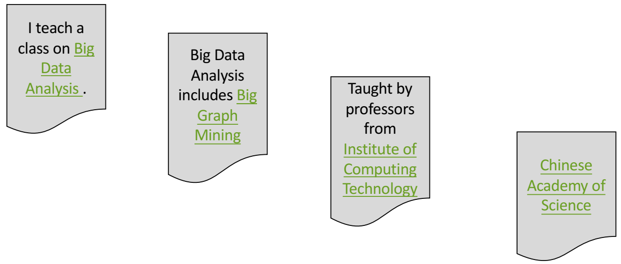
大数据分析

PageRank

刘盛华

Web as a Graph

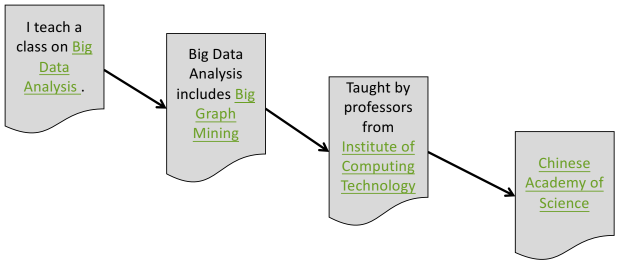
- Web as a directed graph: 有向图
  结点：网页
  边：超链接
  - Nodes: Webpages
  - Edges: Hyperlinks



2

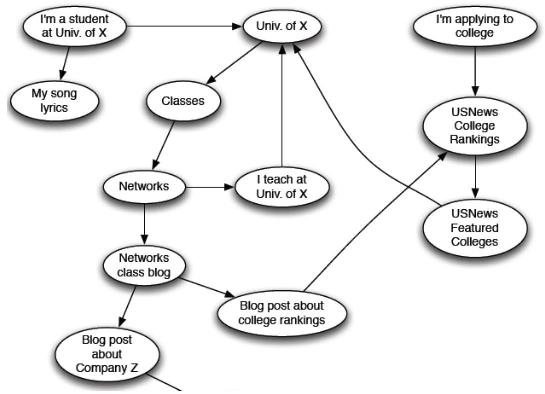
Web as a Graph

- Web as a directed graph:
  - Nodes: Webpages
  - Edges: Hyperlinks



3

Web as a Directed Graph



4

Broad Question

- How to organize the Web?
- First try: Human curated Web directories 人工策划web目录
  - Yahoo, DMOZ, LookSmart
- Second try: Web Search
  - Information Retrieval investigates: Find relevant docs in a small and trusted set 信息检索调查:在一个小而可信的集合中查找相关文档
    - Newspaper articles, Patents, etc.
  - But: Web is huge, full of untrusted documents, random things, web spam, etc. 网络是巨大的，充满了不可信的文件，随机的东西，网络垃圾邮件，等等

5

Web Search: 2 Challenges

2 challenges of web search:

- (1) Web contains many sources of information Who to "trust"? 网络上有很多信息源，谁来"信任"?问窍门:值得信任的页面可能会相互指向对方
  - Trick: Trustworthy pages may point to each other!
- (2) What is the "best" answer to query "newspaper"? 问"报纸"的"最佳"答案是什么?没有一个正确答案
  - No single right answer 诀窍:真正了解报纸的页面可能都指向许多报纸
  - Trick: Pages that actually know about newspapers might all be pointing to many newspapers

6

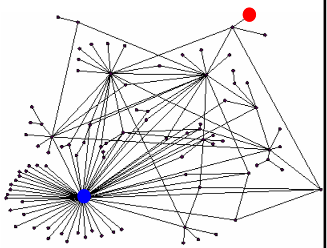
Ranking Nodes on the Graph

- All web pages are not equally "important"

  www.joe-schmoe.com vs. www.stanford.edu

- There is large diversity in the web-graph node connectivity. Let's rank the pages by the link structure!

网络图节点的连接具有很大的多样性。让我们按照链接结构对页面进行排序!



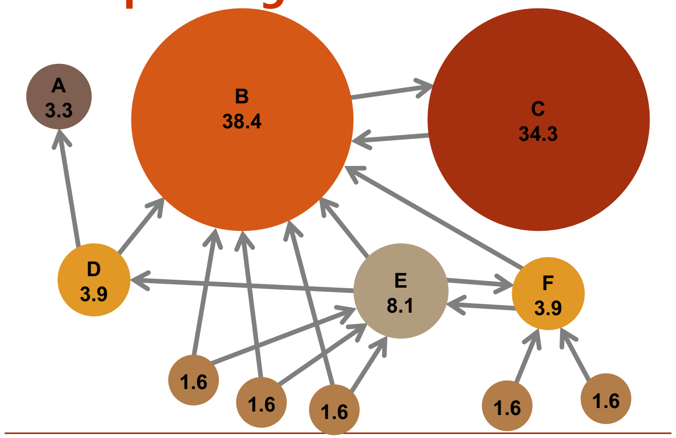
7

PageRank: The "Flow" Formulation

Links as Votes

- Idea: Links as votes 连接当作投票
  - Page is more important if it has more links
    - In-coming links? Out-going links?
- Think of in-links as votes:
  - www.stanford.edu has 23,400 in-links
  - www.joe-schmoe.com has 1 in-link
- Are all in-links are equal?
  - Links from important pages count more 重要页面的链接值更大
  - Recursive question! 递归问题

