Rank

笔记本：	sam408130的笔记本		
创建时间：	2014/12/1 10:09	**更新时间：**	2014/12/5 10:43
URL：	http://slides.com/saiding/deck--2#/1		

一：Pagerank:

- PageRank是Google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

- 对于某个互联网网页A来说，该网页PageRank的计算基于以下两个基本假设

数量假设：在Web图模型中，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。

- 基本思想

如果网页T存在一个指向网页A的连接，则表明T的所有者认为A比较重要，从而把T的一部分重要性得分赋予A。

这个重要性得分值为：

$$PR（T）/L(T)$$

其中PR（T）为T的PageRank值，L(T)为T的出链数

则A的PageRank值为一系列类似于T的页面重要性得分值的累加。

- 简单计算

假设一个由只有4个页面组成的集合：A，B，C和D。如果所有页面都链向A，那么A的PR（PageRank）值将是B，C及D的和:

$$PR(A) = PR(B) + PR(C) + PR(D)$$

继续假设B也有链接到C，并且D也有链接到包括A的3个页面。一个页面不能投票2次。所以B给每个页面半票。

以同样的逻辑，D投出的票只有三分之一算到了A的PageRank上:

$$PR(A)=PR(B)/2 + PR(C)/1 + PR(D)/3$$

换句话说，根据链出总数平分一个页面的PR值。

$$PR(A)=PR(B)/L(B) + PR(C)/L(C) + PR(D)/L(D)$$

- 修正PageRank计算公式

由于存在一些出链为0，也就是那些不链接任何其他网页的网， 也称为孤立网页，使

得很多网页能被访问到。

因此需要对 PageRank公式进行修正，即在简单公式的基础上增加了阻尼系数（damping factor）q， q一般取值q=0.85

其意义是，在任意时刻，用户到达某页面后并继续向后浏览的概率。 1- q= 0.15就是用户停止点击，随机跳到新URL的概率）的算法被用到了所有页面上，估算页面可能被上网者放入书签的概率。

$$\mathrm{PageRank}(p_i) = \frac{1-q}{N} + q\sum_{p_j} \frac{\mathrm{PageRank}(p_j)}{L(p_j)}$$

PageRank值是一个特殊矩阵中的特征向量。这个特征向量为：

$$\mathbf{R} = \begin{bmatrix} \mathrm{PageRank}(p_1) \\ \mathrm{PageRank}(p_2) \\ \vdots \\ \mathrm{PageRank}(p_N) \end{bmatrix}$$

R是如下等式的一个解：

$$\mathbf{R} = \begin{bmatrix} (1-q)/N \\ (1-q)/N \\ \vdots \\ (1-q)/N \end{bmatrix} + q \begin{bmatrix} \ell(p_1,p_1) & \ell(p_1,p_2) & \cdots & \ell(p_1,p_N) \\ \ell(p_2,p_1) & \ddots & & \\ \vdots & & \ell(p_i,p_j) & \\ \ell(p_N,p_1) & & & \ell(p_N,p_N) \end{bmatrix} \mathbf{R}$$

如果网页i有指向网页j的一个链接，则

$$\sum_{i=1}^{N} \ell(p_i,p_j) = 1,$$

否则$\ell(p_i,p_j)=0$。

那我们PageRank 公式可以转换为求解 $\lim_{n\to\infty} A^n X$ 的值，

其中矩阵为 A = q × P + (1 — q) * ee^t /N 。 P 为概率转移矩阵，e^t为 n 维的全 1 行. 则 ee^t=

$$\begin{bmatrix} 1 & \cdots & 1 \\ \vdots & 1 & \vdots \\ 1 & \cdots & 1 \end{bmatrix}$$

