Information Retrieval on the Web
- a brief intro to page rank

(Based on lecture material by Prof. Weiyi Meng)

Web vs. Ordinary Text Retrieval

- **Web pages are**
  - very voluminous and diversified
  - widely distributed on many servers.
  - extremely dynamic/volatile.
- **Web pages have**
  - more structures (extensively tagged).
  - are extensively linked.
  - may often have other associated metadata
- **Web users are**
  - ordinary people without special training
    - they tend to submit short queries. [~1.7wrds/'97->~2.4/'00]
  - There is a very large user community.

Use of Link Information (1)

Hyperlinks among web pages provide new document retrieval opportunities.

e.g.:

- Anchor text can be used as *index term* for a *referenced page*
- **The ranking score (similarity) of a page with a query can be spread to its neighboring pages.**
- **Links can be used to compute the importance of web pages based on citation analysis.**
- Links can be combined with a regular query to find authoritative pages on a given topic.

PageRank

To combat term spam, Google introduced two innovations:

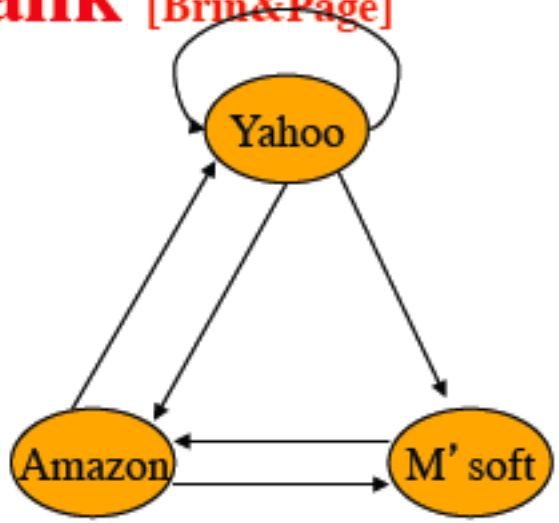
1. PageRank was used to simulate where Web surfers, starting at a random page, would tend to congregate if they followed randomly chosen outlinks from the page at which they were currently located, and this process were allowed to iterate many times. Pages that would have a large number of surfers were considered more “important” than pages that would rarely be visited. Google prefers important pages to unimportant pages when deciding which pages to show first in response to a search query.

2. The content of a page was judged not only by the terms appearing on that page, but by the terms used in or near the links to that page. Note that while it is easy for a spammer to add false terms to a page they control, they cannot as easily get false terms added to the pages that link to their own page, if they do not control those pages.

II. PageRank [Brin&Page]

Initial Idea:

– Web as a big graph.



– A "surfer/pigeon" keeps randomly clicking on links.

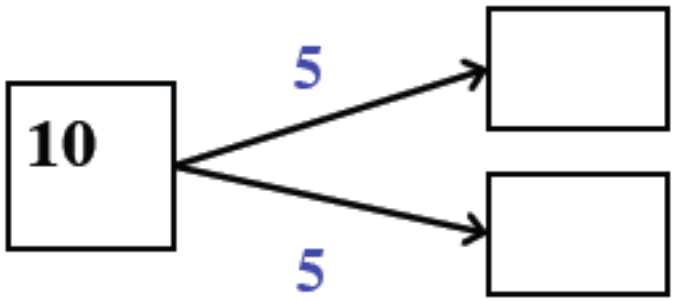
– The "*importance*" *of a page* is the probability that the surfer finds herself on that page.

– then rank the returned pages in decreasing order of importance

Computing PageRank (cont'd)

PageRank principles:

- A page has an "importance" weight
- If a page is linked to by *many* pages, then the page is likely to be important.
- If a page is linked to by *important* pages, then the page is likely to be important even though there aren't too many pages linking to it.
- The importance of a page is divided evenly and propagated to the pages it points to:



Computing PageRank (cont'd)

PageRank Definition: For each web page u, let

- **F_u = the set of pages u points to,**
- **B_u = the set of pages that point to u,**
- **N_u = the number pages in F_u.**

If all pages equally likely then probability R(u) of landing on page u would be $\mathbf{R(u) = \Sigma_{v \in Bu} \; 1 / N_v}$ since from each node **v** only 1 out of $\mathbf{N_v}$ edges lead to **u**.

