PageRank, HITS and Link Prediction
Web: How to organize it?

- How to organize/navigate it?

- First try:
  - Web directories
    - Yahoo,
    - DMOZ,
    - LookSmart

![Web directories example](image-url)
Information Retrieval

- SEARCH!

- Find relevant docs in a small and trusted set:
  - Newspaper articles
  - Patents, etc.

- Two traditional problems:
  - Synonymy: buy – purchase, sick – ill
  - Polysemy: jaguar
The Index Size Wars

Does more documents mean better results?
What is “best” answer to query “Stanford”?

- **Anchor Text**: I go to [Stanford](http://cs224w.stanford.edu) where I study

What about query “newspaper”?

- No single right answer

Scarcity (IR) vs. abundance (Web) of information

- **Web**: Many sources of information. Who to “trust”

Trick:

- Pages that actually know about newspapers might all be pointing to many newspapers

**Ranking!**
Goal (back to the newspaper example):
- Don’t just find newspapers. Find “experts” – people who link in a coordinated way to good newspapers

Idea: Links as votes
- Page is more important if it has more links
  - In-coming links? Out-going links?

Hubs and Authorities
- Quality as an expert (hub):
  - Total sum of votes of pages pointed to
- Quality as an content (authority):
  - Total sum of votes of experts
- Principle of repeated improvement
Counting in-links: Authority

- SJ Merc News: 2 votes
- Wall St. Journal: 2 votes
- New York Times: 4 votes
- USA Today: 3 votes
- Facebook: 1 vote
- Yahoo!: 3 votes
- Amazon: 3 votes
Expert quality: Hub

![Diagram showing network analysis with nodes and votes]

- S.I. Merc News: 2 votes
- Wall St. Journal: 2 votes
- New York Times: 4 votes
- USA Today: 3 votes
- Facebook: 1 vote
- Yahoo!: 3 votes
- Amazon: 3 votes
Reweighting
Hubs and Authorities

- Each page $i$ has 2 kinds of scores:
  - Hub score: $h_i$
  - Authority score: $a_i$

- HITS algorithm:
  - Initialize: $a_i = h_i = 1$
  - Then keep iterating:
    - Authority: $a_j = \sum_{i \to j} h_i$
    - Hub: $h_i = \sum_{i \to j} a_j$
    - Normalize: $\sum a_i = 1, \sum h_i = 1$
Hubs and Authorities

- HITS converges to a single stable point
- Slightly change the notation:
  - Vector \( a = (a_1, \ldots, a_n) \), \( h = (h_1, \ldots, h_n) \)
  - Adjacency matrix (\( n \times n \)): \( M_{ij} = 1 \) if \( i \to j \)
- Then:
  \[
  h_i = \sum_{j \to i} a_j \iff h_i = \sum_j M_{ij} a_j
  \]
- So: \( h = Ma \)
- And likewise: \( a = M^T h \)
Algorithm in new notation:

- Set: $a = h = 1^n$
- Repeat:
  - $h = Ma$, $a = M^T h$
  - Normalize

Then: $a = M^T (Ma)$

Thus, in 2k steps:

$$a = (M^T M)^k a$$
$$h = (MM^T)^k h$$

a is being updated (in 2 steps):

$$M^T (Ma) = (M^T M) a$$

h is updated (in 2 steps):

$$M (M^T h) = (MM^T) h$$

Repeated matrix powering
Definition:
- Let $Ax = \lambda x$ for some scalar $\lambda$, vector $x$ and matrix $A$
- Then $x$ is an eigenvector, and $\lambda$ is its eigenvalue

Fact:
- If $A$ is symmetric ($A_{ij} = A_{ji}$) (in our case $M^TM$ and $MM^T$ are symmetric)
- Then $A$ has $n$ orthogonal unit eigenvectors $w_1 \ldots w_n$ that form a basis (coordinate system) with eigenvalues $\lambda_1 \ldots \lambda_n$ ($|\lambda_i| \geq |\lambda_{i+1}|$)
How to think about Ax?

- Write $x$ in coordinate system $w_1 \ldots w_n$
  
  $$x = \sum_i \alpha_i w_i$$

  - $x$ has coordinates $(\alpha_1, \ldots, \alpha_n)$

- Suppose: $\lambda_1 \ldots \lambda_n$ ($|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n|$)

- $A^kx = (\lambda_1^k, \lambda_2^k, \ldots, \lambda_n^k)$

- As $k \to \infty$, if we normalize
  
  $$A^kx \to \lambda_1 \alpha_1 w_1$$

  (all other coordinates $\to 0$)

- So authority $a$ is eigenvector of $M^TM$ associated with largest eigenvalue $\lambda_1$
PageRank: The “flow” model

