Analysis of Large Graphs: Link Analysis, PageRank

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University

New Topic: Graph Data!

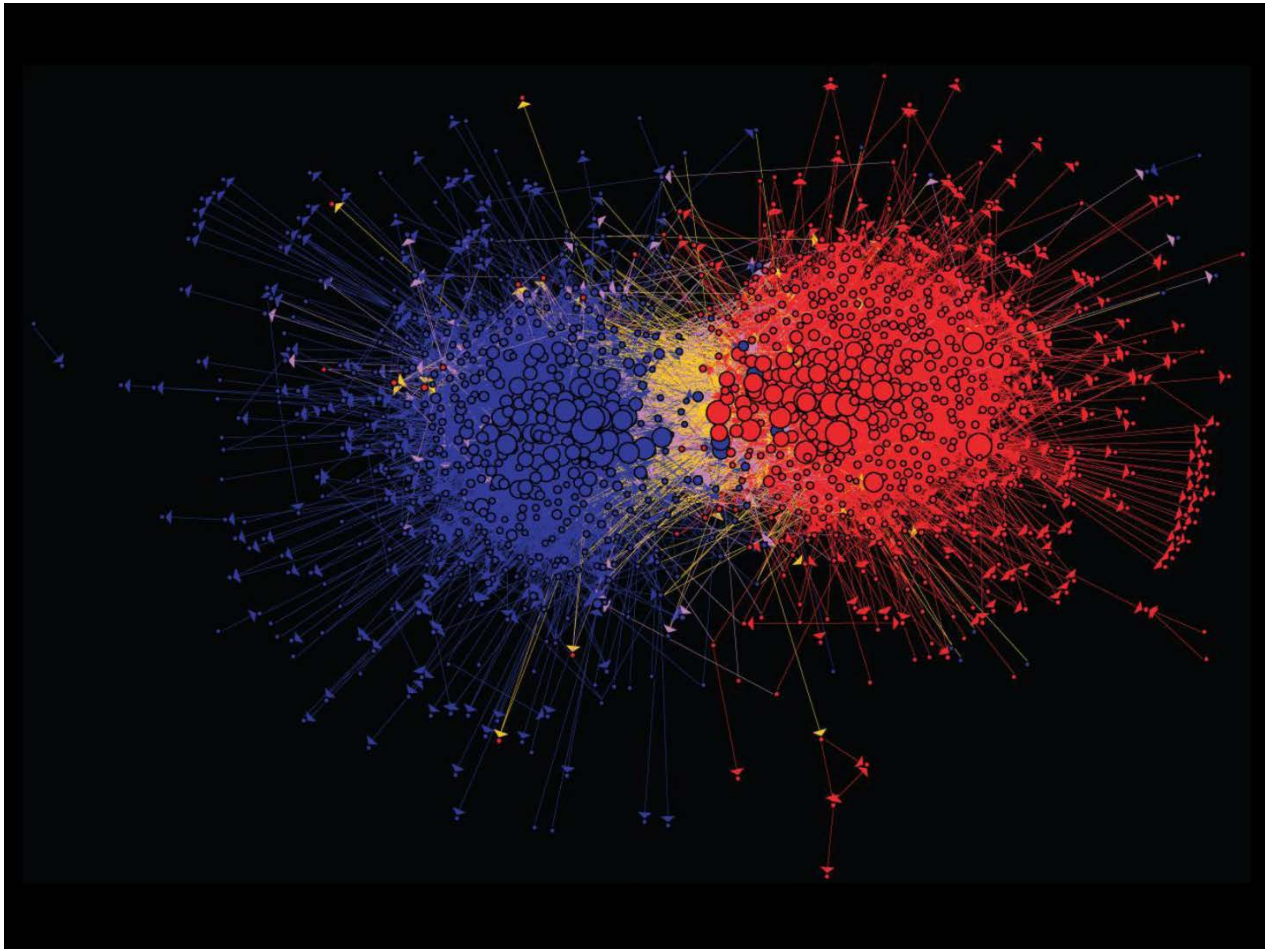
High dim. data	Graph data	Infinite data	Machine learning	Apps
Locality sensitive hashing	**PageRank,** SimRank	Filtering data streams	SVM	Recommender systems
Clustering	Community Detection	Web advertising	Decision Trees	Association Rules
Dimensionality reduction	Spam Detection	Queries on streams	Perceptron, kNN	Duplicate document detection

Graph Data: Social Networks

Facebook social graph

4-degrees of separation [Backstrom-Boldi-Rosa-Ugander-Vigna, 2011]

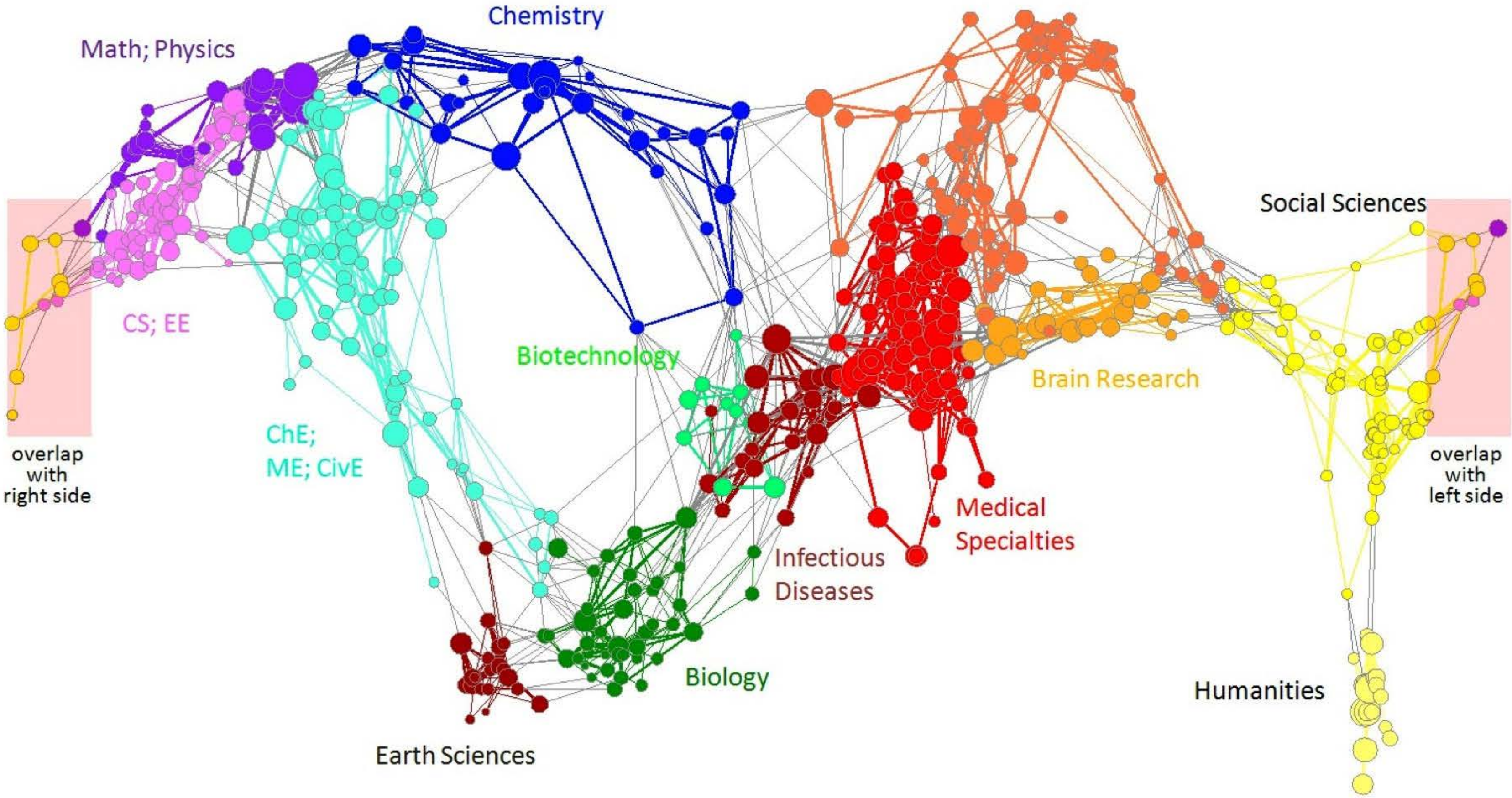
Graph Data: Media Networks



Connections between political blogs

Polarization of the network [Adamic-Glance, 2005]

