**Today’s topics**

- FAQs
- Dead end and Spider Traps
- Managing sparse matrix for PageRank with MapReduce
- Link Spam and Link Farm

---

**FAQs**

- PA2 is available

---

**Problems we need to avoid**

- Dead end
  - A page that has no links out
  - Surfers reaching such a page will disappear
  - In the limit, no page that can reach a dead end can have any PageRank at all

- Spider traps
  - Groups of pages that all have outlinks but they never link to any other pages

---

**Link Analysis**

Challenges in PageRank Algorithm for the real Web

(1) Dead ends
Avoiding Dead Ends

- If we allow dead ends
  - The transition matrix of the Web is no longer stochastic
    - Some of the columns will sum to 0 rather than 1

- If we compute $M^i$ for increasing powers of a substochastic matrix
  - Some of all of the components of the vector go to 0
  - A matrix whose column sums are at most 1
  - Importance “drains out” of the Web
  - No information about the relative importance of pages

\[
M = \begin{bmatrix}
0 & 1/2 & 0 & 0 \\
1/3 & 0 & 0 & 1/2 \\
1/3 & 1/2 & 0 & 0 \\
1/3 & 0 & 1/2 & 0 \\
\end{bmatrix}
\]

- Remove the arc from C to A
  - C becomes a dead end
  - If a random surfer reaches C, they disappear at the next round

\[
M' = \begin{bmatrix}
0 & 1/2 & 0 & 0 \\
1/3 & 0 & 0 & 1/2 \\
1/3 & 1/2 & 0 & 0 \\
1/3 & 0 & 1/2 & 0 \\
\end{bmatrix}
\]

Repeatedly multiplying the vector by $M$:

\[
\begin{bmatrix}
1/4 \\
1/4 \\
1/4 \\
1/4 \\
\end{bmatrix}
\begin{bmatrix}
5/24 \\
5/24 \\
5/24 \\
5/24 \\
\end{bmatrix}
\begin{bmatrix}
21/288 \\
31/288 \\
31/288 \\
31/288 \\
\end{bmatrix}
\begin{bmatrix}
0 \\
0 \\
0 \\
0 \\
\end{bmatrix}
\]

- The probability of a surfer being anywhere goes to 0 as the number of steps increase

Example of recursive deletion (1/4)

Approaches to dealing with dead ends

- Recursive deletion
  - Drop the dead ends from the graph
  - Drop their incoming arcs as well
  - Doing so may create more dead ends
  - Drop those new dead ends

- Taxation
  - Modify the process by which random surfers are assumed to move about the Web

Example of recursive deletion (2/4)

The final matrix for the graph is

\[
M = \begin{bmatrix}
0 & 1/2 & 0 \\
1/2 & 0 & 1 \\
1/2 & 1/2 & 0 \\
\end{bmatrix}
\begin{bmatrix}
1/6 \\
1/6 \\
1/6 \\
\end{bmatrix}
\begin{bmatrix}
5/12 \\
11/24 \\
8/24 \\
\end{bmatrix}
\begin{bmatrix}
2/9 \\
4/9 \\
5/9 \\
\end{bmatrix}
\]
Example of recursive deletion (3/4)

- We still need to compute deleted nodes (C and E)
  - C was the last to be deleted
  - We know all its predecessors have PageRanks (A and D)
  - Therefore,
    - PageRank of C = $\frac{1}{3} \times \frac{2}{9} + \frac{1}{2} \times \frac{3}{9} = \frac{13}{54}$

Example of recursive deletion (3/4)

- Now, we can compute the PageRank for E
  - Only one predecessor, C
    - The PageRank of E is the same as that of C ($\frac{13}{54}$)
  - The sums of the PageRanks exceeds 1
    - It cannot represent the distribution of a random surfer
    - It provides a good estimate

Spider Traps and Taxation

- Spider traps
  - A set of nodes with no dead ends but no arcs out (from the set of nodes)
  - This can appear intentionally or unintentionally on the Web
- Spider traps causes the PageRank calculation to place all the PageRank within the spider traps

Example 1

- There is a simple spider trap of 4 nodes

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1/2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1/3</td>
<td>0</td>
<td>0</td>
<td>1/2</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1/3</td>
<td>1/2</td>
<td>0</td>
<td>0</td>
<td>1/2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1/2</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

