Markov Chains, SANs and Search Engines

Amy Langville

College of Charleston

May 3, 2002
Outline

- Markov chains and SANs
  - definitions
  - stationary analysis techniques
  - difficulties with MC and SAN analysis
  - NKP Preconditioner

- Search Engines
  - Introduction
  - VSM
  - Google and MCs
PART 1: MCs and SANs
Markov chains

- stochastic process which follows Markov property. \( \Rightarrow \) State of system at time \( t \) only depends on the most recent past.

- DTMC

\[
P = \begin{pmatrix}
p_{1,1} & p_{1,2} & \cdots & p_{1,n} \\p_{2,1} & p_{2,2} & \cdots & p_{2,n} \\
\vdots & \vdots & \ddots & \vdots \\p_{n,1} & p_{n,2} & \cdots & p_{n,n}
\end{pmatrix},
\]

where \( p_{i,j} = \) conditional probability of moving from \( i \) to \( j \) in one time step.
• CTMC

\[ Q = \begin{pmatrix}
q_{1,1} & q_{1,2} & \cdots & q_{1,n} \\
q_{2,1} & q_{2,2} & \cdots & q_{2,n} \\
\vdots & \vdots & \ddots & \vdots \\
q_{n,1} & q_{n,2} & \cdots & q_{n,n}
\end{pmatrix}, \]

where \( q_{i,j} = \) transition rate of moving from \( i \) to \( j \) and \( q_{i,i} = -\sum_{j \neq i} q_{i,j} \).

• Relationship between \( P \) and \( Q \).
• Once we have $P$ or $Q$ for MC, we can begin analysis.

There are two main types of analysis:

– Transient analysis: find prob. dist. vector $\pi(t)$ at any time $t$

  * EX: $\pi_1(5) =$ prob. it will be good weather in Charleston 5 days from now.

– Stationary analysis: find prob. dist. vector $\pi$ as $t \to \infty$

  * EX: $\pi_1 =$ long-run proportion of time Charleston will have good weather.
The Stationary Problem

• Now let $t \to \infty$.

GOAL: Find $\pi$, the long-run steady-state probability vector.

$\pi_i =$ proportion of time system spends in state $i$.

• The solution $\pi$ is calculated from the homogeneous linear system

$$\pi Q = 0, \quad \pi e = 1.$$ 

• Or eigenvalue problem of finding e-vector associated with unit e-value.

$$\pi P = \pi, \quad \pi e = 1.$$
Stochastic Automata Networks (SANs)

- Problem with MC analysis = size of $P$ or $Q$.

- One solution = SANs (represent $Q$ in compact form)

- A SAN is a collection of stochastic automata which act more or less independently requiring only infrequent interaction.
Types of Infrequent Interaction

- *Functional Transitions*: rate at which transition may occur in one automaton may be function of states of other automata.
  
  - Example: Resource Sharing Model

- *Synchronizing events*: transition in one automaton may force transition to occur in one or more other automata.
  
  - Example: Two service centers in tandem
Kronecker Algebra: \( \otimes, \oplus \) are key SAN operations.

- \( A \in \mathbb{R}^{m_1 \times n_1}, B \in \mathbb{R}^{m_2 \times n_2} \), then \( A \otimes B \in \mathbb{R}^{m_1 m_2 \times n_1 n_2} \) and is defined by

\[
A \otimes B = \begin{pmatrix}
a_{1,1}B & \cdots & a_{1,n_1}B \\
\vdots & \ddots & \vdots \\
am_{m_1,1}B & \cdots & a_{m_1,n_1}B
\end{pmatrix}.
\]

- \( A \oplus B = A \otimes I_{m_2} + I_{m_1} \otimes B. \) (\( \oplus \) is defined in terms of \( \otimes \).)

