PART 1. LARGE SCALE DATA ANALYTICS
WEB-SCALE LINK AND SOCIAL NETWORK ANALYSIS
Sangmi Lee Pallickara
Computer Science, Colorado State University
http://www.cs.colostate.edu/~cs435

FAQs
• How to success in PA2 and PA3
  • Step 1: Attend the recitation class or watch the video clip
  • Step 2: Read your PA description carefully
  • Step 3: Work on your program by yourself
  • Step 4: Consult with your GTA, if needed

• For PA2 and 3, GTAs will not be allowed to access more than 30% of your code while they are helping with the debugging.
  • Select the relevant code snippets to discuss your programming assignment
Topics

• Large-scale Analytics 1. Web-Scale Link and Social Network Analysis

Part 1. Large Scale Data Analytics
1. Web-Scale Link Analysis and Social Network Analysis

Web-Scale Link Analysis
This material is built based on,

  - Chapter 5

- http://infolab.stanford.edu/~ullman/mmds.html

What does this matrix mean? [revisit]

- Multiplying the initial vector $v_0$ by $M$ a total of $i$ times
  - The distribution of the surfer after $i$ steps

\[
\begin{bmatrix}
0 & 1/2 & 1 & 0 \\
1/3 & 0 & 0 & 1/2 \\
1/3 & 0 & 0 & 1/2 \\
1/3 & 1.2 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
1/4 \\
1/4 \\
1/4 \\
1/4
\end{bmatrix}
\]

The probability for the next step from the current location  
The probability for being in the current location  
The distribution of the surfer after the first step  
- Probability for being in the next possible location
What does this matrix mean? [revisit]

• The distribution of the surfer approaches a limiting distribution \( \nu \) that satisfies \( \nu = M\nu \) provided two conditions are met:

1. The graph is strongly connected
   • It is possible to get from any node to any other node

2. There are no dead ends
   • Nodes that have no arcs out

What does this matrix mean? [revisit]

• The probability \( x_i \) that a random surfer will be at node \( i \) at the next step

\[
x_i = \sum_j m_{ij}v_j
\]

• \( m_{ij} \) is the probability that a surfer at node \( j \) will move to node \( i \) at the next step
• \( v_j \) is the probability that the surfer was at node \( j \) at the previous step
Example

• The sequence of approximations to the limit
  • We get by multiplying at each step by $M$ is

\[
\begin{bmatrix}
\frac{3}{4} & \frac{9}{24} & \frac{15}{48} & \frac{11}{32} & \frac{3}{9} \\
\frac{3}{4} & \frac{5}{24} & \frac{11}{48} & \frac{7}{32} & \frac{2}{9} \\
\frac{3}{4} & \frac{5}{24} & \frac{11}{48} & \frac{7}{32} & \ldots & \frac{2}{9} \\
\frac{3}{4} & \frac{5}{24} & \frac{11}{48} & \frac{7}{32} & \frac{2}{9}
\end{bmatrix}
\]

• Calculate until the difference in probability is not noticeable
• In the real Web, there are billions of nodes of greatly varying importance
  • The probability of being at a node like www.amazon.com is orders of magnitude greater than others

### VERSION 2: WITH VERY LARGE $v$

Division of a matrix and vector into five stripes

The $i$th stripe of the matrix multiplies only components from the $i$th stripe of the initial vector

Mapper 1
Mapper 2
Mapper 3
Mapper 4
Mapper 5
The $i$th stripe of the matrix multiplies only components from the $i$th stripe of the initial vector.

Data should be ordered by the page_IDs.

1. How will you design your input files?
Matrix $M$  
Initial Vector $v$

**Division of a matrix and vector into five stripes**

1. How will you design your Input files?

- Mapping blocks within a stripe of $M$ to the corresponding split of $v$:
  - $x^{th}$ block in the $i^{th}$ strip of $M$ should be paired with the $i^{th}$ split of $v$.
  - A split of $v$ will be paired with $l$ different blocks.

- $n$ splits of $v$  
- $(n \times l)$ blocks of $M$

See the `InputFormat()` for the Cartesian Product (in the Join design pattern).

2. What does Map function do?

- $k$ stripes
- $k$ splits

- $l$ blocks

The $i^{th}$ stripe of the matrix multiplies only components from the $i^{th}$ stripe of the initial vector.

