Analysis of Large Graphs: Link Analysis, PageRank
New Topic: Graph Data!

High dim. data
- Locality sensitive hashing
- Clustering
- Dimensionality reduction

Graph data
- PageRank, SimRank
- Community Detection
- Spam Detection

Infinite data
- Filtering data streams
- Web advertising
- Queries on streams

Machine learning
- SVM
- Decision Trees
- Perceptron, kNN

Apps
- Recommender systems
- Association Rules
- Duplicate document detection
Graph Data: Social Networks

Facebook social graph
4-degrees of separation [Backstrom-Boldi-Rosa-Ugander-Vigna, 2011]
Connections between political blogs
Polarization of the network [Adamic-Glance, 2005]
Citation networks and Maps of science
[Börner et al., 2012]
Graph Data: Communication Nets

Internet
Graph Data: Technological Networks

Seven Bridges of Königsberg

[Euler, 1735]

Return to the starting point by traveling each link of the graph once and only once.
Web as a directed graph:

- **Nodes:** Webpages
- **Edges:** Hyperlinks

I teach a class on Networks.

CS242W: Classes are in the Gates building

Computer Science Department at Stanford

Stanford University
Web as a Graph

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

I teach a class on Networks.

CS224W: Classes are in the Gates building

Computer Science Department at Stanford

Stanford University
Web as a Directed Graph

I'm a student at Univ. of X

My song lyrics

Classes

I teach at Univ. of X

Networks

Networks class blog

Blog post about Company Z

Blog post about college rankings

Univ. of X

I'm applying to college

USNews College Rankings

USNews Featured Colleges
Broad Question

- How to organize the Web?

  - First try: Human curated Web directories
    - Yahoo, DMOZ, LookSmart
  
- 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.
2 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”
  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!
We will cover the following link analysis approaches for computing importances of nodes in a graph:

- Page Rank
- Hubs and Authorities (HITS)
- Topic-Specific (Personalized) Page Rank
- Web Spam Detection Algorithms
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 are equal?

- Links from important pages count more
- Recursive question!
Example: PageRank Scores

The diagram illustrates the PageRank scores for different nodes in a network. Each node is represented by a circle, and the score is indicated within the circle. Arrows indicate the direction of links between nodes.

- **Node B** with a score of 38.4 is the highest-scoring node.
- **Node C** with a score of 34.3.
- **Node D** with a score of 3.9.
- **Node E** with a score of 8.1.
- **Node F** with a score of 3.9.

The scores are distributed such that nodes with more incoming links from other high-score nodes tend to have higher scores. This reflects the idea of PageRank, which is a key algorithm used by Google to rank web pages in their search engine results.
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.

$$r_j = \frac{r_i}{3} + \frac{r_k}{4}$$
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 $j$

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

$d_i$ ... out-degree of node $i$

“Flow” equations:

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

The web in 1839
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: $r_y = \frac{2}{5}$, $r_a = \frac{2}{5}$, $r_m = \frac{1}{5}$
- Gaussian elimination method works for small examples, but we need a better method for large web-size graphs
- We need a new formulation!

Flow equations:

- $r_y = \frac{r_y}{2} + \frac{r_a}{2}$
- $r_a = \frac{r_y}{2} + r_m$
- $r_m = \frac{r_a}{2}$
PageRank: Matrix Formulation

- **Stochastic adjacency matrix** $M$
  - Let page $i$ has $d_i$ out-links
  - If $i \rightarrow j$, then $M_{ji} = \frac{1}{d_i}$ 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
  
  $$r = M \cdot r$$

2/5/2013
Jure Leskovec, Stanford C246: Mining Massive Datasets
Example

- Remember the flow equation: $r_j = \sum_{i \rightarrow j} \frac{r_i}{d_i}$
- Flow equation in the matrix form:
  \[ M \cdot r = r \]
- Suppose page $i$ links to 3 pages, including $j$
The flow equations can be written
\[ \mathbf{r} = \mathbf{M} \cdot \mathbf{r} \]
So the rank vector \( \mathbf{r} \) is an eigenvector of the stochastic web matrix \( \mathbf{M} \)

- In fact, its first or principal eigenvector, with corresponding eigenvalue \( 1 \)
  - Largest eigenvalue of \( \mathbf{M} \) is \( 1 \) since \( \mathbf{M} \) is column stochastic
    - We know \( \mathbf{r} \) is unit length and each column of \( \mathbf{M} \) sums to one, so \( \mathbf{M}\mathbf{r} \leq 1 \)

We can now efficiently solve for \( \mathbf{r}! \)
The method is called Power iteration

