CS145: INTRODUCTION TO DATA MINING

16: Graph and Network: Random Walk

Instructor: Yizhou Sun

yzsun@cs.ucla.edu

November 27, 2021

Methods to Learn

	Vector Data	Set Data	Sequence Data/Time Series	Text Data	Graph Data
Classification	**Logistic Regression; Decision Tree; NN**			**Naïve Bayes for Text**	Label Propagation
Clustering	**K-means; Mixture Models**			**PLSA**	Spectral Clustering
Prediction	**Linear Regression; Regression Tree; NN** GLM*		**AR Model**		
Frequent Pattern Mining		**Apriori; FP growth**	**GSP; PrefixSpan**		
Similarity Search			**DTW**		**P-PageRank**
Ranking					**PageRank**

Random Walk on Graphs

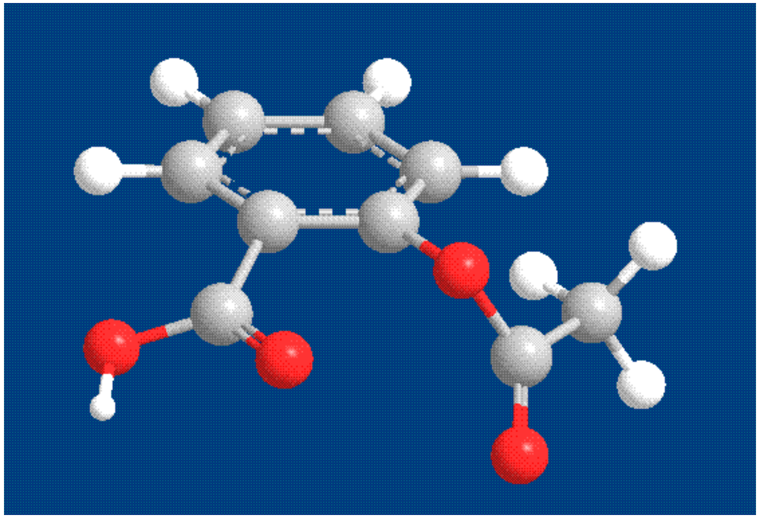
- Introduction to Graph/Network Data

- PageRank

- Personalized PageRank

- Summary

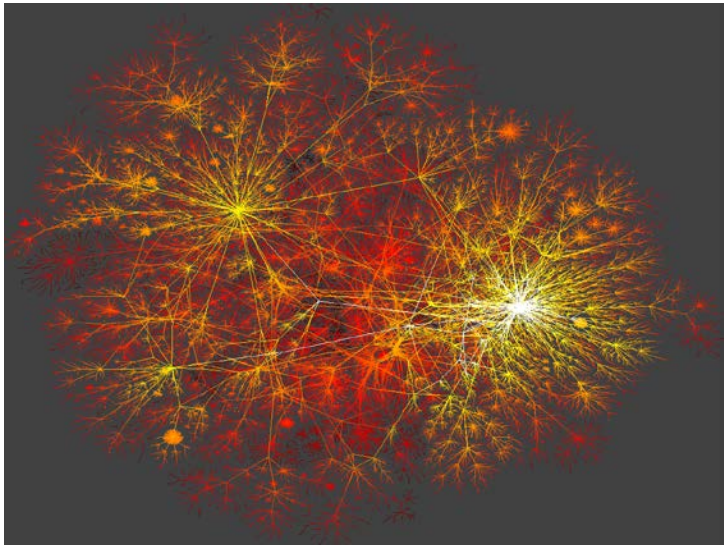
Graph, Graph, Everywhere



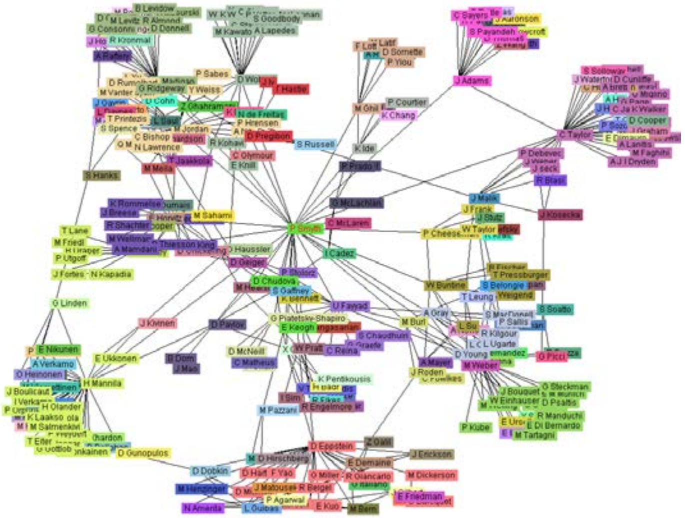
Aspirin

from H. Jeong et al Nature 411, 41 (2001)