- 分解转移矩阵
  迭代公式：X = XP``
  转移矩阵：P``=q ×P` + (1-q) × E

$$E=I^T\cdot V_P, V_P=(1/n\ \ 1/n\ \ \dots\ 1/n)\quad ,\ P`=P+D^T\cdot I$$

其中，$P`$表示对网络中存在的悬挂页面处理后的转移矩阵

$D=(d_1\ \dots\ d_n)$，$d_i=1/n$，页面i是悬挂页面，n为总页面数

$d_i=0$ ，otherwise

新的迭代公式：

$$X^{(K+1)}=X^K\cdot P``$$

$$=q\cdot(X^K\cdot P+X^K\cdot D^T\cdot I)+(1-q)(X^K\cdot I^T)V_P$$

可以将计算过程分解成三部分分别处理

1. $X^K\cdot P$的处理

这部分是处理其他页面链向当前页面，对当前页面产生的贡献

令$B=X^K\cdot P$，$Pji=1/N(j)$，页面$j->i$ （$N(j)$为也变j的出度）

$Pji=0$,otherwise

$$B(i)=X_1^KP_{1i}+X_2^KP_{2i}+\dots+X_n^KP_{ni}$$

$$=\Sigma X_j^KP_{ji}(j=1,n)$$

$$=\Sigma X_j^K/N(j)(j=1,j->i)$$

所以计部分的计算可以转化为提前建立好指向页面A的所有页面的ID,并计算好所有页面的出度。

在本题库模型中每个实体都有一个URI,根据这个ID建立投票数据，和出度数据，以便用于迭代计算

2. $X^K\cdot D^T\cdot I$的处理

令$C=X^K\cdot D^T\cdot I$

其中$X^K\cdot D^T=\Sigma X_i^K\cdot d_i$

$=1/n\cdot\Sigma X_i^K$ （$i=1,n\ \ N(i)=0$）

令$a=\Sigma X_i^K$，那么这一部分就转化为$C=(a/n\ \dots\ a/n)$，其中a表示本轮计算中所有出度为0的页面PR值的和

3. $(1-q)(X^K\cdot I^T)V_P$的处理

令$E=(1-q)(X^K\cdot I^T)V_P$

$X^K\cdot I^T=\Sigma X_i^K$

$=S$ (S为所有网页PR值的和)

而$V_P=$（$1/n\ \dots\ 1/n$），所以$E=(1-q)\cdot S\cdot$（$1/n\ \dots\ 1/n$）

$=(1-q)(s/n\ \dots s/n)$

结果的相对大小，即不影响排序。如果将所有网页PR初始值设为1，那么$s/n=1$,这部分就是（$1-q$）,无需计算

模型建立

以演唱实例作为网络节点，通过演唱实例中的虚拟歌曲，歌手，专辑构建连接关系。同歌手下的演唱实例间互相连接，同专辑下的演唱实例互相建立连接，歌单中的歌曲互相建立连接。

二：HITS:

HITS算法将网页分为两类，即hubs和authorities,而且每个页面也有两个级别，即hubs（中心级别）和authorities（权威级别）。Authorities是具有较高价值的网页，依赖于指向它的页面；hubs为指向较多authorities的网页，依赖于它指向的页面。HITS算法的目标就是通过迭代计算得到针对某个检索提问的排名最高的authority的网页。

