Lecture: Link Analysis

http://bicmr.pku.edu.cn/~wenzw/bigdata2016.html

Acknowledgement: this slides is based on Prof. Jure Leskovec’s lecture notes
Outline

1. Introduction
2. PageRank
3. PageRank in Reality
4. Extensions
   - Topic-Specific PageRank
   - TrustRank: combating the web spam
Communication networks

![Diagram of communication networks]
How to organize the Web?

First try: Human curated Web directories
  - Yahoo, baidu, hao123

Second try: Web Search
  - Information Retrieval investigates: Find relevant docs in a small and trusted set

But: Web is huge, full of untrusted documents, random things, web spam, etc.
Web as a directed graph:

- Nodes: Webpages;
- Edges: Hyperlinks
Web as a directed graph:
Nodes: Webpages; Edges: Hyperlinks
Three basic things of search engines

- Crawl the web and locate all web pages with public access.
- Index the data from step 1, so that it can be searched efficiently for relevant keywords or phrases.
- Rate the importance of each page in the database, so that when a user does a search and the subset of pages in the database with the desired information has been found, the more important pages can be presented first.
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
All web pages are not equally "important"

www.pku.edu.cn vs.
www.tsinghua.edu.cn

There is large diversity in the web-graph node connectivity

Let’s rank the pages by the link structure!
Outline

1. Introduction
2. PageRank
3. PageRank in Reality
4. Extensions
   - Topic-Specific PageRank
   - TrustRank: combating the web spam
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.pku.edu.cn: 6,649 links
  - www.tsinghua.edu.cn: 8579 links

- Are all in-links equal?
  - Links from important pages count more
  - Recursive question!
# Links as votes

## What sites link to pku.edu.cn?

<table>
<thead>
<tr>
<th>Site</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>baidu.com</td>
</tr>
<tr>
<td>2.</td>
<td>msn.com</td>
</tr>
<tr>
<td>3.</td>
<td>qq.com</td>
</tr>
<tr>
<td>4.</td>
<td>hupu.com</td>
</tr>
<tr>
<td>5.</td>
<td>163.com</td>
</tr>
</tbody>
</table>

**Total Sites Linking In:** 6,649

## What sites link to tsinghua.edu.cn?

<table>
<thead>
<tr>
<th>Site</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>yahoo.com</td>
</tr>
<tr>
<td>2.</td>
<td>baidu.com</td>
</tr>
<tr>
<td>3.</td>
<td>msn.com</td>
</tr>
<tr>
<td>4.</td>
<td>yandex.ru</td>
</tr>
<tr>
<td>5.</td>
<td>qq.com</td>
</tr>
</tbody>
</table>

**Total Sites Linking In:** 8,579
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.

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

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

where $d_i$ is the out-degree of node $i$
Solving the flow equations

“Flow” equations:
\[ r_y = r_y / 2 + r_a / 2 \]
\[ r_a = r_y / 2 + r_m \]
\[ r_m = r_a / 2 \]

- 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!
PageRank: matrix formulation

- Stochastic adjacency matrix $M$
  - Let page $i$ has $d_i$ out-links

\[ M_{ji} = \begin{cases} \frac{1}{d_i} & \text{if } i \rightarrow j \\ 0 & \text{otherwise} \end{cases} \]

- $M$ is a column stochastic matrix (column 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 equation $r_j = \sum_{i \rightarrow j} \frac{r_i}{d_i}$ can be written as

\[ r = Mr \]
Example

- Remember the flow equation: \( r_j = \sum_{i \rightarrow j} \frac{r_i}{d_i} \)
- Flow equation in the matrix form: \( Mr = r \)
  - Suppose page \( i \) links to 3 pages, including \( j \)
NOTE: $x$ is an eigenvector of $A$ with the corresponding eigenvalue $\lambda$ if: $Ax = \lambda x$