Yeast protein interaction network



Internet



Co-author network

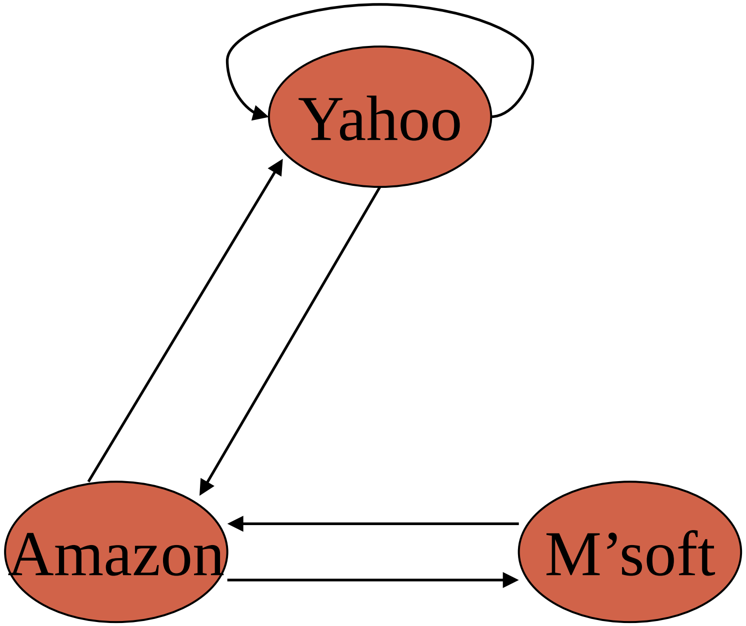
Why Graph Mining?

- Graphs are ubiquitous
  - Chemical compounds (Cheminformatics)
  - Protein structures, biological pathways/networks (Bioinformactics)
  - Program control flow, traffic flow, and workflow analysis
  - XML databases, Web, and social network analysis
- Graph is a general model
  - Trees, lattices, sequences, and items are degenerated graphs
- Diversity of graphs
  - Directed vs. undirected, labeled vs. unlabeled (edges & vertices), weighted vs. unweighted, homogeneous vs. heterogeneous
- Complexity of algorithms: many problems are of high complexity

Representation of a Graph

- $G =< V, E >$
  - $V = \{u_1, \dots, u_n\}$: node set
  - $E \subseteq V \times V$: edge set
- Adjacency matrix
  - $A = \{a_{ij}\}, i, j = 1, \dots, N$
    - $a_{ij} = 1, if \ < u_i, u_j > \in E$
    - $a_{ij} = 0, if \ < u_i, u_j > \notin E$
  - Undirected graph vs. Directed graph
    - $A = A^{\mathrm{T}}\ vs. A \neq A^{\mathrm{T}}$
  - Weighted graph
    - Use *W* instead of *A*, where w_{ij} represents the weight of edge $< u_i, u_j >$

Example



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

Adjacency matrix A

Random Walk on Graphs

- Introduction to Graph/Network Data
- PageRank
- Personalized PageRank
- Summary

The History of PageRank

- PageRank was developed by Larry Page (hence the name *Page*-Rank) and Sergey Brin.

- It is first as part of a research project about a new kind of search engine. That project started in 1995 and led to a functional prototype in 1998.

- Shortly after, Page and Brin founded Google.

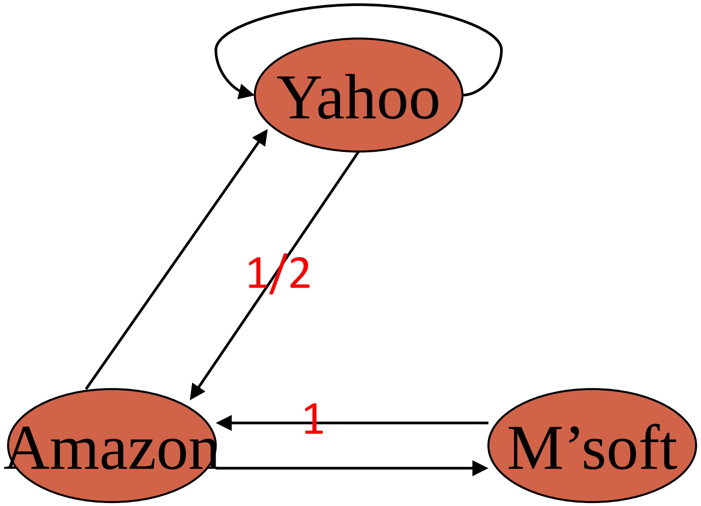
Ranking web pages

- Web pages are not equally “important”
  - www.cnn.com vs. a personal webpage
- Inlinks as votes
  - The more inlinks, the more important
- Are all inlinks equal?
  - Higher ranked inlink should play a more important role
  - 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

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



Matrix Formulation

$$\begin{array}{cccc} & y & a & m \\ y & 1 & 1 & 0 \\ a & 1 & 0 & 1 \\ m & 0 & 1 & 0 \end{array} \Rightarrow \begin{array}{c|c|c} \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & 1 \\ 0 & \frac{1}{2} & 0 \end{array}$$

