COMP4222 Machine Learning with Structured Data

PageRank

Yangqiu Song

Slides credits: Jeffrey D. Ullman and Jure Leskovec @Stanford

PageRank

- The year 1998 was an eventful year for Web link analysis models. Both the PageRank and HITS algorithms were reported in that year.
- The connections between PageRank and HITS are quite striking.
- Since that eventful year, PageRank has emerged as the dominant link analysis model,
  - due to its query-independence,
  - its ability to combat spamming, and
  - Google’s huge business success.

Ranking web pages

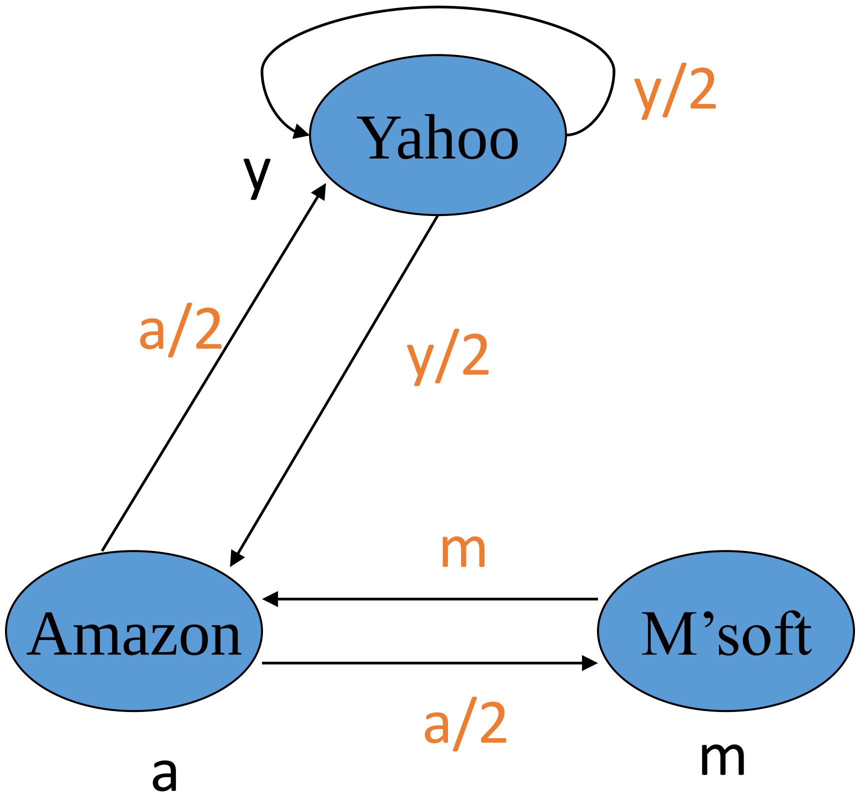
- Web pages are not equally "important"
  - https://www.facebook.com/ vs https://www.ust.hk/
- Inlinks as votes
  - https://www.facebook.com/ 668.7M backlinks from 609.2K domains
  - https://www.ust.hk/ 7.1K backlinks from 270 domains
- Are all inlinks equal?
  - Recursive question!

Simple recursive formulation

- Each link's vote is proportional to the importance of its source page
- If page P with importance x has n outlinks, each link gets x/n votes
- Page P's own importance is the sum of the votes on its inlinks

Simple "flow" model

The web in 1839



$$y = y/2 + a/2$$
$$a = y/2 + m$$
$$m = a/2$$

Solving the flow equations

- 3 equations, 3 unknowns, no constants
  - No unique solution
  - All solutions equivalent modulo scale factor

- Additional constraint forces uniqueness
  - $y+a+m = 1$
  - $y = 2/5$, $a = 2/5$, $m = 1/5$