NOTE: \( \mathbf{x} \) is an eigenvector with the corresponding eigenvalue \( \lambda \) if:
\[ A\mathbf{x} = \lambda \mathbf{x} \]
Example: Flow Equations & M

\[ r = M \cdot r \]

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

\[
\begin{array}{ccc}
    y & a & m \\
    \hline
    y & 1/2 & 1/2 \quad 0 \\
    a & 1/2 \quad 0 \quad 1 \\
    m & 0 \quad 1/2 \quad 0 \\
\end{array}
\]
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 \( L_1 \) norm

\[
\begin{align*}
    r_{j}^{(t+1)} &= \sum_{i \rightarrow j} \frac{r_{i}^{(t)}}{d_i} \\
    d_i &\text{ …. out-degree of node } i
\end{align*}
\]
PageRank: How to solve?

- **Power Iteration:**
  - Set \( r_j = 1/N \)
  - 1: \( r'_j = \sum_{i \rightarrow j} \frac{r_i}{d_i} \)
  - 2: \( r = r' \)
  - Goto 1

- **Example:**

\[
\begin{pmatrix}
  r_y \\
  r_a \\
  r_m
\end{pmatrix} = \begin{pmatrix}
  1/3 \\
  1/3 \\
  1/3
\end{pmatrix}
\]

Iteration 0, 1, 2, …
PageRank: How to solve?

- **Power Iteration:**
  - Set $r_j = 1/N$
  - 1: $r'_j = \sum_i \frac{r_i}{d_i}$
  - 2: $r = r'$
  - Goto 1

- **Example:**

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

Iteration 0, 1, 2, …
Why Power Iteration works? (1)

- **Power iteration:**
  A method for finding dominant eigenvector (the vector corresponding to the largest eigenvalue)
  - \( r^{(1)} = M \cdot r^{(0)} \)
  - \( r^{(2)} = M \cdot r^{(1)} = M(Mr^{(1)}) = M^2 \cdot r^{(0)} \)
  - \( r^{(3)} = M \cdot r^{(2)} = M(M^2r^{(0)}) = M^3 \cdot r^{(0)} \)
- **Claim:**
  Sequence \( M \cdot r^{(0)}, M^2 \cdot r^{(0)}, ... M^k \cdot r^{(0)}, ... \) approaches the dominant eigenvector of \( M \)
Why Power Iteration works? (2)

- **Claim:** Sequence $\mathbf{M} \cdot \mathbf{r}^{(0)}, \mathbf{M}^2 \cdot \mathbf{r}^{(0)}, \ldots \mathbf{M}^k \cdot \mathbf{r}^{(0)}, \ldots$ approaches the dominant eigenvector of $\mathbf{M}$

- **Proof:**
  - Assume $\mathbf{M}$ has $n$ linearly independent eigenvectors, $x_1, x_2, \ldots, x_n$ with corresponding eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$, where $\lambda_1 > \lambda_2 > \cdots > \lambda_n$
  - Vectors $x_1, x_2, \ldots, x_n$ form a basis and thus we can write: $\mathbf{r}^{(0)} = c_1 x_1 + c_2 x_2 + \cdots + c_n x_n$
  - $\mathbf{M} \mathbf{r}^{(0)} = \mathbf{M}(c_1 x_1 + c_2 x_2 + \cdots + c_n x_n)$
    $$= c_1 (\mathbf{M} x_1) + c_2 (\mathbf{M} x_2) + \cdots + c_n (\mathbf{M} x_n)$$
    $$= c_1 (\lambda_1 x_1) + c_2 (\lambda_2 x_2) + \cdots + c_n (\lambda_n x_n)$$
  - Repeated multiplication on both sides produces
    $$\mathbf{M}^k \mathbf{r}^{(0)} = c_1 (\lambda_1^k x_1) + c_2 (\lambda_2^k x_2) + \cdots + c_n (\lambda_n^k x_n)$$
Claim: Sequence $M \cdot r^{(0)}, M^2 \cdot r^{(0)}, \ldots M^k \cdot r^{(0)}, \ldots$ approaches the dominant eigenvector of $M$.

Proof (continued):

- Repeated multiplication on both sides produces
  
  \[ M^k r^{(0)} = c_1 (\lambda_1^k x_1) + c_2 (\lambda_2^k x_2) + \cdots + c_n (\lambda_n^k x_n) \]

- \[ M^k r^{(0)} = \lambda_1^k \left[ c_1 x_1 + c_2 \left( \frac{\lambda_2}{\lambda_1} \right)^k x_2 + \cdots + c_n \left( \frac{\lambda_2}{\lambda_1} \right)^k x_n \right] \]

- Since $\lambda_1 > \lambda_2$ then fractions $\frac{\lambda_2}{\lambda_1}, \frac{\lambda_3}{\lambda_1} \ldots < 1$

  and so $\left( \frac{\lambda_i}{\lambda_1} \right)^k = 0$ as $k \to \infty$ (for all $i = 2 \ldots n$).