- N by N Matrix **M**
- Suppose page j has n outlinks
  - If j -> i, then $M_{ij}=1/n$; Else $M_{ij}=0$
  - Or $M = (D^{-1}A)^T$, where D is a diagonal matrix and $D_{ii} = \sum_j A_{ij}$
- **M** is a column stochastic matrix
  - Columns sum to 1
- Suppose **r** is a vector with one entry per web page
  - r_i is the importance score of page i
  - Call it the rank vector
  - $|\mathbf{r}| = 1$ (i.e., $r_1 + r_2 + \cdots + r_N = 1$)
- ***r = Mr***

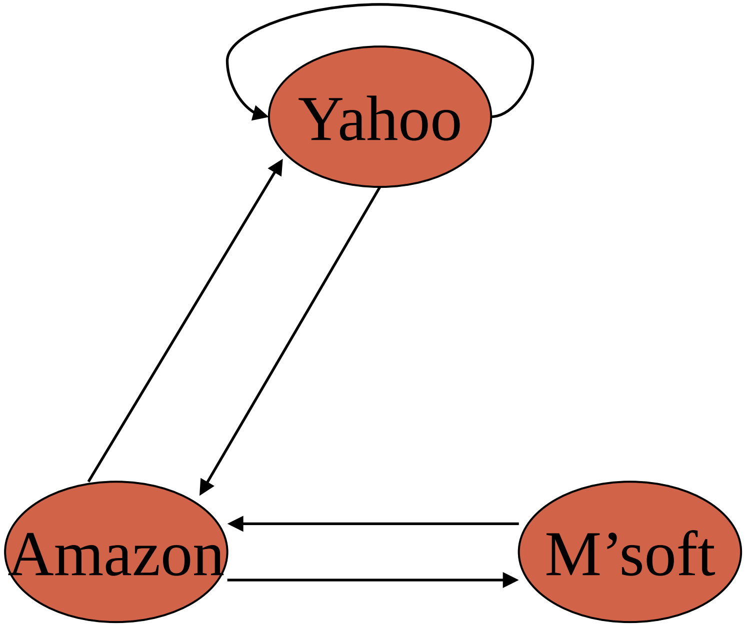
Eigenvector formulation

- The flow equations can be written

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

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

Example



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

$$a = y/2 + m$$

$$m = a/2$$

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

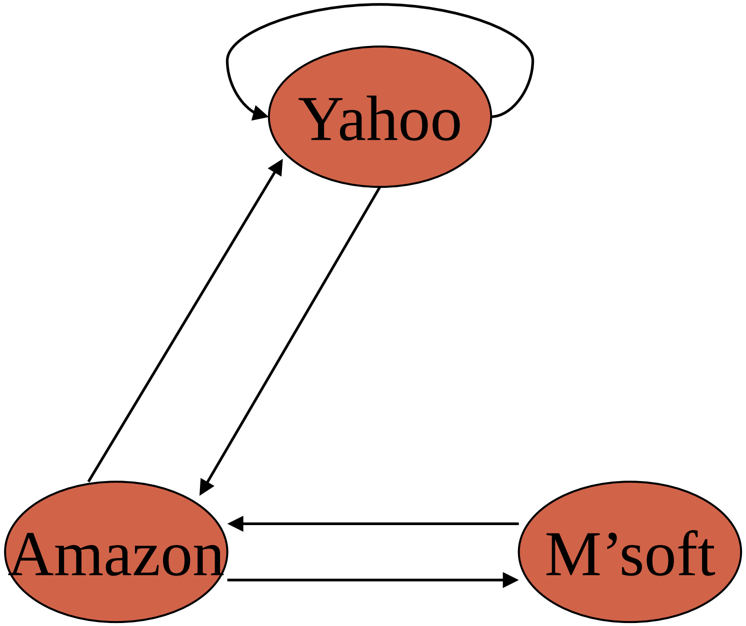
r = M * r

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

Power Iteration method

- Simple iterative scheme
- 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 $|\mathbf{r}^{k+1} - \mathbf{r}^k|_1 < \varepsilon$
    - $|\mathbf{x}|_1 = \sum_{1\le i\le N}|x_i|$ is the L_1 norm
    - Can use any other vector norm e.g., Euclidean

