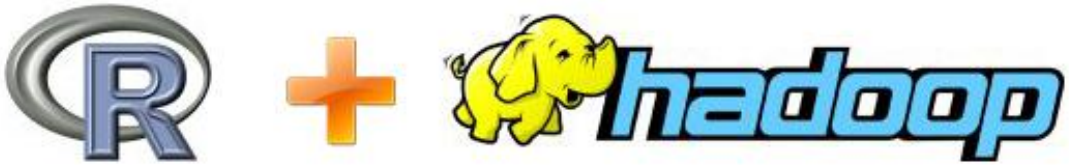
CAPITAL OF STATISTICS
PROFESSION, HUMANITY & INTEGRITY



RHadoop - 张丹

2013.06.22

作者介绍

- **张丹(@Conan_Z) ，** 程序开发者，R语言爱好者。

  在软件和互联网行业从事多年，曾开发多种不同类型的系统及应用。

- 熟练掌握R,JAVA,PHP,Javacript 4 种编程语言。

  对系统架构，编程算法，统计分析，有一定的知识积累。

- 两款有关R语言混编的应用。

  @晒粉丝 http://www.fens.me

  @每日中国天气 http://apps.weibo.com/chinaweatherapp

- **希望找到志同道合的朋友,一起把这份事业做下去!!**

RHadoop安装与使用

- RHadoop安装
  - rhdfs
  - rmr2
- RHadoop程序实例（1，2，3，4）
- rhbase安装与使用
  - rhbase
- 进阶内容：用R现实MapReduce协同过滤算法
  - 基于物品推荐的协同过滤算法介绍
  - R本地程序实现
  - R基于Hadoop分步式程序实现

RHadoop介绍

- RHadoop是RevolutionAnalytics的工程的项目，开源实现代码在GitHub社区可以找到。RHadoop包含三个R包 (rmr，rhdfs，rhbase)，分别是对应Hadoop系统架构中的，MapReduce, HDFS, HBase 三个部分。由于这三个库不能在CRAN中找到，所以需要自己下载。https://github.com/RevolutionAnalytics/RHadoop/wiki

RHadoop安装 – 环境准备

- Linux Ubuntu 12.04 64位
- JDK一定要用Oracle SUN官方的版本，请从官网下载。
  - http://www.oracle.com/technetwork/java/javase/downloads/index.html
  - 操作系统的自带的OpenJDK会有各种不兼容。JDK请选择1.6.x的版本，JDK1.7版本也会有各种的不兼容情况。
- Hadoop版本 1.0.3
- R语言版本2.15，2.14是不能够支持RHadoop的。

Ubuntu环境用apt-get安装R

- Linux Ubuntu 12.04 64位 环境用apt-get安装R，请先更新软件包源，否则只能下载到2.14版本的R。

- 执行命令：

~ sh -c "echo deb http://mirror.bjtu.edu.cn/cran/bin/linux/ubuntu precise/ >>/etc/apt/sources.list"

~ apt-get update

~ apt-get install r-base

请使用下面的安装命令

~ sudo apt-get install r-base-core=2.15.3-1precise0precise1

RHadoop安装 – 依赖库

- 首先是rJava，运行R CMD javareconf命令，R的程序从系统变量中会读取Java配置。然后打开R程序，通过install.packages的方式，安装rJava。

- 然后，我还要安装其他的几个依赖库，reshape2，Rcpp，iterators，itertools，digest，RJSONIO，functional，通过install.packages都可以直接安装。

RHadoop安装 – 依赖库命令

命令行执行

~ R CMD javareconf

~ R

启动R程序

install.packages("rJava")

install.packages("reshape2")

install.packages("Rcpp")

install.packages("iterators")

install.packages("itertools")

install.packages("digest")

install.packages("RJSONIO")

install.packages("functional")