- A vote from an important page is worth more
- A page is important if it is pointed to by other important pages
- Define a “rank” \( r_j \) for node \( j \)

\[
r_j \propto \sum_{i \rightarrow j} \frac{r_i}{\text{outdegree of } i}
\]

Flow equations:

\[
\begin{align*}
y &= y/2 + a/2 \\
a &= y/2 + m \\
m &= a/2
\end{align*}
\]
PageRank: Matrix formulation

- **Stochastic adjacency matrix** $M$
  - Let page $j$ has $d_j$ out-links
  - If $j \rightarrow i$, then $M_{ij} = 1/d_j$ else $M_{ij} = 0$
    - $M$ is a column stochastic matrix
      - Columns sum to 1

- **Rank vector** $r$: vector with 1 entry per page
  - $r_i$ is the importance score of page $i$
  - $|r| = 1$

- The flow equations can be written
  $$r = Mr$$
Imagine a random web surfer:

- At any time \( t \), surfer is on some page \( u \)
- At time \( t+1 \), the surfer follows an out-link from \( u \) uniformly at random
- Ends up on some page \( v \) linked from \( u \)
- Process repeats indefinitely

Let:

- \( p(t) \) ... vector whose \( i^{th} \) coordinate is the prob. that the surfer is at page \( i \) at time \( t \)
- \( p(t) \) is a probability distribution over pages
The Stationary Distribution

- Where is the surfer at time $t+1$?
  - Follows a link uniformly at random
    \[ p(t+1) = Mp(t) \]
- Suppose the random walk reaches a state
  \[ p(t+1) = Mp(t) = p(t) \]
  - then $p(t)$ is **stationary distribution** of a random walk
- **Our rank vector** $r$ satisfies $r = Mr$
  - So it is a stationary distribution for the random surfer
**PageRank: How to solve?**

- **Power Iteration:**
  - Set $r_i = 1$
  - $r_j = \sum_i r_i/d_i$
  - And iterate

- **Example:**

<table>
<thead>
<tr>
<th></th>
<th>y</th>
<th>a</th>
<th>m</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>1/2</td>
<td>1/2</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>1/2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>m</td>
<td>0</td>
<td>1/2</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>y</th>
<th>a</th>
<th>m</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>1</td>
<td>1</td>
<td>5/4</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
<td>3/2</td>
<td>1</td>
</tr>
<tr>
<td>m</td>
<td>1</td>
<td>1/2</td>
<td>3/4</td>
</tr>
</tbody>
</table>
Some pages are “dead ends” (have no out-links)
  - Such pages cause importance to leak out

Spider traps (all out links are within the group)
  - Eventually spider traps absorb all importance
Power Iteration:

- Set $r_i = 1$
- $r_j = \sum_i r_i / d_i$
- And iterate

Example:

<table>
<thead>
<tr>
<th></th>
<th>y</th>
<th>a</th>
<th>m</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>½</td>
<td>½</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>½</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>m</td>
<td>0</td>
<td>½</td>
<td>0</td>
</tr>
</tbody>
</table>

\[
\begin{array}{ccc}
  y & a & m \\
  1 & 1 & 3/4 & 5/8 & 0 \\
  1 & 1/2 & 1/2 & 3/8 & 0 \\
  1 & 1/2 & 1/4 & 1/4 & 0 \\
\end{array}
\]
Power Iteration:

- Set $r_i = 1$
- $r_j = \sum_i r_i / d_i$
- And iterate

Example:

<table>
<thead>
<tr>
<th></th>
<th>y</th>
<th>a</th>
<th>m</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>$\frac{1}{2}$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>m</td>
<td>0</td>
<td>$\frac{1}{2}$</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>y</th>
<th>1</th>
<th>1</th>
<th>$\frac{3}{4}$</th>
<th>$\frac{5}{8}$</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
<td>$\frac{3}{8}$</td>
<td>...</td>
</tr>
<tr>
<td>m</td>
<td>1</td>
<td>$\frac{3}{2}$</td>
<td>$\frac{7}{4}$</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>
Solution: Random jumps

- At each step, random surfer has two options:
  - With probability $1-\beta$, follow a link at random
  - With probability $\beta$, jump to some page uniformly at random

- PageRank equation:
  $$ r_j = (1-\beta) \sum_{i \rightarrow j} r_i/d_i + \beta $$

$d_i$ ... outdegree of node $i$
PageRank & eigenvectors

- PageRank as a principal eigenvector
  \[ r = Mr \iff r_j = \sum_i r_i/d_i \]

- But we really want:
  \[ r_j = (1 - \beta) \sum_i r_i/d_i + \beta \sum_i r_i \]

- Define:
  \[ M'_{ij} = (1 - \beta) M_{ij} + \beta 1/n \]

- Then: \( r = M'r \)