Power Iteration Example



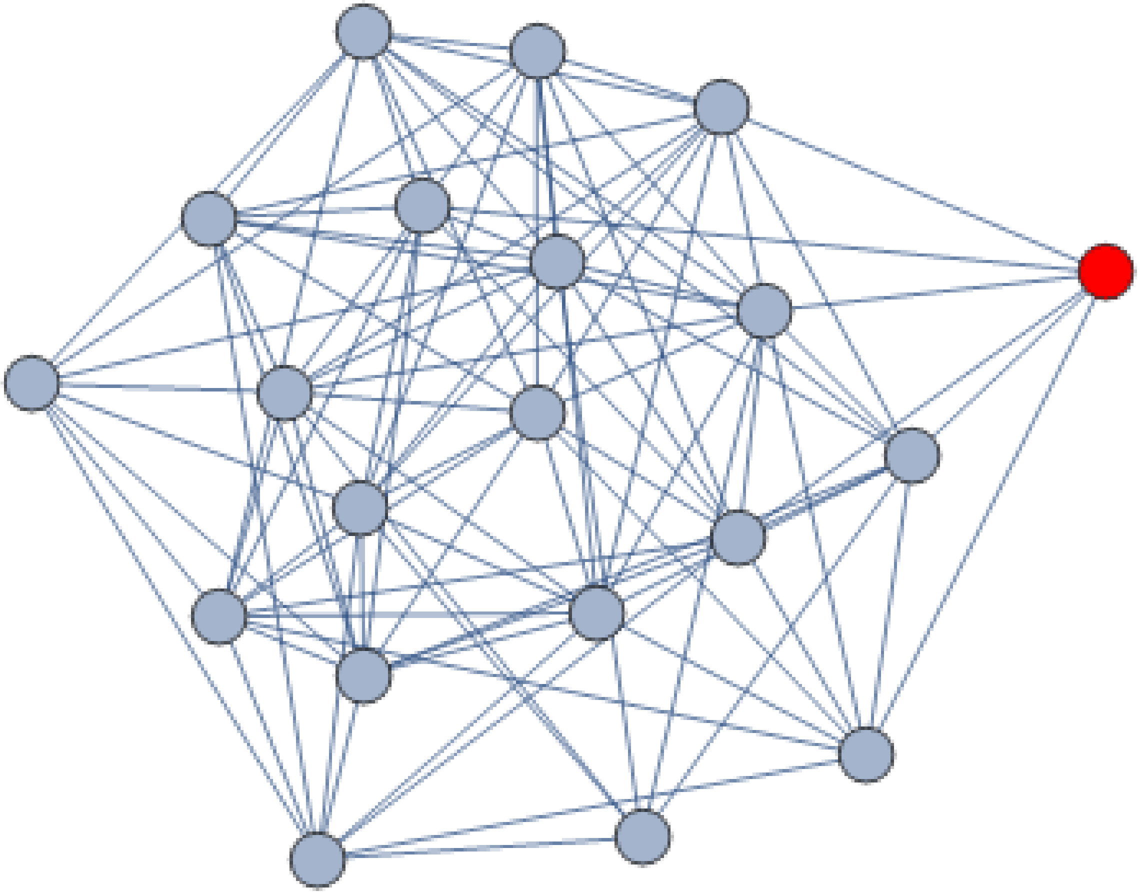
$$\begin{array}{c|ccc} & y & a & m \\ \hline y & 1/2 & 1/2 & 0 \\ a & 1/2 & 0 & 1 \\ m & 0 & 1/2 & 0 \end{array}$$

$$\begin{array}{cccccccc} y & & 1/3 & 1/3 & 5/12 & 3/8 & & 2/5 \\ a & = & 1/3 & 1/2 & 1/3 & 11/24 & \dots & 2/5 \\ m & & 1/3 & 1/6 & 1/4 & 1/6 & & 1/5 \\ & & \boldsymbol{r}_0 & \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{r}_3 & \dots & \boldsymbol{r}^* \end{array}$$

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

Random Walk Example



https://mathematica.stackexchange.com/questions/156626/generate-random-walk-on-a-graph

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

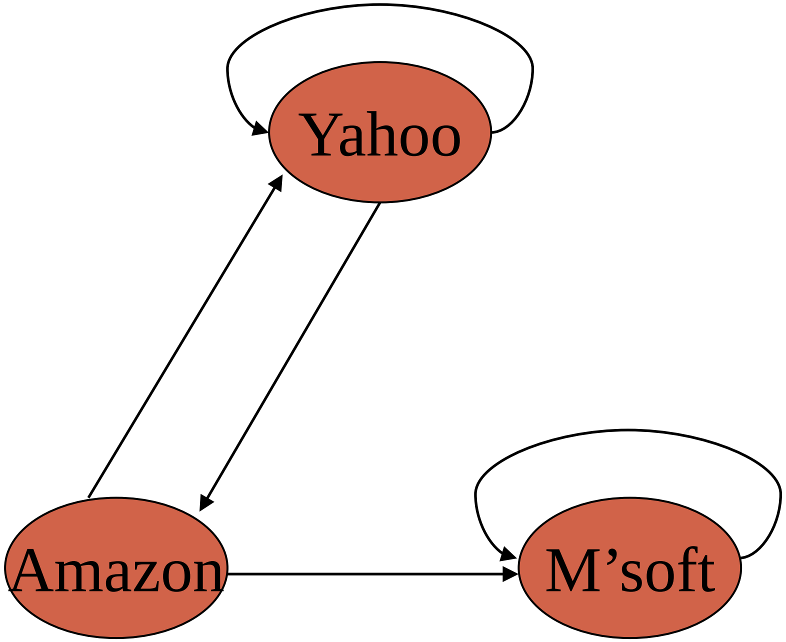
A central result from the theory of random walks (aka Markov processes):

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.

