Link Analysis: PageRank
Web pages are not equally “important”

www.joe-schmoe.com vs. www.stanford.edu

We already know:
Since there is large diversity in the connectivity of the webgraph we can rank the pages by the link structure
link analysis algorithms

- We will cover the following Link Analysis approaches to computing importances of nodes in a graph:
  - Page Rank
  - Hubs and Authorities (HITS)
  - Topic-Specific (Personalized) Page Rank
  - Web Spam Detection Algorithms
**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](http://www.stanford.edu) has 23,400 inlinks
    - [www.joe-schmoe.com](http://www.joe-schmoe.com) has 1 inlink
  - **Are all in-links are 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 $p$ with importance $x$ has $n$ out-links, each link gets $x/n$ votes.
- Page $p$’s own importance is the sum of the votes on its in-links.
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 node $j$.

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

Flow equations:

$$r_y = \frac{r_y}{2} + \frac{r_a}{2}$$

$$r_a = \frac{r_y}{2} + r_m$$

$$r_m = \frac{r_a}{2}$$
Solving the Flow Equations

- **3 equations, 3 unknowns, no constants**
  - No unique solution
  - All solutions equivalent modulo scale factor
- **Additional constraint forces uniqueness**
  - \( y + a + m = 1 \)
  - \( y = 2/5, \ a = 2/5, \ m = 1/5 \)
- Gaussian elimination method works for small examples, but we need a better method for large web-size graphs

Flow equations:
\[
\begin{align*}
\dot{r}_y &= \frac{r_y}{2} + \frac{r_a}{2} \\
\dot{r}_a &= \frac{r_y}{2} + \frac{r_m}{2} \\
\dot{r}_m &= \frac{r_a}{2}
\end{align*}
\]
**PageRank: Matrix Formulation**

- **Stochastic adjacency matrix** $M$
  - Let page $j$ has $d_j$ out-links
  - If $j \rightarrow i$, then $M_{ij} = 1/d_j$ else $M_{ij} = 0$
    - $M$ 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
  $$ r = M r $$
Example

- Suppose page $j$ links to 3 pages, including $i$

\[
M \quad r \quad i
\]

\[
\frac{1}{3}
\]
The flow equations can be written

\[ r = M \cdot r \]

So the rank vector is an eigenvector of the stochastic web matrix

- In fact, its first or principal eigenvector, with corresponding eigenvalue 1
Example

\[ r_y = \frac{r_y}{2} + \frac{r_a}{2} \]
\[ r_a = \frac{r_y}{2} + r_m \]
\[ r_m = \frac{r_a}{2} \]

\[
\begin{array}{ccc}
  y & a & m \\
  \frac{1}{2} & \frac{1}{2} & 0 \\
  a & 0 & 1 \\
  m & \frac{1}{2} & 0 \\
\end{array}
\]

\[ r = Mr \]
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: \( r^{(0)} = [1/N, \ldots, 1/N]^T \)
  
  - Iterate: \( r^{(t+1)} = M \cdot r^{(t)} \)
  
  - Stop when \( |r^{(t+1)} - r^{(t)}|_1 < \varepsilon \)
    
    - \( |x|_1 = \sum_{1 \leq i \leq N} |x_i| \) is the L1 norm
    
    - Can use any other vector norm e.g., Euclidean

\[
    r^{(t+1)}_j = \sum_{i \rightarrow j} \frac{r^{(t)}_i}{d_i}
\]

\( d_i \) .... out-degree of node \( i \)
**PageRank: How to solve?**

- **Power Iteration:**
  - Set $r_j = 1/N$
  - $r_j = \sum_{i \rightarrow j} \frac{r_i}{d_i}$
    - And iterate
  - $r_i = \sum_j M_{ij} \cdot r_j$

- **Example:**

  \[
  \begin{bmatrix}
  r_y \\
  r_a \\
  r_m
  \end{bmatrix} =
  \begin{bmatrix}
  1/3 & 1/3 & 5/12 & 9/24 & 6/15 \\
  1/3 & 3/6 & 1/3 & 11/24 & \ldots & 6/15 \\
  1/3 & 1/6 & 3/12 & 1/6 & 3/15 \\
  \end{bmatrix}
  \]

  Iteration 0, 1, 2, …

<table>
<thead>
<tr>
<th></th>
<th>y</th>
<th>a</th>
<th>m</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>$\frac{1}{2}$</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>m</td>
<td>0</td>
<td>$\frac{1}{2}$</td>
<td>0</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
  r_y &= r_y/2 + r_a/2 \\
  r_a &= r_y/2 + r_m \\
  r_m &= r_a/2
\end{align*}
\]
Imagine a random web surfer:
- At any time $t$, surfer is on some page $u$
- At time $t+1$, the surfer follows an out-link from $u$ uniformly at random
- Ends up on some page $v$ linked from $u$
- Process repeats indefinitely