(defined only for square matrices)

- Some Kronecker Properties:

1. \( (A \otimes B)^{-1} = A^{-1} \otimes B^{-1}. \)

2. \( \oplus_{i=1}^{N} A^{(i)} = \sum_{i=1}^{N} I_{n_1} \otimes \cdots \otimes I_{n_{i-1}} \otimes A^{(i)} \otimes I_{n_{i+1}} \otimes \cdots \otimes I_{n_N}. \)

3. \( (A \otimes B)^\dagger = A^\dagger \otimes B^\dagger. \)

4. \( (A \otimes B)^D = A^D \otimes B^D. \)
SAN Example: A Simple Queueing Network

Queueing network: two exponential, finite-capacity, single-server stations in tandem. When a station is full, customers are lost. For example, for \( C_1 = 1 \) and \( C_2 = 2 \), the transition rate matrix is given by

\[
Q = \begin{pmatrix}
-\lambda & 0 & 0 & \lambda & 0 & 0 \\
\nu & -(\nu + \lambda) & 0 & 0 & \lambda & 0 \\
0 & \nu & -(\nu + \lambda) & 0 & 0 & \lambda \\
\mu(1-p) & \mu p & 0 & -\mu & 0 & 0 \\
0 & \mu(1-p) & \mu p & \nu & -(\mu + \nu) & 0 \\
0 & 0 & \mu(1-p) & 0 & \nu & -(\nu + \mu(1-p))
\end{pmatrix}
\]
Writing $Q$ as a SAN

Use Kronecker notation to compress this.

- $A^{(1)} = \text{station 1}$, $A^{(2)} = \text{station 2}$.

- Separate local and synchronizing transitions.
  
  - Syn. Event 1 = departure from station 1 $\Rightarrow$ arrival to station 2

- Form $Q_l^{(1)}$ and $Q_l^{(2)}$.

- Form $Q_{e_1}$. 
Writing $Q$ as a SAN (cont.)

- 

\[
Q_{l}^{(1)} = \begin{pmatrix}
-\lambda & \lambda \\
\mu(1-p) & -\mu(1-p)
\end{pmatrix}, \quad Q_{l}^{(2)} = \begin{pmatrix}
0 & 0 & 0 \\
\nu & -\nu & 0 \\
0 & \nu & -\nu
\end{pmatrix}.
\]

- 

\[
Q_{e1} = \begin{pmatrix}
0 & 0 \\
1 & 0
\end{pmatrix} \otimes \begin{pmatrix}
0 & \mu p & 0 \\
0 & 0 & \mu p \\
0 & 0 & 0
\end{pmatrix} + \begin{pmatrix}
0 & 0 \\
0 & 1
\end{pmatrix} \otimes \begin{pmatrix}
-\mu p & 0 & 0 \\
0 & -\mu p & 0 \\
0 & 0 & 0
\end{pmatrix}.
\]

- We only need to store $Q_{l}^{(1)}$, $Q_{l}^{(2)}$, and the 4 matrices making up $Q_{e1}$. For very large MCs, this is a huge savings in storage. The Global $Q$ never needs to be formed or stored.

- Note that most of these small matrices are sparse.
Writing $Q$ as a SAN (cont.)

- SAN represents the global generator matrix as

$$Q = \oplus_{i=1}^{2} Q^{(i)} + \sum_{j=1}^{E} Q_{e_j}$$

$$= Q^{(1)}_l \otimes I_{n_2} + I_{n_1} \otimes Q^{(2)}_l + \sum_{j=1}^{E} Q_{e_j}$$

$$= \begin{pmatrix} -\lambda & 1 \\ \lambda & 0 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 0 & 0 & 0 \\ \nu & -\nu & 0 \end{pmatrix}$$

$$+ \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 & \mu p & 0 \\ 0 & 0 & \mu p \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} -\mu p & 0 & 0 \\ 0 & -\mu p & 0 \end{pmatrix}.$$
SAN descriptor

- MC has global $Q$ which can be stored in much more compact form as the ordinary sum of Kronecker products of much smaller matrices.

$$Q = \sum_{j=1}^{2E+N} \otimes_{i=1}^{N} Q_j^{(i)},$$

where

$N = \text{number of automata},$

$E = \text{number of synchronizing events}.$

- Observations:
  
  - Effect of synchronizing events
  
  - Effect of functional transitions
SANs provide powerful storage savings, but at what cost?

Recall GOAL: fast stationary analysis
Stationary analysis of a SAN: \( \pi \left( \sum_{j=1}^{2E+N} \bigotimes_{i=1}^{N} Q_j^{(i)} \right) = 0. \)

- Review of methods employed
  - Direct: \( L, U \) factors hard to get

- Problem: SANs ↓ storage, ↑ complexity \( \Rightarrow \) time til convergence=high.

- Solutions:
  - Preconditioners - Stewart, Buchholz, Chan
Existing SAN Preconditioners

- *ILU* Preconditioners are hard to employ.

