Mining Massive Datasets

Lecture 05

Artur Andrzejak
http://pvs.ifi.uni-heidelberg.de

RUPRECHT-KARLS-
UNIVERSITÄT
HEIDELBERG

Note on Slides

A substantial part of these slides come (either verbatim or in a modified form) from the book Mining of Massive Datasets
by Jure Leskovec, Anand Rajaraman, Jeff Ullman (Stanford University).
For more information, see the website accompanying the book: http://www.mmds.org.

Recommender Systems

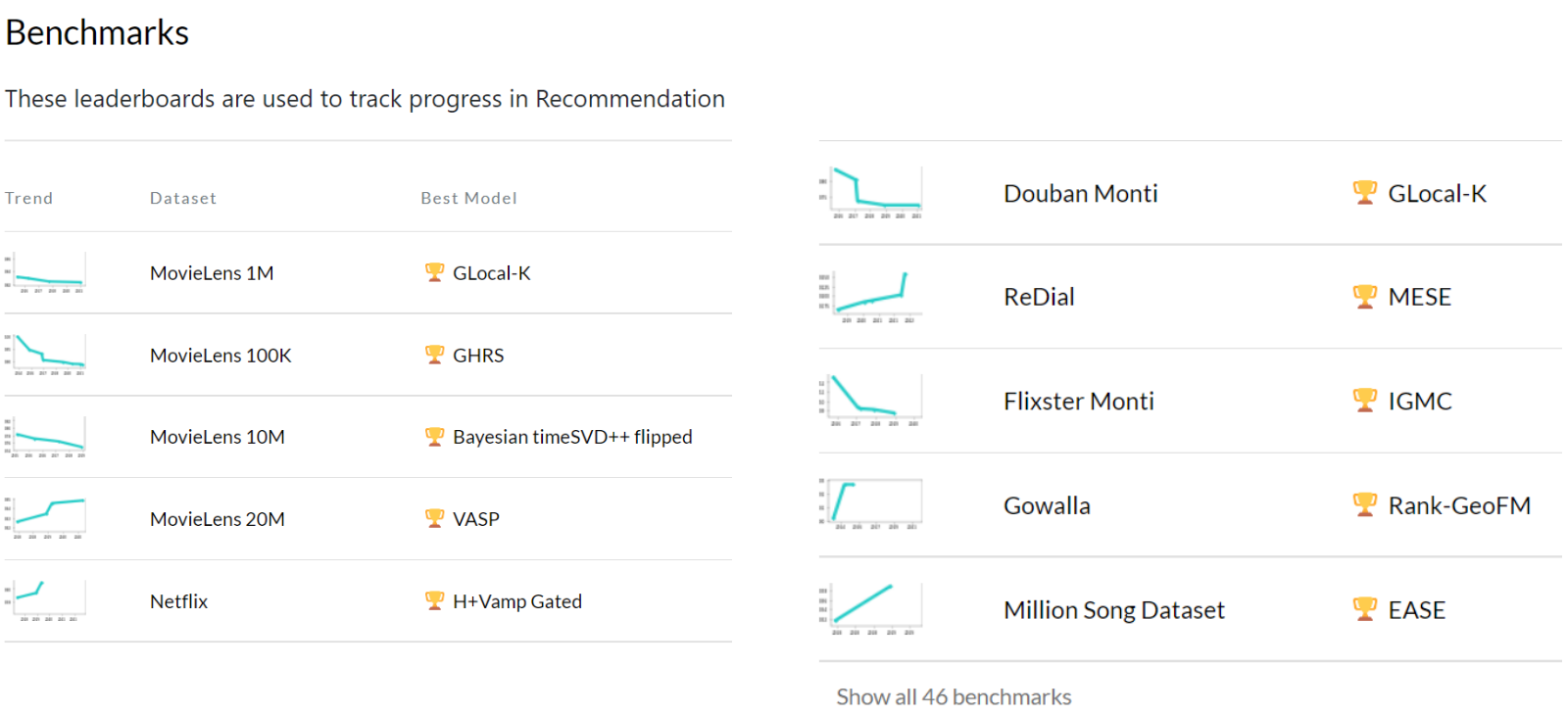
Appendum

Can you Trust Netflix?

- Case “Netflix user Jane Doe against Netflix“, 2009
- Netflix wanted to expand its practice of evaluating user data for individualized offers based on profiles
- => Netflix Challenge:
  - Competition in which the user data of approximately from 500,000 users was used
  - This test data was also used for scientific purposes
- However, this data was not really anonymous!
  - People could be identified with regard to their sensitive data, such as sexual orientation, and **this became public**
- Jane Doe successfully took action against in a *class action lawsuit* to protect her children in the community
- Read the case description here:
  - https://www.wired.com/images_blogs/threatlevel/2009/12/doe-v-netflix.pdf

Latest Approaches

- Web site paperswithcode.com has a great collection of papers, code, and datasets
- For Recommendation Systems see: https://paperswithcode.com/task/recommendation-systems



Benchmarks

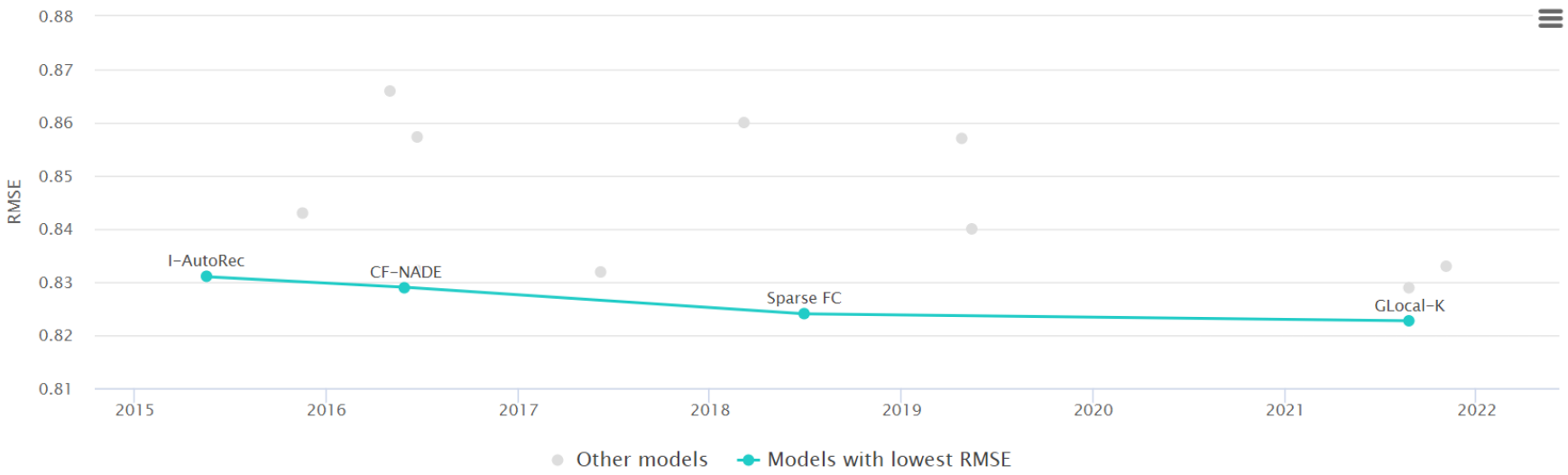
These leaderboards are used to track progress in Recommendation

