Link Analysis (Part I)

With slide contributions from
J. Leskovec, Anand Rajaraman, Jeffrey D. Ullman

OUTLINE

Problem and Motivation

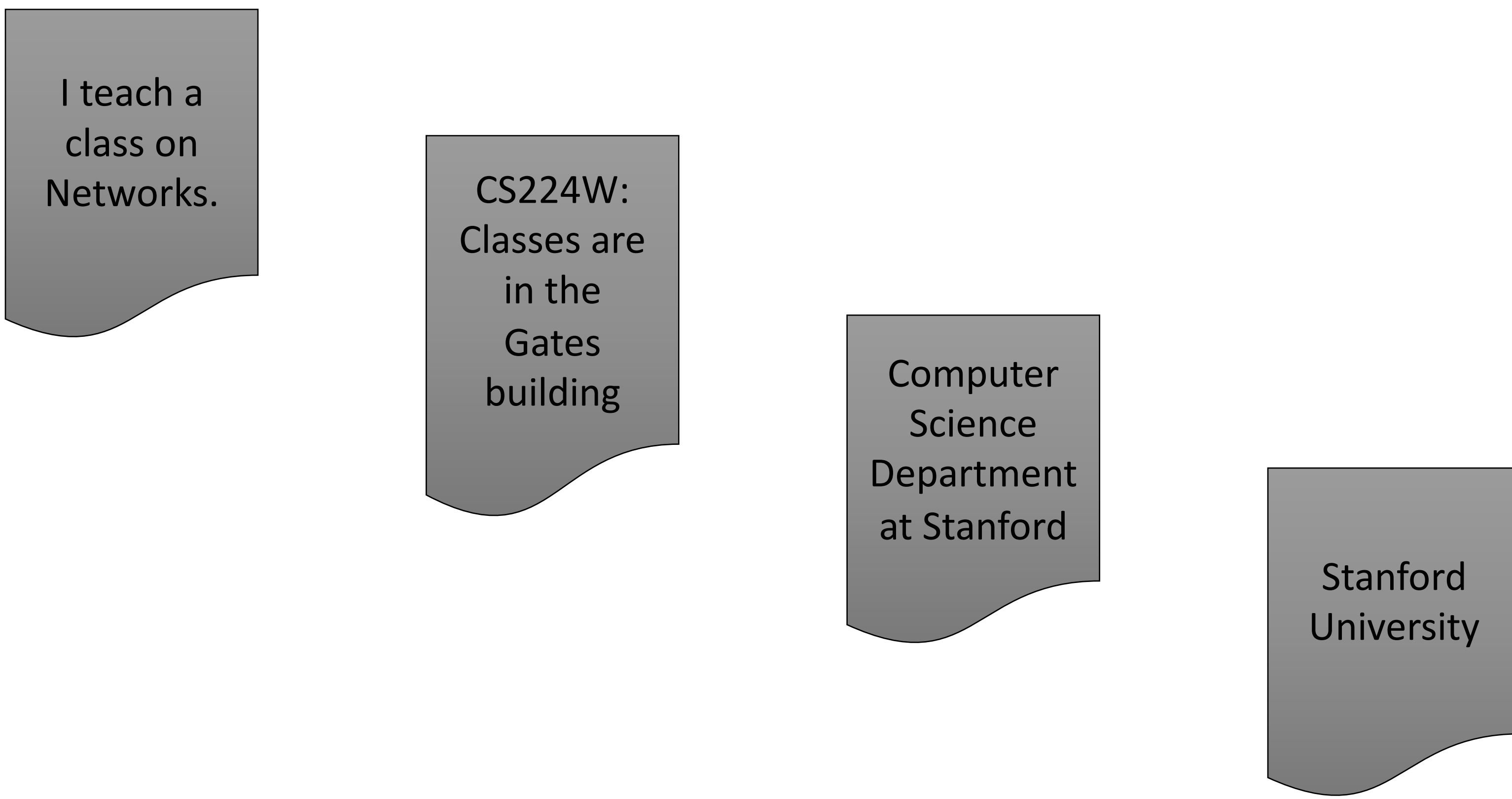
- How to rank a web page?