Flow equation in the matrix form: $Mr = r$

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. We know $r$ is unit length and each column of $M$ sums to one, so $Mr \leq 1$

We can now efficiently solve for $r$ through *power iteration*
Example: flow equations

\[ r_y = r_y/2 + r_a/2 \]
\[ r_a = r_y/2 + r_m \]
\[ r_m = r_a/2 \]

\[
\begin{bmatrix}
y \\
a \\
m
\end{bmatrix}
= \begin{bmatrix}
\frac{1}{2} & \frac{1}{2} & 0 \\
\frac{1}{2} & 0 & 1 \\
0 & \frac{1}{2} & 0
\end{bmatrix}
\begin{bmatrix}
y \\
a \\
m
\end{bmatrix}
\]

\[ r = M \cdot r \]
Power iteration

- 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: $r^{(0)} = [1/N, \ldots, 1/N]^\top$
  - Iterate: $r^{(t+1)} = Mr^{(t)}$, i.e.,
    \[
    r_{j}^{t+1} = \sum_{i \rightarrow j} \frac{r_{i}^{(t)}}{d_i}, \quad d_i : \text{out-degree of node } i
    \]
  - Stop when $\|r^{(t+1)} - r^{(t)}\|_1 \leq \epsilon$
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
The stationary distribution

Where is the surfer at time \( t + 1 \)?

Follows a link uniformly at random

\[ p_{t+1} = Mp_t \]

Suppose the random walk reaches a state

\[ p_{t+1} = Mp_t = p_t \]

then \( p_t \) is the **stationary distribution** of a random walk

Our original rank vector \( r \) satisfies \( r = Mr \)

So \( r \) is a stationary distribution for the random walk
PageRank: three questions

\[ \mathbf{r}_{j}^{t+1} = \sum_{i \rightarrow j} \frac{\mathbf{r}_{i}^{(t)}}{d_{i}}, \]

or equivalently

\[ \mathbf{r} = \mathbf{M}\mathbf{r} \]

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

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

Example:

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

Iteration 0, 1, 2, …
Does it converge to what we want?

Example:

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

Iteration 0, 1, 2, …
PageRank: problems

Two 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:

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

Iteration 0, 1, 2, …

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

$$r_y = \frac{r_y}{2} + \frac{r_a}{2}$$
$$r_a = \frac{r_y}{2}$$
$$r_m = \frac{r_a}{2} + r_m$$
Solution: random teleports

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 random page

Commonly $\beta \in [0.8, 0.9]$

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:

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

Iteration 0, 1, 2, …
Solution: always teleport

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

- Surfer will teleport out of spider trap within a few time steps

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

\[
\begin{array}{ccc}
  \ y & a & m \\
  y & \frac{1}{2} & \frac{1}{2} & \frac{1}{3} \\
  a & \frac{1}{2} & 0 & \frac{1}{3} \\
  m & 0 & \frac{1}{2} & \frac{1}{3} \\
\end{array}
\]
Why teleports solve the problem?

Let's denote the state at time $t+1$ as $r^{t+1}$ and the transition matrix as $M$. Then, we have:

$$r^{t+1} = Mr^{t}$$

Markov chains

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

- 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**

  (By the Perron-Frobenius theorem, an irreducible and aperiodic Markov chain is guaranteed to converge to a unique stationary distribution)
Make $\mathbf{M}$ stochastic

- (column)-stochastic: every column sums to 1
- A possible solution: add green links

$$\mathbf{A} = \mathbf{M} + \mathbf{a}^\top \left( \frac{1}{n} \mathbf{1} \right)$$

where $a_i = 1$ if node $i$ has out deg 0, otherwise $a_i = 0$

<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>$\frac{1}{3}$</td>
</tr>
<tr>
<td>a</td>
<td>$\frac{1}{2}$</td>
<td>0</td>
<td>$\frac{1}{3}$</td>
</tr>
<tr>
<td>m</td>
<td>0</td>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{3}$</td>
</tr>
</tbody>
</table>