- Stewart’s Neumann series inverse: $M$ approximates $Q^\#$.

$$M = \sum_{h=0}^{H} P^h.$$

- Buchholz’s preconditioner, similar to Stewart’s.

- Additive/Multiplicative Schwartz

- Diagonal: $M = D^{-1}$, where $D$ is the diagonal of $Q$.

Potential SAN Preconditioner: NKP ???
NKP Preconditioner for general matrix $R$

- Find $A, B$ so that $R \approx A \otimes B$.

- Precondition with $M = A^{-1} \otimes B^{-1}$.

- Pitsianis and Van Loan:

$$\min \| R - A \otimes B \|_F^2 = \| \tilde{R} - a \circ b \|_F^2.$$  

- requires $\tilde{R}$ and uses SVD.
The NKP, $A \otimes B$, when $R = \sum_{i=1}^{p} G_i \otimes F_i$

- It can be proven that when $R$ has this special structure, the optimal $A, B$ matrices are linear combinations of the matrices making up $R$. 

  \[
  A = \alpha_1 G_1 + \alpha_2 G_2 + \cdots + \alpha_p G_p, \quad B = \beta_1 F_1 + \beta_2 F_2 + \cdots + \beta_p F_p. 
  \]

- Original NKP problem can be written as NLP. Choose $\alpha_1, \alpha_2, \ldots, \alpha_p, \beta_1, \beta_2, \ldots, \beta_p$ so that this nonlinear function is minimized.

  \[
  \| R - A \otimes B \|^2_F = \| \sum_{i=1}^{p} G_i \otimes F_i - (\sum_{i=1}^{p} \alpha_i G_i) \otimes (\sum_{i=1}^{p} \beta_i F_i) \|^2_F. 
  \]
Back to SANs

- $Q = \sum_{j=1}^{2E+N} \otimes_{i=1}^{N} Q_j^{(i)}$ has special structure similar to $R = \sum_{i=1}^{p} G_i \otimes F_i$.

But $Q$ is Kronecker product of $N$ terms, not just 2 terms.

- SAN Problem: find $A, B, \ldots, N$ to min $\|Q - A \otimes B \otimes \cdots \otimes N\|_F^2$.

Precondition with $M = A^{-1} \otimes B^{-1} \otimes \cdots \otimes N^{-1}$.

- Questions:

  - Finding $A, B$ was based on the matrix SVD.

  Extension for finding $A, B, \ldots, N$?

  - When $R = \sum_{i=1}^{p} G_i \otimes F_i$, then $A = \sum_{i=1}^{p} \alpha_i G_i$, $B = \sum_{i=1}^{p} \beta_i F_i$.

  Extension to case when $R = Q = \sum_{j=1}^{2E+N} \otimes_{i=1}^{N} Q_j^{(i)}$?
The tensor is fundamental object of multilinear algebra.

- 1\textsuperscript{st}-order tensor is a vector.
- 2\textsuperscript{nd}-order tensor is a matrix.
- 3\textsuperscript{rd}-order tensor is a 3D box.

Operations on Tensors:

- inner product
- outer product
- scalar product
- tensor-matrix multiplication

Outer product of \( N \) vectors \((a \odot b \odot \cdots \odot n)\) results in an \( N^{th}\)-order tensor.
The HOSVD

- Tensors have rank as well as a SVD, called the HOSVD.
- A rank-1 $N^{th}$-order tensor is written as the outer product of $N$ vectors.
- HOSVD of $N^{th}$-order tensor $R$ shows that $R$ can be written as the sum of rank-1 tensors.

$$R = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} s_{i_1,i_2,\ldots,i_N} u_{i_1}^{(1)} \circ u_{i_2}^{(2)} \circ \cdots \circ u_{i_N}^{(N)}.$$

23
Multilinear Algebra and the NKP

- We want to min $\|R - A \otimes B \otimes \cdots \otimes N\|_F^2$ for a general matrix $R$.

- Following the 2D case, we need to show $\exists \tilde{R} \ni$

$$\|R - A \otimes B \otimes \cdots \otimes N\|_F^2 = \|\tilde{R} - a \circ b \circ \cdots \circ n\|_F^2.$$

- $a \circ b \circ \cdots \circ n$ is a rank-1 $N^{th}$-order tensor

$\Rightarrow$ rearrangement of $R$ ($\tilde{R}$) must be $N^{th}$-order tensor.

