Text/Document Databases
What is a text database?

- User wants to find documents related related to a topic $T$.
- The search program tries to find the documents in the “document database” that contain the string $T$.
- This has two problems:

  1. **Synonymy**: Given a word $T$ (i.e. specifying a topic), the word $T$ does not occur anywhere in a document $D$, even though the document $D$ is in fact closely related to the topic $T$ in question.

  2. **Polysemy**: The same word may mean many different things in different contexts.

- Consider a text database that only indexes the following titles.

<table>
<thead>
<tr>
<th>DocumentID</th>
<th>String</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_1$</td>
<td>Jose Orojuelo’s Operations in Bosnia.</td>
</tr>
<tr>
<td>$d_2$</td>
<td>The Medelin Cartel’s Financial Organization.</td>
</tr>
<tr>
<td>$d_3$</td>
<td>The Cali Cartel’s Distribution Network.</td>
</tr>
<tr>
<td>$d_4$</td>
<td>Banking Operations and Money Laundering.</td>
</tr>
<tr>
<td>$d_5$</td>
<td>Profile of Hector Gomez.</td>
</tr>
<tr>
<td>$d_6$</td>
<td>Connections between Terrorism and Asian Dope Operations.</td>
</tr>
<tr>
<td>$d_7$</td>
<td>Hector Gomez’s: How he Gave Agents the Slip in Cali.</td>
</tr>
<tr>
<td>$d_8$</td>
<td>Sex, Drugs, and Videotape.</td>
</tr>
<tr>
<td>$d_9$</td>
<td>The Iranian Connection.</td>
</tr>
<tr>
<td>$d_{10}$</td>
<td>Boating and Drugs: Slips owned by the Cali Cartel.</td>
</tr>
</tbody>
</table>
Organization of this Topic

- Measures of performance of a text retrieval system.
- Latent Semantic Indexing
- Telescopic-Vector Trees for Document Retrieval.
Precision and Recall

- Suppose \( D \) is a finite set of documents.
- Suppose \( A \) is any algorithm that takes as input, a topic string \( T \), returns as output, a set \( (T) \) of documents.

  - **Precision** of algorithm \( A \) w.r.t. the predicate \textit{relevant} and test set \( D_{test} \) is \( P_t \%) \) for topic \( t \in T_{test} \) iff:
    \[
    P_t = 100 \times \frac{1 + \text{card}\{d \in D_{test} \mid d \in A(t) \land \text{relevant}(t,d) \text{ is true}\}}{1 + \text{card}\{d \in D_{test} \mid d \in A(t)\}}.
    \]
    (To avoid division by zero, we add one to both the numerator and denominator above). We say that the precision of algorithm \( A \) w.r.t. the predicate \textit{relevant}, the document test set \( D_{test} \) and the topic test set \( T_{test} \) is \( P\% \) iff:
    \[
    P = \frac{\sum_{t \in T_{test}} P_t}{\text{card}(T_{test})}.
    \]
- Precision basically says: How many of the answers returned are in fact correct.

  - **Recall** of an algorithm \( A \) is a measure of “how many” of the right documents are in fact retrieved by the query.

  - Recall \( R_t \) associated with a topic \( t \) is given by the formula:
    \[
    R_t = 100 \times \frac{1 + \text{card}\{d \in D_{test} \mid d \in A(t) \land \text{relevant}(t,d) \text{ is true}\}}{1 + \text{card}\{d \in D_{test} \mid \text{relevant}(t,d) \text{ is true}\}}.
    \]
The overall recall rate $R$ associated with test sets $D_{test}$ of documents, and $T_{test}$ of topics, is given by:

$$R = \frac{\sum_{t \in T_{test}} R_t}{\text{card}(T_{test})}.$$
Stop Lists, Word Stems, and Frequency Tables

- **Stop List:** This is a set of words that donot “discriminate” between the documents in a given archive.

- E.g. Cornell SMART system has about 440 words on its stop list.

- **Word Stems:** Many words are small syntactic variants of each other. For example, the words *drug, drugged, drugs*, are all similar in the sense that they share a common “stem”, viz. the word *drug*.

- Most document retrieval systems first eliminate words on stop lists and they also reduce words to their stems, before creating a frequency table.