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!

Microsoft becomes a spider trap



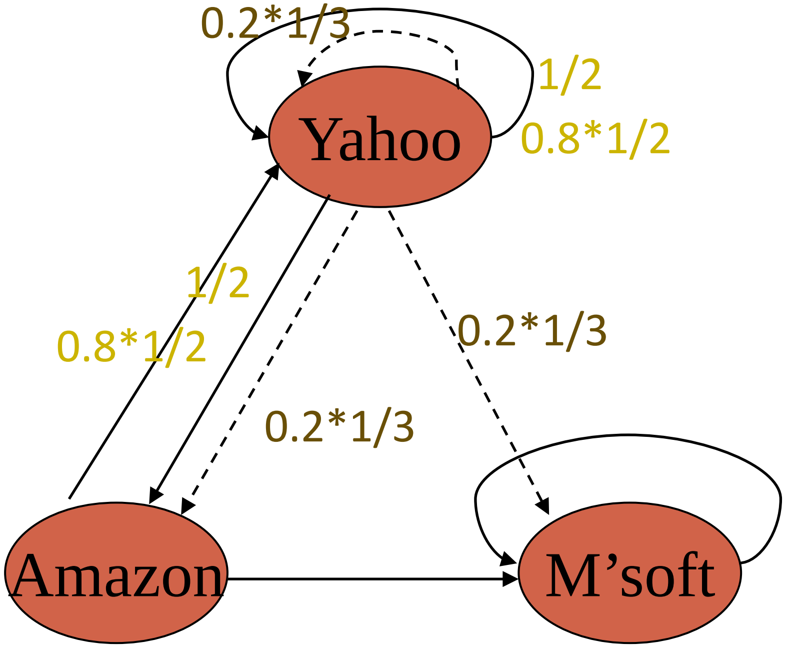
$$\begin{array}{c|ccc|}
 & y & a & m \\
\hline
y & 1/2 & 1/2 & 0 \\
a & 1/2 & 0 & 0 \\
m & 0 & 1/2 & 1 \\
\hline
\end{array}$$

$$\begin{array}{ccccccccc}
y & & 1/3 & 1/3 & 1/4 & 5/24 & & 0 \\
a & = & 1/3 & 1/6 & 1/6 & 1/8 & \dots & 0 \\
m & & 1/3 & 1/2 & 7/12 & 2/3 & & 1
\end{array}$$

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 $1-\beta$, 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 ($\beta = 0.8$)



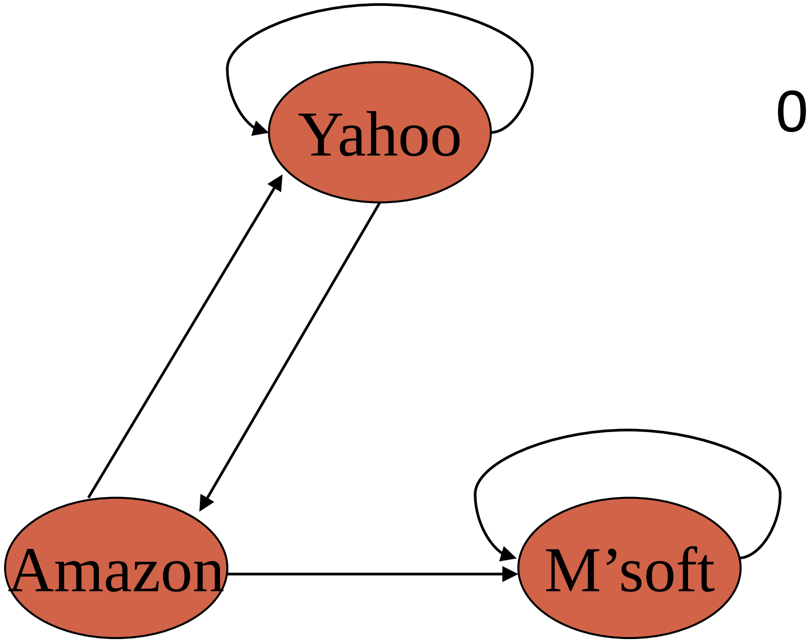
$$\begin{array}{c|c|} & y \\ y & 1/2 \\ a & 1/2 \\ m & 0 \end{array} \quad 0.8* \begin{array}{|c|} y \\ 1/2 \\ 1/2 \\ 0 \end{array} \quad + 0.2* \begin{array}{|c|} y \\ 1/3 \\ 1/3 \\ 1/3 \end{array}$$