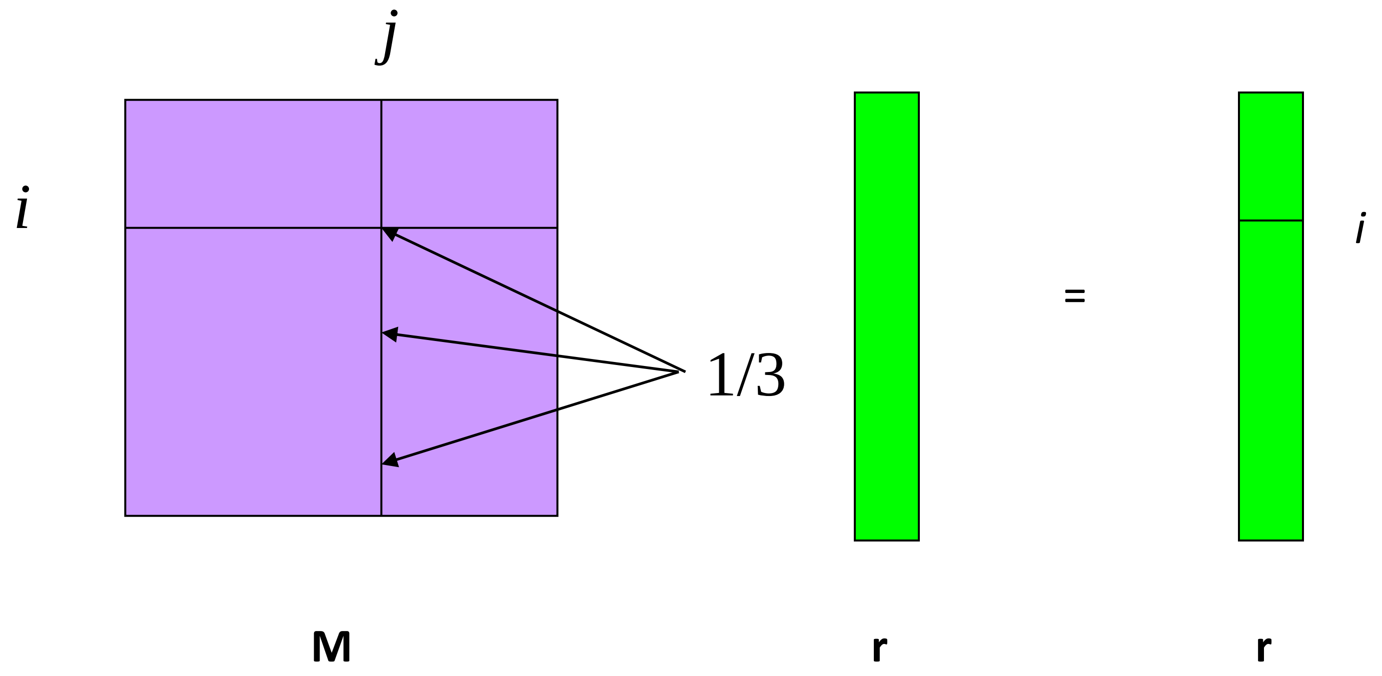
- Gaussian elimination method works for small examples, but we need a better method for large graphs

Matrix formulation

- Matrix **M** has one row and one column for each web page
- Suppose page j has n outlinks
  - If j has an outlink i , then M_{ij}=1/n
  - Else M_{ij}=0
- **M** is a column stochastic matrix
  - Columns sum to 1
- Suppose **r** is a vector with one entry per web page
  - r_j is the importance score of page j
  - Call it the rank vector
  - $|\mathbf{r}| = 1$

