LETOR
Learning to Rank
for Information Retrieval

# 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

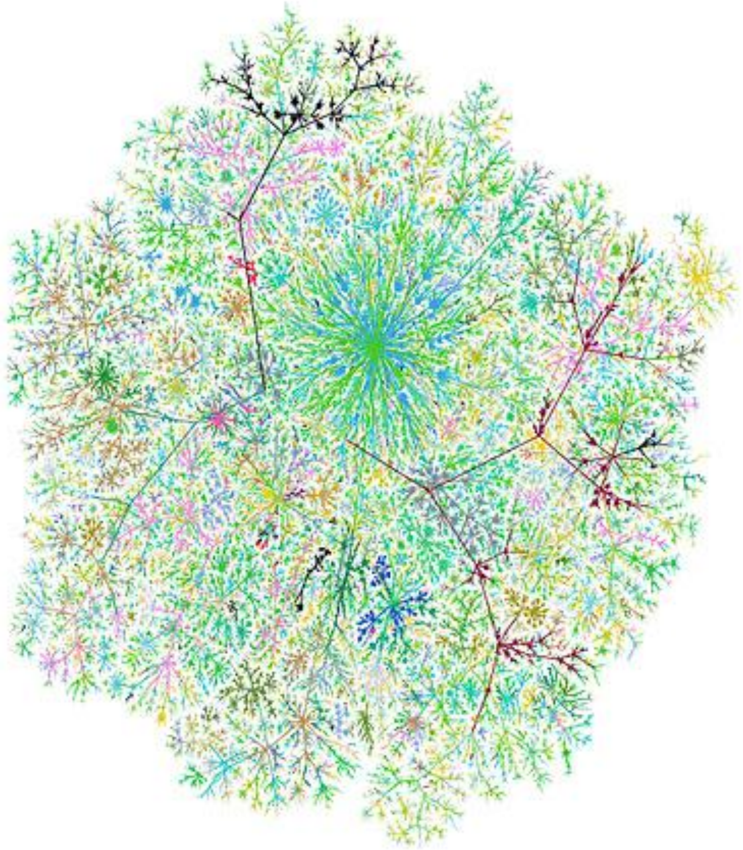
# Graph Ranking

# Graph Ranking

- Problem Definition
  - Given a graph G = {V, E}, where $v_i \epsilon V (i = 1, \dots N)$ represents the i-th node and $e_{i,j} \epsilon E \ (i, j = 1, \dots, N)$ 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, …

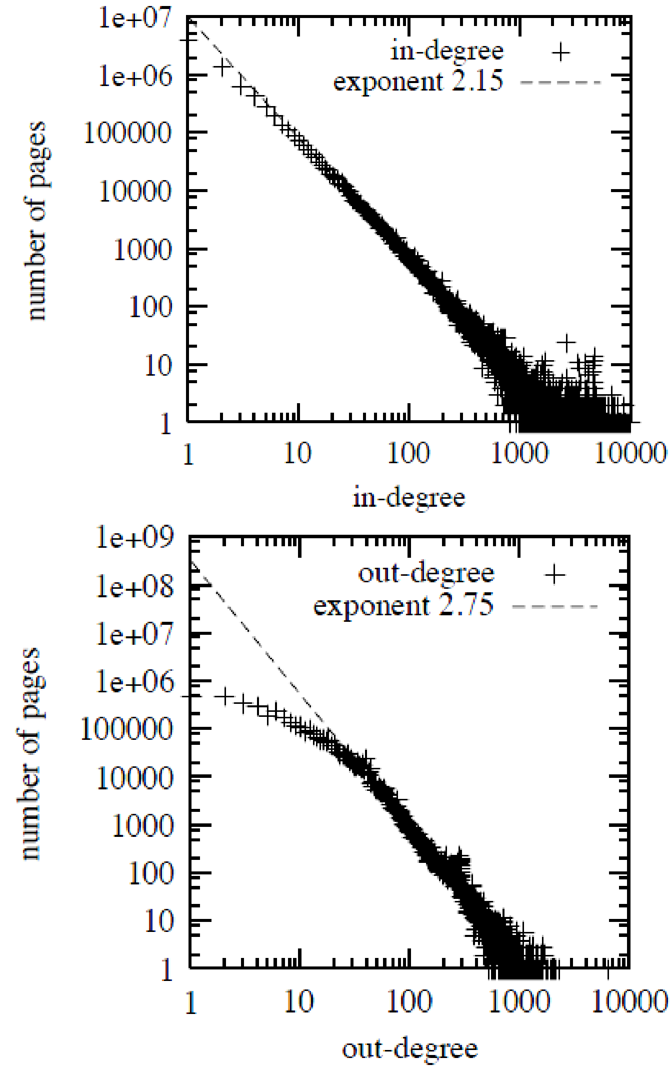
# Example: Ranking on Web Graph

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

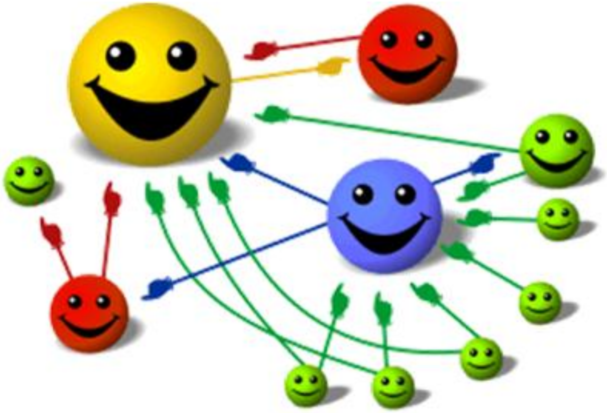


# Example: Ranking on Web Graph

- 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.