- Goal: approximate $N^{th}$-order tensor with rank-1 $N^{th}$-order tensor

$\Rightarrow$ use truncated HOSVD.

**Question:** How do we define this $N^{th}$-order tensor $\tilde{R}$?
Defining Rearrangement Operator for Higher Orders

- For $2^{nd}$-order tensor $R$, Pitsianis and Van Loan have already defined $\tilde{R} = R_2(R)$.

- The $3^{rd}$-order rearrangement operator, $R_3(R)$, uses the $2^{nd}$-order rearrangement operator.

- In general, we define the rearrangement operator recursively.

  We can then prove that with this recursive definition, the sums of squares match.
Example: Approximate $R_{12 \times 12}$ by $A_{2 \times 2}, B_{2 \times 2}, C_{3 \times 3}$.

$$R_{12 \times 12} = \begin{pmatrix} R_{1,1} & R_{1,2} \\ R_{2,1} & R_{2,2} \end{pmatrix}.$$

$$\tilde{R}_{4 \times 4 \times 9} = R_3(R) =$$

and then

$$\| R - A \otimes B \otimes C \|_F^2 = \| R_3(R) - a \circ b \circ c \|_F^2.$$
Approximating $N^{th}$-order tensor by a rank-1 $N^{th}$-order tensor

GOAL: \( \min \| R - A \otimes B \otimes \cdots \otimes N \|_F^2 = \min \| \tilde{R} - a \circ b \circ \cdots \circ n \|_F^2 \), where \( \tilde{R} \) is an $N^{th}$-order tensor and \( a \circ b \circ \cdots \circ n \) is a rank-1 $N^{th}$-order tensor.

- Truncated HOSVD is not the optimal rank-1 approximation to the $N^{th}$-order tensor \( \tilde{R} \).

- de Lathauwer:
  
  - truncated HOSVD is \textit{good} approximation.
  
  - HO power algorithm to find \textit{optimal} rank-1 approximation.

- We are only looking for an approximate inverse preconditioner.

  \( \Rightarrow \) try truncating the HOSVD. Let

  \[ a = s_{1,1,\ldots,1} u_1^{(1)}, b = u_1^{(2)}, \ldots, n = u_1^{(N)}. \]

  We hope \( A \otimes B \otimes \cdots \otimes N \) is a good approximation to \( R \).
Solving $\| \sum_{j=1}^{T} \otimes_{i=1}^{N} Q_{j}^{(i)} - A \otimes B \otimes \cdots \otimes N \|_{F}^2$, where $T = 2E + N$.

- We can show

$$A \approx \alpha_1 Q_1^{(1)} + \alpha_2 Q_2^{(1)} + \cdots + \alpha_T Q_T^{(1)}$$

$$B \approx \beta_1 Q_1^{(2)} + \beta_2 Q_2^{(2)} + \cdots + \beta_T Q_T^{(2)}$$

$$\vdots$$

$$N \approx \eta_1 Q_1^{(N)} + \eta_2 Q_2^{(N)} + \cdots + \eta_T Q_T^{(N)}.$$
Thus, just as in the optimal 2-matrix case, the original problem

$$\min \| Q - A \otimes B \otimes \cdots \otimes N \|^2_F$$

can be transformed into NLP.

$$\| Q - A \otimes B \otimes \cdots \otimes N \|^2_F \approx \| \sum_{j=1}^{T} \otimes_{i=1}^{N} Q^{(i)}_{j} - (\sum_{j=1}^{T} \alpha_{j} Q^{(1)}_{j}) \otimes (\sum_{j=1}^{T} \beta_{j} Q^{(2)}_{j}) \otimes \cdots \otimes (\sum_{j=1}^{T} \eta_{j} Q^{(N)}_{j}) \|^2_F$$

$$= \left[ \sum_{i=1}^{T} \sum_{j=1}^{T} \prod_{k=1}^{N} tr(Q^{(k)}_{i}^T Q^{(k)}_{j}) \right] - 2 \left( \sum_{i=1}^{T} \prod_{k=1}^{N} \left( \sum_{j=1}^{T} \alpha^{(k)}_{j} tr(Q^{(k)}_{i}^T Q^{(k)}_{j}) \right) \right)$$

where $\alpha_{j} = \alpha^{(1)}_{j}, \beta_{j} = \alpha^{(2)}_{j}, \ldots, \eta_{j} = \alpha^{(N)}_{j}$.