算法的基本思想：相互增强关系

基本假设1：一个好的"Authority"页面会被很多好的"Hub"页面指向；

基本假设2：一个好的"Hub"页面会指向很多好的"Authority"页面；

根集合：

将查询q提交给基于关键字查询的检索系统，从返回结果页面的集合取前n个页面，作为根集合（root），root满足：

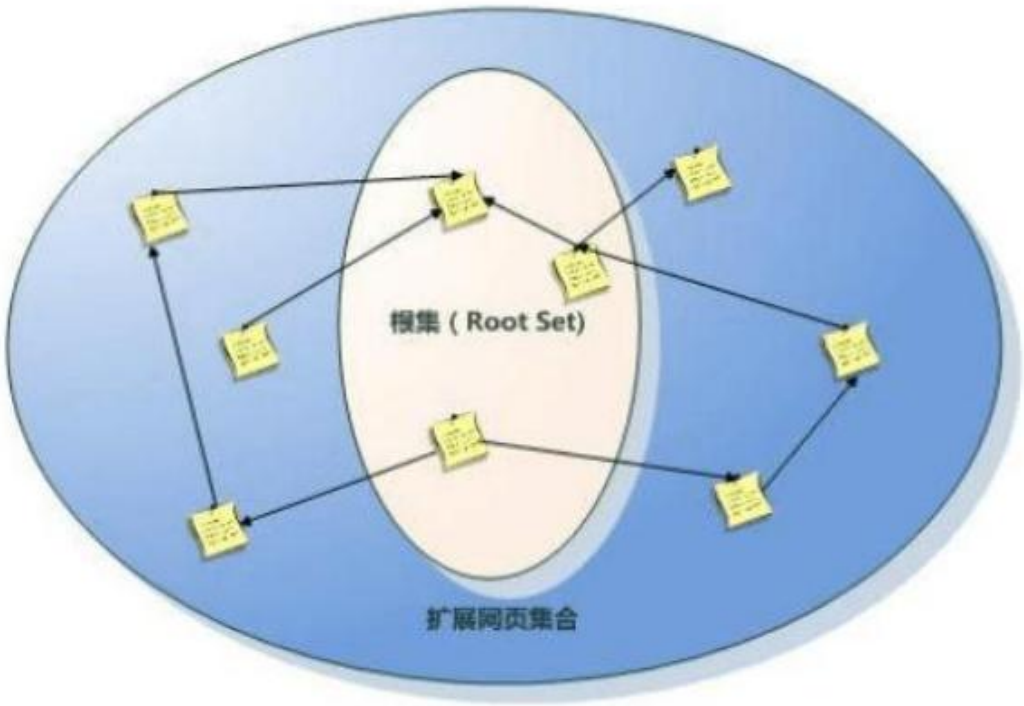
1.root中页面数较少

2.Root中的网页时与查询q相关的网页

3.Root中的网页包含较多权威（Authority）网页

扩展集合base：

在根集合root基础上，HITS算法对网页集合进行扩充（参考图1），扩充原则是：凡是与根集内有直接连接指向关系的网页都被扩充到集合base，无论是有链接指向根集内页面也好，或者是根集页面有链接指向的页面也好，都被扩充进入扩展网页集合base。HITS算法在这个扩充网页集合内寻找好的"Hub"页面与好的"Authority"页面。



算法描述：

网页a(i)在此轮迭代中的Authority权值即为所有指向网页a(i)页面的Hub权值之和：

$$a(i) = \Sigma h(i)$$

网页a(i)的Hub分值即为它指向的页面的Authority权值之和：

$$h(i) = \Sigma a(i)$$

对 a (i)、h (i)进行规范化处理：

将所有网页的中心度都除以最高中心度以将其标准化：

$a(i) = a(i)/|a(i)|$ ；

将所有网页的权威度都除以最高权威度以将其标准化：

$h(i) = h(i)/|h(i)|$ ：

a,h初始化为 1， $a_0 = 1, h_0 = 1$

t=1

do

for each v in V

do $a_t(v) = \sum_{(w,v)\in E} h_{t-1}(w)$

$h_t(v) = \sum_{(v,w)\in E} a_{t-1}(w)$

$a_t = a_t / \|a_t\|$

$h_t = h_t / \|h_t\|$

$t = t + 1$

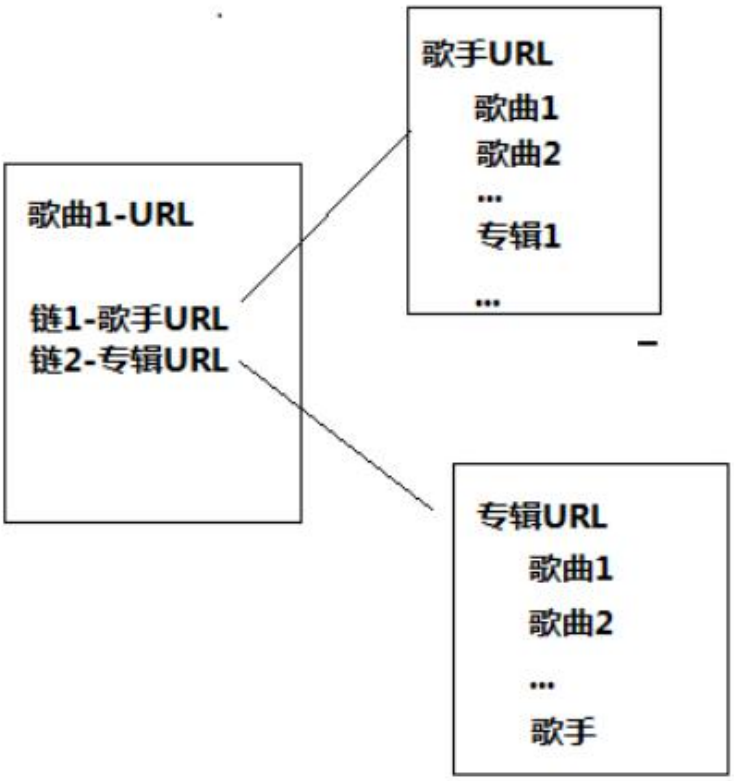
While $\|a_t - a_{t-1}\| + \|h_t - h_{t-1}\| < \varepsilon$

