Learning to Rank on Large-scale Graphs with Rich Metadata

Tie-Yan Liu
Microsoft Research Asia
Outline

• Graph Ranking
• PageRank: graph structure
• BrowseRank: + rich metadata
• Semi-supervised PageRank: + supervision
• Summary
Graph Ranking
Graph Ranking

• Problem Definition
  – Given a graph $G = \{V, E\}$, where $v_i \in V \ (i = 1, \ldots, N)$ represents the i-th node and $e_{i,j} \in E \ (i, j = 1, \ldots, N)$ represents the edge between the i-th and the j-th node,
  – Rank the nodes according to a certain criterion, such as popularity and important.

• Wide Applications
  – Web page ranking, entity ranking in social network, expert finding, ...
Example: Ranking on Web Graph

- Web Graph
  - Web pages all over the world are connected with each other through hyperlinks.
  - The innovation of hypertext changes the world!
Example: Ranking on Web Graph

- A scale-free network
  - Preferential attachment
    - Pages tend to link to important pages
    - Links usually mean recommendation or endorsement
PageRank
The PageRank Algorithm

• PageRank of a web page is proportional to the PageRank of its parents, but inversely proportional to their out-degrees.

\[ R(u) = d + (1 - d) \sum_{v \in B_u} \frac{R(v)}{N_v} \]

• Well motivated by preferential attachment.
A Markov Chain Interpretation

Assume a random surfer walking on the link graph

Use discrete-time Markov chain to simulate the random walk

PageRank = stationary distribution of the Markov chain
Impact of PageRank

• A key technology of Google.
• Although simple, it brings revolution to Web search!
Beyond PageRank

• Beyond graph structure, we usually have other useful information in the graph
  – Metadata on the nodes and edges
  – Supervision on part of the nodes

• Can we leverage such information and improve the accuracy of graph ranking?
Beyond PageRank

• BrowseRank
  – Consider node and edge metadata

• Semi-supervised PageRank
  – Further consider the supervision
Co-work with Yuting Liu, Bin Gao, Shuyuan He, Zhiming Ma, and Hang Li.
Motivation: Problems with PageRank

- Voted by Web content creators but not Web users
- Inappropriate assumptions on Web surfer behavior

### Random Surfer Behavior

<table>
<thead>
<tr>
<th>Behavior</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choosing next page from outlinks in a uniformly random manner.</td>
<td>Easy to be spammed</td>
</tr>
<tr>
<td>Randomly resetting to any page on the Web with a uniform probability.</td>
<td>Cannot reflect user’s real information needs</td>
</tr>
<tr>
<td>Staying at each page for a unit period of time.</td>
<td></td>
</tr>
</tbody>
</table>
Motivation: Problems with PageRank

- Voted by Web content creators but not Web users
- Inappropriate assumptions on Web surfer behavior

<table>
<thead>
<tr>
<th>Random Surfer Behavior</th>
<th>Real User Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choosing next page from outlinks in a uniformly random manner.</td>
<td>Some hyperlinks are popular, and some are never visited.</td>
</tr>
<tr>
<td>Randomly resetting to any page on the Web with a uniform probability.</td>
<td>Search engine pages, bookmarks, and famous pages have higher reset probabilities</td>
</tr>
<tr>
<td>Staying at each page for a unit period of time.</td>
<td>Spending different periods of time on different pages.</td>
</tr>
</tbody>
</table>
Leveraging User Behavior Data

- Not simply a search shortcut
- Record users’ behavior in IE

<User Hash, URL, Time Stamp, Type, … >

Natural session segmentation
Type = 1: user inputs a URL directly, start of a session
Type = 0: user clicks on an existing hyperlink to get to this URL.
User Browsing Graph

A directed graph with rich meta data.