Thus: $M^k r^{(0)} \approx c_1 (\lambda_1^k x_1)$

- Note if $c_1 = 0$ then the method won’t converge.
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

$$r_j = \sum_{i \rightarrow j} \frac{r_i}{d_{out}(i)}$$
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.
PageRank: 3 Questions

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

$$
\begin{align*}
\mathbf{r}_{(t+1)} & = \sum_{i \rightarrow j} \frac{\mathbf{r}_i^{(t)}}{d_i} \\
\text{or equivalently} & \\
\mathbf{r} & = \mathbf{M}\mathbf{r}
\end{align*}
$$

Announcement: We graded HW0 and HW1!
- Stanford students: Pick them up from the submission box in Gates
- SCPD students: SCPD will send you the HW
Does this converge?

- **Example:**

\[
\begin{align*}
\mathbf{r}_a &= \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{bmatrix} \\
\mathbf{r}_b &= \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{bmatrix}
\end{align*}
\]

Iteration 0, 1, 2, …

\[
r_j^{(t+1)} = \sum_{i \rightarrow j} \frac{r_i^{(t)}}{d_i}
\]
Does it converge to what we want?

- Example:

\[
\begin{align*}
\mathbf{r}_a &= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix} \\
\mathbf{r}_b &= \begin{pmatrix} \end{pmatrix}
\end{align*}
\]

Iteration 0, 1, 2, …

\[
r_j^{(t+1)} = \sum_{i \rightarrow j} \frac{r_i^{(t)}}{d_i}
\]
2 problems:

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

2. **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>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>1</td>
</tr>
</tbody>
</table>

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

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

Iteration 0, 1, 2, …

2/5/2013
Jure Leskovec, Stanford C246: Mining Massive Datasets
The Google solution for spider traps: At each time step, the random surfer has two options

- With prob. $\beta$, follow a link at random
- With prob. $1-\beta$, jump to some random page
- 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 $r_j = 1$
  - $r_j = \sum \frac{r_i}{d_i}$
  - And iterate

- **Example:**

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

  Iteration 0, 1, 2, \ldots

<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>

\[
\begin{align*}
  r_y &= r_y/2 + r_a/2 \\
  r_a &= r_y/2 \\
  r_m &= r_a/2
\end{align*}
\]
Solution: Always Teleport

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

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

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

- Set of states $X$
- Transition matrix $P$ where $P_{ij} = P(X_t=i \mid X_{t-1}=j)$
- $\pi$ specifying the stationary probability of being at each state $x \in X$
- Goal is to find $\pi$ such that $\pi = P \pi$
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.
- **Stochastic:** Every column sums to 1
- **A possible solution:** Add green links

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

<table>
<thead>
<tr>
<th></th>
<th>y</th>
<th>a</th>
<th>m</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>1/2</td>
<td>1/2</td>
<td>1/3</td>
</tr>
<tr>
<td>a</td>
<td>1/2</td>
<td>0</td>
<td>1/3</td>
</tr>
<tr>
<td>m</td>
<td>0</td>
<td>1/2</td>
<td>1/3</td>
</tr>
</tbody>
</table>

- \( a_i = 1 \) if node \( i \) has out deg 0, =0 else
- \( e \)...vector of all 1s

\[
\begin{align*}
\text{r}_y &= \frac{r_y}{2} + \frac{r_a}{2} + \frac{r_m}{3} \\
\text{r}_a &= \frac{r_y}{2} + \frac{r_m}{3} \\
\text{r}_m &= \frac{r_a}{2} + \frac{r_m}{3}
\end{align*}
\]
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
Make $M$ Irreducible

- 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 $\beta$, follow a link at random
  - With probability $1 - \beta$, jump to some random page
- PageRank equation [Brin-Page, 98]

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

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]

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

- The Google Matrix \( A \):

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

\( A \) is stochastic, aperiodic and irreducible, so

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

- **What is \( \beta \)?**

  - In practice \( \beta = 0.8, 0.9 \) (make 5 steps and jump)
Random Teleports ($\beta = 0.8$)

\[
\begin{align*}
\mathbf{y} & = 1/3 & 0.33 & 0.24 & 0.26 & 7/33 \\
\mathbf{a} & = 1/3 & 0.20 & 0.20 & 0.18 & \ldots & 5/33 \\
\mathbf{m} & = 1/3 & 0.46 & 0.52 & 0.56 & 21/33
\end{align*}
\]
How do we actually compute the PageRank?
Computing Page Rank

- **Key step is matrix-vector multiplication**
  - \( r_{\text{new}} = A \cdot r_{\text{old}} \)
- Easy if we have enough main memory to hold \( A, r_{\text{old}}, r_{\text{new}} \)
- 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!

