Eigenvector Methods
in
Information Retrieval

Amy Langville
Carl Meyer

SAMSI Institute
Department of Mathematics
North Carolina State University

SIAM AN–New Orleans 7/12/2005
Outline

Part 1: Traditional IR

- Vector Space Model (1960s and 1970s)
- Latent Semantic Indexing (1990s)

Part 2: Web IR

- PageRank (1998)
- HITS (1998)
Vector Space Model (1960s and 1970s)

Gerard Salton’s Information Retrieval System
SMART: System for the Mechanical Analysis and Retrieval of Text
(Salton’s Magical Automatic Retriever of Text)

- turn $n$ textual documents into $n$ document vectors $d_1, d_2, \ldots, d_n$
- create term-by-document matrix $A_{m \times n} = [d_1 \mid d_2 \mid \cdots \mid d_n]$
- to retrieve info., create query vector $q$, which is a pseudo-doc
Vector Space Model (1960s and 1970s)

Gerard Salton’s Information Retrieval System
SMART: System for the Mechanical Analysis and Retrieval of Text
(Salton’s Magical Automatic Retriever of Text)

- turn $n$ textual documents into $n$ document vectors $\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_n$
- create term-by-document matrix $\mathbf{A}_{m \times n} = [\mathbf{d}_1 | \mathbf{d}_2 | \cdots | \mathbf{d}_n ]$
- to retrieve info., create query vector $\mathbf{q}$, which is a pseudo-doc

GOAL: find doc. $\mathbf{d}_i$ closest to $\mathbf{q}$

--- angular cosine measure used: $\delta_i = \cos \theta_i = \mathbf{q}^T \mathbf{d}_i / (\|\mathbf{q}\|_2 \|\mathbf{d}_i\|_2)$
<table>
<thead>
<tr>
<th>Terms</th>
<th>Documents</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1: Bab(y,ies,y’s)</td>
<td>D1: <em>Infant &amp; Toddler First Aid</em></td>
</tr>
<tr>
<td>T2: Child(ren’s)</td>
<td>D2: <em>Babies &amp; Children’s Room (For Your Home)</em></td>
</tr>
<tr>
<td>T3: Guide</td>
<td>D3: <em>Child Safety at Home</em></td>
</tr>
<tr>
<td>T4: Health</td>
<td>D4: <em>Your Baby’s Health &amp; Safety: From Infant to Toddler</em></td>
</tr>
<tr>
<td>T5: Home</td>
<td>D5: <em>Baby Proofing Basics</em></td>
</tr>
<tr>
<td>T6: Infant</td>
<td>D6: <em>Your Guide to Easy Rust Proofing</em></td>
</tr>
<tr>
<td>T7: Proofing</td>
<td>D7: <em>Beanie Babies Collector’s Guide</em></td>
</tr>
<tr>
<td>T8: Safety</td>
<td></td>
</tr>
<tr>
<td>T9: Toddler</td>
<td></td>
</tr>
</tbody>
</table>
Example from Berry’s book

<table>
<thead>
<tr>
<th>Terms</th>
<th>Documents</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1: Bab(y,ies,y’s)</td>
<td>D1: Infant &amp; Toddler First Aid</td>
</tr>
<tr>
<td>T2: Child(ren’s)</td>
<td>D2: Babies &amp; Children's Room (For Your Home)</td>
</tr>
<tr>
<td>T3: Guide</td>
<td>D3: Child Safety at Home</td>
</tr>
<tr>
<td>T4: Health</td>
<td>D4: Your Baby’s Health &amp; Safety : From Infant to Toddler</td>
</tr>
<tr>
<td>T5: Home</td>
<td>D5: Baby Proofing Basics</td>
</tr>
<tr>
<td>T6: Infant</td>
<td>D6: Your Guide to Easy Rust Proofing</td>
</tr>
<tr>
<td>T7: Proofing</td>
<td>D7: Beanie Babies Collector’s Guide</td>
</tr>
<tr>
<td>T8: Safety</td>
<td></td>
</tr>
<tr>
<td>T9: Toddler</td>
<td></td>
</tr>
</tbody>
</table>