- **Vertex:** Web page
- **Edge:** Transition
- **Edge weight** $w_{ij}$: Number of transitions
- **Staying time** $T_i$: Time spend on page $i$
- **Vertex weight** $C_i$: Number of visits for page $i$
- **Reset probability** $\sigma_i$: Normalized frequencies as first page of session
User Browsing Graph

• Another scale-free network
  – Real users tend to visit important pages frequently
  – Web masters and web users perform differently, but generate similar complex networks.
BrowseRank

Get user behavior data from Search engine toolbar logs

Rank pages based on both query-page matching and page importance

Build user browsing graph

Model the graph with continuous-time Markov process

Use its stationary distribution as page importance.

Conventional random walk model cannot be used when there is staying time information
Continuous-time Markov Model

User Browsing Graph

```
Calculating $\pi$

$\pi_i = \frac{\tilde{\pi}_i / \lambda_i}{\sum_{j=1}^{N} \tilde{\pi}_j / \lambda_j}$
```

Estimating staying time distribution $t \sim \exp(\lambda)$

```
Q Process
```

```
Computing the stationary distribution $\tilde{\pi}$ of a discrete-time Markov chain (called embedded Markov chain)
```

$\pi = \pi P(t)$
Continuous-time Markov Model

\[ \pi = \pi P(t) \]
In theory, staying time is governed by an exponential distribution

\[ t \sim \exp(\lambda) \]

- In practice, it is NOT!

Estimation with an additive noise model:

\[ Z = t + u \quad (u \sim \chi^2) \]

\[
\min_{\lambda} \left( \frac{1}{\lambda} \right) - \frac{1}{2} \left( \frac{S^2 - 1}{\lambda^2} \right)^2
\]

s.t. \( \lambda > 0 \).

Sample mean of observed staying time

Sample variance of observed staying time
• Estimate transition probability matrix $P$ of EMC.

$$P_{ij} = \alpha \left( \frac{w_{ij}}{C_i} \right) + \left( \frac{C_i - \sum_k w_{ik}}{C_i} \right) \sigma_i + (1 - \alpha) \sigma_i$$

- Number of jumps to $j$ from all visits on $i$
- For the rest of visit on $i$, random jump to other pages according to reset probability
- Global smoothing according to reset probability

• Compute its stationary distribution: $\tilde{\pi} = \tilde{\pi} P$. 
Results: Top-Ranked Sites

<table>
<thead>
<tr>
<th>No.</th>
<th>PageRank</th>
<th>BrowseRank</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>adobe.com</td>
<td>myspace.com</td>
</tr>
<tr>
<td>2</td>
<td>passport.com</td>
<td>msn.com</td>
</tr>
<tr>
<td>3</td>
<td>msn.com</td>
<td>yahoo.com</td>
</tr>
<tr>
<td>4</td>
<td>microsoft.com</td>
<td>youtube.com</td>
</tr>
<tr>
<td>5</td>
<td>yahoo.com</td>
<td>live.com</td>
</tr>
<tr>
<td>6</td>
<td>google.com</td>
<td>facebook.com</td>
</tr>
<tr>
<td>7</td>
<td>mapquest.com</td>
<td>google.com</td>
</tr>
<tr>
<td>8</td>
<td>miibeian.gov.cn</td>
<td>ebay.com</td>
</tr>
<tr>
<td>9</td>
<td>w3.org</td>
<td>hi5.com</td>
</tr>
<tr>
<td>10</td>
<td>godaddy.com</td>
<td>bebo.com</td>
</tr>
</tbody>
</table>

Web 2.0 websites:

Websites are viewed as important if users pay a lot of visits to, spend much time on, and create rich content for them.
Results: Anti-Spam

<table>
<thead>
<tr>
<th>Bucket No.</th>
<th>Number of Websites</th>
<th>PageRank</th>
<th>BrowseRank</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>148</td>
<td>2</td>
<td>&gt; 1</td>
</tr>
<tr>
<td>3</td>
<td>720</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>2231</td>
<td>22</td>
<td>18</td>
</tr>
<tr>
<td>5</td>
<td>5610</td>
<td>30</td>
<td>39</td>
</tr>
<tr>
<td>6</td>
<td>12600</td>
<td>58</td>
<td>88</td>
</tr>
<tr>
<td>7</td>
<td>25620</td>
<td>90</td>
<td>87</td>
</tr>
</tbody>
</table>

Existing spam techniques can hardly spam BrowseRank, and intuitively, BrowseRank is also robust to new spam technologies:

It is more difficult (and costly) to cheat real Web users than to cheat search engines.