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

- - - -▸ : teleport links from “Yahoo”

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

Random teleports ($\beta = 0.8$)



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

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

$$\begin{array}{c} y\\ a\\ m \end{array} = \begin{bmatrix}0.333\\ 0.333\\ 0.333\end{bmatrix} \begin{bmatrix}0.333\\ 0.200\\ 0.467\end{bmatrix} \begin{bmatrix}0.280\\ 0.200\\ 0.520\end{bmatrix} \begin{bmatrix}0.259\\ 0.179\\ 0.563\end{bmatrix} \dots \begin{bmatrix}7/33\\ 5/33\\ 21/33\end{bmatrix}$$

Matrix formulation

- Suppose there are N pages
  - Consider a page j, with set of outlinks O(j)
  - We have $M_{ij} = 1/|O(j)|$ when j->i and $M_{ij} = 0$ otherwise
  - The random teleport is equivalent to
    - adding a teleport link from j to every other page with probability $(1-\beta)/N$
    - reducing the probability of following each outlink from $1/|O(j)|$ to $\beta/|O(j)|$
    - Equivalent: tax each page a fraction $(1-\beta)$ of its score and redistribute evenly

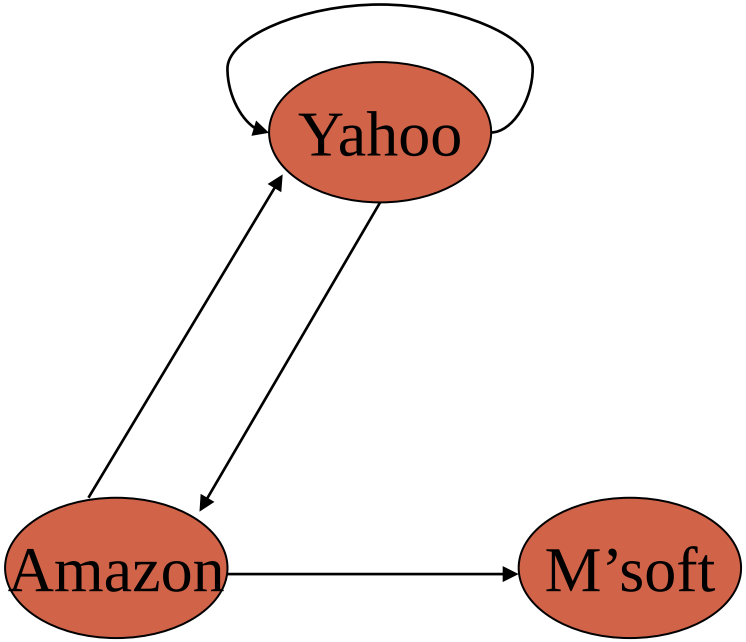
PageRank

- Construct the N-by-N matrix **B** as follows
  - $B_{ij} = \beta M_{ij} + (1-\beta)/N$
- Verify that **B** is a stochastic matrix
- The page rank vector **r** is the principal eigenvector of this matrix
  - satisfying $\mathbf{r} = \mathbf{B}\mathbf{r}$
- 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



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

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

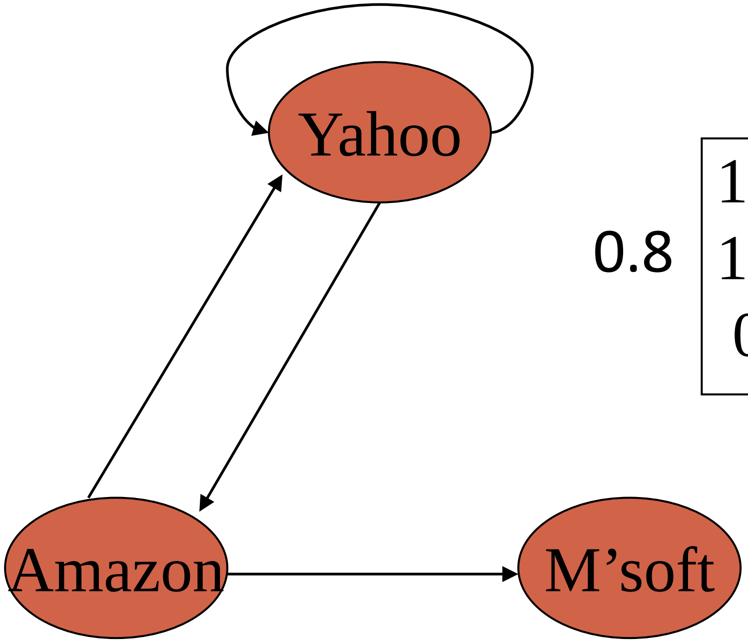
Non-stochastic!

$$\begin{matrix} y \\ a \\ m \end{matrix} = \begin{matrix} 1/3 \\ 1/3 \\ 1/3 \end{matrix} \quad \begin{matrix} 1/3 \\ 0.2 \\ 0.2 \end{matrix} \quad \ldots \quad \begin{matrix} 0 \\ 0 \\ 0 \end{matrix}$$