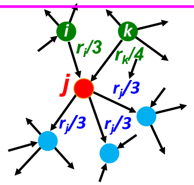
Example: PageRank Scores



Simple Recursive Formulation

- Each link's vote is proportional to the importance of its source page
- If page j with importance r_j has n out-links, each link gets r_j / n votes
- Page j's own importance is the sum of the votes on its in-links

$r_j = r_i/3 + r_k/4$



PageRank: The "Flow" Model

来自重要网页的投票更值钱，

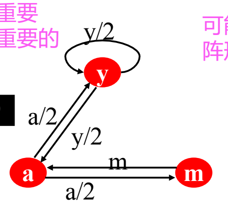
- A "vote" from an important page is worth more
- A page is important if it is pointed to by other important pages 如果一个网页被其他重要网页链接，那他也是重要的
- Define a "rank" r_j for page j

可能考矩阵形式

$$r_j = \sum_{i \to j} \frac{r_i}{d_i}$$

d_i … out-degree of node i

The web in 1839



"Flow" equations:

$r_y = r_y/2 + r_a/2$

$r_a = r_y/2 + r_m$

$r_m = r_a/2$

Solving the Flow Equations

- **3 equations, 3 unknowns, no constants** 3方程，3未知数，无常数 无唯一解
  - No unique solution
  - All solutions equivalent modulo the scale factor 所有解都等于尺度因子取模
- **Additional constraint forces uniqueness:** 附加约束强制得到唯一性
  - $r_y + r_a + r_m = 1$
  - Solution: $r_y = \frac{2}{5}$, $r_a = \frac{2}{5}$, $r_m = \frac{1}{5}$
- **Gaussian elimination method works for small examples, but we need a better method for large web-size graphs**
- **We need a new formulation!**

高斯消去法适用于小例子，但是我们需要一个更好的方法来处理大型网络图。
我们需要一个新的公式

Flow equations:
$r_y = r_y/2 + r_a/2$
$r_a = r_y/2 + r_m$
$r_m = r_a/2$

13

PageRank: Matrix Formulation 矩阵公式

- 随机邻接矩阵
  Stochastic adjacency matrix M
  - Let page i has d_i out-links 外链这里列指向行
  - If $i \to j$, then $M_{ji} = \frac{1}{d_i}$ else $M_{ji} = 0$
    - M is a column stochastic matrix 列随机矩阵
      - Columns sum to 1 列和为1
- **Rank vector r:** vector with an entry per page 每页一个元素
  - r_i is the importance score of page i
  - $\boxed{\sum_i r_i = 1}$ 这里一定考
- **The flow equations can be written**

$$r_j = \sum_{i \to j} \frac{r_i}{d_i}$$

$$r = M \cdot r$$

新的链接等于M乘旧的链接

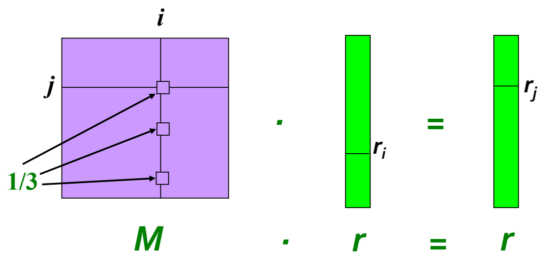
14

Example

- **Remember the flow equation:** $r_j = \sum_{i \to j} \frac{r_i}{d_i}$
- **Flow equation in the matrix form**

$$M \cdot r = r$$

  - Suppose page i links to 3 pages, including j



15

Eigenvector Formulation

- **The flow equations can be written** 特征值是1的特征向量

$$r = M \cdot r$$

- So the $\boxed{\text{rank vector } r}$ is an eigenvector of the stochastic web matrix M
  - In fact, its first or principal eigenvector, with corresponding eigenvalue *1* 相对于特征值1的主特征向量
    - Largest eigenvalue of M is 1 since M is column stochastic (with non-negative entries)
      - *We know r is unit length and each column of M sums to one, so* $\boxed{Mr \le 1}$

M最大的特征值就是1，因为M时列随机的（非负元素）

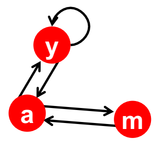
NOTE: x is an eigenvector with the corresponding eigenvalue λ if: $Ax = \lambda x$