\[
A = \begin{pmatrix}
  t_1 & d_1 & d_2 & d_3 & d_4 & d_5 & d_6 & d_7 \\
  t_2 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\
  t_3 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
  t_4 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
  t_5 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
  t_6 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
  t_7 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
  t_8 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
  t_9 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
\end{pmatrix}
\]

\[
q = \begin{bmatrix}
  1 \\
  0 \\
  0 \\
  0 \\
  0 \\
  0 \\
  0 \\
\end{bmatrix}
\]

\[
\delta = \begin{bmatrix}
  \delta_1 \\
  \delta_2 \\
  \delta_3 \\
  \delta_4 \\
  \delta_5 \\
  \delta_6 \\
  \delta_7 \\
\end{bmatrix} = \begin{bmatrix}
  0 \\
  .5774 \\
  0 \\
  .8944 \\
  .7071 \\
  0 \\
  .7071 \\
\end{bmatrix}
\]
Latent Semantic Indexing (1990s)

Susan Dumais’s improvement to VSM = LSI

Idea: use low-rank approximation to $A$ to filter out noise

- Use truncated SVD as low-rank approximation to $A$
\[ A_{m \times n} : \text{rank } r \text{ term-by-document matrix} \]

- **SVD**: \[ A = U \Sigma V^T = \sum_{i=1}^{r} \sigma_i u_i v_i^T \]
- **LSI**: use \[ A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T \] in place of \( A \)
- **Why?**
  - reduce storage when \( k << r \)
  - filter out uncertainty, so that performance on text mining tasks (e.g., query processing and clustering) improves
What’s Really Happening?

Change of Basis using truncated SVD $A_k = U_k \Sigma_k V_k^T$

- **Original Basis:** docs represented in Term Space using Standard Basis $S = \{e_1, e_2, \ldots, e_m\}$

- **New Basis:** docs represented in smaller Latent Semantic Space using Basis $B = \{u_1, u_2, \ldots, u_k\}$ ($k << \min(m,n)$)

$$
\begin{align*}
\text{doc}_1 \\
\vdots \\
A_{*1} \\
\vdots \\
m \times 1
\end{align*}
\begin{align*}
\approx \\
\begin{bmatrix}
\vdots \\
u_1 \\
\vdots \\
\vdots \\
u_k
\end{bmatrix}
\sigma_1 v_{11} + \\
\begin{bmatrix}
\vdots \\
u_2 \\
\vdots \\
\vdots \\
u_k
\end{bmatrix}
\sigma_2 v_{12} + \cdots + \\
\begin{bmatrix}
\vdots \\
u_k
\end{bmatrix}
\sigma_k v_{1k}
\end{align*}
$$
What's Really Happening?

Change of Basis using truncated SVD $A_k = U_k \Sigma_k V_k^T$

- Original Basis: docs represented in Term Space using Standard Basis $S = \{e_1, e_2, \ldots, e_m\}$

- New Basis: docs represented in smaller Latent Semantic Space using Basis $B = \{u_1, u_2, \ldots, u_k\}$ $(k << \min(m,n))$

\[
\begin{pmatrix}
\text{doc}_1 \\
\vdots \\
\text{A*1}
\end{pmatrix}_{m \times 1} \approx
\begin{bmatrix}
\vdots \\
u_1 \\
\vdots
\end{bmatrix} \sigma_1 v_{11} +
\begin{bmatrix}
\vdots \\
u_2 \\
\vdots
\end{bmatrix} \sigma_2 v_{12} + \cdots +
\begin{bmatrix}
\vdots \\
u_k \\
\vdots
\end{bmatrix} \sigma_k v_{1k}
\]

- still use angular cosine measure

\[
\delta_i = \cos \theta_i = q^T d_i / (\|q\|_2 \|d_i\|_2) = q^T A_k e_i / (\|q\|_2 \|A_k e_i\|_2)
\]

\[
= q^T U_k \Sigma_k V_k^T e_i / (\|q\|_2 \| \Sigma_k V_k^T e_i \|_2)
\]
Strengths and Weaknesses of LSI