- If we perform the usual iteration to compute the PageRank of the nodes, we get

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1/6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1/6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1/6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1/6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>1/12</th>
<th>5/36</th>
<th>11/36</th>
<th>4/18</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.042</td>
<td>0.046</td>
<td>0.194</td>
<td>0.222</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Link Analysis

*Challenges in PageRank Algorithm for the real Web*

(2) Spider Traps
Example 2

- The arc out of C changed to point to C itself
- C is a simple spider trap of one node
  - It still has outgoing edge to itself
  - It is not a dead end

\[
M = \begin{bmatrix}
0 & 1/2 & 0 & 0 \\
1/3 & 0 & 0 & 1/2 \\
1/3 & 0 & 1 & 1/2 \\
1/3 & 1/2 & 0 & 0 \\
\end{bmatrix}
\]

If we perform the usual iteration to compute the PageRank of the nodes, we get

\[
\begin{bmatrix}
1/4 \\
5/24 \\
11/24 \\
1/3 \\
\end{bmatrix}
\begin{bmatrix}
5/48 \\
7/48 \\
29/48 \\
7/48 \\
\end{bmatrix}
\begin{bmatrix}
21/288 \\
31/288 \\
205/288 \\
31/288 \\
\end{bmatrix}
\begin{bmatrix}
0 \\
0 \\
1 \\
0 \\
\end{bmatrix}
\]

- All the PageRank is at C
  - Once there, a random surfer can never leave

To avoid Spider traps, we modify the calculation of PageRank

- Allowing each random surfer a small probability of teleporting to a random page
- Rather than following an out-link from their current page

The iterative step, where we compute a new vector estimate of PageRanks \( \mathbf{v}' \) from the current PageRank estimate \( \mathbf{v} \) and the transition matrix \( M \) is

\[
\mathbf{v}' = \beta M \mathbf{v} + (1 - \beta) \mathbf{e} / n
\]

- Where \( \beta \) is a chosen constant
  - Usually in the range 0.8 to 0.9

- \( \mathbf{e} \) is a vector for all 1’s with the appropriate number of components
- \( n \) is the number of nodes in the Web graph

If the graph has no dead ends

- The probability of introducing a new random surfer is exactly equal to the probability that the random surfer will decide not to follow a link from their current page
- Surfer decides either to follow a link or teleport to a random page
- If the graph has dead ends
  - The surfer goes nowhere
  - The term \((1 - \beta)v_0/n\) does not depend on the sum of the components of the vector \(v\), there will be some fraction of a surfer operating on Web
- When there are dead ends, the sum of the components of \(v\) may be less than 1
  - But it will never reach 0

---

**Example of Taxation (1/2)**

\[
\begin{pmatrix}
0 & 1/2 & 0 & 0 \\
1/3 & 0 & 0 & 1/2 \\
1/3 & 1/2 & 0 & 0 \\
\end{pmatrix}
\]

\[
v' = \beta Mv + (1 - \beta)e/n
\]

\[
\beta = 0.8 \\
v' =
\begin{pmatrix}
0 & 2/5 & 0 & 0 \\
4/15 & 0 & 0 & 2/5 \\
4/15 & 2/5 & 0 & 0 \\
\end{pmatrix}
\]

---

**Example of Taxation (2/2)**

- First few iterations:

\[
\begin{array}{cccc}
\frac{1}{3} & \frac{1}{2} & \frac{1}{2} \\
\frac{1}{3} & \frac{1}{2} & 0 \\
\frac{1}{3} & \frac{1}{2} & 0 \\
\end{array}
\]

\[
\begin{pmatrix}
19/300 & 513/300 & 543/4500 \\
707/4500 & 19/148 \\
707/4500 & 19/148 \\
\end{pmatrix}
\]

- By being a spider trap, C gets more than \(\frac{1}{2}\) of the PageRank for itself
- Effect is limited
- Each of the nodes gets some of the PageRank

---

**Example 1**

- Compute the PageRank of each page assuming \(\beta=0.8\)

\[
M =
\begin{pmatrix}
1/3 & 1/2 & 1/2 \\
1/3 & 0 & 1/2 \\
1/3 & 1/2 & 0 \\
\end{pmatrix}
\]

\[
v_0 =
\begin{pmatrix}
1/3 \\
1/3 \\
1/3 \\
\end{pmatrix}
\]

\[
\beta M v_0 + (1 - \beta)e/n
\]

\[
= 0.8 \times
\begin{pmatrix}
1/3 & 1/3 & 1/3 \\
1/3 & 1/3 & 1/3 \\
1/3 & 1/3 & 1/3 \\
\end{pmatrix}
\times
\begin{pmatrix}
1/3 \\
1/3 \\
1/3 \\
\end{pmatrix}

+ (1-0.8)e/3
\]

---

**Example 2**

- **clique**
  - Set of nodes with all possible arcs from one to another

- Suppose the Web consists of a clique of \(n\) nodes and a single additional node that is the successor of each of the \(n\) nodes in the clique
Example 2

- Determine the PageRank of each page assuming $\beta=0.8$
- Is there a dead end?
- Is there a spider trap?