- **We can now efficiently solve for r!** The method is called Power iteration

我们现在可以有效地解出r!
这种方法称为幂次迭代

16

Example: Flow Equations & M



	y	a	m
y	½	½	0
a	½	0	1
m	0	½	0

$r = M \cdot r$

$$r_y = r_y/2 + r_a/2$$
$$r_a = r_y/2 + r_m$$
$$r_m = r_a/2$$

$$\begin{bmatrix} y \\ a \\ m \end{bmatrix} = \begin{bmatrix} ½ & ½ & 0 \\ ½ & 0 & 1 \\ 0 & ½ & 0 \end{bmatrix} \begin{bmatrix} y \\ a \\ m \end{bmatrix}$$

Power Iteration Method

- **Given a web graph with *n* nodes, where the nodes are pages and edges are hyperlinks**
- **Power iteration: a simple iterative scheme**
  - **Suppose there are *N* web pages**
  - **Initialize: $r^{(0)} = [1/N,\ldots,1/N]^T$**
  - **Iterate: $r^{(t+1)} = M \cdot r^{(t)}$**
  - **Stop when $|r^{(t+1)} - r^{(t)}|_1 < \varepsilon$**

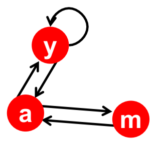
$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{d_i}$$

d_i …. out-degree of node i

$|\mathbf{x}|_1 = \sum_{1 \le i \le N} |x_i|$ is the L_1 norm
Can use any other vector norm, e.g., Euclidean

PageRank: How to solve?

- **Power Iteration:**
  - **Set** $r_j = 1/N$
  - **1:** $r'_j = \sum_{i \to j} \frac{r_i}{d_i}$
  - **2:** $r = r'$
  - **Goto 1**
- **Example:**



	y	a	m
y	½	½	0
a	½	0	1
m	0	½	0

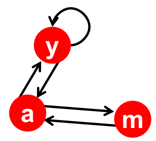
$$r_y = r_y/2 + r_a/2$$
$$r_a = r_y/2 + r_m$$
$$r_m = r_a/2$$

$$\begin{pmatrix} r_y \\ r_a \\ r_m \end{pmatrix} = \begin{array}{c} 1/3 \\ 1/3 \\ 1/3 \end{array}$$

Iteration 0, 1, 2, …

PageRank: How to solve?

- **Power Iteration:**
  - **Set** $r_j = 1/N$
  - **1:** $r'_j = \sum_{i \to j} \frac{r_i}{d_i}$
  - **2:** $r = r'$
  - **Goto 1**
- **Example:**



	y	a	m
y	½	½	0
a	½	0	1
m	0	½	0

$$r_y = r_y/2 + r_a/2$$
$$r_a = r_y/2 + r_m$$
$$r_m = r_a/2$$

$$\begin{pmatrix} r_y \\ r_a \\ r_m \end{pmatrix} = \begin{array}{cccccc} 1/3 & 1/3 & 5/12 & 9/24 & & 6/15 \\ 1/3 & 3/6 & 1/3 & 11/24 & \ldots & 6/15 \\ 1/3 & 1/6 & 3/12 & 1/6 & & 3/15 \end{array}$$

Iteration 0, 1, 2, …

Why Power Iteration works? (1) Details!

- **Power iteration:** 求主特征向量(对应最大特征值的向量)的方法
  A method for finding dominant eigenvector (the vector corresponding to the largest eigenvalue)
  - $r^{(1)} = M \cdot r^{(0)}$
  - 0
  - $r^{(2)} = M \cdot r^{(1)} = M(Mr^{(1)}) = M^2 \cdot r^{(0)}$
  - $r^{(3)} = M \cdot r^{(2)} = M(M^2 r^{(0)}) = M^3 \cdot r^{(0)}$
- **Claim:**
  Sequence $M \cdot r^{(0)}, M^2 \cdot r^{(0)}, \dots M^k \cdot r^{(0)}, \dots$
  approaches the dominant eigenvector of M

Why Power Iteration works? (2) Details!

- Claim: Sequence $M \cdot r^{(0)}, M^2 \cdot r^{(0)}, \dots M^k \cdot r^{(0)}, \dots$ approaches the dominant eigenvector of M
- Proof: 接近M的主导特征向量
  - Assume M has n linearly independent eigenvectors, $x_1, x_2, \dots, x_n$ with corresponding eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_n$, where $\lambda_1 > \lambda_2 > \dots > \lambda_n$
  - Vectors $x_1, x_2, \dots, x_n$ form a basis and thus we can write: $r^{(0)} = c_1 x_1 + c_2 x_2 + \dots + c_n x_n$
  - $Mr^{(0)} = M(c_1 x_1 + c_2 x_2 + \dots + c_n x_n)$
  - $= c_1(Mx_1) + c_2(Mx_2) + \dots + c_n(Mx_n)$
  - $= c_1(\lambda_1 x_1) + c_2(\lambda_2 x_2) + \dots + c_n(\lambda_n x_n)$
  - Repeated multiplication on both sides produces
  - $M^k r^{(0)} = c_1(\lambda_1^k x_1) + c_2(\lambda_2^k x_2) + \dots + c_n(\lambda_n^k x_n)$

Why Power Iteration works? (3) Details!