Let:
- $p(t)$ ... vector whose $i^{th}$ coordinate is the prob. that the surfer is at page $i$ at time $t$
- $p(t)$ is a probability distribution over pages
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 rank vector $r$ satisfies $r = M \cdot r$
  - So, it is a stationary distribution for the random walk
PageRank: Three Questions

\[ r_{j}^{(t+1)} = \sum_{i \rightarrow j} \frac{r_{i}^{(t)}}{d_{i}} \]

or equivalently

\[ r = Mr \]

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

Example:

\[ r_a = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix} \]

\[ r_b = \begin{bmatrix} \end{bmatrix} \]

Iteration 0, 1, 2, …
Example:

\[ r_a = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix} \]

\[ r_b = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \end{pmatrix} \]

Iteration 0, 1, 2, …

\[ r_{j}^{(t+1)} = \sum_{i \rightarrow j} \frac{r_{i}^{(t)}}{d_{i}} \]
Problems with the “Flow” Model

2 problems:

- Some pages are “dead ends” (have no out-links)
  - Such pages cause importance to “leak out”

- **Spider traps** (all out links are within the group)
  - Eventually spider traps absorb all importance
Problem: Spider Traps

- **Power Iteration:**
  - Set $r_j = 1$
  - $r_j = \sum_{i \rightarrow j} \frac{r_i}{d_i}$
  - And iterate

- **Example:**

<table>
<thead>
<tr>
<th></th>
<th>y</th>
<th>a</th>
<th>m</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>$\frac{1}{2}$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>m</td>
<td>0</td>
<td>$\frac{1}{2}$</td>
<td>1</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
\begin{pmatrix} r_y \\ r_a \\ r_m \end{pmatrix} &= \begin{pmatrix} 1/3 \\ 1/3 \\ 1/3 \end{pmatrix} + \begin{pmatrix} 2/6 \\ 1/6 \\ 3/6 \end{pmatrix} + \begin{pmatrix} 3/12 \\ 2/12 \\ 7/12 \end{pmatrix} + \begin{pmatrix} 5/24 \\ 3/24 \\ 16/24 \end{pmatrix} \\
&= \begin{pmatrix} 1/3 \\ 1/3 \\ 1/3 \end{pmatrix} + \begin{pmatrix} 2/6 \\ 1/6 \\ 3/6 \end{pmatrix} + \begin{pmatrix} 3/12 \\ 2/12 \\ 7/12 \end{pmatrix} + \begin{pmatrix} 5/24 \\ 3/24 \\ 16/24 \end{pmatrix} \\
&= \begin{pmatrix} 1/3 + 2/6 + 3/12 + 5/24 \\ 1/3 + 1/6 + 2/12 + 3/24 \\ 1/3 + 3/6 + 7/12 + 16/24 \end{pmatrix} \\
&= \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}
\end{align*}

Iteration 0, 1, 2, …
The Google solution for spider traps: At each time step, the random surfer has two options:

- With probability $\beta$, follow a link at random
- With probability $1-\beta$, jump to some page uniformly at random
- Common values for $\beta$ 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 $r_j = 1$
  - $r_j = \sum_{i \rightarrow j} \frac{r_i}{d_i}$
  - And iterate

- **Example:**

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

  $r_y = \frac{r_y}{2} + \frac{r_a}{2}$
  $r_a = \frac{r_y}{2}$
  $r_m = \frac{r_a}{2}$

  Iteration 0, 1, 2, …
Solution: Dead Ends

- **Teleports:** Follow random teleport links with probability 1.0 from dead-ends
  - Adjust matrix accordingly

<table>
<thead>
<tr>
<th></th>
<th>y</th>
<th>a</th>
<th>m</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>½</td>
<td>½</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>½</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>m</td>
<td>0</td>
<td>½</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>y</th>
<th>a</th>
<th>m</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>½</td>
<td>½</td>
<td>¹⁄₃</td>
</tr>
<tr>
<td>a</td>
<td>½</td>
<td>0</td>
<td>¹⁄₃</td>
</tr>
<tr>
<td>m</td>
<td>0</td>
<td>½</td>
<td>¹⁄₃</td>
</tr>
</tbody>
</table>
Markov Chains

- Set of states $X$
- Transition matrix $P$ where $P_{ij} = P(X_t=i \mid X_{t-1}=j)$
- $\pi$ specifying the probability of being at each state $x \in X$
- Goal is to find $\pi$ such that $\pi = P \pi$

$$r^{(t+1)} = Mr^{(t)}$$
Theory of Markov chains

Fact: For any start vector, the power method applied to a Markov transition matrix $P$ will converge to a unique positive stationary vector as long as $P$ is stochastic, irreducible and aperiodic.
Make M Stochastic