- **Frequency Tables:** Suppose
  - $D$ is a set of $N$ documents, and
  - $T$ is a set of $M$ words/terms occurring in the documents of $D$.
  - Assume that no words on the stop list for $D$ occur in $T$, and that all words in $T$ have been stemmed.

The frequency table, $\text{Freq}_T$, associated with $D$ and $T$ is an $(M \times N)$ matrix such that $\text{Freq}_T(i, j)$ equals the number of occurrences of the word $t_i$ in document $d_j$. 

---

*V.S. Subrahmanian — All Rights Reserved*

These slides may not be duplicated without explicit written permission from Morgan Kaufmann Press.

*Principles of Multimedia Database Systems*  Morgan Kaufmann  Copyright ©1997
Example

<table>
<thead>
<tr>
<th>DocumentID</th>
<th>String</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_1$</td>
<td>Jose Orojuelo’s Operations in Bosnia.</td>
</tr>
<tr>
<td>$d_2$</td>
<td>The Medelin Cartel’s Financial Organization.</td>
</tr>
<tr>
<td>$d_3$</td>
<td>The Cali Cartel’s Distribution Network.</td>
</tr>
<tr>
<td>$d_4$</td>
<td>Banking Operations and Money Laundering.</td>
</tr>
<tr>
<td>$d_5$</td>
<td>Profile of Hector Gomez.</td>
</tr>
<tr>
<td>$d_6$</td>
<td>Connections between Terrorism and Asian Dope Operations.</td>
</tr>
<tr>
<td>$d_7$</td>
<td>Hector Gomez’s: How he Gave Agents the Slip in Cali.</td>
</tr>
<tr>
<td>$d_8$</td>
<td>Sex, Drugs, and Videotape.</td>
</tr>
<tr>
<td>$d_9$</td>
<td>The Iranian Connection.</td>
</tr>
<tr>
<td>$d_{10}$</td>
<td>Boating and Drugs: Slips owned by the Cali Cartel.</td>
</tr>
</tbody>
</table>

The associated frequency table might be given by:

<table>
<thead>
<tr>
<th>Term.Doc</th>
<th>$d_8$</th>
<th>$d_9$</th>
<th>$d_{10}$</th>
<th>$d_{11}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>sex</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>drug</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>videotape</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>iran</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>connection</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>boat</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>slip</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>own</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>cali</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>cartel</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Another example

Consider the frequency table shown below.

<table>
<thead>
<tr>
<th>Term/Doc</th>
<th>$d_1$</th>
<th>$d_2$</th>
<th>$d_3$</th>
<th>$d_4$</th>
<th>$d_5$</th>
<th>$d_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>615</td>
<td>390</td>
<td>10</td>
<td>10</td>
<td>18</td>
<td>65</td>
</tr>
<tr>
<td>$t_2$</td>
<td>15</td>
<td>4</td>
<td>76</td>
<td>217</td>
<td>91</td>
<td>816</td>
</tr>
<tr>
<td>$t_3$</td>
<td>2</td>
<td>8</td>
<td>815</td>
<td>142</td>
<td>765</td>
<td>1</td>
</tr>
<tr>
<td>$t_4$</td>
<td>312</td>
<td>511</td>
<td>677</td>
<td>11</td>
<td>711</td>
<td>2</td>
</tr>
<tr>
<td>$t_5$</td>
<td>45</td>
<td>33</td>
<td>516</td>
<td>64</td>
<td>491</td>
<td>59</td>
</tr>
</tbody>
</table>

- $d_1, d_2$ are similar because the distribution of the words in $d_1$ “mirrors” the distribution of the words for $d_2$.
- Both contain lots of occurrences of $t_1, t_4$ and relatively few occurrences of $t_2, t_3$, and moderately many occurrences of $t_5$.
- $d_3$ and $d_5$ are also similar.
- But $d_4, d_6$ stand out as sharply different.
- But is this enough?
- Merely counting words does not indicate the importance of the words, in the document. What about document lengths?
- Usually, a frequency table represents not just the number of occurrences of a (stemmed) word in a document, but also its important.
Queries

- User wants to execute the query: *Find the 25 documents that are maximally relevant w.r.t. banking operations and drugs?*

- Query $Q$ is trying to retrieve documents relevant to two keywords, which after “stemming” are:
  
  drug, bank.