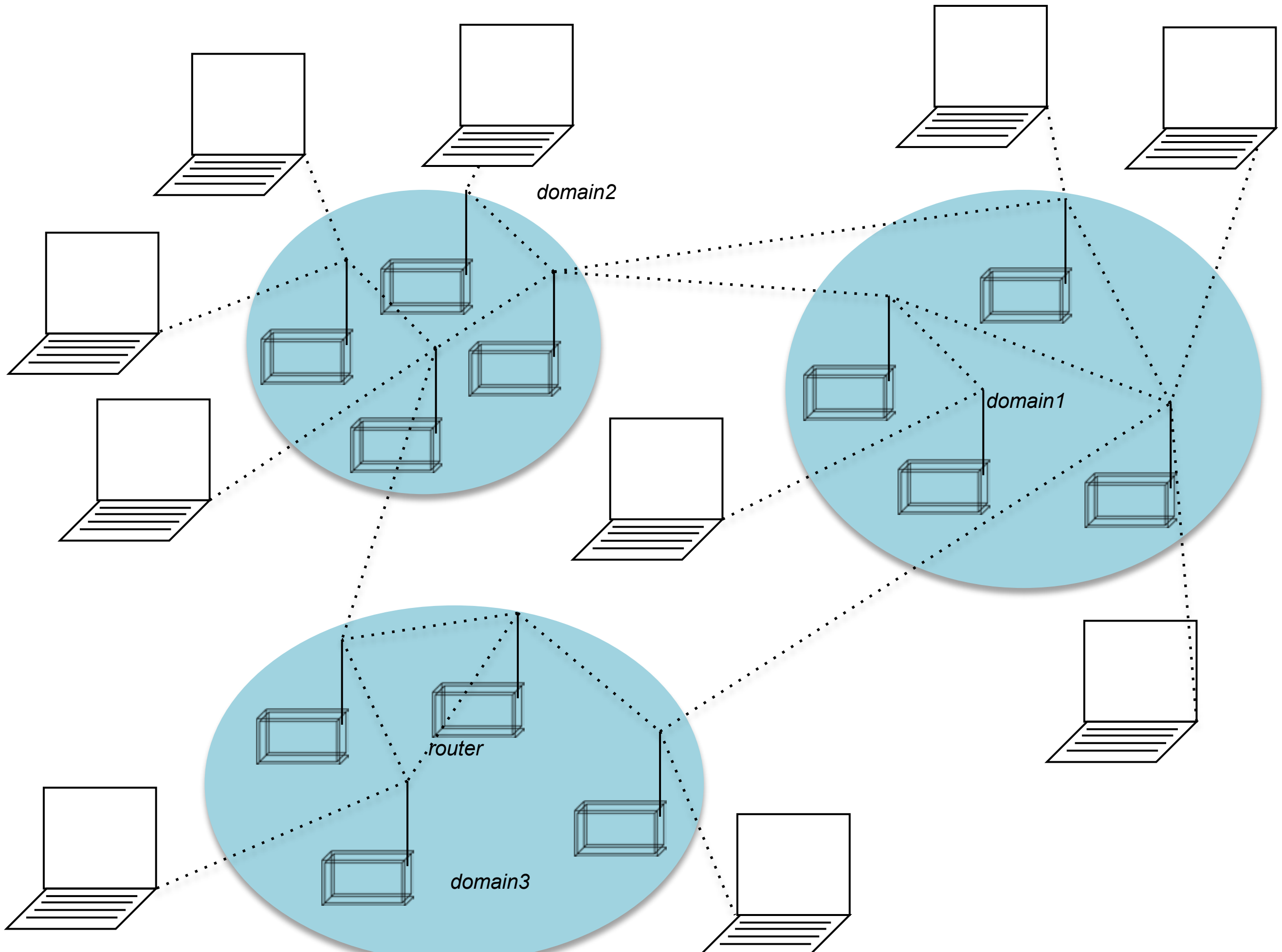
Graph Data: Information Nets



Citation networks and Maps of science

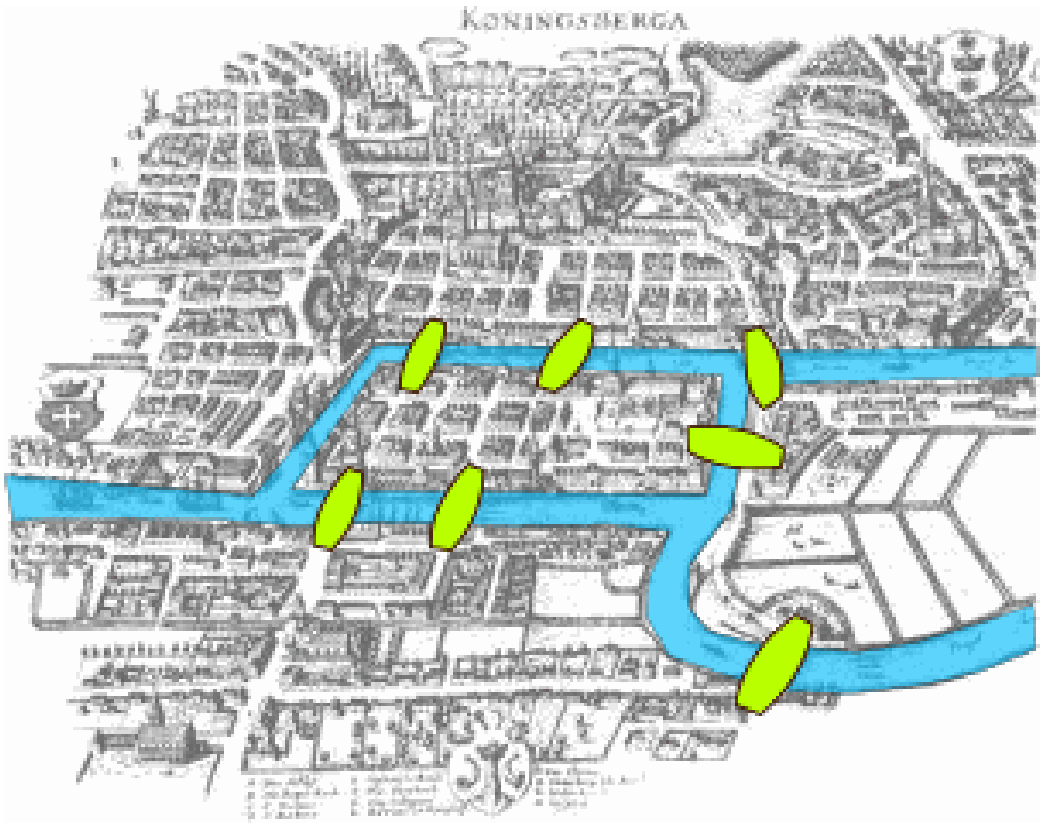
[Börner et al., 2012]