RHadoop安装 –rhdfs,rmr2

- 安装rhdfs库，在环境变量中增加 HADOOP_CMD 和 HADOOP_STREAMING 两个变量，可以用export在当前命令窗口中增加。但为下次方便使用，最好把变量增加到系统环境变更/etc/environment文件中。

HADOOP_CMD=/home/conan/hadoop/hadoop-1.0.3/bin/hadoop

HADOOP_STREAMING=/home/conan/hadoop/hadoop-1.0.3/contrib/streaming/hadoop-streaming-1.0.3.jar

- 用 R CMD INSTALL安装rhdfs包，顺利完成了
- 用 R CMD INSTALL安装rmr2包，顺利完成了

RHadoop安装 –设置环境变量

~ export HADOOP_CMD=/root/hadoop/hadoop-1.0.3/bin/hadoop

~ export HADOOP_STREAMING=/root/hadoop/hadoop-1.0.3/contrib/streaming/hadoop-streaming-1.0.3.jar

- 设置环境变量

~ vi /etc/environment

HADOOP_CMD=/root/hadoop/hadoop-1.0.3/bin/hadoop

HADOOP_STREAMING=/root/hadoop/hadoop-1.0.3/contrib/streaming/hadoop-streaming-1.0.3.jar

~ . /etc/environment

RHadoop安装 –rhdfs,rmr2

- ~ R CMD INSTALL /root/R/rhdfs_1.0.5.tar.gz
- ~ R CMD INSTALL /root/R/rmr2_2.1.0.tar.gz

列出所有安装包

- 由于我的硬盘是外接的，使用mount和软连接(ln -s)挂载了R类库的目录，所以是R的类库在/disk1/system下面
  /disk1/system/usr/local/lib/R/site-library/
  一般R的类库目录是/usr/lib/R/site-library或者/usr/local/lib/R/site-library，用户也可以使用whereis R的命令查询，自己电脑上R类库的安装位置

~ ls /disk1/system/usr/local/lib/R/site-library/

digest functional iterators itertools plyr Rcpp reshape2 rhdfs rJava RJSONIO rmr2 stringr

rhdfs包的使用

- 启动R程序

```
> library(rhdfs)
Loading required package: rJava
HADOOP_CMD=/root/hadoop/hadoop-1.0.3/bin/hadoop
Be sure to run
> hdfs.init()
```

- 注：hdfs.init() 方法会初始化本地的hdfs环境信息

实例1:查看hdfs文件目录

- 查看hdfs文件目录

  hadoop的命令：

  - hadoop fs -ls /user

  R语言函数：

  - hdfs.ls(“ /user/ “)

实例2:查看hadoop数据文件

- 查看hadoop数据文件

  hadoop的命令：

  - hadoop fs -cat /user/hdfs/o_same_school/part-m-00000

  R语言函数：

  - hdfs.cat(" /user/hdfs/o_same_school/part-m-00000")

rmr2包的使用

- 启动R程序

```
> library(rmr2)
Loading required package: Rcpp
Loading required package: RJSONIO
Loading required package: digest
Loading required package: functional
Loading required package: stringr
Loading required package: plyr
Loading required package: reshape2
```

- 注：rhdfs和rmr2包之间没有依赖关系

实例3:rmr实现MapReduce算法

- MapReduce算法

  普通的R语言程序：

  ```
  > small.ints = 1:10
  > sapply(small.ints, function(x) x^2)
  ```

  结果：

  ```
  [1] 1 4 9 16 25 36 49 64 81 100
  ```

实例3:rmr实现MapReduce算法

MapReduce的R语言程序：

```
> small.ints = to.dfs(1:10)
> mapreduce(input = small.ints, map = function(k, v) cbind(v, v^2))
> from.dfs("/tmp/RtmpWnzxl4/file5deb791fcbd5")
```

结果：

```
$key
NULL

$val v
[1,] 1 1
[2,] 2 4
[3,] 3 9
[4,] 4 16
[5,] 5 25
[6,] 6 36
[7,] 7 49
[8,] 8 64
[9,] 9 81
[10,] 10 100
```

