Web Hyperlink Analysis Algorithms

The peach and the plum do not speak, yet a path is worn beneath them.

—— 《Records of the Historian》

Yiqun Liu
Department of Computer Science and Technology
Tsinghua University
2011/3/22
Web Hyperlink Analysis

• Major functions of hyperlink analysis
  – Page quality estimation with hyperlink analysis
  – Page content extension with hyperlink analysis
Outlines

• Web hyperlink graph structures
  – Scale, connectivity, topology
• HITS algorithm
• PageRank algorithm
Web as a directed graph
Web hyperlink graph structures

- Vertex set of Web graph
  - Quantity estimation: difficult
    - very high increase speed
    - dynamic contents/links: SERP, Ajax pages

![Graph showing the increase in the number of websites and pages from 2006 to 2010. The x-axis represents the years (2006.12, 2007.12, 2008.12, 2009.12, 2010.12), and the y-axis represents the number of websites in millions. The graph shows a significant increase in the number of websites and pages from 2006 to 2008, followed by a decrease in 2009 and 2010. The percentage increase and decrease over each year are also indicated, with the highest increase in 2008 (91.4%) and the highest decrease in 2010 (-41.0%).]
Web hyperlink graph structures

• Edge set of Web graph
  – Quantity estimation: even more difficult
    • Automatically generated hyperlinks: advertising links, search result links, navigational links, ...
  – Estimation with popular Web corpuses
    • AltaVista: 1999, 203M pages, 1.466B hyperlinks
    • ClueWeb: 2009, 1.041B pages, 7.944B hyperlinks
    • SogouT: 2008, 139M pages, 3.340B hyperlinks
  – #(edge set) is about ten times the number of #(vertex set)
Attributes of Web graph

• Connectivity of Web graph
  – Basic concepts
    • Strongly Connected Component
    • Weakly Connected Component
Attributes of Web graph

- Size distribution of SCC/WCC

\[ \log(\text{Number}) = -2.54 \cdot \log(\text{Size}) + C \]
Attributes of Web graph

• Power law distribution
  – Distribution of SCC/WCC sizes in Web graph
    \[ \text{Number} = C' \cdot \text{Size}^{-2.54} \]
  – Zipf law in text corpuses
    \[ P_i \cdot i = C'' \quad P_i = C'' \cdot i^{-1} \]
  – A wide variety of natural and man-made phenomena follow a power law:
    • frequencies of family names, sizes of craters on the moon and of solar flares, the sizes of power outages, earthquakes, and wars
Attributes of Web graph

• Topology of Web graph
  – A bow-tie structure
  – Core: the largest SCC
Attributes of Web graph

- In-degree distribution of Web graph
  - Indegree: number of in-links for a Web page
  - Popularity
  - Power exponent 2.09
  - Pages with large in-degree (most popular pages) are quite rare
Attributes of Web graph

• Out-degree distribution of Web graph
  – Out-degree: number of out-links on Web page (not including self hyperlinks)
  – power exponent 2.72
  – Not exactly follow power law distribution

• Pages without hyperlinks are rare
Attributes of Web graph

• Semantic meanings of hyperlinks

- recommendation

- topic relevance

• Possible problems
  – Navigational links
  – Advertising links
  – Registration links
Hyperlink analysis algorithms

- Hyperlink analysis
  - Purpose: quality estimation based on hyperlink structure
  - Input: Web graph
  - Output: page importance
  - Algorithms:
    - PageRank, Inverse-PageRank
    - TrustRank, Anti-TrustRank, DiffusionRank
    - BrowseRank
    - HITS
Outlines

• Web hyperlink graph structures
• HITS algorithm
• PageRank algorithm
HITS: design and implementation

• About HITS algorithm
  – Hyperlink-Induced Topic Search
  – Jon Kleinberg (1971-)
  – Cornell University
  – MacArthur fellowship (genius award)
  – Member of the National Academy of Engineering and the American Academy of Arts and Sciences
HITS: design and implementation

• Basic ideas
  – IBM clever system
  – Page quality can be estimated with the following two aspects
    • Authority score
    • Hub score
HITS: design and implementation

• Basic ideas (cont.)
  – Examples:
    • Which is a “good” paper
    • Which is an “important” city
  – Authority cannot be estimated among topics
  – HITS: hyperlink analysis for certain topics
  – Link structure in relevant document set
    • Pro: easy to combine with content retrieval results
    • Con: on-line computing
HITS: design and implementation

- Algorithm

1. Text Retrieval for query \( Q \)
2. For retrieval result set \( R \), construct \( G \) which is composed of \( R \), pages linked by \( R \) and pages linking to \( R \).
3. For each node \( n \) in \( G \), initial \( A(n) = 1 \) and \( H(n) = 1 \)
4. Iteration

\[
A^{(k)}(n) = \sum_{m_i \Rightarrow n} H^{(k-1)}(m_i)
\]

\[
H^{(k)}(n) = \sum_{n \Rightarrow m_i} A^{(k)}(m_i)
\]
HITS: design and implementation