53 million sessions
Results: Final Relevance Ranking

BrowseRank can significantly improve final ranking

Combining Parameter $\theta$  
8000 queries
Impact of BrowseRank

• Regarded as a *breakthrough in Web search* after PageRank by much of the Internet media.

• Awarded the SIGIR 2008 Best Student Paper.
Generalizing Staying Time

• Staying time $\rightarrow$ Node utility

• Node utility: average value that the node gives to the surfer in a single visit
  – In this way, the model can incorporate more information.
  – The node utility may depend on previous visits, and thus needs more advanced stochastic models (e.g., Markov skeleton process @ CIKM’09).
Semi-Supervised PageRank

Co-work with Bin Gao, Wei Wei, Taifeng Wang, and Hang Li.
Supervision

• In addition to the metadata on nodes and edges, sometimes we can also obtain supervision
  – User click-through and page views
  – Known high-quality websites
  – Known spam websites
  – Human editorial information on website rating
Challenges

• Can we
  – Make good use of both web graph structure and rich metadata?
  – Effectively incorporate supervision?
  – Avoid over-fitting on small training set?
  – Handle very large scale graphs during the learning process?
Existing Work

• LiftHITS
  – Learning to Create Customized Authority Lists (Huan, David, Andrew, ICML’00)

• Adaptive PageRank
  – Adaptive ranking of Web pages (Tsoi, Morini, Scarselli, Hagenbuchner, and Maggini, WWW’03)

• NetRank

• Do not use node features or edge features.
• Cannot scale-up due to complex computation like matrix inversion, pseudo matrix inversion, and successive matrix multiplications.
Our Proposal

• Define the loss function
  – According to the Markov random walk on the graph
    • Incorporate edge features into the transition probability of the Markov process, and incorporate node features to its reset probability
  – According to the difference between the ranking results given by the Markov model and the supervision
Notations

Edge features: \( X = \{x_{ij}\} \)

Node features: \( Y = \{y_i\} \)

Edge parameter vector: \( \omega \)

Node parameter vector: \( \phi \)

Page importance score: \( \pi \)

Link graph: \( G \)

Supervision matrix: \( B \)

Weight vector for supervisions: \( \mu \)
Optimization Problem

\[
\min_{\omega \geq 0, \phi \geq 0, \pi \geq 0} \alpha \| dP^T(\omega)\pi + (1-d)g(\phi) - \pi \|^2 + \beta \mu^T(e - B\pi)
\]

**Loss term #1**: based on PageRank propagation, combining edge features and node features by \( P(\omega) \) and \( g(\phi) \).

**Loss term #2**: compared with supervised information in pairwise preference fashion.

\[
p_{ij}(\omega) = \begin{cases} 
\frac{\sum_k \omega_k x_{ijk}}{\sum_j \sum_k \omega_k x_{ijk}}, & \text{if there is an edge from } i \text{ to } j \\
0, & \text{otherwise.}
\end{cases}
\]

\[
g_i(\phi) = \phi^T y_i
\]
First-Order Optimization

Denote

\[ G(\omega, \phi, \pi) = \alpha \| dP^T(\omega)\pi + (1 - d)g(\phi) - \pi \|^2 + \beta \mu^T(e - B\pi) \]