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

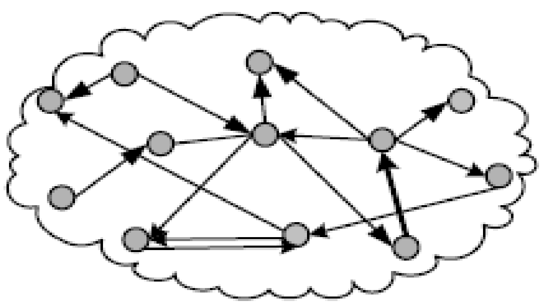
- Well motivated by *preferential attachment*.

# 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



Web link graph

# 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

# BrowseRank


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 |
| --- |
| Choosing next page from outlinks in a uniformly random manner. |
| Randomly resetting to any page on the Web with a uniform probability. |
| Staying at each page for a unit period of time. |

→

Easy to be spammed

Cannot reflect user's real information needs

# Motivation: Problems with PageRank

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

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

# Leveraging User Behavior Data

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

**Search engine toolbar**

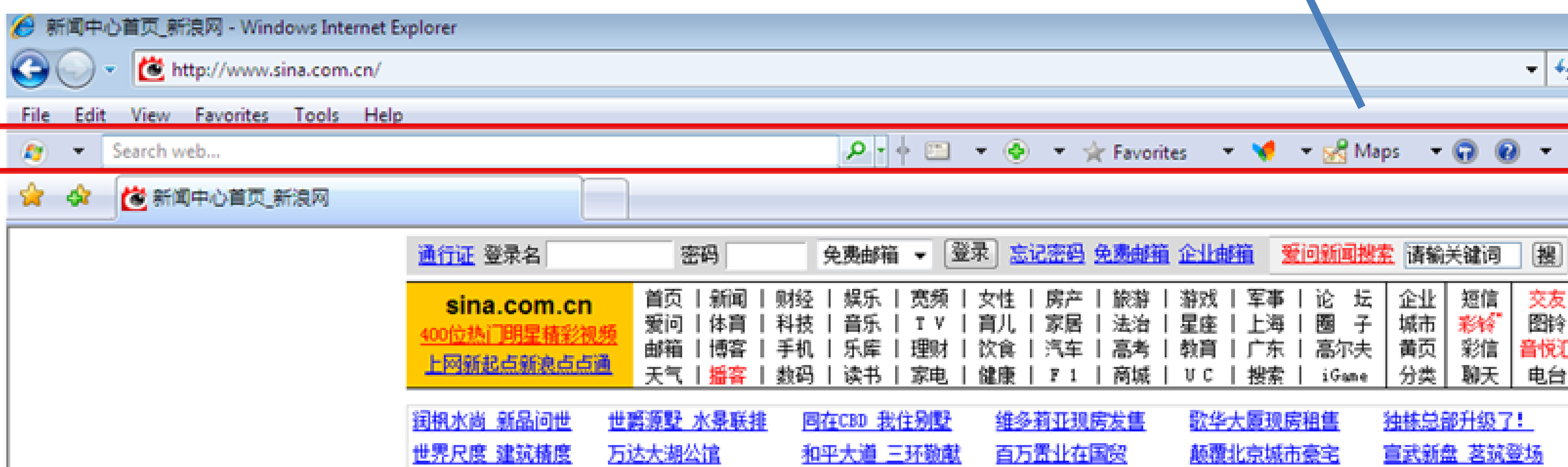
**Toolbar Data**

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

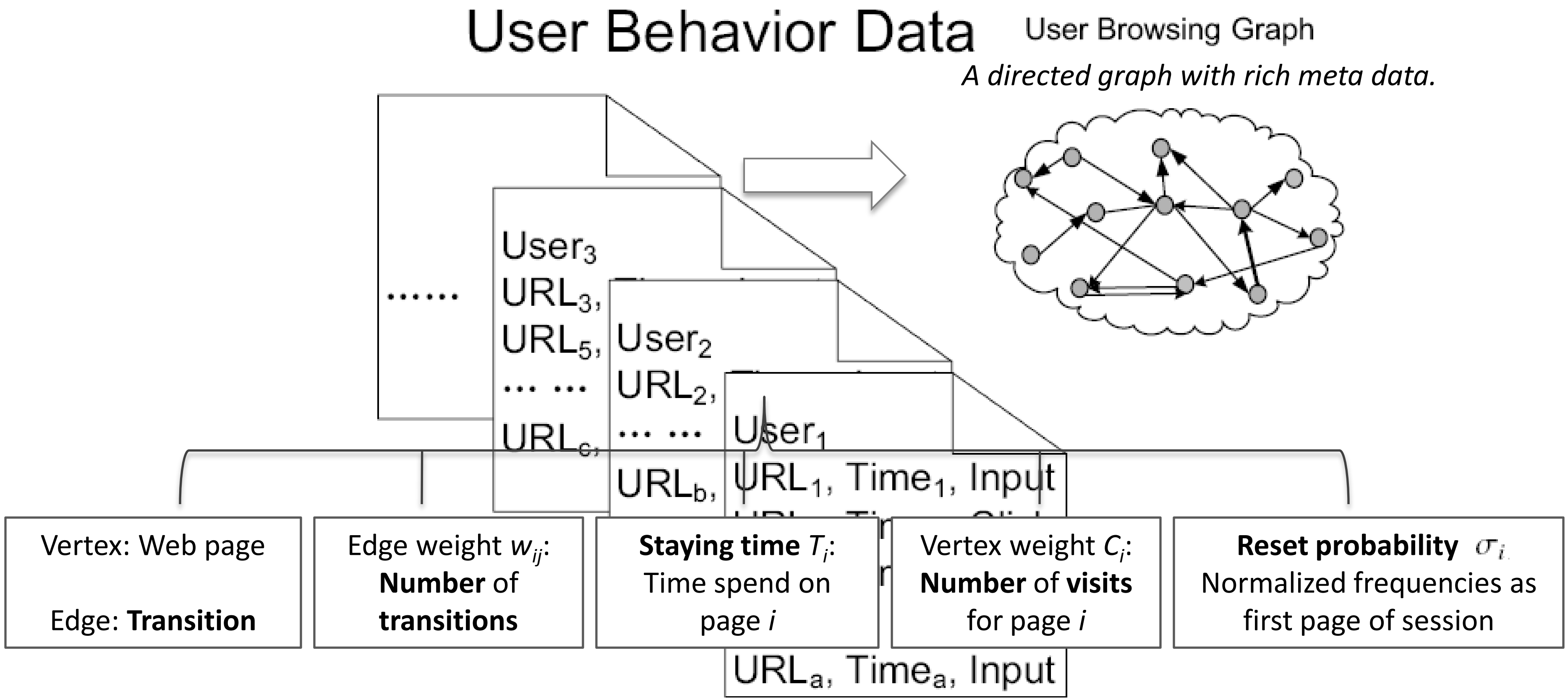
Natural session segmentation

Type = 1: user inputs a URL directly, start of a session

Type = 0: user clicks on an existing hyperlink to get to this URL.

