Data Mining
PageRank

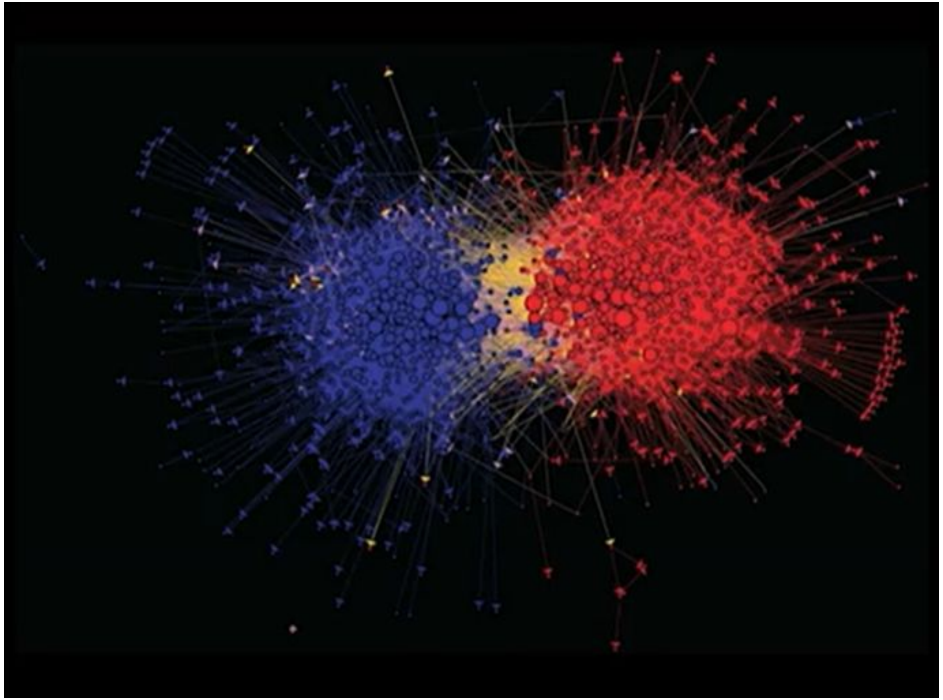
Joe Burdis
Fall 2024
CUNY Graduate Center