With this transformation, $\tilde{Q}$ and its HOSVD are never needed.

This is NLP of $NT$ variables.

As $N$ and $E \uparrow$, practicality of problem transformation $\downarrow$.

But, we can group automata!
Small artificial example with SAN structure

- $Q = \sum_{j=1}^{T} \otimes_{i=1}^{N} Q_{j}^{(i)}$, $N = 3$ and $T = 3$.

- So $Q = Q_{1}^{(1)} \otimes Q_{1}^{(2)} \otimes Q_{1}^{(3)} + Q_{2}^{(1)} \otimes Q_{2}^{(2)} \otimes Q_{2}^{(3)} + Q_{3}^{(1)} \otimes Q_{3}^{(2)} \otimes Q_{3}^{(3)}$.

- 9 matrices generated randomly.

- Generate NKP preconditioner $M = A^{-1} \otimes B^{-1} \otimes C^{-1}$ using NLP.

- Compare $Q$ with $A \otimes B \otimes C$.

- Compare $MQ$ with $I$.

(For rank $(n - 1)$ SANs, compare with $QQ^{\#} = I - e\pi$.)
Small artificial example with SAN structure (cont.)

- **Upper left block of \( Q \):**

\[
\begin{pmatrix}
3.6900 & 2.5575 & 1.1395 & 0.7423 & 4.4001 & 2.8771 & 3.3112 \\
0.6914 & 6.4420 & 0.5986 & 2.0307 & 0.9303 & 7.6925 & 0.5472 \\
2.0044 & 1.4823 & 5.0960 & 3.5434 & 5.1132 & 3.5434 & 1.8152 \\
0.4411 & 3.5066 & 0.3075 & 8.8268 & 0.4827 & 8.8753 & 0.4129 \\
3.9318 & 2.6607 & 3.1883 & 2.1301 & 0.5820 & 0.4889 & 3.6444 \\
0.2161 & 6.8078 & 0.6887 & 5.5756 & 0.4204 & 1.0499 & 0.1620 \\
1.4760 & 1.3742 & 0.7873 & 0.6904 & 1.9557 & 1.3306 & 0.7284
\end{pmatrix}
\]

- **Upper left block of \( A \otimes B \otimes C \):**

\[
\begin{pmatrix}
3.6841 & 2.5885 & 1.0927 & 0.7678 & 4.3038 & 3.0240 & 3.3548 \\
0.4798 & 6.4090 & 0.1423 & 1.9010 & 0.5605 & 7.4871 & 0.4369 \\
2.0423 & 1.4350 & 5.1304 & 3.6048 & 5.1320 & 3.6059 & 1.8597 \\
0.2660 & 3.5528 & 0.6681 & 8.9251 & 0.6683 & 8.9278 & 0.2422 \\
3.9243 & 2.7573 & 3.1395 & 2.2059 & 0.6045 & 0.4247 & 3.5735 \\
0.5111 & 6.8268 & 0.4089 & 5.4616 & 0.0787 & 1.0516 & 0.4654 \\
1.4098 & 0.9906 & 0.4182 & 0.2938 & 1.6470 & 1.1572 & 0.6928
\end{pmatrix}
\]
Small artificial example with SAN structure (cont.)

- Upper left block of $MQ$:

\[
\begin{pmatrix}
1.0920 & -0.0193 & -0.0947 & -0.0468 & -0.0727 & -0.0520 & 0.0201 \\
-0.0583 & 1.0465 & -0.0078 & -0.0976 & -0.0173 & -0.0828 & -0.0191 \\
0.0284 & 0.0121 & 1.1466 & -0.0201 & 0.0649 & -0.0247 & 0.0163 \\
0.0010 & 0.0283 & -0.0733 & 1.0891 & -0.0346 & 0.0389 & 0.0109 \\
-0.0331 & -0.0685 & 0.0113 & -0.0052 & 1.0676 & -0.0103 & -0.0337 \\
-0.0394 & -0.0597 & -0.0066 & 0.0064 & -0.0455 & 1.0317 & 0.0060 \\
-0.0171 & -0.0010 & -0.0126 & 0.0382 & -0.0166 & 0.0460 & 0.9939
\end{pmatrix}
\]
Thorough Testing on MCs and SANs

- MC Testing Motivation: Maybe NKP $M$ will be low-storage, efficient preconditioner and compete with ILU $M$.

