BBS654
Data Mining

Pinar Duygulu

Slides are adapted from

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

Nazli Ikizler

Mustafa Ozdal

http://www.mmds.org

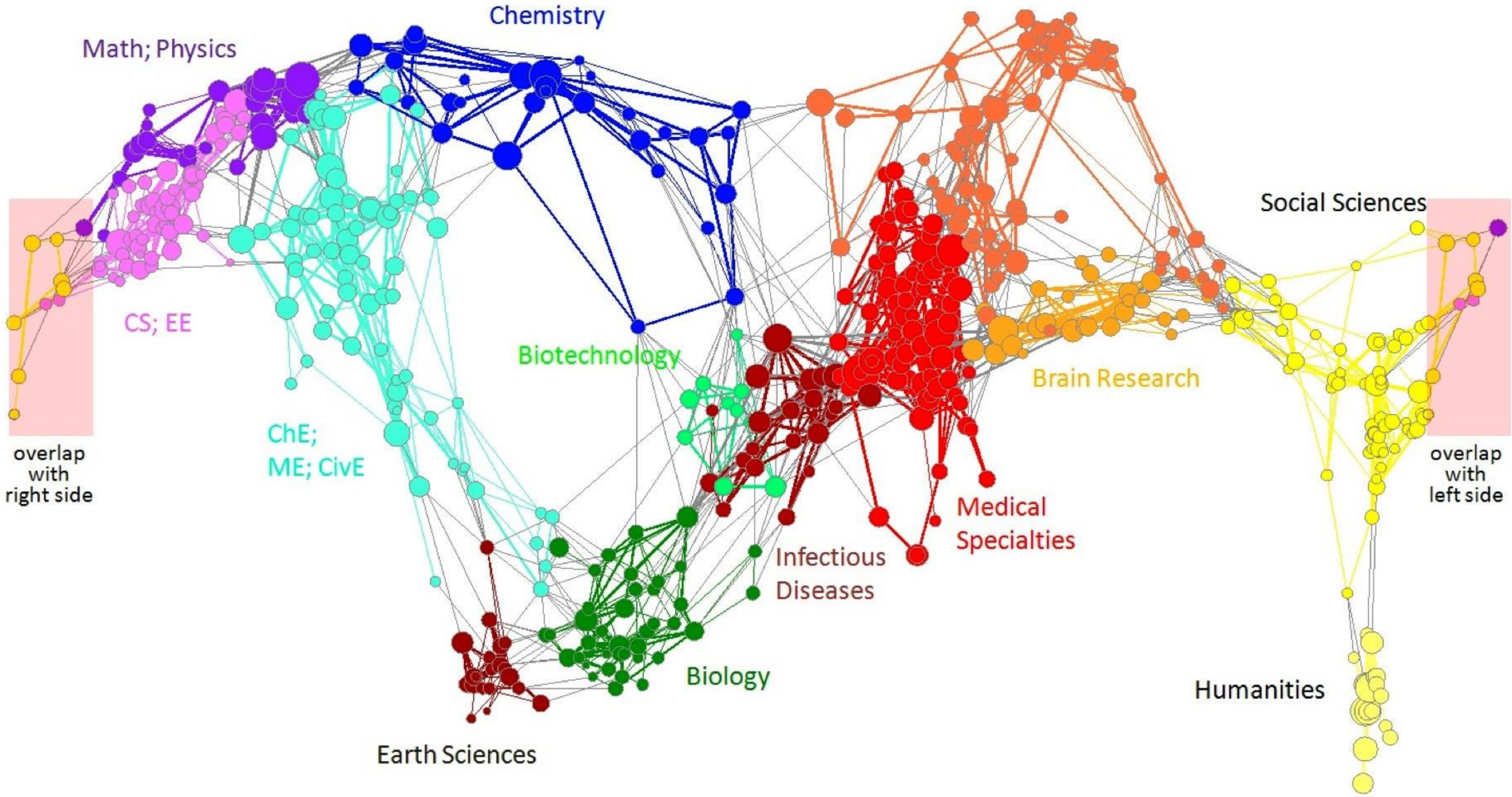
Analysis of Large Graphs: Link Analysis, PageRank