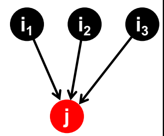
- **Claim:** Sequence $M \cdot r^{(0)}, M^2 \cdot r^{(0)}, \dots M^k \cdot r^{(0)}, \dots$ approaches the dominant eigenvector of M
- **Proof (continued):**
  - Repeated multiplication on both sides produces
    $M^k r^{(0)} = c_1(\lambda_1^k x_1) + c_2(\lambda_2^k x_2) + \dots + c_n(\lambda_n^k x_n)$
  - $M^k r^{(0)} = \lambda_1^k \left[c_1 x_1 + c_2 \left(\frac{\lambda_2}{\lambda_1}\right)^k x_2 + \dots + c_n \left(\frac{\lambda_2}{\lambda_1}\right)^k x_n \right]$
  - Since $\lambda_1 > \lambda_2$ then fractions $\frac{\lambda_2}{\lambda_1}, \frac{\lambda_3}{\lambda_1} \dots < 1$
    and so $\left(\frac{\lambda_i}{\lambda_1}\right)^k = 0$ as $k \to \infty$ (for all $i = 2 \dots n$).
  - **Thus:** $M^k r^{(0)} \approx c_1(\lambda_1^k x_1)$
    - Note if $c_1 = 0$ then the method won't converge

初始的r0，第一个向量不能为0？？？？

Random Walk Interpretation 随机游走解释

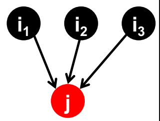
想象一个随机的网络冲浪者

- **Imagine a random web surfer:**
  - At any time t, surfer is on some page i
  - At time $t + 1$, the surfer follows an out-link from i uniformly at random
  - Ends up on some page j linked from i
  - Process repeats indefinitely 从i开始在某页j结束，这个过程无限重复
- **Let:**
  - $p(t)$ … vector whose i^{th} coordinate is the prob. that the surfer is at page i at time t
  - So, $p(t)$ is a probability distribution over pages
    P是页面的概率分布



$$r_j = \sum_{i \to j} \frac{r_i}{d_{out}(i)}$$

The Stationary Distribution 什么时候达到稳态



- Where is the surfer at time $t+1$?
  - Follows a link uniformly at random

$$p(t+1) = M \cdot p(t)$$

$$p(t+1) = \mathrm{M} \cdot p(t)$$

- Suppose the random walk reaches a state $p(t+1) = M \cdot p(t) = p(t)$ then $p(t)$ is stationary distribution of a random walk
- Our original rank vector r satisfies $r = M \cdot r$
  - So, r is a stationary distribution for the random walk

25

Existence and Uniqueness 存在性和唯一性

- A central result from the theory of random walks (a.k.a. Markov processes): 随机游动理论(又称马尔可夫过程)的一个中心结果是:

For graphs that satisfy **certain conditions**, the **stationary distribution is unique** and eventually will be reached no matter what the initial probability distribution at time **t = 0**

对于满足一定条件的图，其平稳分布是唯一的，且无论t = 0时的初始概率分布如何，最终都会达到平稳分布

不管初始是什么值，最终都会达到稳定解

ref to:
Perron–Frobenius theorem [Nonnegative Matrix, irreducible (connected), primitivity (k-connected)]

A自乘k次，那么所有的点都是连接起来了，这样第一大特征值全是正的

26

PageRank: The Google Formulation

PageRank: Three Questions

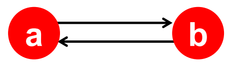
$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{\mathrm{d_i}} \quad \text{or equivalently} \quad r = Mr$$

- Does this converge?
- Does it converge to what we want?
- Are results reasonable?

这是否收敛?
是否收敛于想要的结果?
结果合理吗?

28

Does this converge?



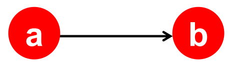
$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{\mathrm{d_i}}$$

- **Example:**

r_a	=	1	0	1	0
r_b		0	1	0	1

Iteration 0, 1, 2, …

Does it converge to what we want?