Example

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



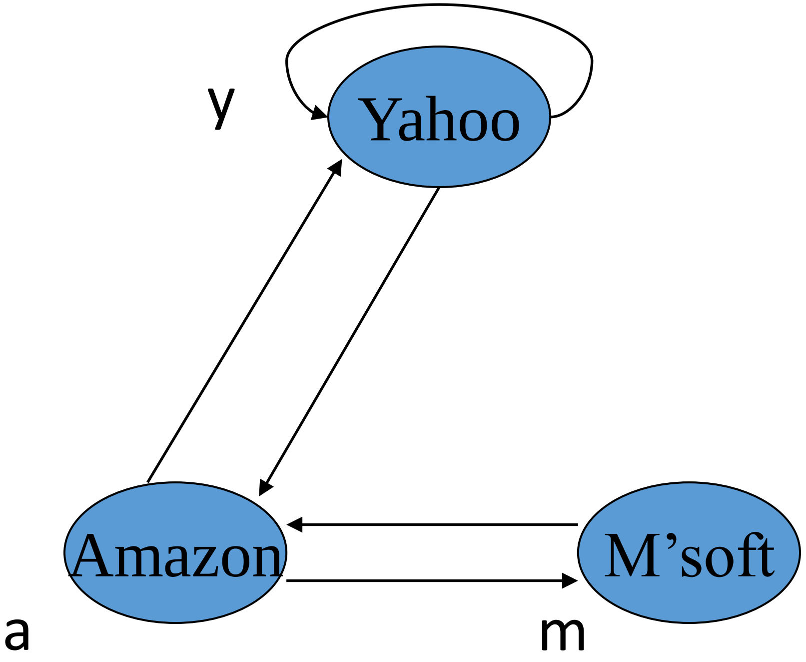
Eigenvector Formulation

- The flow equations can be written

$$\mathbf{r} = \mathbf{M}\mathbf{r}$$

- So the rank vector is an eigenvector of the stochastic web matrix
  - In fact, its first or principal eigenvector, with corresponding eigenvalue 1

Example



$$y = y/2 + a/2$$
$$a = y/2 + m$$
$$m = a/2$$

$$\begin{array}{c|ccc|} & y & a & m \\ y & 1/2 & 1/2 & 0 \\ a & 1/2 & 0 & 1 \\ m & 0 & 1/2 & 0 \end{array}$$

r = Mr

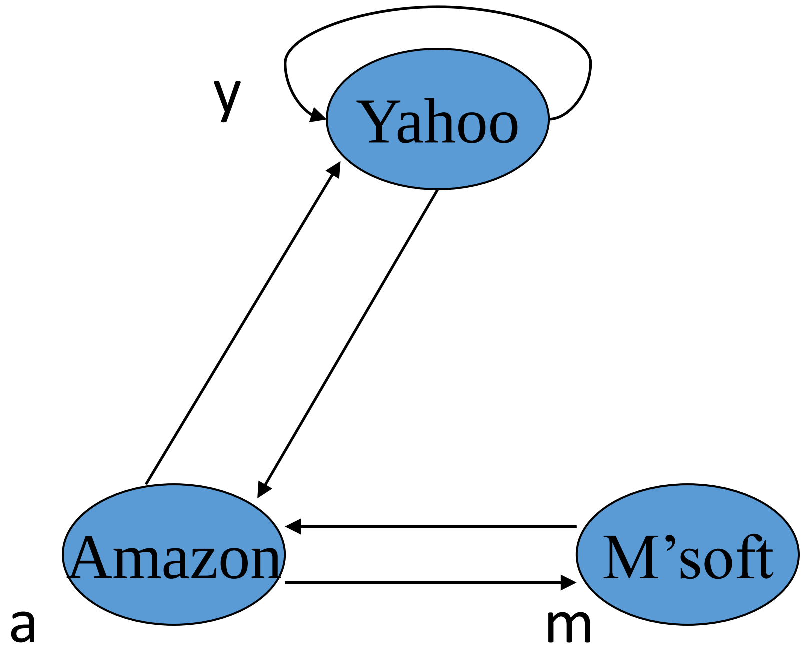
$$\begin{bmatrix} y \\ a \\ m \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix} \begin{bmatrix} y \\ a \\ m \end{bmatrix}$$

y = 2/5, a = 2/5, m = 1/5

Power Iteration method

- Simple iterative scheme (aka relaxation)
- Suppose there are N web pages
- Initialize: $\mathbf{r}^0 = [1/N,\ldots.,1/N]^T$
- Iterate: $\mathbf{r}^{k+1} = \mathbf{M}\mathbf{r}^k$
- Stop when $|\mathbf{r}^{k+1} - \mathbf{r}^k|_1 < \varepsilon$
  - $|\mathbf{x}|_1 = \sum_{1\leq i\leq N}|x_i|$ is the L_1 norm
  - Can use any other vector norm e.g., Euclidean