Graph Data: Social Networks

Facebook social graph

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

Graph Data: Information Nets

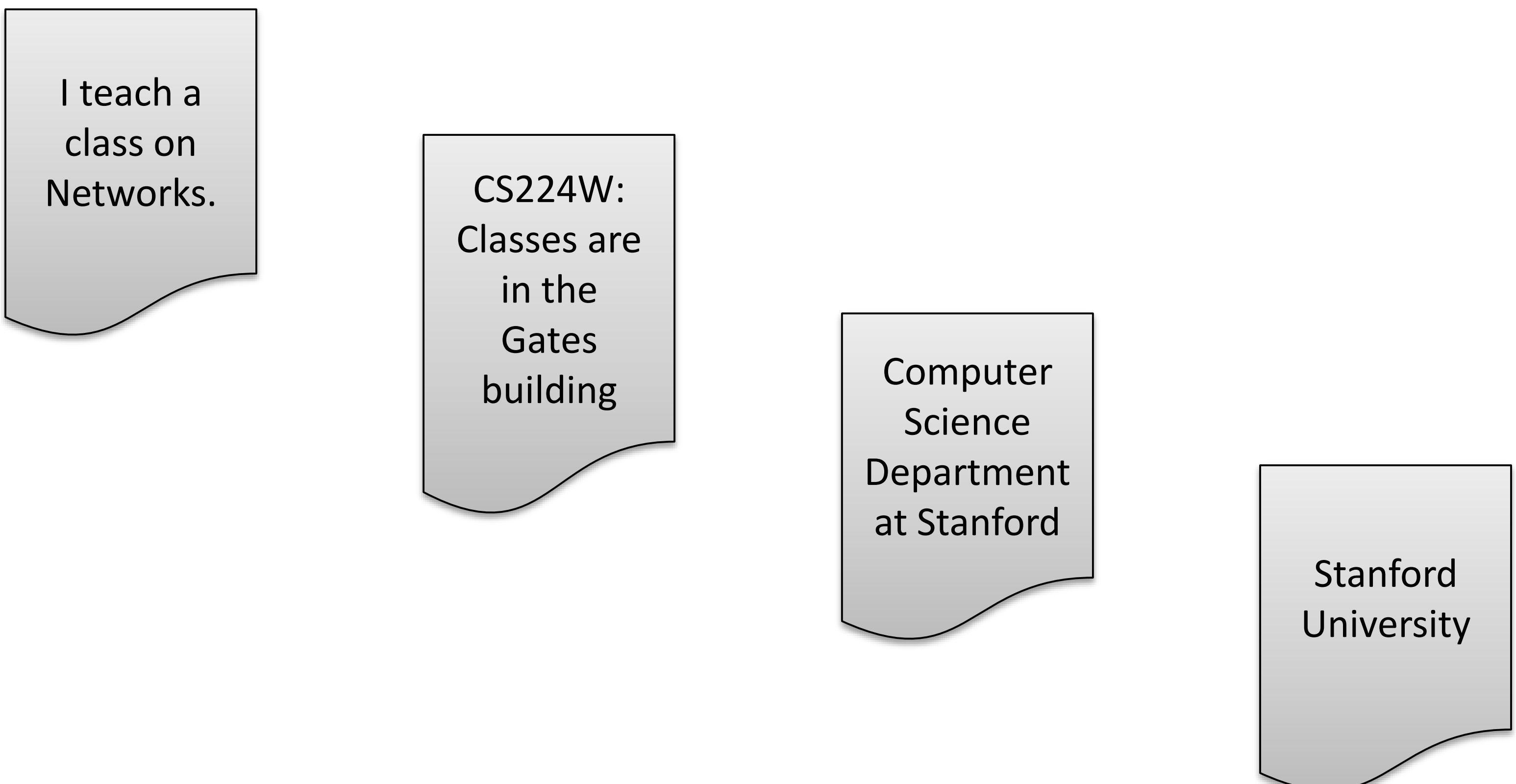


Citation networks and Maps of science

[Börner et al., 2012]