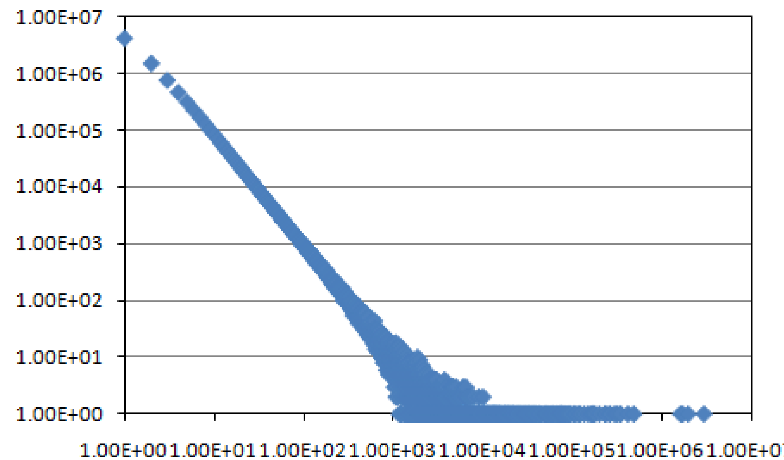


# User Browsing Graph

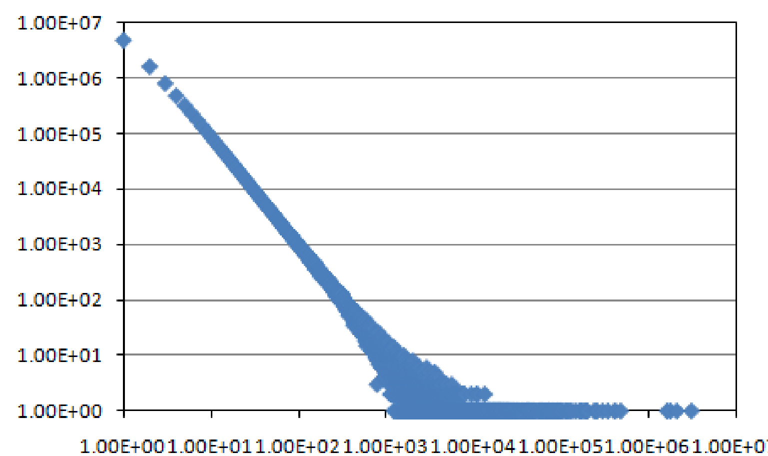


# User Browsing Graph

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

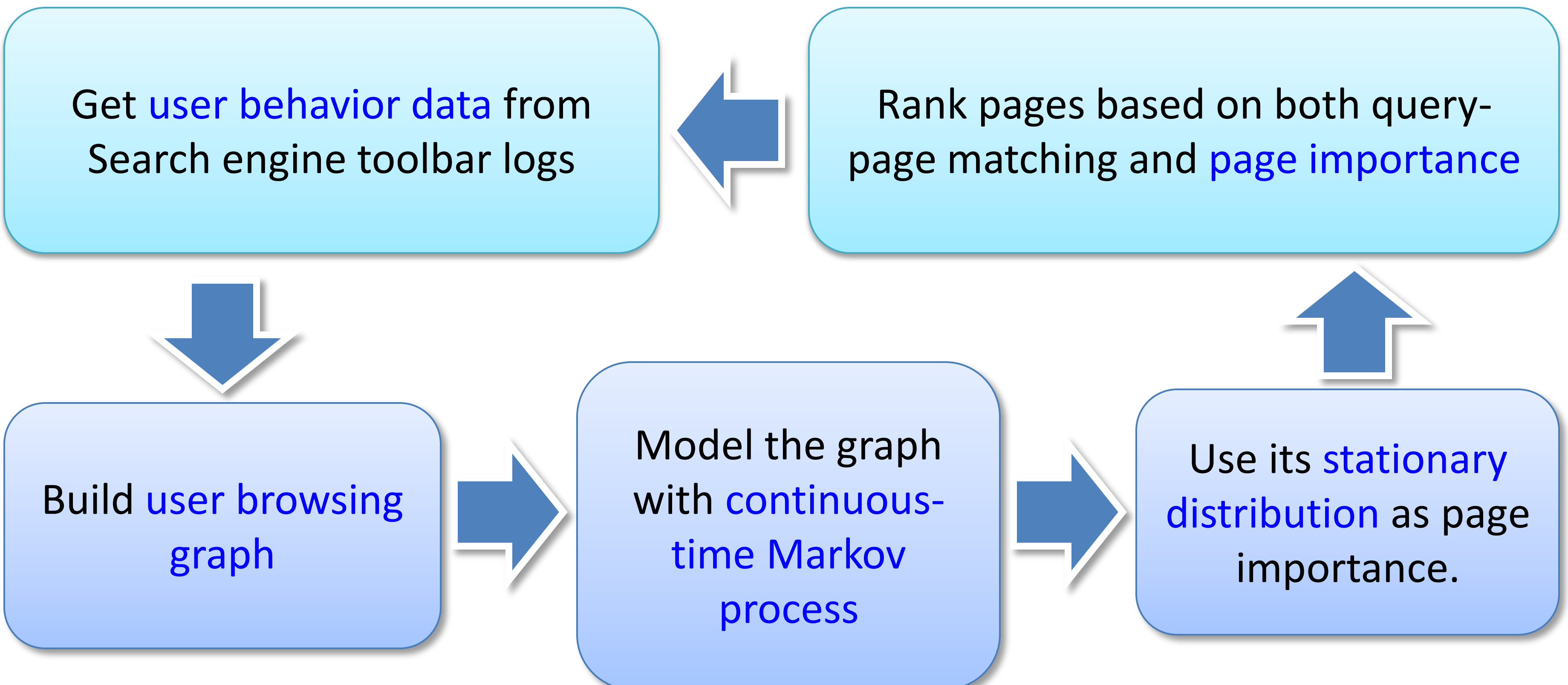



In-degree




Out-degree

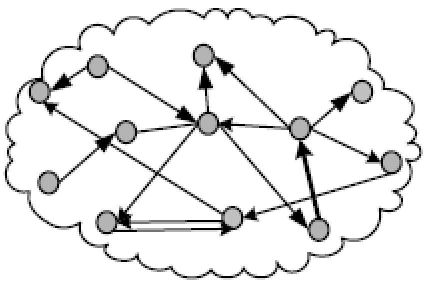
# BrowseRank




*Conventional random walk model cannot be used when there is staying time information*

# Continuous-time Markov Model

User Browsing Graph



Hard

$P(t)$

$\pi = \pi P(t)$

Q Process

Calculating $\pi$

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

=

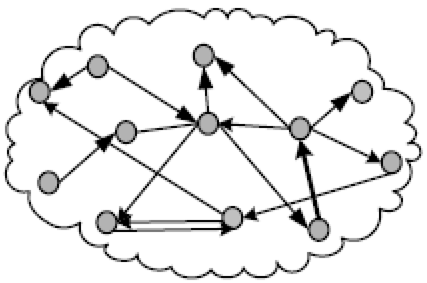
Estimating **staying time** distribution $t \sim \exp(\lambda)$