$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{\mathrm{d_i}}$$

- **Example:**

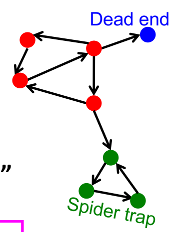
r_a	=	1	0	0	0
r_b		0	1	0	0

Iteration 0, 1, 2, …

PageRank: Problems

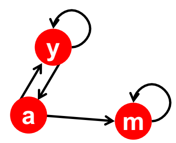
2 problems:

- **(1) Some pages are dead ends (have no out-links)**
  - Random walk has "nowhere" to go to
  - Such pages cause importance to "leak out"
    重要性消失
- **(2) Spider traps: (all out-links are within the group)**
  - Random walked gets "stuck" in a trap
  - And eventually spider traps absorb all importance
    group里的点重要性都很大



Problem: Spider Traps

- **Power Iteration:**
  - **Set** $r_j = 1$
  - $r_j = \sum_{i \to j} \frac{r_i}{d_i}$
    - **And iterate**



m is a spider trap

	y	a	m
y	½	½	0
a	½	0	0
m	0	½	1

$r_y = r_y/2 + r_a/2$
$r_a = r_y/2$
$r_m = r_a/2 + r_m$

- **Example:**

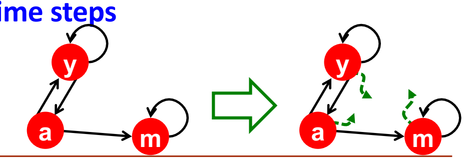
r_y		1/3	2/6	3/12	5/24		0
r_a	=	1/3	1/6	2/12	3/24	…	0
r_m		1/3	3/6	7/12	16/24		1

Iteration 0, 1, 2, …

All the PageRank score gets "trapped" in node m.

Solution: Teleports! 瞬移

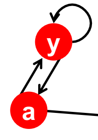
- The Google solution for spider traps: At each time step, the random surfer has two options
  - With prob. β, follow a link at random
  - With prob. 1-β, jump to some random page
  - Common values for β are in the range 0.8 to 0.9
- Surfer will teleport out of spider trap within a few time steps



33

Problem: Dead Ends

- Power Iteration:
  - Set $r_j = 1$
  - $r_j = \sum_{i\to j} \frac{r_i}{d_i}$
    - And iterate



	y	a	m
y	½	½	0
a	½	0	0
m	0	½	0

$r_y = r_y/2 + r_a/2$
$r_a = r_y/2$
$r_m = r_a/2$

- Example:

$$\begin{pmatrix} r_y \\ r_a \\ r_m \end{pmatrix} = \begin{matrix} 1/3 & 2/6 & 3/12 & 5/24 & & 0 \\ 1/3 & 1/6 & 2/12 & 3/24 & \ldots & 0 \\ 1/3 & 1/6 & 1/12 & 2/24 & & 0 \end{matrix}$$

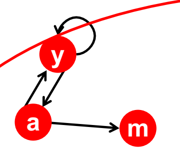
Iteration 0, 1, 2, …

Here the PageRank "leaks" out since the matrix is not stochastic.

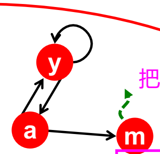
34

Solution: Always Teleport!

- Teleports: Follow random teleport links with probability 1.0 from dead-ends
  - Adjust matrix accordingly



	y	a	m
y	½	½	0
a	½	0	0
m	0	½	0



把一列全为0的换成均值1/d

	y	a	m
y	½	½	⅓
a	½	0	⅓
m	0	½	⅓

35

Why Teleports Solve the Problem?

Why are dead-ends and spider traps a problem and why do teleports solve the problem?

蜘蛛陷阱，不是问题，
只不过不是我们想要的结果

- Spider-traps are not a problem (converge), but with traps PageRank scores are not what we want
  - Solution: Never get stuck in a spider trap by teleporting out of it in a finite number of steps
- Dead-ends are a problem 不收敛
  - The matrix is not column stochastic (zero column) so our initial assumptions are not met
  - Solution: Make matrix column stochastic by always teleporting when there is nowhere else to go

36

Solution: Random Teleports

- Google's solution that does it all:
  At each step, random surfer has two options:
  - With probability β, follow a link at random
  - With probability 1-β, jump to some random page
- PageRank equation [Brin-Page, 98]

$$r_j = \sum_{i \rightarrow j} \beta \frac{r_i}{d_i} + (1-\beta)\frac{1}{N}$$

d_i … out-degree of node i

This formulation assumes that M has no dead ends. We can either preprocess matrix M to remove all dead ends (add 1/N in M) or explicitly follow random teleport links with probability 1.0 from dead-ends (β=0).

这个公式假定M没有dead ends。我们可以预处理矩阵M删除所有dead ends(加1/N)或显式地遵循概率1.0随机瞬移(β=0)。

37

The Google Matrix

- PageRank equation [Brin-Page, '98]

$$r_j = \sum_{i \rightarrow j} \beta \frac{r_i}{d_i} + (1-\beta)\frac{1}{N}$$

- The Google Matrix A:

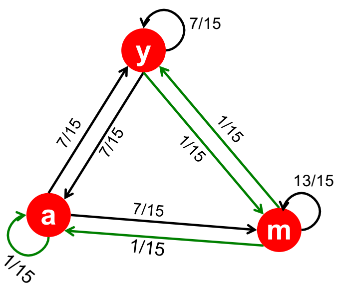
$$A = \beta\, M + (1-\beta)\left[\frac{1}{N}\right]_{N \times N}$$