- What is \( \beta \)?
  - In practice \( \beta = 0.15 \) (5 links and jump)

\( d_i \) ... outdegree of node i
Example
Goal: Evaluate pages not just by popularity but by how close they are to the topic

Teleporting can go to:
- Any page with equal probability
  - (we used this so far)
- A topic-specific set of “relevant” pages
  - Topic-specific (personalized) PageRank

\[
M'_{ij} = (1-\beta) M_{ij} + \beta c
\]

(c...teleport vector)
- **Graphs and web search:**
  - Ranks nodes by “importance”

- **Personalized PageRank:**
  - Ranks proximity of nodes to the teleport nodes \( c \)

- **Proximity on graphs:**
  - **Q:** What is most related conference to ICDM?
  - **Random Walks with Restarts**
    - Teleport back: \( c = (0 \ldots 0, 1, 0 \ldots 0) \)
Application: TrustRank

- **Link Farms:** networks of millions of pages design to focus PageRank on a few undeserving webpages

- To minimize their influence use a teleport set of trusted webpages
  - E.g., homepages of universities
Link prediction task:

- Given $G[t_0, t_0']$ a graph on edges up to time $t_0'$, output a ranked list $L$ of links (not in $G[t_0, t_0']$) that are predicted to appear in $G[t_1, t_1']$.

Evaluation:

- $n = |E_{\text{new}}|$: # new edges that appear during the test period $[t_1, t_1']$.
- Take top $n$ elements of $L$ and count correct edges.
## Link prediction via node distance

- **Predict links a evolving collaboration network**

|    | training period | | | | Core |
|----|-----------------|---|-------------|---|
|    | authors | papers | collaborations | authors | $|E_{old}|$ | $|E_{new}|$
| astro-ph | 5343 | 5816 | 41852 | 1561 | 6178 | 5751 |
| cond-mat | 5469 | 6700 | 19881 | 1253 | 1899 | 1150 |
| gr-qc | 2122 | 3287 | 5724 | 486 | 519 | 400 |
| hep-ph | 5414 | 10254 | 47806 | 1790 | 6654 | 3294 |
| hep-th | 5241 | 9498 | 15842 | 1438 | 2311 | 1576 |

- **Core**: Since network data is very sparse
  - Consider only nodes with in-degree and out-degree of at least 3
Link prediction via node distance

- Rank potential links \((x, y)\) based on:

<table>
<thead>
<tr>
<th>Method</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graph distance</td>
<td>((\text{negated}) \text{ length of shortest path between } x \text{ and } y)</td>
</tr>
<tr>
<td>Common neighbors</td>
<td>(</td>
</tr>
<tr>
<td>Jaccard’s coefficient</td>
<td>(</td>
</tr>
<tr>
<td>Adamic/Adar</td>
<td>(\sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{\log</td>
</tr>
<tr>
<td>Preferential attachment</td>
<td>(</td>
</tr>
<tr>
<td>Katz(\beta)</td>
<td>(\sum_{\ell=1}^{\infty} \beta^\ell \cdot</td>
</tr>
</tbody>
</table>

where \(\text{paths}_{x,y}^{(\ell)} := \{\text{paths of length exactly } \ell \text{ from } x \text{ to } y\}\)

- Weighted: \(\text{paths}_{x,y}^{(1)} := \text{number of collaborations between } x, y\).
- Unweighted: \(\text{paths}_{x,y}^{(1)} := 1 \text{ iff } x \text{ and } y \text{ collaborate.}\)

- Hitting time
  - Stationary-normed: \(-H_{x,y}\)
  - Stationary-normed: \(-H_{x,y} \cdot \pi_y\)

- Commute time
  - Standard: \(-H_{x,y} + H_{y,x}\)
  - Stationary-normed: \(-H_{x,y} \cdot \pi_y + H_{y,x} \cdot \pi_x\)

where

- \(H_{x,y} := \text{expected time for random walk from } x \text{ to reach } y\)
- \(\pi_y := \text{stationary-distribution weight of } y\)

(proportion of time the random walk is at node \(y\))
Link prediction via node distance

$$\sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{\log |\Gamma(z)|}$$
- Recommend a list of possible friends
- **Supervised machine learning setting:**
  - **Training example:**
    - For every node $s$ have a list of nodes she will create links to $\{g_1, \ldots, g_k\}$
  - **Problem:**
    - Learn a model that will for a given node $s$ rank nodes $\{g_1, \ldots, g_k\}$ higher than other nodes in the network
- **How to combine node/edge attributes and network structure?**
  - Let’s learn how to bias random walks!
Let $s$ be the center node
Let $f_w(u, v)$ be a function that assigns a strength to each edge:

$$a_{uv} = f_w(u, v) = \exp(-w \Psi_{uv})$$

- $\Psi_{uv}$ is a feature vector
  - Features of node $u$
  - Features of node $v$
  - Features of edge $(u,v)$