$r_y = r_y/2 + r_a/2 + r_m/3$
$r_a = r_y/2 + r_m/3$
$r_m = r_a/2 + r_m/3$
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: random jumps

- Google’s solution that does it all:
  - Makes matrix \( M \) **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]

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

- 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
Google’s solution: random jumps

- PageRank equation [Brin-Page, 98]

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

- Since \( \mathbf{1}^\top \mathbf{r} = 1 \), the Google matrix \( \mathbf{A} \):

\[ \mathbf{A} = \beta \mathbf{M} + (1 - \beta) \frac{1}{n} \mathbf{1} \mathbf{1}^\top \]

- \( \mathbf{A} \) is stochastic, aperiodic and irreducible, so

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

- In practice \( \beta \in [0.8, 0.9] \) (make around 5 steps and jump)
Random teleports \((\beta = 0.8)\)

\[
\begin{align*}
\kappa & = 0.8 \\
\begin{bmatrix}
\mathbf{y} & \mathbf{a} & \mathbf{m}
\end{bmatrix}
& = 
\begin{bmatrix}
\frac{1}{3} & 0.33 & 0.24 & 0.26 & \ldots & 7/33 \\
\frac{1}{3} & 0.20 & 0.20 & 0.18 & \ldots & 5/33 \\
\frac{1}{3} & 0.46 & 0.52 & 0.56 & & 21/33
\end{bmatrix}

\end{align*}
\]
Simple proof using linear algebra

- Every stochastic matrix has 1 as an eigenvalue.
- \( V_1(A) \) : eigenspace for eigenvalue 1 of a stochastic matrix \( A \).

Fact 1: If \( M \) is positive and stochastic, then any eigenvector in \( V_1(M) \) has all positive or all negative components.

Fact 2: If \( M \) is positive and stochastic, then \( V_1(M) \) has dimension 1.
Proof of Fact 1

- Suppose \( x \in V_1(M) \) contains elements of mixed sign.
- Since \( M_{ij} > 0 \), each \( M_{ij}x_j \) are of mixed sign. Then

\[
|x_i| = | \sum_{j=1}^{n} M_{ij} x_j | < \sum_{j=1}^{n} M_{ij} |x_j|
\]

- Since \( M \) is stochastic, we can obtain a contradiction

\[
\sum_{i=1}^{n} |x_i| < \sum_{i=1}^{n} \sum_{j=1}^{n} M_{ij} |x_j| = \sum_{j=1}^{n} \left( \sum_{i=1}^{n} M_{ij} \right) |x_j| = \sum_{j=1}^{n} |x_j|
\]

- If \( x_j \geq 0 \) for all \( j \), then \( x_i > 0 \) since \( M_{ij} > 0 \) and not all \( x_j \) are zero.
Proof of Fact 2

- **Claim:** Let \( v, w \in \mathbb{R}^m \) with \( m \geq 2 \) and linearly independent. Then for some \( s \) and \( t \) that are not both zero, the vector \( x = sv + tw \) has both positive and negative components.
  - Linear independence implies neither \( v \) nor \( w \) is zero. Let \( d = \sum_i v_i \).
  - If \( d = 0 \), then \( v \) must contain mixed sign, and taking \( s = 1, t = 0 \).
  - If \( d \neq 0 \), set \( s = -\sum_i w_i / d \), \( t = 1 \) and \( x = sv + w \). Then \( x \neq 0 \) but \( \sum_i x_i = 0 \).

- **Fact 2: Proof by contradiction.** Suppose there are two linearly independent eigenvectors \( v \) and \( w \) in the subspace \( V_1(M) \). For any real numbers \( s \) and \( t \) that are not both zero, the nonzero vector \( x = sv + tw \) must be in \( V_1(M) \), and so have components that are all negative or all positive. But by the above claim, for some choice of \( s \) and \( t \) the vector \( x \) must contain components of mixed sign.
Convergence of Power Iteration

Claim 1: Let $M$ be positive and stochastic. Let $V$ be a subspace of $v$ such that $\sum_i v_i = 0$. Then $Mv \in V$ and $\|Mv\|_1 \leq c\|v\|_1$ for any $v \in V$ and $0 < c < 1$.