Dealing with dead-ends

- Method 1: Teleport
  - Follow random teleport links with probability 1.0 from dead-ends
  - Adjust matrix accordingly
- Method 2: Prune and propagate
  - Preprocess the graph to eliminate dead-ends
    - Might require multiple passes
  - Compute pagerank on reduced graph
  - Approximate values for deadends by propagating values from reduced graph

Dealing dead end: teleport

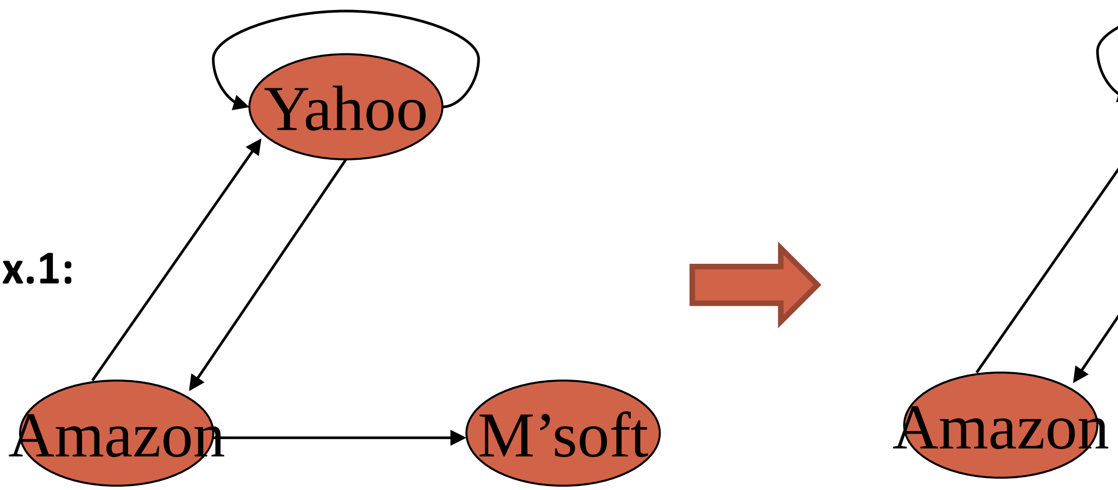


$$0.8 \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 0 \end{bmatrix} + \begin{bmatrix} 0.2*1/3 & 0.2*1/3 & 1*1/3 \\ 0.2*1/3 & 0.2*1/3 & 1*1/3 \\ 0.2*1/3 & 0.2*1/3 & 1*1/3 \end{bmatrix}$$

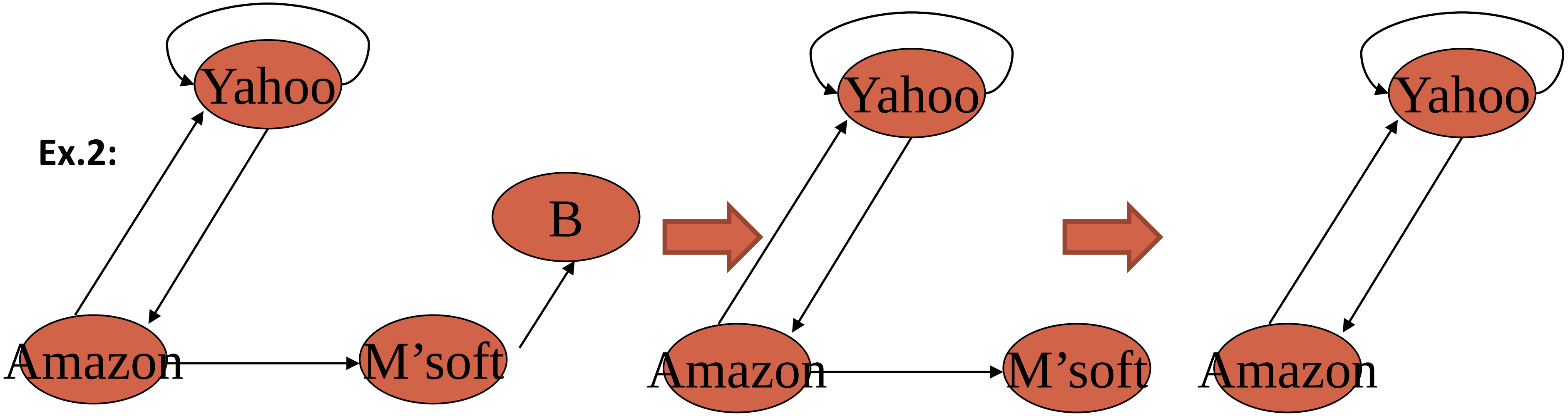
$$\begin{array}{c|ccc|} y & 7/15 & 7/15 & 1/3 \\ a & 7/15 & 1/15 & 1/3 \\ m & 1/15 & 7/15 & 1/3 \end{array}$$