Graph Data: Communication Nets



Internet

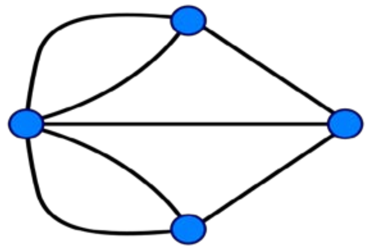
Graph Data: Technological Networks



Seven Bridges of Königsberg

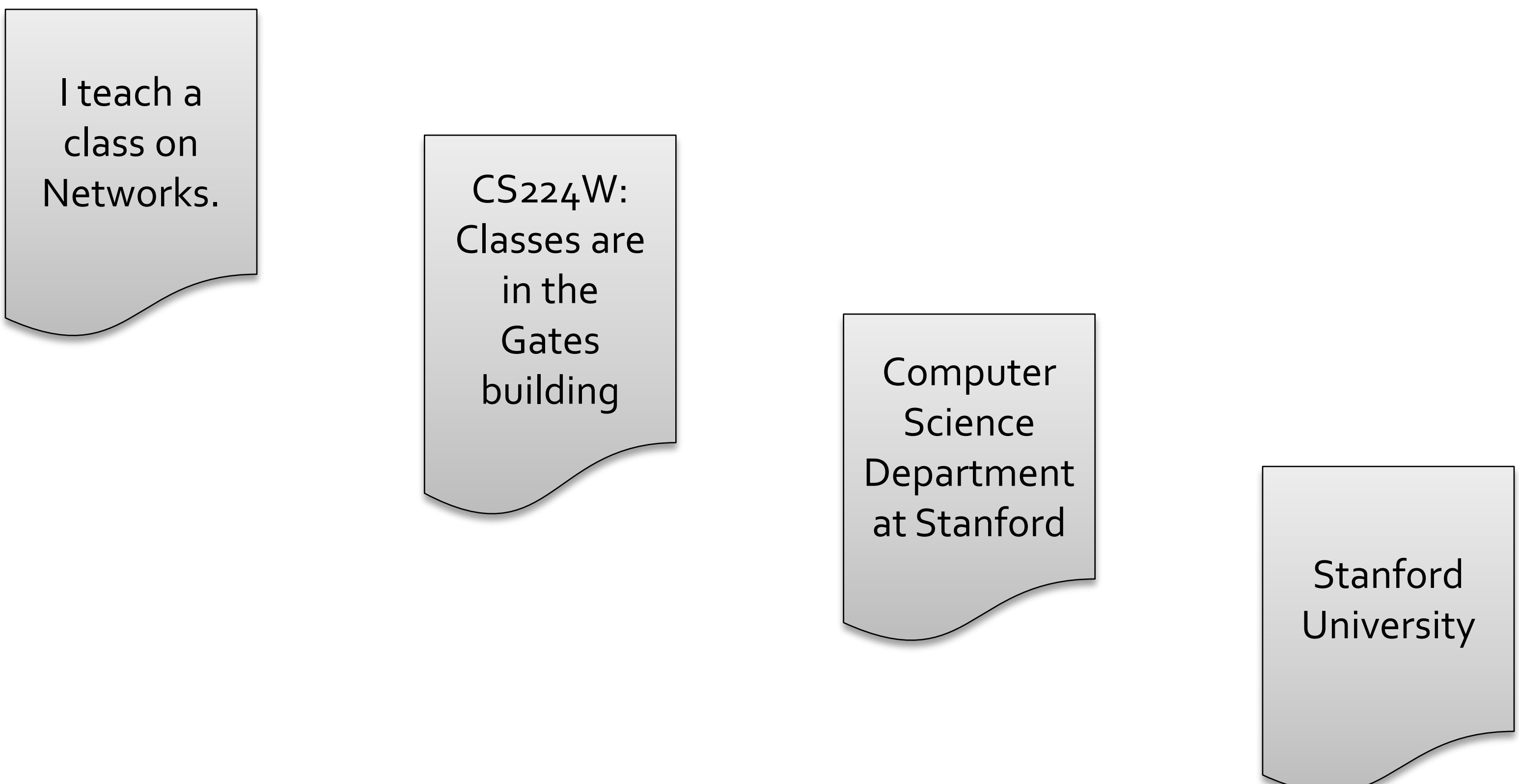
[Euler, 1735]

Return to the starting point by traveling each link of the graph once and only once.