Return (a_t, h_t)

模型建立

在音乐本体库中，每个实体的结构可以看做一个网页：

例如上图中，名为"歌曲1"的网页有两个出链，歌手、专辑。
在模型中，输入内容为歌曲名，root集合为与搜索歌曲同名的所有歌曲网页（这里是为了防止HITS算法中主题漂移问题），base集合为root集合扩展出来歌手网页，专辑网页：



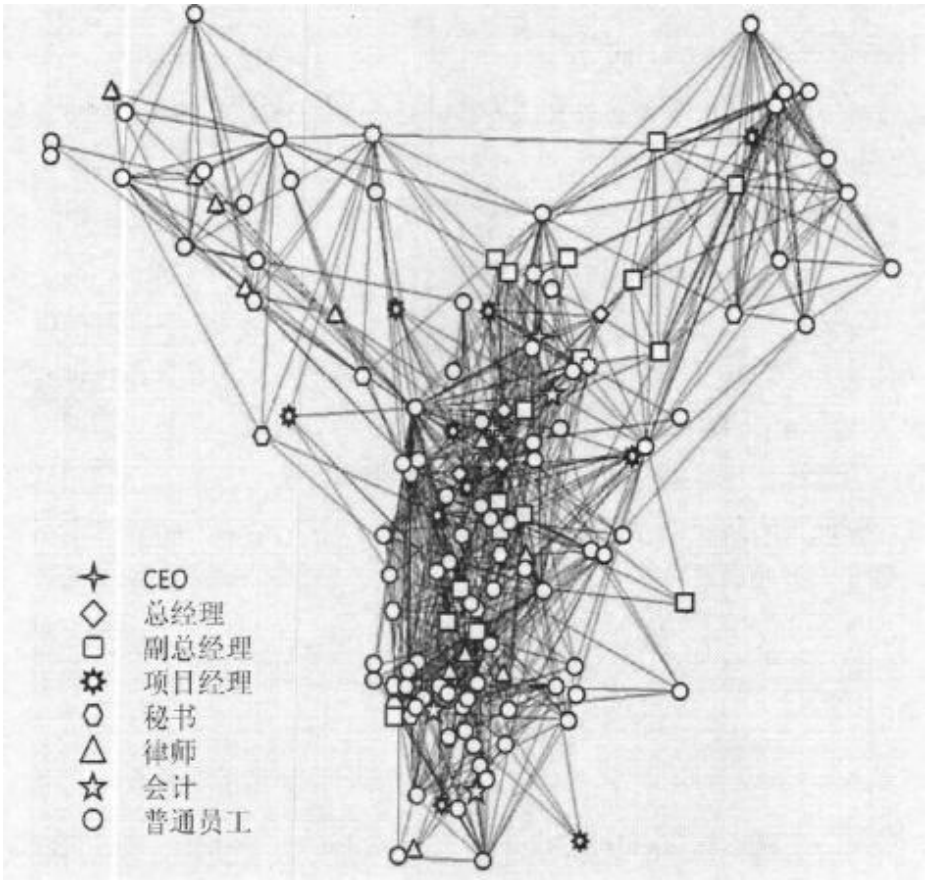
因为我们的搜索queey为一首歌曲，目的是为了找到最合适唱这首歌的歌手，因此最终结果将根据结合的方式给出：歌曲的Authority与其对应歌手的Authority之和

三：SimRank:

SimRank算法是由美国斯坦福大学的Glen Jeh和Jennifer Widom二人提出的，主要利用图的全局信息来计算任意两个节点间的相似度。SimRank的思想是：两个节点的邻居节点相似，那么这两个节点也相似。跟基于共现的相似度算法比较，它采取的是一种比较"间接"的

方式，即A和B相似，C和D相似，那么，如果A和C相似，则B和D也相似。

SimRank算法可用于社会网络中的用户推荐等的实现，比如在某一个社交网中，由用户之间的好友关系构成用户关系图，经过一些社区划分算法之后，会出现多个子社区。在子社区计算中，可以用Simrank算法计算用户两两之间的形似度，按照相似度从高到低，为用户推荐好友。



上图为某相关公司的员工关系网络图，不同类型的节点表示不同类型的员工。员工之间可能存在一种联系，假设员工A和员工B都与项目经理C相连，那么员工A和B可能同属一个部门，同时被C领导，他们之间存在一种相似关系。假设项目经理C和D之间存在相似关系（可能是同公司或者同行业等），那么，分别与C和D相连的员工E和F之间也有形似关系。

Simrank算法通常应用于图算法中，假设图G是一个无向图，a和b表示图上任意两个节点，I(a)表示节点a相连的节点集合，同理，I(b)表示与节点b相连的节点集合，则节点a和节点b的相似度SimRank公式表示如下：

$$s(a,b)=\frac{C}{|I(a)||I(b)|}\sum_{i=1}^{|I(a)|}\sum_{j=1}^{|I(b)|}s(I_i(a),I_j(b))$$

其中，C为衰减系数，取值为0到1，一般去0.8。|I(a)|和|I(b)|分别表示与节点a和b相连的节点的集合大小，为正整数。根据公式可知，节点a和节点b的相似度等于分别与a和b相连的节点之间的两两相似度的平均值。

Simrank算法是一种递归迭代的过程，初始时，节点与自身的相似度最高，一般设为1，与其他节点的相似度为0，如：

$$R_0(a,b)=\begin{cases}0 & (\text{if} \quad a=b)\\ 1 & (\text{if} \quad a\neq b)\end{cases}$$

假设在第k迭代是，节点a,b之间的相似度用Rk(a,b)表示，那么，第k+1次迭代是，a和

b只见的相似度为：

$$R_{k+1}(a,b)=\frac{C}{|I(a)||I(b)|}\sum_{i=1}^{|I(a)|}\sum_{j=1}^{|I(b)|}R_k(I_i(a),I_j(b))$$

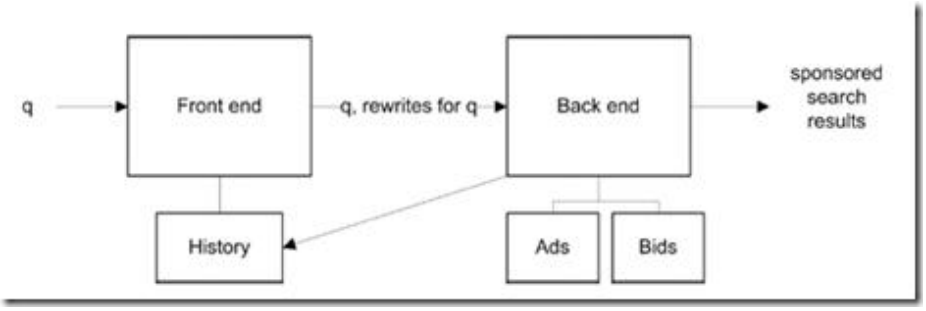
当迭代次数足够多(趋向无穷)时，R(a,b)就是节点a和节点b的相似度。

SimRank算法复杂度较高，用d表示节点a,b的入度积的平均值，n为阶段的数目，k表示迭代次数，则该算法时间复杂度为：

$O(kn^2d_2)$，空间复杂度为$O(n^2)$，在具体应用中，可根据实践情况做一些剪枝操作来减少运算的时间复杂度和空间复杂度。

利用Simrank算法进行Query Rewriting

搜索引擎的检索结果页下方一般会提示多个相似的搜索关键词，这些词可以被看作查询关键词query的rewriting。在计算广告中，当某一个query没有对应的bid phase出价广告，或者该query对应的bid phase较少的时候，可以利用query rewriting获取相似query对应的广告进行显示，以期望获得更多的click。相似query的确定可以利用用户session中的搜索关键词上下文，但此方法需要确定query的边界，例如用户搜索过程中可能会突然跳到一个完全不相干的搜索，然后又跳回来。或者利用传统的文本相似度匹配，而由于query一般都很短，传统相似度匹配的语料也不足。