Dealing dead end: reduce graph

Ex.1:



Ex.2:



Computing PageRank

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

Rearranging the equation

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

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

$r_i = \sum_{1 \le j \le N} B_{ij}\, r_j$

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

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

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

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 $\mathbf{r}^{\text{new}} = \beta\mathbf{M}\mathbf{r}^{\text{old}}$
  - Add a constant value $(1-\beta)/N$ to each entry in $\mathbf{r}^{\text{new}}$

*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
  - still won’t fit in memory, but will fit on disk

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

Basic Algorithm

- Assume we have enough RAM to fit $\mathbf{r}^{new}$, plus some working memory
    - Store $\mathbf{r}^{old}$ and matrix $\mathbf{M}$ on disk

Basic Algorithm:

- Initialize: $\mathbf{r}^{old} = [1/N]_N$
- Iterate:
    - Update: Perform a sequential scan of $\mathbf{M}$ and $\mathbf{r}^{old}$ to update $\mathbf{r}^{new}$
    - Write out $\mathbf{r}^{new}$ to disk as $\mathbf{r}^{old}$ for next iteration
    - Every few iterations, compute $|\mathbf{r}^{new}-\mathbf{r}^{old}|$ and stop if it is below threshold
        - Need to read in both vectors into memory

Random Walk on Graphs

- Introduction to Graph/Network Data
- PageRank
- Personalized PageRank
- Summary

Personalized PageRank

- Query-dependent Ranking
  - For a query webpage u, which webpages are most important to u?
    - We need a measure s(u,v)
  - The relative important webpages to different queries would be different

Calculation of P-PageRank

- Recall PageRank calculation:
  - $\mathbf{r} = \beta\mathbf{M}\mathbf{r} + [(1-\beta)/N]_N$ or
  - $\mathbf{r} = \beta\mathbf{M}\mathbf{r} + (1-\beta)\, q$, where $q = \begin{pmatrix} 1/N \\ 1/N \\ \dots \\ 1/N \end{pmatrix}$

- For P-PageRank, $s(u,v) = \mathbf{r}_u(v)$, where $\mathbf{r}_u = \beta\mathbf{M}\,\mathbf{r}_u + (1-\beta)\, q$

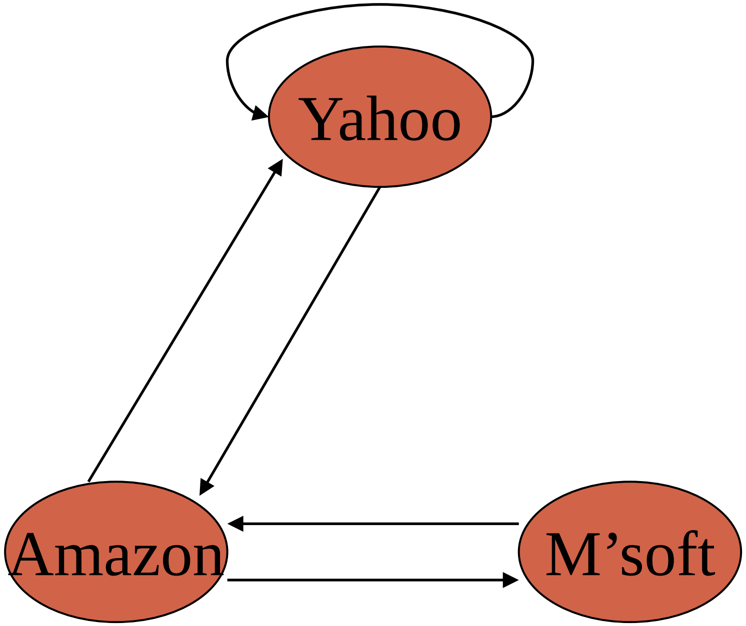
  by replacing q with $q = \begin{pmatrix} 0 \\ 0 \\ \dots \\ 1 \\ \dots \\ 0 \end{pmatrix} \leftarrow$ uth webpage

  - Teleport to webpage u

Example: r_Y

$$\mathbf{r}_Y = \beta \mathbf{M}\, \mathbf{r}_Y + (1-\beta)\, q,\ \beta = 0.8$$

$$q = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$



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

$$\begin{array}{ccccccccc} y & & 1/3 & 0.47 & 0.55 & 0.54 & & 0.55 \\ a & = & 1/3 & 0.4 & 0.29 & 0.31 & \ldots & 0.32 \\ m & & 1/3 & 0.13 & 0.16 & 0.14 & & 0.13 \\ & & r_{Y0} & r_{Y1} & r_{Y2} & r_{Y3} & \ldots & r_Y^* \end{array}$$

Random Walk on Graphs

- Introduction to Graph/Network Data
- PageRank
- Personalized PageRank
- Summary

Summary

- Random walk on graph
  - PageRank for ranking
  - P-PageRank for similarity search