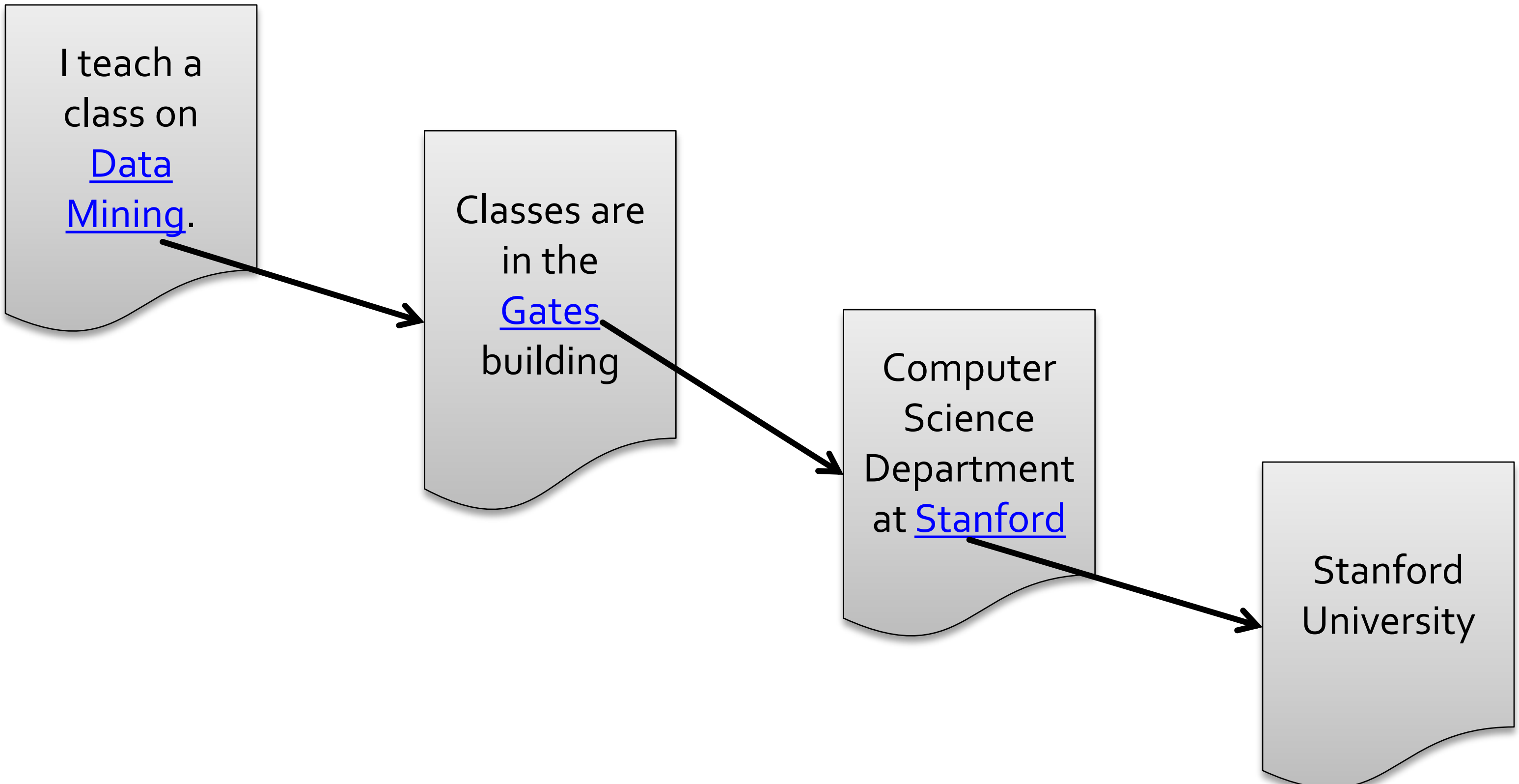
Web as a Graph

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

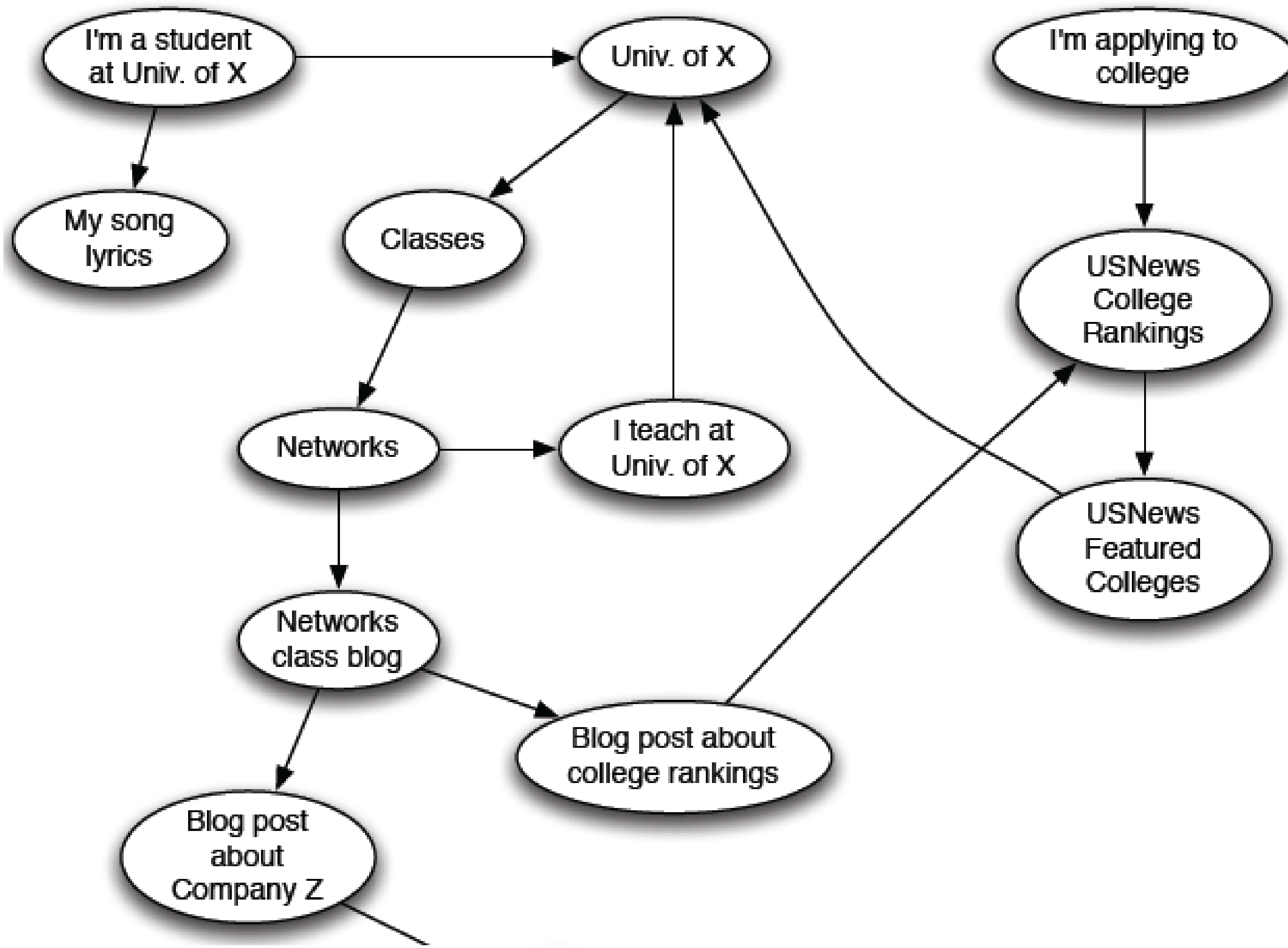


Web as a Graph

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



Web as a Directed Graph



Broad Question

- **How to organize the Web?**
- **First try:** Human curated **Web directories**
  - Yahoo, DMOZ, LookSmart
- **Second try: Web Search**
  - **Information Retrieval** investigates: Find relevant docs in a small and trusted set
    - Newspaper articles, Patents, etc.
  - **But:** Web is **huge**, full of untrusted documents, random things, web spam, etc.

Web Search: 2 Challenges

2 challenges of web search:

- **(1) Web contains many sources of information Who to "trust"?**
  - **Trick:** Trustworthy pages may point to each other!
- **(2) What is the "best" answer to query "newspaper"?**
  - No single right answer
  - **Trick:** Pages that actually know about newspapers might all be pointing to many newspapers

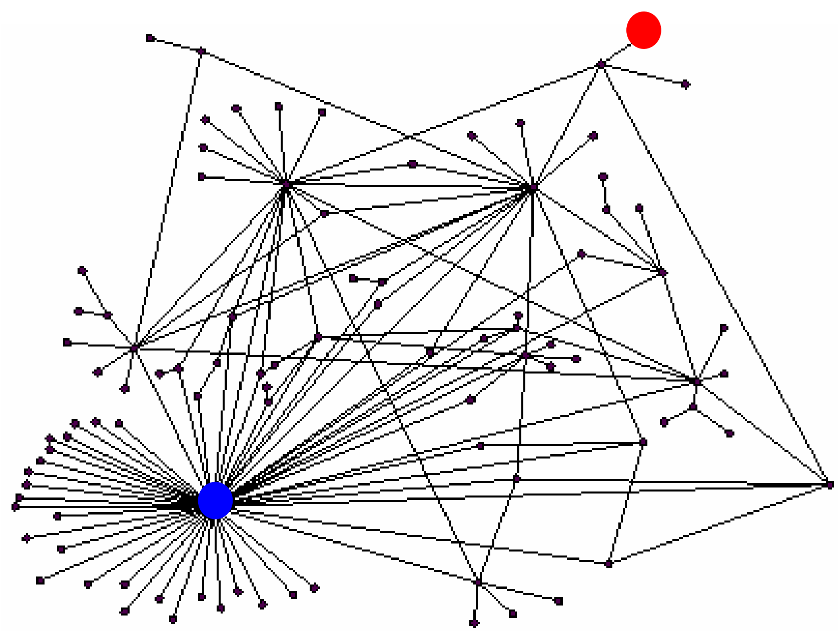
搜索的关键词比较含糊，或者问题比较含糊，这时候需要搞清楚意思：例如 coreference resolution

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!



Link Analysis Algorithms

- We will cover the following **Link Analysis approaches** for computing **importances** of nodes in a graph:
  - Page Rank
  - Hubs and Authorities (HITS)
  - Topic-Specific (Personalized) Page Rank
  - Web Spam Detection Algorithms

PageRank:
The "Flow" Formulation

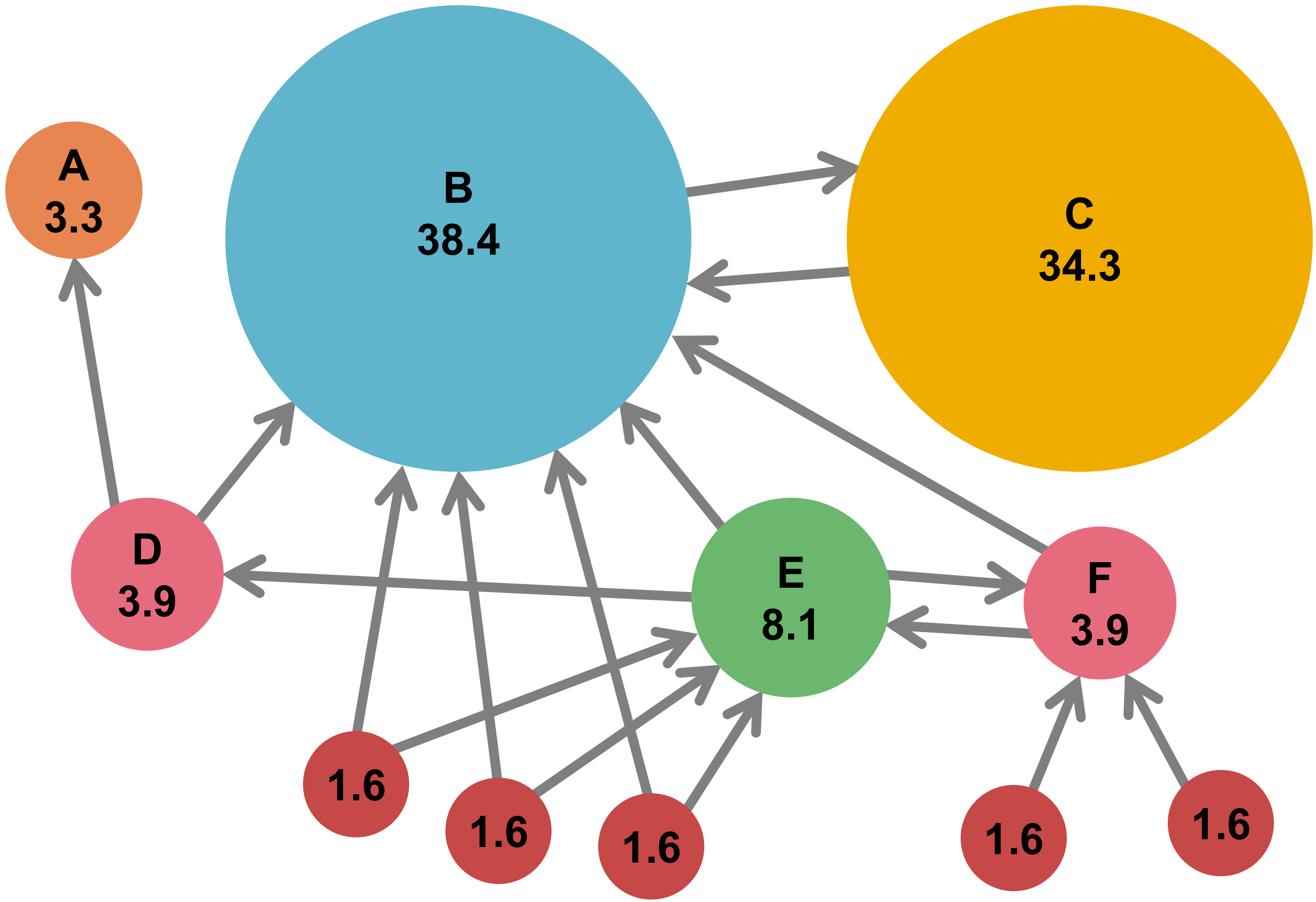
Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University

Links as Votes

- **Idea: Links as votes**
  - **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?** 类似于：本文猛不猛，看引用。还是个recursive的
  - Links from important pages count more
  - Recursive question!