Strengths

- using $A_k$ in place of $A$ gives improved performance
- dimension reduction considers only essential components of term-by-document matrix, filters out noise
- best rank-$k$ approximation: $\|A - A_k\|_F = \min_{\text{rank}(B) \leq k} \|A - B\|_F$

Weaknesses

- storage—$U_k$ and $V_k$ are usually completely dense
- interpretation of basis vectors $u_i$ is impossible due to mixed signs
- good truncation point $k$ is hard to determine
- orthogonality restriction
Web Information Retrieval

IR before the Web = traditional IR
IR on the Web = web IR
Web Information Retrieval

IR before the Web = traditional IR
IR on the Web = web IR

How is the Web different from other document collections?
Web Information Retrieval

IR before the Web = traditional IR
IR on the Web = web IR

How is the Web different from other document collections?

- It’s huge.
  - over 10 billion pages, average page size of 500KB
  - 20 times size of Library of Congress print collection
  - Deep Web - 550 billion pages
Web Information Retrieval

IR before the Web = traditional IR
IR on the Web = web IR

How is the Web different from other document collections?

• It’s huge.
  - over 10 billion pages, average page size of 500KB
  - 20 times size of Library of Congress print collection
  - Deep Web - 550 billion pages

• It’s dynamic.
  - content changes: 40% of pages change in a week, 23% of .com change daily
  - size changes: billions of pages added each year
Web Information Retrieval

IR before the Web = traditional IR
IR on the Web = web IR

How is the Web different from other document collections?

- It’s huge.
  - over 10 billion pages, average page size of 500KB
  - 20 times size of Library of Congress print collection
  - Deep Web - 550 billion pages

- It’s dynamic.
  - content changes: 40% of pages change in a week, 23% of .com change daily
  - size changes: billions of pages added each year

- It’s self-organized.
  - no standards, review process, formats
  - errors, falsehoods, link rot, and spammers!
Web Information Retrieval

IR before the Web = traditional IR
IR on the Web = web IR

How is the Web different from other document collections?

- It’s huge.
  - over 10 billion pages, average page size of 500KB
  - 20 times size of Library of Congress print collection
  - Deep Web - 550 billion pages

- It’s dynamic.
  - content changes: 40% of pages change in a week, 23% of .com change daily
  - size changes: billions of pages added each year

- It’s self-organized.
  - no standards, review process, formats
  - errors, falsehoods, link rot, and spammers!

A Herculean Task!
Web Information Retrieval

IR before the Web = traditional IR
IR on the Web = web IR

How is the Web different from other document collections?

- **It’s huge.**
  - over 10 billion pages, each about 500KB
  - 20 times size of Library of Congress print collection
  - Deep Web - 550 billion pages

- **It’s dynamic.**
  - content changes: 40% of pages change in a week, 23% of .com change daily
  - size changes: billions of pages added each year

- **It’s self-organized.**
  - no standards, review process, formats
  - errors, falsehoods, link rot, and spammers!

- **Ah, but it’s hyperlinked!**
  - Vannevar Bush’s 1945 memex
Term-by-Document Matrix for Web

- Too big for factorizations

- $\Rightarrow$ fast inverted file + link analysis
Elements of a Web Search Engine

Diagram:
- WWW
- Crawler Module
- Page Repository
- User
- Query Module
- Ranking Module
- Indexing Module
- Indexes
- Content Index
- Structure Index
- Special-purpose indexes

Queries:
- Results
Query Processing

Step 1: User enters query, i.e., aztec baby

Step 2: Inverted file consulted

- term 1 (aardvark) - 3, 117, 3961
  -
  -
  -
- term 10 (aztec) - 3, 15, 19, 101, 673, 1199
- term 11 (baby) - 3, 31, 56, 94, 673, 909, 11114, 253791
  -
  -
- term m (zymurgy) - 1159223

Step 3: Relevant set identified, i.e. (3, 673)

Simple traditional engines stop here.
Link Analysis

- uses *hyperlink* structure to focus the relevant set
- combine IR score with *popularity* or *importance* score

PageRank - Brin and Page

HITS - Kleinberg
The Web as a Graph

Nodes = webpages  Arcs = hyperlinks
How to Use Web Graph for Search

Hyperlink = Recommendation

• page with 20 recommendations (inlinks) must be more important than page with 2 inlinks.