- Let $w = Mv$. Then $w \in V$ since

$$\sum_{i=1}^n w_i = \sum_{i=1}^n \sum_{j=1}^n M_{ij}v_j = \sum_{j=1}^n v_j \left( \sum_{i=1}^n M_{ij} \right) = \sum_{j=1}^n v_j = 0$$

- Let $e_i = \text{sgn}(w_i)$ and $a_j = \sum_{i=1}^n e_i M_{ij}$, then $e_i$ are of mixed sign

$$\|w\|_1 = \sum_{i=1}^n e_i w_i = \sum_{i=1}^n e_i \left( \sum_{j=1}^n M_{ij}v_j \right) = \sum_{j=1}^n a_j v_j$$

- Since $\sum_{i=1}^n M_{ij} = 1$ with $0 < M_{ij} < 1$, there exists $0 < c < 1$ such that $|a_j| < c$. 
Claim 2: Let $M$ be positive and stochastic. Then it has a unique $q > 0$ such that $Mq = q$ with $\|q\|_1 = 1$. The vector $q$ can be computed as $q = \lim_{k \to \infty} M^k x_0$ with $x_0 > 0$ and $\|x_0\|_1 = 1$.

- The existence of $q$ has been proved.

- We can write $x_0 = q + v$ where $v \in V$ defined in Claim 1. We have

$$M^k x_0 = M^k q + M^k v = q + M^k v$$

- Since $\|M^k v\|_1 \leq c^k \|v\|_1$ for $0 < c < 1$, then $\lim_{k \to \infty} M^k x_0 = q$. 
Convergence rate of Power Iteration

- \( r^{(1)} = Mr^{(0)}, r^{(2)} = Mr^{(1)} = M^2 r^{(0)}, \ldots \)

- **Claim:** The sequence \( Mr^{(0)}, \ldots, M^k r^{(0)}, \ldots \) approaches the dominant eigenvector of \( M \).

- **Proof:** Assume \( M \) has \( n \) linearly independent eigenvectors, \( x_1, x_2, \ldots, x_n \) with corresponding eigenvalues \( \lambda_1, \lambda_2, \ldots, \lambda_n \) such that \( \lambda_1 > \lambda_2 \geq \ldots \geq \lambda_n \)

- Since \( x_1, x_2, \ldots, x_n \) is a basis in \( \mathbb{R}^n \), we can write

\[
r^{(0)} = c_1 x_1 + c_2 x_2 + \ldots + c_n x_n
\]

- Using \( M x_i = \lambda_i x_i \), we have

\[
Mr^{(0)} = M (c_1 x_1 + c_2 x_2 + \ldots + c_n x_n) = \sum_{i=1}^{n} c_i \lambda_i x_i
\]
Convergence rate of Power Iteration

Repeated multiplication on both sides produces

$$M^k r^{(0)} = \sum_{i=1}^{n} c_i \lambda_i^k x_i$$

$$= \lambda_1^k \left( \sum_{i=1}^{n} c_i \left( \frac{\lambda_i}{\lambda_1} \right)^k x_i \right)$$

Since $\lambda_1 > \lambda_i$, $i = 2, \ldots, n$. Then $\frac{\lambda_i}{\lambda_1} < 1$, and $\lim_{k \to \infty} \left( \frac{\lambda_i}{\lambda_1} \right)^k = 0$, $i = 2, \ldots, n$.