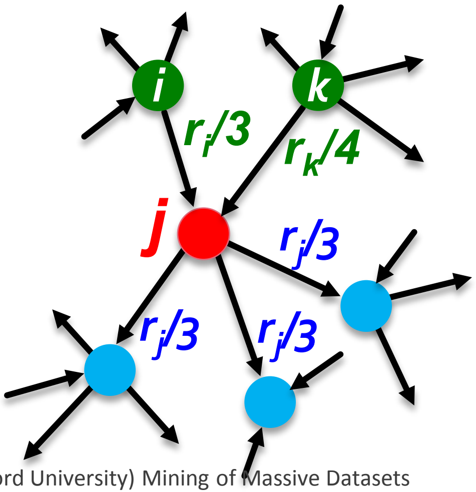
Inlinks = Links on other websites that send traffic to your site (进来的).
Inlinks are harder to fake than outlinks 所以通常采用比较多，类似于他引
Outlinks = Links on your site that send people to other sites （出去的）

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

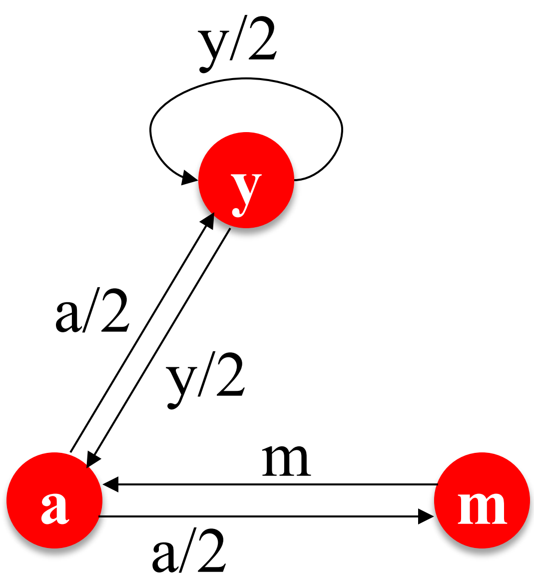
- A "vote" from an important page is worth more
- A page is important if it is pointed to by other important pages
- Define a "rank" r_j for page j

$$r_j = \sum_{i \to j} \frac{r_i}{\mathrm{d}_i}$$

此处的i是所有指向j的i网页的集合

d_i out-degree of node i

The web in 1839



"Flow" equations:

$$\mathbf{r_y} = \mathbf{r_y}/2 + \mathbf{r_a}/2$$
$$\mathbf{r_a} = \mathbf{r_y}/2 + \mathbf{r_m}$$
$$\mathbf{r_m} = \mathbf{r_a}/2$$

Solving the Flow Equations

Flow equations:

$r_y = r_y/2 + r_a/2$

$r_a = r_y/2 + r_m$

$r_m = r_a/2$

- **3 equations, 3 unknowns, no constants**
  - No unique solution
  - All solutions equivalent modulo the scale factor
- **Additional constraint forces uniqueness:**
  - $r_y + r_a + r_m = 1$
  - **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!**

PageRank: Matrix Formulation

- **Stochastic adjacency matrix M**
  - Let page i has d_i out-links
  - If $i \rightarrow j$, then $M_{ji} = \frac{1}{d_i}$ else $M_{ji} = 0$
    - M is a **column stochastic matrix**
      - Columns sum to **1**

对于一列，各个elements就是去各个j,自然和是1

- **Rank vector r:** vector with an entry per page
  - r_i is the importance score of page i
  - $\sum_i r_i = 1$
- **The flow equations can be written**

$$r_j = \sum_{i \to j} \frac{r_i}{\mathrm{d}_\mathrm{i}}$$

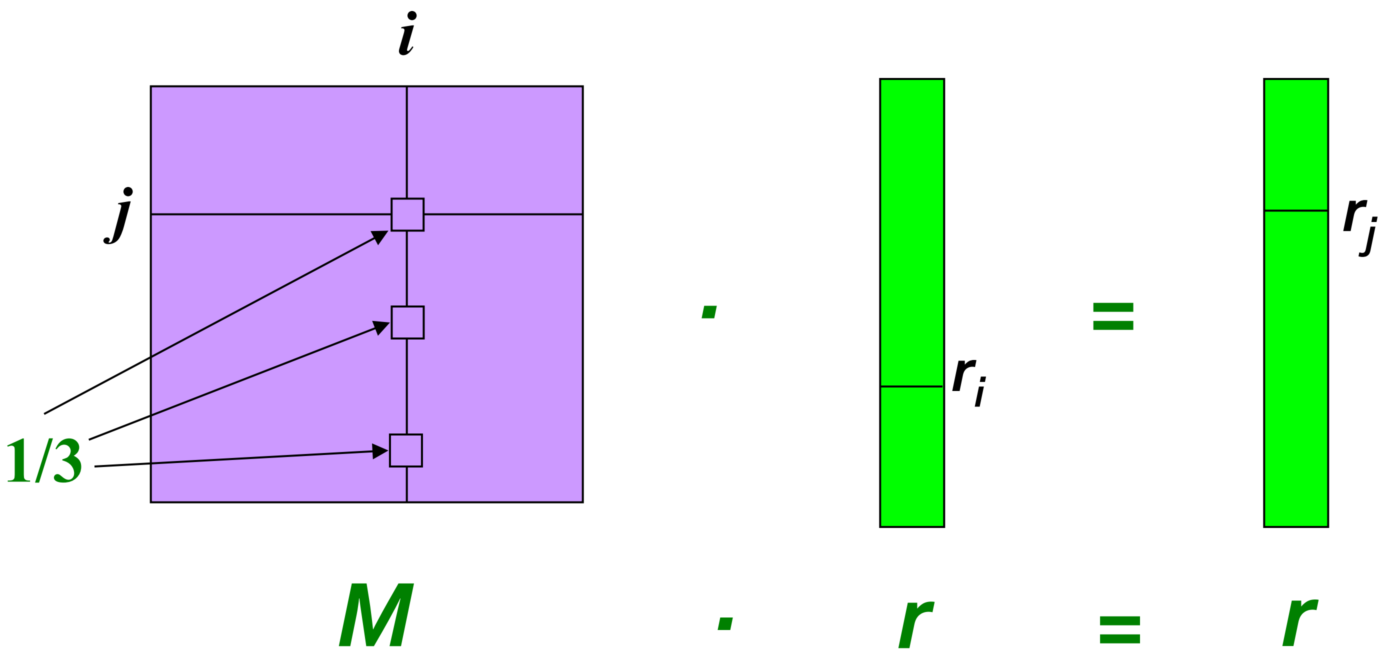
$$r = M \cdot r$$

Example

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

$$\boldsymbol{M} \cdot \boldsymbol{r} = \boldsymbol{r}$$

  - Suppose page *i* links to 3 pages, including *j*



Eigenvector Formulation

NOTE: x is an eigenvector with the corresponding eigenvalue λ if:

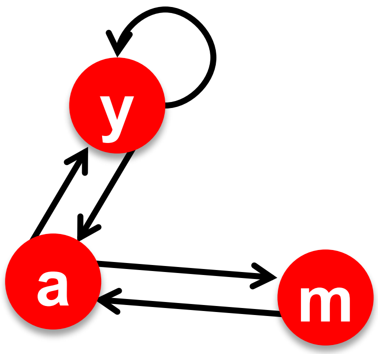
$$Ax = \lambda x$$

- The flow equations can be written

$$r = M \cdot r$$

- So the **rank vector** r is an **eigenvector** of the stochastic web matrix M
  - In fact, its first or principal eigenvector, with corresponding eigenvalue *1*
    - Largest eigenvalue of M is **1** since M is column stochastic
      - ***Why?*** *We know* $\mathbf{r}$ *is unit length and each column of* $\mathbf{M}$ *sums to one, so* $\boldsymbol{Mr} \leq \mathbf{1}$
- **We can now efficiently solve for *r*!**
  The method is called Power iteration

Example: Flow Equations & M



	y	a	m
y	½	½	0
a	½	0	1
m	0	½	0

$$\boldsymbol{r = M \cdot r}$$

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

$$\begin{bmatrix} y \\ a \\ m \end{bmatrix} = \begin{bmatrix} \tfrac{1}{2} & \tfrac{1}{2} & 0 \\ \tfrac{1}{2} & 0 & 1 \\ 0 & \tfrac{1}{2} & 0 \end{bmatrix} \begin{bmatrix} y \\ a \\ m \end{bmatrix}$$

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: $\mathbf{r}^{(0)} = [1/N,\ldots.,1/N]^T$
  - Iterate: $\mathbf{r}^{(t+1)} = \mathbf{M} \cdot \mathbf{r}^{(t)}$
  - Stop when $|\mathbf{r}^{(t+1)} - \mathbf{r}^{(t)}|_1 < \varepsilon$

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

d_i …. out-degree of node i

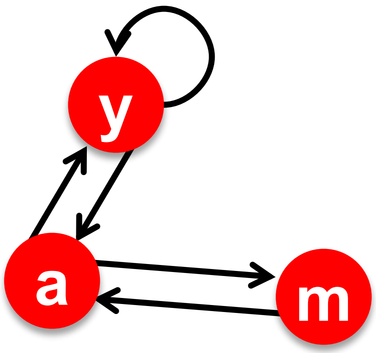
$|\mathbf{x}|_1 = \sum_{1 \le i \le N} |x_i|$ is the $\mathbf{L_1}$ norm
Can use any other vector norm, e.g., Euclidean

PageRank: How to solve?

- **Power Iteration:**
  - Set $r_j = 1/N$
  - **1:** $r'_j = \sum_{i \to j} \frac{r_i}{d_i}$
  - **2:** $r = r'$
  - If not converged: goto **1**
- **Example:**

$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{matrix} 1/3 \\ 1/3 \\ 1/3 \end{matrix}$$

Iteration 0, 1, 2,

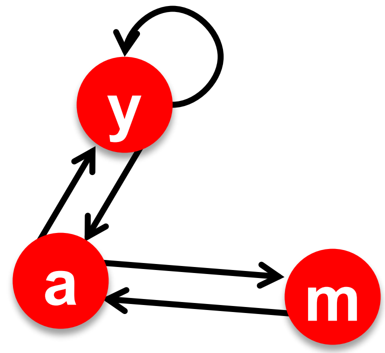


	y	a	m
y	½	½	0
a	½	0	1
m	0	½	0

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

PageRank: How to solve?

- **Power Iteration:**
  - Set $r_j = 1/N$
  - **1:** $r'_j = \sum_{i \to j} \frac{r_i}{d_i}$
  - **2:** $r = r'$
  - If not converged: goto **1**
- **Example:**



	y	a	m
y	½	½	0
a	½	0	1
m	0	½	0

$r_y = r_y/2 + r_a/2$

$r_a = r_y/2 + r_m$

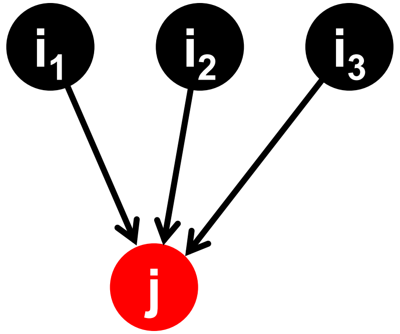
$r_m = r_a/2$

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

Iteration 0, 1, 2,

Random Walk Interpretation

- **Imagine a random web surfer:**
  - At any time $\boldsymbol{t}$, surfer is on some page $\boldsymbol{i}$
  - At time $\boldsymbol{t+1}$, the surfer follows an out-link from $\boldsymbol{i}$ uniformly at random
  - Ends up on some page $\boldsymbol{j}$ linked from $\boldsymbol{i}$
  - Process repeats indefinitely
- **Let:**
  - $\boldsymbol{p(t)}$ … vector whose $\boldsymbol{i}^{\text{th}}$ coordinate is the prob. that the surfer is at page $\boldsymbol{i}$ at time $\boldsymbol{t}$
  - So, $\boldsymbol{p(t)}$ is a probability distribution over pages

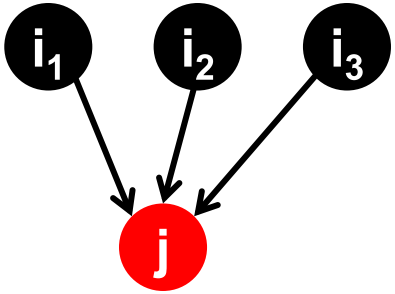


$$r_j = \sum_{i \to j} \frac{r_i}{\mathrm{d}_{\mathrm{out}}(\mathrm{i})}$$

The Stationary Distribution

- **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**

Existence and Uniqueness

- **A central result from the theory of random walks (a.k.a. Markov processes):**

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**

PageRank: The Google Formulation

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University

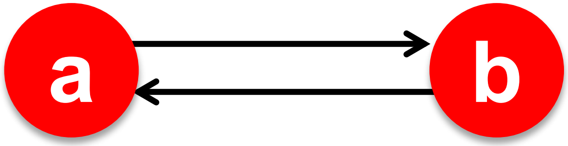
PageRank: Three Questions

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

- **Does this converge?** 层层前进，感觉很稳 记住这个套路
- **Does it converge to what we want?**
- **Are results reasonable?**

Does this converge?

- The "Spider trap" problem:



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

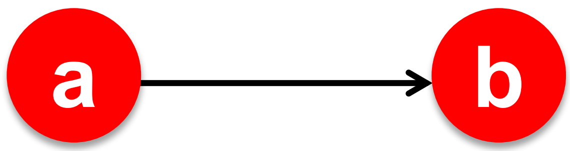
- Example:

$$\begin{array}{ccccc} \mathrm{r_a} & & 1 & 0 & 1 & 0 \\ & = & & & & \\ \mathrm{r_b} & & 0 & 1 & 0 & 1 \end{array}$$

Iteration 0, 1, 2,

Does it converge to what we want?

- **The "Dead end" problem:**



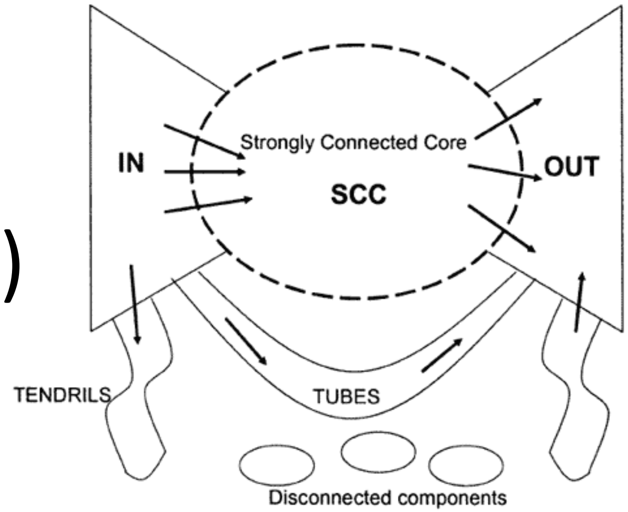
$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{\mathrm{d}_\mathrm{i}}$$

- **Example:**

$$\begin{array}{c} r_a \\ r_b \end{array} = \begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{array}$$

Iteration 0, 1, 2,

RageRank: Problems



2 problems:

- **(1)** Some pages are **dead ends** (have no out-links)

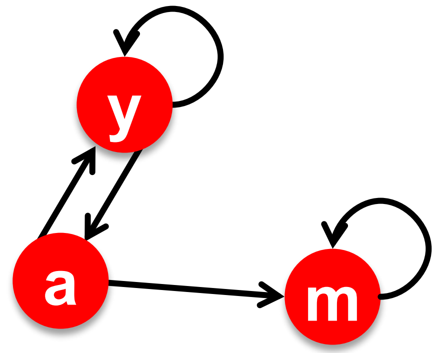
导致结果变成零

  - Such pages cause importance to "leak out"

- **(2) Spider traps** (all out-links are within the group)
  - Eventually spider traps absorb all importance

Problem: Spider Traps

- **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	1

$$\mathbf{r_y} = \mathbf{r_y}/2 + \mathbf{r_a}/2$$
$$\mathbf{r_a} = \mathbf{r_y}/2$$
$$\mathbf{r_m} = \mathbf{r_a}/2 + \mathbf{r_m}$$

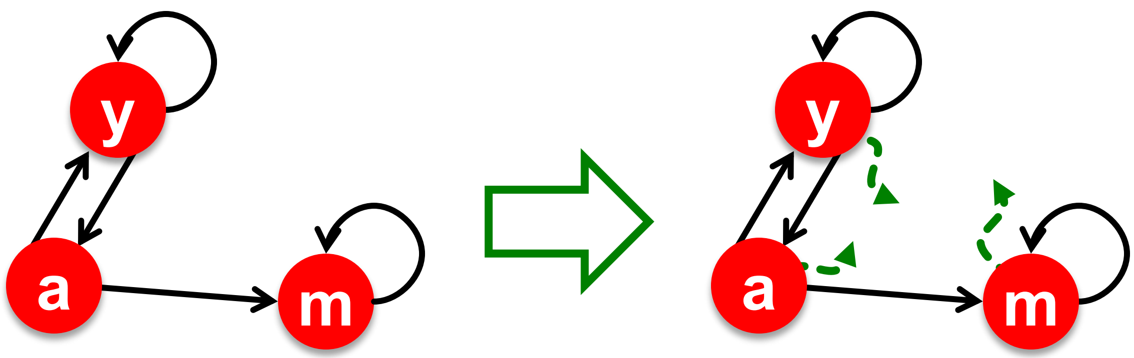
- **Example:**

$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{matrix} 1/3 & 2/6 & 3/12 & 5/24 & & 0 \\ 1/3 & 1/6 & 2/12 & 3/24 & \ldots & 0 \\ 1/3 & 3/6 & 7/12 & 16/24 & & 1 \end{matrix}$$

Iteration 0, 1, 2,

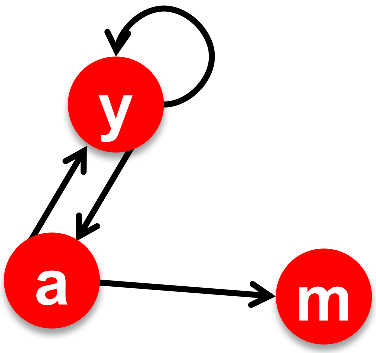
Solution: Random Teleports

- **The Google solution for spider traps: At each time step, the random surfer has two options**
  - With prob. β, follow a link at random 直接乱跳
  - 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**



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

$\mathbf{r_y = r_y/2 + r_a/2}$
$\mathbf{r_a = r_y/2}$
$\mathbf{r_m = r_a/2}$

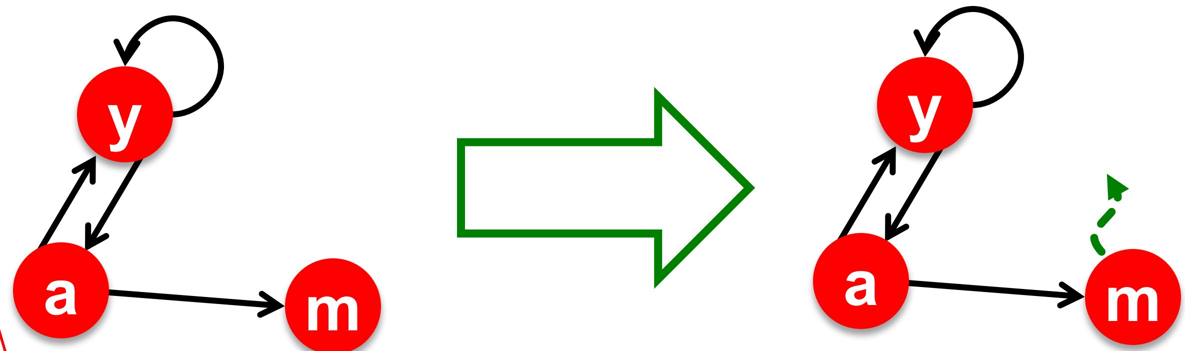
- **Example:**

$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{array}{cccccc} 1/3 & 2/6 & 3/12 & 5/24 & & 0 \\ 1/3 & 1/6 & 2/12 & 3/24 & \dots & 0 \\ 1/3 & 1/6 & 1/12 & 2/24 & & 0 \end{array}$$

Iteration 0, 1, 2,

Solution: Always Teleport

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



起始点

终点

	y	a	m
y	½	½	0
a	½	0	0
m	0	½	0

	y	a	m
y	½	½	⅓
a	½	0	⅓
m	0	½	⅓

Why Teleports Solve the Problem?

$$r^{(t+1)} = M r^{(t)}$$

Markov chains

- Set of states ***X***
- Transition matrix ***P*** where $P_{ij} = P(X_t = i \mid X_{t-1} = j)$
- **π** specifying the stationary probability of being at each state $x \in X$
- Goal is to find **π** such that **π = P π**

Why is This Analogy Useful?

- **Theory of Markov chains**
- **Fact:** For **any start vector**, the power method applied to a Markov transition matrix ***P*** will **converge** to a **unique** positive stationary vector as long as ***P*** is **stochastic**, **irreducible** and **aperiodic**.

带telereport的P满足
这三个条件

Make M Stochastic

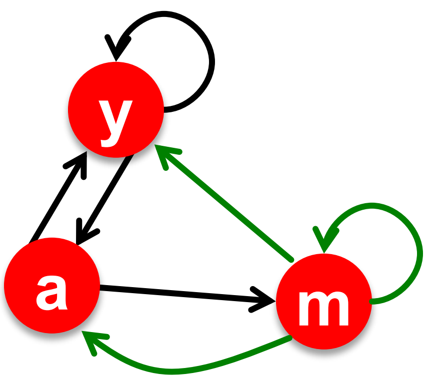
- **Stochastic:** Every column sums to **1**
- **Solution:** Add **green** links

无出度时为
1，其他是0

$$A = M + a^T\left(\frac{1}{n}e\right)$$

- a_i =1 if node *i* has out deg 0, =0 else
- **e** vector of all 1s

全都是1



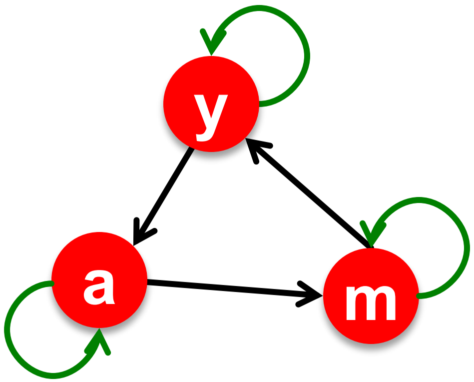
	y	a	m
y	½	½	1/3
a	½	0	1/3
m	0	½	1/3

$$\mathbf{r}_y = \mathbf{r}_y/2 + \mathbf{r}_a/2 + \mathbf{r}_m/3$$
$$\mathbf{r}_a = \mathbf{r}_y/2 + \mathbf{r}_m/3$$
$$\mathbf{r}_m = \mathbf{r}_a/2 + \mathbf{r}_m/3$$

Make M Aperiodic

- A chain is **periodic** if there exists ***k* > 1** such that the interval between two visits to some state ***s*** is always a multiple of ***k***

- **Solution:** Add **green** links

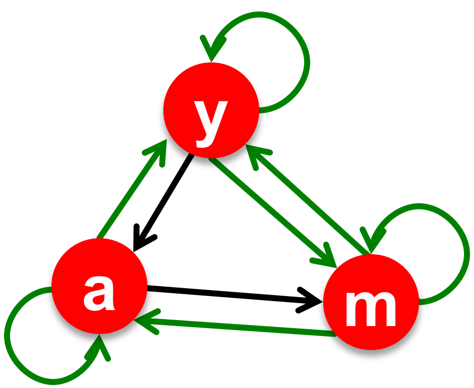


Make *M* Irreducible

- From any state, there is a non-zero probability of going from any one state to any another

不会stuck在某一个点

- **Solution:** Add **green** links



Solution: Random Jumps

- **Google's solution that does it all:**
  - Makes ***M*** **stochastic, aperiodic, irreducible**
- **At each step, random surfer has two options:**
  - With probability β, follow a link at random
  - With probability $1\text{-}\beta$, jump to some random page
- **PageRank equation** [Brin-Page, 98]

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

d_i ... out-degree of node i

The above formulation assumes that M has no dead ends. We can either preprocess matrix M **(bad!)** or explicitly follow random teleport links with probability 1.0 from dead-ends.