- SAN Testing Motivation: Find good SAN preconditioner.
Findings from MC Testing (cont.)

- Perturbation away from Kronecker structure:

Table 1: Effect of perturbations away from exact Kronecker product

<table>
<thead>
<tr>
<th>$|E|_F$</th>
<th>$|Q - NKP|_F$</th>
<th>$\frac{\sigma_1}{\sum_{i=1}^{n} \sigma_i}$</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>.1726</td>
<td>.1571</td>
<td>.9999</td>
<td>succeeds</td>
</tr>
<tr>
<td>1.7263</td>
<td>1.4077</td>
<td>.9988</td>
<td>succeeds</td>
</tr>
<tr>
<td>5.1788</td>
<td>4.2848</td>
<td>.9965</td>
<td>succeeds</td>
</tr>
<tr>
<td>17.2627</td>
<td>14.8018</td>
<td>.9890</td>
<td>succeeds</td>
</tr>
<tr>
<td>51.9423</td>
<td>47.7126</td>
<td>.9676</td>
<td>succeeds</td>
</tr>
<tr>
<td>86.3134</td>
<td>74.8579</td>
<td>.9501</td>
<td>succeeds</td>
</tr>
<tr>
<td>127.2702</td>
<td>102.6748</td>
<td>.9346</td>
<td>fails</td>
</tr>
<tr>
<td>233.5594</td>
<td>201.3349</td>
<td>.8521</td>
<td>fails</td>
</tr>
</tbody>
</table>

- **Conclusion:** NKP $M$ for MCs takes too much work and gives no benefit UNLESS there is some inherent Kronecker structure.
Encouraging for SAN NKP $M$
Testing NKP $M$ on SANs

- Recall: Find $M = A^{-1} \otimes B^{-1} \otimes \cdots \otimes N^{-1}$ by NLP.

- Use $M$ as preconditioner on SAN system $\pi(I - M(\sum_{j=1}^{2E+N} \otimes_{i=1}^{N} Q_{j}^{(i)})) = 0$.

- Outline for remainder of SAN talk:
  - Example 5B
  - Summary graphs: BiCGSTAB on example 5B
NKP $M$ on SAN example 5B: Description

- Description: $N = 5$, $E = 4$. 5-server queueing network with respective capacities $C_1, C_2, C_3, C_4, C_5$.

- Size: $T = 13$, $NT = 65$.

- Diagram:

Figure 1: Example 5B: SAN queueing network with $N = 5$ automata
## NKP M on SAN example 5B: Results

Table 2: Number of iterations and CPU times for stationary analysis of 5-D queueing network, example 5B

<table>
<thead>
<tr>
<th>$C_1$</th>
<th>$C_2$</th>
<th>$C_3$</th>
<th>$C_4$</th>
<th>$C_5$</th>
<th>order</th>
<th>Preconditioner</th>
<th>Power</th>
<th>GMRES</th>
<th>BiCGSTAB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Iter.</td>
<td>Time</td>
<td>Iter.</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>108</td>
<td>None</td>
<td>144</td>
<td>66.38</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Neumann</td>
<td>48</td>
<td>67.15</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Indiv. Inv.</td>
<td>96</td>
<td>68.77</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Diagonal</td>
<td>261</td>
<td>123.34</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>NKP</td>
<td>57</td>
<td>29.18</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ILU0</td>
<td>25</td>
<td>11.68</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ILUTH</td>
<td>19</td>
<td>8.82</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>432</td>
<td>None</td>
<td>273</td>
<td>437.67</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Neumann</td>
<td>91</td>
<td>418.16</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Indiv. Inv.</td>
<td>156</td>
<td>378.43</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Diagonal</td>
<td>438</td>
<td>675.85</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>NKP</td>
<td>103</td>
<td>174.25</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ILU0</td>
<td>40</td>
<td>65.00</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ILUTH</td>
<td>37</td>
<td>57.60</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>1024</td>
<td>None</td>
<td>301</td>
<td>988.21</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Neumann</td>
<td>100</td>
<td>983.18</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Indiv. Inv.</td>
<td>173</td>
<td>884.69</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Diagonal</td>
<td>375</td>
<td>1257.41</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>NKP</td>
<td>123</td>
<td>463.85</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ILU0</td>
<td>49</td>
<td>168.73</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ILUTH</td>
<td>47</td>
<td>161.56</td>
<td>4</td>
</tr>
</tbody>
</table>
Summary Plots - BiCGSTAB (Example 5D-B)