Example 3

- Suppose that we recursively eliminate dead ends from the Web graph to solve the remaining graph
- Suppose that the graph is a chain of dead ends, headed by a node with a self-loop
- What would be the PageRank assigned to each of the nodes?

Example 3

• Remove all of the dead ends recursively

- What is $v_0$ and $M$?
Example 3

- What is $v_0$ and $M$?

  - $v_0 = [1]$, $M = [1]$
  - PageRank of $A = 1$
  - PageRank of $B = \frac{1}{2}$
  - PageRank of $C = \frac{1}{2}$
  - PageRank of $D = \frac{1}{2}$

...
Representing Transition Matrices (1/2)

- The average Web page has about 10 out-links
  - We are analyzing a graph of 1.4 billion pages
  - Only one in 0.14 billion (140 million) entries is not 0
- Can we list the location of the nonzero entries and their values?
  - If we use two 4-byte integers for coordinates (row#, col#) of an element and an 8-byte double-precision number for the probability value
  - 16-bytes per nonzero entry
  - The space needed is linear of nonzero entries

M =
\[
\begin{pmatrix}
0 & 1/2 & 0 & 0 \\
1/3 & 0 & 0 & 1/2 \\
1/3 & 0 & 1 & 0 \\
1/3 & 1/2 & 0 & 0
\end{pmatrix}
\]

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

Mapper generates
(key, value) = (destinations, 1/degree)
  - e.g. for the source A, (B, 1/3), (C, 1/3), (D, 1/3)
  - For the source B, (A, 1/2), (D, 1/2)
  - For the source C, (C, 1)
  - For the source D, (B, 1/2), (C, 1/2)

Reducer calculates

Value x (key's index)^th item in v

Suppose that the index for A = 0, B = 1, C = 2, and D = 3

\[ v_{\text{previous}} = \langle l, m, n, o \rangle \]

\[ v_{\text{current}} = \beta M v + \frac{1 - \beta}{n} \]

PageRank Iteration Using MapReduce

- One iteration of the PageRank algorithm involves,
  \[ v' = \beta M v + \frac{(1 - \beta)}{n} \]
  - First round of MapReduce
    - Calculate \( Mv \) and store the result to \( v' \)
  - Second round of MapReduce
    - For each component, multiply \( \beta \) and add \( \frac{(1 - \beta)}{n} \)

PageRank Iteration Using MapReduce

- If \( n \) is small enough that each Map task can store the full vector \( v \) in main memory
  - And \( v' \)
- For the Web, \( v \) is much too large to fit in main memory
- We need striping
  - \( M \) into vertical stripes and break \( v \) into corresponding horizontal strips

Link Analysis

Link spam
Architecture of a Spam Farm

- **Spam Farm**
  - A collection of pages whose purpose is to increase the PageRank of a certain page or pages

- From the point of view of the spammer, the Web is divided into two parts
  - Inaccessible pages
    - The pages that the spammer cannot affect
    - Most of the Web
  - Accessible pages
    - Those pages that, while they are not controlled by the spammer, can be affected by the spammer

The Web from the point of view of the link spammer

Understanding Spam Farm (1/2)

- Setting the links to the target page
  - Without link from outside, the spam farm is not useful
  - e.g. Blogs or news papers
  - Comments like “I agree. Please see my article at www.mySpamFarm.com”

Understanding Spam Farm (2/2)

- There is one page \( t \), the target page
  - Spammer attempts to place as much PageRank as possible

- There are a large number of \( m \) of supporting pages
  - Accumulate the portion of the PageRank that is distributed equally to all pages
  - The fraction \( 1 - \beta \) of the PageRank that represents surfers going to a random page
  - Prevent the PageRank of \( t \) from being lost
  - Note that all of the supporting pages links only to \( t \)

Analysis of a Spam Farm (1/6)