Therefore,

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

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

1 Introduction

2 PageRank

3 PageRank in Reality

4 Extensions
   - Topic-Specific PageRank
   - TrustRank: combating the web spam
Computing the PageRank

- The matrix $A = \beta M + (1 - \beta) \frac{1}{N} 11^\top$

- Key step is matrix-vector multiplication

  $$r^{new} = Ar^{old}$$

- Easy if we have enough main memory to hold $A$, $r^{old}$, $r^{new}$

- Suppose there are $N = 1$ billion pages
  - Suppose we need 4 bytes for each entry
  - 2 billion entries for vectors, approx 8GB
  - Matrix $A$ has $N^2$ entries - $10^{18}$ is huge!
Sparse matrix formulation

- We just rearranged the PageRank equation

\[ r = \beta M r + \frac{1 - \beta}{N} \mathbf{1}_N \]

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

- So in each iteration, we need to
  - Compute \( r^{new} = A r^{old} \)
  - Add a constant value \( (1 - \beta)/N \) to each entry in \( r^{new} \)
  - Note if \( M \) contains dead-ends then \( \sum_i r^{new}_i < 1 \) and we also have to renormalize \( r^{new} \) so that it sums to 1
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 $j = 1$ to $n$: $r_{\text{new}}(\text{dest}_j) += \beta r_{\text{old}}(p)/n$

<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>

<table>
<thead>
<tr>
<th>$r_{\text{new}}$</th>
<th>src</th>
<th>degree</th>
<th>destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>3</td>
<td>1, 5, 6</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>4</td>
<td>17, 64, 113, 117</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>13, 23</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$r_{\text{old}}$</th>
<th>src</th>
<th>degree</th>
<th>destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>3</td>
<td>1, 5, 6</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>4</td>
<td>17, 64, 113, 117</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>13, 23</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>
Assume enough RAM to fit $r^{\text{new}}$ into memory
- Store $r^{\text{old}}$ and matrix $M$ on disk

In each iteration, we have to
- Read $r^{\text{old}}$ and $M$
- Write $r^{\text{new}}$ back to disk
- IO cost: $2|r| + |M|$

Question: What if we could not even fit $r^{\text{new}}$ in memory?
### Block Based Update Algorithm

The table below illustrates the block-based update algorithm for a network with five nodes (0 to 5).

<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>

The diagram on the right visualizes the block-based update process.

- **$r^{new}$** indicates the updated routing table.
- **$r^{old}$** represents the previous routing table.

Nodes are updated in blocks to minimize network disruption.
Analysis of block update

- Similar to nested-loop join in databases
  - Break $r^{new}$ into $k$ blocks that fit in memory
  - Scan $M$ and $r^{old}$ once for each block

- $k$ scans of $M$ and $r^{old}$
  - $k(|r| + |M|) + |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

\[
\begin{array}{l}
\text{src} \quad \text{degree} \quad \text{destination} \\
0 \quad 4 \quad 0, 1 \\
1 \quad 3 \quad 0 \\
2 \quad 2 \quad 1 \\
\end{array}
\]

\[
\begin{array}{l}
\text{src} \quad \text{degree} \\
0 \quad 4 \quad 3 \\
2 \quad 2 \quad 3 \\
\end{array}
\]

\[
\begin{array}{l}
\text{src} \quad \text{degree} \\
0 \quad 4 \quad 5 \\
1 \quad 3 \quad 5 \\
2 \quad 2 \quad 4 \\
\end{array}
\]
Analysis of block stripe update

- Break $\mathbf{M}$ into stripes
  - Each stripe contains only destination nodes in the corresponding block of $\mathbf{r}^{new}$

- Some additional overhead per stripe
  - But it is usually worth it

- Cost per iteration: $|\mathbf{M}|(1 + \epsilon) + (k + 1)|\mathbf{r}|$
Some problems 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 e.g., hubs-and-authorities
  - Solution: Hubs-and-Authorities (HITS)

- Susceptible to Link spam
  - Artificial link topographies created in order to boost page rank
  - Solution: TrustRank
Outline

1 Introduction
2 PageRank
3 PageRank in Reality
4 Extensions
   • Topic-Specific PageRank
   • TrustRank: combating the web spam
Topic-Specific PageRank

- Instead of generic popularity, can we measure popularity within a topic?