Three Approaches

1. Page Rank
2. Topic-Specific (personalized) Page Rank
3. Web Spam Detection

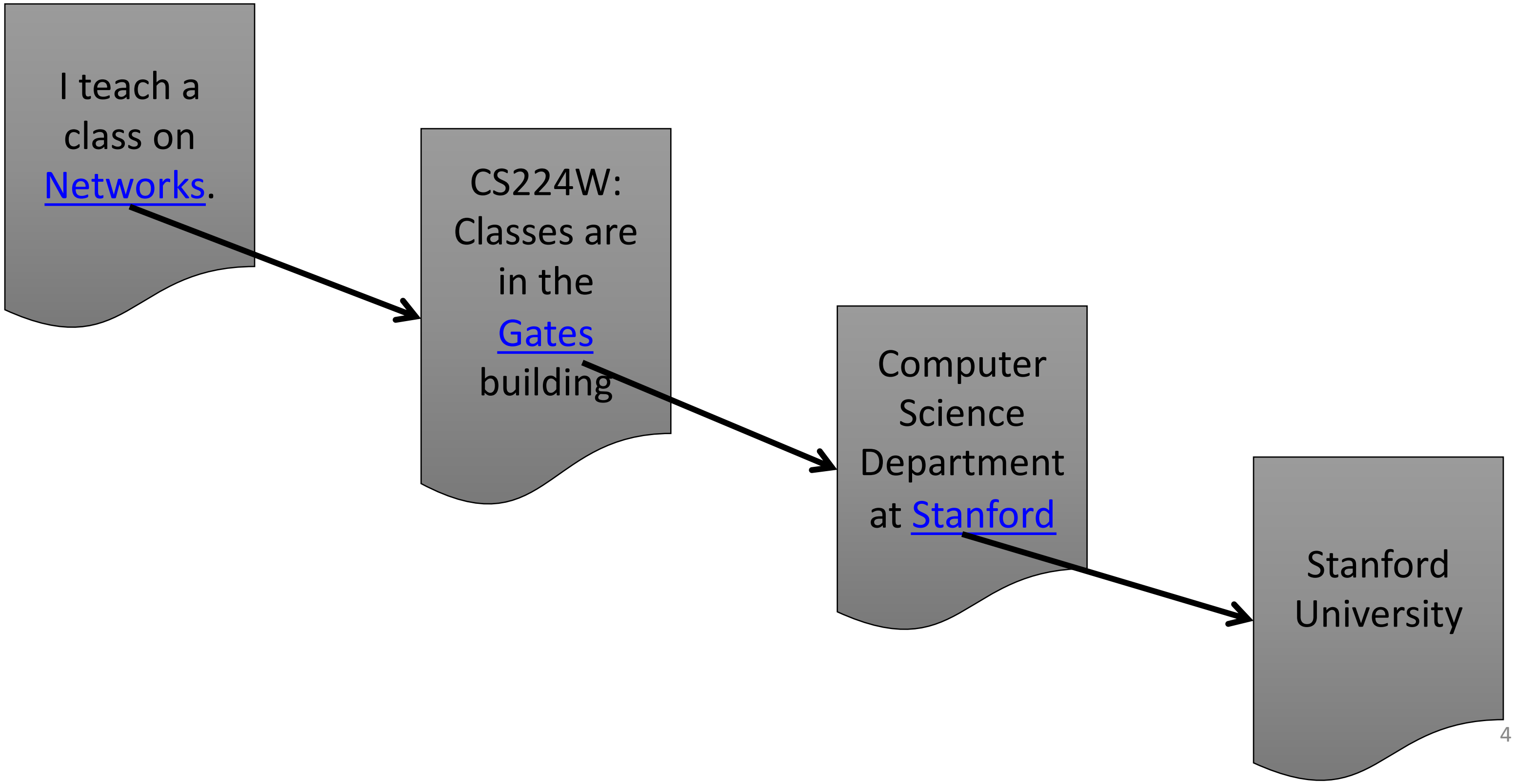
Web as a Graph

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

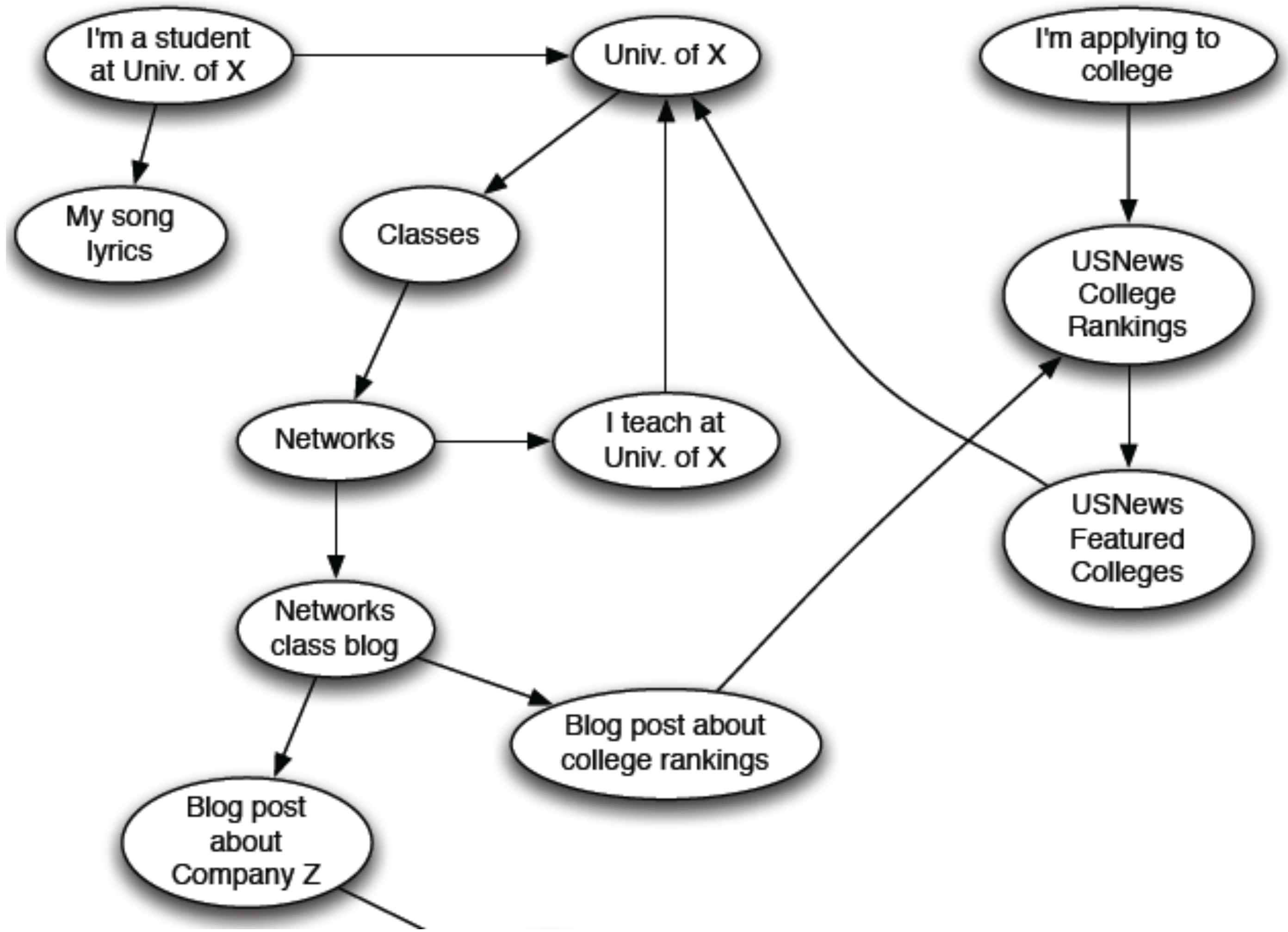


Web as a Graph

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

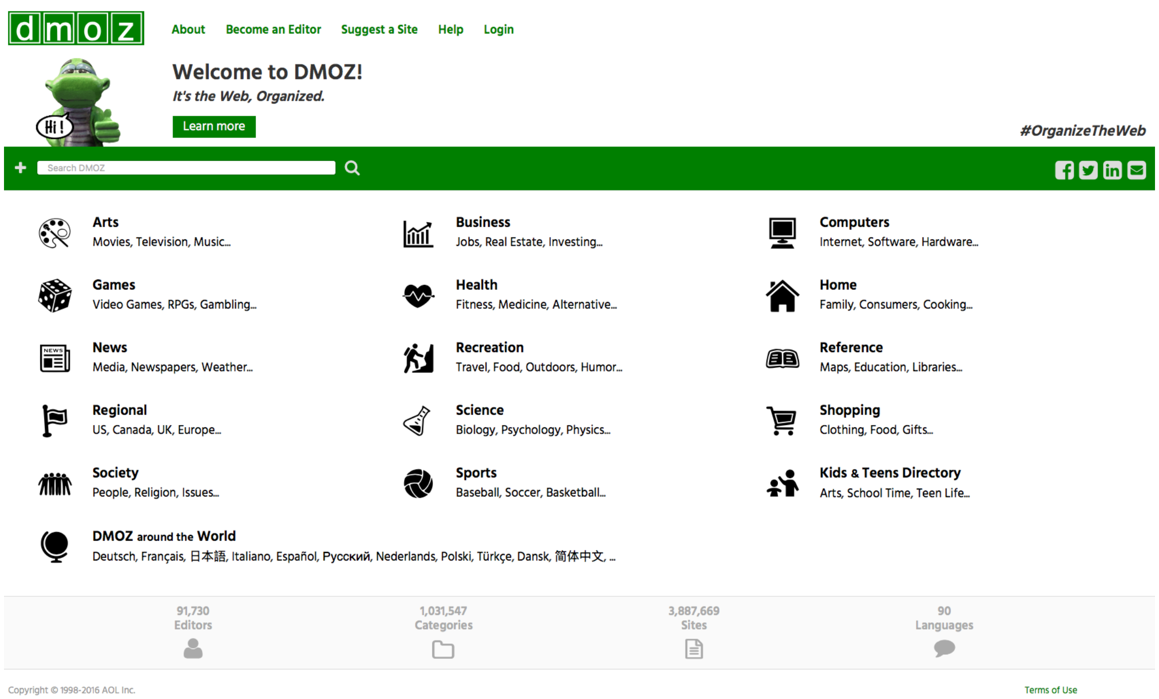


Web as a Directed Graph



Broad Question

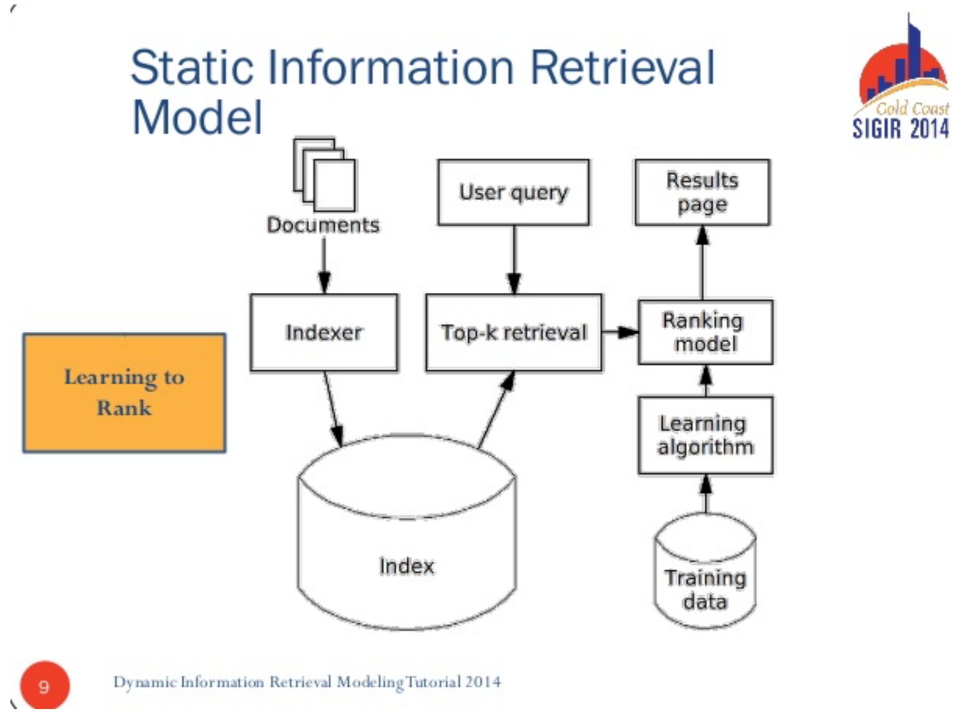
- **How to organize the Web?**
  - **First try:** Human curated **Web directories**
    - Yahoo, DMOZ, LookSmart



Broad Question (Cont'd)

•How to organize the Web?

- **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.

Early Web Search

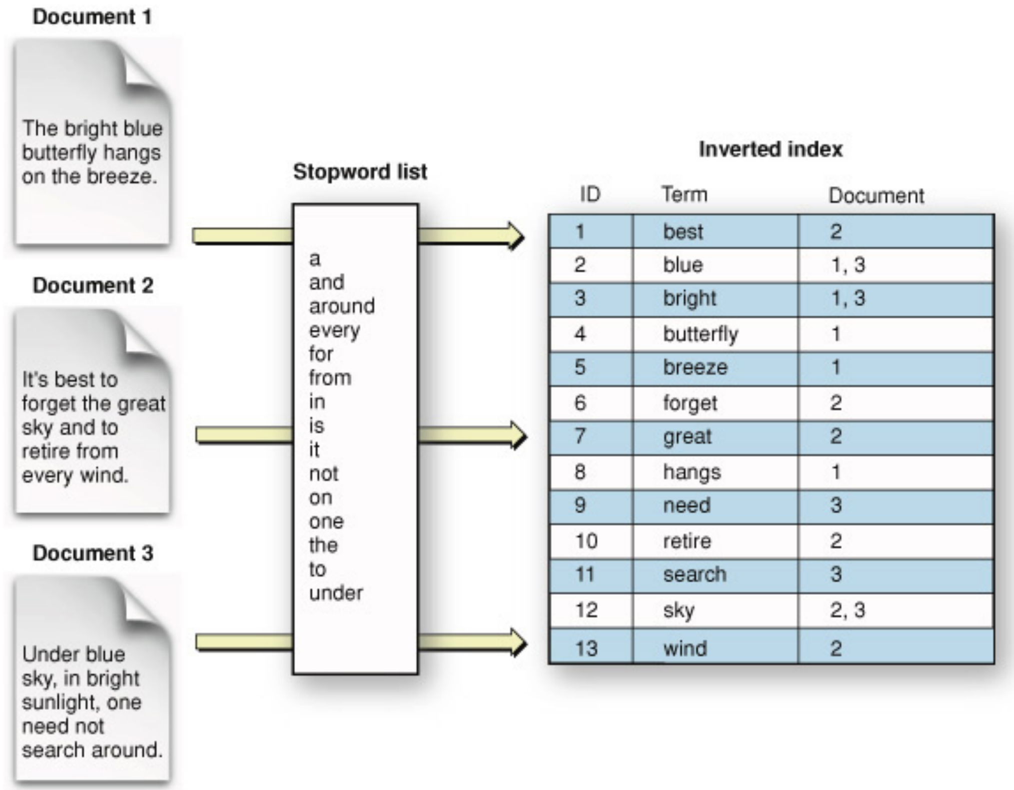
- Keywords extracted from web pages
  - E.g., title, content
  - Used to build inverted index

- Queries are matched with web pages
  - Via lookup in the inverted index
  - Pages ranked by occurrences of query keywords

```
"a":      {2}
"banana": {2}
"is":     {0, 1, 2}
"it":     {0, 1, 2}
"what":   {0, 1}
```

Inverted Index

- Problem: susceptible to term spam



ID	Term	Document
1	best	2
2	blue	1, 3
3	bright	1, 3
4	butterfly	1
5	breeze	1
6	forget	2
7	great	2
8	hangs	1
9	need	3
10	retire	2
11	search	3
12	sky	2, 3
13	wind	2

https://developer.apple.com/library/mac/documentation/UserExperience/Conceptual/SearchKitConcepts/searchKit_basics/searchKit_basics.html

Term Spam

- Disguise a page as something it is not about
  - E.g., adding thousands of keyword “movies”
  - Actual content may be some advertisement
  - Fool search engine to return it for query “movies”

- May even <u>fade spam words into background</u>

- Spam pages may be based on <u>top-ranked pages</u>

Web Search: Two Challenges

Two challenges of web search:

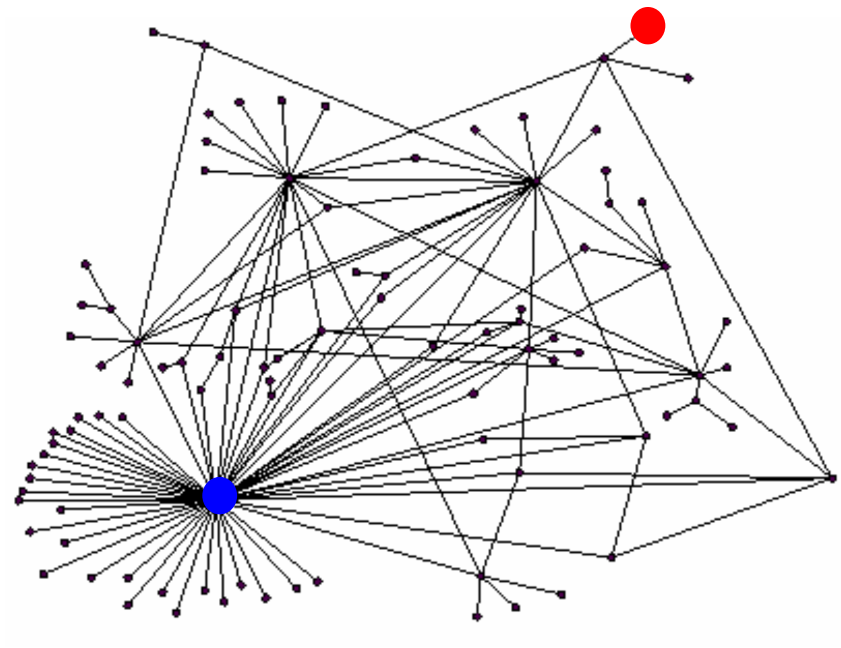
- **(1) Web contains many sources of information**
  Who to "trust"?
  - **Trick:** Trustworthy pages may point to each other!
- **(2) What is the "best" answer to query "newspaper"?**
  - No single right answer
  - **Trick:** Pages that actually know about newspapers might all be pointing to many newspapers

Ranking Nodes on the Graph

- **All web pages are not equally "important"**

  blog.bob.com vs. www.usc.edu

- There is large diversity in the web-graph node connectivity
- **Let's rank the pages by the link structure!**



Link Analysis Algorithms

- We will cover the following **Link Analysis approaches** for computing **importance** of nodes in a graph:
  - Page Rank
  - Topic-Specific (Personalized) Page Rank
  - Web Spam Detection Algorithms

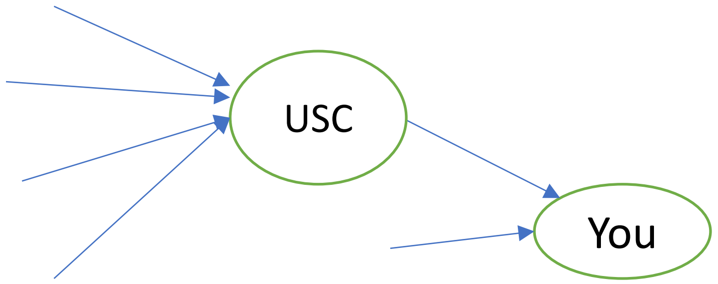
PageRank:
The “Flow” Formulation

PageRank: Combating Term Spam

- Key idea: rank pages by linkage too
  - How many pages point to a page
  - How important these pages are

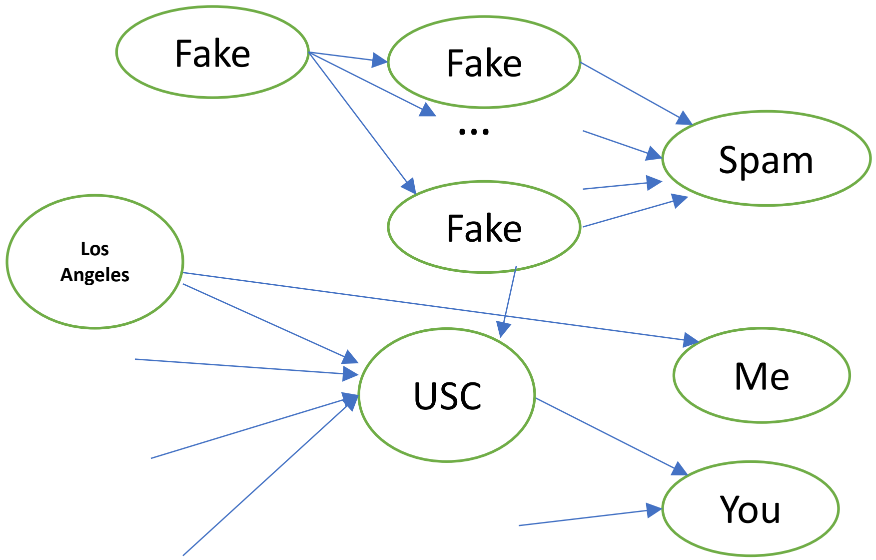
  => PageRank

- USC.edu can be important
  - because many pages point to it
- Your home page can be important
  - If it is pointed to by USC ☺



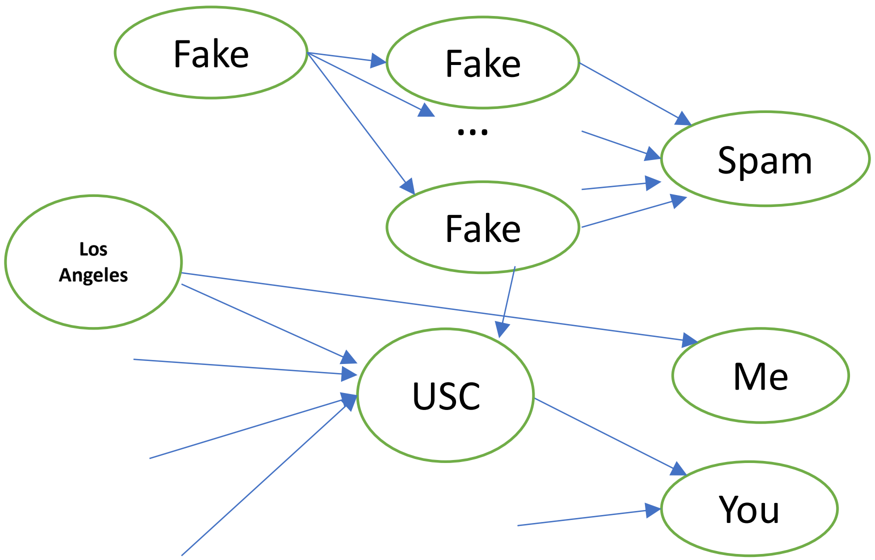
Random Surfer Model

- Random surfer of web
  - starts from any page
  - follows its outgoing links randomly
- Page is important if it attracts a large # of surfers