Example 5D-B: BiCGSTAB number of iterations for various problem sizes

- None
- Neumann
- Indiv. Inv.
- Diagonal
- NKP
- ILU0
- ILUTH

Number of iterations vs. problem size graph.
Summary Plots - BiCGSTAB (Example 5D-B)

Example 5D–B: BiCGSTAB Time for various problem sizes
Verbal Summary of SAN NKP Tests

- NKP is best SAN preconditioner in terms of both storage and time.

  (ILU used for comparison purposes only.)

- Computation of NKP is worth effort.

  NKP computation $\leq 6$ sec.

- Example 5D is representative example. $T = 13$, $NT = 65$ are typical limits after grouping is done. Size of each $A^{(i)}$ does not affect NLP.
PART 2: Search Engines
Outline

- Introduction
- Various IR models and measures
- VSM
- LSI
- MCs and IR
- WWW IR
- Related fields
Overview

Data Mining – gleaning facts, trends, observations from large collection of data

EX: ○ minor/outlier clustering (stolen credit cards)
○ building classes of customer profiles
○ merging seemingly incompatible databases

Text Mining – mining document collections

EX: ○ determining authorities/hubs
○ determining synonyms, semantic structure
○ language translation

Information Retrieval
– retrieving documents in a collection most relevant to query
Some IR Models

- Boolean – EX: query = “Average Salary” and (US or Canada)

- Probabilistic

- Vector space models (VSM)

- Other:
  
  - Meta-search engines
  
  - Graph theoretic models – EX: WWW, node=doc, edge=hyperlink

\[ \text{Doc. } i \quad \text{hyperlink} \quad \text{Doc. } j \]
IR Performance Measures

• Recall: maximize # useful docs

\[
Recall = \frac{\text{# relevant docs retrieved}}{\text{# relevant docs in collection}}
\]

• Precision: minimize # useless docs

\[
Precision = \frac{\text{# relevant docs retrieved}}{\text{# docs retrieved}}
\]

• Other:
  - Time
  - Storage
  - User frustration
\[ \text{LA} + \text{IR} = \text{VSM} \]

- Term-by-Document Matrix \( A_{m \times n} \)

\[
A_{m \times n} = \begin{pmatrix}
1 & 0 & 0 & .20 & .65 & .50 & .90 \\
0 & 1 & .70 & .40 & .35 & .50 & .10 \\
0 & 0 & .30 & .40 & 0 & 0 & 0
\end{pmatrix}
\]

- Variety of weighting schemes—Most common = tf.idf

- Properties of \( A_{m \times n} \):

  * \( A \) can be huge.

  * \( A \) is sparse but otherwise unstructured.

  * \( A \) may contain a lot of uncertainty (noise) due to synonymy, polysemy, indexer bias.
• Query vector $q$

$$q = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}.$$ 

• GOAL: determine which $d_1, d_2, \ldots, d_n$ are closest to $q$. And do it FAST!
How to measure “closeness”?

- Length of vectors

- Similarity measures

- Angle measure between $q$ and $d_i$ for $i = 1 : n$.

  $$\cos(\theta_i) = \frac{q^T d_i}{\|q\| \|d_i\|}. $$

  Let $\delta_i = \cos(\theta_i)$. Retrieve doc. $i$ if $\delta_i \geq tol$. 
Problems with basic VSM

- size of $A$ – can we compress $A$?

- noise in $A$ – ambiguity in vocab. and writing style, polysems (bank), synonyms (car, automobile), indexing conventions (manual, automatic).

- Time for cosine calculations, $\cos(\theta_i) = \frac{q^T d_i}{\|q\| \|d_i\|}$.

  - one solution: remove denom. calculation by normalizing so that $\|q\| = \|d_i\| = 1$.

  - But $q^T d_i$ for $i = 1 : n$ still costly, despite parallelization.

    $q^T d_i$ is inner product of $m$-dimensional vectors, where $m$ is large, possibly $O(10^6)$.

- Need to: reduce “Noise” in $A$ and do cosine calculation FAST!
Data Compression and Noise Reduction with LSI

• $A$ has SVD, $A = U\Sigma V^T$.