注： 因为MapReduce只能访问HDFS文件系统，先要用to.dfs()把数据存储到HDFS文件系统里。MapReduce的运算结果再用from.dfs()函数从HDFS文件系统中取出。

实例4:rmr对文件中的单词计数


```
> input<- '/user/hdfs/o_same_school/part-m-00000'
>  wordcount = function(input, output = NULL, pattern = " "){
            wc.map = function(., lines) {
                        keyval(unlist( strsplit( x = lines,split = pattern)),1)
            }
            wc.reduce =function(word, counts ) {
                        keyval(word, sum(counts))
            }
            mapreduce(input = input ,output = output, input.format = "text",
                                    map = wc.map, reduce = wc.reduce,combine = T)
    }
> wordcount(input)
```

rhbase安装与使用

开始之前请大家把HBase，Thrift配置好

Hbase版本：hbase-0.94.2

Thrift版本： thrift-0.8.0

- rhbase安装
- rhbase程序用例

rhbase安装

- 启动Hadoop, Hbase , Thrift Server

~/hbase-0.94.2/bin/start-hbase.sh

~/hbase-0.94.2/bin/hbase-daemon.sh start thrift

~ jps

12041 HMaster

12209 HRegionServer

13222 ThriftServer

31734 TaskTracker

31343 DataNode

31499 SecondaryNameNode

13328 Jps

31596 JobTracker

11916 HQuorumPeer

31216 NameNode

安装rhbase

~ R CMD INSTALL /root/R/rhbase_1.1.1.tar.gz

rhbase函数列表

- hb.compact.table
- hb.describe.table
- hb.insert
- hb.regions.table
- hb.defaults
- hb.get
- hb.insert.data.frame
- hb.scan
- hb.delete
- hb.delete
- hb.get.data.frame
- hb.list.tables
- hb.scan.ex
- hb.delete.table
- hb.init
- hb.new.table
- hb.set.table.mode

实例1：rhbase建表，列出所有表

- 建表 ：

HBASE:

  - create 'student_shell','info'

RHBASE:

  - hb.new.table("student_rhbase","info")

- 列出所有表：

HBASE:

  - list

RHBASE:

  - hb.list.tables()

实例2：rhbase显示表结构，插入数据

- 显示表结构

HBASE:

  - describe 'student_shell'

RHBASE:

  - hb.describe.table("student_rhbase")

- 插入一条数据

HBASE:

  - put 'student_shell','mary','info:age','19'

RHBASE:

  - hb.insert("student_rhbase",list(list("mary","info:age", "24")))

实例3：rhbase读数据，删除表

- 读取数据

HBASE:

  - get 'student_shell','mary'

RHBASE:

  - hb.get('student_rhbase','mary')

- 删除表(HBASE需要两条命令，rhbase仅是一个操作)

HBASE:

  - disable 'student_shell'
  - drop 'student_shell'

RHBASE:

  - hb.delete.table('student_rhbase')

用R现实MapReduce协同过滤算法

接下是高阶一些的内容

- 基于物品推荐的协同过滤算法介绍
- R本地程序实现
- R基于Hadoop分步式程序实现

基于物品推荐的协同过滤算法介绍

- 越来越多的互联网应用，都开始使用推荐算法(协同过滤算法)。
- 根据用户活跃度和物品流行度，可以分为”基于用户的协同过滤算法”和”基于物品的协同过滤算法”。

- 基于**用户**的协同过滤算法，是给用户推荐和他兴趣相似的其他用户喜欢的物品。
- 基于**物品**的协同过滤算法，是给用户推荐和他之前喜欢的物品相似的物品。基于物品的协同过滤算法，是目前广泛使用的一种推荐算法，像Netflix, YouTube, Amazon等。