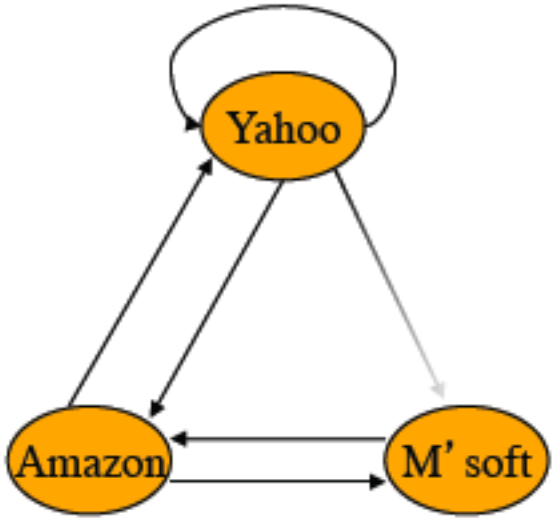
But now all pages are no longer equally likely! Instead, each page gives away some of its importance to its neighbors; define iterative formula $\mathbf{R_i(u) = \sum_{v \in B_u} (R_{i-1}(v) / N_v)}$

PageRank can be computed *iteratively*:

- Initialize all page ranks to be 1/N --- N = number of vertices in the Web graph.
- This gives $R_0(u)$

Note: for computational simplicity, sometimes we initialize all page ranks to be 1

Example 2 (by Ullman)



- Equations:
  - $y = y/3 + a/2$
  - $a = y/3 + m$
  - $m = y/3 + a/2$
- Can be rephrased using matrices

$$\begin{pmatrix} y \\ a \\ m \end{pmatrix} = \underbrace{\begin{pmatrix} 1/3 & 1/2 & 0 \\ 1/3 & 0 & 1 \\ 1/3 & 1/2 & 0 \end{pmatrix}}_{\text{Call it matrix } M} \begin{pmatrix} y \\ a \\ m \end{pmatrix}$$

***M* is the transition matrix of the Web Graph**

Computing PageRank (cont' d)

Matrix representation

- Let M be an N×N matrix and M[u,v] be the entry at the u-th row and v-th column.

  M[u,v] = $1/N_v$ if page ***v*** has a link to page ***u***

  M[u,v] = 0 if there is no link from v to u

- Let R_i be the N×1 rank vector for i-th iteration and R_0 be the initial rank vector (say 1 for each node)

Then $R_i = M \times R_{i-1}$

or we can just try to solve equation

$$R = M \times R$$

But, will the ranks R converge?

Computing PageRank (cont'd)

If the ranks converge, i.e., there is a rank vector R such that

$$R = M \times R,$$

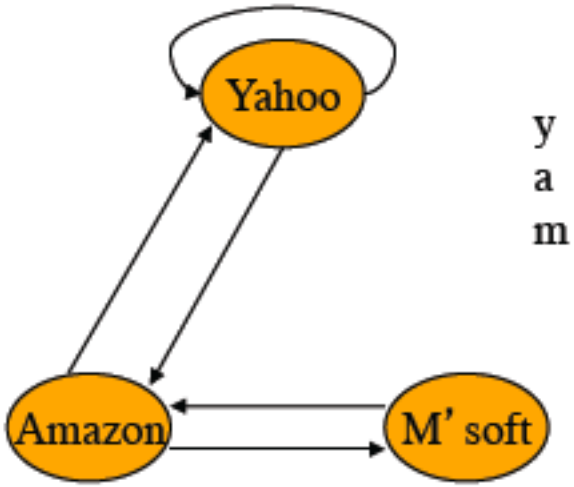
i.e. R is the eigenvector of matrix M with eigenvalue being 1.

Principal eigen value for A stochastic matrix is 1

Convergence is guaranteed only if

- **M is aperiodic (i.e. the Web graph is not a big cycle). This is practically guaranteed for Web.**
- **M is irreducible (i.e. the Web graph is strongly connected). This is usually not true.**

Example 2 (by Ullman)

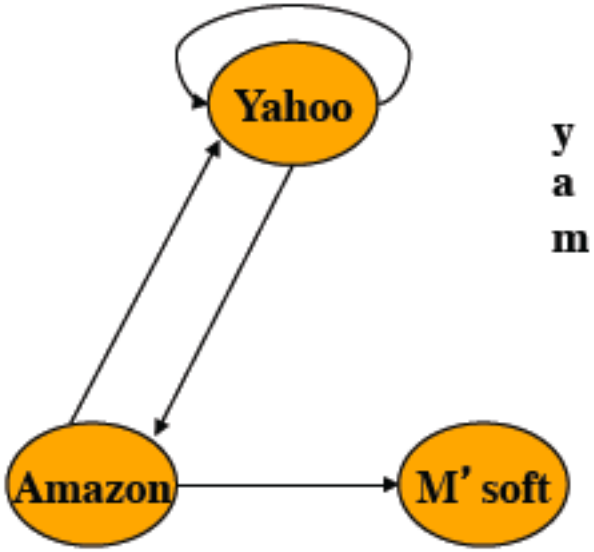


	y	a	m
y	1/2	1/2	0
a	1/2	0	1
m	0	1/2	0

- Equation $R = M R$:
  - $y = y/2 + a/2$
  - $a = y/2 + \quad m$
  - $m = \quad a/2$