PageRank

- Probability that a random surfer lands on the page



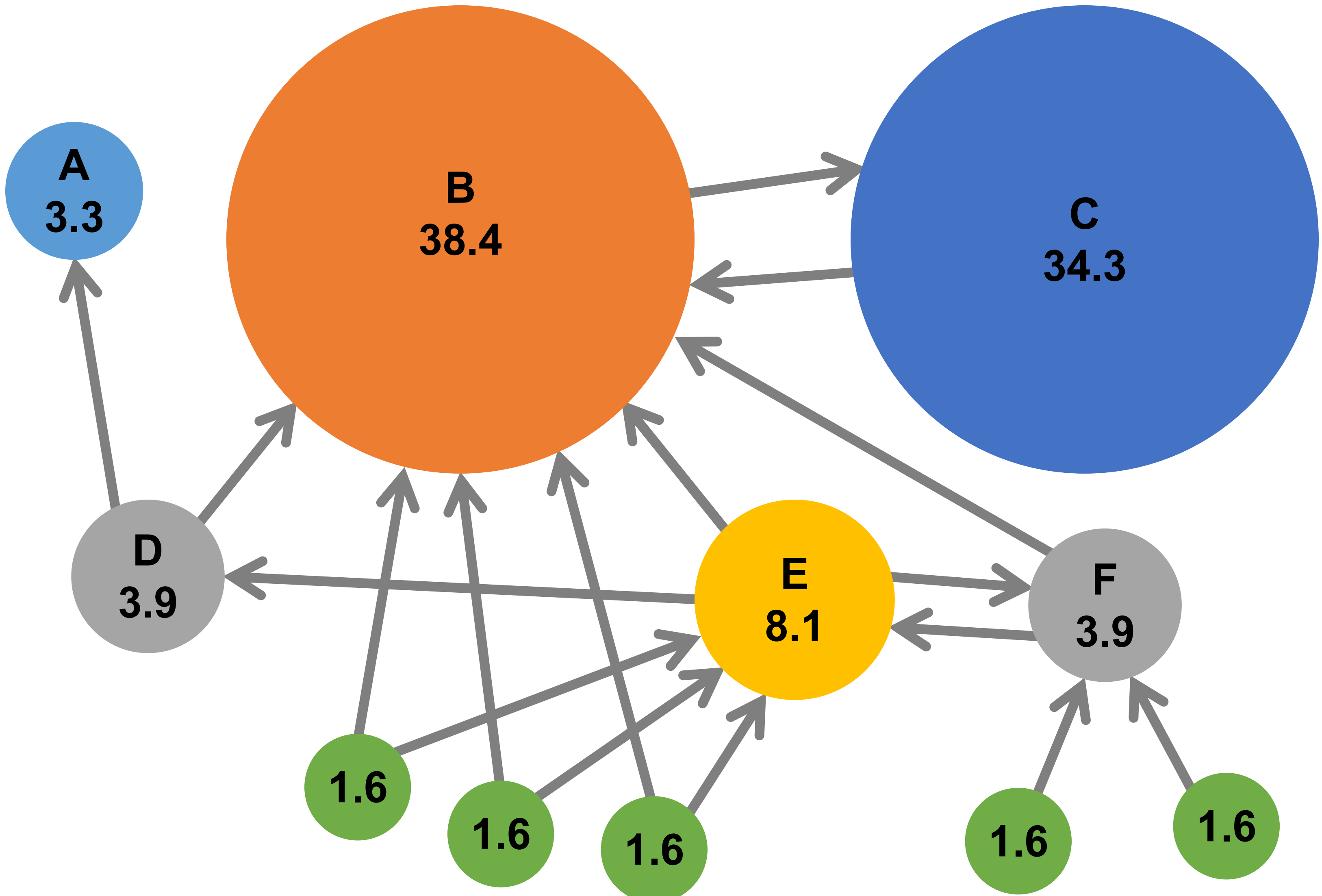
Intuition

- If a page is important, then
  - many other pages may directly/indirectly link to it
  - random surfer can easily find it

- Spam pages are less connected
  - So less chance to attract random surfer

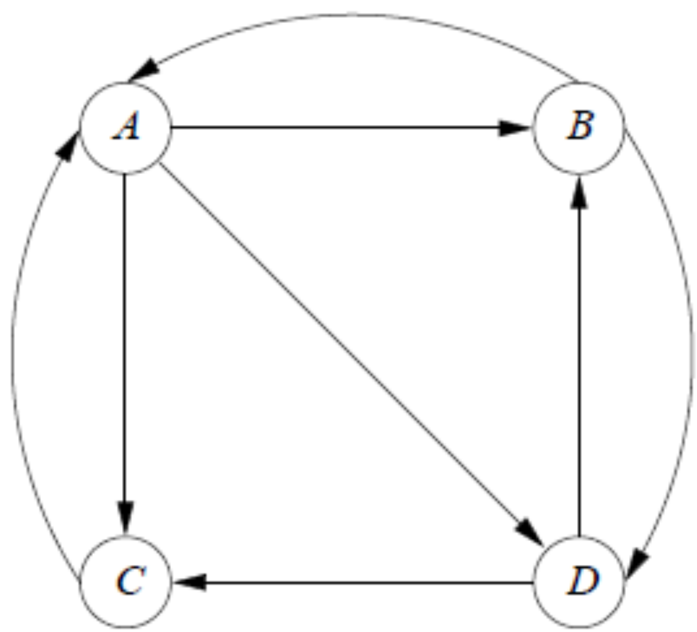
- Random surfer model more robust than manual approach
  - A collective voting scheme

Example: PageRank Scores



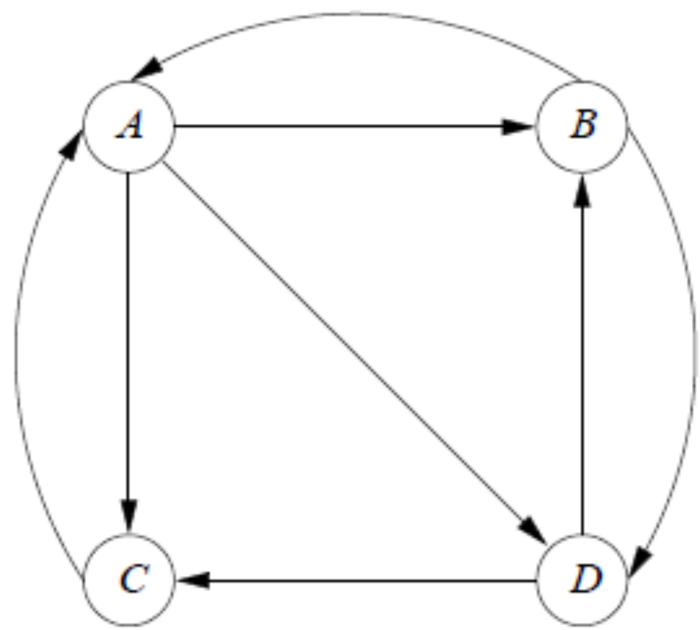
Assumption: A Strongly Connected Web Graph

- Nodes = pages
- Edges = hyperlinks between pages
- every node is reachable from every other node