Trend	Dataset	Best Model
	MovieLens 1M	GLocal-K
	MovieLens 100K	GHRS
	MovieLens 10M	Bayesian timeSVD++ flipped
	MovieLens 20M	VASP
	Netflix	H+Vamp Gated
	Douban Monti	GLocal-K
	ReDial	MESE
	Flixster Monti	IGMC
	Gowalla	Rank-GeoFM
	Million Song Dataset	EASE

Show all 46 benchmarks

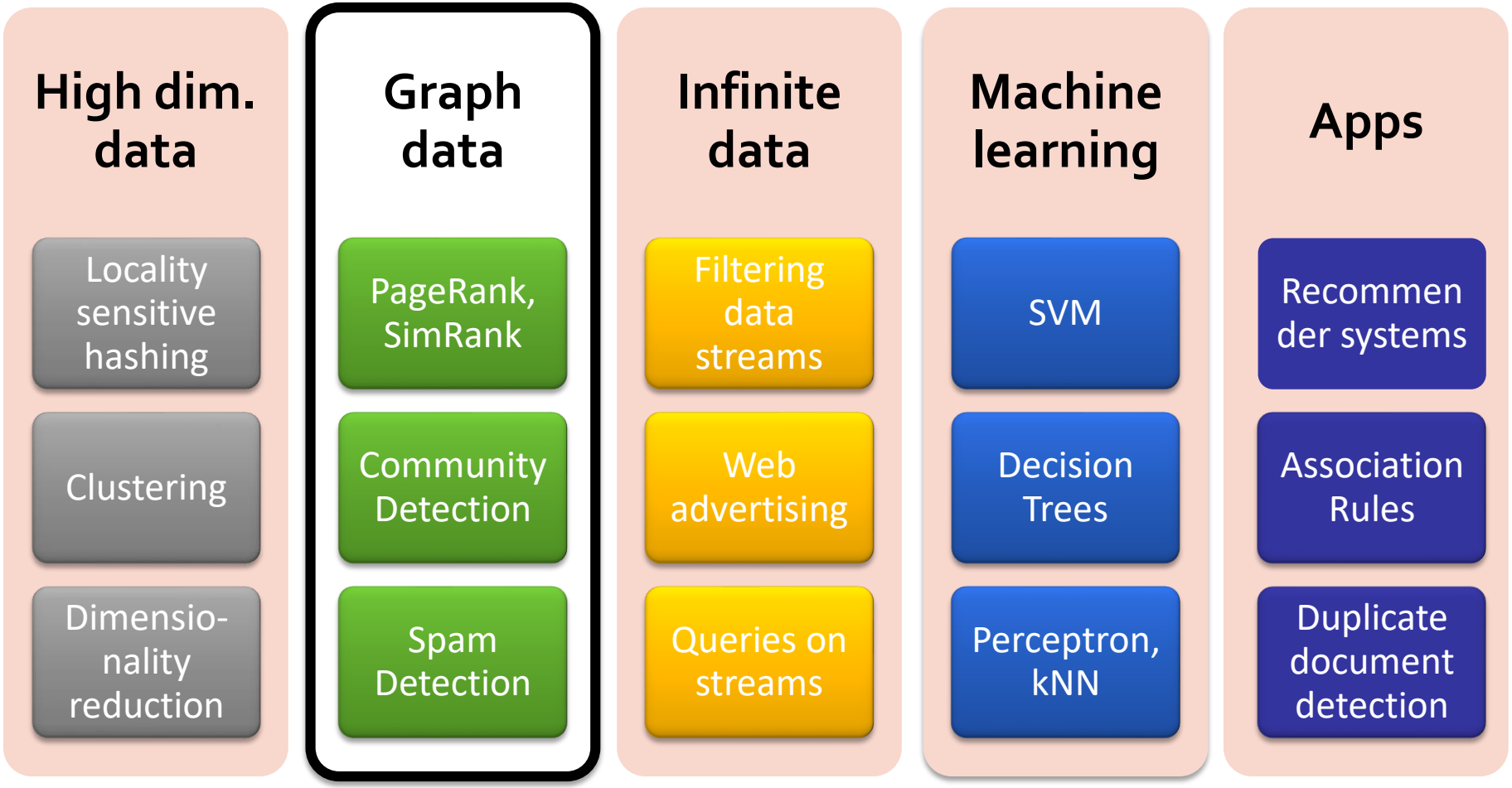
Latest Approaches /2

- You see there more datasets than Netflix:
  - MovieLens 100k,…, 20M, Douban Monti, ReDial, Gowalla, …
- … and best approaches per dataset over time
- E.g. MovieLens 1M