Power Iteration Example



$$
\begin{array}{c|ccc|}
 & y & a & m \\
\hline
y & 1/2 & 1/2 & 0 \\
a & 1/2 & 0 & 1 \\
m & 0 & 1/2 & 0 \\
\hline
\end{array}
$$

$$
\begin{array}{c}
y \\ a \\ m
\end{array}
=
\begin{array}{c}
1/3 \\ 1/3 \\ 1/3
\end{array}
\quad
\begin{array}{c}
1/3 \\ 1/2 \\ 1/6
\end{array}
\quad
\begin{array}{c}
5/12 \\ 1/3 \\ 1/4
\end{array}
\quad
\begin{array}{c}
3/8 \\ 11/24 \\ 1/6
\end{array}
\quad \ldots \quad
\begin{array}{c}
2/5 \\ 2/5 \\ 1/5
\end{array}
$$

Random Walk Interpretation

- Imagine a random web surfer
  - At any time t, surfer is on some page P
  - At time t+1, the surfer follows an outlink from P uniformly at random
  - Ends up on some page Q linked from P
  - Process repeats indefinitely

- Let **p**(t) be a vector whose i^{th} component is the probability that the surfer is at page i at time t
  - **p**(t) is a probability distribution on pages

The stationary distribution

- Where is the surfer at time t+1?
  - Follows a link uniformly at random
  - **p**(t+1) = **Mp**(t)
- Suppose the random walk reaches a state such that **p**(t+1) = **Mp**(t) = **p**(t)
  - Then **p**(t) is called a stationary distribution for the random walk
- Our rank vector **r** satisfies **r** = **Mr**
  - So it is a stationary distribution for the random surfer

Existence and Uniqueness

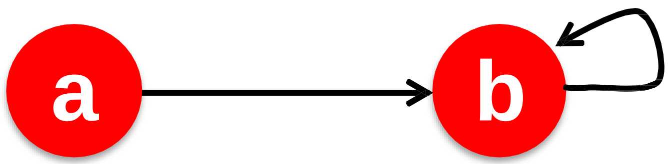
A central result from the theory of random walks (aka Markov processes):

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 $t = 0$.

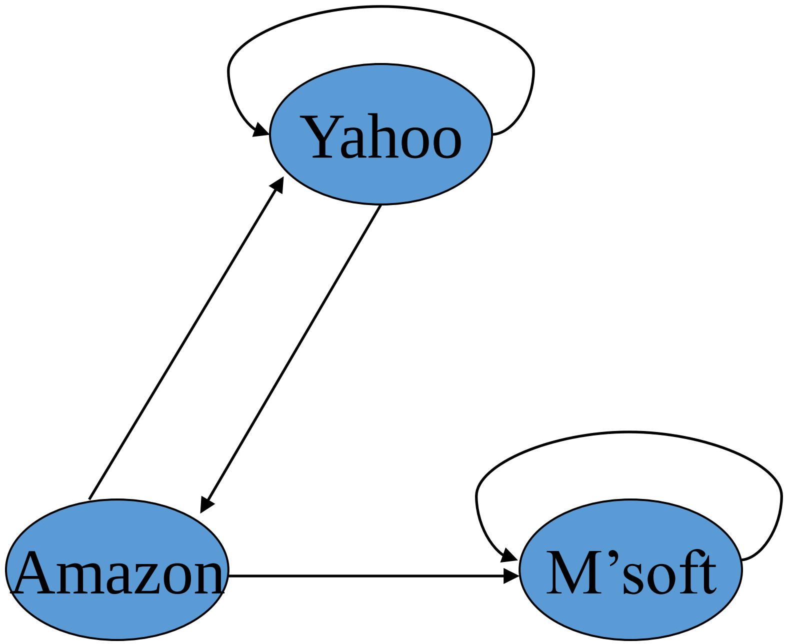
Problems?

