PageRank算法R语言实现

时间 2014-01-16 16:51:18  [粉丝日志](http://www.tuicool.com/sites/uQbyYn)

原文  [http://blog.fens.me/algorithm-pagerank-r/](http://blog.fens.me/algorithm-pagerank-r/?utm_source=tuicool&utm_medium=referral)

主题 [算法](http://www.tuicool.com/topics/11000083)[R语言](http://www.tuicool.com/topics/11080057)

算法为王系列文章，涵盖了计算机算法，数据挖掘(机器学习)算法，统计算法，金融算法等的多种跨学科算法组合。在大数据时代的背景下，算法已经成为了金字塔顶的明星。一个好的算法可以创造一个伟大帝国，就像Google。

算法为王的时代正式到来….

**关于作者：**

* 张丹(Conan), 程序员Java,R,PHP,Javascript
* weibo：@Conan\_Z
* blog: http://blog.fens.me
* email: bsspirit@gmail.com

**转载请注明出处：**   
[***http://blog.fens.me/algorithm-pagerank-r/***](http://blog.fens.me/algorithm-pagerank-r/)



**前言**

Google搜索，早已成为我每天必用的工具，无数次惊叹它搜索结果的准确性。同时，我也在做Google的SEO，推广自己的博客。经过几个月尝试，我的博客PR到2了，外链也有几万个了。总结下来，还是感叹PageRank的神奇！

改变世界的算法，PageRank！

**目录**

1. PageRank算法介绍
2. PageRank算法原理
3. PageRank算法的R语言实现

**1. PageRank算法介绍**

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让链接来”投票”

一个页面的“得票数”由所有链向它的页面的重要性来决定，到一个页面的超链接相当于对该页投一票。一个页面的PageRank是由所有链向它的页面（“链入页面”）的重要性经过递归算法得到的。一个有较多链入的页面会有较高的等级，相反如果一个页面没有任何链入页面，那么它没有等级。

简单一句话概括：从许多优质的网页链接过来的网页，必定还是优质网页。

PageRank的计算基于以下两个基本假设：

* 数量假设：如果一个页面节点接收到的其他网页指向的入链数量越多，那么这个页面越重要
* 质量假设：指向页面A的入链质量不同，质量高的页面会通过链接向其他页面传递更多的权重。所以越是质量高的页面指向页面A，则页面A越重要。

要提高PageRank有3个要点：

* 反向链接数
* 反向链接是否来自PageRank较高的页面
* 反向链接源页面的链接数

**2. PageRank算法原理**

在初始阶段：网页通过链接关系构建起有向图，每个页面设置相同的PageRank值，通过若干轮的计算，会得到每个页面所获得的最终PageRank值。随着每一轮的计算进行，网页当前的PageRank值会不断得到更新。

在一轮更新页面PageRank得分的计算中，每个页面将其当前的PageRank值平均分配到本页面包含的出链上，这样每个链接即获得了相应的权值。而每个页面将所有指向本页面的入链所传入的权值求和，即可得到新的PageRank得分。当每个页面都获得了更新后的PageRank值，就完成了一轮PageRank计算。

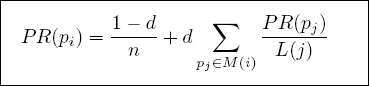
**1). 算法原理**

PageRank算法建立在随机冲浪者模型上，其基本思想是：网页的重要性排序是由网页间的链接关系所决定的，算法是依靠网页间的链接结构来评价每个页面的等级和重要性，一个网页的PR值不仅考虑指向它的链接网页数，还有指向’指向它的网页的其他网页本身的重要性。

PageRank具有两大特性：

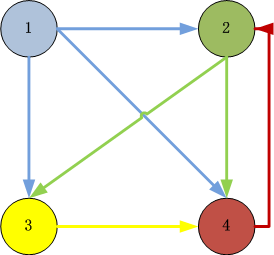
* PR值的传递性：网页A指向网页B时，A的PR值也部分传递给B
* 重要性的传递性：一个重要网页比一个不重要网页传递的权重要多

**2). 计算公式：**



* PR(pi): pi页面的PageRank值
* n: 所有页面的数量
* pi: 不同的网页p1,p2,p3
* M(i): pi链入网页的集合
* L(j): pj链出网页的数量
* d:阻尼系数, 任意时刻，用户到达某页面后并继续向后浏览的概率。   
  (1-d=0.15) :表示用户停止点击，随机跳到新URL的概率   
  取值范围: 0 < d ≤ 1, Google设为0.85

**3). 构造实例：以4个页面的数据为例**



图片说明：

* ID=1的页面链向2,3,4页面，所以一个用户从ID=1的页面跳转到2,3,4的概率各为1/3
* ID=2的页面链向3,4页面，所以一个用户从ID=2的页面跳转到3,4的概率各为1/2
* ID=3的页面链向4页面，所以一个用户从ID=3的页面跳转到4的概率各为1
* ID=4的页面链向2页面，所以一个用户从ID=4的页面跳转到2的概率各为1

构造邻接表：

链接源页面 链接目标页面

1 2,3,4

2 3,4

3 4

4 2

构造邻接矩阵(方阵):

* 列：源页面
* 行：目标页面

[,1] [,2] [,3] [,4]

[1,] 0 0 0 0

[2,] 1 0 0 1

[3,] 1 1 0 0

[4,] 1 1 1 0

转换为概率矩阵(转移矩阵)

[,1] [,2] [,3] [,4]

[1,] 0 0 0 0

[2,] 1/3 0 0 1

[3,] 1/3 1/2 0 0

[4,] 1/3 1/2 1 0

通过链接关系，我们就构造出了“转移矩阵”。

**3. R语言单机算法实现**

创建数据文件：page.csv

1,2

1,3

1,4

2,3

2,4

3,4

4,2

分别用下面3种方式实现PageRank:

* 未考虑阻尼系统的情况
* 包括考虑阻尼系统的情况
* 直接用R的特征值计算函数

**1). 未考虑阻尼系统的情况**

R语言实现

#构建邻接矩阵

adjacencyMatrix<-function(pages){

n<-max(apply(pages,2,max))

A <- matrix(0,n,n)

for(i in 1:nrow(pages)) A[pages[i,]$dist,pages[i,]$src]<-1

A

}

#变换概率矩阵

probabilityMatrix<-function(G){

cs <- colSums(G)

cs[cs==0] <- 1

n <- nrow(G)

A <- matrix(0,nrow(G),ncol(G))

for (i in 1:n) A[i,] <- A[i,] + G[i,]/cs

A

}

#递归计算矩阵特征值

eigenMatrix<-function(G,iter=100){

iter<-10

n<-nrow(G)

x <- rep(1,n)

for (i in 1:iter) x <- G %\*% x

x/sum(x)

}

> pages<-read.table(file="page.csv",header=FALSE,sep=",")

> names(pages)<-c("src","dist");pages

src dist

1 1 2

2 1 3

3 1 4

4 2 3

5 2 4

6 3 4

7 4 2

> A<-adjacencyMatrix(pages);A

[,1] [,2] [,3] [,4]

[1,] 0 0 0 0

[2,] 1 0 0 1

[3,] 1 1 0 0

[4,] 1 1 1 0

> G<-probabilityMatrix(A);G

[,1] [,2] [,3] [,4]

[1,] 0.0000000 0.0 0 0

[2,] 0.3333333 0.0 0 1

[3,] 0.3333333 0.5 0 0

[4,] 0.3333333 0.5 1 0

> q<-eigenMatrix(G,100);q

[,1]

[1,] 0.0000000

[2,] 0.4036458

[3,] 0.1979167

[4,] 0.3984375

结果解读：

* ID=1的页面，PR值是0，因为没有指向ID=1的页面
* ID=2的页面，PR值是0.4，权重最高，因为1和4都指向2，4权重较高，并且4只有一个链接指向到2，权重传递没有损失
* ID=3的页面，PR值是0.19，虽有1和2的指向了3，但是1和2还指向的其他页面，权重被分散了，所以ID=3的页面PR并不高
* ID=4的页面，PR值是0.39，权重很高，因为被1,2,3都指向了

从上面的结果，我们发现ID=1的页面，PR值是0，那么ID=1的页，就不能向其他页面输出权重了，计算就会不合理！所以，增加d阻尼系数，修正没有链接指向的页面，保证页面的最小PR值>0，。

**2). 包括考虑阻尼系统的情况**

增加函数：dProbabilityMatrix

#变换概率矩阵,考虑d的情况

dProbabilityMatrix<-function(G,d=0.85){

cs <- colSums(G)

cs[cs==0] <- 1

n <- nrow(G)

delta <- (1-d)/n

A <- matrix(delta,nrow(G),ncol(G))

for (i in 1:n) A[i,] <- A[i,] + d\*G[i,]/cs

A

}

> pages<-read.table(file="page.csv",header=FALSE,sep=",")

> names(pages)<-c("src","dist");pages

src dist

1 1 2

2 1 3

3 1 4

4 2 3

5 2 4

6 3 4

7 4 2

> A<-adjacencyMatrix(pages);A

[,1] [,2] [,3] [,4]

[1,] 0 0 0 0

[2,] 1 0 0 1

[3,] 1 1 0 0

[4,] 1 1 1 0

> G<-dProbabilityMatrix(A);G

[,1] [,2] [,3] [,4]

[1,] 0.0375000 0.0375 0.0375 0.0375

[2,] 0.3208333 0.0375 0.0375 0.8875

[3,] 0.3208333 0.4625 0.0375 0.0375

[4,] 0.3208333 0.4625 0.8875 0.0375

> q<-eigenMatrix(G,100);q

[,1]

[1,] 0.0375000

[2,] 0.3738930

[3,] 0.2063759

[4,] 0.3822311

增加阻尼系数后，ID=1的页面，就有值了PR(1)=(1-d)/n=(1-0.85)/4=0.0375，即无外链页面的最小值。

**3). 直接用R的特征值计算函数**

增加函数：calcEigenMatrix

#直接计算矩阵特征值

calcEigenMatrix<-function(G){

x <- Re(eigen(G)$vectors[,1])

x/sum(x)

}

> pages<-read.table(file="page.csv",header=FALSE,sep=",")

> names(pages)<-c("src","dist");pages

src dist

1 1 2

2 1 3

3 1 4

4 2 3

5 2 4

6 3 4

7 4 2

> A<-adjacencyMatrix(pages);A

[,1] [,2] [,3] [,4]

[1,] 0 0 0 0

[2,] 1 0 0 1

[3,] 1 1 0 0

[4,] 1 1 1 0

> G<-dProbabilityMatrix(A);G

[,1] [,2] [,3] [,4]

[1,] 0.0375000 0.0375 0.0375 0.0375

[2,] 0.3208333 0.0375 0.0375 0.8875

[3,] 0.3208333 0.4625 0.0375 0.0375

[4,] 0.3208333 0.4625 0.8875 0.0375

> q<-calcEigenMatrix(G);q

[1] 0.0375000 0.3732476 0.2067552 0.3824972

直接计算矩阵特征值，可以有效地减少的循环的操作，提高程序运行效率。

在了解PageRank的原理后，使用R语言构建PageRank模型，是非常容易的。实际应用中，我们也愿意用比较简单的方式建模，验证后，再用其他语言语言去企业应用！

下一篇文章，将介绍如何用MapReduce分步式算法来实现PageRank模型。