Graph Data: Social Networks

Reference: Mining of Massive Datasets, Chapter 5. http://mmds.org

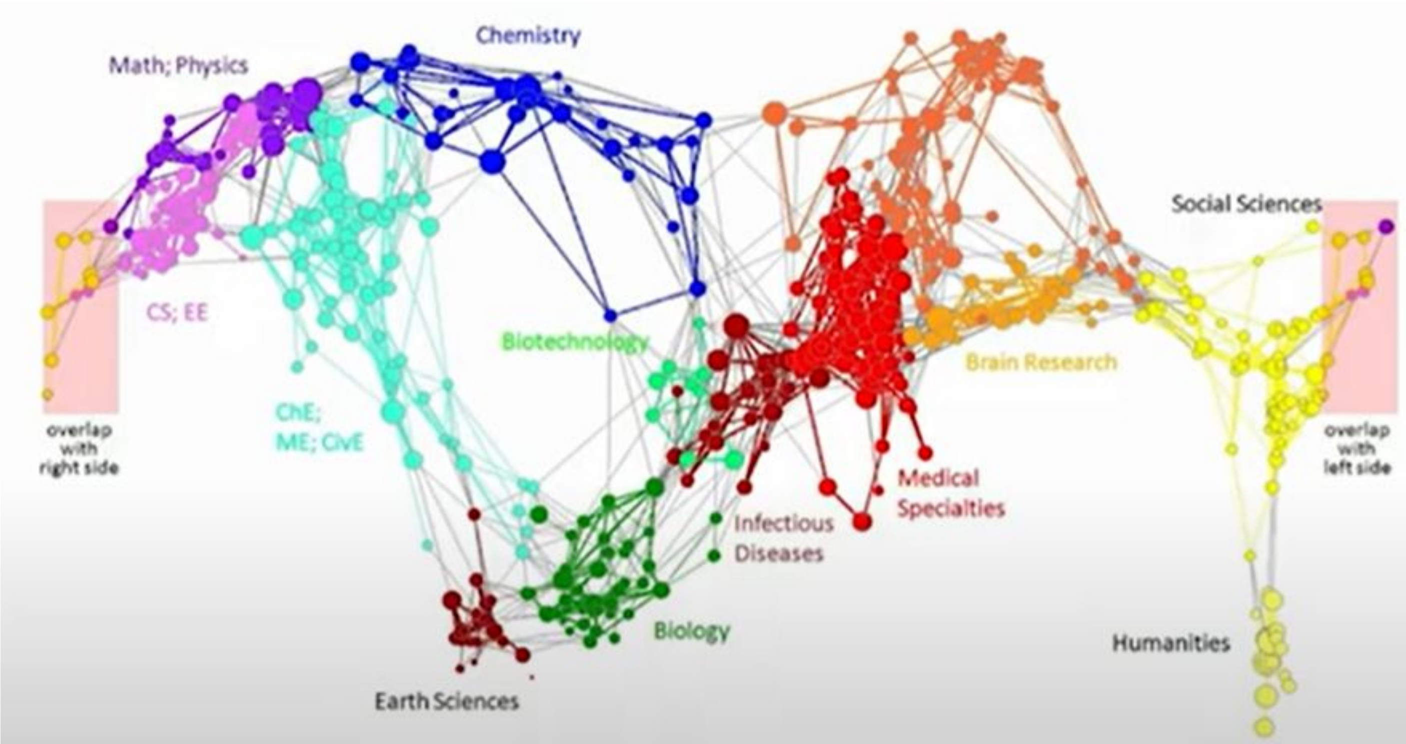
Graph Data: Media Networks

Polarization of the network [Adamic-Glance, 2005]



Graph Data: Information Networks

Citation networks and maps of science [Borner et al., 2012]



Graph Data

- Computer network
- Road network
- Power network
- The internet
    - Nodes: webpages
    - Edges: hyperlinks
    - Directed graph

How Is the Web Organized?

- First attempt (Yahoo): Web directories
  - Keywords
  - Word frequencies
  - Multimedia content
- Second attempt: Web search
  - Which webpages are relevant?
  - Which webpages are trustworthy?
  - Idea: ranking webpages through the links

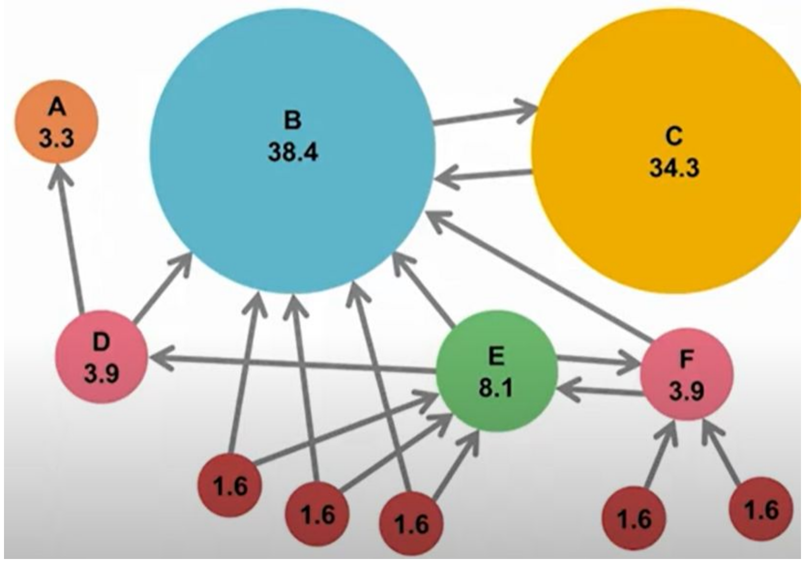
PageRank

- Initial idea
- Mathematical formulation
- Theoretical concerns
- Implementation techniques

Idea: Links as Votes

- Page is more important if it has more links.
- In-coming links are more important than out-going links.
- Links from important pages count more.

Challenge: it is a recursive question!