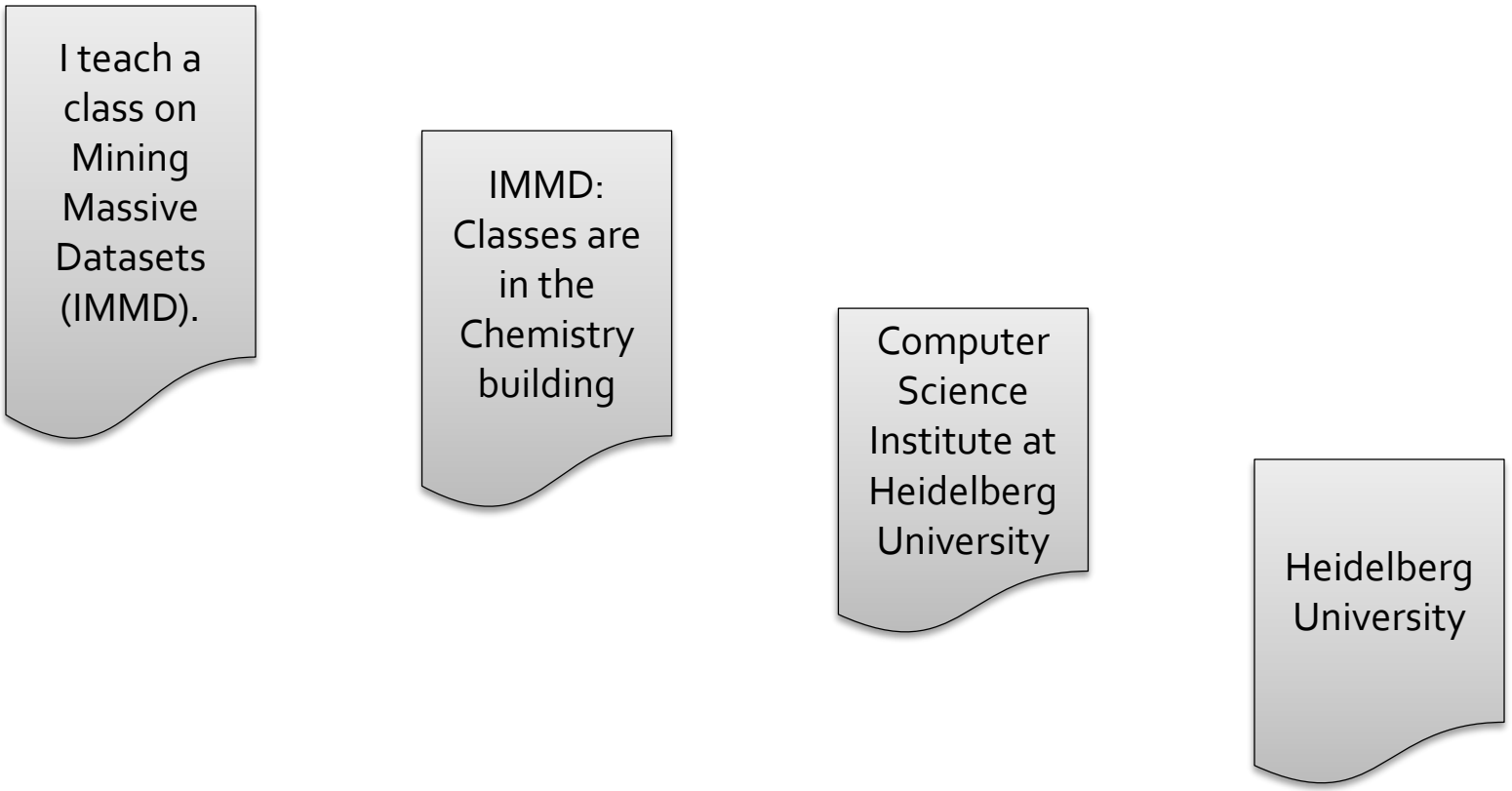
Link Analysis

Infinite Data



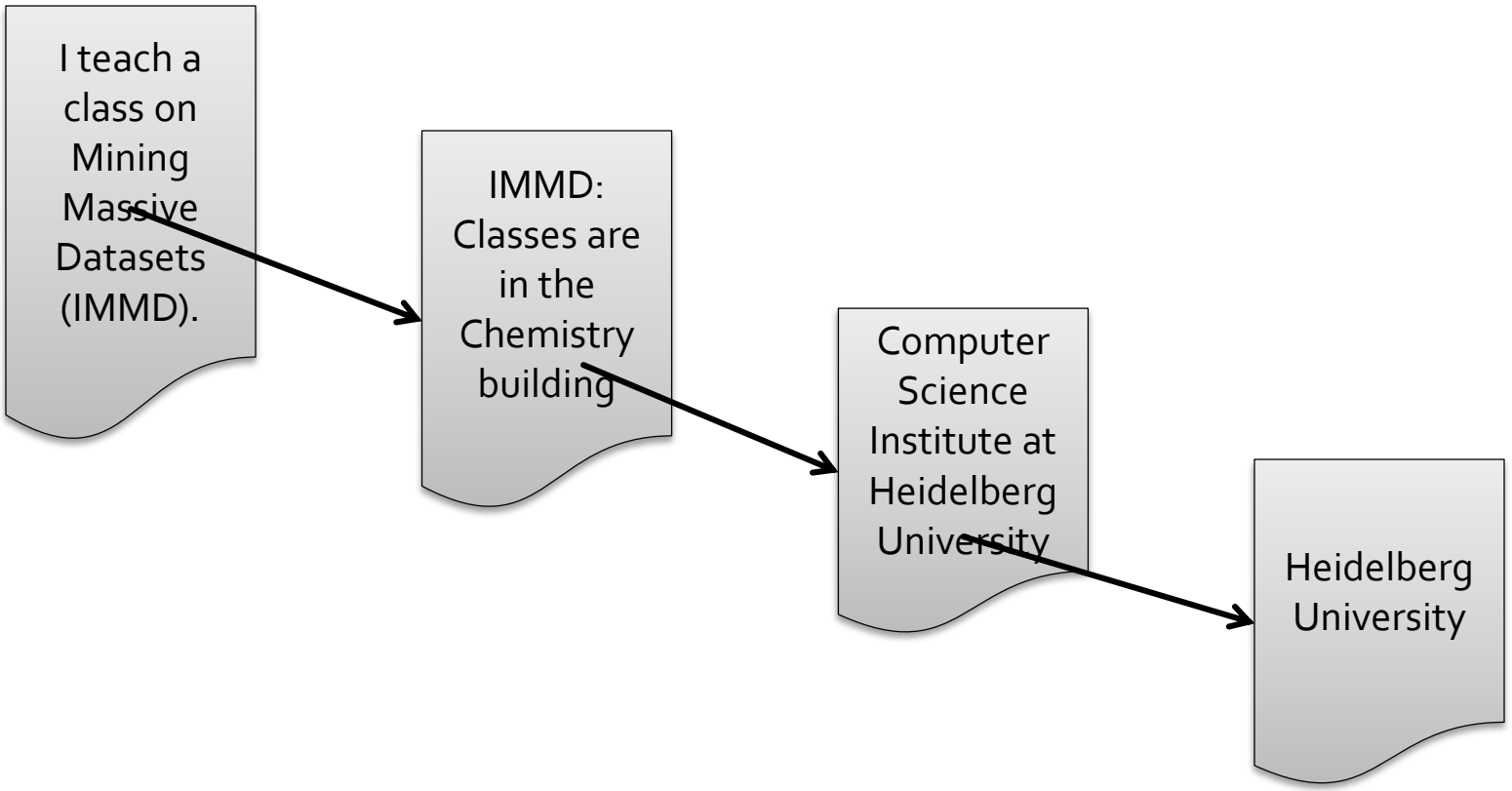
Web as a Graph

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

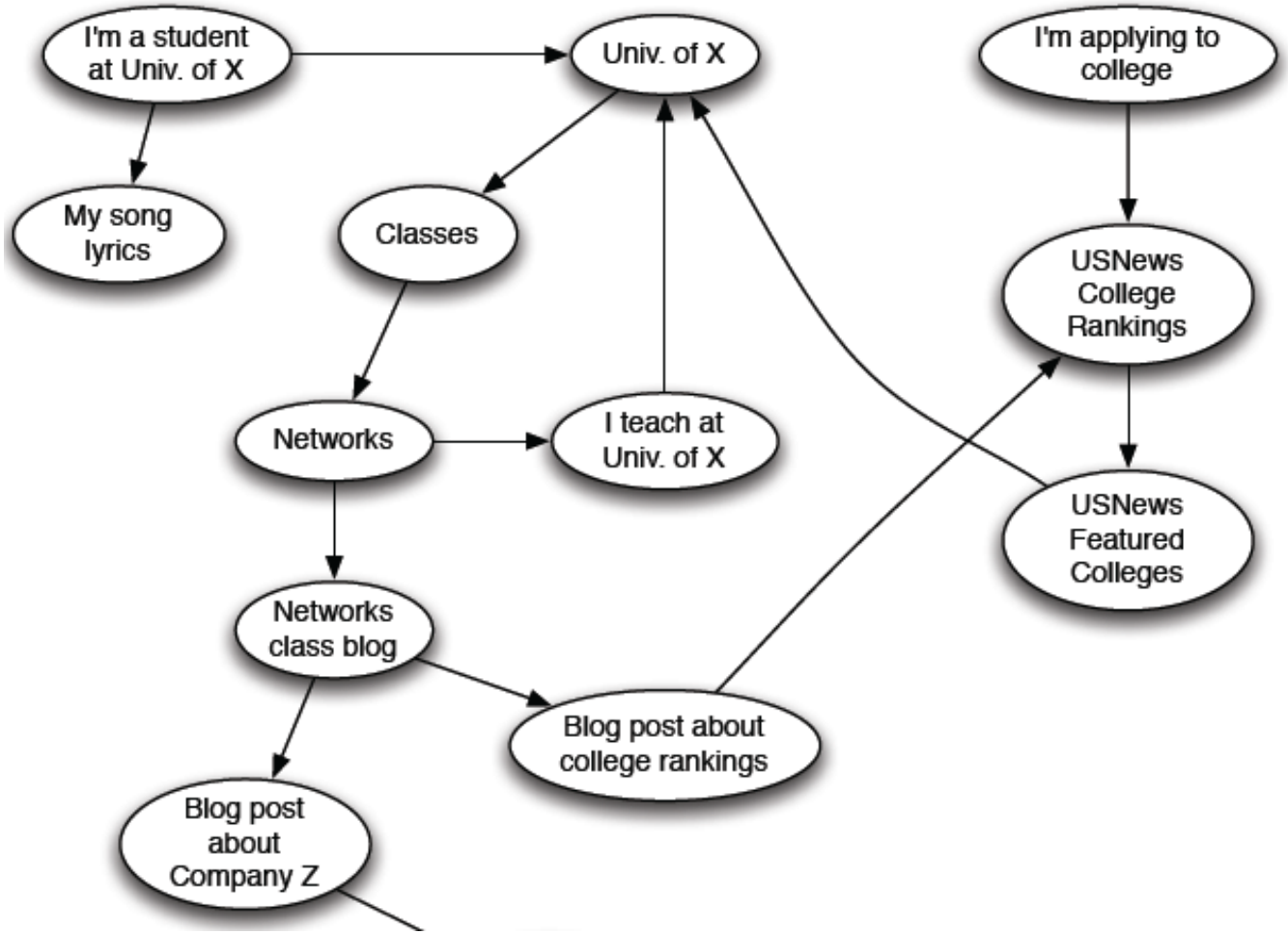


Web as a Graph

- **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

