Analysis of Large Graphs: Link Analysis, PageRank

Mining of Massive Datasets
Jure Leskovec, Anand Rajaraman, Jeff Ullman
Stanford University
http://www.mmds.org

Web Search: 2 Challenges

2 challenges of web search:

- **(1) Web contains many sources of information Who to "trust"?**
  - Web is huge, full of untrusted documents, random things, web spam, etc.
  - **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

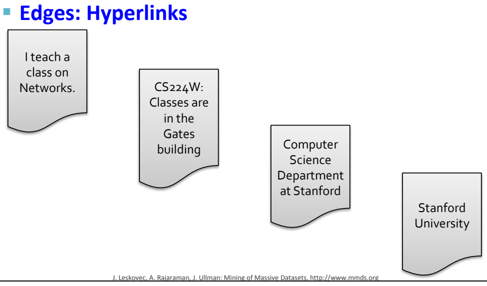
Google's PageRank (Brin/Page 98)

Google's PageRank (Brin/Page 98)

- A technique for estimating page quality
  - Based on web link graph
- Results are combined with IR score
  - Think of it as: TotalScore = IR score * PageRank
  - In practice, search engines use many other factors
  - (for example, Google says it uses more than 200 features)

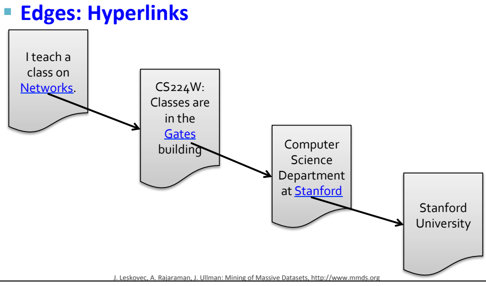
Web as a Graph

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

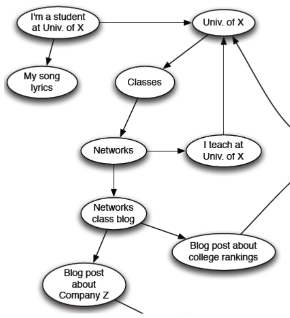


Web as a Graph

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

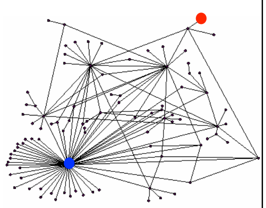


Web as a Directed Graph



Ranking Nodes on the Graph

- **All web pages are not equally "important"**
  www.joe-schmoe.com vs. www.stanford.edu
- There is large diversity in the web-graph node connectivity. **Let's rank the pages by the link structure!**

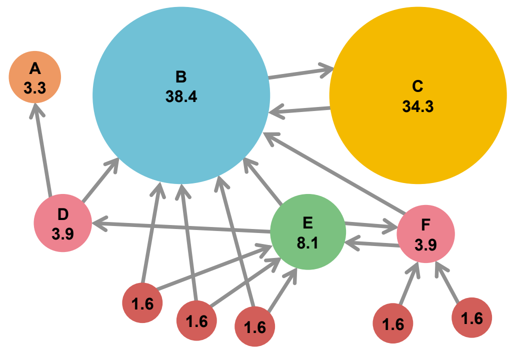


PageRank: The "Flow" Formulation

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!

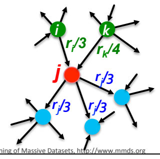
Example: PageRank Scores



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

$r_j = r_i/3 + r_k/4$

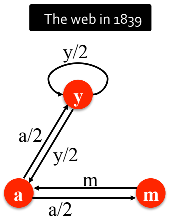


PageRank: The "Flow" Model

- 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

$$r_j = \sum_{i \to j} \frac{r_i}{d_i}$$

d_i… out-degree of node



"Flow" equations:

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

Solving the Flow Equations

- 3 equations, 3 unknowns, no constants
  - No unique solution
  - All solutions equivalent modulo the scale factor
- Additional constraint forces uniqueness:
  - $r_y + r_a + r_m = 1$
  - Solution: $r_y = 2/5$, $r_a = 2/5$, $r_m = 1/5$
- Gaussian elimination method works for small examples, but we need a better method for large web-size graphs
- We need a new formulation!