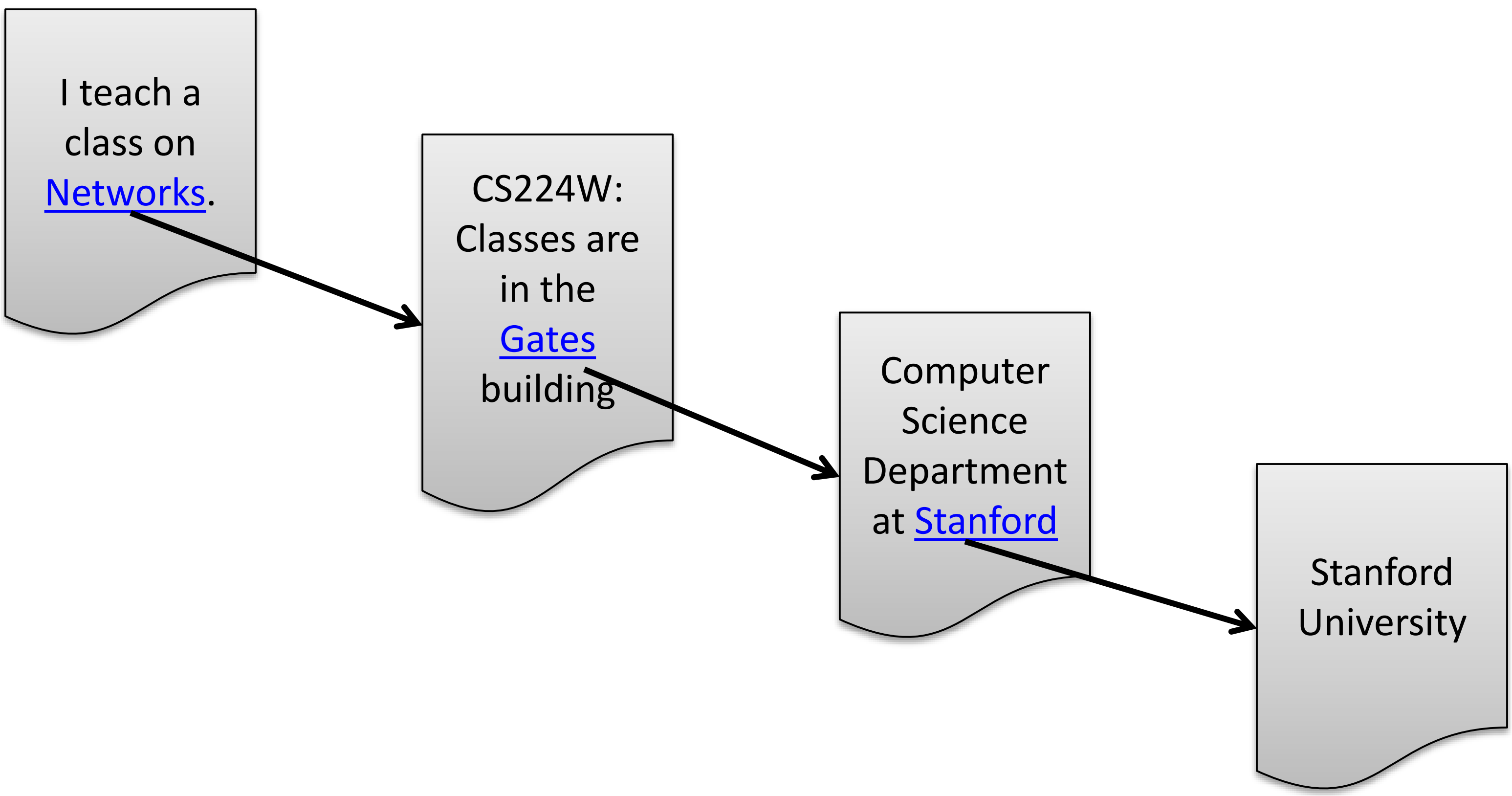
Web as a Graph

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

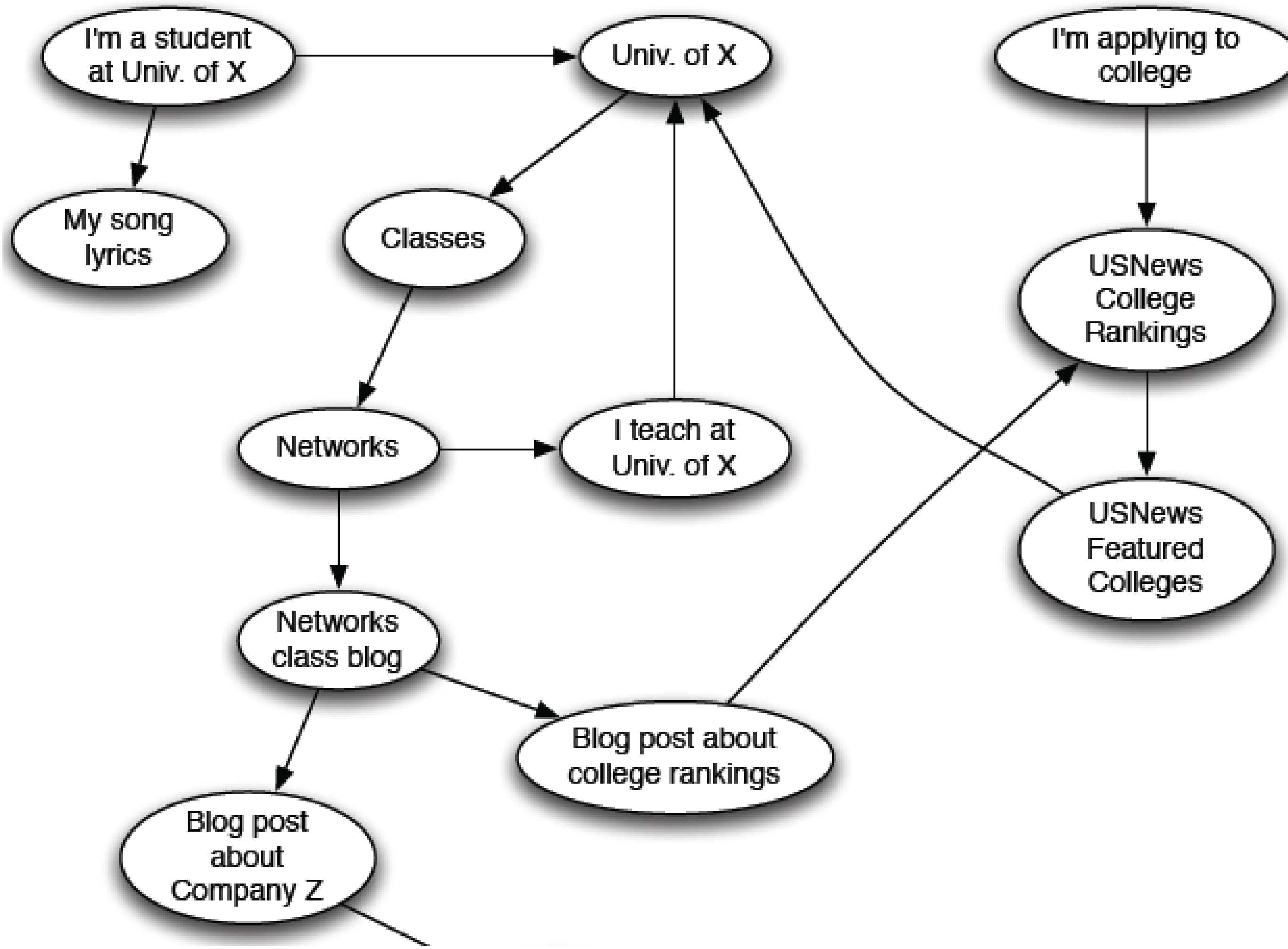


Web as a Graph

- **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

Early Search Engines

- Inverted index
  - Data structure that return pointers to all pages a term occurs
- Which page to return first?
  - Where do the search terms appear in the page?
  - How many occurrences of the search terms in the page?
- What if a spammer tries to fool the search engine?

Fooling Early Search Engines

- Example: A spammer wants his page to be in the top search results for the term “movies”.
- <u>Approach 1</u>:
  - Add thousands of copies of the term “movies” to your page.
  - Make them invisible.
- <u>Approach 2</u>:
  - Search the term “movies”.
  - Copy the contents of the top page to your page.
  - Make it invisible.
- Problem: Ranking only based on page contents
- Early search engines almost useless because of spam.

Google's Innovations

- *<u>Basic idea</u>: Search engine believes what other pages say about you instead of what you say about yourself.*

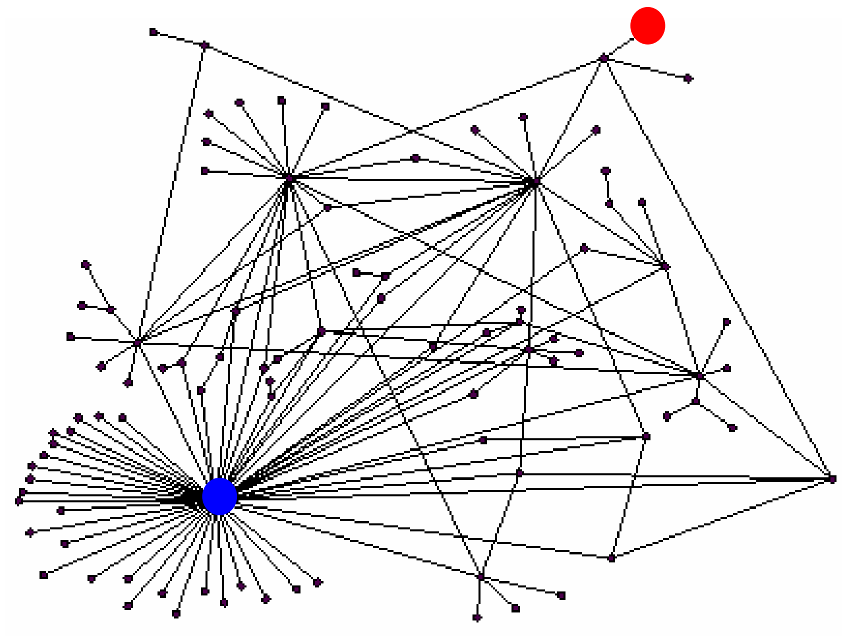
- Main innovations:
  1. Define the importance of a page based on:
     - How many pages point to it?
     - **How important are those pages?**
  2. Judge the contents of a page based on:
     - Which terms appear in the page?
     - **Which terms are used to link to the page?**

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!

