呈现出来。

2 任务一：数据预处理，人名的提取

2.1 任务描述

从原始的金庸小说文本中抽取出人物信息，即将人名无关的信息都删掉，只保留名字。这样做可以大大减少后面步骤的数据处理量。注意到，我们在做人物同现的时候，每一段中如果某个名字多次出现，实际只会统计一次。以及同现必须要至少有两个人存在，因此对某一段人名提取完后发现没有人名或者只有一个人名的情况，也可以过滤掉。

输入：未分词的全本金庸小说，金庸武侠小说人名列表。
输出：分词后，仅保留人名的金庸武侠小说，人名与人名之间用空格隔开。

样例：
输入：金庸 03 连城诀.txt 中某一段：
狄云和戚芳一走到万家大宅之前，瞧见那高墙朱门、挂灯结彩的气派，心中都是暗自嘀咕。戚芳紧紧拉住了父亲的衣袖。戚长发正待向门公询问，忽见卜垣从门里出来，心中一喜，叫道："卜贤侄，我来啦。"
输出：
狄云 戚芳 戚长发 卜垣

2.2 具体实现

2.2.1 实现思路

在提取人名的时候，我们先将这个人名表读取，存储到一个全局的数组中，检测每一行读入的文本是否含有某个人名。考虑到提取完名字后有很多重复的人名行，我们在 Map 过程中，将人名行作为 key，1 作为 value，表示这个人名行出现了一次。Reduce 过程首先统计对每个 key，value 的和是多少，然后就直接输出这个 value 和次的 key 即可，value 为空。

2.2.2 伪代码

Map：

```
class Mapper
method Setup //初始化
    person <- read person_list file into a array
end method
method Map(docid n,doc d)
    Temp <- new Array
    for all p <- person do
        Pattern pattern <- compiled by p //使用正则表达式匹配
        Matcher matcher <- line
```

```
            if find & not in Temp //去掉重复的名字
                Temp.add(p);
            end if
        end for
        outstring <- null
        if Temp.size > =2
            outstring <- append(Temp); //连接字符串
        end if
        Emit(outstring,1) //写入文件
    end method
    end class
```

Reduce：

```
class Reducer
method Reduce(t,{...})
    for all t <-values
        sum++
    end for
    for i=0 to sum
        emit(key,"")
end method
end class
```

3 任务二：特征抽取：人物的同现统计

3.1 任务描述

在人物同现分析中，如果两个人在原文的同一段落中出现，则认为两个人发生了一次同现关系。我们需要对人物之间的同现关系次数进行统计，同现关系次数越多，则说明两人的关系越密切。

输入：任务一的输出
输出：在金庸的所有武侠小说中，人物之间的同现次数。

样例：
输入：狄云 戚芳 戚芳 戚长发 卜垣
戚芳 卜垣 卜垣
输出：

狄云，戚芳	1	戚长发，狄云	1
狄云，戚长发	1	戚长发，戚芳	1
狄云，卜垣	1	戚长发，卜垣	1
戚芳，狄云	1	卜垣，狄云	1
戚芳，戚长发	1	卜垣，戚芳	2
戚芳，卜垣	2	卜垣，戚长发	1

3.2 具体实现

3.2.1 实现思路

这里的人物同现比常规的情况还要简单一些，因为人物 A 和人物 B 是无向的，<A,B>和<B,A>这两种情

况都要输出，因此实现的时候只要做二重循环，只要不是同一个人，都输出一个<A,B> 1 的键值对给 reduce 去统计。reduce 过程也很简单，只要统计每种不同的<A,B>有多少个就行了。

3.2.2 伪代码

Map:

```
class Mapper
method Map(docid n,doc d)
     temp <- new arrey
     temp <- d.split
     for all s <- temp
          for all t <- temp
               if s!=t //排除相同情况
                    Emit(s+t,1)
               end if
          end for
     end for
end method
end class
```

Reduce：

```
class Reducer
method Reduce(t,{1,1,1,...})
     for all d <- {1,1,1,...}
          Sum up(d)
     end for
end method
end class
```

4 任务三：人物关系图构建与特征归一化

4.1 任务描述

当获取了人物之间的共现关系之后，我们就可以根据共现关系，生成人物之间的关系图了。人物关系图使用邻接表的形式表示，方便后面的 PageRank 计算。在人物关系图中，人物是顶点，人物之间的互动关系是边。人物之间的互动关系靠人物之间的共现关系确定。如果两个人之间具有共现关系，则两个人之间就具有一条边。两人之间的共现次数体现出两人关系的密切程度，反映到共现关系图上就是边的权重。边的权重越高则体现了两个人的关系越密切。

输入：任务二的输出

输出：归一化权重后的人物关系图

样例：

输入：

狄云，戚芳	1	戚长发，狄云	1
狄云，戚长发	1	戚长发，戚芳	1
狄云，卜垣	1	戚长发，卜垣	1
戚芳，狄云	1	卜垣，狄云	1
戚芳，戚长发	1	卜垣，戚芳	2

戚芳，卜垣　　2　　卜垣，戚长发　　1

输出：狄云 [戚芳,0.33333|戚长发，0.333333|卜垣 0.333333]

戚芳 [狄云,0.25 |戚长发，0.25|卜垣 0.5]

戚长发 [狄云,0.33333|戚芳，0.333333|卜垣 0.333333]

卜垣 [狄云 0.25|戚芳,0.5|戚长发，0.25]

首先是将统计出的人物共现次数结果，转换成邻接表的形式表示：每一行表示一个邻接关系。

“戚芳 [狄云,0.25 |戚长发，0.25|卜垣 0.5] ”表示了顶点“戚芳”，有三个邻接点，分别是“狄云”、“戚长发”和“卜垣”，对应三条邻接边，每条邻接边上分别具有不同的权重。这个邻接边的权重是依靠某个人与其他人共现的“概率”得到的，以“戚芳”为例，她分别与三个人（“狄云”共现 1 次、“戚长发”，共现 1 次、“卜垣”共现 2 次）有共现关系，则戚芳与三个人共现的“概率”分别为 1/(1+1+2) = 0.25，1/(1+1+2) =0.25，2/(1+1+2) = 0.5。这三个“概率”值对应与三条边的权重。通过这种归一化，我们确保了某个顶点的出边权重的和为 1。

4.2 具体实现

4.2.1 实现思路

对于每一个输入“A,B　X”，其中，A,B 表示两个名字，X 表示这两个名字的同现次数，我们只需要构建一个以 A 为 key，多组 B，X 构成的 value 的长字符串。

因此 Map 和 reduce 过程如下：

Map：将输入字符串拆分成 key:A　value：B　X，发射到 reduce 节点，这样可以保证所有 key 相同的键值对都发射到同一个 reduce 节点。

Reduce：对于某个 key，reduce 收到了一组该 key 对应的 value，每个 value 的形式是“B　X”，对这些 X 求和得到 SUM，然后将 X 改成概率表示的形式：“B　P”，其中 P=X/SUM，最后将这些“B　P”拼接起来作为 value 输出。

4.2.2 伪代码

```
class Mapper
method Map(docid n,doc d)
     array str <- d.split("\t\t")
     array name <- d[0].split(",")
     Emit(name[0],name[1]+str[1])
 end method
end class

class Reducer
 method Reduce(t,{d1,n1 d2,n2 ...})
     n <- n1+n2+... //统计总次数
     name <- new array
     count <- new array
     for all d,n <- {d1,n1 d2,n2 ...}
         name.add(d)
         count.add(n)
     end for
     for i=0 to count.size
         weight = count[i]/sum //计算每个权重
         result = result + name[i] + weight
```

```
    end for
    Emit(t,result)//将 key 与每个权重写入文件
end method
end class
```