Page 0: 1/n  
Page 1: 1/n  
Page 2: 1/n  
Page 3: 1/n

<table>
<thead>
<tr>
<th>Page</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.002</td>
<td>0.017</td>
<td>0.003</td>
<td>0.010</td>
<td></td>
</tr>
<tr>
<td>0.000</td>
<td>0.000</td>
<td>0.003</td>
<td>0.000</td>
<td></td>
</tr>
<tr>
<td>0.002</td>
<td>0.000</td>
<td>0.003</td>
<td>0.000</td>
<td></td>
</tr>
<tr>
<td>0.002</td>
<td>0.017</td>
<td>0.000</td>
<td>0.000</td>
<td></td>
</tr>
<tr>
<td>..</td>
<td>..</td>
<td>..</td>
<td>..</td>
<td></td>
</tr>
</tbody>
</table>
### Division of a matrix and vector into five stripes

The $i^{th}$ stripe of the matrix multiplies only components from the $i^{th}$ stripe of the initial vector.

<table>
<thead>
<tr>
<th>Page</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.002</td>
<td>0.017</td>
<td>0.003</td>
</tr>
<tr>
<td>1</td>
<td>0.000</td>
<td>0.000</td>
<td>0.003</td>
</tr>
<tr>
<td>2</td>
<td>0.002</td>
<td>0.000</td>
<td>0.003</td>
</tr>
<tr>
<td>3</td>
<td>0.002</td>
<td>0.017</td>
<td>0.000</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

### Mapper 1

**Key and value?**

- Page: 0: 1/n
- Page 1: 1/n
- Page 2: 1/n
- Page 3: 1/n

### Mapper 2

**2. What does Map function do?**

Calculate the local sum of multiplications between transition probability and the current page rank -- per row of blocks in $M$.

### Mapper 3

**3. What is the Mapper output?**

- Key and value?
3. What is the Mapper output?
Key: row number
Value: local sum

4. What does the reduce function do?
Division of a matrix and vector into five stripes

The $i$th stripe of the matrix multiplies only components from the $i$th stripe of the initial vector.

Mapper 1
Mapper 2
Mapper 3
Mapper 4
Mapper 5

4. What does the reduce function do?
Add all of the local sums of the row $k$, and store it in the $k$th element of $v$.

Part 1. Large Scale Data Analytics
1. Web-Scale Link Analysis and Social Network Analysis
PageRank Algorithm for the real Web
Structure of Web (1/3)

• Is the Web strongly connected?

Structure of Web (2/3)

Consisting of pages reachable from the In Component but not unable to reach the In Component

Consisting of pages that could reach SCC by following links but not reachable from the SCC

Consisting of pages reachable from the SCC, but unable to reach the SCC

Disconnected Components

In Component

Out Component

Strongly Connected Component

Tendrils In

Tendrils Out

Tubes
Structure of Web (3/3)

- **Tubes**
  - Pages reachable from the in-component and able to reach the output-component, but unable to reach the SCC or be reached from the SCC

- **Isolated components**
  - Unreachable from the large components

Anomalies from the Web structure

- These structures **violate the assumptions** needed for the Markov process **iteration** to converge to a limit
  - When a random surfer enters the out-component, they can never come back to SCC
  - Surfers starting in either the SCC or in-component are going to wind up in either the out-component or a tendril off the in-component
  - No page in the SCC or in-component winds up with any probability of a surfer being there

- **Nothing in the SCC or in-component will be of any importance**
Part 1. Large Scale Data Analytics

1. Web-Scale Link Analysis and Social Network Analysis

Challenges in PageRank Algorithm for the real Web

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 out links but they never link to any other pages
Part 1. Large Scale Data Analytics

1. Web-Scale Link Analysis and Social Network Analysis

Challenges in PageRank Algorithm for the real Web: (1) Dead ends

Avoiding Dead Ends

• **What 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 $M^i \nu$ for increasing powers of a substochastic matrix
  • Some of all of the components of the vector go to 0
  • substochastic matrix
    • A matrix whose column sums are **at most** 1
  • **Importance “drains out” of the Web**
    • No information about the relative importance of pages
• Remove the arc from C to A
  → C becomes a **dead end**

If a random surfer reaches C, they disappear at the next round

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

• Repeatedly multiplying the vector by M:

$$\begin{align*}
\frac{1}{4} & \begin{bmatrix} 3/24 \end{bmatrix} \\
\frac{1}{4} & \begin{bmatrix} 5/24 \end{bmatrix} \\
\frac{1}{4} & \begin{bmatrix} 5/24 \end{bmatrix} \\
\frac{1}{4} & \begin{bmatrix} 5/24 \end{bmatrix}
\end{align*}
\begin{align*}
\begin{bmatrix} 5/48 \end{bmatrix} \\
\begin{bmatrix} 7/48 \end{bmatrix} \\
\begin{bmatrix} 7/48 \end{bmatrix} \\
\begin{bmatrix} 7/48 \end{bmatrix}
\end{align*}
\begin{align*}
\begin{bmatrix} 21/288 \end{bmatrix} \\
\begin{bmatrix} 31/288 \end{bmatrix} \\
\begin{bmatrix} 31/288 \end{bmatrix} \\
\begin{bmatrix} 31/288 \end{bmatrix}
\end{align*}
\begin{align*}
0 \\
0 \\
0 \\
0
\end{align*}$$