Derivatives

\[ \frac{\partial G}{\partial \omega} = 2\alpha d[P^T\pi \otimes \pi - \pi \otimes \pi + (1 - d)g \otimes \pi] \quad \frac{\partial \text{vec}(P)}{\partial \omega^T} \]

\[ \frac{\partial G}{\partial \phi} = 2\alpha(1 - d)[(1 - d)g + dP^T\pi - \pi] \frac{\partial g}{\partial \phi} \]

\[ \frac{\partial G}{\partial \pi} = 2\alpha[(dPP^T - dP - dP^T + I)\pi - (1 - d)(I - dP)g] - \beta B^T\mu \]

Matrix size \( n^2 \times l \)

Matrix size \( n^2 \times 1 \)

Matrix multiplication \( O(n^3) \)
First-Order Optimization: Details

\[ \frac{\partial \text{vec}(P)}{\partial \omega} = \left( \begin{array}{ccc} \frac{\partial p_{11}}{\partial \omega_1} & \ldots & \frac{\partial p_{11}}{\partial \omega_l} \\ \vdots & \ddots & \vdots \\ \frac{\partial p_{n1}}{\partial \omega_1} & \ldots & \frac{\partial p_{n1}}{\partial \omega_l} \end{array} \right), \quad \frac{\partial r}{\partial \phi} = \left( \begin{array}{c} \frac{\partial r}{\partial \phi_1} \\ \vdots \\ \frac{\partial r}{\partial \phi_i} \\ \vdots \\ \frac{\partial r}{\partial \phi_h} \end{array} \right) \]

\( \mathcal{O}(n^3+n^2l) \) seems very difficult to scale up to web scale!
Solve the Problem in Linear Time

Denote

\[ \pi' = \mathbf{P}^T \pi \]

\[ \pi'' = \pi' - \pi \]

\[ \frac{\partial G}{\partial \pi} = 2\alpha [d(\mathbf{P} \pi'' - \pi'') + (1 - d)(\pi - g + dP\mathbf{g})] - \beta B^T \mu \]

\[ \frac{\partial G}{\partial \omega} = 2\alpha d \left\{ [\pi'' + (1 - d)g] \otimes \pi \right\}^T \frac{\partial \text{vec}(\mathbf{P})}{\partial \omega^T} \]

\[ \frac{\partial G}{\partial \phi} = 2\alpha (1 - d) [(1 - d)g + d\pi' - \pi] \frac{\partial g}{\partial \phi} \]

Iteration 4

Kronecker Product of Vectors on a Sparse Graph

Solved with only four iterations of propagation by O(ml+n)
Map-Reduce Logics

• Matrix-Vector Multiplication

\[ \pi'_i = P^T \pi \quad \quad \pi'_i = \sum_j p_{ji} \pi_j \]

- **Map**: map \(<i, j, p_{ji} >\) on \(i\) such that tuples with the same \(i\) are shuffled to the same machine in the form of \(<i, (j, p_{ji}) >\).

- **Reduce**: take \(<i, (j, p_{ji}) >\) and calculate \(<i, \sum_j p_{ji} \pi_j >\) and then emit \(\pi'_i, \pi'_i = \sum_j p_{ji} \pi_j\).

• Kronecker Product of Vectors on a Sparse Graph

\[ z = x \otimes y \]

- **Map**: map \(<i, x_i >\) on \(i\) such that tuples with the same \(i\) are shuffled to the same machine.