Simrank

Simrank算法是一种用于衡量结构上下文中个体相似度的方法，直观上的含义是利用已有个体的相似度来推算其他与有关联个体的相似度。形式化的定义如下：

有向图G={V,E}中，节点a的入边集合记为I(a)，而出边集合记为O(a)，则两个节点a,b(a != b时)相似度s(a,b)按照如下公式计算，其含义是个体a,b的相似度取决于所有与a,b相连节点的相似度，s(a,b)∈[0,1]；当a=b时，s(a,b)=1；如果a != b，且只有同一个入节点c，我们不希望从c计算得到的s(a,b)也为1，因此c做为衰减因子，取值是[0,1]，即s(a,b) = C。

$$s(a,b)=\frac{C}{|I(a)||I(b)|}\sum_{i=1}^{|I(a)|}\sum_{j=1}^{|I(b)|}s(I_i(a),I_j(b))$$

对于二分图G' = (V1,V2,E)，对于任意的(A,a)∈E，有A∈V1和a∈V2，则所有V1集合中的节点相似度按照出度O(A)计算。V2集合中的节点相似度则按照上述公式，利用入度I(a)

计算。

$$s(A,B) = \frac{C_1}{|O(A)||O(B)|} \sum_{i=1}^{|O(A)|} \sum_{j=1}^{|O(B)|} s(O_i(A), O_j(B))$$

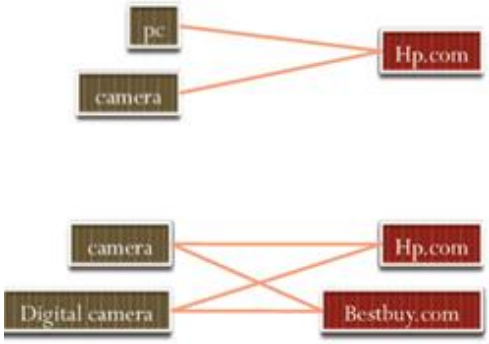
在搜索广告中，把query和ad看作节点，当用户搜索某个查询关键词q时，点击了广告a，则建立q至a的一条边，这样构成一个由query和ad组成的二分图G=(V,E)，其中V=Vq∪Va，任何边e=(q,a,w)，q∈Vq并且a∈Va，可以利用simrank算法来求query之间的相似度。E(q)表示q的边，N(q)表示E(q)的个数：

- $sim_k(q,q')$: q-q' similarity at k-th iteration
- Initially sim(q,q) = 1, sim(q,q') = 0, sim(a,a) = 1, sim(a,a') = 0

$$sim_k(q,q') = \frac{C}{N(q)N(q')} \sum_{i \in E(q)} \sum_{j \in E(q')} sim_{k-1}(i,j)$$

$$sim_k(a,a') = \frac{C}{N(a)N(a')} \sum_{i \in E(a)} \sum_{j \in E(a')} sim_{k-1}(i,j)$$

按照上述算法，我们看以下的例子，假设C等于0.8，则利用上述公式计算出的sim(pc,camera)的相似度在前5次迭代中都大于sim(camera,digital camera)，但从直观上说，由于camera和digital camera的共同邻居较多，应该具备更高的相似度，而simrank的结果是反直观的。针对上述问题，Antonellis等人在VLDB 08上提出了Simrank在计算广告方面的改进——Simrank++。



Iteration	Camera – digital camera	Pc - camera
1	0.4	0.8
2	0.62	0.8
3	0.65	0.8
4	0.66	0.8
5	0.664	0.8
6	0.665	0.8