• Iteration Process

- End of iteration: the result vector converges
HITS: design and implementation

• Iteration process (cont.)
  – Adjacent Matrix \( M \)
    \[
    m_{i,j} = \begin{cases} 
    1 & \exists (i, j) \in \text{Graph} \\
    0 & \text{otherwise}
    \end{cases}
    \]
  – \( A(n) \) and \( H(n) \)
    \[
    A^{(k)}(n) = \sum_{m_i \rightarrow n} H^{(k-1)}(m_i) \Rightarrow \quad \tilde{A}^{(k)} = M^T \cdot \tilde{H}^{(k-1)}
    \]
    \[
    H^{(k)}(n) = \sum_{n \rightarrow m_i} A^{(k)}(m_i) \Rightarrow \quad \tilde{H}^{(k)} = M \cdot \tilde{A}^{(k)}
    \]
HITS: design and implementation

• Iteration process (cont.)

\[ A^{(k)}(n) = \sum_{m_i \Rightarrow n} H^{(k-1)}(m_i) \Rightarrow \tilde{A}^{(k)} = M^T \cdot \tilde{H}^{(k-1)} \]

\[ H^{(k)}(n) = \sum_{n \Rightarrow m_i} A^{(k)}(m_i) \Rightarrow \tilde{H}^{(k)} = M \cdot \tilde{A}^{(k)} \]

\[ \tilde{A}^{(k)} = M^T \cdot \tilde{H}^{(k-1)} = M^T \cdot M \cdot \tilde{A}^{(k-1)} = M^T \cdot M \cdot M^T \cdot \tilde{H}^{(k-2)} = (M^T \cdot M)^2 \cdot \tilde{A}^{(k-2)} = \ldots = (M^T \cdot M)^{k-1} \tilde{A}^{(1)} = (M^T \cdot M)^{k-1} M^T \cdot \tilde{H}^{(0)} = (M^T \cdot M)^{k-1} M^T z \]

\[ \tilde{H}^{(k)} = M \cdot \tilde{A}^{(k)} = M \cdot M^T \cdot \tilde{H}^{(k-1)} = M \cdot M^T \cdot M \cdot \tilde{A}^{(k-1)} = (M \cdot M^T)^2 \cdot \tilde{H}^{(k-2)} = \ldots = (M \cdot M^T)^{k-1} \tilde{H}^{(1)} = (M \cdot M^T)^{k-1} M \cdot \tilde{A}^{(1)} = (M \cdot M^T)^{k} \tilde{H}^{(0)} = (M \cdot M^T)^{k} z \]
HITS: design and implementation

• Problems with HITS
  – Noises introduced by text retrieval process
    • some documents are not relevant at all
    • Topic drifting (caused by partially relevant documents)
  – Efficiency problem
    • On-line computation
    • Connectivity server
  – Not widely-adopted by search engines
    • Other applications: SNS
Outlines

• Web hyperlink graph structures
• HITS algorithm
• PageRank algorithm
PageRank: design and implementation

• About PageRank
  – Named after Larry Page

  Internet + = Google

  – Evaluating the importance of Web pages
PageRank: design and implementation

• Basic ideas
  – Hyperlink as voting
  – Pages with many votes are high-quality ones
  – In-degree: each vote is treated the same
  – PageRank: high quality pages’ votes are more important

• How to define “quality”
  – HITS: authority value, hub value
  – PageRank: probability of visiting
PageRank: design and implementation

- How to estimate the probability of visiting
  - By analyzing Web access logs
    - Estimating the probability of visiting
    - Estimating stay time
  - A random walk model
    - Simulation of Web navigation
    - Randomness in
      - Where to start
      - Which hyperlink to choose

How to collect?
Privacy issues
• Random Explorer Ver. 1.0
  – No address bar, no back, no forward. only “Surprise Me”
  • Randomly leading to a page

• A trained monkey
  – click “Surprise Me”
    • possibility = $\alpha$
  – random click hyperlinks
    • possibility = $1 - \alpha$
PageRank: design and implementation

- Random walk model
  - In the process of the trained monkey using “Random Explorer 1.0” to surf the Web, what is the probability of page A being visited?
  - Source 1: leading to A by “Surprise Me”
    - \( P_1(A) = \frac{1}{N} \) (\( N \) is the total number of Web pages)
  - Source 2: leading to A by hyperlinks
    - \( P_2(A) = P(P_1 \rightarrow A) + P(P_2 \rightarrow A) + \ldots + P(P_k \rightarrow A) \) (\( P_1, P_2, \ldots, P_k \) are the pages which connect to A)
    - \( P(P_i \rightarrow A) = \frac{P(P_i)}{\text{outdegree}(P_i)} \)
PageRank: design and implementation