- **Stochastic**: Every column sums to 1
- **A possible solution**: Add green links

\[ S = M + a^T \left( \frac{1}{n} \right) \]

\[
\begin{array}{ccc}
  y & a & m \\
  \frac{1}{2} & \frac{1}{2} & \frac{1}{3} \\
  \frac{1}{2} & 0 & \frac{1}{3} \\
  0 & \frac{1}{2} & \frac{1}{3} \\
\end{array}
\]

\[
\begin{align*}
  r_y &= r_y / 2 + r_a / 2 + r_m / 3 \\
  r_a &= r_y / 2 + r_m / 3 \\
  r_m &= r_a / 2 + r_m / 3
\end{align*}
\]

- \( a_i \ldots = 1 \) if node \( i \) has out deg 0, =0 else
- 1…vector of all 1s
A chain is **periodic** if there exists $k > 1$ such that the interval between two visits to some state $s$ is always a multiple of $k$.

**A possible solution:** Add green links
From any state, there is a non-zero probability of going from any one state to any another

A possible solution: Add green links
Google’s solution that does it all:
- Makes $M$ stochastic, aperiodic, irreducible
- At each step, random surfer has two options:
  - With probability $1 - \beta$, follow a link at random
  - With probability $\beta$, jump to some random page
- PageRank equation [Brin-Page, 98]

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

Assuming we follow random teleport links with probability 1.0 from dead-ends
The Google Matrix

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

\[ r_j = \sum_{i \rightarrow j} (1 - \beta) \frac{r_i}{d_i} + \beta \frac{1}{n} \]

- The Google Matrix \( A \):

\[ A = (1 - \beta)S + \beta \frac{1}{n} \mathbf{1} \cdot \mathbf{1}^T \]

- \( G \) is stochastic, aperiodic and irreducible, so

\[ r^{(t+1)} = A \cdot r^{(t)} \]

- **What is \( \beta \)?**
  - In practice \( \beta = 0.15 \) (make 5 steps and jump)
Random Teleports ($\beta = 0.8$)

\[
S = \begin{bmatrix}
0.5 & 0.5 & 0 \\
0.5 & 0.5 & 1 \\
0 & 0.5 & 1 \\
\end{bmatrix}
\]

\[
A = \begin{bmatrix}
\frac{7}{15} & \frac{7}{15} & \frac{1}{15} \\
\frac{7}{15} & \frac{1}{15} & \frac{1}{15} \\
\frac{1}{15} & \frac{7}{15} & \frac{13}{15} \\
\end{bmatrix}
\]

\[
y = \begin{bmatrix}
\frac{1}{3} \\
\frac{1}{3} \\
\frac{1}{3} \\
\end{bmatrix}
\]

\[
a = \begin{bmatrix}
0.33 \\
0.20 \\
0.46 \\
\end{bmatrix}
\]

\[
m = \begin{bmatrix}
0.24 \\
0.20 \\
0.52 \\
\end{bmatrix}
\]

\[
\begin{align*}
y & = \frac{7}{33} \\
a & = \frac{5}{33} \\
m & = \frac{21}{33}
\end{align*}
\]
Matrix Formulation

- Suppose there are N pages
- Consider a page $j$, with set of out-links $O(j)$
- We have $M_{ij} = 1/|O(j)|$ when $j \rightarrow i$ 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 out-link from $1/|O(j)|$ to $\beta/|O(j)|$
  - Equivalent: Tax each page a fraction $(1-\beta)$ of its score and redistribute evenly
Construct the $N \times N$ matrix $A$ as follows
- $A_{ij} = \beta \cdot M_{ij} + (1-\beta)/N$
- Verify that $A$ is a stochastic matrix
- The page rank vector $r$ is the principal eigenvector of this matrix $A$
  - satisfying $r = A \cdot r$
- Equivalently, $r$ is the stationary distribution of the random walk with teleports
Key step is matrix-vector multiplication
- \( \mathbf{r}^{\text{new}} = \mathbf{A} \cdot \mathbf{r}^{\text{old}} \)
- Easy if we have enough main memory to hold \( \mathbf{A}, \mathbf{r}^{\text{old}}, \mathbf{r}^{\text{new}} \)
- Say \( N = 1 \) billion pages
  - We need 4 bytes for each entry (say)
  - 2 billion entries for vectors, approx 8GB
  - Matrix \( \mathbf{A} \) has \( N^2 \) entries
    - \( 10^{18} \) is a large number!