+

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

# Continuous-time Markov Model

User Browsing Graph



Hard

$P(t)$

$\boxed{\pi = \pi P(t)}$

Q Process

Calculating $\pi$

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

=

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

+

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

Calculating $\pi$

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

=

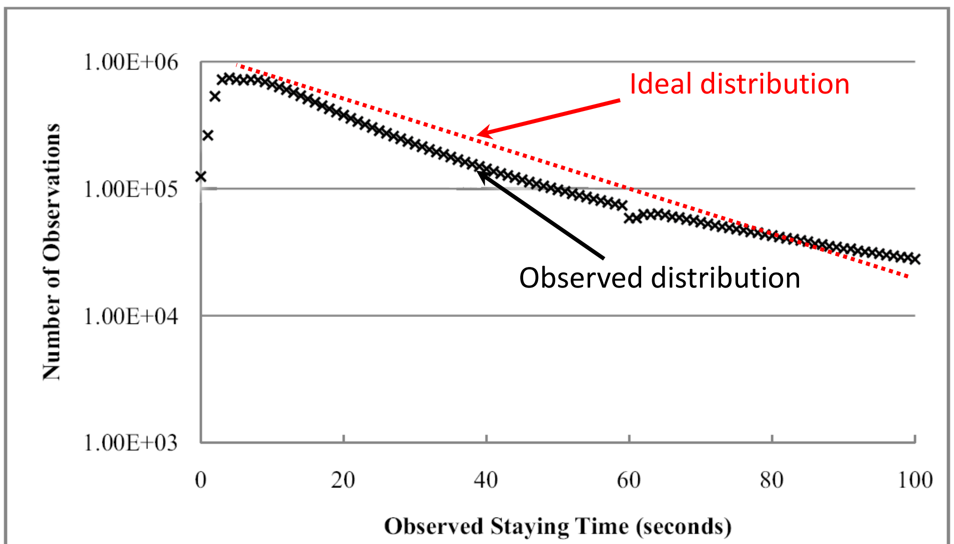
Estimating staying time distribution $t \sim \exp(\lambda)$

+

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

In theory, staying time is governed by an exponential distribution

- In practice, it is NOT!




Estimation with an additive noise model:

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

$$\min_{\lambda} \left( \left(\boxed{\bar{Z}} - \frac{1}{\lambda}\right) - \frac{1}{2}\left(\boxed{S^2} - \frac{1}{\lambda^2}\right) \right)^2$$

$$\text{s.t.} \quad \lambda > 0.$$

Sample mean of observed staying time

Sample variance of observed staying time

Calculating $\pi$

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

=

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

+

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

- Estimate transition probability matrix *P* of EMC.

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

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

| No. | PageRank | BrowseRank |
|---|---|---|
| 1 | adobe.com | ***myspace.com*** |
| 2 | passport.com | msn.com |
| 3 | msn.com | yahoo.com |
| 4 | microsoft.com | ***youtube.com*** |
| 5 | yahoo.com | live.com |
| 6 | google.com | ***facebook.com*** |
| 7 | mapquest.com | google.com |
| 8 | miibeian.gov.cn | ebay.com |
| 9 | w3.org | ***hi5.com*** |
| 10 | godaddy.com | ***bebo.com*** |
| 18 | paypal.com | ***wikipedia.org*** |
| 19 | aol.com | ***pogo.com*** |
| 20 | ***blogger.com*** | ***photobucket.com*** |

Web 2.0 websites

Web 2.0 sites are ranked high:

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

53 million sessions

# Results: Anti-Spam

| Bucket No. | Number of Websites | PageRank | | BrowseRank |
|---|---|---|---|---|
| 1 | 15 | 0 | | 0 |
| 2 | 148 | 2 | > | 1 |
| 3 | 720 | 9 | | 4 |
| 4 | 2231 | 22 | | 18 |
| 5 | 5610 | 30 | | 39 |
| 6 | 12600 | 58 | | 88 |
| 7 | 25620 | 90 | | 87 |
| 14 | 1414172 | 407 | | 463 |
| 15 | 2361420 | 306 | < | 756 |