Two 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

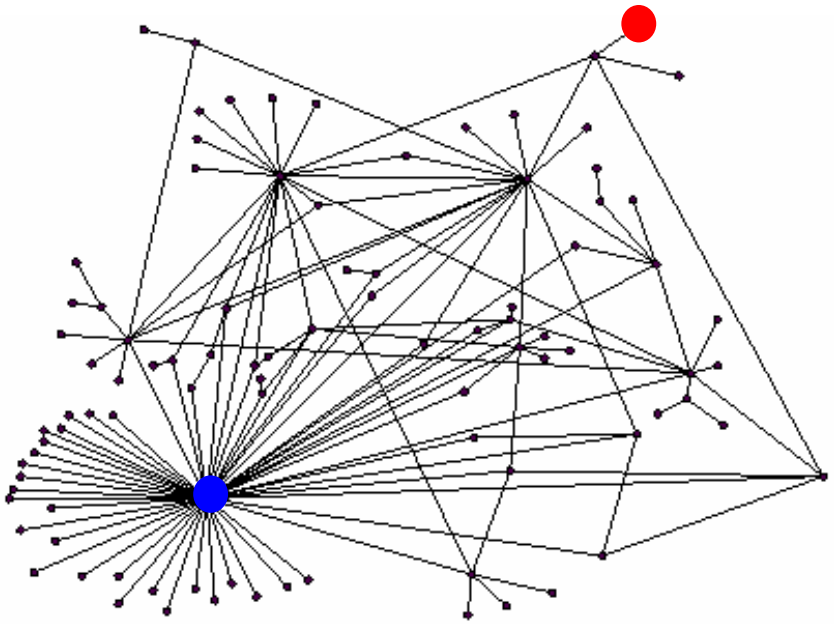
Ranking Nodes on the Graph

- All web pages are not equally “important”

  http://endless.horse/ vs. www.stanford.edu

- There is large diversity in the web-graph node connectivity

  Let’s rank the pages by the link structure!