Spider traps

- A group of pages is a spider trap if there are no links from within the group to outside the group
  - Random surfer gets trapped
- Spider traps violate the conditions needed for the random walk theorem



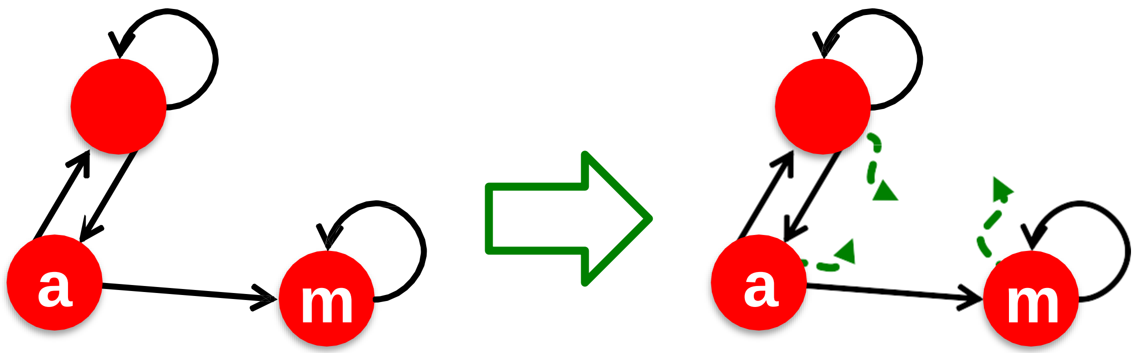
Microsoft becomes a spider trap



$$\begin{array}{c|ccc|} & y & a & m \\ \hline y & 1/2 & 1/2 & 0 \\ a & 1/2 & 0 & 0 \\ m & 0 & 1/2 & 1 \\ \hline \end{array}$$

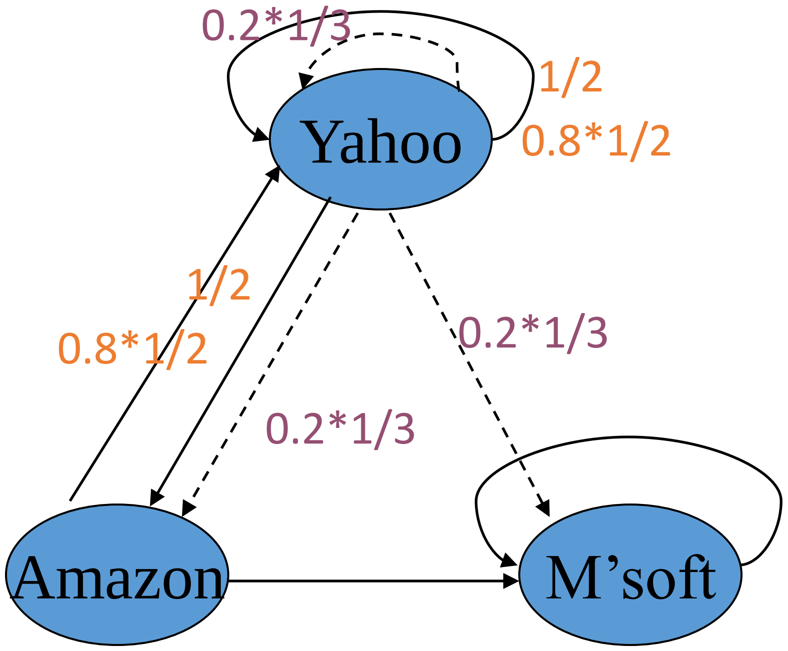
$$\begin{matrix} y \\ a \\ m \end{matrix} = \begin{matrix} 1 \\ 1 \\ 1 \end{matrix} \quad \begin{matrix} 1 \\ 1/2 \\ 3/2 \end{matrix} \quad \begin{matrix} 3/4 \\ 1/2 \\ 7/4 \end{matrix} \quad \begin{matrix} 5/8 \\ 3/8 \\ 2 \end{matrix} \quad \dots \quad \begin{matrix} 0 \\ 0 \\ 3 \end{matrix}$$