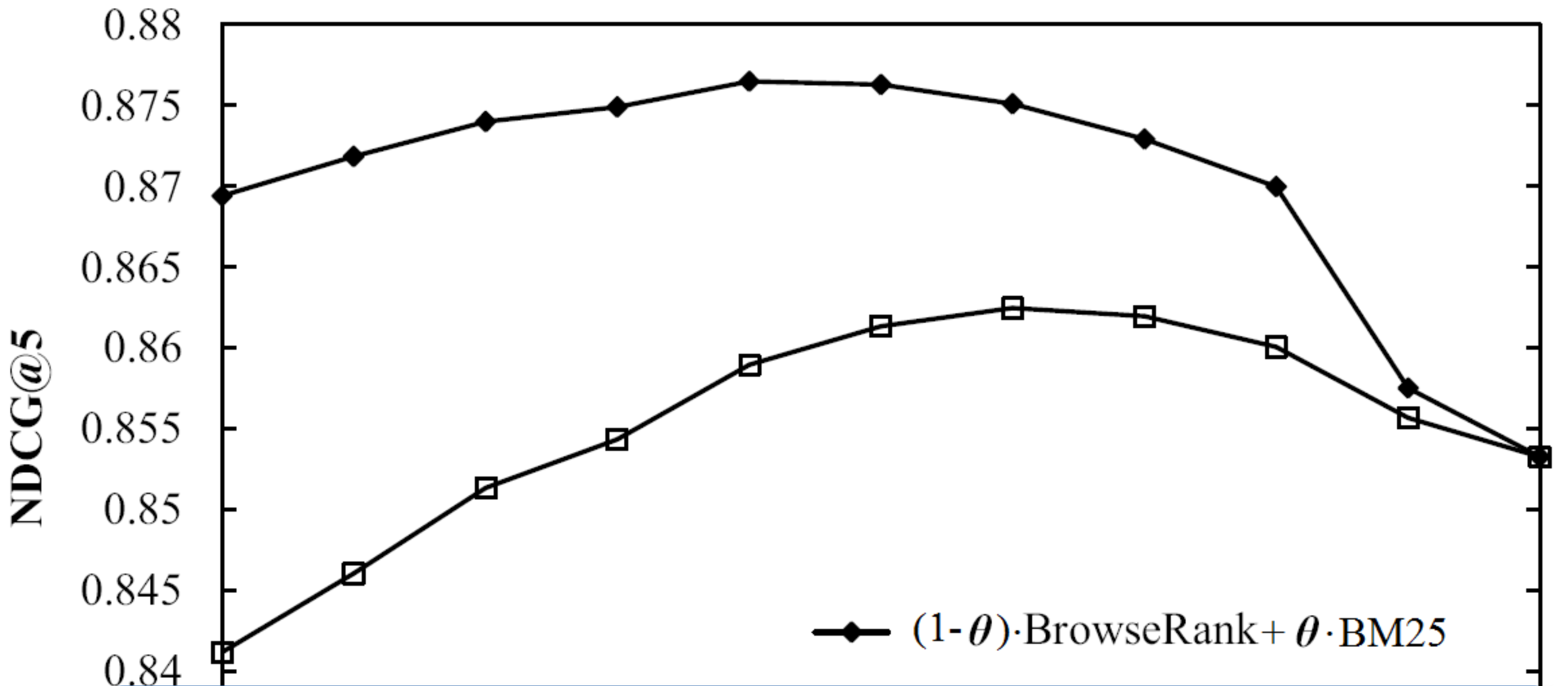
Number of spam websites in each bucket

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



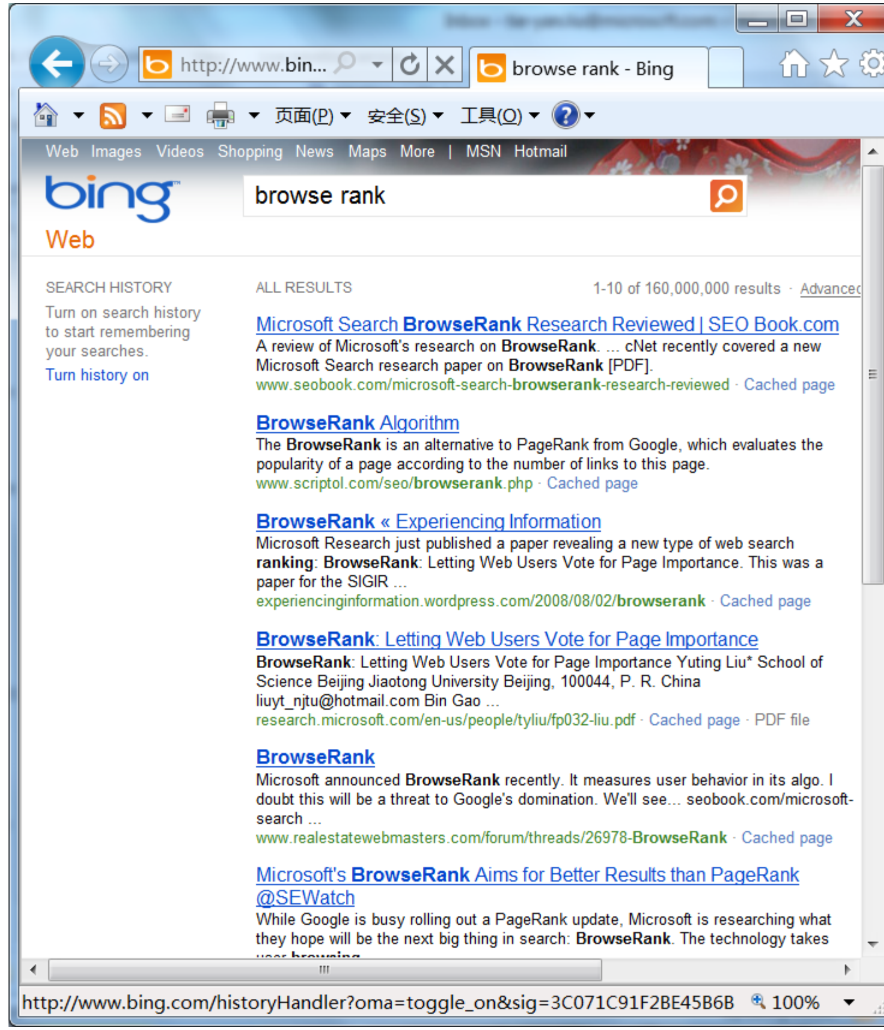

BrowseRank can significantly improve final ranking

Combining Parameter θ

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 → 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: $X = \{x_{ij}\}$ $x_{ij} = (x_{ij1}, x_{ij2}, \cdots, x_{ijl})^T$

Node features: $Y = \{y_i\}$ $y_i = (y_{i1}, y_{i2}, \cdots, y_{ih})^T$

Edge parameter vector: $\omega$

Node parameter vector: $\phi$

Page importance score: $\pi$

Link graph: $\mathcal{G}$

Supervision matrix: $B$

Weight vector for supervisions: $\mu$

# Optimization Problem

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

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

**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

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

Matrix size $n^2 \times l$

Matrix size $n^2 \times 1$

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

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

Matrix multiplication O($n^3$)

# First-Order Optimization: Details

$$\frac{\partial vec(P)}{\partial \omega^T} = \begin{pmatrix} \frac{\partial p_{11}}{\partial \omega_1} & \cdots & \frac{\partial p_{11}}{\partial \omega_l} \\ \vdots & \ddots & \vdots \\ \frac{\partial p_{n1}}{\partial \omega_1} & \cdots & \frac{\partial p_{n1}}{\partial \omega_l} \\ \vdots & \ddots & \vdots \\ \frac{\partial p_{1n}}{\partial \omega_1} & \cdots & \frac{\partial p_{1n}}{\partial \omega_l} \\ \vdots & \ddots & \vdots \end{pmatrix}$$

$$\frac{\partial r}{\partial \phi} = \begin{pmatrix} \frac{\partial r}{\partial \phi_1} \\ \vdots \\ \frac{\partial r}{\partial \phi_i} \\ \vdots \\ \frac{\partial r}{\partial \phi_h} \end{pmatrix}$$

O($n^3+n^2l$) seems very difficult to scale up to web scale!