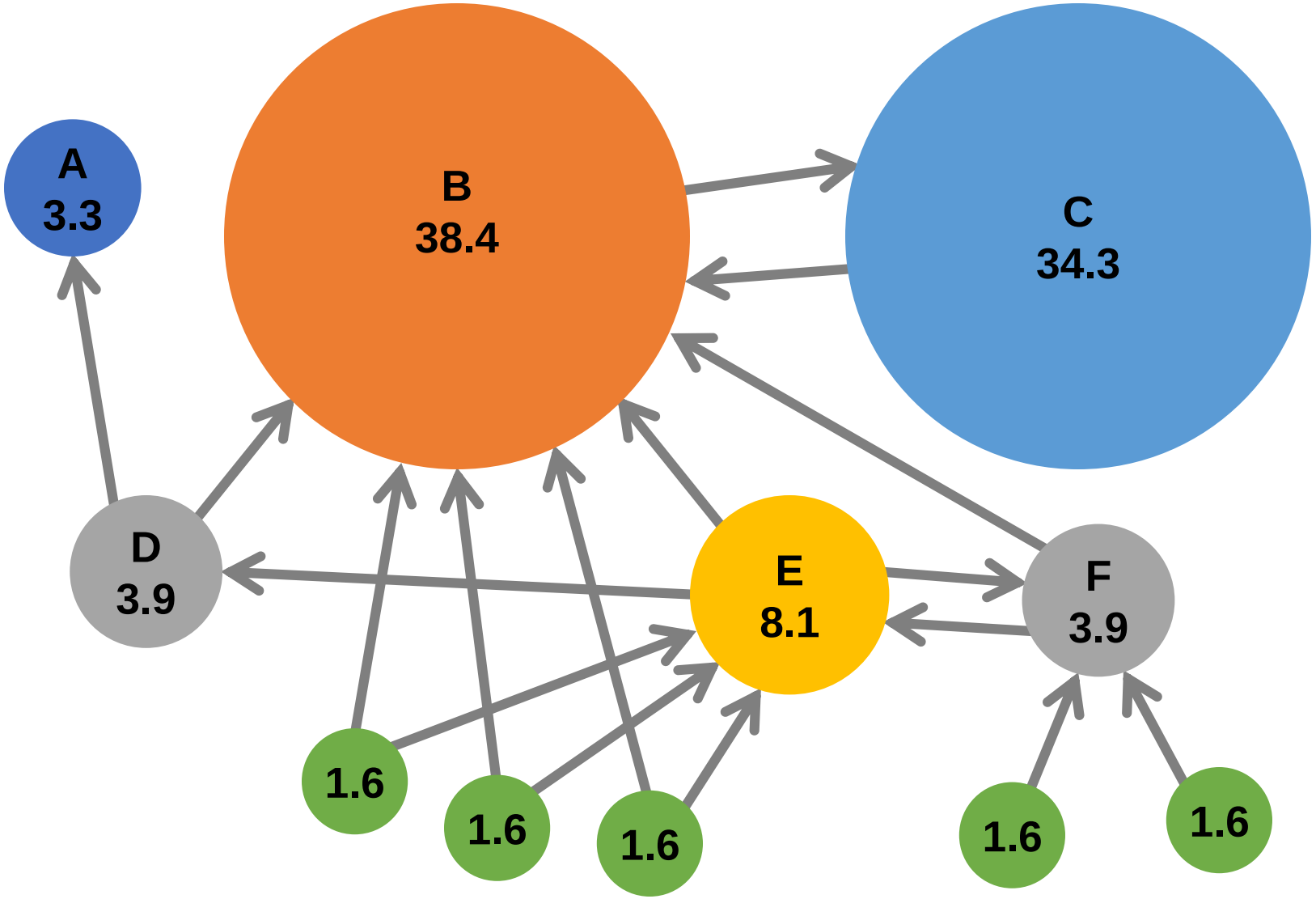
PageRank

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 100,000+ in-links
  - http://endless.horse/ has only few in-links
- Are all in-links are equal?
  - Links from important pages count more
  - Recursive question!

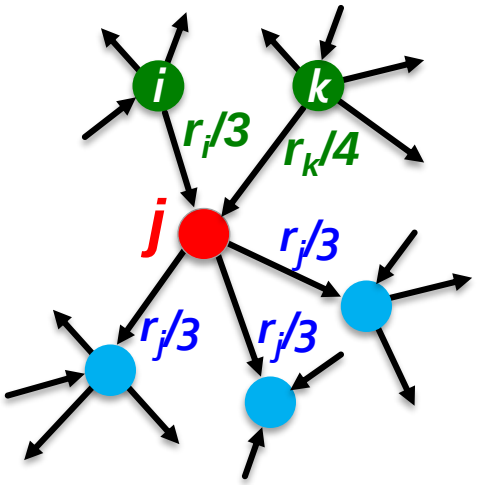
Example: PageRank Scores



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

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

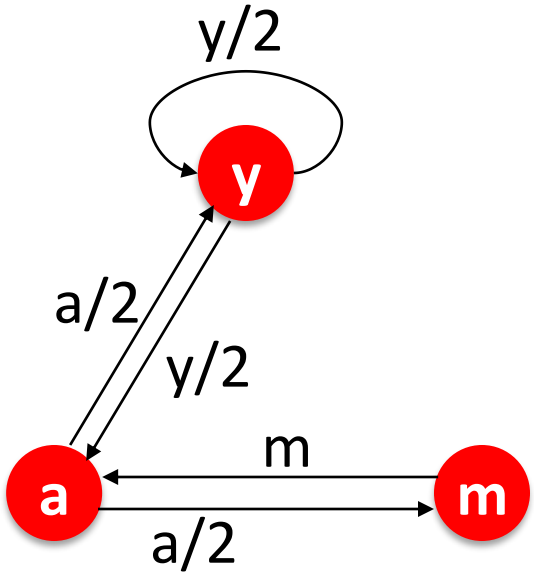


PageRank: The "Flow" Model

- 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



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 are 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 $M_{ji} = 0$
  - $\boldsymbol{M}$ is a **column stochastic matrix** (columns sum to 1)
- Rank vector r: vector with an entry per page
  - r_i is the importance score of page i
  - $\sum_i r_i = 1$