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

PageRank: Matrix Formulation

- Stochastic adjacency matrix M
  - Let page i has d_i out-links
  - If $i \to j$ then, $M_{ji} = 1/d_i$ else $M_{ji} = 0$
    - M is a column stochastic matrix
      - Columns sum to 1
- Rank vector r : vector with an entry per page
  - r_i is the importance score of page i
  - $\sum_i r_i = 1$
- The flow equations can be written

$$r = M . r$$

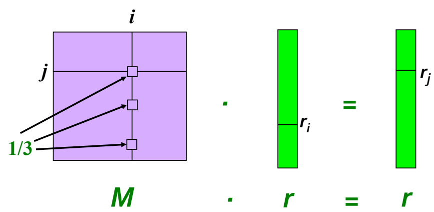
$$r_j = \sum_{i \to j} \frac{r_i}{d_i}$$

Example

- Remember the flow equation: $r_j = \sum_{i \to j} \frac{r_i}{d_i}$
- Flow equation in the matrix form

$M \cdot r = r$

  - Suppose page i links to 3 pages, including j



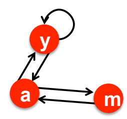
Eigenvector Formulation

- The flow equations can be written
  $r = M . r$
- So the **rank vector** r is an **eigenvector** of the stochastic web matrix M
  - In fact, its first or principal eigenvector, with corresponding eigenvalue **1**
    - Largest eigenvalue of M is 1 since M is column stochastic
      - ***Why?*** We know r is unit length and each column of M sums to one, so $Mr \le 1$
- We can now efficiently solve for r!

The method is called Power iteration

NOTE: x is an eigenvector with the corresponding eigenvalue λ if: $Ax = \lambda x$

Example: Flow Equations & M



	y	a	m
y	½	½	0
a	½	0	1
m	0	½	0

$r = M \cdot r$

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

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

Power Iteration Method

- **Given a web graph with n nodes, where the nodes are pages and edges are hyperlinks**
- **Power iteration:** a simple iterative scheme
  - Suppose there are N web pages
  - Initialize: $\mathbf{r}^{(0)} = [1/N,\ldots,1/N]^T$
  - Iterate: $\mathbf{r}^{(t+1)} = \mathbf{M} \cdot \mathbf{r}^{(t)}$
  - Stop when $|\mathbf{r}^{(t+1)} - \mathbf{r}^{(t)}|_1 < \varepsilon$

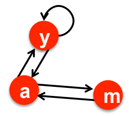
$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{d_i}$$

d_i …. out-degree of node i

$|\mathbf{x}|_1 = \sum_{1 \le i \le N} |x_i|$ is the $\mathbf{L_1}$ norm
Can use any other vector norm, e.g., Euclidean

PageRank: How to solve?

- **Power Iteration:**
  - Set $r_j = 1/N$
  - 1: $r'_j = \sum_{i \to j} \frac{r_i}{d_i}$
  - 2: $r = r'$
  - If not converged: goto 1
- **Example:**



	y	a	m
y	½	½	0
a	½	0	1
m	0	½	0

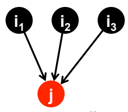
$\mathbf{r_y} = \mathbf{r_y}/2 + \mathbf{r_a}/2$
$\mathbf{r_a} = \mathbf{r_y}/2 + \mathbf{r_m}$
$\mathbf{r_m} = \mathbf{r_a}/2$

$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{matrix} 1/3 & 1/3 & 5/12 & 9/24 & & 6/15 \\ 1/3 & 3/6 & 1/3 & 11/24 & \cdots & 6/15 \\ 1/3 & 1/6 & 3/12 & 1/6 & & 3/15 \end{matrix}$$

Iteration 0, 1, 2,

Random Walk Interpretation

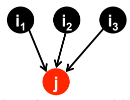
- **Imagine a random web surfer:**
  - At any time t, surfer is on some page i
  - At time $t+1$, the surfer follows an out-link from i uniformly at random
  - Ends up on some page j linked from i
  - Process repeats indefinitely
- **Let:**
  - $p(t)$ … vector whose i^{th} coordinate is the prob. that the surfer is at page i at time t
  - So, $p(t)$ is a probability distribution over pages



$$r_j = \sum_{i \to j} \frac{r_i}{d_{out}(i)}$$

The Stationary Distribution

- **Where is the surfer at time $t+1$?**
  - Follows a link uniformly at random
    $p(t+1) = M \cdot p(t)$
- Suppose the random walk reaches a state
  $p(t+1) = M \cdot p(t) = p(t)$
  then $p(t)$ is **stationary distribution** of a random walk
- **Our original rank vector** r satisfies $r = M \cdot r$
  - **So, r is a stationary distribution for the random walk**



$$p(t+1) = \mathrm{M} \cdot p(t)$$

Existence and Uniqueness

- **A central result from the theory of random walks (a.k.a. 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**