# Solve the Problem in Linear Time

Denote $\pi' = \boxed{P^T \pi}$ $\qquad \pi'' = \pi' - \pi$

Iteration 1

Iteration 2

Iteration 3

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

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

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

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

$$\pi' = P^T \pi \longleftarrow \pi'_i = \sum_j p_{ji} \pi_j$$

  - **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 $\pi'_i$, $\pi'_i = \sum_j p_{ji}\pi_j$.

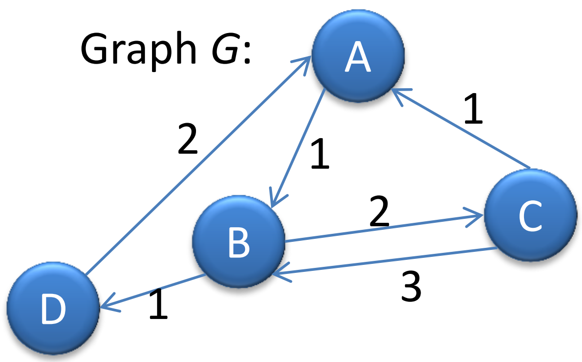
- Kronecker Product of Vectors on a Sparse Graph

$$z = x \otimes y$$

  - **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 $< i, x_i y_j >$ only if there is an edge from page $i$ to page $j$, and then emit $z_{(i-1)n+j} = x_i y_j$; otherwise, $z_{(i-1)n+j} = 0$.

# Details: Sparse Graph Index

Graph *G*:




Matrix P:

| | A | B | C | D |
|---|---|---|---|---|
| **A** | 0 | 1 | 0 | 0 |
| **B** | 0 | 0 | 2 | 1 |
| **C** | 1 | 3 | 0 | 0 |
| **D** | 2 | 0 | 0 | 0 |

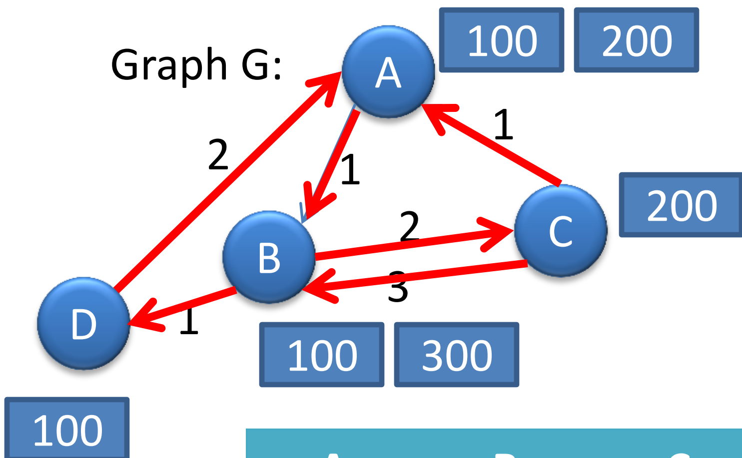
Sparse Matrix Data Stream:

[A, <(B,1)>], [B,<(C,2),(D,1)>], [C,<(A,1),(B,3)>], [D,<(A,2)>]

# Details: Matrix-Vector Multiplication

$Px$

| A | B | C | D |
|---|---|---|---|
| 100 | 100 | 100 | 100 |




| A | B | C | D |
|---|---|---|---|
| 300 | 400 | 200 | 100 |

# Details: Kronecker Product

| X1 | X2 | X3 | X4 |
|---|---|---|---|
| 1 | 2 | 3 | 4 |

$\otimes$

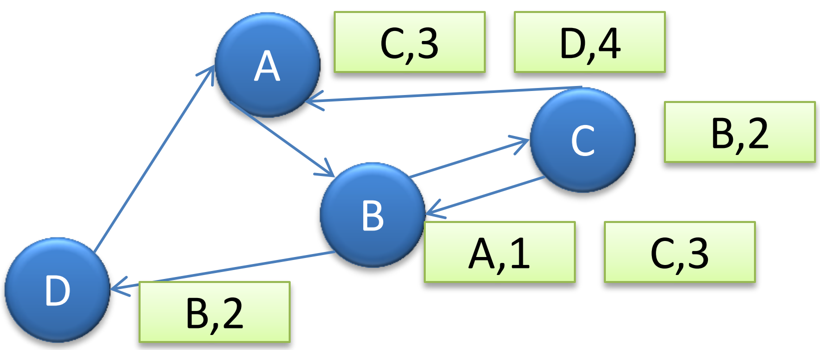
| Y1 | Y2 | Y3 | Y4 |
|---|---|---|---|
| 1 | 2 | 3 | 4 |