- Think of the query $Q$ as a document. Thus, $Q$ is a column vector.

- We want to find the columns in $\text{FreqT}$ that are “as close” as possible to the vector associated with $Q$.

- Example closeness metrics include:

  1. **Term Distance**: Suppose $vec_Q(i)$ denotes the number of occurrences of term $t_i$ in $Q$. Then the term distance between $Q$ and document $d_r$ is given by:

     $$
     \sqrt{\sum_{j=1}^{M} (vec_Q(j) - \text{FreqT}(j, r))^2}.
     $$

     Of course, this is a rather arbitrary metric.

  2. **Cosine Distance**: This metric is used extensively in the document database world, and it may be described as follows:

     $$
     \frac{\sum_{j=1}^{M} (vec_Q(j) \times \text{FreqT}(j, r))}{\sqrt{\sum_{j=1}^{M} vec_Q(j)^2 \times \sum_{j=1}^{M} \text{FreqT}(j, r)^2}}.
     $$


• Complexity of retrievals may be $O(N \times M)$ which could be staggering in size.
Latent Semantic Indexing: The Basic Idea

- The number of documents \( M \) and the number of terms \( N \) is very large. For example, \( N \) could be over 10,000,000, as English words as well as proper nouns can be indexed.

- What LSI tries to do is to find a relatively small subset of \( K \) words which discriminate between the \( M \) documents in the archive.

- LSI is claimed to work effectively for around \( K = 200 \).

- Advantage: Each document is now a column vector of length 200, instead of length \( N \).

- This is obviously a big plus.

- Bottom line: How do we find a relatively small subset of \( K \) words which discriminate between the \( M \) documents in the archive.

- Use a technique called singular valued decomposition.

- 4-step approach used by LSI:
  1. (Table Creation) Create frequency matrix \( \text{Freq}_T \).
  2. (SVD Construction) Compute the singular valued decompositions, \((A, S, B)\) of \( \text{Freq}_T \).
  3. (Vector Identification) For each document \( d \), let \( \text{vec}(d) \) be the set of all terms in \( \text{Freq}_T \) whose corresponding rows have not been eliminated in the singular matrix \( S \).
4. **(Index Creation)** Store the set of all $vec(d)$’s, indexed by any one of a number of techniques (later we will discuss one such technique called a TV-tree).
Singular Valued Decompositions

- A matrix $\mathcal{M}$ is said to be of order $(m \times n)$ if it has $m$ rows and $n$ columns.

- If $\mathcal{M}_1, \mathcal{M}_2$ are matrices of order $(m_1 \times n_1)$ and $(m_2 \times n_2)$ respectively, then we say that the product, $(\mathcal{M}_1 \times \mathcal{M}_2)$, is well defined iff $n_1 = m_2$.

- If:

$$
\mathcal{M}_1 = \begin{pmatrix}
  a_1^1 & a_2^1 & \cdots & a_{m_1}^1 \\
  a_1^2 & a_2^2 & \cdots & a_{m_1}^2 \\
  \vdots & \vdots & \ddots & \vdots \\
  a_1^{n_1} & a_2^{n_1} & \cdots & a_{m_1}^{n_1}
\end{pmatrix}, \\
\mathcal{M}_2 = \begin{pmatrix}
  b_1^1 & b_2^1 & \cdots & b_{m_2}^1 \\
  b_1^2 & b_2^2 & \cdots & b_{m_2}^2 \\
  \vdots & \vdots & \ddots & \vdots \\
  b_1^{n_2} & b_2^{n_2} & \cdots & b_{m_2}^{n_2}
\end{pmatrix}
$$

then the product, $(\mathcal{M}_1 \times \mathcal{M}_2)$ is the matrix

$$
(\mathcal{M}_1 \times \mathcal{M}_2) = \begin{pmatrix}
  c_1^1 & c_2^1 & \cdots & c_{m_1}^1 \\
  c_1^2 & c_2^2 & \cdots & c_{m_1}^2 \\
  \vdots & \vdots & \ddots & \vdots \\
  c_1^{n_2} & c_2^{n_2} & \cdots & c_{m_1}^{n_2}
\end{pmatrix}
$$

where:

$$
c_j^i = \sum_{r=1}^{n_1} (a_r^i \times b_j^r).
$$

- EX:

$$
\begin{pmatrix}
  3 & 2 \\
  4 & 8
\end{pmatrix} \times \begin{pmatrix}
  1 & 4 & 3 \\
  2 & 4 & 6
\end{pmatrix} = \begin{pmatrix}
  7 & 20 & 21 \\
  20 & 48 & 60
\end{pmatrix}.
$$
SVD Continued