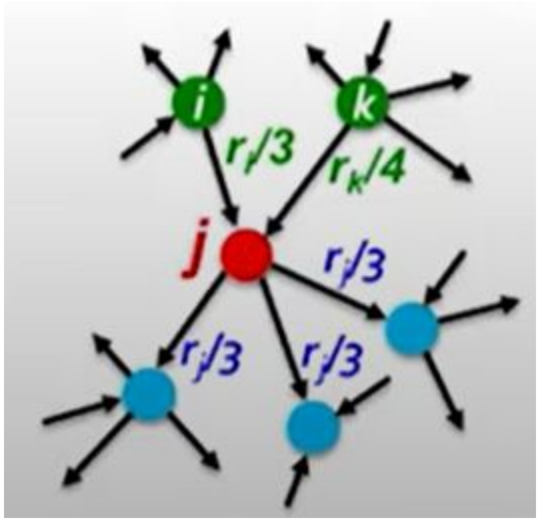


Simple Recursive Formulation

- 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.

Exercise: How to express r_j?

$r_j = r_i / 3 + r_k / 4$



Properties of 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.
- The rank of page j is defined as

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

Challenge: How to solve for the ranks?

Example

What formulas can be constructed based on this graph?

$y = y / 2 + a / 2$

$a = y / 2 + m$

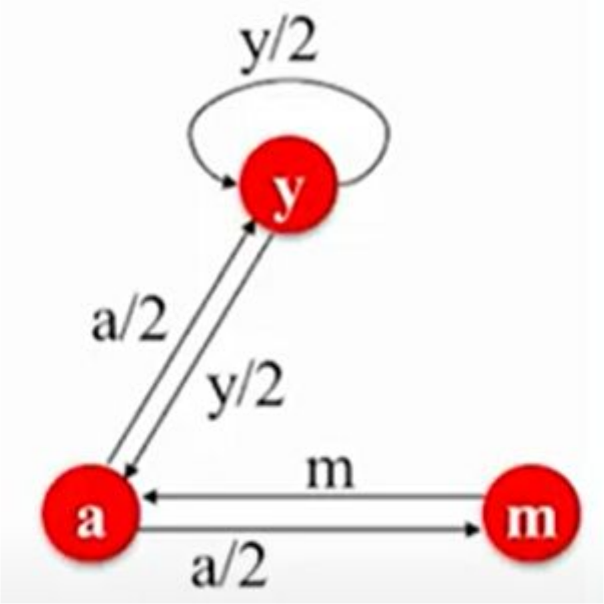
$m = a / 2$

Matrix formulation:

$$\begin{vmatrix} y \\ a \\ m \end{vmatrix} = \begin{vmatrix} ½ & ½ & 0 \\ ½ & 0 & 1 \\ 0 & ½ & 0 \end{vmatrix} . \begin{vmatrix} y \\ a \\ m \end{vmatrix}$$

It takes the form of

$r = M * r$.



Solving the Flow Equations

- 3 equations, 3 unknowns, no constraints
- No unique solution
- An additional constraint forces uniqueness.
- Add $y + a + m = 1$ (the sum of all page ranks is 1).
- Solver: Gaussian elimination?
- Limitation of Gaussian elimination

The Power Iteration Method

The power iteration algorithm

- Input: An N*N matrix M, a threshold eps > 1
- Initialize: $r_0 = [1/N, \ldots, 1/N]^T$
- Iterate: $r_{t+1} = M * r_t$
- Stop when $|r_{t+1} - r_t| < eps$

What can the power iteration achieve? In the end, $r = M * r$

What if we apply the power iteration to the PageRank matrix?

Answer: The vector r returned by power iteration will be the pagerank vector.

Random Walk Interpretation

Imagine a random web surfer:

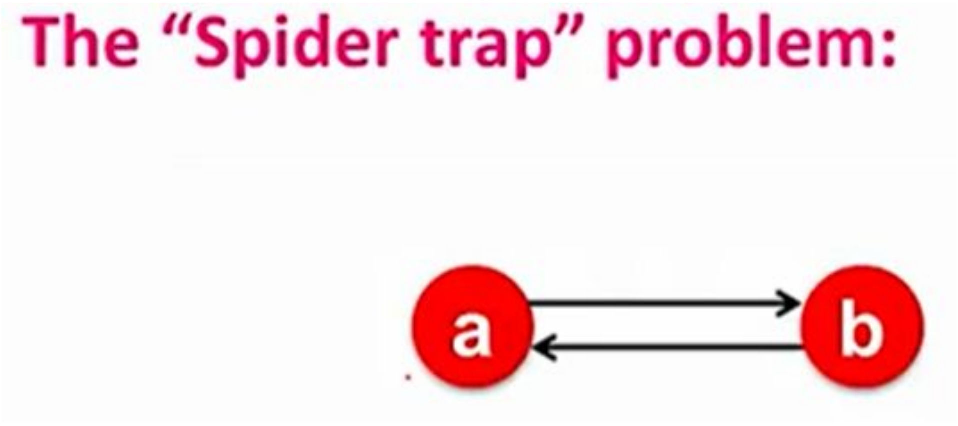
- At any time, surfer is on some page i
- At time t+1, the surfer follows an out-link from i uniformly at random
- Ends up on some page j linked from i
- Repeat this process infinitely

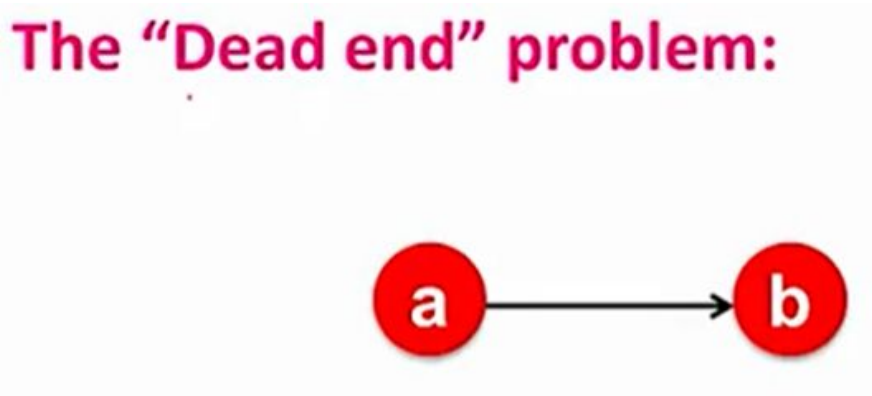
The page rank is equal to the probability that the surfer is at page i at a given time.

The power iteration methods is a simulation of such random walk.

Theoretical Concerns

- Can the power iteration always stop regardless of the input matrix?
- Is the solution unique?





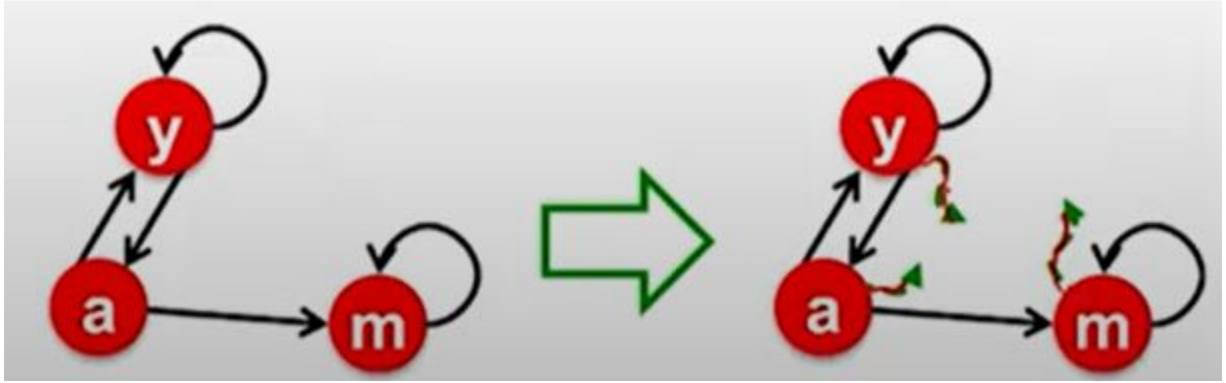
Problems for PageRank Formulation

- Some pages are dead ends (have no out-links)
- Such pages causes importance to “leak out”
- Some pages only have out-links within a certain group
- Such pages eventually absorb all importance

Solution for Spider Traps

The Google solution for spider traps: At each time step, the random surfer has two options:

- With probability p, follow a link at random
- With probability 1 - p, jump to some random page (teleport)
- Common values for p are in the range 0.8 to 0.9



Solution for Dead-Ends

Solution: Perform a random teleport with probability 1.0 from any dead-end.