• but status of recommender matters.
  EX: letters of recommendation: 1 letter from Trump vs. 20 from unknown people

• but what if recommender is generous with recommendations?
  EX: suppose Trump has written over 40,000 letters.

• each inlink should be weighted to account for status of recommender and # of outlinks from that recommender
How to Use Web Graph for Search

**Hyperlink** = Recommendation

- page with 20 recommendations (inlinks) must be more important than page with 2 inlinks.

- but status of recommender matters.
  
  **EX:** letters of recommendation: 1 letter from Trump vs. 20 from unknown people

- but what if recommender is generous with recommendations?
  
  **EX:** suppose Trump has written over 40,000 letters.

- each inlink should be weighted to account for status of recommender and # of outlinks from that recommender

**PAGERANK** - importance/popularity score given to each page
Ranking by PageRank

The PageRank Idea

- Ranking is preassigned
- Your page $P$ has some rank $r(P)$
- Adjust $r(P)$ higher or lower depending on ranks of pages that point to $P$
- Importance is not just number, but \textit{quality} of in-links
  - role of outlinks relegated
  - much less sensitive to spamming

(Sergey Brin & Lawrence Page 1998)
PageRank

The Definition

- \( r(P) = \sum_{P \in B_P} \frac{r(P)}{|P|} \)
- \( B_P = \{ \text{all pages pointing to } P \} \)
- \( |P| = \text{number of out links from } P \)

Successive Refinement

- Start with \( r_0(P_i) = 1/n \) for all pages \( P_1, P_2, \ldots, P_n \)
- Iteratively refine rankings for each page

\[
\begin{align*}
  r_1(P_i) &= \sum_{P \in B_{P_i}} \frac{r_0(P)}{|P|} \\
  r_2(P_i) &= \sum_{P \in B_{P_i}} \frac{r_1(P)}{|P|} \\
  \vdots \\
  r_{j+1}(P_i) &= \sum_{P \in B_{P_i}} \frac{r_j(P)}{|P|}
\end{align*}
\]
In Matrix Notation

After Step $j$

$$\pi_j^T = [r_j(P_1), r_j(P_2), \cdots, r_j(P_n)]$$

$$\pi_{j+1}^T = \pi_j^T H \quad \text{where} \quad h_{ij} = \begin{cases} 
\frac{1}{|P_i|} & \text{if } i \rightarrow j \\
0 & \text{o.w.}
\end{cases}$$
In Matrix Notation

After Step $j$

$$\pi_j^T = [r_j(P_1), r_j(P_2), \cdots, r_j(P_n)]$$

$$\pi_{j+1}^T = \pi_j^T H$$

where

$$h_{ij} = \begin{cases} 1/|P_i| & \text{if } i \rightarrow j \\ 0 & \text{o.w.} \end{cases}$$

$$H = \begin{pmatrix}
    p_1 & p_2 & p_3 & p_4 & p_5 & p_6 \\
    p_1 & 0 & 1/2 & 1/2 & 0 & 0 & 0 \\
    p_2 & 0 & 0 & 0 & 0 & 0 & 0 \\
    p_3 & 1/3 & 1/3 & 0 & 0 & 1/3 & 0 \\
    p_4 & 0 & 0 & 0 & 0 & 1/2 & 1/2 \\
    p_5 & 0 & 0 & 0 & 1/2 & 0 & 1/2 \\
    p_6 & 0 & 0 & 0 & 1 & 0 & 0
\end{pmatrix}$$
In Matrix Notation