基于物品的协同过滤算法

- 算法主要分为两步：

  1. 计算物品之间的相似度
  2. 根据物品的相似度和用户的历史行为给用户生成推荐列表

- 有关算法的细节请参考：” Mahout In Action”和”推荐系统实践”两本书。
- 为开发方便，我们选择一组很小的测试数据集。

测试数据集

1,101,5.0
1,102,3.0
1,103,2.5
2,101,2.0
2,102,2.5
2,103,5.0
2,104,2.0
3,101,2.0
3,104,4.0
3,105,4.5
3,107,5.0
4,101,5.0
4,103,3.0
4,104,4.5
4,106,4.0
5,101,4.0
5,102,3.0
5,103,2.0
5,104,4.0
5,105,3.5
5,106,4.0

R语言程序实现 1

- 首先，通过R语言实现基于物品的协同过滤算法，为和RHadoop实现进行对比。这里我使用”Mahout In Action”书里，第一章第六节介绍的分步式基于物品的协同过滤算法进行实现。Chapter 6: Distributing recommendation computations

- 算法的思想：

  1. 建立物品的同现矩阵
  2. 建立用户对物品的评分矩阵
  3. 矩阵计算推荐结果

R语言程序实现 2

建立物品的同现矩阵

- 按用户分组，找到每个用户所选的物品，单独出现计数，及两两一组计数。
- 例如：用户ID为3的用户，分别给101,104,105,107，这4个物品打分。

  1) (101,101),(104,104),(105,105),(107,107)，单独出现计算各加1。

  2) (101,104),(101,105),(101,107),(104,105),(104,107),(105,107)，两个一组计数各加1。

  3) 把所有用户的计算结果求和，生成一个三角矩阵，再补全三角矩阵，就建立了物品的同现矩阵。

	[101]	[102]	[103]	[104]	[105]	[106]	[107]
[101]	5	3	4	4	2	2	1
[102]	3	3	3	2	1	1	0
[103]	4	3	4	3	1	2	0
[104]	4	2	3	4	2	2	1
[105]	2	1	1	2	2	1	1
[106]	2	1	2	2	1	2	0
[107]	1	0	0	1	1	0	1

R语言程序实现 3

建立用户对物品的评分矩阵

- 按用户分组，找到每个用户所选的物品及评分
- 例如：用户ID为3的用户，分别给(3,101,2.0),(3,104,4.0),(3,105,4.5),(3,107,5.0)这4个物品打分。

  1) 找到物品评分(3,101,2.0),(3,104,4.0),(3,105,4.5),(3,107,5.0)

  2) 建立用户对物品的评分矩阵

```
       U3
[101] 2.0
[102] 0.0
[103] 0.0
[104] 4.0
[105] 4.5
[106] 0.0
[107] 5.0
```

R语言程序实现 4

矩阵计算推荐结果

- 同现矩阵*评分矩阵=推荐结果

	101	102	103	104	105	106	107		U3		R
101	5	3	4	4	2	2	1		2.0		40.0
102	3	3	3	2	1	1	0		0.0		18.5
103	**4**	**3**	**4**	**3**	**1**	**2**	**0**	**X**	**0.0**	**=**	**24.5**
104	4	2	3	4	2	2	1		4.0		40.0
105	2	1	1	2	2	1	1		4.5		26.0
106	2	1	2	2	1	2	0		0.0		16.5
107	1	0	0	1	1	0	1		5.0		15.5

图片摘自Mahout In Action

R语言程序实现 5 – 代码1

```
#引用plyr包
library(plyr)

#读取数据集
train<-read.csv(file="small.csv",header=FALSE)
names(train)<-c("user","item","pref")

#计算用户列表
usersUnique<-function(){
    users<-unique(train$user)
    users[order(users)]
}

#计算商品列表方法
itemsUnique<-function(){
    items<-unique(train$item)
    items[order(items)]
}

# 用户列表
users<-usersUnique()
users
[1] 1 2 3 4 5

# 商品列表
items<-itemsUnique()
items
[1] 101 102 103 104 105 106 107
```