流程如下：

5 任务四：基于人物关系图的 PageRank 计算

5.1 任务描述

在给出人物关系图之后，我们就可以对人物关系图进行一个数据分析。其中一个典型的分析任务是：PageRank 值计算。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。

输入：任务 3 的输出
输出：每个人的 PageRank 值

5.2 具体实现

5.2.1 实现思路

PageRank 需要迭代，显然如果直接把任务 3 的输出作为我们的输入，似乎迭代的时候会不方便，比如说每个人物的 PageRank 值应该在首次迭代的时候就初始化好。因此我们修改了一下任务 3 的程序，单独给 PageRank 生成一个初始化输入，改动如下：

任务三原有结果：

一灯大师　　乔寨主,0.0022988506;华筝,0.0022988506;……

现在的结果：

一灯大师　　1&乔寨主,0.0022988506;华筝,0.0022988506;……

其中 1 表示初始的 PageRank 值，我们给所有人物的初始 PageRank 值都是 1，&符号仅仅为了在解析字符串时做分隔用。这样就确保了 PageRank 整个迭代周期都保证了统一的文件格式，在 Map 和 reduce 过程中就不需要针对第一次输入做额外的检测。

迭代过程中，有两个部分需要考虑，第一：为了保证文件格式的一致性，每个人名后面跟的一个“人名，权值”的长字符串要传给 reduce 以供写到文件。第二：计算新的 PageRank 值并更新，在实现的时候只要遍历每个人名后面的长字符串，计算出当前作为 key 的人名对每个人名 PageRank 值的贡献，举例：

一灯大师　　1&乔寨主,0.0022988506;华筝,0.0022988506;……

首先解析出有很多对“人名，权值”，比如“乔寨主，0.00229…”，那么我们将“乔寨主”作为 key，用“一灯大师”的 PageRank 值乘以这个权值，就得到了“一灯大师”对“乔寨主”的 PageRank 值的贡献，这个乘积作为 value，将这个键值对发射给 reduce 节点。

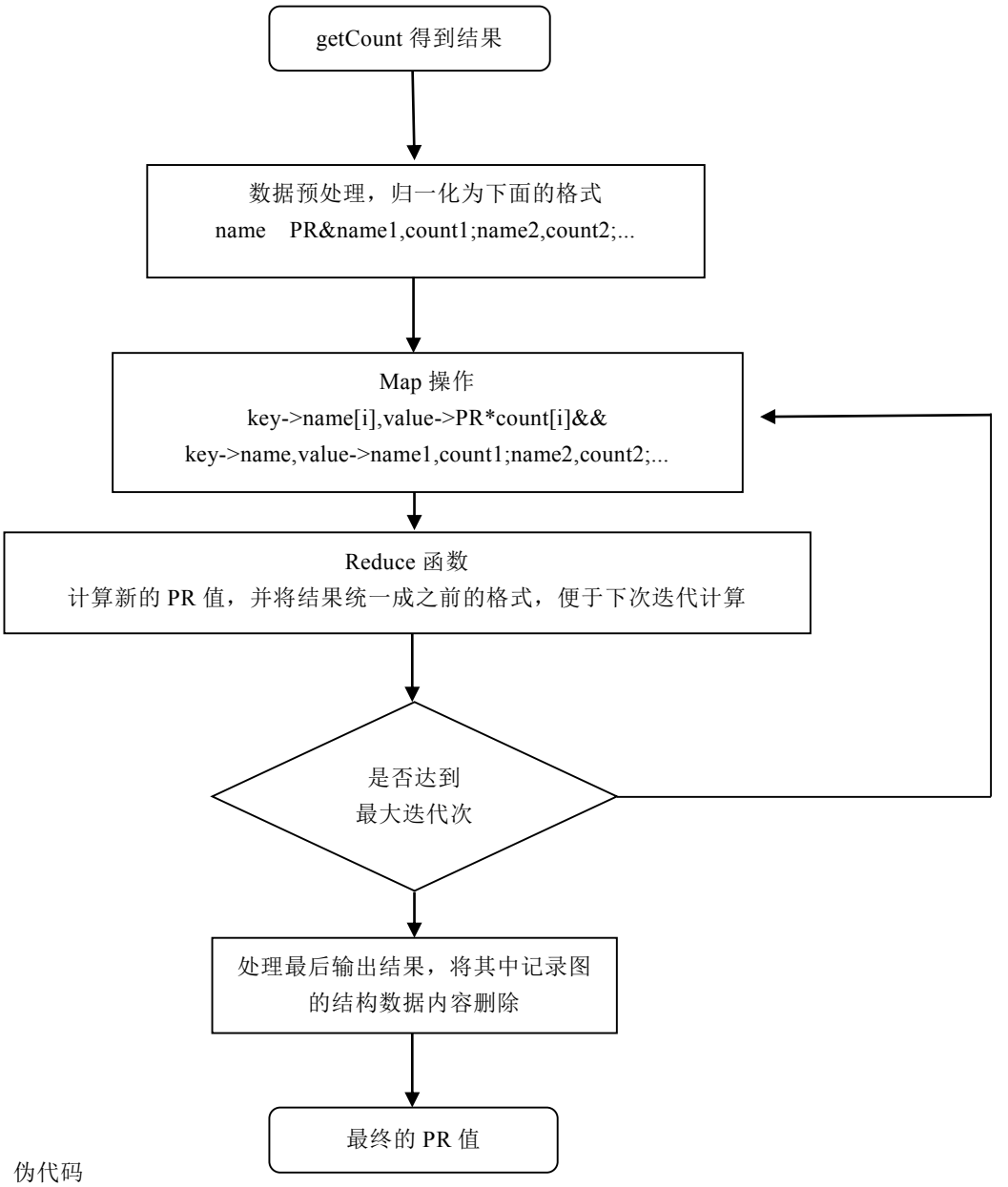
Reduce 过程接收这些键值对，将同一个人得到的不同贡献相加，就是它的新的 RageRank 值了，不过要注意的是，这里还应该引入一个阻尼系数，新的 PageRank 值最后被确定为：

$$1 - 0.85 + 0.85 * \text{PRsum}$$

一般来讲这个松弛系数取 0.85，PRsum 就是不同贡献值的和。

最后，做一步清理工作，一是只保留人名和 PageRank 值，二是利用 mapreduce 自带的排序，将 PageRank 值作为主键，得到一个从高到低排序的人名表。

5.2.2 流程图



5.2.3 伪代码

PageRank 主程序

```
class Mapper
```

```
method Map(docid n,doc d)
    array line <- d.split("\t")
    array temp <- line[1].split("&")
    PR <- temp[0] //迭代用的 PR 值
    Emit(line[0],temp[1]) //将记录写入文件以保留图的结构
    array NameAndWeight <- temp[1].split(";")
    for all n <- NameAndWeight
        weight <- n[1]
        name <- n[0]
        Emit(name,$&+(PR*weight)) //计算新的 PR 值的中间结果,并使用“$&”标记
    end for
end method
end class

class Reducer
method Reduce(t,{d1,d2,d3 ...})
    for all d <- {d1,d2,d3 ...}
        if d contains "$" //是 PR 的中间计算结果
            newPR <- d[1]
            PRsum += newPR
        else //是图的结构记录
            list <- d
        end
    end for
    Emit(t,1-0.8+0.8*PRsum+&+list)//计算新的 PR 值并结合图的结构写入文件
end method
end class
```