=

| X1 Y1 | X1 Y2 | X1 Y3 | X1 Y4 | X2 Y1 | X2 Y2 | X2 Y3 | X2 Y4 | X3 Y1 | X3 Y2 | X3 Y3 | X3 Y4 | X4 Y1 | X4 Y2 | X4 Y3 | X4 Y4 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 2 | 4 | 6 | 8 | 3 | 6 | 9 | 12 | 4 | 8 | 12 | 16 |
| | | 0 | 0 | | | | 0 | | | 0 | 0 | 0 | 0 | | |

# Details: Kronecker Product

- ✓ *Propagate y along graph G'(the inverted graph of G)*




y =

| A | B | C | D |
|---|---|---|---|
| 1 | 2 | 3 | 4 |

x =

| A | B | C | D |
|---|---|---|---|
| 5 | 6 | 7 | 8 |

- ✓ *Multiple x with the received y values*

Result =

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

# Output of the Learning Process

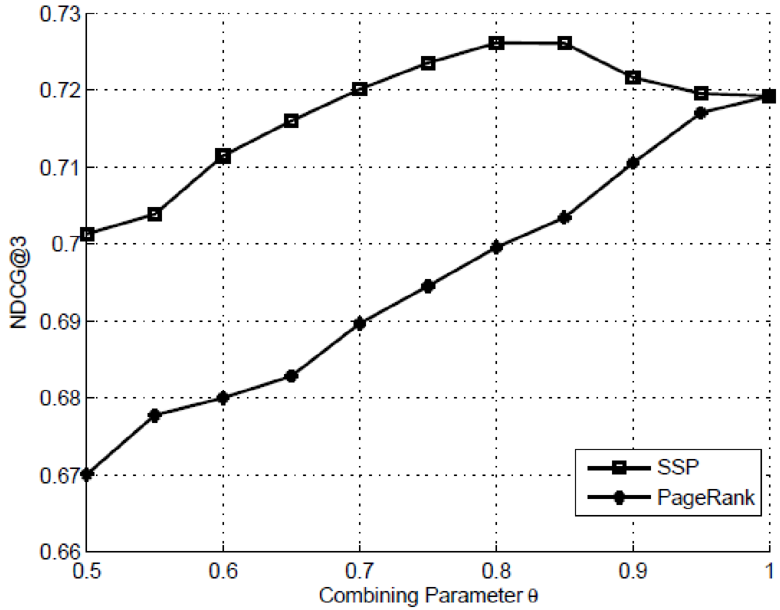
- $\pi$: can be used to direct rank nodes in the given graph.
- $\varphi$ and $\omega$ can be used to rank nodes in new graphs with similar generating mechanisms to the given graph (advantages of the parametric formulation).

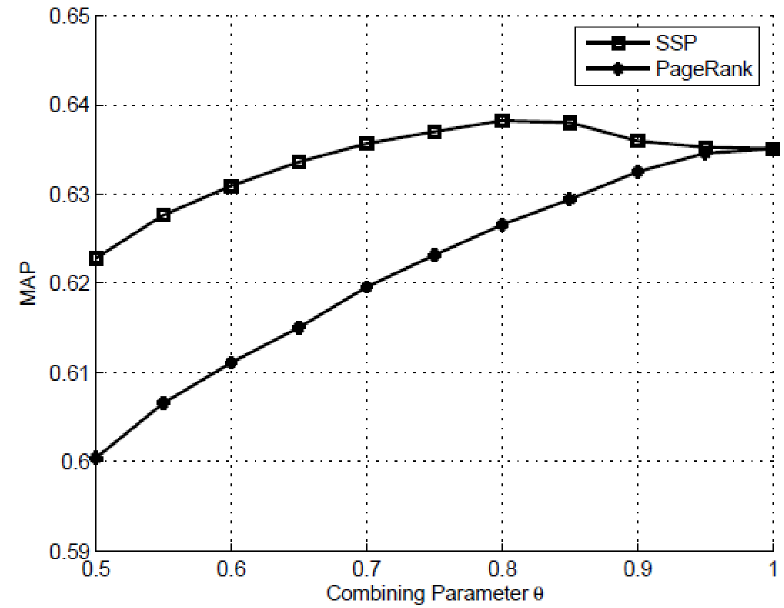
# Results: Anti-Spam

Table 3: Number of spam websites over buckets.

| No. | # of Websites | PageRank | AP | RankNet | SSP |
|---|---|---|---|---|---|
| 1 | 150 | 2 | 0 | 0 | 0 |
| 2 | 537 | 2 | 0 | 1 | 0 |
| 3 | 1257 | 1 | 1 | 1 | 0 |
| 4 | 2660 | 2 | 8 | 4 | 6 |
| 5 | 4788 | 4 | 7 | 4 | 6 |
| 6 | 8344 | 12 | 7 | 5 | 7 |
| 7 | 13708 | 7 | 16 | 23 | 12 |
| 8 | 20846 | 13 | 33 | 18 | 33 |
| 9 | 29008 | 19 | 25 | 34 | 27 |
| 10 | 33231 | 60 | 25 | 32 | 31 |

# Results: Relevance Ranking




- SSP consistently outperforms the other algorithms, with all θ 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/