- Given a matrix $\mathcal{M}$ of order $(m \times n)$, the *transpose* of $\mathcal{M}$, denoted $\mathcal{M}^T$, is obtained by converting each row of $\mathcal{M}$, into a column of $\mathcal{M}^T$.

- EX:

\[
\begin{pmatrix}
7 & 20 & 21 \\
20 & 48 & 60
\end{pmatrix}^T
= \begin{pmatrix}
7 & 20 \\
20 & 48 \\
21 & 60
\end{pmatrix}.
\]

- Vector = matrix of order $(1 \times m)$.

- Two vectors $x, y$ of the same order are said to be *orthogonal* iff $x^T y = 0$.

- EX:

\[
x = (10, 5, 20).
\]
\[
y = (1, 2, -1).
\]

These two vectors are orthogonal because

\[
x^T y = \begin{pmatrix}
10 \\
5 \\
20
\end{pmatrix} \times \begin{pmatrix}
1 & 2 & -1
\end{pmatrix}
= \begin{pmatrix}
0 \\
0 \\
0
\end{pmatrix}.
\]
SVD Continued

- Matrix $\mathcal{M}$ is said to be **orthogonal** iff $(\mathcal{M}^T \times \mathcal{M})$ is the identity matrix (i.e. the matrix, all of whose entries are 1). For example, consider the matrix:

$$\mathcal{M} = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}.$$  

This matrix is orthogonal.

- Matrix $\mathcal{M}$ is said to be a **diagonal matrix** iff the order of $\mathcal{M}$ is $(m \times m)$ and for all $1 \leq i, j \leq m$, it is the case that:

$$i \neq j \rightarrow \mathcal{M}(i, j) = 0.$$  

- EX: $A$ and $B$ below are diagonal matrices, but $C$ is not:

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 5 \end{pmatrix}; \quad B = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}; \quad C = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}.$$  

- Diagonal matrix $\mathcal{M}$ of order $(m \times m)$ is said to be **non-decreasing** iff for all $1 \leq i, j \leq m$,

$$i \leq j \rightarrow \mathcal{M}(i, i) \leq \mathcal{M}(j, j).$$  

- Above, $A$ is a non-decreasing diagonal matrix, but $B$ is not.
SVD Continued

- A **Singular Value Decomposition** of $\text{FreqT}$ is a triple $(A, S, B)$ where:
  1. $\text{FreqT} = (A \times S \times B^T)$ and
  2. $A$ is an $(M \times M)$ orthogonal matrix such that $A^T A = I$ and
  3. $B$ is an $(N \times N)$ orthogonal matrix such that $B^T B = I$ and
  4. $S$ is a diagonal matrix called a **singular matrix**.

- Theorem: Given any matrix $\mathcal{M}$ of order $(m \times n)$, it is possible to find a singular value decomposition, $(A, S, B)$ of $\mathcal{M}$ such that $S$ is a **non-decreasing** diagonal matrix.

- EX: The SVD of the matrix

$$
\begin{pmatrix}
1.44 & 0.52 \\
0.92 & 1.44
\end{pmatrix}
$$

is given by:

$$
\begin{pmatrix}
0.6 & -0.8 \\
0.8 & 0.6
\end{pmatrix}
\begin{pmatrix}
5 & 0 \\
0 & 2
\end{pmatrix}
\begin{pmatrix}
0.8 & 0.6 \\
0.6 & -0.8
\end{pmatrix}.
$$

Here, the singular values are 5 and 2, and it is easy to see that the singular matrix is non-decreasing.
Returning to LSI

- Given a frequency matrix $\text{FreqT}$, we can decompose it into an SVD $\mathbf{T} \mathbf{S} \mathbf{D}^T$ where $\mathbf{S}$ is non-decreasing.