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

- The flow equations can be written as

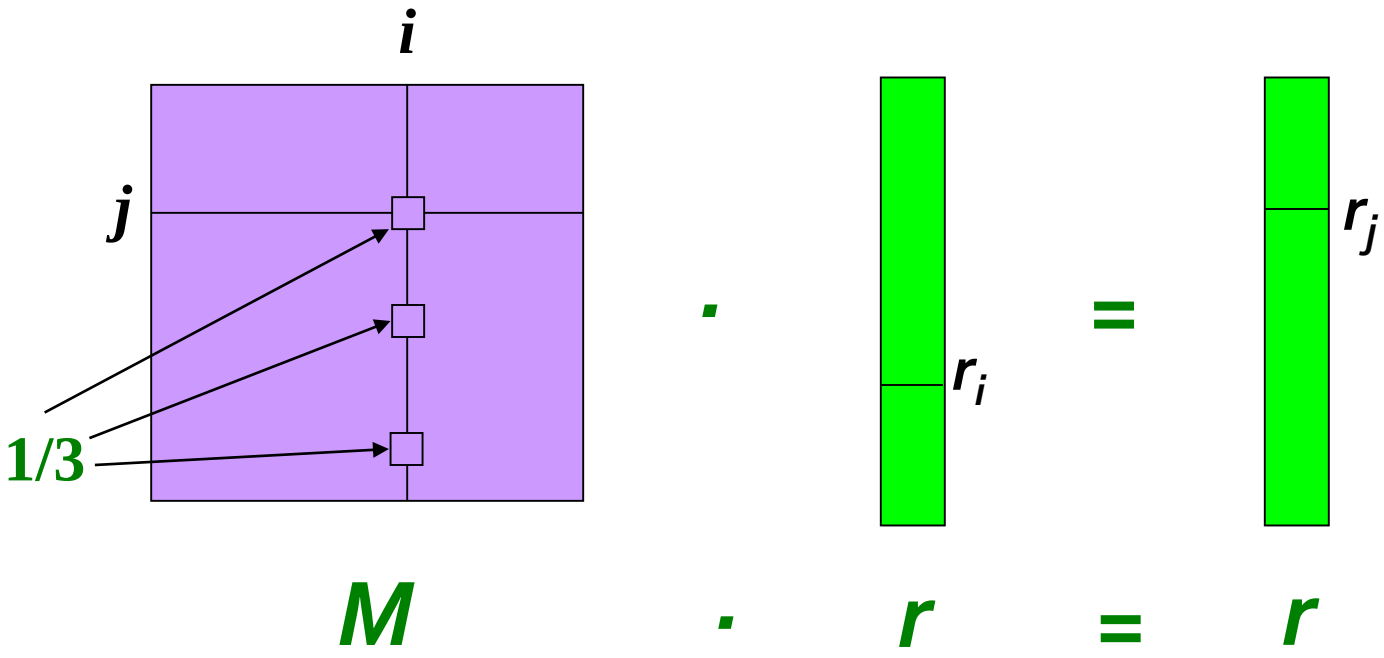
$$\boldsymbol{r = M \cdot r}$$

Example

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

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

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



Eigenvector Formulation

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

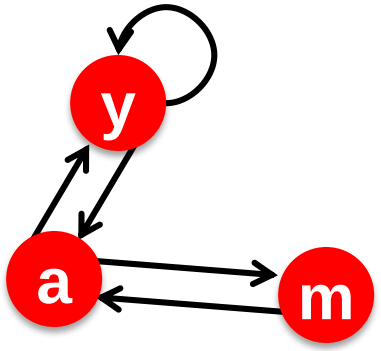
$$\boldsymbol{Ax = \lambda x}$$

- The flow equations can be written

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

- So the **rank vector** ***r*** is an **eigenvector** of the stochastic web matrix ***M***
  - In fact, it is 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* $\boldsymbol{Mr \leq 1}$
- We can now solve for *r*!
- To do this efficiently, we use **Power iteration**

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: $\mathbf{r}^{(0)} = [1/N,\ldots,1/N]^T$
  - 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

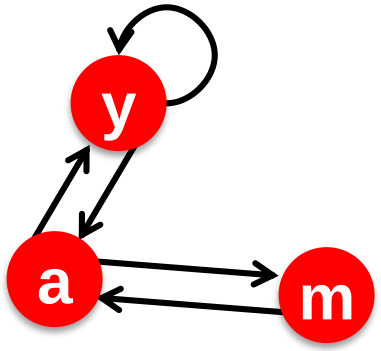
$|\mathbf{x}|_1 = \sum_{1 \le i \le N} |x_i|$ is the $\mathbf{L_1}$ norm
We can use any other vector norm, e.g., Euclidean

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:**

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

Iteration 0, 1, 2, …



	y	a	m
y	½	½	0
a	½	0	1
m	0	½	0

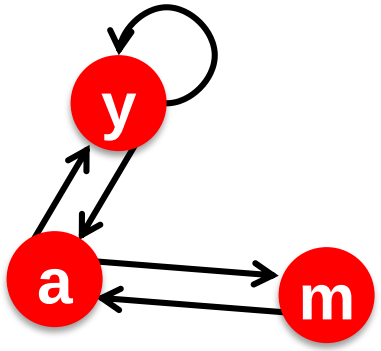
$r_y = r_y/2 + r_a/2$

$r_a = r_y/2 + r_m$

$r_m = r_a/2$

PageRank: How to solve?