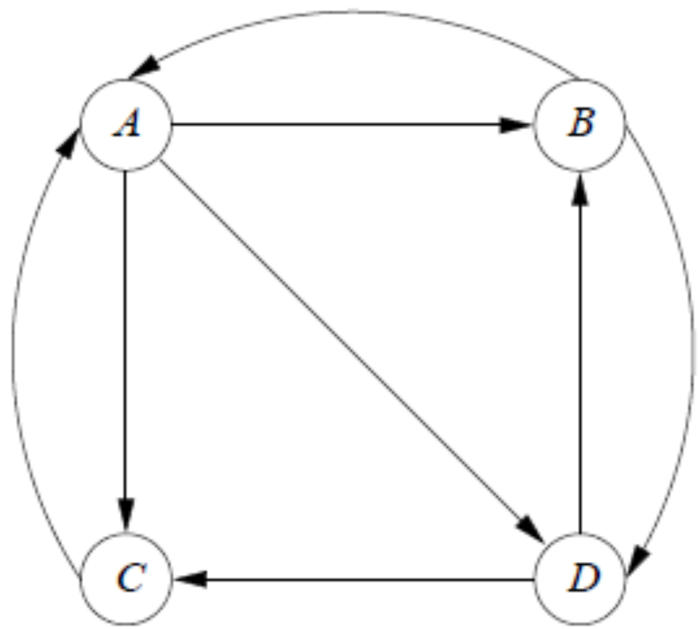
Model: Random Surfer on the Graph

- Can start at any node, say A
  - Can next go to B, C, or D, each with 1/3 prob.
  - If at B, can go to A and D, each with 1/2 prob.
  - So on…



Random Surfer Property: Memoryless

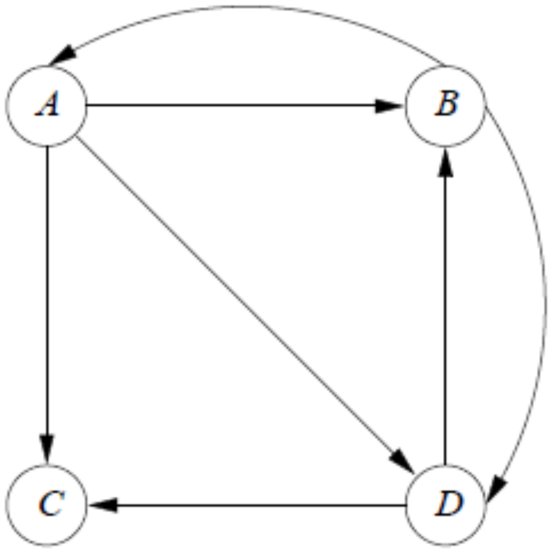
- Where to go from node X is not affected by how the surfer got to X



Extreme Case: Dead End

- Dead end: a page with no edges out
  - Absorb PageRanks
  - PageRank → 0 for any page that can reach the dead end (including the dead end itself)

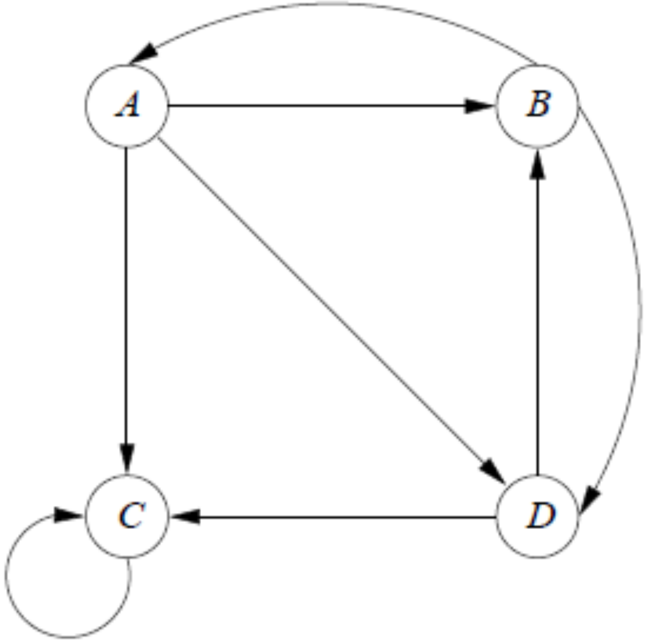
Dead end



Extreme Case: Spider Trap

- Group of pages with no edges going out of group
  - Absorb all PageRanks (rank of C → 1, others → 0)
  - Surfer can never leave, once trapped
  - Can have > 1 nodes

Spider trap



PageRank: Formulation Details

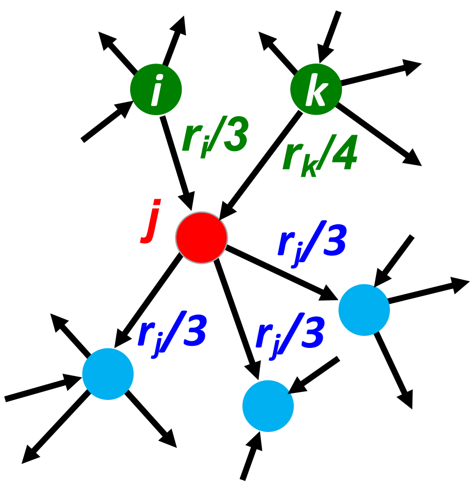
PageRank: Links as Votes

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

Simple Recursive Formulation

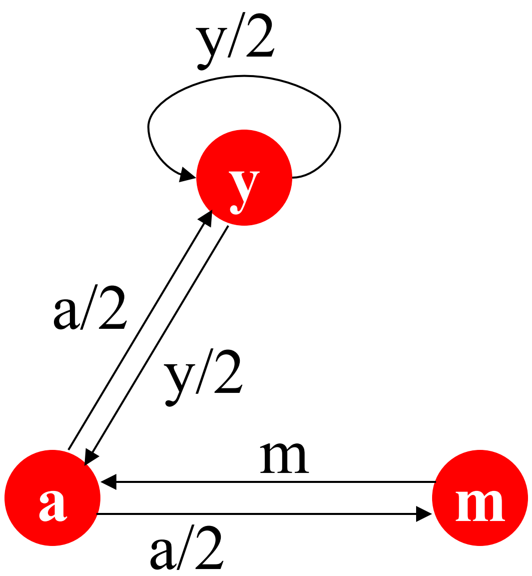
- Each link's vote is proportional to the **importance** of its source page
- If page j with importance r_j has n out-links, each link gets r_j / n votes
- Page j's own importance is the sum of the votes on its in-links

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



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**



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

"Flow" equations:

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

d$_i$ **= out-degree of node *i***

Solving the flow equations

"Flow" equations:

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

- **3 equations, 3 unknowns, no constants**
  - No unique solution
  - All solutions equivalent modulo scale factor
- **Additional constraint forces uniqueness**
  - **Add some constant constraints: e.g., $r_y + r_m + r_a = 1$**
  - Solve for unique solutions: $r_y = 2/5$, $r_a = 2/5$, $r_m = 1/5$
- **Gaussian elimination method (in the later slides) works for small examples, but we need a better method for large graphs**

$$\left[\begin{array}{ccc|c} 1 & 3 & 1 & 9 \\ 1 & 1 & -1 & 1 \\ 3 & 11 & 5 & 35 \end{array}\right] \to \left[\begin{array}{ccc|c} 1 & 3 & 1 & 9 \\ 0 & -2 & -2 & -8 \\ 0 & 2 & 2 & 8 \end{array}\right] \to \left[\begin{array}{ccc|c} 1 & 3 & 1 & 9 \\ 0 & -2 & -2 & -8 \\ 0 & 0 & 0 & 0 \end{array}\right] \to \left[\begin{array}{ccc|c} 1 & 0 & -2 & -3 \\ 0 & 1 & 1 & 4 \\ 0 & 0 & 0 & 0 \end{array}\right]$$

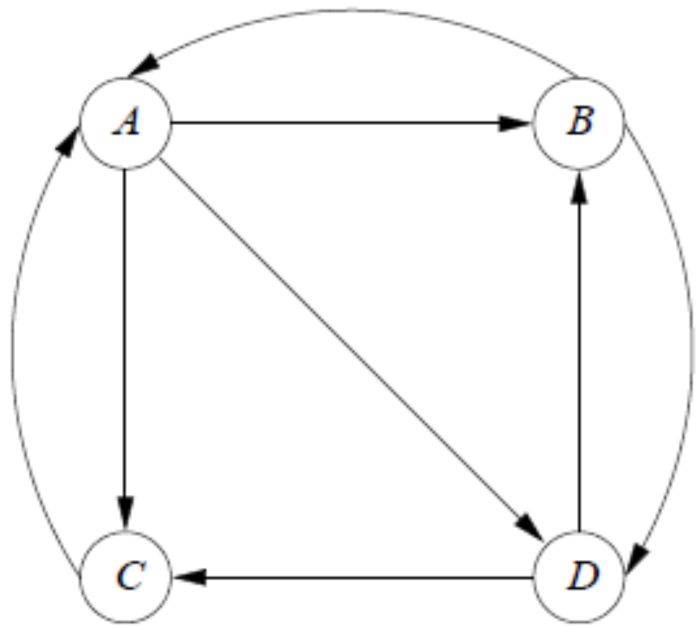
PageRank: Matrix formulation

- **Stochastic Transition (or adjacency) Matrix M**
- **Suppose page j has n outlinks**
  - If outlink j -> i, then $\mathbf{M_{ij}=1/n}$
  - Else $\mathbf{M_{ij}=0}$
- **M** is a **column stochastic matrix**
  - Columns sum to 1

Transition Matrix

- M[i,j] = prob. of going from node j to node i
  - If j has k outgoing edges, prob. for each edge = 1/k

$$
M = \begin{array}{c} \\ A \\ B \\ C \\ D \end{array}
\begin{array}{c}
\begin{array}{cccc} A & B & C & D \end{array} \\
\begin{bmatrix}
0 & 1/2 & 1 & 0 \\
1/3 & 0 & 0 & 1/2 \\
1/3 & 0 & 0 & 1/2 \\
1/3 & 1/2 & 0 & 0
\end{bmatrix}
\end{array}
$$

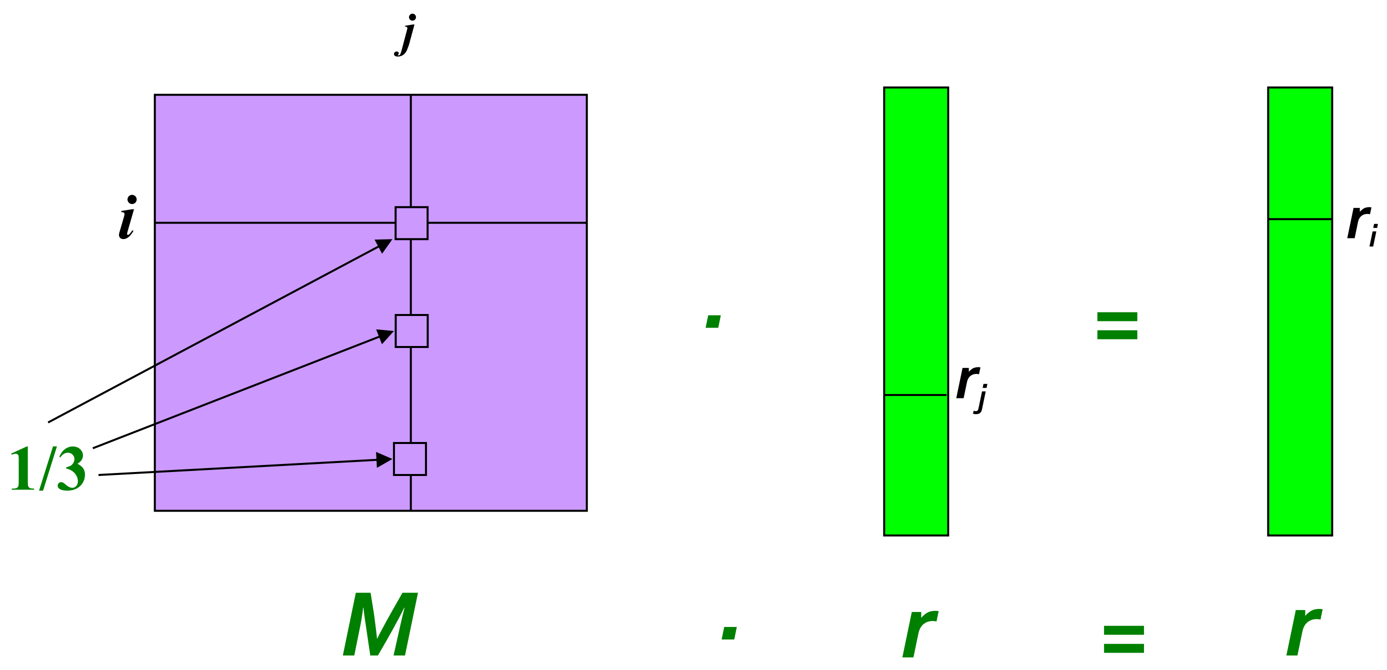


PageRank: Matrix formulation (Cont'd)

- **Stochastic Transition (or adjacency) Matrix M**
- **Suppose page j has n outlinks**
  - If outlink j -> i, then $\mathbf{M_{ij}=1/n}$
  - Else $\mathbf{M_{ij}=0}$
- **M** is a **column stochastic matrix**
  - Columns sum to 1
- **Rank vector r** is a vector with one entry per web page
  - **r_i is the importance score of page I**
- The flow equations can be written as **r = Mr**

Example

- **Flow equation in matrix form: Mr = r**
- Suppose page *j* links to 3 pages, including *i*



Stationary Distribution

- Limiting prob. distribution of random surfer
  - PageRanks are based on limiting distribution
  - the probability destribution will converge eventually
- Requirement for its existence
  - Graph is strongly connected: a node can reach any other node in the graph