• Use truncated SVD of $A$ to capture semantic essence of $A$ and drop noise.

\[
A \approx A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T = U_k \Sigma_k V_k^T.
\]

• Advantages:

  – $A_k$ requires much less storage.
  
  – $\cos(\theta_i)$ can be computed in projected space.

\[
\cos(\theta_i) = \frac{q^T d_i}{\|q\| \|d_i\|} = \frac{q^T (A e_i)}{\|q\| \|A e_i\|} \approx \frac{q^T (A_k e_i)}{\|q\| \|A_k e_i\|} = \frac{q^T U_k \Sigma_k V_k^T e_i}{\|q\| \|U_k \Sigma_k V_k^T e_i\|} = \frac{q^T U_k s_i}{\|q\| \|s_i\|},
\]

where $s_i = \Sigma_k V_k^T e_i$.

51
- $s_i$ and $\|s_i\|$ can be computed once and stored for all $i$. After normalization, only 1 inner product between $k$-dimensional vectors is required.

\[(k << m.)\]

- Latent semantic associations made.

• Disadvantages:

  - How to determine $k$? Empirical testing shows $k \leq 200$ works well.

  - Dynamic doc. collection – requires updating and downdating SVD.

  - Computing $A_k$, SVD, for large doc. collections is very costly,

    even if done only once a week at midnight.
Other Factorizations/Compression Techniques

- $ULV^T$

- SDD: $A_k = \sum_{i=1}^{k} \sigma_i x_i y_i^T$, where $x_i, y_i$ use only \{-1,0,1\}.

- GSVD

- Riemann SVD

- ADE

- Concept Decomposition Cluster

- DFT–FFT?

- Wavelets?
MCs + IR = PageRank concept

- Google’s PageRank concept:

  \[ \text{PageRank}(d_i) = \text{measure of importance of } d_i. \]

- Google uses Backlinks, we’ll use MC \( \pi \).

- Steps in Google search:
  1. enter query.
  2. direct search (full text scan) for group of docs containing query terms.
  3. sort by Google PageRank.

\[
\begin{bmatrix}
0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 \\
0 & 1/3 & 0 & 1/3 & 1/3 \\
0 & 0 & 1/2 & 0 & 1/2 \\
0 & 0 & 1/2 & 1/2 & 0 \\
\end{bmatrix}
\]
• Replace with MCPageRank.

\[
P = \begin{pmatrix}
0 & .9 & .1 & 0 & 0 \\
.9 & 0 & 0 & .1 & 0 \\
0 & .2 & 0 & .7 & .1 \\
0 & 0 & .5 & 0 & .5 \\
0 & 0 & .5 & .5 & 0
\end{pmatrix}
\]

• Google PageRank: involves dominant eigenvector computation.

• MC PageRank: involves finding \( \pi \).

\[\pi_i = \text{proportion of time random user accesses } d_i.\]

• Google: distributes probability of moving from \( d_i \) to \( d_j \) equally along all outgoing links. MC is more flexible, could use web log usage files to fill in \( p_{ij} \).
IR on the WEB

• Unique, Challenging Features:

  – dynamic and volatile (rapid updates, broken links)

  – immense and exponentially growing

  – hyperlinking structure (similar to citation structure)

  – real-time results (user patience remains fixed with viewing first 20 docs, while Web continues to grow ⇒ precision must increase in pace with Web growth.)

  – lack of editorial review process (errors, lack of structure, redundancy)

  – mercantile, proprietary (advertising, financial motives)
• Web stats:
  
  – estimated 3 billion webpages

  – 10K each

  – doubles every 18 months

• Google stats:

  – database of 1.5 billion documents

  – receives 150 million searches/day

  – 4,000 searches/sec. during peak times

  – uses 15,000 computers

• VSM and its variants–leader among IR techniques for small, well-controlled
doc collections. What is common Web IR technique?

  Direct search + post-sort by preassigned PageRank for each $d_i$. 

57
Future Work

- Combine LSI (semantic structure) with MC (link structure).

- Use SANs to cluster WWW MC and save storage.

- Use MFPT from MC derived from VSM-LSI model to find nearest neighbors.
Related Fields

- Information filtering
- Image retrieval
- Multimedia/Sound retrieval
- Multilingual/cross-language retrieval
The End