- A taxation parameter \( \beta \)
  - The fraction of a page’s PageRank that gets distributed to its successors at the next round

- Let there be,
  - \( n \) pages on the Web in total
  - A target page \( t \)
  - \( m \) supporting pages

Analysis of a Spam Farm (2/6)

- Let \( x \) be the amount of PageRank contributed by the accessible pages
  - \( x \) is the sum over all accessible page \( p \) with a link to \( t \), of the PageRank of \( p \) times \( \beta \) divided by the number of successors of \( p \)

- Finally, let \( y \) be the unknown PageRank of \( t \)
Analysis of a Spam Farm (3/6)

- The PageRank of each supporting page
  \[ \beta y = \beta y + (1 - \beta) / n \]
- First term represents the contribution from \( t \)
  - \( \beta y \) is distributed to \( t \)'s successors
  - Second term is the supporting page's share of the fraction \( 1 - \beta \) of the PageRank that is divided equally among all pages on the Web

Analysis of a Spam Farm (4/6)

- PageRank of \( y \) of target page \( t \) is \( (1) + (2) + (3) \)
  1. Contribution \( x \) from outside
  2. \( \beta \) times the PageRank of every supporting page
    - \( \beta m (\beta y/m + (1 - \beta)/n) \)
  3. \( (1 - \beta)/n \), the share of the fraction \( 1 - \beta \) of the PageRank that belongs to \( t \)
    - This amount is negligible

Analysis of a Spam Farm (5/6)

- From (1) and (2),
  \[ y = x + \beta m (\beta y/m + (1 - \beta)/n) = x + \beta y + \beta (1 - \beta) m/n \]
  \[ y = x(1 - \beta^2) + c m/n \]
  Where
  \[ c = \beta(1 - \beta) / (1 + \beta) \]

Analysis of a Spam Farm (6/6)

- If we choose \( \beta = 0.85 \), then \( 1/(1 - \beta^2) = 3.6 \)
  - \( c = \beta(1 + \beta) = 0.46 \)

  - The structure has amplified the external PageRank contribution by 360%
  - Also, it obtained an amount of PageRank that is 46% of the fraction of the Web, \( m/n \), that is in the spam farm

Combatting Link Spam

- Detecting and eliminating link spam have been critical for search engines
  - Just as it was critical to eliminate term spam in the previous decade

- Detecting particular structures
  - Spam farm
    - One page links to a very large number of pages
    - Each of which links back to it

Combatting Link Spam

- Modifying PageRank to lower the rank of link-spam pages automatically
  - TrustRank
  - Spam mass
TrustRank

- TrustRank is a topic-sensitive PageRank
  - "topic" is a set of pages believed to be trustworthy (not spam)
- Develop a suitable teleport set of trustworthy pages
  - Let humans examine a set of pages and decide which of them are trustworthy
- Pick a domain whose membership is controlled
  - University pages
    - .mil, or .gov

Calculating TrustRank (1/2)

- Then the topic-sensitive PageRank for $S$ is the limit of the iteration,
  \[ v' = \beta M v + (1 - \beta) e_S / |S| \]
- $M$ is the transition matrix of the Web, and $|S|$ is the size of set $S$

Calculating TrustRank (2/2)

- Suppose we use $\beta=0.8$, and our trust rank is represented by the teleport set $S={B,D}$

Spam Mass

- Measures the fraction of its PageRank that comes from spam for each page
- For an arbitrary page $p$,
  - Computing ordinary PageRank $r$
    - Computing the TrustRank $t$
      - Computing the TrustRank based on some teleport set of trustworthy pages
      - The spam mass $\frac{r - t}{r}$

Example

- Suppose that both the PageRank and TrustRank were computed
  - Teleport set was page $B$ and $D$
  - Which nodes are not the link spams?
  - Is there any link spam?

<table>
<thead>
<tr>
<th>Web Page</th>
<th>PageRank</th>
<th>TrustRank</th>
<th>SpamMass</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3/9</td>
<td>54/210</td>
<td>0.229</td>
</tr>
<tr>
<td>B</td>
<td>2/9</td>
<td>59/210</td>
<td>-0.264</td>
</tr>
<tr>
<td>C</td>
<td>2/9</td>
<td>38/210</td>
<td>0.188</td>
</tr>
<tr>
<td>D</td>
<td>2/9</td>
<td>59/210</td>
<td>-0.264</td>
</tr>
</tbody>
</table>