The Google Matrix

- **PageRank equation** [Brin-Page, 98]

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

- **The Google Matrix *A*:**

$$A = \beta\, M + (1 - \beta) \frac{1}{n} \boldsymbol{e} \cdot \boldsymbol{e}^T$$

e … vector of all 1s

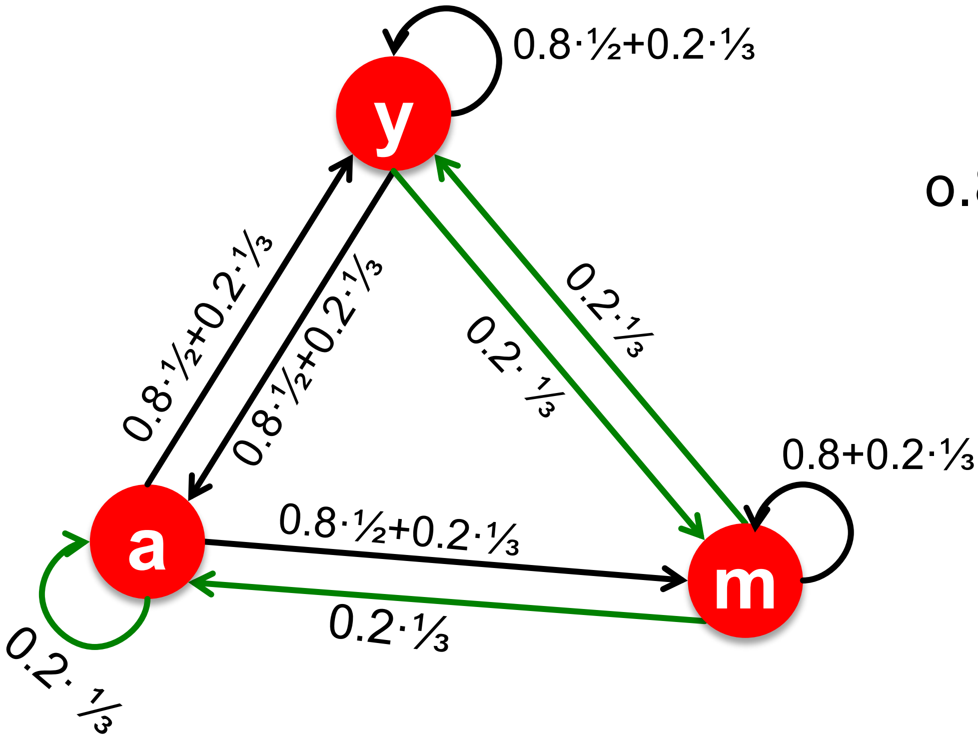
- ***A* is stochastic, aperiodic and irreducible, so**

$$\boldsymbol{r}^{(t+1)} = \boldsymbol{A} \cdot \boldsymbol{r}^{(t)}$$

- **What is β?**
  - In practice $\beta = 0.8, 0.9$ (make 5 steps and jump)

0.85

Random Teleports ($\beta = 0.8$)



$$
\overset{\mathbf{M}}{0.8\begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 1 \end{bmatrix}} + 0.2 \overset{\mathbf{1/n \cdot 1 \cdot 1^T}}{\begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix}}
$$

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

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

How do we really compute the PageRank?

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University

Computing Page Rank

- **Key step is matrix-vector multiplication**
  - $\boldsymbol{r}^{new} = \boldsymbol{A} \cdot \boldsymbol{r}^{old}$
- Easy if we have enough main memory to hold $\mathbf{A}$, $\mathbf{r}^{old}$, $\mathbf{r}^{new}$
- **Say N = 1 billion pages**
  - We need 4 bytes for each entry (say)
  - 2 billion entries for vectors, approx 8GB
  - Matrix **A** has **N²** entries
    - 10^{18} is a large number!

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

$$\boldsymbol{A} = 0.8 \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & 1 \end{bmatrix} + 0.2 \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix}$$

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

Matrix Formulation

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

Rearranging the Equation

- $\boldsymbol{r} = \boldsymbol{A} \cdot \boldsymbol{r}$, **where** $\boldsymbol{A_{ij}} = \boldsymbol{\beta\, M_{ij}} + \frac{\mathbf{1}-\boldsymbol{\beta}}{\boldsymbol{N}}$
- $r_i = \sum_{j=1}^{N} A_{ij} \cdot r_j$
- $r_i = \sum_{j=1}^{N} \left[\beta\, M_{ij} + \frac{1-\beta}{N}\right] \cdot r_j$

$$= \sum_{j=1}^{N} \beta\, M_{ij} \cdot r_j + \sum_{j=1}^{N} \frac{1-\beta}{N} r_j$$

$$= \sum_{j=1}^{N} \beta\, M_{ij} \cdot r_j + \frac{1-\beta}{N}$$ since $\sum r_j = 1$

- **So we get:** $\boldsymbol{r} = \boldsymbol{\beta\, M} \cdot \boldsymbol{r} + \left[\frac{\mathbf{1}-\boldsymbol{\beta}}{\boldsymbol{N}}\right]_{\boldsymbol{N}}$

此处把M提取出来，因为M是稀疏的

Note: Here we assumed **M** has no dead-ends.

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

Sparse Matrix Formulation

- We just rearranged the **PageRank equation**

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

  - where **[(1-β)/N]$_N$** is a vector with all ***N*** entries **(1-β)/N**

- ***M*** is a **sparse matrix!** (with no dead-ends)
  - 10 links per node, approx 10N entries
- **So in each iteration, we need to:**
  - Compute $\boldsymbol{r}^{\text{new}} = \beta \boldsymbol{M} \cdot \boldsymbol{r}^{\text{old}}$
  - Add a constant value **(1-β)/N** to each entry in $\boldsymbol{r}^{\text{new}}$
    - **Note if *M* contains dead-ends then $\sum_i r_i^{new} < 1$ and we also have to renormalize *r*$^{\text{new}}$ so that it sums to 1**

PageRank: The Complete Algorithm

- **Input: Graph G and parameter β**
  - Directed graph G with **spider traps** and **dead ends**
  - Parameter β
- **Output: PageRank vector r**
  - **Set:** $r_j^{(0)} = \frac{1}{N}, \quad t = 1$
  - **do:**
    - $\forall j: \boldsymbol{r'}_j^{(t)} = \sum_{i \to j} \boldsymbol{\beta} \frac{\boldsymbol{r}_i^{(t-1)}}{\boldsymbol{d}_i}$

      $\boldsymbol{r'}_j^{(t)} = \boldsymbol{0}$ if in-deg. of $\boldsymbol{j}$ is $\boldsymbol{0}$

      Dead Ends
    - **Now re-insert the leaked PageRank:**

      $\forall \boldsymbol{j}: \boldsymbol{r}_j^{(t)} = \boldsymbol{r'}_j^{(t)} + \frac{\boldsymbol{1-S}}{\boldsymbol{N}}$ **where:** $S = \sum_j r'^{(t)}_j$
    - $\boldsymbol{t = t + 1}$
  - **while** $\sum_j \left| r_j^{(t)} - r_j^{(t-1)} \right| > \varepsilon$