处理输出结果

```
class Mapper
method Map(docid n,doc d)
    array terms <- d.split("\t")
    if terms[1] contains "&"
        array temp <- terms[1].split("&")
        Emit(temp[0],term[0]) //将 PR 值放在前面，以便利用自带的排序
    end if
end method
end class

class Reducer
method Reduce(t,{d})
    Emit(t,d)
end method
end class
```

自定义降序排序

```
method (w1,w2)
return -1 * compare(w1,w2)
end method
```

6 任务五：在人物关系图上的标签传播

6.1 任务描述

标签传播（Label Propagation）是一种半监督的图分析算法，他能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（Community Detection）。该任务中，我们用每个聚类的核心人物作为这个类的标签，比如下面输出中，“张无忌”实际上是一个标签名，表示“杨逍”属于张无忌这个类。

输入：任务 3 的输出
输出：标签名　人名

样例：
输出：张无忌　杨逍

6.2 具体实现

6.2.1 实现思路

LPA 需要迭代，因此我们采用了和 PageRank 差不多的方式做了初始化，初始化文件的格式如下：

一灯大师　一灯大师&乔寨主,0.0022988506,乔寨主;华筝,0.0022988506,华筝;……

这里，“一灯大师”是 key，value 中，“&”前面的“一灯大师”表示 key 对应的标签，我们的初始化标签均和自己相同。后面，“乔寨主”是人名，中间数字是它的权重，后一个“乔寨主”是它的标签，在迭代的时候我们会更新这个标签，这里初始化的时候使用的是自己的名字。

迭代过程中最关键的是如何选取新的标签，遍历“&”后面的“人名，权值，标签”列表，我们要统计出那个贡献值最高的标签。举例：

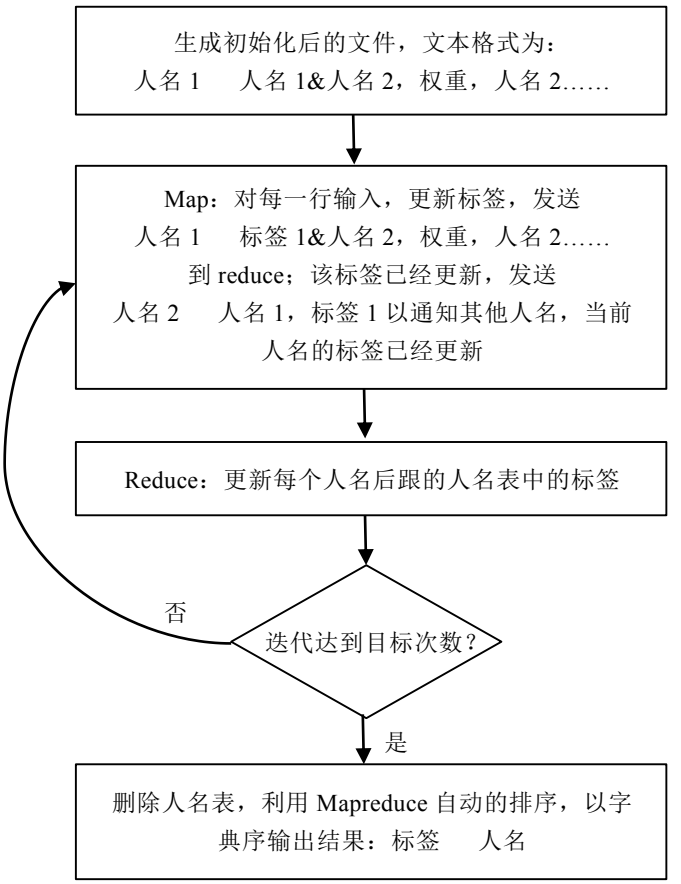
一灯大师　一灯大师&乔寨主,0.0022988506,乔寨主;华筝,0.0022988506,华筝;……

假设我们现在正在统计的就是这一行数据，在解析“&”后面的字符串时，除了第一次迭代之外，会出现多个人对应同一个标签的情况，因此需要一个数组来存储每个标签在当前字符串中的总权值，如果该标签还没出现，那把该标签新家进去，该标签的权值作为总权值的一个初值。比如说现在的这个数组是空的，我发现“乔寨主”的标签是“乔寨主”，那么这个数组里就应该添加一条记录：“乔寨主”这个标签的当前权值是 0.00229…，后面还有“乔寨主”的话，直接用那个权值加上当前的“乔寨主”的总权值即可。我们还需要一个数组记录下这整个字符串中的所有人名，这个后面会用到。现在我们已经有了一组“标签，权重”的记录，因此从中挑选权重最高的作为 key 的标签就可以了，注意输出到 reduce 的时候，为了保持迭代时文件格式一致，应当将这个长字符串一起发送给 reduce。我们还有一件事要做，告诉所有和当前 key 相关的其他 key，当前 key 对应的标签变了，所以我们输出遍历之前记录了整个字符串中所有人名的数组，以单个人名为 key，将更新过的当前的 key 和标签作为 value 也输出到 reduce 以供 reduce 更新。

Reduce 过程做的工作就是更新长字符串中的标签，最后按照统一格式写到文件。

最后还有一步清理工作，只保留标签和人名输出一个最终文件即可。

6.2.2 流程图



6.2.3 伪代码

```
class Mapper
method Map(docid n,doc d)
    HashMap labelAndWeight
       ArrayList Names
    array line <- d.split("\t")
    array temp <- line[1].split("&")
    array list <- temp.split("&")[1]
    array nameAndWeightSet <- list.split(";")
    for all s <- nameAndWeightSet
        Array nameAndWeight <- s.split(",")
        Label=nameAndWeight[2]
        Name=nameAndWeight[0]
        Weight=nameAndWeight[1]
        weightSum=getWeight(labelAndWeight)
        if(weightSum is null)
            weightSum=weight
        else
            weightSum=weightSum+weight
        Names.add(Name)
        labelAndWeight.put(Label,weightSum)
    end for
```

```
    LabelOfMax=getMaxLabel(labelAndWeight)
    if(LabelOfMax != null)
        Emit(line[0],LabelOfMax+"&"+list)
        for all n<-Names
            Emit(n,line[0]+","+LabelOfMax)
        end for
end method
end class

class Reducer
method Reduce(t,{d1,d2,d3 ...})
    HashMap nameAndLabel
    for All text <- values
    if(text contains "&")
        list=text.split("&")[1]
        Label=text.split("&")[0]
    else
        nameAndLabel.put(text.split(",")[0],text.split(",")[1])
    outstring.append(Label)
    outstring.append("&")
    tempList <- list.split(";")
    for all s <- tempList
        temp=s.split(",")
        outstring.append(temp[0].append(","))
        outstring.append(temp[1].append(","))
        outstring.append(nameAndLabel.get(temp[0])
        outstring.append(";")
    Emit(key,outstring)
end method
end class
```

7 实验结果

7.1 运行

在 driver 中我们提供了三种运行方式：

1. default 方式，所有的参数都内置好了，只需要输入：

   Hadoop jar xxx.jar default

   HDFS 上小说路径应为 novel，注意不能有 output 文件夹，否则会冲突。其中 PageRank 和 LPA 的迭代次数均为 20 次。

2. 迭代次数可选方式，输入：

   Hadoop jar xxx.jar [input path] [output path] num1 num2

   其中 num1 是 LPA 的迭代次数，num2 是 PageRank 的迭代次数。

3. 纯手动模式，输入：

   Hadoop jar xxx.jar single

   会出现如下选项：

   usage:

   num,inpath,outpath,times(times is only useful for LPA and PageRank)

   num:1->extract names

   num:2->get Count

   num:3->Normalize

num:4->LPA(LPA input path is the output of getCount)

num:5->Page Rank(PageRank input path is the output of getCount)