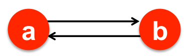
PageRank: The Google Formulation

PageRank: Three Questions

$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{\mathrm{d_i}} \quad \text{or equivalently} \quad r = Mr$$

- **Does this converge?**
- **Does it converge to what we want?**
- **Are results reasonable?**

Does this converge?



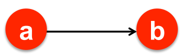
$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{\mathrm{d_i}}$$

- **Example:**

$$\begin{matrix} r_a \\ r_b \end{matrix} = \begin{matrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{matrix}$$

Iteration 0, 1, 2, …

Does it converge to what we want?



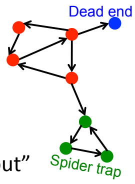
$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{\mathrm{d_i}}$$

- **Example:**

$$\begin{matrix} r_a \\ r_b \end{matrix} = \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{matrix}$$

Iteration 0, 1, 2, …

PageRank: Problems

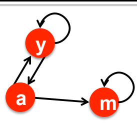


2 problems:

- **(1)** Some pages are **dead ends** (have no out-links)
  - Random walk has “nowhere” to go to
  - Such pages cause importance to “leak out”
- **(2) Spider traps:** (all out-links are within the group)
  - Random walked gets “stuck” in a trap
  - And eventually spider traps absorb all importance

Problem: Spider Traps

- **Power Iteration:**
  - Set $r_i = 1/N$
  - 1: $r'_j = \sum_{i \to j} \frac{r_i}{d_i}$
  - 2: $r = r'$
  - If not converged: goto 1



m is a spider trap

	y	a	m
y	½	½	0
a	½	0	0
m	0	½	1

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

- **Example:**

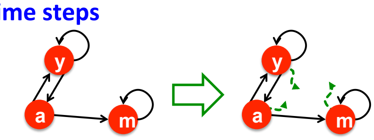
$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{matrix} 1/3 & 2/6 & 3/12 & 5/24 & & 0 \\ 1/3 & 1/6 & 2/12 & 3/24 & \cdots & 0 \\ 1/3 & 3/6 & 7/12 & 16/24 & & 1 \end{matrix}$$

Iteration 0, 1, 2,

All the PageRank score gets “trapped” in node m.

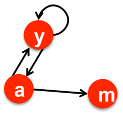
Solution: Random Teleports!

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



Problem: Dead Ends

- **Power Iteration:**
  - Set $r_i = 1/N$
  - 1: $r'_j = \sum_{i \to j} \frac{r_i}{d_i}$
  - 2: $r = r'$
  - If not converged: goto 1
- **Example:**



m is a spider trap

	y	a	m
y	½	½	0
a	½	0	0
m	0	½	0

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

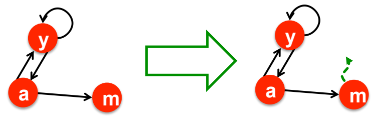
$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{matrix} 1/3 & 2/6 & 3/12 & 5/24 & & 0 \\ 1/3 & 1/6 & 2/12 & 3/24 & \cdots & 0 \\ 1/3 & 1/6 & 1/12 & 2/24 & & 0 \end{matrix}$$

Iteration 0, 1, 2,

Here the PageRank "leaks" out since the matrix is not stochastic.

Solution: Always Teleport!

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

Solution: Random Teleports

- **Google's solution that does it all:**
  - At each step, random surfer has two options:
    - With probability β, follow a link at random
    - With probability 1-β, jump to some random page
- PageRank equation [Brin-Page, 98]

$$r_j = \sum_{i \to j} \beta \frac{r_i}{d_i} + (1-\beta)\frac{1}{n}$$

.. out-degree of node i

This formulation assumes that M has no dead ends. We can either preprocess matrix M to remove all dead ends or explicitly follow random teleport links with probability 1.0 from dead-ends.