- Goal: Evaluate Web pages not just according to their popularity, but by how close they are to a particular topic, e.g. "sports" or "history"

- Allows search queries to be answered based on interests of the user
  - Example: Query "Trojan" wants different pages depending on whether you are interested in sports, history and computer security
Topic-Specific PageRank

- Random walker has a small probability of teleporting at any step
- Teleport can go to:
  - Standard PageRank: Any page with equal probability
  - Topic Specific PageRank: A topic-specific set of "relevant" pages (teleport set)
- Idea: Bias the random walk
  - When walker teleports, she pick a page from a set $S$
  - $S$ contains only pages that are relevant to the topic
  - For each teleport set $S$, we get a different vector $r_S$
Matrix formulation

To make this work all we need is to update the teleportation part of the PageRank formulation

\[ A_{ij} = \begin{cases} \beta M_{ij} + (1 - \beta) / |S| & \text{if } i \in S \\ \beta M_{ij} & \text{otherwise} \end{cases} \]

- \( A \) is stochastic!
- We have weighted all pages in the teleport set \( S \) equally
  - Could also assign different weights to pages
- Compute as for regular PageRank
  - Multiply by \( M \), then add a vector
  - Maintains sparseness
Suppose $S = \{1\}, \beta = 0.8$

<table>
<thead>
<tr>
<th>Node</th>
<th>Iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0.25</td>
</tr>
<tr>
<td>2</td>
<td>0.25</td>
</tr>
<tr>
<td>3</td>
<td>0.25</td>
</tr>
<tr>
<td>4</td>
<td>0.25</td>
</tr>
</tbody>
</table>

$S = \{1, 2, 3, 4\}, \beta = 0.8$: \[r = [0.13, 0.10, 0.39, 0.36]\]

$S = \{1, 2, 3\}, \beta = 0.8$: \[r = [0.17, 0.13, 0.38, 0.30]\]

$S = \{1, 2\}, \beta = 0.8$: \[r = [0.26, 0.20, 0.29, 0.23]\]

$S = \{1\}, \beta = 0.8$: \[r = [0.29, 0.11, 0.32, 0.26]\]

$S = \{1\}, \beta = 0.90$: \[r = [0.17, 0.07, 0.40, 0.36]\]

$S = \{1\}, \beta = 0.8$: \[r = [0.29, 0.11, 0.32, 0.26]\]
Discovering the topic

- Create different PageRanks for different topics

Which topic ranking to use?
- User can pick from a menu
- Classify query into a topic
- Can use the context of the query
  - E.g., query is launched from a web page talking about a known topic
  - History of queries e.g., "basketball" followed by "Jordan"
- User context, e.g., user’s bookmarks, ...
TrustRank: combating the web spam
Web spam

- Spamming: any deliberate action to boost a web page’s position in search engine results, incommensurate with page’s real value
- Spam: web pages that are the result of spamming
- This is a very broad definition
  - SEO (Search Engine Optimization) industry might disagree!
- Approximately 10-15% of web pages are spam
Web search

- Early search engines:
  - Crawl the Web
  - Index pages by the words they contained
  - Respond to search queries (lists of words) with the pages containing those words

- Early page ranking:
  - Attempt to order pages matching a search query by "importance"
  - First search engines considered
    1. Number of times query words appeared
    2. Prominence of word position, e.g. title, header
First spammers

- As people began to use search engines to find things on the Web, those with commercial interests tried to exploit search engines to bring people to their own site — whether they wanted to be there or not.