对于前三个任务，只需要输入数字来选择运行某个任务，紧跟着两个参数是输入输出路径。比如输入：

1,input,output

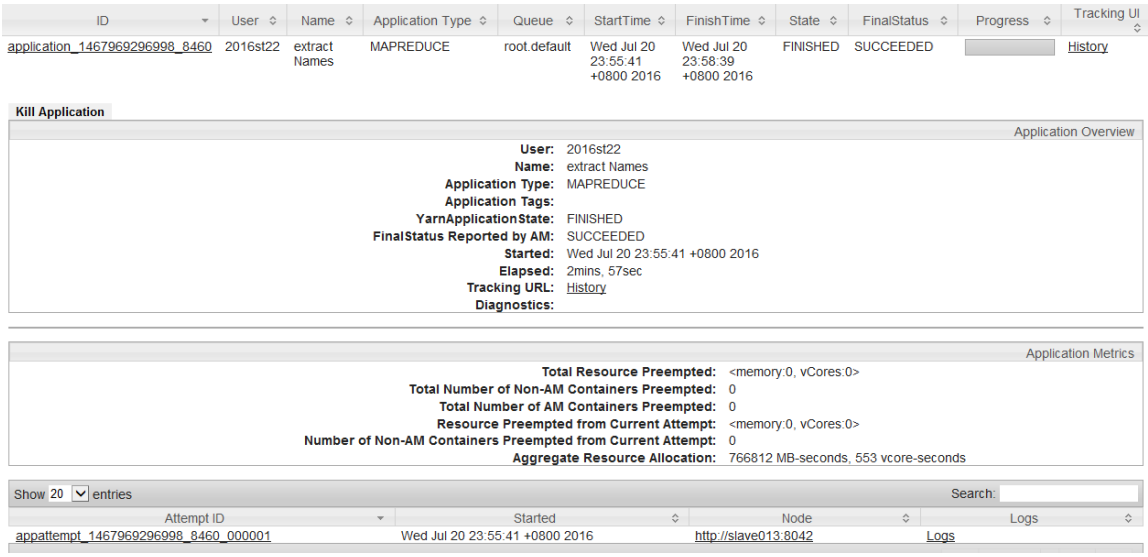
就可以运行第一个任务了，4,5 两个是 LPA 和 PageRank，在最后加一个迭代次数即可。

7.2 任务一：数据预处理，人名的提取的结果

部分结果如下：

```
1  一灯大师    周伯通
2  一灯大师    周伯通
3  丁不三  丁不四
4  丁不三  丁不四
5  丁不三  丁不四
6  丁不三  丁不四
7  丁不三  丁不四
8  丁不三  丁不四
9  丁不三  丁不四
10 丁不三  丁不四
11 丁不三  丁不四  封万里
12 丁不三  丁不四  封万里
13 丁不三  丁不四  封万里
14 丁不三  丁不四  小翠
15 丁不三  丁不四  石破天
```

集群截图如下：



ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI
application_1467969296998_8460	2016st22	extract Names	MAPREDUCE	root.default	Wed Jul 20 23:55:41 +0800 2016	Wed Jul 20 23:58:39 +0800 2016	FINISHED	SUCCEEDED		History

Kill Application

Application Overview

User:	2016st22
Name:	extract Names
Application Type:	MAPREDUCE
Application Tags:	
YarnApplicationState:	FINISHED
FinalStatus Reported by AM:	SUCCEEDED
Started:	Wed Jul 20 23:55:41 +0800 2016
Elapsed:	2mins, 57sec
Tracking URL:	History
Diagnostics:	

Application Metrics

Total Resource Preempted:	<memory:0, vCores:0>
Total Number of Non-AM Containers Preempted:	0
Total Number of AM Containers Preempted:	0
Resource Preempted from Current Attempt:	<memory:0, vCores:0>
Number of Non-AM Containers Preempted from Current Attempt:	0
Aggregate Resource Allocation:	766812 MB-seconds, 553 vcore-seconds

Show 20 entries Search:

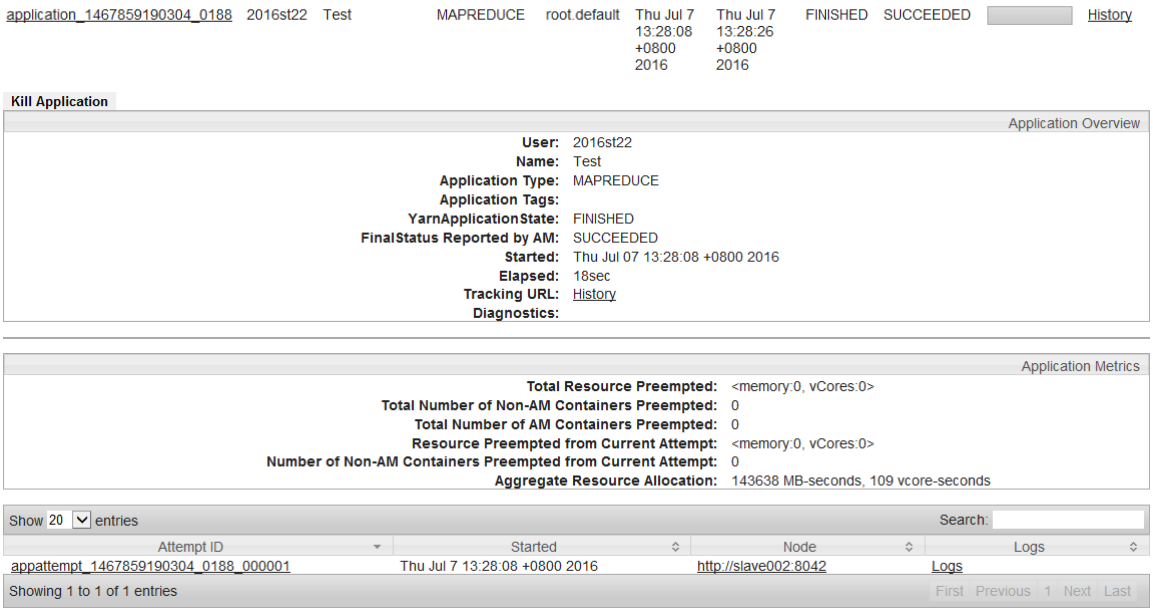
Attempt ID	Started	Node	Logs
appattempt_1467969296998_8460_000001	Wed Jul 20 23:55:41 +0800 2016	http://slave013:8042	Logs

7.3 任务二：特征抽取：人物的同现统计

部分结果如下：

```
1  一灯大师,上官        1
2  一灯大师,丘处机      5
3  一灯大师,乔寨主      1
4  一灯大师,农夫        17
5  一灯大师,华筝        1
6  一灯大师,卫璧        2
7  一灯大师,吕文德      1
8  一灯大师,周伯通      28
9  一灯大师,哑巴        1
10 一灯大师,哑梢公      1
11 一灯大师,大汉        1
12 一灯大师,天竺僧      4
13 一灯大师,天竺僧人        3
14 一灯大师,完颜萍      2
15 一灯大师,小沙弥      3
```