• **The probability of a surfer being anywhere goes to 0** as the number of steps increase
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
    - Random surfers are assumed to move around on the Web without links

Example of recursive deletion (1/4)
Example of recursive deletion (1/4)

Diagram of recursive deletion:

A → B → C → D

C → D → A → B

Example of recursive deletion (1/4)

Diagram of recursive deletion:

A → B → C → D

C → D → A → B
Example of recursive deletion (1/4)

![Diagram of recursive deletion]

Example of recursive deletion (1/4)

![Diagram of recursive deletion]
Example of recursive deletion (2/4)

- The final matrix for the graph is

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

\[
\begin{bmatrix}
\frac{1}{3} & \frac{1}{6} & \frac{3}{12} & \frac{5}{24} & \cdots & \frac{2}{9} \\
\frac{1}{3} & \frac{3}{6} & \frac{5}{12} & \frac{11}{24} & \cdots & \frac{4}{9} \\
\frac{1}{3} & \frac{2}{6} & \frac{4}{12} & \frac{8}{24} & \cdots & \frac{3}{9} \\
\end{bmatrix}
\]

Example of recursive deletion (3/4)

- We still need to compute for the deleted nodes (C and E)

- C was the last one 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 (4/4)

- Now, we can compute the PageRank value for E
  - Only one predecessor, C
    - The PageRank of E is the same as that of C (13/54)

- The sums of the PageRanks exceeds 1
  - Still, it provides a good estimate

Part 1. Large Scale Data Analytics

1. Web-Scale Link Analysis and Social Network Analysis

Challenges in PageRank Algorithm for the real Web: (2) Spider Traps
Spider Traps and Taxation

- **Spider traps**
  - A set of nodes with **no dead ends but no arcs out** (from the set of nodes to the outside of the set)
  - 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

\[
M = \begin{bmatrix}
0 & 1/2 & 0 & 0 & 0 \\
1/3 & 0 & 0 & 1/2 & 0 \\
1/3 & 0 & 0 & 1/2 & 1/2 \\
1/3 & 1/2 & 1 & 0 & 0 & 1/2 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1/2 & 0
\end{bmatrix}
\]
• If we perform the usual iteration to compute the PageRank of the nodes, we get

\[
\begin{pmatrix}
\frac{1}{6} & 1/12 \\
\frac{1}{6} & \frac{5}{36} \\
\frac{1}{6} & \frac{11}{36} \\
\frac{1}{6} & \frac{4}{18} \\
\frac{1}{6} & 0 \\
\frac{1}{6} & 1/12
\end{pmatrix}
\begin{pmatrix}
0.042 \\
0.046 \\
0.194 \\
0.222 \\
0 \\
0
\end{pmatrix}
\]...

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

\[
M = \begin{pmatrix}
0 & 1/2 & 0 & 0 \\
1/3 & 0 & 0 & 1/2 \\
1/3 & 0 & 1 & 1/2 \\
1/3 & 1/2 & 0 & 0
\end{pmatrix}
\]
• If we perform the usual iteration to compute the PageRank of the nodes, we get

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

• **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 \( v' \) from the current PageRank estimate \( v \) and the transition matrix \( M \) is

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

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

• \( e \) 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 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 \((1-\beta)e/n\) 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)

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

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

\[ \beta = 0.8 \]

\[ v' = \begin{bmatrix} 0 & 2/5 & 0 & 0 \\ 13/60 & 53/300 & 2543/4500 & 95/148 \\ 41/300 & 543/4500 & 15/148 \\ 13/60 & 53/300 & 707/4500 & 19/148 \end{bmatrix} \]

Example of Taxation (2/2)

- First few iterations:

\[ \frac{1}{4} \begin{bmatrix} 9/60 \\ 13/60 \\ 25/60 \\ 13/60 \end{bmatrix} + \frac{1}{4} \begin{bmatrix} 41/300 \\ 53/300 \\ 153/300 \\ 53/300 \end{bmatrix} + \frac{1}{4} \begin{bmatrix} 543/4500 \\ 707/4500 \\ 2543/4500 \\ 707/4500 \end{bmatrix} + \frac{1}{4} \begin{bmatrix} 15/148 \\ 19/148 \\ 95/148 \\ 19/148 \end{bmatrix} \]

- By being a spider trap, C gets more than ½ of the PageRank for itself
  - Effect is limited
  - Each of the nodes gets some of the PageRank
Questions?