$$\begin{bmatrix} y \\ a \\ m \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ 3/2 \\ 1/2 \end{bmatrix} \begin{bmatrix} 5/4 \\ 1 \\ 3/4 \end{bmatrix} \begin{bmatrix} 9/8 \\ 11/8 \\ 1/2 \end{bmatrix} \ldots \begin{bmatrix} 6/5 \\ 6/5 \\ 3/5 \end{bmatrix}$$

PageRank: **problem 1**



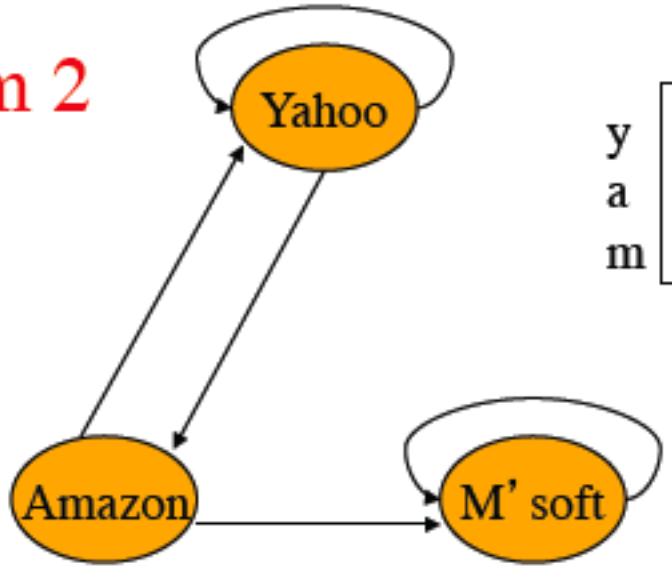
	y	a	m
y	1/2	1/2	0
a	1/2	0	0
m	0	1/2	0

- Equation $R = M\,R$:
  - $y = y/2 + a/2$
  - $a = y/2$
  - $m = a/2$

y		1	1	3/4	5/8		0
a	=	1	1/2	1/2	3/8	. . .	0
m		1	1/2	1/4	1/4		0

M' soft is a **dead-end**

Page Rank: problem 2



	y	a	m
y	1/2	1/2	0
a	1/2	0	0
m	0	1/2	1

- Equations $R = M\,R$:
  - $y = y/2 + a/2$
  - $a = y/2$
  - $m = a/2 + m$

y		1	1	3/4	5/8		0
a	=	1	1/2	1/2	3/8	. . .	0
m		1	3/2	7/4	2		3

M' soft is a **spider trap:** once you enter it, you never leave, so all other pages end up looking *useless* - ***rank 0!***

Page Rank (called Pigeon Rank in text)

A solution to spider trap and other problems

- Conceptually, at any point the surfer may randomly choose to jump to some totally different page, rather than follow links from current page. Let probability of staying be d, and hence of jumping be 1-*d*.
- This leads to the equation

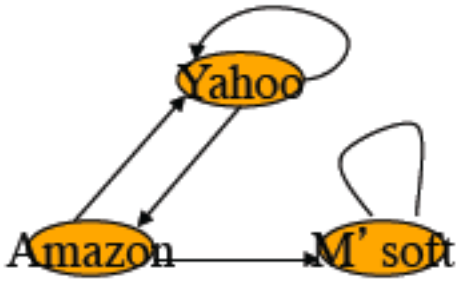
$$R = d\,(M \times R) + (1 - d)$$

- (Actually, more complex formula but don't worry)

Note: R = M' x R , where M' = d M + (1 – d) K ,
and K is the Reset Matrix with all entries being 1/N.

Notice M' is stochastic except when Web Graph has sinks.