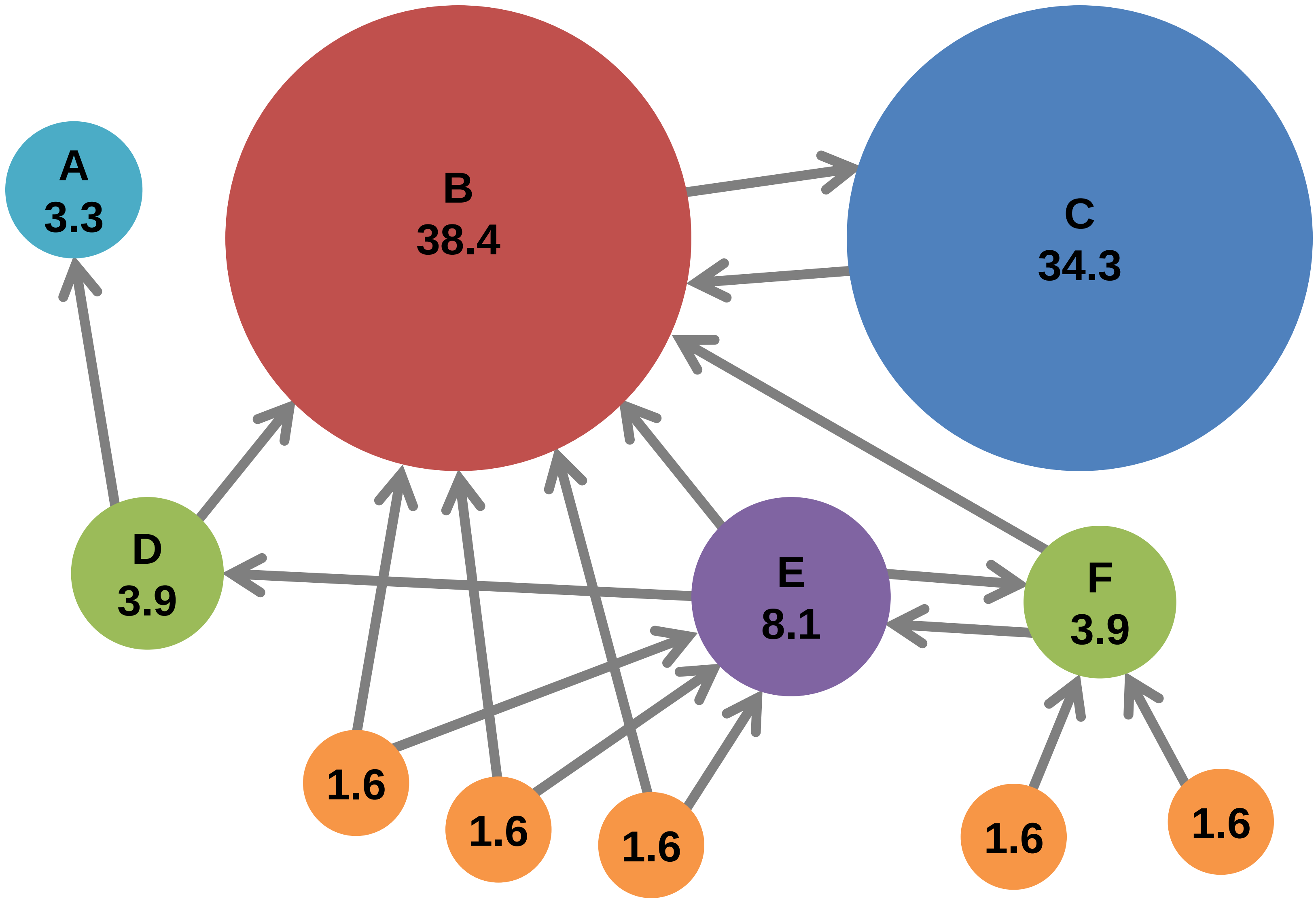


PageRank:
The "Flow" Formulation

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?**
  - **Links from important pages count more**
  - Recursive question!

Example: PageRank Scores



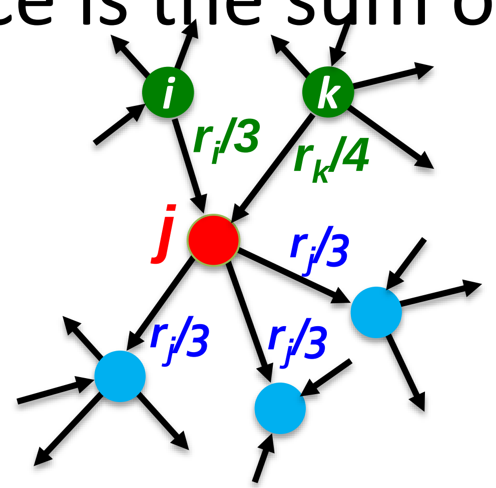
Simple Recursive Formulation

- 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

$$r_j = r_i/3 + r_k/4$$



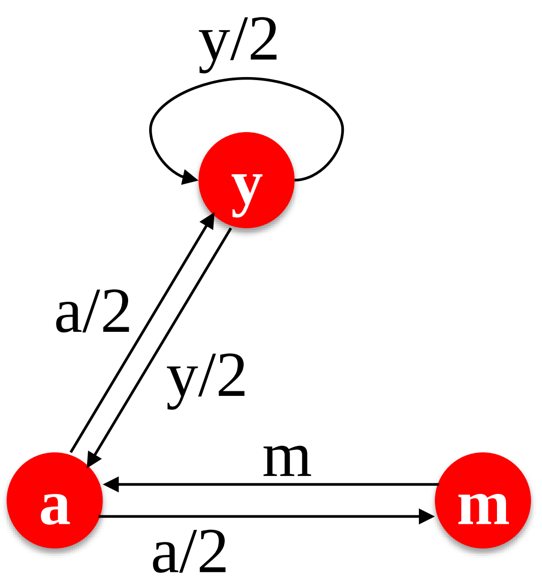
PageRank: The "Flow" Model

- 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

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

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

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:**
  - $\boldsymbol{r_y + r_a + r_m = 1}$
  - **Solution:** $\boldsymbol{r_y = \frac{2}{5},\ r_a = \frac{2}{5},\ r_m = \frac{1}{5}}$
- **Gaussian elimination method works for small examples, but we need a better method for large web-size graphs**
- **We need a new formulation!**