  Example: shirt-sellers might pretend to be about "movies"
  - Add the word movie 1,000 times to your page and set text color to the background color
  - Or, run the query "movie" on your target search engine, copy the first result into your page and make it "invisible"

- Techniques for achieving high relevance/importance for a web page

- These and similar techniques are term spam
Google’s solution to term spam

- Believe what people say about you, rather than what you say about yourself
  - Use words in the anchor text (words that appear underlined to represent the link) and its surrounding text
- PageRank as a tool to measure the "importance" of Web pages
Why it works?

- Our hypothetical shirt-seller looses
  - Saying he is about movies doesn’t help, because others don’t say he is about movies
  - His page isn’t very important, so it won’t be ranked high for shirts or movies

- Example:
  - Shirt-seller creates 1,000 pages, each links to his with "movie" in the anchor text
  - These pages have no links in, so they get little PageRank
  - So the shirt-seller can’t beat truly important movie pages like IMDB
Spam farming

- Once Google became the dominant search engine, spammers began to work out ways to fool Google

- **Spam farms** were developed to concentrate PageRank on a single page

- Link farm: creating link structures that boost PageRank of a particular page
Link spamming

Three kinds of web pages from a spammer’s point of view

- Inaccessible pages
- Accessible pages
  - e.g., blog comments pages
  - Spammer can post links to his pages
- Own pages
  - Completely controlled by spammer
  - May span multiple domain names
Link farms

- Spammer’s goal: maximize the PageRank of target page \( t \)
- Technique:
  - Get as many links from accessible pages as possible to target page \( t \)
  - Construct "link farm" to get PageRank multiplier effect
Analysis

- $x$: PageRank contributed by accessible pages
- $y$: PageRank of target page $t$
- Rank of each "farm" page $= \frac{\beta y}{M} + \frac{1-\beta}{N}$

$$
y = x + \beta M \left[ \frac{\beta y}{M} + \frac{1-\beta}{N} \right] + \frac{1-\beta}{N}
= x + \beta^2 y + \frac{\beta (1-\beta) M}{N} + \frac{1-\beta}{N}
$$

Ignore the last term (very small) and solve for $y$:

$$
y = \frac{x}{1-\beta^2} + c \frac{M}{N}
$$

where $c = \frac{\beta}{1+\beta}$

- For $\beta = 0.85$, $1/(1-\beta^2) = 3.6$
- Multiplier effect for "acquired" PageRank
- By making $M$ large, we can make $y$ as large as we want
Combating spam

- **Combating term spam**
  - Analyze text using statistical methods
  - Similar to email spam filtering
  - Also useful: Detecting approximate duplicate pages

- **Combating link spam**
  - Detection and blacklisting of structures that look like spam farms
  - TrustRank = topic-specific PageRank with a teleport set of "trusted" pages
TrustRank: idea

- Basic principle: Approximate isolation
  - It is rare for a "good" page to point to a "bad" (spam) page

- Sample a set of seed pages from the web

- Have an oracle (human) to identify the good pages and the spam pages in the seed set
  - Expensive task, so we must make seed set as small as possible
Trust propagation

- Call the subset of seed pages that are identified as good the trusted pages
- Perform a topic-sensitive PageRank with teleport set = trusted pages
  - Propagate trust through links: each page gets a trust value between 0 and 1
- Use a threshold value and mark all pages below the trust threshold as spam
Why is it a good idea?

- **Trust attenuation**
  - The degree of trust conferred by a trusted page decreases with the distance in the graph

- **Trust splitting**
  - The larger the number of out-links from a page, the less scrutiny the page author gives each out-link
  - Trust is split across out-links
Picking the seed set

Two conflicting considerations

- Human has to inspect each seed page, so seed set must be as small as possible
- Must ensure every good page gets adequate trust rank, so need make all good pages reachable from seed set by short paths

Suppose we want to pick a seed set of $k$ pages, how?

1. PageRank: pick the top-$k$ pages by PageRank
2. Use trusted domains, e.g. .edu, .mil, .gov