PART 1.
LARGE SCALE DATA ANALYSIS USING MAPREDUCE

Sangmi Lee Pallickara
Computer Science, Colorado State University
http://www.cs.colostate.edu/~cs435

Today’s topics
- FAQs
- Link Analysis

No CSU Scholarship App = No Merit Awards

At times, the CS Department wants to make an award of merit to a talented student. But we cannot.
Why? Because to qualify at least a basic CSU Scholarship application must be completed.
Details were sent out in email, so please find those details (or ask) and finish filling out the online form by March 1st!

FAQs
- Programming Assignment 2
  - Content Based Authorship Detection using TF/IDF Scores and Cosine Similarity
  - This assignment will require more than 40 hours of work
  - March 23 5:00PM via Canvas
  - Individual submission
  - Please do not submit your output file!

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
Link Analysis

Challenges in PageRank Algorithm for the real Web

(1) Dead ends

- 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 $M^v$ 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

Avoiding Dead Ends

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

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

- Repeatedly multiplying the vector $v$ by $M$:

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

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

Example of recursive deletion (1/4)
Example of recursive deletion (2/4)

- The final matrix for the graph is

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

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

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 = 1/3 x 2/9 + 1/2 x 3/9 = 13/54

Example 1

- There is a simple spider trap of 4 nodes

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

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

Link Analysis

Challenges in PageRank Algorithm for the real Web

(2) Spider Web
If we perform the usual iteration to compute the PageRank of the nodes, we get:

\[
\begin{bmatrix}
\frac{1}{6} & \frac{1}{12} & 0.042 \\
\frac{1}{6} & \frac{5}{36} & 0.046 \\
\frac{1}{6} & \frac{11}{36} & 0.194 \\
\frac{1}{6} & 0 & 0.222 \\
\frac{1}{6} & \frac{1}{12} & 0
\end{bmatrix}
\]

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.

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

To avoid Spider traps, we modify the calculation of PageRank:
- All the PageRank is at C.
  - Once there, a random surfer can never leave.

\[
v' = \beta Mv + \frac{(1 - \beta)}{n}
\]

The term \( \beta Mv \) represents the case where,
- With probability \( \beta \), the random surfer decides to follow an out-link from their present page.

The term \( (1-\beta)e/n \) is a vector
- Each of whose components has value \( (1-\beta)/n \)
  - Represents the introduction with probability \( 1-\beta \) of a new random surfer at a random page.
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 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)

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

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

\[
\beta = 0.8
\]

\[
v' = \begin{bmatrix}
0 & 2/5 & 0 & 0 \\
4/15 & 0 & 0 & 2/5 \\
4/15 & 0 & 4/5 & 2/5 \\
4/15 & 2/5 & 0 & 0
\end{bmatrix} v + \begin{bmatrix}
1/20 \\
1/20 \\
1/20 \\
1/20
\end{bmatrix}
\]

Example of Taxation (2/2)

First few iterations:

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

- By being a spider trap, C gets more than 1/3 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 $\beta=0.8$

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

\[
v' = \begin{bmatrix}
1/3 \\
1/3 \\
1/3
\end{bmatrix}
\]

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

\[
= 0.8 \times \begin{bmatrix}
1/2 & 1/2 & 1/2 \\
1/2 & 0 & 1/2 \\
1/2 & 1/2 & 0
\end{bmatrix} \times \begin{bmatrix}
1/3 \\
1/3 \\
1/3
\end{bmatrix} + (1-0.8)e/3
\]