Page Rank: Matrix Formulation

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

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

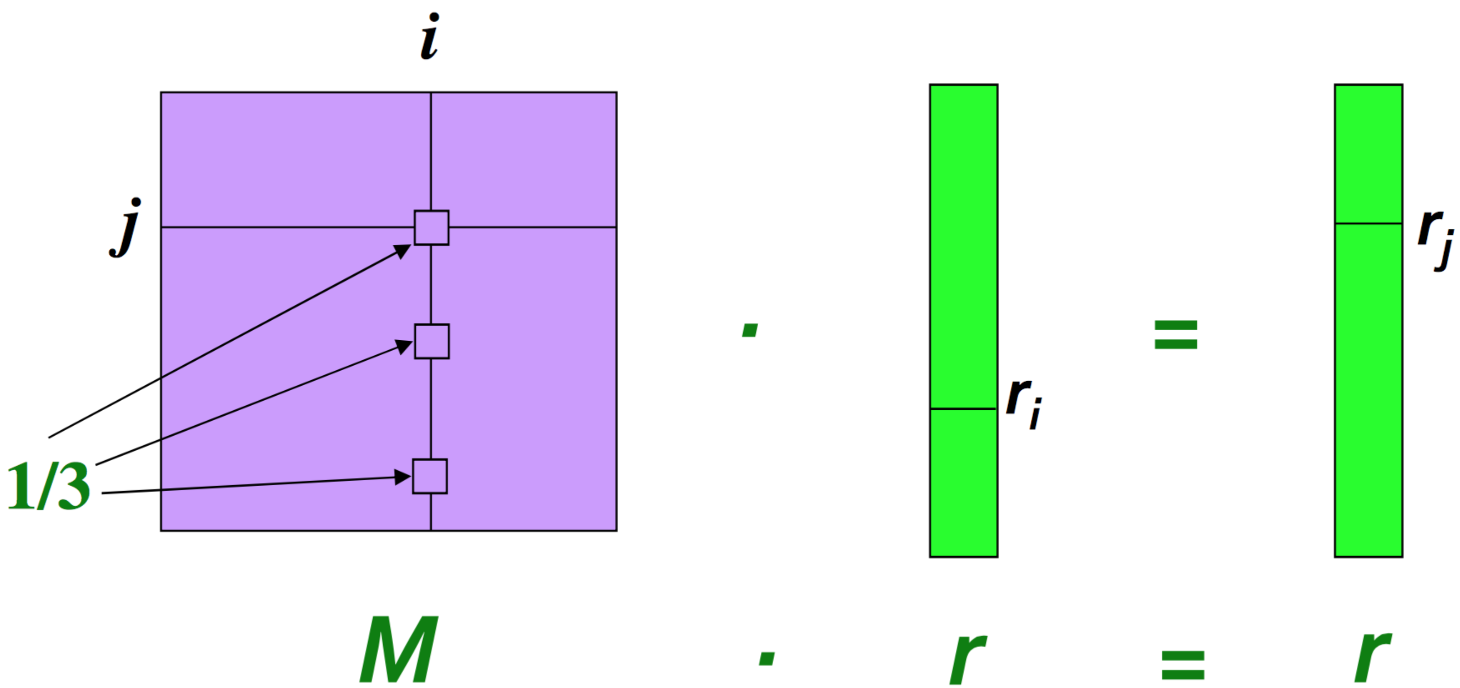
$$\boldsymbol{r} = \boldsymbol{M} \cdot \boldsymbol{r}$$

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

- **The flow equations can be written**

$$r = M \cdot r$$

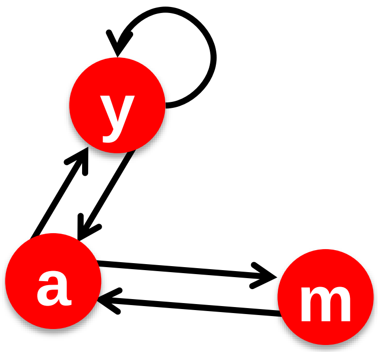
- So the **rank vector** ***r*** is an **eigenvector** of the stochastic web matrix ***M***
  - In fact, its first or principal eigenvector, with corresponding eigenvalue ***1***
    - Largest eigenvalue of ***M*** is **1** since ***M*** is column stochastic (with non-negative entries)
      - *We know* ***r*** *is unit length and each column of* ***M*** *sums to one, so* $\boldsymbol{Mr} \leq \mathbf{1}$

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

$$\boldsymbol{Ax} = \boldsymbol{\lambda x}$$

- **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

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

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

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

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 $|\mathbf{r}^{(t+1)} - \mathbf{r}^{(t)}|_1 < \varepsilon$

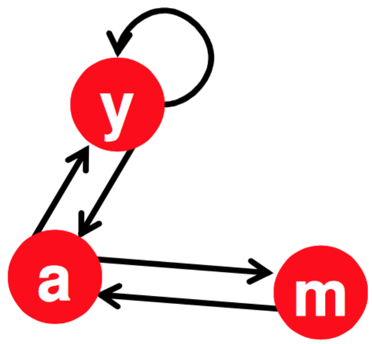
$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{\mathrm{d}_\mathrm{i}}$$

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:** $r = r'$
  - Goto **1**
- **Example:**



	y	a	m
y	½	½	0
a	½	0	1
m	0	½	0

$r_y = r_y/2 + r_a/2$

$r_a = r_y/2 + r_m$

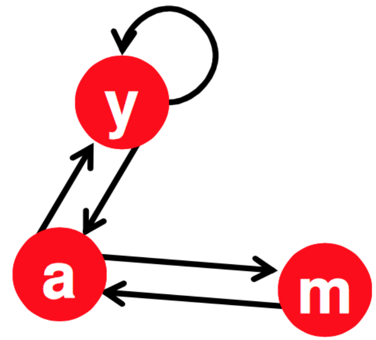
$r_m = r_a/2$

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

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

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

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

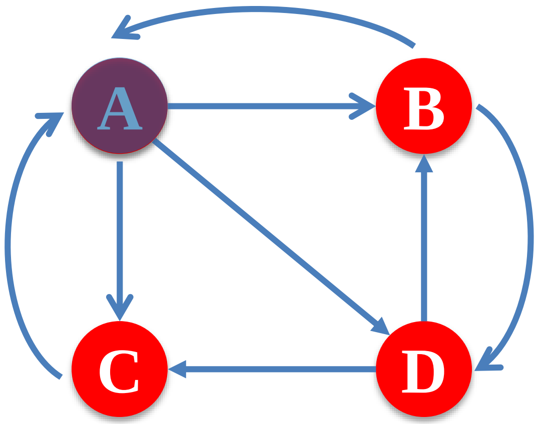
Iteration 0, 1, 2, …

PageRank:
Random Walk Interpretation

Random Walk Interpretation of PageRank

- Consider a web surfer:
  - He starts at a random page
  - He follows a random link at every time step
  - After a sufficiently long time:
    - What is the probability that he is at page j?
      - This probability corresponds to the page rank of j.