- If $\text{FreqT}$ is of size $(M \times N)$, then $\mathbf{T}$ is of size $(M \times M)$ and $\mathbf{S}$ is of order $(M \times R)$ where $R$ is the rank of $\text{FreqT}$, and $\mathbf{D}^T$ is of order $(R \times N)$.

- We can now “shrink” the problem substantially by eliminating the least significant singular values from the singular matrix $\mathbf{S}$.
LSI Continued

Shrinking the matrices is done as follows.

- Choose an integer $k$ that is substantially smaller than $R$.
- Replace $S$ by $S^*$, which is a $(k \times k)$ matrix, such that $S^*(i, j) = S(i, j)$ for $1 \leq i, j \leq k$.
- Replace the $(R \times N)$ matrix $\mathcal{D}^T$ by the $(k \times N)$ matrix $\mathcal{D}^{*T}$ where: $\mathcal{D}^{*T}(i, j) = \mathcal{D}^T(i, j)$ if $1 \leq i \leq k$ and $1 \leq j \leq N$.

Bottom line:

- Throw away the least significant values, and retain the rest of the matrix involved.
- Key claim in LSI is that if $k$ is chosen judiciously, then the $k$ rows appearing in the singular matrix $S^*$ represent the $k$ “most important” (from the point of view of retrieval) terms occurring in the entire document collection.
Suppose $\text{Freq}_T$ has the SVD

$$
\begin{pmatrix}
  a_1 & a_2 & a_3 & a_4 & a_5 \\
  a_2^2 & a_2 & a_3 & a_4^2 & a_5^2 \\
  \cdots & \cdots & \cdots & \cdots & \cdots \\
  a_M^1 & a_M^2 & a_M^3 & a_M^4 & a_M^5
\end{pmatrix}
\begin{pmatrix}
  20 & 0 & 0 & 0 & 0 \\
  0 & 16 & 0 & 0 & 0 \\
  0 & 0 & 12 & 0 & 0 \\
  0 & 0 & 0 & 0.08 & 0 \\
  0 & 0 & 0 & 0 & 0.004
\end{pmatrix}
\begin{pmatrix}
  b_1^1 & b_2^1 & b_3^1 & \cdots & b_N^1 \\
  b_1^2 & b_2^2 & b_3^2 & \cdots & b_N^2 \\
  \cdots & \cdots & \cdots & \cdots & \cdots \\
  b_1^5 & b_2^5 & b_3^5 & \cdots & b_N^5
\end{pmatrix}
$$

- If we set 3 as the threshold, then we obtain the following result (after eliminating the 4th and 5th singular values).

$$
\begin{pmatrix}
  a_1 & a_2 & a_3 \\
  a_2^2 & a_2 & a_3 \\
  \cdots & \cdots & \cdots \\
  a_M^1 & a_M^2 & a_M^3
\end{pmatrix}
\begin{pmatrix}
  20 & 0 & 0 \\
  0 & 16 & 0 \\
  0 & 0 & 12
\end{pmatrix}
\begin{pmatrix}
  b_1^1 & b_2^1 & b_3^1 & \cdots & b_N^1 \\
  b_1^2 & b_2^2 & b_3^2 & \cdots & b_N^2 \\
  b_1^3 & b_2^3 & b_3^3 & \cdots & b_N^3
\end{pmatrix}
$$
Analysis

• Usually, $R$ is taken to be 200.

• The size of the original frequency table is $(M \times N)$ where $M$ is the number of terms, and $N$ is the number of documents. We may easily have $M = 1,000,000$ and $N = 10,000$, even for just a small document database such as that consisting of the University of Maryland’s Computer Science technical reports.

• The size of the three matrices, after we have reduced the size of the singular matrix to, say 200, is:
  
  - The first matrix’s size is $M \times R$. With the above numbers, this is $1000000 \times 200 = 200,000,000$.
  - The singular matrix’s size is $200 \times 200 = 400,000$. (In fact, of these 400,000 entries, only 200 at most need to be stored, as all other entries are zero).
  - The last matrix’s size is $R \times N$. With the above numbers, this is $200 \times 10000$.

Adding up the above, we get a total of 202,000,200 entries in the tables after SVD’s are applied. This is approximately 200 million.