    => Cannot have dead ends, spider traps

Eigenvectors and Eigenvalues

- An **eigenvector** of a square matrix **A** is a non-zero vector **v** that, when the matrix multiplies **v**, yields the same as when some scalar multiplies **v**, the scalar multiplier often being denoted by λ
- That is:

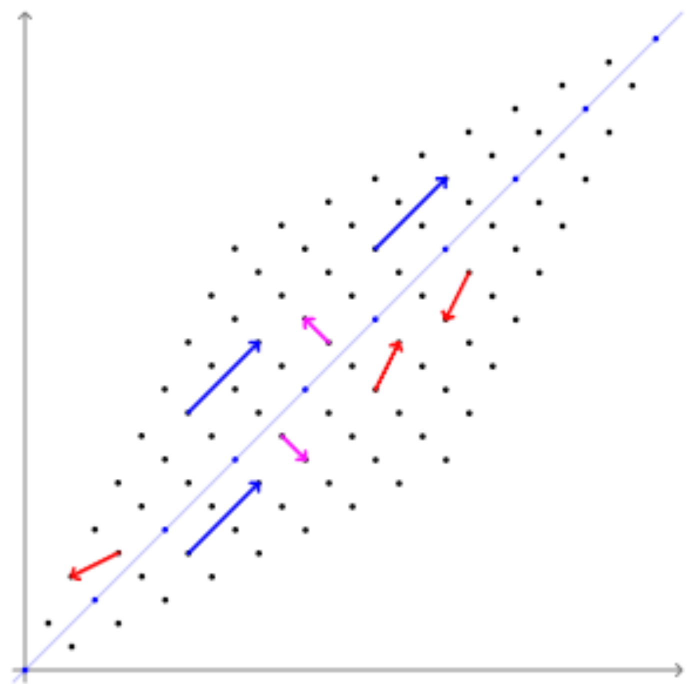
$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$$

- The number λ is called the **eigenvalue** of **A** corresponding to **v**

Eigenvalues and Eigenvectors Example

- The transformation matrix $A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$ preserves the direction of vectors parallel to $\mathbf{v} = (1,-1)^T$ (in purple) and $\mathbf{w} = (1,1)^T$ (in blue). The vectors in red are not parallel to either eigenvector, so, their directions are changed by the transformation.

$$A\mathbf{v} = \lambda\mathbf{v}$$



http://setosa.io/ev/eigenvectors-and-eigenvalues/

https://en.wikipedia.org/wiki/Eigenvalues_and_e
igenvectors

Eigenvector Formulation

- **The flow equations can be written**

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

limiting distribution

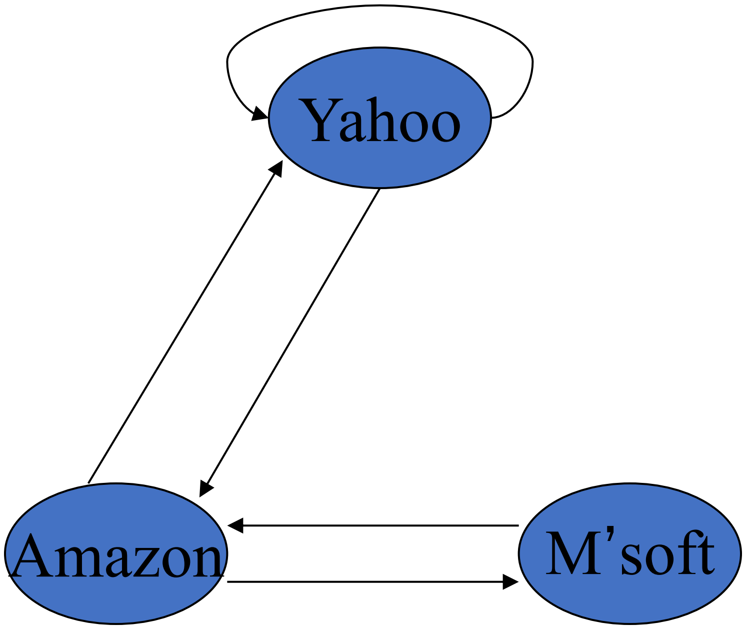
- So the **rank vector** ***r*** is an **eigenvector** of the stochastic web matrix ***M***
  - ***r*** is M's first or principal eigenvector, with corresponding eigenvalue ***1***
  - Largest eigenvalue of ***M*** is **1** since ***M*** is column stochastic (with non-negative entries)
    - *We know* ***r*** *is unit length and each column of* ***M*** *sums to one*

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

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

- **We can now efficiently solve for *r*!**
  - **1. Power Iteration:** https://en.wikipedia.org/wiki/Power_iteration
  - **2. Use the principal eigenvector**

Example



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

$$r_a = r_y/2 + r_m$$

$$r_m = r_a/2$$

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

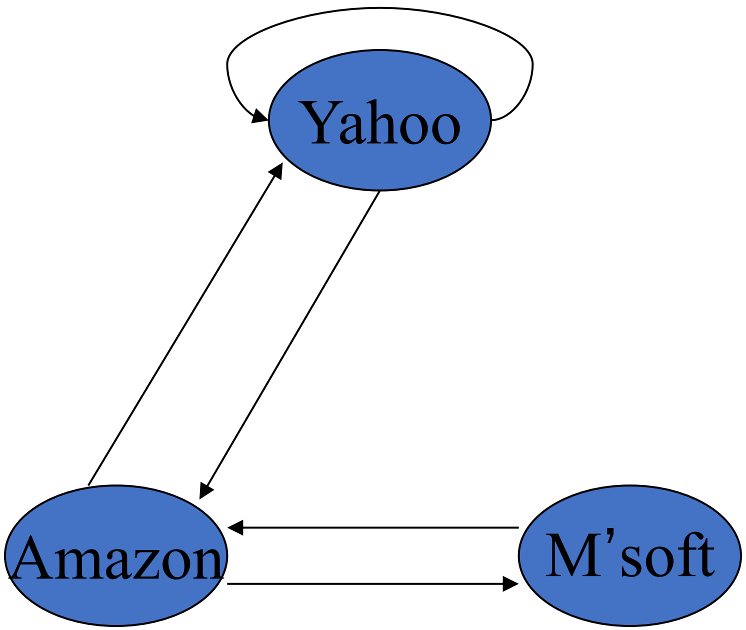
r = Mr

$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix} \begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix}$$

Power Iteration method

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

Power Iteration Example



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

$\mathbf{r}^{k+1} = \mathbf{M}\mathbf{r}^{k}$

$$\begin{matrix} r_y \\ r_a \\ r_m \end{matrix} = \begin{matrix} 1/3 \\ 1/3 \\ 1/3 \end{matrix} \quad \begin{matrix} 1/3 \\ 1/2 \\ 1/6 \end{matrix} \quad \begin{matrix} 5/12 \\ 1/3 \\ 1/4 \end{matrix} \quad \begin{matrix} 3/8 \\ 11/24 \\ 1/6 \end{matrix} \quad \ldots \quad \begin{matrix} 2/5 \\ 2/5 \\ 1/5 \end{matrix}$$