- The probability of visiting A under random walk model is the PageRank of A

\[
\text{PageRank}(A) = \frac{1}{N} \sum_{P_i \Rightarrow A} \frac{\text{PageRank}(P_i)}{\text{Outdegree}(P_i)} + \alpha \cdot \frac{1}{N} + (1 - \alpha) \cdot \sum_{P_i \Rightarrow A} \frac{\text{PageRank}(P_i)}{\text{Outdegree}(P_i)}
\]
PageRank: design and implementation

- PageRank (a simplification algorithm)
  1. For Web graph G, the vertex size of G is N
  2. For each node \( n \) in \( G \), its initial \( PR^{(0)}(n) = \frac{1}{N} \)
  3. For \( k = 1, 2, 3, \ldots, TN \), for each node \( n \) in \( G \):
     \[
     PR^{(k)}(n) = \alpha \cdot \frac{1}{N} + (1 - \alpha) \cdot \sum_{P_{i} \rightarrow n} \frac{PR^{(k-1)}(P_{i})}{Outdegree(P_{i})}
     \]
  4. Output the results.

PageRank from pages which connects to \( n \)
PageRank: design and implementation

- Example
  - Initial: 0.25, 0.25, 0.25, 0.25
  - $\alpha = 0.2$

\[
PR^{(1)}(A) = 0.2 \cdot \frac{1}{4} + (1 - 0.2) \cdot \left( PR^{(0)}(D) \big/ \text{Outdegree}(D) \right) = 0.05 + 0.8 \cdot \frac{1}{4} = 0.25
\]

\[
PR^{(1)}(B) = 0.2 \cdot \frac{1}{4} + (1 - 0.2) \cdot \left( PR^{(0)}(A) \big/ \text{Outdegree}(A) \right) = 0.05 + 0.8 \cdot \frac{1}{4} = 0.15
\]

\[
PR^{(1)}(C) = 0.2 \cdot \frac{1}{4} + (1 - 0.2) \cdot \left( PR^{(0)}(A) \big/ \text{Outdegree}(A) \right) = 0.05 + 0.8 \cdot \frac{1}{4} = 0.15
\]

\[
PR^{(1)}(D) = 0.2 \cdot \frac{1}{4} + (1 - 0.2) \cdot \left( PR^{(0)}(B) \big/ \text{Outdegree}(B) + PR^{(0)}(C) \big/ \text{Outdegree}(C) \right)
\]
\[= 0.05 + 0.8 \cdot \left( \frac{1}{4} + \frac{1}{4} \right) = 0.45\]
## PageRank: design and implementation

<table>
<thead>
<tr>
<th>Iteration</th>
<th>PR(A)</th>
<th>PR(B)</th>
<th>PR(C)</th>
<th>PR(D)</th>
<th>SUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>0.2500</td>
<td>0.2500</td>
<td>0.2500</td>
<td>0.2500</td>
<td>1.0000</td>
</tr>
<tr>
<td>#2</td>
<td>0.2500</td>
<td>0.1500</td>
<td>0.1500</td>
<td>0.4500</td>
<td>1.0000</td>
</tr>
<tr>
<td>#3</td>
<td>0.4100</td>
<td>0.1500</td>
<td>0.1500</td>
<td>0.2900</td>
<td>1.0000</td>
</tr>
<tr>
<td>#4</td>
<td>0.2820</td>
<td>0.2140</td>
<td>0.2140</td>
<td>0.2900</td>
<td>1.0000</td>
</tr>
<tr>
<td>#5</td>
<td>0.2820</td>
<td>0.1628</td>
<td>0.1628</td>
<td>0.3924</td>
<td>1.0000</td>
</tr>
<tr>
<td>#20</td>
<td>0.3144</td>
<td>0.1758</td>
<td>0.1758</td>
<td>0.3341</td>
<td>1.0000</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#30</td>
<td>0.3158</td>
<td>0.1762</td>
<td>0.1762</td>
<td>0.3319</td>
<td>1.0000</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#99</td>
<td>0.3156</td>
<td>0.1762</td>
<td>0.1762</td>
<td>0.3320</td>
<td>1.0000</td>
</tr>
</tbody>
</table>
PageRank: design and implementation

• Problems with the simplified PageRank
  – Dead end pages
    • Pages without out-links
    • TXT, DOC, JPG, …
  – Surprise Me is the only option
    • A virtual out-link is added from the dead end page to each pages in $G$
    • The PageRank of the dead end page is equally divided by all pages in $G$
  – How to improve?