R语言程序实现 5 – 代码2

```
#建立商品列表索引
index<-function(x) which(items %in% x)
data<-ddply(train,.(user,item,pref),summarize,idx=index(item))

#同现矩阵
cooccurrence<-function(data){
    n<-length(items)
    co<-matrix(rep(0,n*n),nrow=n)
    for(u in users){
        idx<-index(data$item[which(data$user==u)])
        m<-merge(idx,idx)
        for(i in 1:nrow(m)){
            co[m$x[i],m$y[i]]=co[m$x[i],m$y[i]]+1
        }
    }
    return(co)
}
```

```
#推荐算法
recommend<-function(udata=udata,co=coMatrix,num=0){
    n<-length(items)
    # all of pref
    pref<-rep(0,n)
    pref[udata$idx]<-udata$pref

    # 用户评分矩阵
    userx<-matrix(pref,nrow=n)

    # 同现矩阵*评分矩阵
    r<-co %*% userx

    # 推荐结果排序
    r[udata$idx]<-0
    idx<-order(r,decreasing=TRUE)
    topn<-data.frame(user=rep(udata$user[1],length(idx)),
item=items[idx],val=r[idx]) topn0),]

    # 推荐结果取前num个
    if(num>0){
        topn<-head(topn,num)
    }

    #返回结果
    return(topn)
}
```

R语言程序实现 5 – 代码3

```
#生成同现矩阵
co<-cooccurrence(data)
> co
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]    5    3    4    4    2    2    1
[2,]    3    3    3    2    1    1    0
[3,]    4    3    4    3    1    2    0
[4,]    4    2    3    4    2    2    1
[5,]    2    1    1    2    2    1    1
[6,]    2    1    2    2    1    2    0
[7,]    1    0    0    1    1    0    1

#计算推荐结果
recommendation<-data.frame()
for(i in 1:length(users)){
  udata<-data[which(data$user==users[i]),]
  recommendation<-rbind(recommendation,recommend(udata,co,0))
}
```

```
> recommendation
   user item  val
1  1  104 33.5
2  1  106 18.0
3  1  105 15.5
4  1  107 5.0
5  2  106 20.5
6  2  105 15.5
7  2  107 4.0
8  3  103 24.5
9  3  102 18.5
10 3  106 16.5
11 4  102 37.0
12 4  105 26.0
13 4  107 9.5
14 5  107 11.5
```

通过上面几步操作，我们用R语言实现了，基于物品的协同过滤算法。

注：这种算法是基于分步式设计的，单机有更多种选择

R基于Hadoop分步式程序实现 1

- 算法思想同上面R语言实现思想，但略有复杂。**(真正难点)**
- R语言实现的MapReduce算法，可以基于R的数据对象实现，不必如JAVA一样使用文本存储。