集群截图如下：

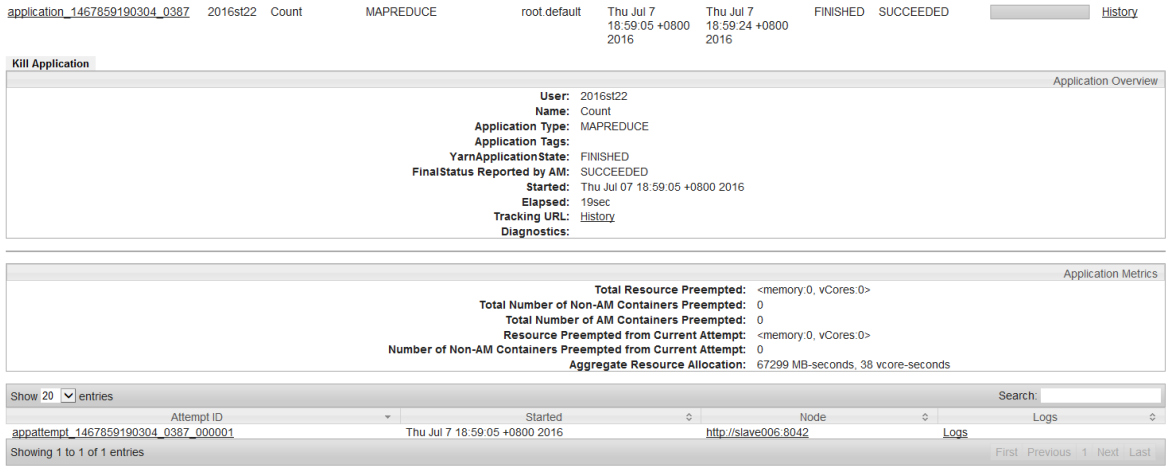


7.4 任务三：人物关系图构建与特征归一化

部分结果如下：

```
一灯大师	乔寨主,0.0022988506;华等,0.0022988506;卫璧,0.004597701;吕文德,
丁不三	小翠,0.012048192;大悲老人,0.006024096;李万山,0.006024096;柯万钧,0.
丁不四	封万里,0.015873017;小翠,0.04761905;史小翠,0.022222223;尤得胜,0.003
丁典	空心菜,0.0042918455;万圭,0.047210302;万震山,0.017167382;农夫,0.004
丁勉	令狐冲,0.06818182;任我行,0.011363637;何足道,0.011363637;冲虚,0.011
丁同	李文秀,0.5555556;霍元龙,0.11111111;老头子,0.11111111;李三,0.222222
丁坚	胖子,0.014705882;丹青生,0.14705883;令狐冲,0.22058824;任我行,0.0147
丁大全	郭襄,0.05882353;陈大方,0.23529412;小王将军,0.11764706;宋五,0.05882
丁敏君	杨不悔,0.021645023;何太冲,0.017316017;俞岱岩,0.0043290043;卫璧,0.0
丁春秋	公冶乾,0.018897638;不平道人,0.0047244094;乌老大,0.007874016;乔峰,0
丁游	袁承志,0.15;刘培生,0.05;大汉,0.1;孟伯飞,0.1;孟铮,0.05;孟铸,0.05;归
万圭	凌退思,0.005988024;卜垣,0.035928145;吴坎,0.0748503;周圻,0.01197604
万大平	刘菁,0.16666667;史登达,0.33333334;汉子,0.16666667;刘正风,0.3333333
万庆澜	杨成协,0.05147059;上官,0.007352941;书僮,0.014705882;余鱼同,0.01470
万里风	刘培生,0.0625;梅剑,0.09375;梅剑和,0.09375;洞玄,0.0625;焦公礼,0.125
```

集群截图如下：



7.5 任务四：基于人物关系图的PageRank计算

中间迭代结果如下（第 10 次迭代）：

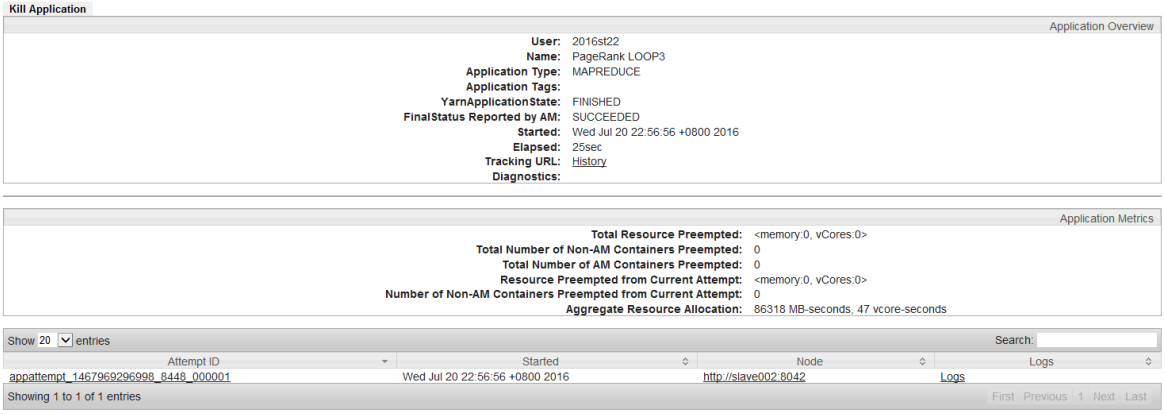
```
一灯大师	1.3170335251485812&乔寨主,0.0022988506;华等,0.0022988506;卫璧,0.0
丁不三	1.300001210752789&小翠,0.012048192;大悲老人,0.006024096;李万山,0.006
丁不四	2.113710331653162&封万里,0.015873017;小翠,0.04761905;史小翠,0.022222
丁典	1.636335340164677&空心菜,0.0042918455;万圭,0.047210302;万震山,0.0171
丁勉	0.6152647713174106&令狐冲,0.06818182;任我行,0.011363637;何足道,0.011
丁同	0.2679345006987132&李文秀,0.5555556;霍元龙,0.11111111;老头子,0.11111
丁坚	0.48002606267449494&胖子,0.014705882;丹青生,0.14705883;令狐冲,0.2205
丁大全	0.5750828441877988&郭襄,0.05882353;陈大方,0.23529412;小王将军,0.1176
丁敏君	1.0164985013570387&杨不悔,0.021645023;何太冲,0.017316017;俞岱岩,0.00
丁春秋	2.990085730891159&公冶乾,0.018897638;不平道人,0.0047244094;乌老大,0.0
丁游	0.31952326050108126&袁承志,0.15;刘培生,0.05;大汉,0.1;孟伯飞,0.1;孟铮,
万圭	1.9228328464951443&凌退思,0.005988024;卜垣,0.035928145;吴坎,0.074850
万大平	0.30569524099017636&刘菁,0.16666667;史登达,0.33333334;汉子,0.166666
万庆澜	0.6278272275880228&杨成协,0.05147059;上官,0.007352941;书僮,0.014705
万里风	0.3738174414530879&刘培生,0.0625;梅剑,0.09375;梅剑和,0.09375;洞玄,0.0
```

最终结果如下：

```
32.17037330341801    韦小宝
18.918754683600465   令狐冲
18.04737650892436    张无忌
14.496420047779958   郭靖
13.324667465017745   杨过
13.19604530212978    袁承志
12.705785774418702   段誉
12.690495185474713   胡斐
12.521543368003876   汉子
12.261053771496547   黄蓉
8.714854397988535    陈家洛
7.444010662826274    石破天
7.286120405615178    吴三桂
6.907639712643204    岳不群
6.508123974401445    大汉
6.482460234967278    赵敏
6.272255425284179    谢逊
5.972001079932537    狄云
5.583707556020852    张三
5.503026872756377    小龙女
5.454537072855865    虚竹
5.3643092929276515   乔峰
5.205741340893409    文泰来
5.1832490546619345   骆冰
5.115116590146002    徐天宏
4.888770755698396    林平之
4.871926907441662    王语嫣
4.752782722710236    张翠山
4.710259322508395    杨逍
```

