FAQs

- Deadline of PA1 has been extended
  - Feb. 22, 5:00PM via Canvas
  - Individual submission (No team submission)
  - Late submission is still available until Feb. 25, 5:00PM via Canvas

Programming Assignment 2

[1/4]

- Document Summarization using TF/IDF Scores and
  - Due: March 22nd 5:00PM
  - Via Canvas
- We have a collection of M documents
- Term Frequency
  - Augmented TF to prevent a bias towards longer documents
    - \( TF_{ij} = 0.5 + 0.5 \left( \frac{f_{ij}}{\max_k f_{kj}} \right) \)
  - The most frequent term in the document will have an augmented TF value of 1.

[2/4]

- Term \( i \) appears in \( n_i \) articles within the corpus
- Inverted Document Frequency
  - \( IDF_i = \log_{10}(N/n_i) \)
    - where, \( N \) is the total number of articles
- TF.IDF
  - \( TF_{ij} \times IDF_i \)

[3/4]

- How to score a sentence
  - \( Sentence.TF.IDF(S_k) = \text{Sum of top } n \text{ TF.IDF values} \)
  - Use 5 for PA2
- Select top 3 sentences and order them based on the original order

[4/4]

- You should calculate the TF, IDF, and TF.IDF values for all terms for all sub-collections in your corpus. You are required to use MapReduce(s) for this step. Custom implementations without using MapReduce is disallowed.
- Create the summaries of articles (Use 1G data files).
- You should store the results in a HDFS file.
- For a given article (GTA will provide an article for the demo), your software should be able to generate a summary using values generated in (1). You do not need to re-calculate IDF for this step. You are required to use MapReduce for this step. Again, custom implementations that do not use MapReduce is disallowed.
Topics

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

This material is built based on,

  - Chapter 5

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

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

<table>
<thead>
<tr>
<th>Source (PR)</th>
<th>Degree</th>
<th>Destinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>A (l)</td>
<td>3</td>
<td>B, C, D</td>
</tr>
<tr>
<td>B (m)</td>
<td>2</td>
<td>A, D</td>
</tr>
<tr>
<td>C (n)</td>
<td>1</td>
<td>C</td>
</tr>
<tr>
<td>D (o)</td>
<td>2</td>
<td>B, C</td>
</tr>
</tbody>
</table>

Mapper generates

$(key, value) = (destinations, current PR/degree)$

E.g. for the source A, $(B, 1/3), (C, 1/3), (D, 1/3)$

For the source B, $(A, m/2), (D, m/2)$

For the source C, $(C, n)$

For the source D, $(B, o/2), (C, o/2)$

Reducer calculates

Add values

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

<table>
<thead>
<tr>
<th>Source (PR)</th>
<th>Degree</th>
<th>Destinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>A (l)</td>
<td>3</td>
<td>B, C, D</td>
</tr>
<tr>
<td>B (m)</td>
<td>2</td>
<td>A, D</td>
</tr>
<tr>
<td>C (n)</td>
<td>1</td>
<td>C</td>
</tr>
<tr>
<td>D (o)</td>
<td>2</td>
<td>B, C</td>
</tr>
</tbody>
</table>

PageRank Iteration Using MapReduce

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

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

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

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

- 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

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
  - 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 \( m \) supporting pages
  - Accumulate the portion of the PageRank that is distributed equally to all pages
  - The fraction \( 1 / m \) 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 $1 - \beta$ 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
     $\beta m y/m + (1 - \beta)/n$
  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),
  
  $y = x \cdot (\beta y/m + (1 - \beta)/n) = x \cdot \beta y + m \cdot (1 - \beta)/n$
  
  $y = x \cdot (1 - \beta)/m + m \cdot (1 - \beta)/n$
  
  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
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>56/210</td>
<td>0.229</td>
</tr>
<tr>
<td>B</td>
<td>2/9</td>
<td>56/210</td>
<td>-0.364</td>
</tr>
<tr>
<td>C</td>
<td>2/9</td>
<td>56/210</td>
<td>0.186</td>
</tr>
<tr>
<td>D</td>
<td>2/9</td>
<td>56/210</td>
<td>-0.364</td>
</tr>
</tbody>
</table>

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

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, .gov

Calculating TrustRank (1/2)

- Then the topic-sensitive PageRank for $S$ is the limit of the iteration,

\[ v' = \beta Mv + (1 - \beta) e_S / |S| \]

- $M$ 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 (trustworthy pages) $S=\{B,D\}$

<table>
<thead>
<tr>
<th>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>0.186</td>
</tr>
<tr>
<td>D</td>
<td>2/9</td>
<td>59/210</td>
<td>-0.264</td>
</tr>
</tbody>
</table>

Spam Mass

- Measures the fraction of its PageRank that comes from spam for each page

- For an arbitrary page $p$,
  - PageRank $r$
  - TrustRank $t$
  - Computing the TrustRank based on some teleport set of trustworthy pages
  - The spam mass $\delta^p = (r - t)/r$

A negative or small positive spam mass
- $p$ is probably not a spam page
- Page with high spam mass score
- Should be eliminated

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?