选题 1.1——在 hadoop 上实现 PageRank 算法

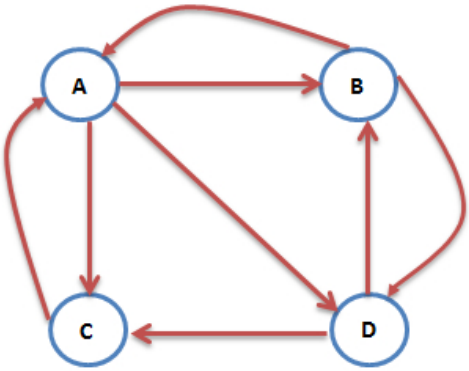
一． PageRank 算法介绍

1. 算法简介

PageRank 是 Google 专有的算法，用于衡量特定网页相对于搜索引擎索引中的其他网页而言的重要程度它由 Larry Page 和 Sergey Brin 在 20 世纪 90 年代后期发明。PageRank 实现了将链接价值概念作为排名因素。PageRank 将对页面的链接看成投票，指示了重要性。简单来说，PageRank 算法计算每一个网页的 PageRank 值，然后根据这个值的大小对网页的重要性进行排序。它的思想是模拟一个悠闲的上网者，上网者首先随机选择一个网页打开，然后在这个网页上呆了几分钟后，跳转到该网页所指向的链接，这样无所事事、漫无目的地在网页上跳来跳去，PageRank 就是估计这个悠闲的上网者分布在各个网页上的概率。

2. 基本模型

将每个网页抽象成有向图中的一个结点，如果网页 A 可以通过链接跳转到网页 B，那么就在 A 和 B 之间建一条有向边 A->B，如下：



在上面这个例子中，A 可以跳转到 B、C、D，那么一个悠闲的上网者就会以 1/3 的概率跳转到这 3 个页面的某一个。同理一个网页有 K 个可跳转网页，那么他会以 1/K 的概率跳转到这 K 个页面中的一个。假设一共有 n 个网页，我们可以将所有网页间的转移概率写成一个 n*n 转移矩阵 M，其中如果网页 j 有 k 个出链，那么对每一个出链指向的网页 i，有 M[i][j]=1/k，而其他网页的 M[i][j]=0。那么上图例子中的转移矩阵如下：

$$\mathrm{M} = \begin{bmatrix} 0 & 1/2 & 1 & 0 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{bmatrix}$$

初始时，假设上网者在每一个网页的概率都是相等的，那么初始的 PageRank 值（为方便，以下简称 PR 值）就是 1/n，将其表示成一个 1*n 的列向量 PR，上面例子的 PR 如下：

$$PR_0 = \begin{bmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{bmatrix}$$

新的 PR 值由下面的矩阵乘法计算得到：

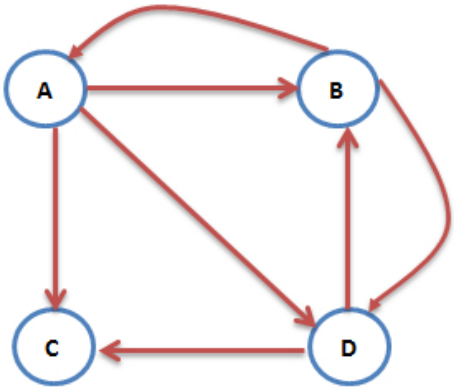
$$PR_{i+1} = M * PR_i$$

重复上式计算知道 PR 收敛为止，例子中最终 PR 收敛成 PR=[3/9, 2/9, 2/9, 2/9]'

3. 终止点问题

上述上网者的行为是一个马尔科夫的过程的实例，要满足收敛性，需要具备一个条件：图是强连通的，即从任意网页可以到达其他任意网页。实际上，互联网的网页不满足强连通性质，因为有一些网页不指向任何网页，这样到最后上网者到达这样的网页之后就没办法继续走，导致前面累计的概率会被清 0。一

个例子如下：



对应的转移矩阵：

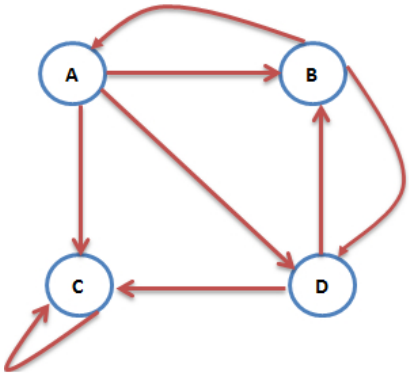
$$M = \begin{bmatrix} 0 & 1/2 & 0 & 0 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{bmatrix}$$

PR 的迭代变化过程，最终会到 0：

$$\begin{bmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{bmatrix} \begin{bmatrix} 3/24 \\ 5/24 \\ 5/24 \\ 5/24 \end{bmatrix} \begin{bmatrix} 5/48 \\ 7/48 \\ 7/48 \\ 7/48 \end{bmatrix} \begin{bmatrix} 21/288 \\ 31/288 \\ 31/288 \\ 31/288 \end{bmatrix} \cdots \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

4. 陷阱问题

还有一个问题是陷阱问题，即有些网页不存在指向其他网页的链接，但存在指向自己的链接，比如下面这个例子：



上网者到达 C 页面后，就陷入了陷阱，再也不能从 C 出来，这会导致最终概率分布值全部转移到 C，使得其他网页的概率分布值为 0.上述问题的转移矩阵为：

$$M = \begin{bmatrix} 0 & 1/2 & 0 & 0 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 0 & 1 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{bmatrix}$$