Link Analysis
Examples
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

\[
\begin{align*}
M &= \begin{bmatrix}
0 & 1 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0
\end{bmatrix} \\
\beta M = \begin{bmatrix}
0 & 0.8 & 0.8 & 0.8 & 0 \\
0 & 0 & 0.8 & 0.8 & 0 \\
0 & 0 & 0 & 0.8 & 0.8 \\
0 & 0 & 0 & 0 & 0.8 \\
0 & 0 & 0 & 0 & 0
\end{bmatrix} \\
\frac{1}{5} \left( \beta M + \left(1 - \beta \right) \frac{1}{5} \right) &= \frac{1}{5} \begin{bmatrix}
0.8 & 0.8 & 0.8 & 0.8 & 0.2 \\
0 & 0 & 0.8 & 0.8 & 0.2 \\
0 & 0 & 0 & 0.8 & 0.8 \\
0 & 0 & 0 & 0 & 0.8 \\
0 & 0 & 0 & 0 & 0
\end{bmatrix}
\end{align*}
\]

Example 2
- Determine the PageRank of each page assuming \(\beta=0.8\)
- Is there a dead end?
  - Yes. E
- Is there a spider trap?
  - No. Having clique does not mean that there is 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

A → B
A

- What is \( v_0 \) and \( M \)?

\( v_0 = [1], M = [1], \) PageRank of \( A = 1 \)
PageRank of \( B = \frac{1}{2} \times 1 \)
PageRank of \( C = \) PageRank of \( B = \frac{1}{2} \)
PageRank of \( D = \) PageRank of \( C = \frac{1}{2} \)

... 

A
B
C
D
Z

Searching pages

- Each search engine has a secret formula that decides the order in which to show pages to the user in response to a search query consisting of one or more search terms

- Google uses more than 250 different properties of pages

Generating the final lists

- Selecting candidate pages
  - A page has to have at least one of the search terms in the query
  - Applying weight
  - Presence or absence of search terms in prominent places
    - e.g. headers or the links to the page itself
- Among the qualified pages, a score is computed for each
  - PageRank score

Link Analysis

Using PageRank in a search engine

Efficient Computation of PageRank
Problems in performing PageRank

- To compute the PageRank for a Web graph
  - We should perform a matrix-vector multiplication of the order of 50 times
  - Until the vector is close to unchanged at one iteration

- The transition matrix of the Web $M$ is very sparse
  - Representing it by all its elements is highly inefficient
  - We want to represent the matrix by only its nonzero elements

- We want to reduce the amount of data that must be passed from the Map tasks to Reduce tasks

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

Representing Transition Matrices (2/2)

- For the Web graph
  - The value will be 1 divided by the out-degree of the page

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

<table>
<thead>
<tr>
<th>Source</th>
<th>Degree</th>
<th>Destinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3</td>
<td>B, C, D</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>A, D</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>B, C</td>
</tr>
</tbody>
</table>

Mapper generates $(\text{key, value}) = (\text{destinations, 1/degree})$

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

- From $(A, B, 1/2)$
- From $(B, 1/3)$

PageRank Iteration Using MapReduce

- One iteration of the PageRank algorithm involves,
  \[ v' = \beta M v + (1 - \beta) e / n \]

  - First round of MapReduce
    - Calculate $Mv$ and store the result to $v'$

  - Second round of MapReduce
    - For each component, multiply $\beta$ and add $(1 - \beta) / n$

PageRank Iteration Using MapReduce

- If $n$ is small enough then 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 view of 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
  - $\beta y/m + (1-\beta)/n$
  - 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 $(1-\beta)/m$ 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
    - $\beta y/m + (1-\beta)/n$
  3. $(1-\beta)/m$, the share of the fraction $(1-\beta)/m$ of the PageRank that belongs to $t$
    - This amount is negligible

Analysis of a Spam Farm (5/6)

- From (1) and (2),
  $$ y = x + \beta y/m + (1-\beta)/n = x + \beta y/m + \beta (1-\beta)/n $$
  $$ y = x + \beta y/m + \beta (1-\beta)/n $$
  $$ y = \frac{x}{1-\beta} + \frac{\beta m}{n} $$
  Where
  $$ c = \beta(1-\beta) / (1-\beta^2) = \beta / (1 + \beta) $$

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