- 算法的思想：

  1. 建立物品的同现矩阵

  1) 按用户分组，得到所有物品出现的组合列表。

  2) 对物品组合列表进行计数，建立物品的同现矩阵

  2. 建立用户对物品的评分矩阵

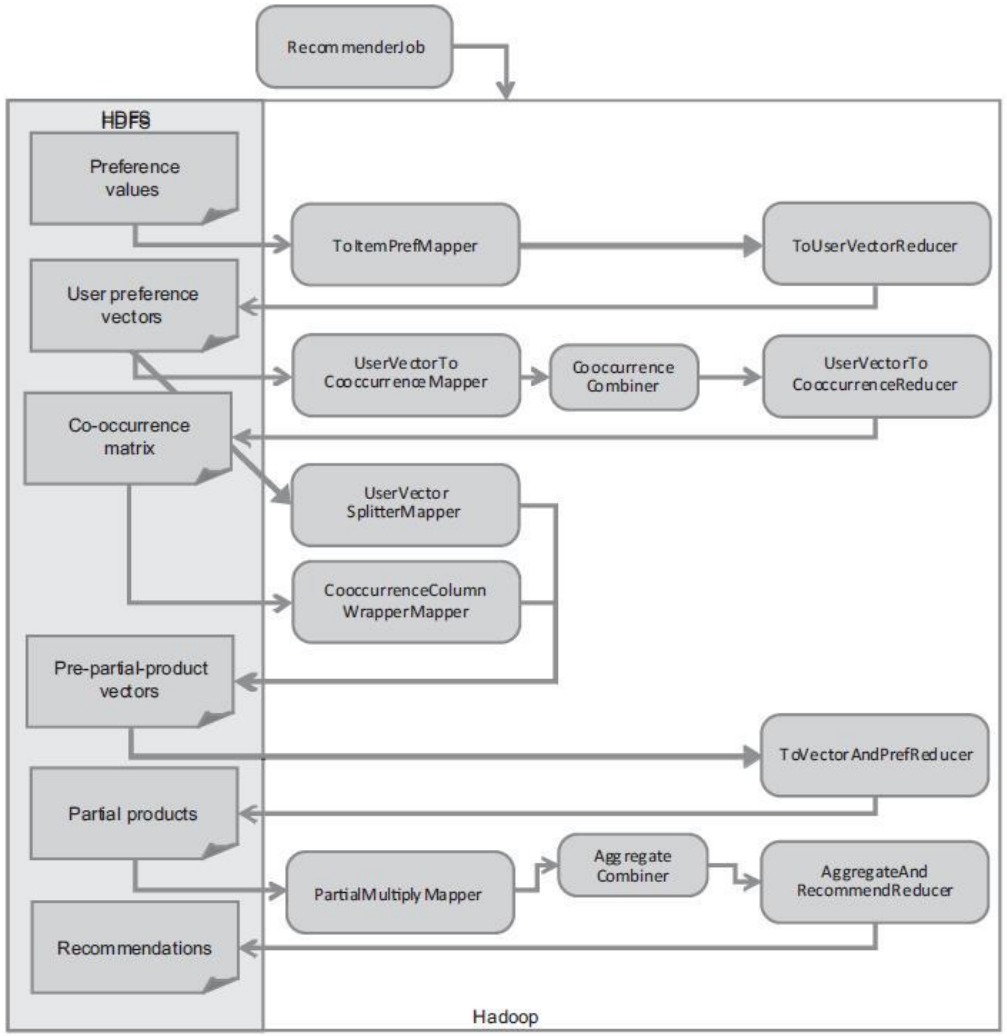
  3. 合并同现矩阵和评分矩阵

  4. 计算推荐结果列表

  5. 按输入格式得到推荐评分列表

R基于Hadoop分步式程序实现 2

- 通过MapReduce实现时，所有操作都要使用Map和Reduce的任务完成，程序实现过程略有变化。



图片摘自Mahout In Action

R基于Hadoop分步式程序实现 3

- 1. 建立物品的同现矩阵
- 1) 按用户分组，得到所有物品出现的组合列表。
- key:物品列表向量

  val:物品组合向量

```
$key
[1] 101 101 101 101 101 101 101 101 101 101 101 101 101 101 101 101 102 102 102 102
[20] 102 102 102 103 103 103 103 103 103 103 103 103 103 103 103 104 104 104 104 104
[39] 104 104 104 104 104 104 104 105 105 105 105 106 106 106 106 107 107 107 107
[58] 101 101 101 101 101 101 102 102 102 102 102 102 103 103 103 103 103 103 104
[77] 104 104 104 104 104 105 105 105 105 105 105 106 106 106 106 106 106

$val
[1] 101 102 103 101 102 103 104 101 104 105 107 101 103 104 106 101 102 103 101
[20] 102 103 104 101 102 103 101 102 103 104 101 103 104 106 101 102 103 104 101
[39] 104 105 107 101 103 104 106 101 104 105 107 101 103 104 106 101 104 105 107
[58] 101 102 103 104 105 106 101 102 103 104 105 106 101 102 103 104 105 106 101
[77] 102 103 104 105 106 101 102 103 104 105 106 101 102 103 104 105 106
```