Example: Random Walk

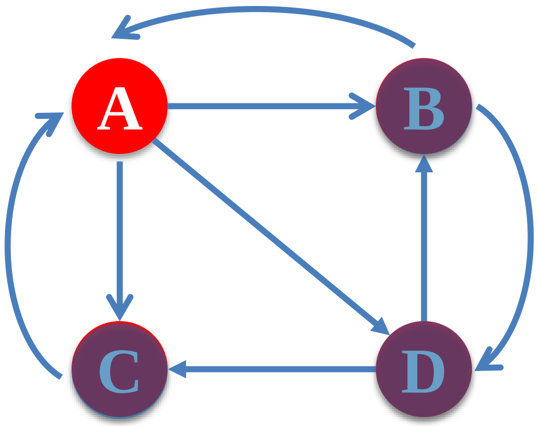


Time t = 0: Assume the random surfer is at A.

Time t = 1:

p(A, 1) = ? 0
p(B, 1) = ? 1/3
p(C, 1) = ? 1/3
p(D, 1) = ? 1/3

Example: Random Walk



Time t = 1:

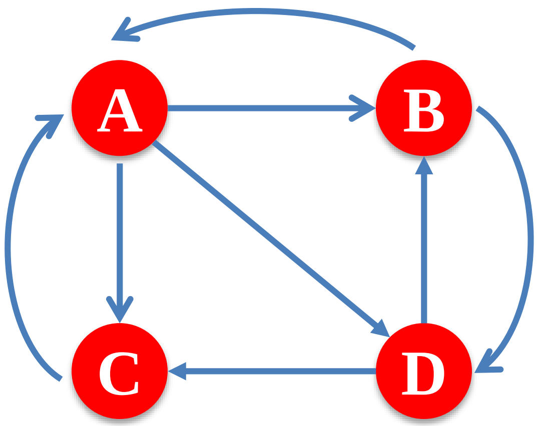
p(B, 1) = 1/3
p(C, 1) = 1/3
p(D, 1) = 1/3

Time t=2:

p(A, 2) = ?

$$p(A, 2) = p(B, 1) \,.\, p(B \to A) + p(C, 1) \,.\, p(C \to A)$$
$$= 1/3 \,.\, 1/2 + 1/3 \,.\, 1 = 3/6$$

Example: Transition Matrix



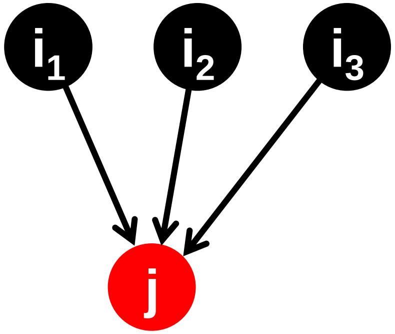
$$\begin{array}{ccccc} \mathbf{M} & & \mathbf{p(t)} & & \mathbf{p(t+1)} \\ \begin{bmatrix} 0 & 1/2 & 1 & 0 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{bmatrix} & \cdot & \begin{bmatrix} p_A \\ p_B \\ p_C \\ p_D \end{bmatrix} & = & \begin{bmatrix} p_A \\ p_B \\ p_C \\ p_D \end{bmatrix} \end{array}$$

$p(A, t+1) = p(B, t) \,.\, p(B \rightarrow A) + p(C, t) \,.\, p(C \rightarrow A)$

$p(C, t+1) = p(A, t) \,.\, p(A \rightarrow C) + p(D, t) \,.\, p(D \rightarrow C)$

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



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

PageRank:
The Google Formulation

PageRank: Three Questions

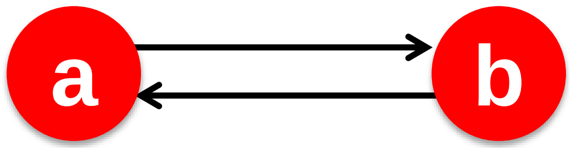
$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{\mathrm{d_i}}$$

or equivalently

$$r = Mr$$

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

Does this converge?



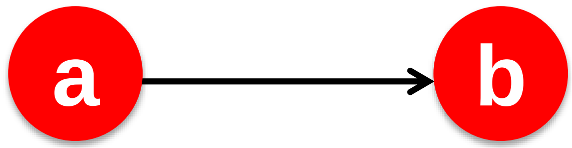
$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{\mathrm{d}_i}$$

- **Example:**

r_a	=	1	0	1	0
r_b		0	1	0	1

Iteration 0, 1, 2, …

Does it converge to what we want?



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

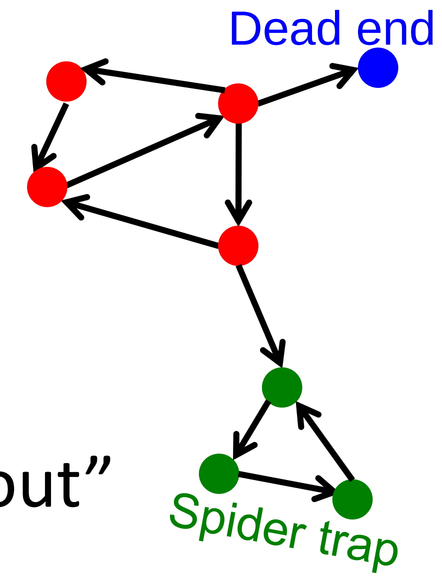
- **Example:**

r_a	=	1	0	0	0
r_b		0	1	0	0

Iteration 0, 1, 2, …

PageRank: Problems

2 problems:

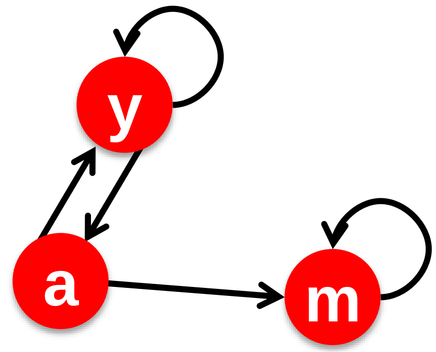


- **(1)** Some pages are **dead ends** (have no out-links)
  - Random walk has "nowhere" to go to
  - Such pages cause importance to "leak out"

- **(2)** **Spider traps:** (all out-links are within the group)
  - Random walk gets "stuck" in a trap
  - And eventually spider traps absorb all importance