\[
\mathbf{A} = \beta \cdot \mathbf{M} + (1-\beta) \left[ \frac{1}{N} \right]_{N \times N}
\]

\[
\begin{pmatrix}
\frac{1}{2} & \frac{1}{2} & 0 \\
\frac{1}{2} & 0 & 0 \\
0 & \frac{1}{2} & 1 \\
\end{pmatrix} \quad + 0.2 \\
= 
\begin{pmatrix}
7/15 & 7/15 & 1/15 \\
7/15 & 1/15 & 1/15 \\
1/15 & 7/15 & 13/15 \\
\end{pmatrix}
\]
Rearranging the Equation

- \( \mathbf{r} = \mathbf{A} \cdot \mathbf{r} \), where \( A_{ij} = \beta M_{ij} + \frac{1-\beta}{N} \)

- \( r_i = \sum_{j=1}^{N} A_{ij} \cdot r_j \)

- \( r_i = \sum_{j=1}^{N} \left[ \beta M_{ij} + \frac{1-\beta}{N} \right] \cdot r_j \)

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

- \[ = \sum_{j=1}^{N} \beta M_{ij} \cdot r_j + \frac{1-\beta}{N}, \text{ since } \sum r_j = 1 \]

- So, \( \mathbf{r} = \beta \mathbf{M} \cdot \mathbf{r} + \left[ \frac{1-\beta}{N} \right]_{N} \)

\([x]_{N} \ldots \text{ a vector of length } N \text{ with all entries } x\)
Sparse Matrix Formulation

- We can rearrange the PageRank equation
  \[ r = \beta M \cdot r + \left[ \frac{1-\beta}{N} \right]_N \]
  - \([1-\beta]/N_N\) 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 \(r^{new} = \beta M \cdot r^{old}\)
  - Add a constant value \((1-\beta)/N\) to each entry in \(r^{new}\)
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

<table>
<thead>
<tr>
<th>source node</th>
<th>degree</th>
<th>destination nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>1, 5, 7</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>17, 64, 113, 117, 245</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>13, 23</td>
</tr>
</tbody>
</table>
Basic Algorithm: Update Step

- Assume enough RAM to fit \( r^{\text{new}} \) into memory
  - Store \( r^{\text{old}} \) and matrix \( M \) on disk
- Then 1 step of power-iteration is:
  Initialize all entries of \( r^{\text{new}} \) to \( (1-\beta)/N \)
  For each page \( p \) (of out-degree \( n \)):
    Read into memory: \( p, n, \text{dest}_1, \ldots, \text{dest}_n, r^{\text{old}}(p) \)
    for \( j = 1 \ldots n \): \( r^{\text{new}}(\text{dest}_j) += \beta \ r^{\text{old}}(p) / n \)
Assume enough RAM to fit $r^{new}$ into memory
- Store $r^{old}$ and matrix $M$ on disk
- In each iteration, we have to:
  - Read $r^{old}$ and $M$
  - Write $r^{new}$ back to disk
  - IO cost = $2|r| + |M|$

**Question:**
- What if we could not even fit $r^{new}$ in memory?
### Block-based Update Algorithm

#### Table: src, degree, destination

<table>
<thead>
<tr>
<th>src</th>
<th>degree</th>
<th>destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>0, 1, 3, 5</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0, 5</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3, 4</td>
</tr>
</tbody>
</table>

#### Diagram: r<sub>new</sub> and r<sub>old</sub>
Analysis of Block Update

- Similar to nested-loop join in databases
  - Break $r^\text{new}$ into $k$ blocks that fit in memory
  - Scan $M$ and $r^\text{old}$ once for each block
- $k$ scans of $M$ and $r^\text{old}$
  - $k(|M| + |r|) + |r| = k|M| + (k+1)|r|$

- Can we do better?
  - Hint: $M$ is much bigger than $r$ (approx 10-20x), so we must avoid reading it $k$ times per iteration
## Block-Stripe Update Algorithm

<table>
<thead>
<tr>
<th>src</th>
<th>degree</th>
<th>destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>0, 1</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

```plaintext
r_{\text{new}}
```

<table>
<thead>
<tr>
<th>src</th>
<th>degree</th>
<th>destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

```plaintext
r_{\text{old}}
```

<table>
<thead>
<tr>
<th>src</th>
<th>degree</th>
<th>destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>
Block-Stripe Analysis

- Break $M$ into stripes
  - Each stripe contains only destination nodes in the corresponding block of $r^{\text{new}}$
- Some additional overhead per stripe
  - But it is usually worth it
- Cost per iteration
  - $|M|(1+\varepsilon) + (k+1)|r|$
Some Problems with Page Rank

- **Measures generic popularity of a page**
  - Biased against topic-specific authorities
  - **Solution:** Topic-Specific PageRank (next)

- **Uses a single measure of importance**
  - Other models e.g., hubs-and-authorities
  - **Solution:** Hubs-and-Authorities (next)

- **Susceptible to Link spam**
  - Artificial link topographies created in order to boost page rank
  - **Solution:** TrustRank (next)