C = 0.8

Simrank++

Simrank++的改进主要包括两点，一是针对上面的问题对sim(q,q`)进行调整，如果两个节点共同节点**{E(q)∩E(q`)}**越多，则给予这两个节点相似度更高的权值。

$$evidence(q,q') = \sum_{i=1}^{|E(q)\cap E(q')|} \frac{1}{2^i}$$

$$sim^k_{evidence}(q,q') = evidence(q,q') \cdot sim_k(q,q')$$

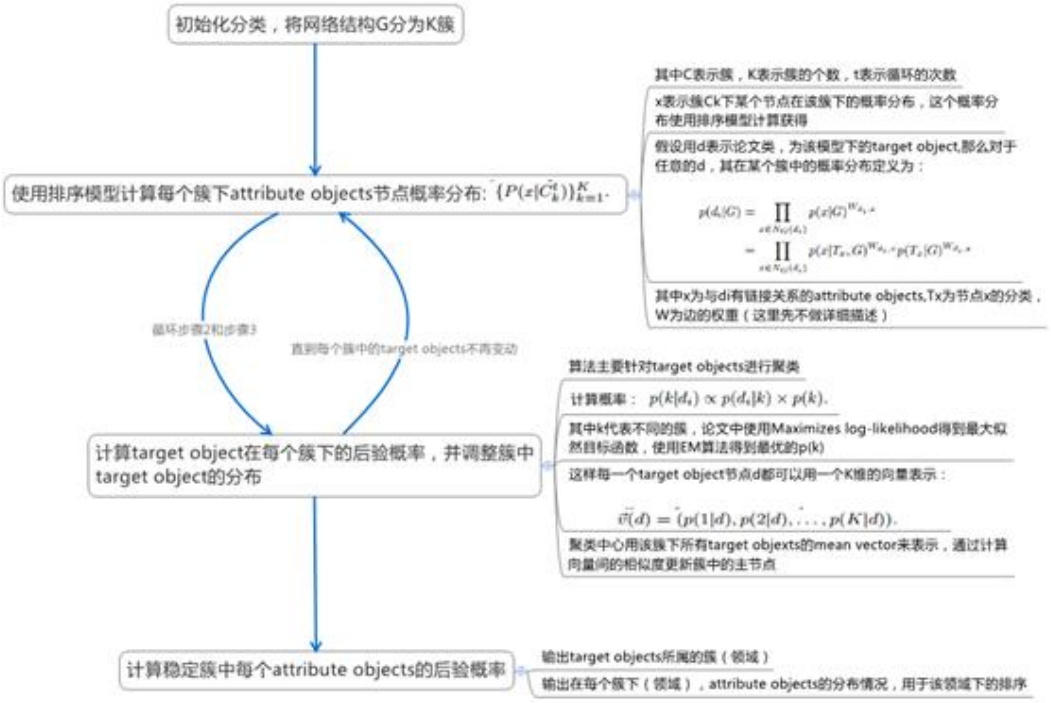
按照上述调整后，上面示例的计算结果如下。同时Antonellis等人也证明了，如果迭代的次数足够多，对于未调整前的simrank，最终sim(pc,camera)会和sim(camera, digital camera)相等，但实际使用中肯定无法计算那么多次迭代。

iteration	Camera – digital camera	Pc - camera
1	0.3	0.4
2	0.42	0.4
3	0.468	0.4
4	0.4872	0.4
5	0.49488	0.4
6	0.497952	0.4

四：NetClus:

异构网络中基于排序模型的聚类算法

针对异构网络一种新的基于排序模型的聚类算法。这里的异构网络主要指星型结构的网络模型，其主要特点是：网络中包含多种类别的节点，其中一种为主节点（target object），以及其他属性节点（attribute object），主节点和属性节点都有链接关系。以论文领域举例，论文，作者，关键词，发表论文机构组成一个星型网络模型，其中论文为target object,作者，关键词，发表论文的机构等都为attribute object。该算法利用不同节点之间的链接关系构建了两种排序模型（Simple Ranking,Authority Ranking），并使用排序模型得到的节点概率分布构建高质量的用于描述节点在不同簇下分布的向量，从而完成聚类过程。聚类稳定后每一个簇都相当于一个领域，例如数据库领域，管理领域等。利用排序模型计算稳定聚类下每个簇中节点的概率分布，便可以得到该领域下，各属性节点的排序状况。
算法的主要步骤如下：



（流程图在这里展示不开，请打开http://www.xmind.net/embed/5TKK/）