Surprise Me!
• The standard PageRank algorithm

1. For Web graph $G$, the vertex size of $G$ is $N$

2. For each node $n$ in $G$, its initial PageRank $PR^{(0)}(n) = \frac{1}{N}$ and its temporal variable $I(n) = \frac{\alpha}{N}$

3. For $k = 1, 2, 3, \ldots, TN$, for each node $n$ in $G$ if $Outdegree(n) > 0$, we have:

$$I(P_i) = I(P_i) + (1 - \alpha) \cdot \frac{PR^{(k-1)}(n)}{Outdegree(n)}$$

\[ \forall P_i, \text{ if } n \Rightarrow P_i, \]

Different from the simplified algorithm
• The standard PageRank algorithm (cont.)
  
  – If $\text{Outdegree}(n) = 0$, we have

\[
\forall P_i \in G, \quad I(P_i) = I(P_i) + (1 - \alpha) \cdot \frac{PR^{(k-1)}(n)}{N}
\]

Different from the simplified algorithm

– $PR$: $PR^{(k)} = I$

– $I = (\frac{\alpha}{N}, \frac{\alpha}{N}, \frac{\alpha}{N}, \ldots, \frac{\alpha}{N})$

4. Output results
PageRank: design and implementation

• The standard PageRank algorithm (cont.)
  – Standard PageRank v.s. Simplified PageRank
    • A virtual out-link is added from the dead end page to each pages in $G$
  – The structure of Web graph
    • original adjacent matrix $M$
    • improved adjacent matrix $A$

\[
m_{i,j} = \begin{cases} 
1 & \exists (i, j) \in \text{Graph} \\
0 & \text{otherwise}
\end{cases}
\]

\[
a_{i,j} = \begin{cases} 
\frac{1}{\sum_j m_{i,j}} & \exists (i, j) \in G \\
\frac{1}{n} & \sum_j m_{i,j} = 0 \\
0 & \text{otherwise}
\end{cases}
\]
The standard PageRank algorithm (cont.)

- If $I = (1,1,\ldots,1)$, then the algorithm can be rewritten as:

$$ PR^{(k)} = \alpha \cdot \frac{1}{N} \cdot I + (1 - \alpha) \cdot A^T PR^{(k-1)} $$

- Problem: $A$ is a sparse matrix

  - size: $N \times N$, $N$ is the number of vertexes in $G$
  - number of non-zero elements: $M$, $M$ is the number of edges in $G$
  - $M < N$ ($M$ is about ten times the number of $N$)
  - $M << N \times N$
PageRank: design and implementation

- PageRank implementation
  - Input: Web graph $G$ (size of vertex set is $N$, including all hyperlinks), parameter $\alpha$, maximum number of iteration $M$;
  - How to record $G$
    - only non-zero elements in $A$ is recorded
      - Page A -> Page B
      - Page A -> Page C
      - ...

- $I$: a temporal record of PageRank for each node
- $S$: a temporal record of PageRank of all dead ends
PageRank: design and implementation

- PageRank implementation (cont.)

1. For each record $E(i, j)$ in document $D$,

$$\text{Outdegree}(i) = \text{Outdegree}(i) + 1$$

2. For $n = 1, 2, 3, \ldots , N$

$$PR^{(0)}(n) = \frac{1}{N}, \quad I(n) = \frac{\alpha}{N} \quad S^{(1)} = S^{(1)} + PR^{(0)}(n)$$

1. For $k = 1, 2, 3, \ldots , TN$

   a. For each record $E(i, j)$ in document $D$,

   $$I(j) = I(j) + (1 - \alpha) \cdot \frac{PR^{(k-1)}(i)}{\text{Outdegree}(i)}$$

   a. For $n = 1, 2, 3, \ldots , N$

   $$PR^{(k)}(n) = I(n) + (1 - \alpha) \cdot \frac{S^{(k)}}{N} \quad I(n) = \frac{\alpha}{N} \quad S^{(k+1)} = S^{(k+1)} + PR^{(k)}(n)$$
PageRank: design and implementation

- PageRank implementation (cont.)
  - Number of visiting $D$
    
    $$(TN+1)(N+L)$$

  - Storage: result $PR$ (size = $N$), temporal variable $I$ (size = $N$); temporal variable $S$ (size = $TN$)
  - Parameter: $\alpha=0.15$, $TN=20\sim30$
PageRank: design and implementation

- Limitation of PageRank algorithms
  - PageRank doesn’t work well in retrieval experiments
    - Upstill: introducing of PageRank slightly improves navigational performance
    - Amento: PageRank/HITS doesn’t perform well on medium scale datasets.
    - Our experiments: PageRank is not so useful on TREC benchmark
  - Possible reasons: data size, data quality
Thank you!

Questions or comments?