Problem: Spider Traps

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



m is a spider trap

	y	a	m
y	½	½	0
a	½	0	0
m	0	½	1

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

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

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

- **Example:**

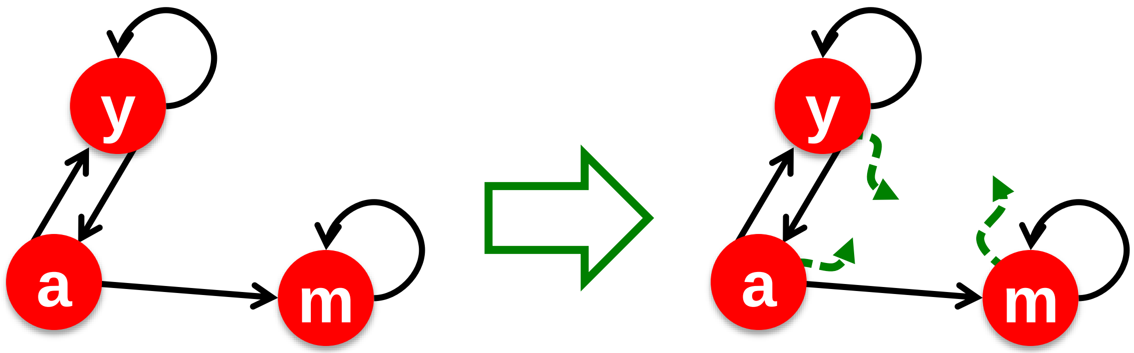
$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{array}{cccccc} 1/3 & 2/6 & 3/12 & 5/24 & & 0 \\ 1/3 & 1/6 & 2/12 & 3/24 & \ldots & 0 \\ 1/3 & 3/6 & 7/12 & 16/24 & & 1 \end{array}$$

Iteration 0, 1, 2, …

All the PageRank score gets “trapped” in node m.

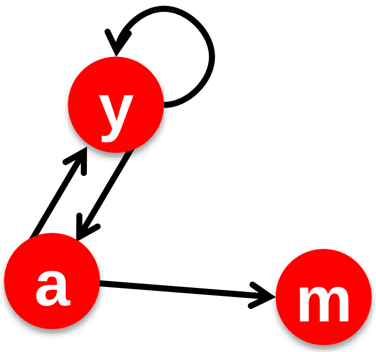
Solution: Teleports!

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

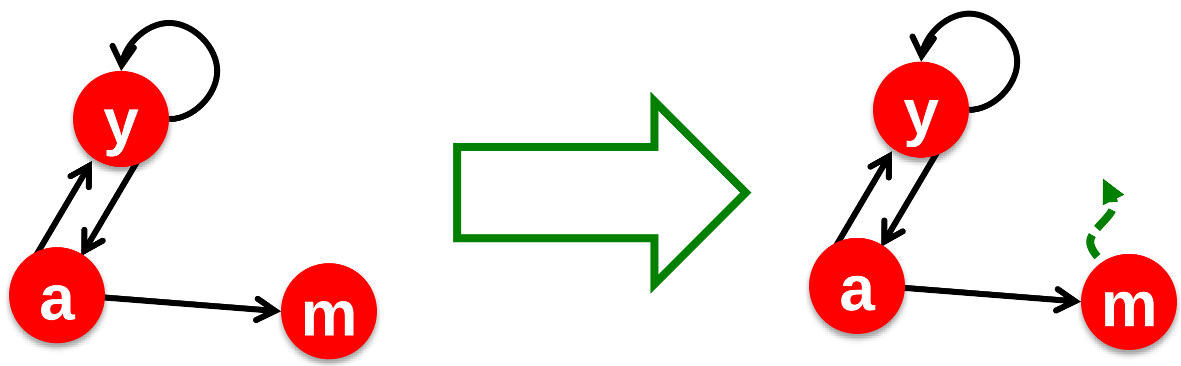
$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{array}{ccccccc} 1/3 & 2/6 & 3/12 & 5/24 & & 0 \\ 1/3 & 1/6 & 2/12 & 3/24 & \ldots & 0 \\ 1/3 & 1/6 & 1/12 & 2/24 & & 0 \end{array}$$

Iteration 0, 1, 2, …

Here the PageRank “leaks” out since the matrix is not stochastic

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

	y	a	m
y	½	½	⅓
a	½	0	⅓
m	0	½	⅓

Why Teleports Solve the Problem?

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

- **Spider-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
  - The matrix is not column stochastic so our initial assumptions are not met
  - **Solution:** Make matrix column stochastic by always teleporting when there is nowhere else to go

Solution: Random Teleports

- <u>**Google's solution that does it all:**</u>
  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]

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

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 or explicitly follow random teleport links with probability 1.0 from dead-ends.

The Google Matrix

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

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

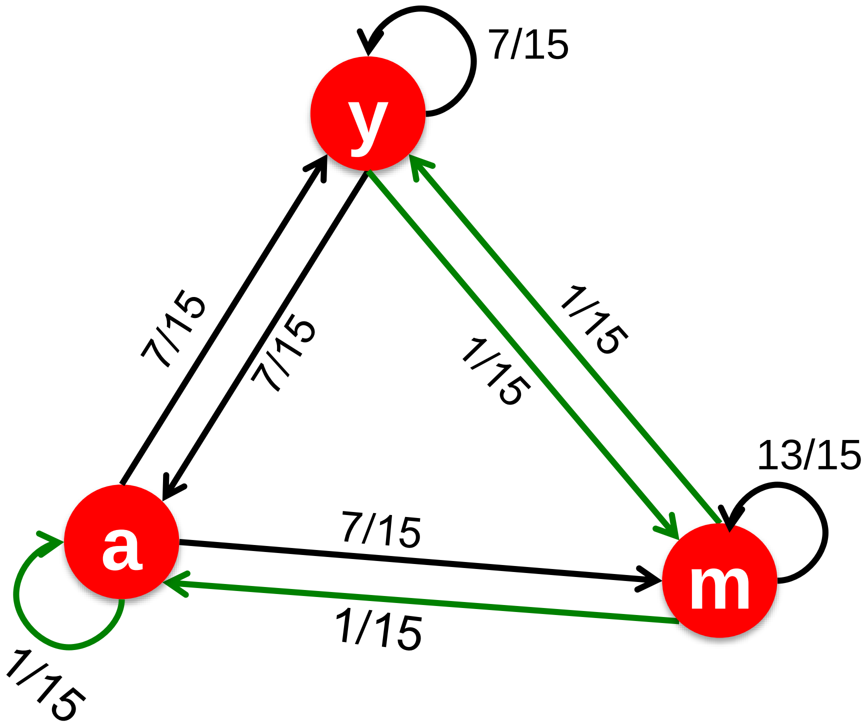
- **The Google Matrix *A*:**

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

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

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

Random Teleports ($\beta = 0.8$)



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

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

y		1/3	0.33	0.24	0.26		7/33
a	=	1/3	0.20	0.20	0.18	. . .	5/33
m		1/3	0.46	0.52	0.56		21/33

Matrix Formulation

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

How do we actually compute the PageRank?