- $w$ is the parameter vector we want to learn

Do a random walk from $s$ where transitions are according to edge strengths

How to learn $f_w(u, v)$?
Personalized PageRank

- Random walk transition matrix:
  \[ Q'_{uv} = \begin{cases} 
  \frac{a_{uv}}{\sum_w a_{uw}} & \text{if } (u, v) \in E, \\
  0 & \text{otherwise}
  \end{cases} \]

- PageRank transition matrix:
  \[ Q_{ij} = (1 - \alpha)Q'_{ij} + \alpha \mathbf{1}(j = s) \]
  - with prob. \( \alpha \) jump back to \( s \)

- Compute PageRank vector: \( p = p^T Q \)
- Rank nodes by \( p_u \)
The Optimization Problem

- Each node \( u \) has a score \( p_u \)
- Destination nodes \( D = \{v_1, \ldots, v_K\} \)
- No-link nodes \( L = \{\text{the rest}\} \)

What do we want?

\[
\min_w F(w) = ||w||^2
\]

such that

\[
\forall d \in D, l \in L : p_l < p_d
\]

- Hard constraints, make them soft
Want to minimize:

$$\min_w F(w) = \|w\|^2 + \lambda \sum_{ld} h(p_l - p_d)$$

- **Loss:** \(h(x) = 0\) if \(x < 0\), \(x^2\) else

How to minimize \(F\)?

- \(p_l\) and \(p_d\) depend on \(w\):
  - Given \(w\) assign edge weights \(a_{uv} = f_w(u,v)\)
  - Using transition matrix \(Q = [a_{uv}]\) compute PageRank scores \(p_u\)
  - Want to set \(w\) such that \(p_l < p_d\)
How to minimize F?

Take the derivative!

\[
\frac{\partial F}{\partial w} = 2w + \sum_{l,d} \frac{\partial h(p_l - p_d)}{\partial w} = 2w + \sum_{l,d} \frac{\partial h(\delta_{ld})}{\partial \delta_{ld}} \left( \frac{\partial p_l}{\partial w} - \frac{\partial p_d}{\partial w} \right)
\]

We know:

\[ p = p^T Q \] i.e. \[ p_u = \sum_j p_j Q_{ju} \]

So:

\[
\frac{\partial p_u}{\partial w} = \sum_j Q_{ju} \frac{\partial p_j}{\partial w} + p_j \frac{\partial Q_{ju}}{\partial w}
\]

Looks like the PageRank equation!
Iceland Facebook network
- 174,000 nodes (55% of population)
- Avg. degree 168
- Avg. person added 26 new friends/month

For every node $s$:
- Positive examples:
  - $D$={ new friendships of $s$ in Nov ‘09 }
- Negative examples:
  - $L$={ other nodes $s$ did not create new links to }
Node and Edge features for learning:

- Node:
  - Age
  - Gender
  - Degree

- Edge:
  - Age of an edge
  - Communication,
  - Profile visits
  - Co-tagged photos

Baselines:

- Decision trees and logistic regression:
  - Above features + 10 network features (PageRank, common friends)

Evaluation:

- AUC and precision at Top20
### Facebook: predicting your future friends

<table>
<thead>
<tr>
<th>Learning Method</th>
<th>ROC area</th>
<th>Prec@Top20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Walk with Restart</td>
<td>0.81725</td>
<td>6.80</td>
</tr>
<tr>
<td>Degree</td>
<td>0.58535</td>
<td>3.25</td>
</tr>
<tr>
<td>DT: Node features</td>
<td>0.59248</td>
<td>2.38</td>
</tr>
<tr>
<td>DT: Path features</td>
<td>0.62836</td>
<td>2.46</td>
</tr>
<tr>
<td>DT: All features</td>
<td>0.72986</td>
<td>5.34</td>
</tr>
<tr>
<td>LR: Node features</td>
<td>0.54134</td>
<td>1.38</td>
</tr>
<tr>
<td>LR: Path features</td>
<td>0.51418</td>
<td>0.74</td>
</tr>
<tr>
<td>LR: All features</td>
<td>0.81681</td>
<td>7.52</td>
</tr>
<tr>
<td>SRW: one edge type</td>
<td>0.82502</td>
<td>6.87</td>
</tr>
<tr>
<td>SRW: multiple edge types</td>
<td>0.82799</td>
<td>7.57</td>
</tr>
</tbody>
</table>
Results:
- 2.3X improvement over previous FB-PYMK system

How to scale to FB size?
- FB network:
  - >500 million people, >65 billion edges
  - 40 machines, each 72GB of RAM (total 2.8TB)
  - System makes 8.6 million suggests per second