Random Teleports



- The Google solution for spider traps
- At each time step, the random surfer has two options:
  - With probability β, follow a link at random
  - With probability 1-β, jump to some page uniformly at random
  - 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

Random teleports ($\beta = 0.8$)

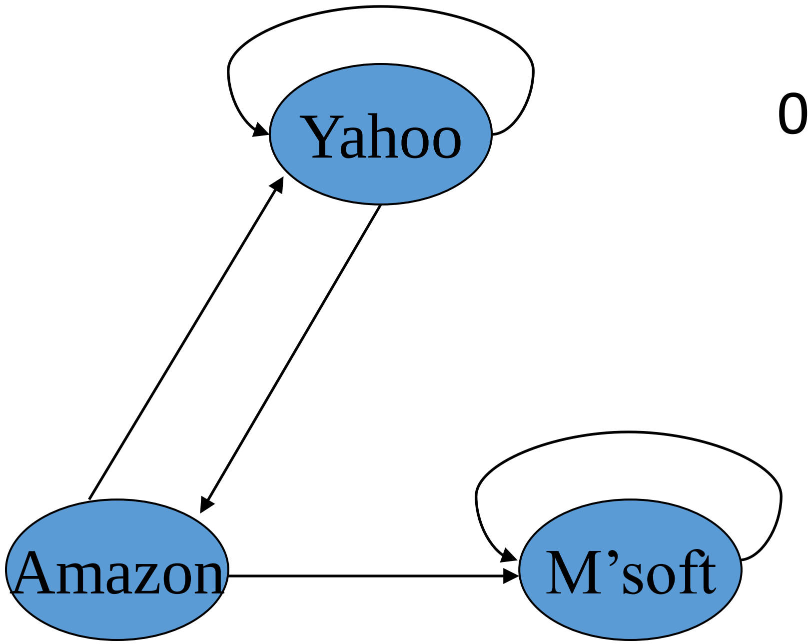


$$\begin{array}{c|c|} & y \\ y & 1/2 \\ a & 1/2 \\ m & 0 \end{array} \qquad 0.8* \begin{array}{|c|} y \\ 1/2 \\ 1/2 \\ 0 \end{array} \; + 0.2* \begin{array}{|c|} y \\ 1/3 \\ 1/3 \\ 1/3 \end{array}$$

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

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

Random teleports ($\beta = 0.8$)



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

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

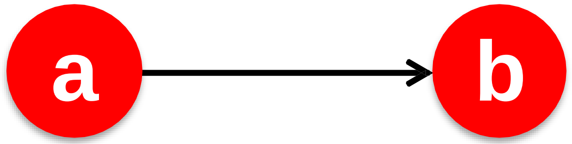
$$\begin{array}{c} y \\ a \\ m \end{array} = \begin{array}{c} 1 \\ 1 \\ 1 \end{array} \quad \begin{array}{c} 1.00 \\ 0.60 \\ 1.40 \end{array} \quad \begin{array}{c} 0.84 \\ 0.60 \\ 1.56 \end{array} \quad \begin{array}{c} 0.776 \\ 0.536 \\ 1.688 \end{array} \quad \cdots \quad \begin{array}{c} 7/11 \\ 5/11 \\ 21/11 \end{array}$$