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

\[
\begin{pmatrix}
\frac{1}{2} & \frac{1}{2} & 0 \\
\frac{1}{2} & 0 & 0 \\
0 & \frac{1}{2} & 1 \\
\end{pmatrix} =
\begin{pmatrix}
\frac{7}{15} & \frac{7}{15} & \frac{1}{15} \\
\frac{7}{15} & \frac{1}{15} & \frac{1}{15} \\
\frac{1}{15} & \frac{7}{15} & \frac{13}{15} \\
\end{pmatrix}
\]
Suppose there are $N$ pages.

Consider page $j$, with $d_j$ out-links.

We have $M_{ij} = \frac{1}{|d_j|}$ 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 and setting transition probability to $(1-\beta)/N$.
- Reducing the probability of following each out-link from $1/|d_j|$ to $\beta/|d_j|$.
- Equivalent: Tax each page a fraction $(1-\beta)$ of its score and redistribute evenly.
Rearranging the Equation

- \[ r = A \cdot r, \text{ where } A_{ij} = \beta M_{ij} + \frac{1-\beta}{N} \]
- \[ r_i = \sum_{j=1}^{N} A_{ij} \cdot r_j \]
- \[ r_i = \sum_{j=1}^{N} \left[ \beta M_{ij} + \frac{1-\beta}{N} \right] \cdot r_j \]
  - \[ = \sum_{j=1}^{N} \beta M_{ij} \cdot r_j + \frac{1-\beta}{N} \sum_{j=1}^{N} r_j \]
  - \[ = \sum_{j=1}^{N} \beta M_{ij} \cdot r_j + \frac{1-\beta}{N} \] 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 \ldots a vector of length N with all entries x
We just rearranged the PageRank equation

\[ r = \beta M \cdot r + \left[ \frac{1 - \beta}{N} \right] \]

- where \([(1-\beta)/N]_N\) is a vector with all \(N\) entries \((1-\beta)/N\)

\(M\) is a sparse matrix! (with no dead-ends)
- 10 links per node, approx 10\(N\) entries

So in each iteration, we need to:
- Compute \(r^{\text{new}} = \beta M \cdot r^{\text{old}}\)
- Add a constant value \((1-\beta)/N\) to each entry in \(r^{\text{new}}\)
  - Note if \(M\) contains dead-ends then \(\sum_i r_i^{\text{new}} < 1\) and we also have to renormalize \(r^{\text{new}}\) so that it sums to 1
PageRank: The Complete Algorithm

- **Input:** Graph $G$ and parameter $\beta$
  - Directed graph $G$ with *spider traps* and *dead ends*
  - Parameter $\beta$

- **Output:** PageRank vector $r$

  - **Set:** $r_j^{(0)} = \frac{1}{N}, \quad t = 1$
  - **do:**
    - $\forall j$: $r_j^{(t)} = \sum_{i \rightarrow j} \beta \frac{r_i^{(t-1)}}{d_i}$
    - $r_j^{(t)} = 0$ if in-deg. of $j$ is 0
  - **Now re-insert the leaked PageRank:**
    - $\forall j$: $r_j^{(t)} = r_j^{(t)} + \frac{1-S}{N}$ where: $S = \sum_j r_j^{(t)}$
  - $t = t + 1$
  - **while** $\sum_j \left| r_j^{(t)} - r_j^{(t-1)} \right| > \varepsilon$
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 $r^{new}$ into memory
  - Store $r^{old}$ and matrix $M$ on disk
- Then 1 step of power-iteration is:
  Initialize all entries of $r^{new}$ to $(1-\beta)/N$
  For each page $p$ (of out-degree $n$):
    Read into memory: $p$, $n$, $dest_1, \ldots, dest_n$, $r^{old}(p)$
    for $j = 1\ldots n$: $r^{new}(dest_j) += \beta r^{old}(p) / n$

<table>
<thead>
<tr>
<th>src</th>
<th>degree</th>
<th>destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>1, 5, 6</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>17, 64, 113, 117</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>13, 23</td>
</tr>
</tbody>
</table>
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>
<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>
```

$r^{new}$

$r^{old}$
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>

<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>

<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>

<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)