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



	y	a	m
y	½	½	0
a	½	0	1
m	0	½	0

$r_y = r_y/2 + r_a/2$

$r_a = r_y/2 + r_m$

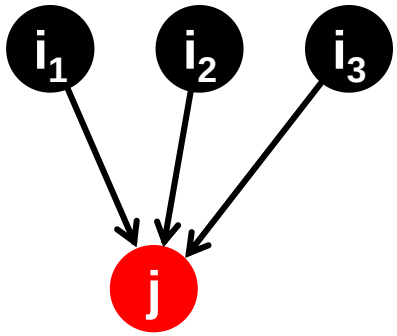
$r_m = r_a/2$

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

Iteration 0, 1, 2, …

Random Walk Interpretation

- **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



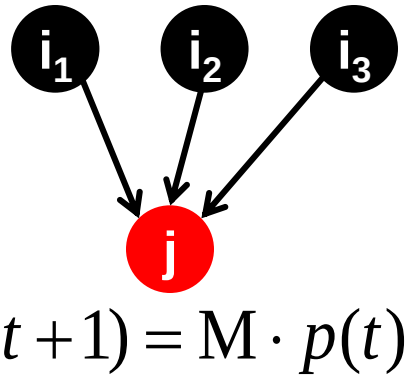
$$r_j = \sum_{i \to j} \frac{r_i}{\mathrm{d}_{\mathrm{out}}(\mathrm{i})}$$

- **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

The Stationary Distribution

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

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



- Suppose the random walk reaches a state

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

then $\boldsymbol{p(t)}$ is **stationary distribution** of a random walk

- Our original rank vector $\boldsymbol{r}$ satisfies $\boldsymbol{r = M \cdot r}$
  - So, $\boldsymbol{r}$ 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:

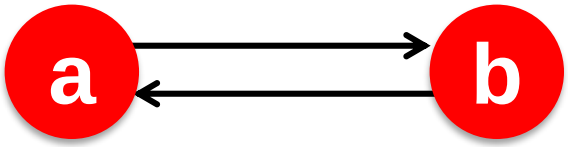
The Google Formulation

PageRank: Three Questions

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

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

Does this converge?



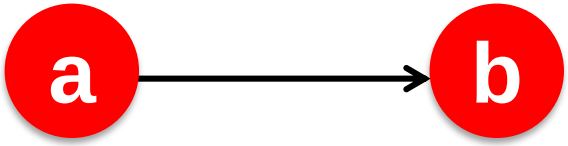
$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{\mathrm{d_i}}$$

- **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:**

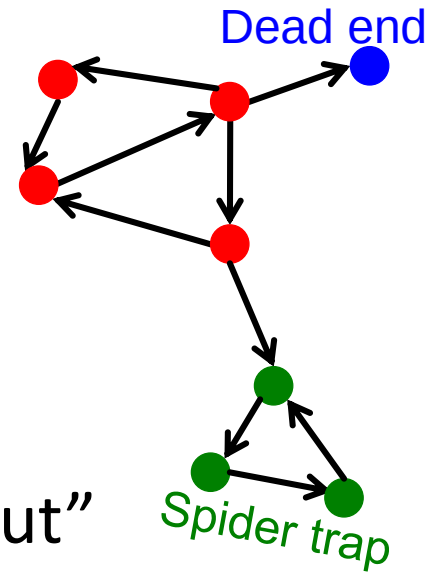
$$\begin{matrix} r_a \\ r_b \end{matrix} = \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{matrix}$$

Iteration 0, 1, 2, …

PageRank: Problems

2 problems:

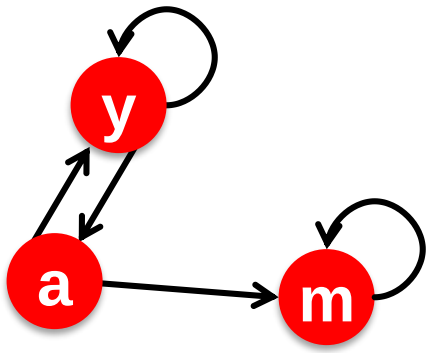
- **(1)** Some pages are **dead ends** (have no out-links)
  - Random walk has "nowhere" to go to
  - Such pages cause importance to "leak out"



- **(2) Spider traps**: (all out-links are within the group)
  - Random walked gets "stuck" in a trap
  - And 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



m is a spider trap

	y	a	m
y	½	½	0
a	½	0	0
m	0	½	1

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

$\mathbf{r_a = r_y/2}$

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

- **Example:**

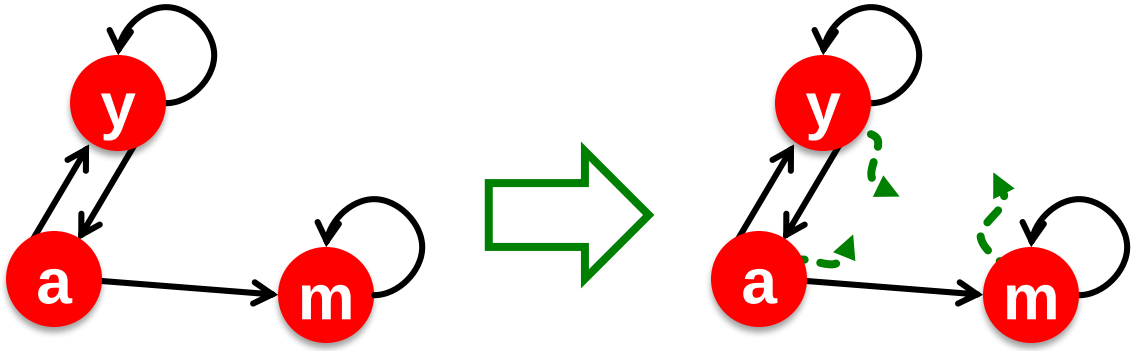
$$\begin{pmatrix} r_y \\ r_a \\ r_m \end{pmatrix} = \begin{matrix} 1/3 & 2/6 & 3/12 & 5/24 & & 0 \\ 1/3 & 1/6 & 2/12 & 3/24 & \ldots & 0 \\ 1/3 & 3/6 & 7/12 & 16/24 & & 1 \end{matrix}$$

Iteration 0, 1, 2, …

All the PageRank score gets "trapped" in node m

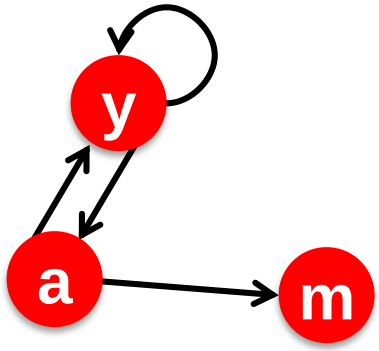
Solution: Teleports!