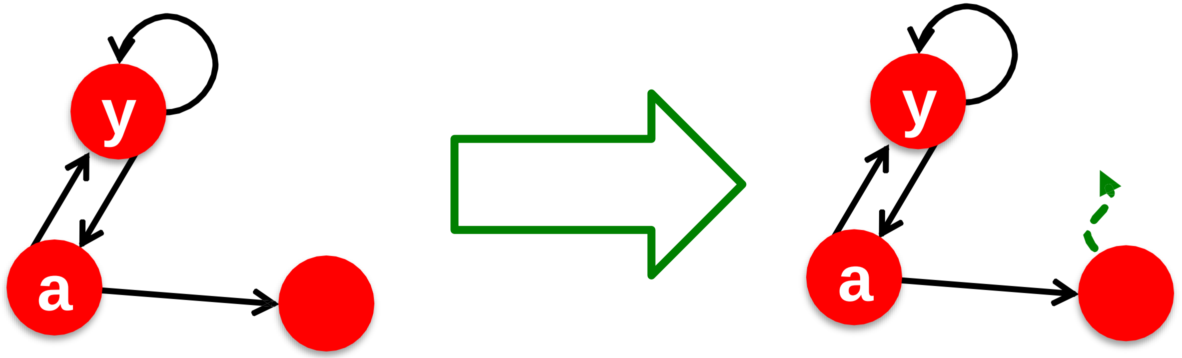
Page Rank

- Construct the N*N matrix **A** as follows
  - $A_{ij} = \beta M_{ij} + (1-\beta)/N$
- Verify that **A** is a stochastic matrix
- The page rank vector **r** is the principal eigenvector of this matrix
  - satisfying **r** = **Ar**
- Equivalently, **r** is the stationary distribution of the random walk with teleports

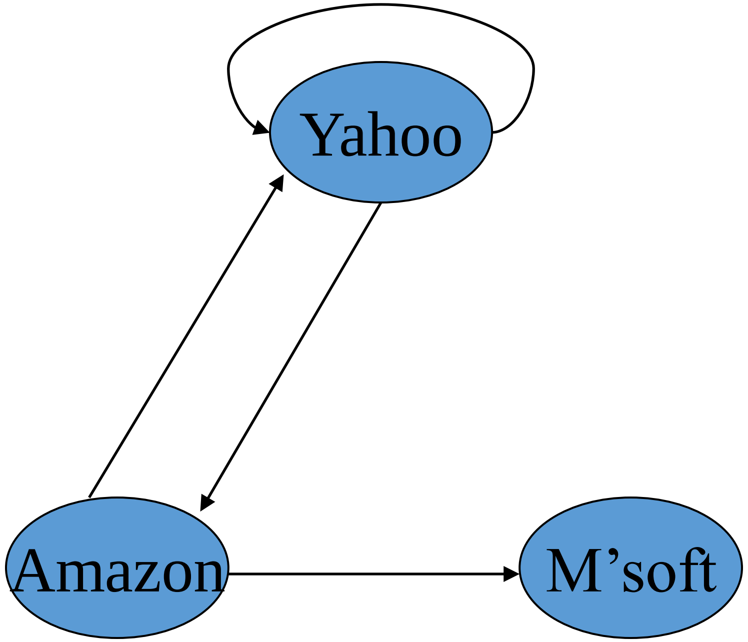
Dead ends

- Pages with no outlinks are "dead ends" for the random surfer
  - Nowhere to go on next step