不断迭代 PR 的变化如下：

$$\begin{bmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{bmatrix} \begin{bmatrix} 3/24 \\ 5/24 \\ 11/24 \\ 5/24 \end{bmatrix} \begin{bmatrix} 5/48 \\ 7/48 \\ 29/48 \\ 7/48 \end{bmatrix} \begin{bmatrix} 21/288 \\ 31/288 \\ 205/288 \\ 31/288 \end{bmatrix} \cdots \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

5. PageRank 优化模型

在实际上网过程中，上网者不止会通过网页中的链接去往其他网而已，他也可能有一定概率直接在地址栏里输入新网址去往其他网页，这就给解决终止点问题和陷阱问题提供了方法。很显然，在地址栏输入某个网址的概率是 1/n。这里再设置一个阻尼系数α，这个系数的作用在于假设上网者查看当前网页的概率为α，那么他通过地址栏输入新网址去其他页面的概率就是（1-α）。所以原来的 PR 值的计算公式就转换成如下式子：

$$PR_{i+1} = \alpha * M * PR_i + \ (1\text{-}\alpha) * P2$$

其中$P2 = \begin{bmatrix} 1/n \\ 1/n \\ \cdots \\ 1/n \\ 1/n \end{bmatrix}$

至此，一个完善的 PageRank 计算模型就做好了，不会再陷入终止点问题和陷阱问题。对于阻尼系数α，Google 公司将其设为 0.85，因此在本次实验中我也将采用这个值。

二． 实验设计

1. 基本思想

1. 将每一次迭代过程作为一次完整的 Map-Reduce 过程，也就是说下一次迭代 Map 过程的输入是上一次迭代的 Reduce 的输出。
2. 设置一个最大迭代次数，本实验将其设为 20。

2. 数据格式处理

本实验我使用了一个稀疏矩阵的形式来表示输入数据：将每一个网页和其能跳转到的网页作为一行，那么原理中的例子就如下：

```
A:B,C,D
B:A,D
C:A
D:B,C
```

第一行 A：B,C,D 就表示从 A 可以跳转去 B,C,D。后面以此类推。然后转移概率可以根据有多少个可跳转网站算出来。这样就解决了转移矩阵的输入问题。而对于 PR 初始向量，这里直接将 n 当成命令行参数输入，然后输入文件中先手动加上初始 PR 概率，如下：

```
A 0.25:B,C,D
B 0.25:A,D
C 0.25:A
D 0.25:B,C
```

第一行 A 0.25：B，C，D 就表示，网页 A 当前 PR 值为 0.25，可跳转页面为

B,C,D，后面每行以此类推。Reduce 也按这个格式输出即可，这样只有 PR 值是变的，其他都不变，不用重新思考如何读入上一次迭代的结果。

三． 实验结果

运行截图（数据集使用上述的简单例子）

```
ter:~# hadoop jar PageRank.jar PageRank 4 /PageRank_input /PageRank_output/output
step is finished
 17:59:53 INFO client.RMProxy: Connecting to ResourceManager at master/192.168.1.212:8032
 17:59:54 WARN mapreduce.JobSubmitter: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with To
 to remedy this.
 17:59:55 INFO input.FileInputFormat: Total input paths to process : 1
 17:59:56 INFO mapreduce.JobSubmitter: number of splits:1
 17:59:56 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1515318970163_0001
 17:59:57 INFO impl.YarnClientImpl: Submitted application application_1515318970163_0001
 17:59:57 INFO mapreduce.Job: The url to track the job: http://master:8088/proxy/application_1515318970163_0001/
 17:59:57 INFO mapreduce.Job: Running job: job_1515318970163_0001
 18:00:10 INFO mapreduce.Job: Job job_1515318970163_0001 running in uber mode : false
 18:00:10 INFO mapreduce.Job:  map 0% reduce 0%
 18:00:22 INFO mapreduce.Job:  map 100% reduce 0%
 18:00:29 INFO mapreduce.Job:  map 100% reduce 100%
 18:00:29 INFO mapreduce.Job: Job job_1515318970163_0001 completed successfully
 18:00:29 INFO mapreduce.Job: Counters: 49
File System Counters
        FILE: Number of bytes read=327
        FILE: Number of bytes written=211495
        FILE: Number of read operations=0
        FILE: Number of large read operations=0
        FILE: Number of write operations=0
        HDFS: Number of bytes read=152
        HDFS: Number of bytes written=104
        HDFS: Number of read operations=6
        HDFS: Number of large read operations=0
        HDFS: Number of write operations=2
Job Counters
        Launched map tasks=1
        Launched reduce tasks=1
        Data-local map tasks=1
        Total time spent by all maps in occupied slots (ms)=8612
        Total time spent by all reduces in occupied slots (ms)=4016
        Total time spent by all map tasks (ms)=8612
        Total time spent by all reduce tasks (ms)=4016
        Total vcore-seconds taken by all map tasks=8612
        Total vcore-seconds taken by all reduce tasks=4016
        Total megabyte-seconds taken by all map tasks=8818688
        Total megabyte-seconds taken by all reduce tasks=4112384
Map-Reduce Framework
        Map input records=4
        Map output records=20
        Map output bytes=281
        Map output materialized bytes=327
        Input split bytes=108
        Combine input records=0
        Combine output records=0
        Reduce input groups=4
        Reduce shuffle bytes=327
        Reduce input records=20
        Reduce output records=4
        Spilled Records=40
```

结果截图

```
root@master:~# hdfs dfs -cat /PageRank_output/output20/*
A       0.32456140075268647:B,C,D
B       0.22514619974910452:A,D
C       0.22514619974910452:A
D       0.22514619974910452:B,C
root@master:~#
```

其中被框住的数据为对应网页的 PR 值。

四． 代码及详细解释见 github 仓库