集群截图如下：

PageRank 迭代较多，只选择了其中某一次解了详细结果。



ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI
application_1467969296998_8428	2016st22	PR Cleanup	MAPREDUCE	root.default	Wed Jul 20 21:51:25 +0800 2016	Wed Jul 20 21:51:51 +0800 2016	FINISHED	SUCCEEDED		History
application_1467969296998_8427	2016st22	PageRank LOOP20	MAPREDUCE	root.default	Wed Jul 20 21:50:50 +0800 2016	Wed Jul 20 21:51:20 +0800 2016	FINISHED	SUCCEEDED		History
application_1467969296998_8426	2016st22	PageRank LOOP19	MAPREDUCE	root.default	Wed Jul 20 21:50:13 +0800 2016	Wed Jul 20 21:50:45 +0800 2016	FINISHED	SUCCEEDED		History
application_1467969296998_8425	2016st22	PageRank LOOP18	MAPREDUCE	root.default	Wed Jul 20 21:49:37 +0800 2016	Wed Jul 20 21:50:08 +0800 2016	FINISHED	SUCCEEDED		History
application_1467969296998_8424	2016st22	PageRank LOOP17	MAPREDUCE	root.default	Wed Jul 20 21:49:02 +0800 2016	Wed Jul 20 21:49:36 +0800 2016	FINISHED	SUCCEEDED		History
application_1467969296998_8423	2016st22	PageRank LOOP16	MAPREDUCE	root.default	Wed Jul 20 21:48:31 +0800 2016	Wed Jul 20 21:49:00 +0800 2016	FINISHED	SUCCEEDED		History
application_1467969296998_8422	2016st22	PageRank LOOP15	MAPREDUCE	root.default	Wed Jul 20 21:47:20 +0800 2016	Wed Jul 20 21:48:27 +0800 2016	FINISHED	SUCCEEDED		History
application_1467969296998_8420	2016st22	PageRank LOOP14	MAPREDUCE	root.default	Wed Jul 20 21:45:54 +0800 2016	Wed Jul 20 21:47:15 +0800 2016	FINISHED	SUCCEEDED		History
application_1467969296998_8418	2016st22	PageRank LOOP13	MAPREDUCE	root.default	Wed Jul 20 21:44:38 +0800 2016	Wed Jul 20 21:45:48 +0800 2016	FINISHED	SUCCEEDED		History
application_1467969296998_8416	2016st22	PageRank LOOP12	MAPREDUCE	root.default	Wed Jul 20 21:43:27 +0800 2016	Wed Jul 20 21:44:35 +0800 2016	FINISHED	SUCCEEDED		History
application_1467969296998_8414	2016st22	PageRank LOOP11	MAPREDUCE	root.default	Wed Jul 20 21:42:12 +0800 2016	Wed Jul 20 21:43:23 +0800 2016	FINISHED	SUCCEEDED		History
application_1467969296998_8412	2016st22	PageRank LOOP10	MAPREDUCE	root.default	Wed Jul 20 21:40:56 +0800 2016	Wed Jul 20 21:42:07 +0800 2016	FINISHED	SUCCEEDED		History
application_1467969296998_8410	2016st22	PageRank LOOP9	MAPREDUCE	root.default	Wed Jul 20 21:39:40 +0800 2016	Wed Jul 20 21:40:51 +0800 2016	FINISHED	SUCCEEDED		History
application_1467969296998_8408	2016st22	PageRank LOOP8	MAPREDUCE	root.default	Wed Jul 20 21:38:29 +0800 2016	Wed Jul 20 21:39:36 +0800 2016	FINISHED	SUCCEEDED		History
application_1467969296998_8406	2016st22	PageRank LOOP7	MAPREDUCE	root.default	Wed Jul 20 21:37:09 +0800 2016	Wed Jul 20 21:38:25 +0800 2016	FINISHED	SUCCEEDED		History
application_1467969296998_8404	2016st22	PageRank LOOP6	MAPREDUCE	root.default	Wed Jul 20 21:35:52	Wed Jul 20 21:37:03	FINISHED	SUCCEEDED		History

7.6 任务五：在人物关系图上的标签传播

中间结果如下：

```
1  一灯大师    郭靖&乔寨主,0.0022988506,郭靖;华箏,0.0022988506,郭靖;卫璧,0.
2  丁不三    石破天&小翠,0.012048192,石破天;大悲老人,0.006024096,石破天;李万山
3  丁不四    石破天&封万里,0.015873017,石破天;小翠,0.04761905,石破天;史小翠,0
4  丁典    狄云&空心菜,0.0042918455,狄云;万圭,0.047210302,狄云;万震山,0.017
5  丁勉    令狐冲&令狐冲,0.06818182,令狐冲;任我行,0.011363637,令狐冲;何足道
6  丁同    苏普&李文秀,0.5555556,苏普;霍元龙,0.11111111,苏普;老头子,0.11111
7  丁坚    令狐冲&胖子,0.014705882,郭靖;丹青生,0.14705883,令狐冲;令狐冲,0.2
8  丁大全    韦小宝&郭襄,0.05882353,杨过;陈大方,0.23529412,韦小宝;小王将军,0.
9  丁敏君    张无忌&杨不悔,0.021645023,张无忌;何太冲,0.017316017,张无忌;俞岱岩
10 丁春秋    段誉&公冶乾,0.018897638,段誉;不平道人,0.0047244094,段誉;乌老大,0
11 丁游    袁承志&袁承志,0.15,袁承志;刘培生,0.05,袁承志;大汉,0.1,韦小宝;孟伯
12 万圭    狄云&凌退思,0.005988024,狄云;卜垣,0.035928145,狄云;吴坎,0.074850
13 万大平    令狐冲&刘菁,0.16666667,令狐冲;史登达,0.33333334,令狐冲;汉子,0.16
14 万庆澜    陈家洛&杨成协,0.05147059,陈家洛;上官,0.007352941,令狐冲;书僮,0.0
15 万里风    袁承志&刘培生,0.0625,袁承志;梅剑,0.09375,袁承志;梅剑和,0.09375,袁
```

最终结果如下：

1	令狐冲	木高峰
2	令狐冲	张夫人
3	令狐冲	钟镇
4	令狐冲	吴天德
5	令狐冲	刘正风
6	令狐冲	丁勉
7	令狐冲	葛长老
8	令狐冲	丁坚
9	令狐冲	向问天
10	令狐冲	蓝凤凰
11	令狐冲	建除
12	令狐冲	上官
13	令狐冲	桃干仙
14	令狐冲	刘芹

167	张无忌	杨不悔
168	张无忌	杜百当
169	张无忌	乌旺阿普
170	张无忌	李四摧
171	张无忌	李四
172	张无忌	朱九真
173	张无忌	朱长龄
174	张无忌	朱元璋
175	张无忌	马法通
176	张无忌	乔福
177	张无忌	云鹤
178	张无忌	五姑
179	张无忌	韩林儿
180	张无忌	韩千叶

774	袁承志	水云道人
775	袁承志	水鉴
776	袁承志	沙广天
777	袁承志	归辛树
778	袁承志	归二娘
779	袁承志	张若谷
780	袁承志	沙老大
781	袁承志	张朝唐
782	袁承志	张春九
783	袁承志	洞玄
784	袁承志	义生
785	袁承志	洪胜海
786	袁承志	张信
787	袁承志	温仪

集群截图如下：

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
5778	0	1	5777	8	9 GB	104 GB	0 B	8	104	0	13	0	0	0	0

User Metrics for dr.who

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Containers Pending	Containers Reserved	Memory Used	Memory Pending	Memory Reserved	VCores Used	VCores Pending	VCores Reserved
0	0	0	0	0	0	0 B	0 B	0 B	0	0	0	