Microsoft becomes a dead end



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

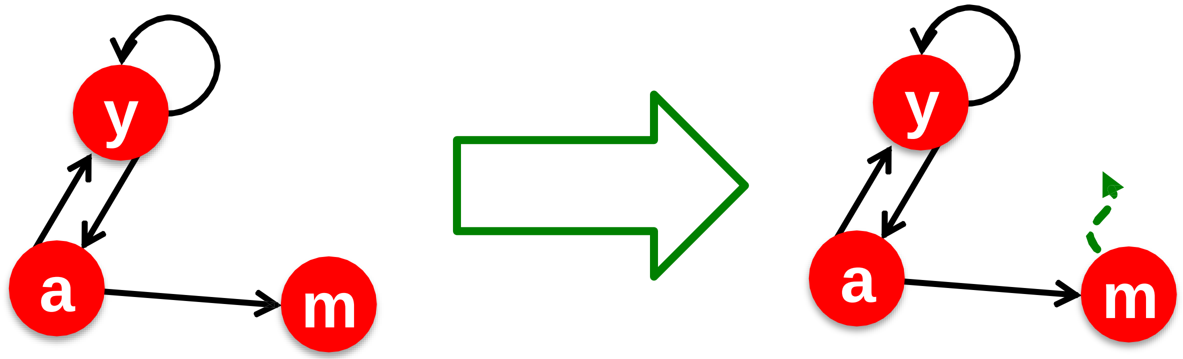
$$\begin{array}{c} y \\ a \\ m \end{array} \begin{bmatrix} 7/15 & 7/15 & 1/15 \\ 7/15 & 1/15 & 1/15 \\ 1/15 & 7/15 & 1/15 \end{bmatrix}$$

Non-stochastic!
(Sum of column is not 1)

$$\begin{array}{c} y \\ a \\ m \end{array} = \begin{array}{c} 1 \\ 1 \\ 1 \end{array} \quad \begin{array}{c} 1 \\ 0.6 \\ 0.6 \end{array} \quad \begin{array}{c} 0.787 \\ 0.547 \\ 0.387 \end{array} \quad \begin{array}{c} 0.648 \\ 0.430 \\ 0.333 \end{array} \quad \cdots \quad \begin{array}{c} 0 \\ 0 \\ 0 \end{array}$$

Solution to Dead Ends

- **Teleports:** Follow random teleport links with total 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** 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 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

Computational Issues

Computing PageRank

- Key step is matrix-vector multiplication
  - $\mathbf{r}^{new} = \mathbf{A}\mathbf{r}^{old}$
- 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 N^2 entries
    - 10^{18} is a large number!

Rearranging the Equation

$\mathbf{r} = \mathbf{A}\mathbf{r}$, where

$A_{ij} = \beta M_{ij} + (1-\beta)/N$

$r_i = \sum_{1\leq j\leq N} A_{ij}\, r_j$

$r_i = \sum_{1\leq j\leq N} [\beta M_{ij} + (1-\beta)/N]\, r_j$

$\quad = \beta \sum_{1\leq j\leq N} M_{ij}\, r_j + (1-\beta)/N \sum_{1\leq j\leq N} r_j$

$\quad = \beta \sum_{1\leq j\leq N} M_{ij}\, r_j + (1-\beta)/N$, since $|\mathbf{r}| = 1$

$\mathbf{r} = \beta\mathbf{M}\mathbf{r} + [(1-\beta)/N]_N$

where $[x]_N$ is an N-vector with all entries x

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

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

Sparse Matrix Formulation

- We can rearrange the page rank equation:
  - $\mathbf{r} = \beta\mathbf{M}\mathbf{r} + [(1-\beta)/N]_N$
  - $[(1-\beta)/N]_N$ is an N-vector with all entries $(1-\beta)/N$
- **M** is a sparse matrix!
  - 10 links per node, approx 10N entries
- So in each iteration, we need to:
  - Compute $\mathbf{r}^{new} = \beta\mathbf{M}\mathbf{r}^{old}$
  - Add a constant value $(1-\beta)/N$ to each entry in $\mathbf{r}^{new}$

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

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

Basic Algorithm

- Initialize: $\mathbf{r}^{old} = [1/N]_N$
- Iterate:
  - Update: Perform a sequential scan of **M** and $\mathbf{r}^{old}$ to update $\mathbf{r}^{new}$
  - Every few iterations, compute $|\mathbf{r}^{new}-\mathbf{r}^{old}|$ and stop if it is below threshold

Summary

- We introduced
  - Network analysis, centrality
  - PageRank, which powers Google

- **Important to note**: Hyperlink based ranking is not the only algorithm used in search engines. In fact, it is combined with many content based factors to produce the final ranking presented to the user.