The Google Matrix

- **PageRank equation** [Brin-Page, 98]

$$r_j = \sum_{i \to j} \beta \frac{r_i}{d_i} + (1-\beta)\frac{1}{n}$$

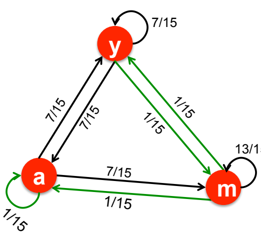
- **The Google Matrix A:**

$$A = \beta\, M + (1-\beta)\frac{1}{n}\boldsymbol{e} \cdot \boldsymbol{e}^T$$

e vector of all 1s

- We have a recursive problem: $r = A \cdot r$
  - **Power iteration still works!**
- **What is β?**
  - In practice β =0.8,0.9 (make 5 steps and jump)

Random Teleports ($\beta = 0.8$)



$$0.8 \overbrace{\begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 1 \end{bmatrix}}^{\textbf{M}} + 0.2 \overbrace{\begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix}}^{[1/N]_{NxN}}$$

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

$$\begin{matrix} y \\ a \\ m \end{matrix} = \begin{matrix} 1/3 & 0.33 & 0.24 & 0.26 & & 7/33 \\ 1/3 & 0.20 & 0.20 & 0.18 & \ldots & 5/33 \\ 1/3 & 0.46 & 0.52 & 0.56 & & 21/33 \end{matrix}$$

How do we actually compute the PageRank?

Computing Page Rank

- **Key step is matrix-vector multiplication**
  - $r^{new} = A \cdot r^{old}$
- Easy if we have enough main memory to hold **A**, r^{old}, 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!

$\mathbf{A} = \beta \cdot \mathbf{M} + (1-\beta)\,[1/N]_{N\times N}$

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

$$= \begin{bmatrix} 7/15 & 7/15 & 1/15 \\ 7/15 & 1/15 & 1/15 \\ 1/15 & 7/15 & 13/15 \end{bmatrix}$$

Matrix Formulation

- Suppose there are ***N*** pages
- Consider page ***i***, with d_i out-links
- We have $M_{ji} = 1/|d_i|$ when $i \to j$ and $M_{ji} = 0$ otherwise
- **The random teleport is equivalent to:**
  - Adding a **teleport link** from ***i*** to every other page and setting transition probability to ***(1-β)/N***
  - Reducing the probability of following each out-link from $1/|d_i|$ to $\beta/|d_i|$
  - **Equivalent:** Tax each page a fraction ***(1-β)*** of its score and redistribute evenly

Rearranging the Equation

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

  $= \sum_{j=1}^{N} \beta\, M_{ij} \cdot r_j + \sum_{j=1}^{N} \frac{1-\beta}{N} r_j$

  $= \sum_{j=1}^{N} \beta\, M_{ij} \cdot r_j + \frac{1-\beta}{N}$ since $\sum r_j = 1$
- **So we get:** $r = \beta\, M \cdot r + \left[\frac{1-\beta}{N}\right]_N$

Note: Here we assumed **M** has no dead-ends.

$[x]_N$ … a vector of length N with all entries x

Sparse Matrix Formulation

- We just rearranged the **PageRank equation**
  - where $[(1-\beta)/N]_N$ is a vector with all ***N*** entries **(1-β)/N**
- ***M*** is a **sparse matrix!** (with no dead-ends)
  - 10 links per node, approx 10N entries
- So in each iteration, we need to:
  - Compute $r^{new} = \beta\, M \cdot r^{old}$
  - Add a constant value **(1-β)/N** to each entry in r^{new}
    - **Note if M contains dead-ends then and we also have to renormalize r^{new} so that it sums to 1**

PageRank: The Complete Algorithm

- **Input: Graph G and parameter β**
  - Directed graph G with **spider traps** and **dead ends**
  - Parameter β
- **Output: PageRank vector r**
  - **Set:** $r_j^{(0)} = \frac{1}{N}$, $t = 1$
  - **do:**
    - $\forall j$: $r'^{(t)}_j = \sum_{i \to j} \beta \frac{r_i^{(t-1)}}{d_i}$

      $r'^{(t)}_j = 0$ if in-deg. of j is **0**
    - **Now re-insert the leaked PageRank:**

      $\forall j$: $r_j^{(t)} = r'^{(t)}_j + \frac{1-S}{N}$ **where:** $S = \sum_j r'^{(t)}_j$
    - $t = t + 1$
  - **while** $\sum_j \left| r_j^{(t)} - r_j^{(t-1)} \right| > \varepsilon$