R基于Hadoop分步式程序实现 4

- 2) 对物品组合列表进行计数，建立物品的同现矩阵
- key:物品列表向量

  val:同现矩阵的数据框值(item,item,Freq)

注：矩阵格式，要与”2. 建立用户对物品的评分矩阵”的格式一致，把异构的两种数据源，合并为同一种数据格式，为”3. 合并 同现矩阵 和 评分矩阵”做数据基础。

```
$key
[1] 101 101 101 101 101 101 101 102 102 102 102 102 102 103 103 103 103 103 103
[20] 104 104 104 104 104 104 104 105 105 105 105 105 105 105 106 106 106 106 106
[39] 106 107 107 107 107

$val
k v freq
1 101 101 5
2 101 102 3
3 101 103 4
4 101 104 4
5 101 105 2
6 101 106 2
7 101 107 1
8 102 101 3
9 102 102 3
10 102 103 3
11 102 104 2
12 102 105 1
13 102 106 1
14 103 101 4
15 103 102 3
16 103 103 4
```

R基于Hadoop分步式程序实现 5

- 2. 建立用户对物品的评分矩阵
- key:物品列表

  val:用户对物品打分矩阵

注：矩阵格式，要与”2) 对物品组合列表进行计数，建立物品的同现矩阵”的格式一致，把异构的两种数据源，合并为同一种数据格式，为”3. 合并 同现矩阵 和 评分矩阵”做数据基础

```
$key
[1] 101 101 101 101 101 102 102 102 103 103 103 103 104 104 104 104 105 105 106
[20] 106 107

$val
item user pref
1 101 1 5.0
2 101 2 2.0
3 101 3 2.0
4 101 4 5.0
5 101 5 4.0
6 102 1 3.0
7 102 2 2.5
8 102 5 3.0
9 103 1 2.5
10 103 2 5.0
11 103 4 3.0
12 103 5 2.0
13 104 2 2.0
14 104 3 4.0
15 104 4 4.5
16 104 5 4.0
17 105 3 4.5
18 105 5 3.5
19 106 4 4.0
20 106 5 4.0
21 107 3 5.0
```

R基于Hadoop分步式程序实现 6

- 3. 合并 同现矩阵 和 评分矩阵
- 这一步操作是MapReduce比较特殊的，因为数据源是两个异构数据源，进行MapReduce的操作。在之前，我们已经把两种格式合并为一样的。使用equijoin这个rmr2包的函数，进行矩阵合并。
- key:NULL
  val:合并的数据框

```
$key
NULL

$val
k.l v.l freq.l item.r user.r pref.r
1 103 101 4 103 1 2.5
2 103 102 3 103 1 2.5
3 103 103 4 103 1 2.5
4 103 104 3 103 1 2.5
5 103 105 1 103 1 2.5
6 103 106 2 103 1 2.5
7 103 101 4 103 2 5.0
8 103 102 3 103 2 5.0
9 103 103 4 103 2 5.0
10 103 104 3 103 2 5.0
11 103 105 1 103 2 5.0
12 103 106 2 103 2 5.0
13 103 101 4 103 4 3.0
....
```