$[1/N]_{N\times N}$…N by N matrix where all entries are 1/N

- We have a recursive problem: $r = A \cdot r$
  And the Power method still works!
- What is β ?
  - In practice $\boxed{\beta = 0.8, 0.9}$ (make 5 steps on avg., jump)

38

Random Teleports ($\beta = 0.8$)



$$0.8 \overset{\textbf{M}}{\begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 1 \end{bmatrix}} + 0.2 \overset{[\mathbf{1/N}]_{N\times N}}{\begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix}}$$

$$\begin{array}{c} y \\ a \\ m \end{array} \underset{\textbf{A}}{\begin{bmatrix} 7/15 & 7/15 & 1/15 \\ 7/15 & 1/15 & 1/15 \\ 1/15 & 7/15 & 13/15 \end{bmatrix}}$$

$$\begin{matrix} y \\ a \\ m \end{matrix} = \begin{matrix} 1/3 \\ 1/3 \\ 1/3 \end{matrix} \quad \begin{matrix} 0.33 \\ 0.20 \\ 0.46 \end{matrix} \quad \begin{matrix} 0.24 \\ 0.20 \\ 0.52 \end{matrix} \quad \begin{matrix} 0.26 \\ 0.18 \\ 0.56 \end{matrix} \quad \cdots \quad \begin{matrix} 7/33 \\ 5/33 \\ 21/33 \end{matrix}$$

39

How do we actually compute the PageRank?

Computing Page Rank

- Key step is matrix-vector multiplication
  - $r^{new} = A \cdot r^{old}$
- Easy if we have enough main memory to hold A, r^{old}, r^{new}
- Say N = 1 billion pages
  - We need 4 bytes for each entry (say)
  - 2 billion entries for vectors, approx 8GB
  - Matrix A has N^2 entries
    - 10^{18} is a large number!

$A = \beta \cdot M + (1-\beta)\,[1/N]_{N\times N}$

$$A = 0.8 \begin{bmatrix} ½ & ½ & 0 \\ ½ & 0 & 0 \\ 0 & ½ & 1 \end{bmatrix} + 0.2 \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix}$$

$$= \begin{bmatrix} 7/15 & 7/15 & 1/15 \\ 7/15 & 1/15 & 1/15 \\ 1/15 & 7/15 & 13/15 \end{bmatrix}$$

41

Matrix Formulation

- Suppose there are N pages
- Consider page i, with d_i out-links
- We have $M_{ji} = 1/|d_i|$ when $i \to j$
  and $M_{ji} = 0$ otherwise
- The random teleport is equivalent to:
  - Adding a teleport link from i to every other page and setting transition probability to $(1-\beta)/N$
  - Reducing the probability of following each out-link from $1/|d_i|$ to $\beta/|d_i|$
  - Equivalent: Tax each page a fraction $(1-\beta)$ of its score and redistribute evenly

假设有N个页面，
考虑第i页，di向外链接，
当i→j时，我们有Mji = 1/|di|，
否则Mji = 0，

随机传送等于:
添加一个从i到其他页面的传送链接，并设置转换概率为(1-β)/N
降低每个外链的概率，从1/|di|到β/|di|
相当于:对每一页征收得分除以(1-β)的税，然后重新平均分配

42

Rearranging the Equation

- $r = A \cdot r$, where $A_{ji} = \beta M_{ji} + \frac{1-\beta}{N}$
- $r_j = \sum_{i=1}^{N} A_{ji} \cdot r_i$
- $r_j = \sum_{i=1}^{N} \left[\beta M_{ji} + \frac{1-\beta}{N}\right] \cdot r_i$

$$= \sum_{i=1}^{N} \beta M_{ji} \cdot r_i + \frac{1-\beta}{N} \sum_{i=1}^{N} r_i$$

$$= \sum_{i=1}^{N} \beta M_{ji} \cdot r_i + \frac{1-\beta}{N} \quad \text{since } \sum r_i = 1$$

- So we get: $r = \beta M \cdot r + \left[\frac{1-\beta}{N}\right]_N$

Note: Here we assumed M has no dead-ends

$[x]_N$ … a vector of length N with all entries x

43

Sparse Matrix Formulation

- We just rearranged the PageRank equation

$$r = \beta M \cdot r + \left[\frac{1-\beta}{N}\right]_N$$

  - where $[(1-\beta)/N]_N$ is a vector with all N entries $(1-\beta)/N$
- M is a sparse matrix! (with no dead-ends) M是无dead ends的稀疏矩阵
  - 10 links per node, approx 10N entries
- So in each iteration, we need to:
  - Compute $r^{new} = \beta M \cdot r^{old}$
  - Add a constant value $(1-\beta)/N$ to each entry in r^{new}
    - Note if M contains dead-ends then $\sum_j r_j^{new} < 1$ and we also have to renormalize r^{new} so that it sums to 1