- **Reduce**: take \(<i, x_i >\) and calculate \(<i, x_iy_j >\) only if there is an edge from page \(i\) to page \(j\), and then emit \(z_{(i-1)n+j} = x_iy_j\); otherwise, \(z_{(i-1)n+j} = 0\).
Details: Sparse Graph Index

Graph $G$: $A \xrightarrow{1} B \xrightarrow{2} C \xrightarrow{3} D$

Matrix $P$:

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

Sparse Matrix Data Stream:

$[A, \langle B, 1 \rangle], [B, \langle C, 2 \rangle, \langle D, 1 \rangle], [C, \langle A, 1 \rangle, \langle B, 3 \rangle], [D, \langle A, 2 \rangle]$
Details: Matrix-Vector Multiplication

$$Px$$

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

Graph G:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>100</td>
<td>200</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>200</td>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>100</td>
<td>300</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>300</td>
<td>400</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>200</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>100</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Details: Kronecker Product

\[
\begin{array}{cccc}
X_1 & X_2 & X_3 & X_4 \\
1 & 2 & 3 & 4 \\
\end{array} \quad \otimes \quad \begin{array}{cccc}
Y_1 & Y_2 & Y_3 & Y_4 \\
1 & 2 & 3 & 4 \\
\end{array}
\]

\[
\begin{array}{ccccccccc}
X_1 & Y_1 & X_1 & Y_2 & X_1 & Y_3 & X_1 & Y_4 & X_2 & Y_1 & X_2 & Y_2 & X_2 & Y_3 & X_2 & Y_4 & X_3 & Y_1 & X_3 & Y_2 & X_3 & Y_3 & X_3 & Y_4 & X_4 & Y_1 & X_4 & Y_2 & X_4 & Y_3 & X_4 & Y_4 \\
1 & 2 & 3 & 4 & 2 & 4 & 6 & 8 & 3 & 6 & 9 & 12 & 4 & 8 & 12 & 16 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]
**Details: Kronecker Product**

1. **Propagate y along graph G’ (the inverted graph of G)**

   - **A**
   - **B**
   - **C**
   - **D**

   - **C,3**
   - **D,4**
   - **B,2**
   - **A,1**
   - **C,3**

2. **Multiple x with the received y values**

   - **Result** =
     - [AC,15]
     - [AD,20]
     - [BA,6]
     - [BC,18]
     - [CB,14]
     - [DB,16]

   - **x** =
     - 5
     - 6
     - 7
     - 8

   - **y** =
     - 1
     - 2
     - 3
     - 4
Output of the Learning Process

- $\pi$: can be used to direct rank nodes in the given graph.
- $\wp$ and $\omega$ can be used to rank nodes in new graphs with similar generating mechanisms to the given graph (advantages of the parametric formulation).
Results: Anti-Spam

Table 3: Number of spam websites over buckets.

<table>
<thead>
<tr>
<th>No.</th>
<th># of Websites</th>
<th>PageRank</th>
<th>AP</th>
<th>RankNet</th>
<th>SSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>150</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>537</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1257</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>2660</td>
<td>2</td>
<td>8</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>4788</td>
<td>4</td>
<td>7</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>8344</td>
<td>12</td>
<td>7</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>13708</td>
<td>7</td>
<td>16</td>
<td>23</td>
<td>12</td>
</tr>
<tr>
<td>8</td>
<td>20846</td>
<td>13</td>
<td>33</td>
<td>18</td>
<td>33</td>
</tr>
<tr>
<td>9</td>
<td>29008</td>
<td>19</td>
<td>25</td>
<td>34</td>
<td>27</td>
</tr>
<tr>
<td>10</td>
<td>33231</td>
<td>60</td>
<td>25</td>
<td>32</td>
<td>31</td>
</tr>
</tbody>
</table>
Results: Relevance Ranking

- SSP consistently outperforms the other algorithms, with all $\theta$ values, and in terms of all evaluation measures.
Summary
Summary

• Graph ranking is important.
• It is challenging yet important task to leverage rich metadata and supervision to enhance graph ranking.
• Advanced stochastic models, first-order optimization, and large-scale distributed computation can help us define effective and efficient algorithms to perform the task.
Future Work

• Semi-supervised BrowseRank
• Advanced optimization
  – Incremental learning
  – High-order optimization
Thanks!

tyliu@microsoft.com
http://research.microsoft.com/people/tyliu/