Computing Page Rank

- **Key step is matrix-vector multiplication**
  - $\boldsymbol{r}^{\text{new}} = \boldsymbol{A} \cdot \boldsymbol{r}^{\text{old}}$
- Easy if we have enough main memory to hold $\mathbf{A}$, $\mathbf{r}^{\text{old}}$, $\mathbf{r}^{\text{new}}$
- **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!

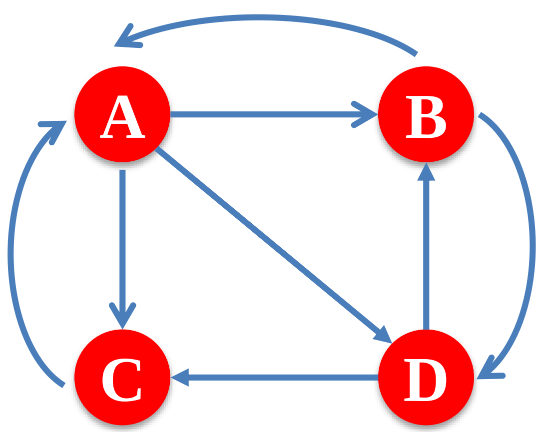
$$\mathbf{A} = \beta \cdot \mathbf{M} + (1-\beta)\,[1/N]_{N\times N}$$

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

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

Matrix Sparseness

- Reminder: Our original matrix was sparse.
  - On average: ~10 out-links per vertex
  - # of non-zero values in matrix M: ~10N
- **Teleport links make matrix M dense**.
- Can we convert it back to the sparse form?



Original matrix without teleports

0	1/2	1	0
1/3	0	0	1/2
1/3	0	0	1/2
1/3	1/2	0	0

Rearranging the Equation

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

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

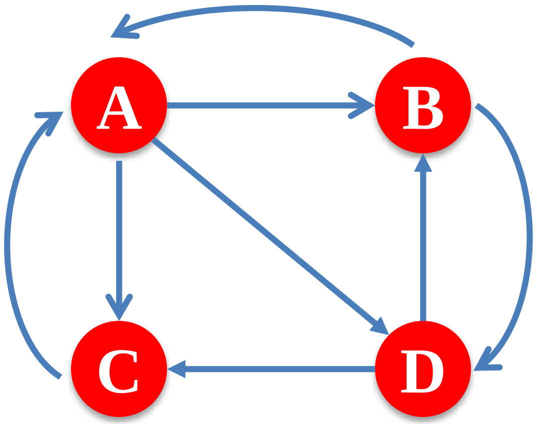
$$= \sum_{\mathrm{i}=1}^{N} \beta\, M_{ji} \cdot r_i + \frac{1-\beta}{N} \qquad \text{since } \sum r_i = 1$$

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

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

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

Example: Equation with Teleports



$$\overset{\mathbf{r}^{\mathbf{new}}}{\begin{bmatrix} r_A \\ r_B \\ r_C \\ r_D \end{bmatrix}} = \beta \overset{\mathbf{M}}{\begin{bmatrix} 0 & 1/2 & 1 & 0 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{bmatrix}} \cdot \overset{\mathbf{r}^{\mathbf{old}}}{\begin{bmatrix} r_A \\ r_B \\ r_C \\ r_D \end{bmatrix}} + (1-\beta) \begin{bmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{bmatrix}$$

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

Sparse Matrix Formulation

- We just rearranged the **PageRank equation**

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

  - where $\mathbf{[(1-\beta)/N]_N}$ is a vector with all ***N*** entries $\mathbf{(1-\beta)/N}$

- ***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 $\mathbf{(1-\beta)/N}$ to each entry in $\boldsymbol{r}^{\text{new}}$
    - **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**

PageRank: Without Dead Ends

- **Input:** **Graph G and parameter β**
  - Directed graph $\boldsymbol{G}$ (cannot have **dead ends**)
  - Parameter $\boldsymbol{\beta}$
- **Output:** **PageRank vector $\boldsymbol{r}^{new}$**
  - **Set:** $r_j^{old} = \frac{1}{N}$
  - **repeat until convergence:** $\sum_j \left| r_j^{new} - r_j^{old} \right| > \varepsilon$
    - $\forall j$: $\boldsymbol{r_j^{new}} = \sum_{i \to j} \boldsymbol{\beta} \, \frac{\boldsymbol{r_i^{old}}}{\boldsymbol{d_i}}$

      $\boldsymbol{r_j^{new}} = \mathbf{0}$ if in-degree of $\boldsymbol{j}$ is **0**
    - **Add constant terms:**

      $\forall \boldsymbol{j}$: $\boldsymbol{r_j^{new}} = \boldsymbol{r_j^{new}} + \frac{\mathbf{1}-\boldsymbol{\beta}}{\boldsymbol{N}}$
    - $\boldsymbol{r^{old}} = \boldsymbol{r^{new}}$

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:** $\sum_j \left| r_j^{new} - r_j^{old} \right| > \varepsilon$
    - $\forall j$: $r'^{new}_j = \sum_{i \to j} \beta \frac{r_i^{old}}{d_i}$

      $r'^{new}_j = 0$ if in-degree of j is $\mathbf{0}$
    - **Now re-insert the leaked PageRank:**

      $\forall j$: $r_j^{new} = r'^{new}_j + \frac{1-S}{N}$ **where:** $S = \sum_j r'^{new}_j$
    - $r^{old} = r^{new}$

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**.

Some Problems with Page Rank

- **Measures generic popularity of a page**
  - Biased against topic-specific authorities
  - **Solution:** Topic-Specific PageRank (**next**)
- **Susceptible to Link spam**
  - Artificial link topographies created in order to boost page rank
  - **Solution:** TrustRank
- **Uses a single measure of importance**
  - Other models of importance
  - **Solution:** Hubs-and-Authorities