44

PageRank: The Complete Algorithm

- Input: Graph G and parameter β
  - Directed graph G (can have spider traps and dead ends)
  - Parameter β
- Output: PageRank vector r^{new}
  - Set: $r_j^{old} = \frac{1}{N}$
  - repeat until convergence: $\sum_j |r_j^{new} - r_j^{old}| > \varepsilon$
    - $\forall j$: $\boldsymbol{r'}_j^{new} = \sum_{i \to j} \beta \frac{r_i^{old}}{d_i}$
      $\boldsymbol{r'}_j^{new} = \mathbf{0}$ if in-degree of j is 0
    - Now re-insert the leaked PageRank:
      $\forall j$: $\boldsymbol{r}_j^{new} = \boldsymbol{r'}_j^{new} + \frac{1-S}{N}$ where: $S = \sum_j r'^{new}_j$
    - $\boldsymbol{r}^{old} = \boldsymbol{r}^{new}$

If the graph has no dead-ends then the amount of leaked PageRank is **1-β**. But since we have dead-ends the amount of leaked PageRank may be larger. We have to explicitly account for it by computing **S**.

如果图没有死角则1-β泄露的PageRank的数量。但由于我们有死胡同，泄露的PageRank的数量可能会更大。我们必须通过计算S来明确地解释它

Sparse Matrix Encoding

- Encode sparse matrix using only nonzero entries
  - Space proportional roughly to number of links
  - Say 10N, or 4*10*1 billion = 40GB
  - Still won't fit in memory, but will fit on disk

source node	degree	destination nodes 目的节点
0	3	1, 5, 7
1	5	17, 64, 113, 117, 245
2	2	13, 23

Basic Algorithm: Update Step

- Assume enough RAM to fit rnew into memory
  - Store r^{old} and matrix M on disk
- 1 step of power-iteration is:

Initialize all entries of $r^{new} = (1-\beta) / N$
For each page i (of out-degree d_i):
  Read into memory: i, d_i, $dest_1$, …, $dest_{d_i}$, $r^{old}(i)$
  For $j = 1…d_i$
    $r^{new}(dest_j)$ += $\beta\, r^{old}(i) / d_i$

r^{new}: 0 1 2 3 4 5 6

source	degree	destination
0	3	1, 5, 6
1	4	17, 64, 113, 117
2	2	13, 23

r^{old}: 0 1 2 3 4 5 6

Analysis

- Assume enough RAM to fit r^{new} into memory
  - Store r^{old} and matrix M on disk
- In each iteration, we have to:
  - Read r^{old} and M
  - Write r^{new} back to disk
  - Cost per iteration of Power method:
    $= 2|r| + |M|$
- Question:
  - What if we could not even fit r^{new} in memory?

如果内存装不下r_new怎么办？分块

Block-based Update Algorithm

- Break r^{new} into k blocks that fit in memory
- Scan M and r^{old} once for each block

r^{new}: 0, 1 | 2, 3 | 4, 5

src	degree	destination
0	4	0, 1, 3, 5
1	2	0, 5
2	2	3, 4

M

r^{old}: 0, 1, 2, 3, 4, 5

Analysis of Block Update

- **Similar to nested-loop join in databases** 类似于数据库中的嵌套循环连接
  - Break r^{new} into k blocks that fit in memory
  - Scan M and r^{old} once for each block
- **Total cost:**
  - k scans of M and r^{old}
  - Cost per iteration of Power method: $k(|M| + |r|) + |r| = k|M| + (k+1)|r|$
- **Can we do better?**
  - Hint: M is much bigger than r (approx 10-20x), so we must avoid reading it k times per iteration

M很大，
应该避免读M次数为k

Block-Stripe Update Algorithm

先根据R_new 把M分好，会重复存很多，但是空间复杂度小了

r^{new}: 0, 1

src	degree	destination
0	4	0, 1
1	3	0
2	2	1

r^{new}: 2, 3

0	4	3
2	2	3

r^{new}: 4, 5

0	4	5
1	3	5
2	2	4

r^{old}: 0, 1, 2, 3, 4, 5

Break *M* into stripes! Each stripe contains only destination nodes in the corresponding block of r^{new}

Block-Stripe Analysis

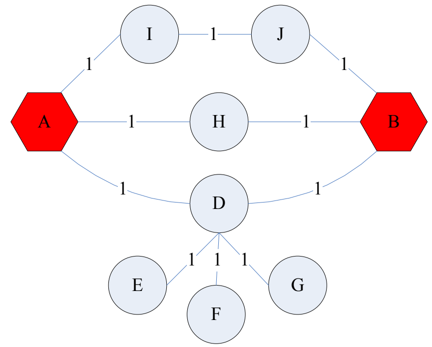
- **Break *M* into stripes** M分成条!每个条带只包含rnew相应块中的目标节点
  - Each stripe contains only destination nodes in the corresponding block of r^{new}
- **Some additional overhead per stripe** 每个条纹q有一些额外的开销
  - But it is usually worth it 但这通常是值得的
- **Cost per iteration of Power method:** $=|M|(1+\varepsilon) + (k+1)|r|$