After Step $j$

$$\pi^T_j = [r_j(P_1), r_j(P_2), \cdots, r_j(P_n)]$$

$$\pi^T_{j+1} = \pi^T_j H \quad \text{where} \quad h_{ij} = \begin{cases} 
1/|P_i| & \text{if } i \to j \\
0 & \text{o.w.}
\end{cases}$$

$$H = \begin{pmatrix}
p_1 & p_2 & p_3 & p_4 & p_5 & p_6 \\
p_1 & 0 & 1/2 & 1/2 & 0 & 0 & 0 \\
p_2 & 0 & 0 & 0 & 0 & 0 & 0 \\
p_3 & 1/3 & 1/3 & 0 & 0 & 1/3 & 0 \\
p_4 & 0 & 0 & 0 & 0 & 1/2 & 1/2 \\
p_5 & 0 & 0 & 0 & 1/2 & 0 & 1/2 \\
p_6 & 0 & 0 & 0 & 1 & 0 & 0 
\end{pmatrix}$$

PageRank = \( \lim_{j \to \infty} \pi^T_j = \pi^T \) (provided limit exists)

It’s Almost a Markov Chain

$H$ has row sums = 1 for ND nodes, row sums = 0 for D nodes
In Matrix Notation

It’s Almost a Markov Chain

- $H$ has row sums = 1 for ND nodes, row sums = 0 for D nodes
In Matrix Notation

It’s Almost a Markov Chain

- $H$ has row sums $= 1$ for ND nodes, row sums $= 0$ for D nodes

Stochasticity Fix: $S = H + av^T$. $\quad (a_i = 1 \text{ for } i \in D, \ 0, \ o.w.)$
In Matrix Notation

It’s Almost a Markov Chain

- \( H \) has row sums = 1 for ND nodes, row sums = 0 for D nodes

Stochasticity Fix: \( S = H + av^T. \) \( (a_i=1 \text{ for } i\in D, \ 0, \text{ o.w.}) \)

\[
S = \begin{bmatrix}
0 & 1/2 & 1/2 & 0 & 0 & 0 \\
1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\
1/3 & 1/3 & 0 & 0 & 1/3 & 0 \\
0 & 0 & 0 & 0 & 1/2 & 1/2 \\
0 & 0 & 0 & 1/2 & 0 & 1/2 \\
0 & 0 & 0 & 1 & 0 & 0
\end{bmatrix}
\]

where \( a = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \]

\( v^T = 1/6 \ e^T \)
In Matrix Notation

It’s Almost a Markov Chain

- \( H \) has row sums = 1 for ND nodes, row sums = 0 for D nodes

Stochasticity Fix: \( S = H + av^T. \)  \((a_i=1 \text{ for } i \in D, \text{ 0, o.w.})\)

\[
S = \begin{bmatrix}
0 & 1/2 & 1/2 & 0 & 0 & 0 \\
1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\
1/3 & 1/3 & 0 & 0 & 1/3 & 0 \\
0 & 0 & 0 & 0 & 1/2 & 1/2 \\
0 & 0 & 0 & 1/2 & 0 & 1/2 \\
0 & 0 & 0 & 1 & 0 & 0 \\
\end{bmatrix}
\]

where \( a = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \), \( v^T = \frac{1}{6} \ e^T \)

- Each \( \pi^T_j \) is a probability distribution vector \((\sum_i r_j(P_i)=1)\)
- \( \pi^T_{j+1} = \pi^T_j S \) is random walk on the graph defined by links
- \( \pi^T = \lim_{j \to \infty} \pi^T_j \) = stationary probability distribution
Random Surfer

Could still encounter Convergence Problems
(dangling nodes, cycles, reducibility)

Irreducibility Fix: \( G = \alpha S + (1 - \alpha)E \) \( e_{ij} = \frac{1}{n} \) \( \alpha \approx .85 \)

\( G = \alpha H + \alpha a v^T + (1 - \alpha)E \) (trivially irreducible)

- \( \pi^T \) is now guaranteed to exist and be unique and power method is guaranteed to converge to \( \pi^T \).
Random Surfer

Could still encounter Convergence Problems

Irreducibility Fix: \[ G = \alpha S + (1 - \alpha)E \quad e_{ij} = 1/n \quad \alpha \approx .85 \]

\[ G = \alpha H + \alpha a v^T + (1 - \alpha)E \quad \text{(trivially irreducible)} \]