R基于Hadoop分步式程序实现 7

- 4. 计算推荐结果列表
- 把第三步中的矩阵，进行合并计算，得到推荐结果列表
- key:物品列表
  val:推荐结果数据框

```
$key
[1] 101 101 101 101 101 101 101 101 101 101 101 101 101 101 101 101 101 101 101
[19] 101 101 101 101 101 101 101 101 101 101 101 101 101 101 101 101 101 101 102
[37] 102 102 102 102 102 102 102 102 102 102 102 102 102 102 102 102 102 102 103
[55] 103 103 103 103 103 103 103 103 103 103 103 103 103 103 103 103 103 103 103
[73] 103 103 103 103 103 104 104 104 104 104 104 104 104 104 104 104 104 104 104
[91] 104 104 104 104 104 104 104 104 104 104 104 104 104 104 104 104 105 105 105
[109] 105 105 105 105 105 105 105 105 105 105 105 106 106 106 106 106 106 106
[127] 106 106 106 106 106 107 107 107 107

$val
k.1 v.1 user.r v
1 101 101 1 25.0
2 101 101 2 10.0
3 101 101 3 10.0
4 101 101 4 25.0
5 101 101 5 20.0
6 101 102 1 15.0
7 101 102 2 6.0
8 101 102 3 6.0
9 101 102 4 15.0
10 101 102 5 12.0
11 101 103 1 20.0
12 101 103 2 8.0
13 101 103 3 8.0
14 101 103 4 20.0
15 101 103 5 16.0
16 101 104 1 20.0
17 101 104 2 8.0
```

R基于Hadoop分步式程序实现 8

- 5. 按输入格式得到推荐评分列表
- 对推荐结果列表，进行排序处理，输出排序后的推荐结果。
- key:用户ID
  val:推荐结果数据框

我们得到推荐结果！

```
$key
[1] 1 1 1 1 1 1 1 2 2 2 2 2 2 2 3 3 3 3 3 3 3 4 4 4 4 4 4 4 5 5 5 5 5 5 5

$val
user item pref
1 1 101 44.0
2 1 103 39.0
3 1 104 33.5
4 1 102 31.5
5 1 106 18.0
6 1 105 15.5
7 1 107 5.0
8 2 101 45.5
9 2 103 41.5
10 2 104 36.0
11 2 102 32.5
12 2 106 20.5
13 2 105 15.5
14 2 107 4.0
15 3 101 40.0
16 3 104 38.0
17 3 105 26.0
18 3 103 24.5
19 3 102 18.5
20 3 106 16.5
21 3 107 15.5
```

R基于Hadoop分步式程序实现 – 代码

由于代码比较长，我就不贴代码了。请大家在我的blog中查看代码。

http://blog.fens.me/rhadoop-mapreduce-rmr/

rmr2使用提示

- 1) rmr.options(backend = ‘hadoop’)

  这里backend有两个值，hadoop,local。hadoop是默认值，使用hadoop环境运行程序。local是一个本地测试的设置，已经不建议再使用。我在开发时，试过local设置，运行速度非常快，模拟了hadoop的运行环境。但是，local模式下的代码，不能和hadoop模式下完全兼容，变动也比较大，因此不建议大家使用。

- 2) equijoin(...,outer=c(‘left’))

  这里outer包括了4个值，c(“”，“left”，“right”，“full”)，非常像数据库中两个表的join操作

- 3) keyval(k,v)

  mapReduce的操作，需要key和valve保存数据。如果直接输出，或者输出的未加key，会有一个警告Converting to.dfs argument to keyval with a NULL key。再上一篇文章中，rmr2的例子中就有类似的情况，请大家注意修改代码。

PPT内容的补充：

- RHadoop实践系列之一 Hadoop环境搭建
- RHadoop实践系列之二 RHadoop安装与使用
- RHadoop实践系列之三 R实现MapReduce的协同过滤算法
- RHadoop实践系列之四 rhbase安装与使用

联系作者

- 张丹, 编程爱好者(Java,R,PHP,Javascript)
- Weibo: @Conan_Z
- Blog：http://blog.fens.me
- Email: bsspirit@gmail.com

CAPITAL OF STATISTICS
PROFESSION, HUMANITY & INTEGRITY

Thanks

FAQ时间