Example 4



- Equations $R = 0.8(M \times R) + 0.2$:
  - $y = 0.8(y/2 + a/2) + 0.2$
  - $a = 0.8(y/2) + 0.2$
  - $m = 0.8(a/2 + m) + 0.2$

	y	a	m
y	1/2	1/2	0
a	1/2	0	0
m	0	1/2	1

y		1	1.00	0.84	0.776		7/11
a	=	1	0.60	0.60	0.536	. . .	5/11
m		1	1.40	1.56	1.688		21/11

PageRank SUMMARY

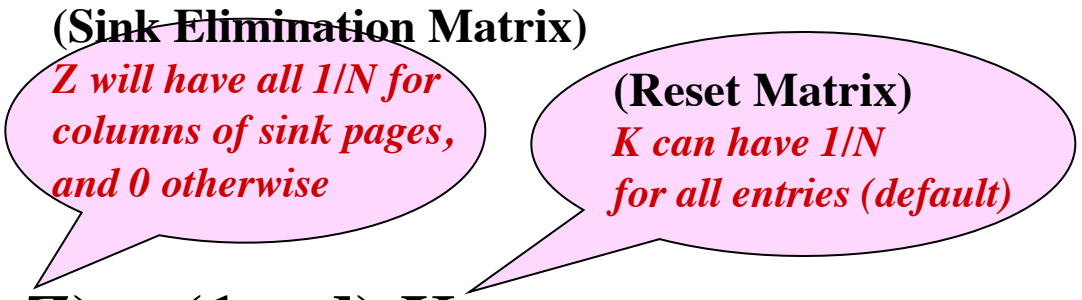
- for every node v, the probability of following one of the links to a neighbor is 1/(number of outgoing links from v)
- construct matrix M, where row j entries show for every other node k the probability of following link from k to j
- Solve equation like

$$R = d\,(M \times R) + (1 - d)$$

to get the ranks of each node

Computing PageRank (cont'd)

For Web Graphs with Sinks:



$$M^* = d\,(M + Z) + (1 - d)\,K$$

- **M* is irreducible.**
- **M* is stochastic, the sum of all entries of each column is 1 and there are no negative entries.**

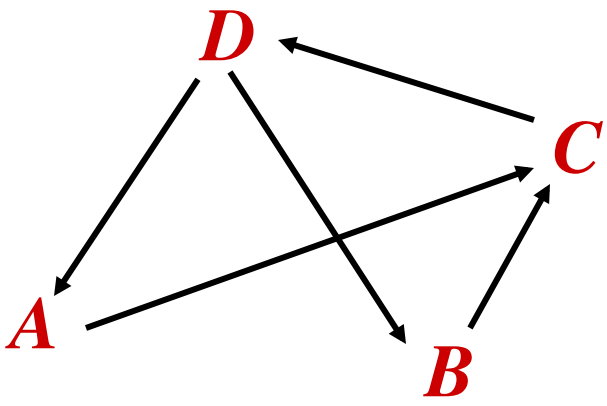
Therefore, if M is replaced by M* as in

$$R_i = M^* \times R_{i-1}$$

then the convergence is guaranteed

Computing PageRank (cont'd)

Example: Suppose the Web graph is:



$$M = \begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & ½ \\ 0 & 0 & 0 & ½ \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

Computing PageRank (cont'd)

Example (continued): Suppose d = 0.8. All entries in Z are 0 and all entries in K are ¼.

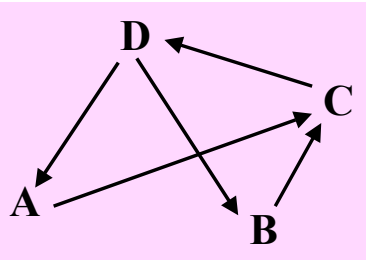
$$M^* = 0.8\,(M+Z) + 0.2\,K = \begin{pmatrix} 0.05 & 0.05 & 0.05 & 0.45 \\ 0.05 & 0.05 & 0.05 & 0.45 \\ 0.85 & 0.85 & 0.05 & 0.05 \\ 0.05 & 0.05 & 0.85 & 0.05 \end{pmatrix}$$

Compute rank by iterating

R := M* x R

MATLAB says:
R(A)=.338 (.176)
R(B)=.338 (.176)
R(C)=.6367 (.332)
R(D)=.6052 (.315)

*Eigen decomposition gives the *unit* vector.. To get the "probabilities" just normalize by dividing every number by the sum of the entries..*



PageRank

- Computation
  - expensive
  - but done once for all docs/queries

- Bad: Query-independent!
- Hard to spam

To make up for this, combine IR-style similarity measure with global page rank:

ranking_score(query q, page p)

= w*sim(q, p) + (1-w) * R(p), if sim(q, p) > 0

= 0, otherwise where 0 < w < 1 (*empirical*)