- \( \pi^T \) is now guaranteed to exist and be unique and power method is guaranteed to converge to \( \pi^T \).
- Different \( E = ev^T \) and \( \alpha \) allow customization & speedup, yet rank-one update maintained; \( G = \alpha H + (\alpha a + (1 - \alpha)e)v^T \)

\[
G = \alpha S + (1 - \alpha)E = \begin{bmatrix}
    1/60 & 7/15 & 7/15 & 1/60 & 1/60 & 1/60 \\
    1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\
    19/60 & 19/60 & 1/60 & 1/60 & 19/60 & 1/60 \\
    1/60 & 1/60 & 1/60 & 1/60 & 7/15 & 7/15 \\
    1/60 & 1/60 & 1/60 & 7/15 & 1/60 & 7/15 \\
    1/60 & 1/60 & 1/60 & 11/12 & 1/60 & 1/60
\end{bmatrix}
\]
PageRank Example

\[ \pi^T = \begin{pmatrix} .03721 & .05396 & .04151 & .3751 & .206 & .2862 \end{pmatrix} \]

Global ranking of pages = [4 6 5 2 3 1]

Query-independent way of ranking relevant set
Ranking by HITS

- give each page 2 scores (hub and authority scores) instead of just 1.

- **DEFN:**
  - **Authorities**
  - **Hubs**

- pages can be both hubs and authorities (EX: ATL airport)

- Good hub pages point to good authority pages, and good authorities are pointed to by good hubs.

**HITS** - hub and authority score given to each page

**HITS** - (Hypertext Induced Topic Search) ⇒ Teoma
HITS Algorithm
Hypertext Induced Topic Search
(J. Kleinberg 1998)

Determine Authority & Hub Scores
- \( a_i = \text{authority score for } P_i \)
- \( h_i = \text{hub score for } P_i \)

Successive Refinement
- Start with \( h_i(0) = 1 \) for all pages \( P_i \)
- Successively refine rankings
  - For \( k = 1, 2, \ldots \)
    \[
    a_i(k) = \sum_{j: P_j \rightarrow P_i} h_j(k - 1) \quad \Rightarrow \quad a_k = L^T h_{k-1}
    \]
    \[
    h_i(k) = \sum_{j: P_i \rightarrow P_j} a_j(k) \quad \Rightarrow \quad h_k = L a_k
    \]
- \( A = L^T L \) \( a_k = A a_{k-1} \rightarrow \text{e-vector} \)
- \( H = LL^T \) \( h_k = H h_{k-1} \rightarrow \text{e-vector} \)
1. Find relevant set by consulting inverted file

2. Build neighborhood graph

3. Compute authority & hub scores for just the neighborhood
HITS Example

1. Relevant set = [1, 6]

2. Neighborhood graph \( N \)

3. Compute authority & hub scores.

Adjacency matrix for \( N = L = \)

\[
\begin{pmatrix}
1 & 0 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & 0 & 0 & 0 & 1 & 0 & 0 \\
5 & 0 & 0 & 0 & 0 & 0 & 0 \\
6 & 0 & 0 & 1 & 1 & 0 & 0 \\
10 & 0 & 0 & 0 & 0 & 1 & 0
\end{pmatrix}
\]
HITS Example (cont.)

Authority matrix \( A = L^T L \)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>5</th>
<th>6</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Hub matrix \( H = LL^T \)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>5</th>
<th>6</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Authority score vector \( a \)

\[
\begin{bmatrix}
1 \\
0 \\
.3660 \\
.1340 \\
.5 \\
0
\end{bmatrix}
\]

Hub score vector \( h \)

\[
\begin{bmatrix}
1 \\
.3660 \\
0 \\
.2113 \\
0 \\
.2113 \\
0 \\
.2113
\end{bmatrix}
\]
Conclusions

- These three information retrieval methods rely on eigenvector calculations.

- Large-scale matrices involved.

- Robust, efficient algorithms are essential.