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation
Fair Scheduler	[MEMORY, CPU]	<memory:1024, vCores:1>	<memory:8192, vCores:8>

Show 20 entries Search: 2016st22

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI
application_1467969296998_5909	2016st22	LPA final	MAPREDUCE	root.default	Sun Jul 17 15:43:12 +0800 2016	Sun Jul 17 15:43:42 +0800 2016	FINISHED	SUCCEEDED		History
application_1467969296998_5908	2016st22	LPA LOOP20	MAPREDUCE	root.default	Sun Jul 17 15:42:46 +0800 2016	Sun Jul 17 15:43:08 +0800 2016	FINISHED	SUCCEEDED		History
application_1467969296998_5907	2016st22	LPA LOOP19	MAPREDUCE	root.default	Sun Jul 17 15:42:19 +0800 2016	Sun Jul 17 15:42:42 +0800 2016	FINISHED	SUCCEEDED		History
application_1467969296998_5905	2016st22	LPA LOOP18	MAPREDUCE	root.default	Sun Jul 17 15:41:48 +0800 2016	Sun Jul 17 15:42:16 +0800 2016	FINISHED	SUCCEEDED		History
application_1467969296998_5904	2016st22	LPA LOOP17	MAPREDUCE	root.default	Sun Jul 17 15:41:17 +0800 2016	Sun Jul 17 15:41:44 +0800 2016	FINISHED	SUCCEEDED		History
application_1467969296998_5902	2016st22	LPA LOOP16	MAPREDUCE	root.default	Sun Jul 17 15:40:56 +0800 2016	Sun Jul 17 15:41:14 +0800 2016	FINISHED	SUCCEEDED		History
application_1467969296998_5901	2016st22	LPA LOOP15	MAPREDUCE	root.default	Sun Jul 17 15:40:30 +0800 2016	Sun Jul 17 15:40:50 +0800 2016	FINISHED	SUCCEEDED		History
application_1467969296998_5899	2016st22	LPA LOOP14	MAPREDUCE	root.default	Sun Jul 17 15:40:09 +0800 2016	Sun Jul 17 15:40:27 +0800 2016	FINISHED	SUCCEEDED		History
application_1467969296998_5898	2016st22	LPA LOOP13	MAPREDUCE	root.default	Sun Jul 17 15:39:48 +0800 2016	Sun Jul 17 15:40:07 +0800 2016	FINISHED	SUCCEEDED		History
application_1467969296998_5897	2016st22	LPA LOOP12	MAPREDUCE	root.default	Sun Jul 17 15:39:28 +0800 2016	Sun Jul 17 15:39:46 +0800 2016	FINISHED	SUCCEEDED		History
application_1467969296998_5896	2016st22	LPA LOOP11	MAPREDUCE	root.default	Sun Jul 17 15:39:02 +0800 2016	Sun Jul 17 15:39:21 +0800 2016	FINISHED	SUCCEEDED		History
application_1467969296998_5895	2016st22	LPA LOOP10	MAPREDUCE	root.default	Sun Jul 17 15:38:41 +0800 2016	Sun Jul 17 15:39:00 +0800 2016	FINISHED	SUCCEEDED		History
application_1467969296998_5893	2016st22	LPA LOOP9	MAPREDUCE	root.default	Sun Jul 17 15:38:20 +0800 2016	Sun Jul 17 15:38:39 +0800 2016	FINISHED	SUCCEEDED		History
application_1467969296998_5892	2016st22	LPA LOOP8	MAPREDUCE	root.default	Sun Jul 17 15:38:00 +0800 2016	Sun Jul 17 15:38:18 +0800 2016	FINISHED	SUCCEEDED		History
application_1467969296998_5891	2016st22	LPA LOOP7	MAPREDUCE	root.default	Sun Jul 17 15:37:34 +0800 2016	Sun Jul 17 15:37:54 +0800 2016	FINISHED	SUCCEEDED		History
application_1467969296998_5890	2016st22	LPA LOOP6	MAPREDUCE	root.default	Sun Jul 17 15:37:08 +0800 2016	Sun Jul 17 15:37:27 +0800 2016	FINISHED	SUCCEEDED		History
application_1467969296998_5888	2016st22	LPA LOOP5	MAPREDUCE	root.default	Sun Jul 17 15:36:42 +0800 2016	Sun Jul 17 15:37:01 +0800 2016	FINISHED	SUCCEEDED		History
application_1467969296998_5887	2016st22	LPA LOOP4	MAPREDUCE	root.default	Sun Jul 17 15:36:21 +0800 2016	Sun Jul 17 15:36:40 +0800 2016	FINISHED	SUCCEEDED		History
application_1467969296998_5886	2016st22	LPA LOOP3	MAPREDUCE	root.default	Sun Jul 17 15:35:59 +0800 2016	Sun Jul 17 15:36:18 +0800 2016	FINISHED	SUCCEEDED		History
application_1467969296998_5885	2016st22	LPA LOOP2	MAPREDUCE	root.default	Sun Jul 17 15:35:33 +0800 2016	Sun Jul 17 15:35:52 +0800 2016	FINISHED	SUCCEEDED		History

Showing 1 to 20 of 22 entries (filtered from 5,778 total entries) First Previous 1 2 Next Last

Kill Application

Application Overview

User:	2016st22
Name:	LPA LOOP1
Application Type:	MAPREDUCE
Application Tags:	
YarnApplicationState:	FINISHED
FinalStatus Reported by AM:	SUCCEEDED
Started:	Sun Jul 17 15:35:07 +0800 2016
Elapsed:	19sec
Tracking URL:	History
Diagnostics:	

Application Metrics

Total Resource Preempted:	<memory:0, vCores:0>
Total Number of Non-AM Containers Preempted:	0
Total Number of AM Containers Preempted:	0
Resource Preempted from Current Attempt:	<memory:0, vCores:0>
Number of Non-AM Containers Preempted from Current Attempt:	0
Aggregate Resource Allocation:	64408 MB-seconds, 37 vcore-seconds

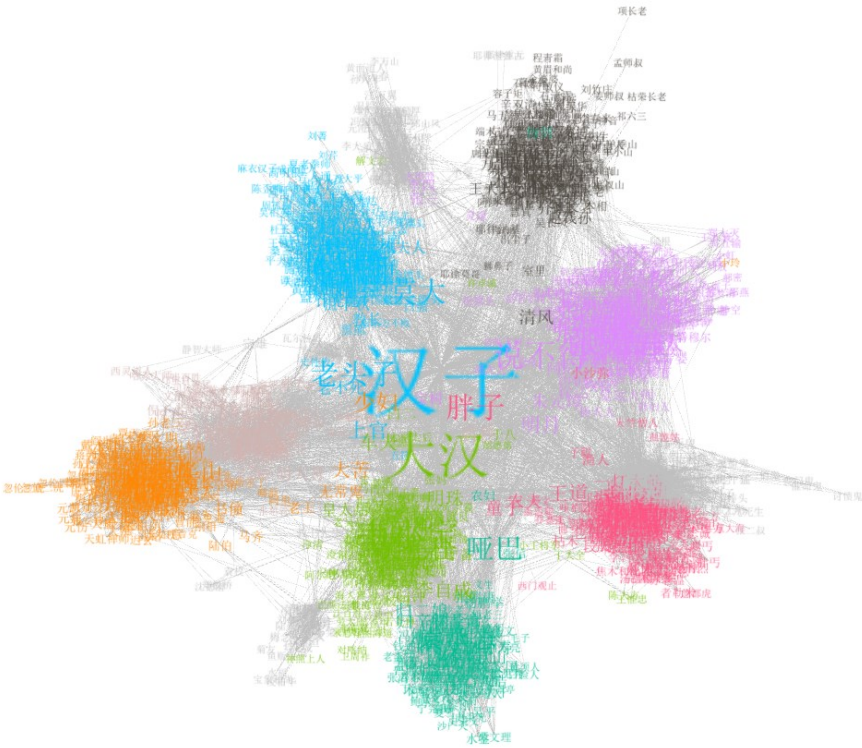
Show 20 entries Search:

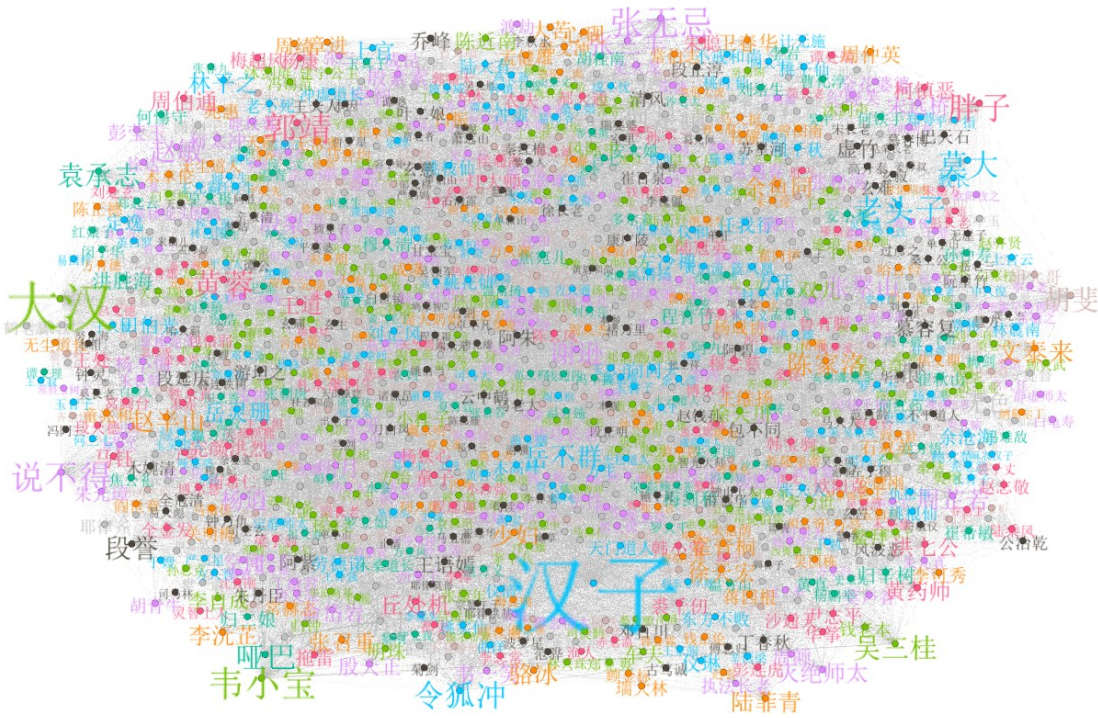
Attempt ID	Started	Node	Logs
appattempt_1467969296998_5884_000001	Sun Jul 17 15:35:07 +0800 2016	http://slave003:8042	Logs

Showing 1 to 1 of 1 entries First Previous 1 Next Last

7.7 数据可视化

第一张图是聚类结果，我们看一看到汉子、大汉等处于整个金庸江湖中的主角地位，实际上令狐冲，韦小宝，张无忌等也很大，但是因为被背景色掩盖，不是很清楚，可以看第二张图。第二张图虽然没有展示聚类结果，但是把一些主要角色都突出了，可以很明显地看到韦小宝、吴三桂、张无忌、袁承志、令狐冲等人很清晰地显现了出来。





8 优化

8.1 人名提取方式的选择

我们并没有采用 Ansj_seg 来做分词，提取人名的方式有两种，一是根据人名表，直接对每一行输入文本做字符串匹配，也就是我们采用的方法。二是对文本做分词，根据词性识别出人名。从效率来讲方法二要比方法一快不少，但实际上当我在检查分词版本的《越女剑》（并未提供人名表）的时候，发现“阿青”这个人物，在分词版本的小说中显示的是“阿 青”（中间有空格），这个时候用“阿青”去匹配文中的词是匹配不到的，也就会导致某本小说中某个重要角色因为匹配问题而完全缺失，这是我们不想看到的。因此我怀疑用词性的方式分词没办法保证 100%正确，正如我前面举得“阿 青”的例子。但是这种方法在处理人名特别多的时候，效果应该会比后者要好很多（即使这个时候牺牲一小部分的正确率）后者虽然比较粗暴，但是可以保证 100%的正确率。考虑到金庸小说中一共只有 1000 多个人物，因此用后者并不会有太多的性能劣势。

8.2 LPA和PageRank初始化

LPA 和 PageRank 在实现的时候也可以直接用归一化的结果做输入，只不过归一化的结果在第一次迭代中需要加工，也就是说在第一次迭代中我们需要有一个分支去专门处理首次迭代的情况，比如 PageRank 就需要为每个人名初始化一个 PageRank 值，从代码的简洁性角度来讲，这并不是一种很好的实现方式。因此我们就在归一化的输出结果中稍微加了一些统一的初始化值，一些标记等等，这样使我们的两个数据分析程序显得相对简洁明白。

8.3 标签传播算法的选择

我们在一开始想采用一个简单版本的标签传播算法，即不考虑权重，每个节点选择自己邻居节点中最频

繁出现的标签作为自己的标签，在测试时我们发现这种传播方式的效果不是很好，一方面，当出现多个标签出现次数相同的情况，随机选择产生的震荡问题会比较严重；二，得到的标签不一定是这个集合的主要角色。而加入了权值之后，从一开始高权值的节点就对其他节点有更大的影响力，随着不断迭代，它对其他节点的影响也不断被强化。

9 性能分析

由于服务器近期节点情况较差，而且本任务在本机上速度也较快，因此我就直接在本机上做了时间的统计。PageRank 和 LPA 都做了 10 次迭代。

TaskID	Description	Job	Time
1	提取每段人名	ExtractNames	110s
2	得到人名对及出现次数	Count	20s
3	把人名对的频率转换为权重	Normalize	19s
4	基于人物关系图的 PageRank 计算	LOOP1	18s
		LOOP2	23s
		LOOP3	22s
		LOOP4	22s
		LOOP5	25s
		LOOP6	21s
		LOOP7	20s
		LOOP8	19s
		LOOP9	19s
		LOOP10	18s
	对 PageRank 值计算结果进行排序	Final	20s
5	在人物关系图上的标签传播	LOOP1	20s
		LOOP2	23s
		LOOP3	25s
		LOOP4	22s
		LOOP5	27s
		LOOP6	21s
		LOOP7	19s
		LOOP8	20s
		LOOP9	20s
		LOOP10	21s
	对标签计算结果进行按标签排序（按字典序）	Final	22s
			616s

集群上的运行速度和集群中节点的健康度又很大关系，因此最近跑的任务都相当慢，还不及本机。在很

早之前做单个任务的测试运行的时候，我对服务器和本机的速度也做了一个比较，发现还是本机跑得快。考虑到本机是伪分布式，在数据的传输上基本没有开销，同时该任务的数据量又比较小，反而在数据的传输上，集群会消耗较多的时间。

10 任务分工

王屹：标签传播算法的实现，数据可视化，人名提取
周博聪：PageRank 的实现，归一化

References:

[1] 《深入理解大数据》机械工业出版社，黄宜华，苗凯翔
[2] PIG 标签传播算法和超级英雄们的社区发现 http://www.cnphp6.com/archives/24136