- **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. $1 - \beta$, 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:

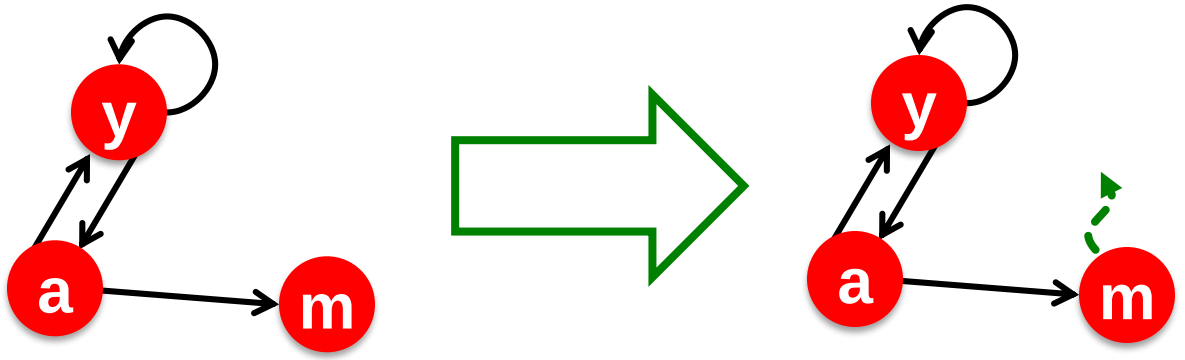
$$\begin{pmatrix} r_y \\ r_a \\ r_m \end{pmatrix} = \begin{matrix} 1/3 & 2/6 & 3/12 & 5/24 & & 0 \\ 1/3 & 1/6 & 2/12 & 3/24 & \ldots & 0 \\ 1/3 & 1/6 & 1/12 & 2/24 & & 0 \end{matrix}$$

Iteration 0, 1, 2, …

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

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?

- **Spider-traps** are not a problem, 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 "formal" 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

- **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 $1-\beta$, jump to a 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

This formulation assumes that $\boldsymbol{M}$ has no dead ends. We can either **preprocess matrix $\boldsymbol{M}$ 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]

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

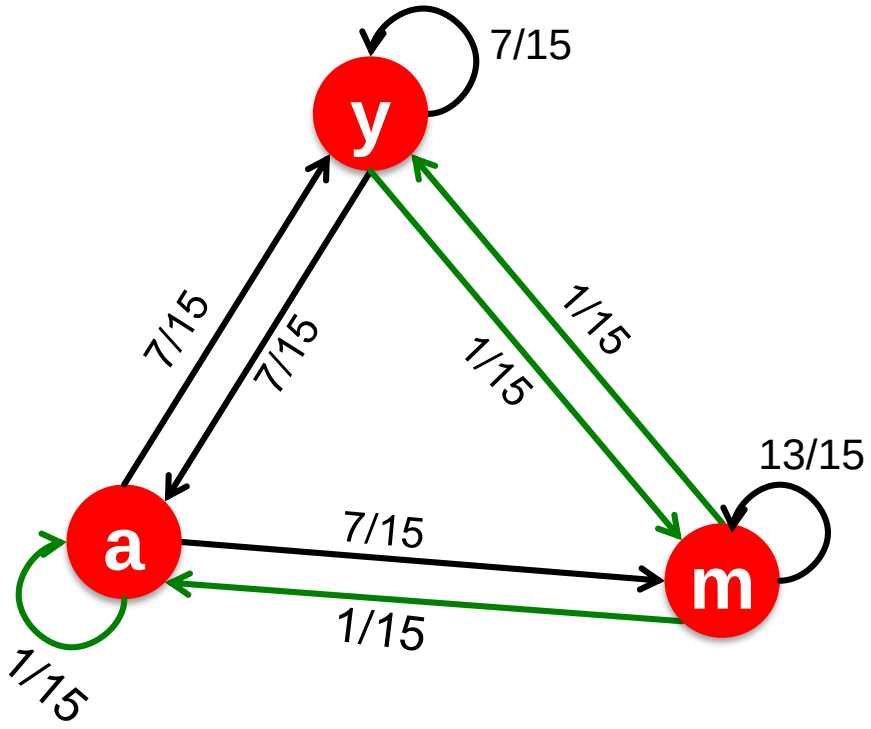
- The Google Matrix A:

$[1/N]_{NxN}$…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: $r = A \cdot r$
  And the Power method still works!
- What is β?
  - In practice $\beta = 0.8, 0.9$ (make 5 steps on avg., jump)

Random Teleports ($\beta = 0.8$)



M

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

$[1/N]_{NxN}$

A

y	7/15	7/15	1/15
a	7/15	1/15	1/15
m	1/15	7/15	13/15

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

PageRank

Why Power Iteration works?

Why Power Iteration works? (1)

Details!

- **Power iteration:**
  A method for finding dominant eigenvector (the vector corresponding to the largest eigenvalue)
  - $\boldsymbol{r^{(1)} = M \cdot r^{(0)}}$
  - $\boldsymbol{r^{(2)} = M \cdot r^{(1)} = M\left(Mr^{(1)}\right) = M^2 \cdot r^{(0)}}$
  - $\boldsymbol{r^{(3)} = M \cdot r^{(2)} = M\left(M^2 r^{(0)}\right) = M^3 \cdot r^{(0)}}$
- **Claim:**
  Sequence $\boldsymbol{M \cdot r^{(0)}, M^2 \cdot r^{(0)}, \ldots M^k \cdot r^{(0)}, \ldots}$
  approaches the dominant eigenvector of $\boldsymbol{M}$

Why Power Iteration works? (2)

Details!

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

Why Power Iteration works? (3)

Details!

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

Thank you.

Questions?

Additional Slides