Implementation Issues

The key step of the power iteration method is the matrix-vector multiplication

$$r_{t+1} = A * r_t$$

The matrix is too large to fit in memory.

- Suppose N = 1 billion pages
- We need 4 bytes for each entry
- 2 billion entries for the two vectors, approximately 8 GB
- Matrix A has N^2 entries!
- The matrix is dense thanks to random teleports

Simplify the Computation

Observation: The random teleport is equivalent to:

- Tax each page rank with a fraction (1 - p)
- And redistribute the taxed importance evenly to other nodes.

Without the probabilities generated from random teleports, the link matrix is a highly sparse matrix. Computation with sparse matrix is much simpler.

Mathematical Proof

- $\boldsymbol{r} \;= \boldsymbol{A} \cdot \boldsymbol{r}$, **where** $\boldsymbol{A_{ji} = \beta\, M_{ji} + \frac{1-\beta}{N}}$
- $r_j \;= \sum_{i=1}^{N} A_{ji} \cdot r_i$
- $r_j \;= \sum_{i=1}^{N} \left[\beta\, M_{ji} + \frac{1-\beta}{N}\right] \cdot r_i$

$$= \sum_{i=1}^{N} \beta\, M_{ji} \cdot r_i + \frac{1-\beta}{N} \sum_{i=1}^{N} r_i$$

$$= \sum_{i=1}^{N} \beta\, M_{ji} \cdot r_i + \frac{1-\beta}{N} \qquad \text{since } \sum r_i = 1$$

- **So we get:** $\boldsymbol{r = \beta\, M \cdot r + \left[\frac{1-\beta}{N}\right]_N}$

The Complete PageRank Algorithm

- Input: Graph G and parameter β
  - Directed graph G (can have **spider traps** and **dead ends**)
  - Parameter β
- Output: PageRank vector r^{new}
  - **Set:** $r_j^{old} = \frac{1}{N}$
  - **repeat until convergence:** $\sum_j \left| r_j^{new} - r_j^{old} \right| > \varepsilon$
    - $\forall j$: $\boldsymbol{r'}_j^{new} = \sum_{i \to j} \boldsymbol{\beta} \frac{\boldsymbol{r}_i^{old}}{\boldsymbol{d}_i}$

      $\boldsymbol{r'}_j^{new} = \boldsymbol{0}$ if in-degree of $\boldsymbol{j}$ is **0**
    - Now re-insert the leaked PageRank:

      $\forall \boldsymbol{j}$: $\boldsymbol{r}_j^{new} = \boldsymbol{r'}_j^{new} + \frac{\boldsymbol{1-S}}{\boldsymbol{N}}$ where: $S = \sum_j r'^{new}_j$
    - $\boldsymbol{r}^{old} = \boldsymbol{r}^{new}$

Complexity of PageRank

- If matrix M is too large to fit in memory but r fits: is 2|r| + |M|.

Source	Degree	Destinations
A	3	B, C, D
B	2	A, D
C	1	A
D	2	B, C

Figure 5.11: Represent a transition matrix by the out-degree of each node and the list of its successors

- If the memory can only load one kth of r: (1 + eps) |M| + (k + 1) |r|

$$\begin{bmatrix} \mathbf{v}'_1 \\ \mathbf{v}'_2 \\ \mathbf{v}'_3 \\ \mathbf{v}'_4 \end{bmatrix} = \begin{bmatrix} M_{11} & M_{12} & M_{13} & M_{14} \\ M_{21} & M_{22} & M_{23} & M_{24} \\ M_{31} & M_{32} & M_{33} & M_{34} \\ M_{41} & M_{42} & M_{43} & M_{44} \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \\ \mathbf{v}_4 \end{bmatrix}$$

Figure 5.12: Partitioning a matrix into square blocks

Issues with PageRank

- Measures generic popularity of a page
  - Biased against topic-specific authorities
  - Solution: Topic-Specific PageRank
- Uses a single measure of importance
  - Other models of importance
  - Solution: Hubs-and-Authorities
- Susceptible to Link Spam
  - Artificial link topographies created in order to boost page rank
  - Solution: TrustRank