Application to Measuring Proximity in Graphs

应用于测量接近度的图表
带重启的随机游走:S是单个元素

Random Walk with Restarts: S is a single element

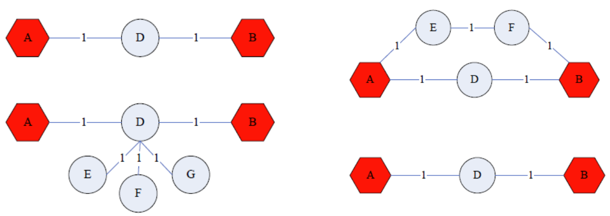
Proximity on Graphs



a.k.a.: Relevance, Closeness, 'Similarity'…

Good proximity measure?

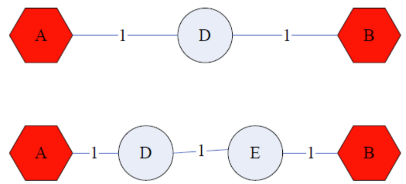
- **Shortest path is not good:**



- **No effect of degree-1 nodes (E, F, G)!**
- **Multi-faceted relationships**

Good proximity measure?

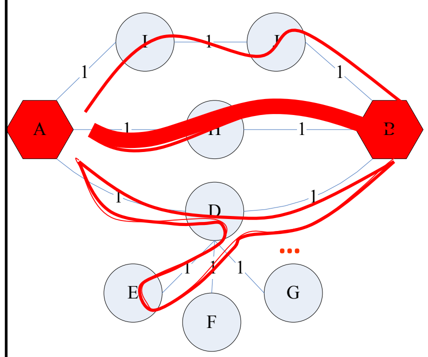
- **Network flow is not good:**



- **Does not punish long paths** 不惩罚长路吗

什么是好的相似的定义？

What is good notion of proximity?

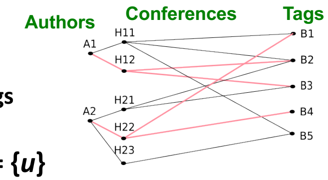


- Multiple connections
- Quality of connection
  - Direct & Indirect connections
  - Length, Degree, Weight…

连接质量
直接连接和间接连接
长度、度、重量...
...

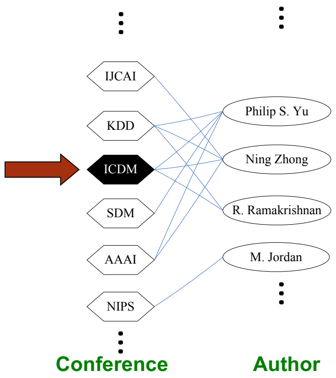
SimRank: Idea

- **SimRank:** Random walks from a fixed node on *k*-partite graphs
- **Setting:** *k*-partite graph with *k* types of nodes
  - E.g.: Authors, Conferences, Tags
- **PageRank from node *u*:** teleport set *S* = {*u*}
- **Resulting scores measures similarity to node *u***
  结果得分衡量与u的相似度
- **Problem:**
  - Must be done once for each node *u* 每个结点u都得做一次
  - Suitable for sub-Web-scale applications 不适应大的图



SimRank: Example

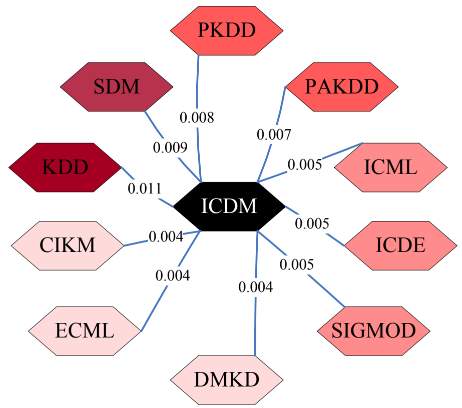
问:与ICDM最相关的会议是什么?
A:具有传送集S={ICDM}的特定主题PageRank



Q: What is most related conference to **ICDM**?

A: Topic-Specific PageRank with teleport set S={ICDM}

SimRank: Example



PageRank: Summary

- "Normal" PageRank:标准页面排序
  - Teleports uniformly at random to any node均匀随机瞬移到任意结点
  - All nodes have the same probability of surfer landing there: S = [0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1]
- Topic-Specific PageRank also known as Personalized PageRank: 特定主题的页面排序也称为个性化页面排序
  - Teleports to a topic specific set of pages 跳到特定主题页面的集合
  - Nodes can have different probabilities of surfer landing there: S = [0.1, 0, 0, 0.2, 0, 0, 0.5, 0, 0, 0.2]
- Random Walk with Restarts:带重启的随机游走
  - Topic-Specific PageRank where teleport is always to the same node. S=[0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]

61

Questions?