• In contrast, $(M \times N)$ is close to 10,000-million: in other words, the SVD trick reduced the space utilized to about $\frac{1}{50}$‘th of that required by the original frequency table.
LSI: Document Retrieval using SVDs

Two questions:

1. Given two documents $d_1, d_2$ in the archive, how “similar” are they?
2. Given a query string/document $Q$, what are the $n$ documents in the archive that are “most relevant” for the query?

- Suppose $\mathbf{x} = (x_1, \ldots, x_w)$ and $\mathbf{y} = (y_1, \ldots, y_w)$.
- Dot product of $\mathbf{x}$ and $\mathbf{y}$, denoted $\mathbf{x} \odot \mathbf{y}$ is given by
  \[
  \mathbf{x} \odot \mathbf{y} = \sum_{i=1}^{w} x_i \times y_i.
  \]
- The similarity of these two documents w.r.t. the SVD representation, $TS^* \times D^{*T}$, of a frequency table is given by computing the dot product of the two columns in the matrix $D^{*T}$ associated with these two documents.
  \[
  \sum_{z=1}^{R} D^{*T}[i, z] \times D^{*T}[j, z].
  \]
- When we are asked to find the top $p$ matches for $Q$, we are trying to find $p$ documents $d_{\alpha(1)}, \ldots, d_{\alpha(p)}$ such that:
  1. for all $1 \leq i \leq j \leq p$, the similarity between $\text{vec}_Q$ and $d_{\alpha(i)}$ is greater than or equal to the similarity between $\text{vec}_Q$ and $d_{\alpha(j)}$. 

...
2. There is no other document $d_z$ such that the similarity between $d_z$ and $vec_Q$ exceeds that of $d_{\alpha(p)}$.

- This can be done by using any generalized, high dimensional data structure that supports nearest neighbor searches.
Telescopic Vector (TV) Trees

- Access to point data in very large dimensional spaces should be highly efficient.
- A document $d$ may be viewed as a vector $\tilde{d}$ of length $k$, where the singular valued matrix, after decomposition, is of size $(k \times k)$.
- Thus, each document may be thought of as a point in a $k$-dimensional space.
- A document database may be thought of as a collection of such points, indexed appropriately.
- When a user $u$ presents a query $Q$, s/he is in effect specifying, a vector $\text{vec}(Q)$ of length $k$. We must find the $p$ documents in the database that are maximally relevant to $Q$.
- This boils down to attempting to find the $k$-nearest neighbors present in the document database, of the query $Q$.
- The TV-tree is a data structure that borrows from R-trees in this effort.
- The TV-tree attempts to dynamically and flexibly decide how to branch, based on the data that is being examined. If lots of vectors all agree on certain attributes (e.g. if lots of documents all have many common terms), then we must organize our index by branching on those terms (i.e. fields of the vectors) that distinguish between these vectors/documents.
Organization of a TV-Tree

- **NumChild**: This is the maximal number of children that any node in the TV-tree is allowed to have.

- **α**: α is a number, greater than 0 and less than k, called the *number of active dimensions*.

- **TV(k, NumChild, α)** denotes TV-tree used to store k-dimensional data with **NumChild** as the maximal number of children, and α as the number of active dimensions.

- Each node in a TV-tree represents a region. For this purpose, each node **N** in a TV-tree contains three fields:
  
  1. **N.Center**: This represents a point in k-dimensional space.
  2. **N.Radius**: This is a real number greater than 0.
  3. **N.ActiveDims**: This is a list of at most α dimensions. Each of these dimensions is a number between 1 and k. Thus, **N.ActiveDims** is a subset of \{1, \ldots, k\} of cardinality α or less,
Region associated with a node $N$

- Suppose $\mathbf{x}$ and $\mathbf{y}$ are points in $k$-dimensional space, and $Active\ Dims$ is some set of active dimensions. The active distance between $\mathbf{x}$ and $\mathbf{y}$, denoted $\text{act\_dist}(\mathbf{x}, \mathbf{y})$ is given by:

$$\text{act\_dist}(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i \in Active\ Dims} x_i^2 - y_i^2}.$$  

Here, $x_i$ and $y_i$ denote the value of the $i$’th dimension of $\mathbf{x}$ and $\mathbf{y}$, respectively.

- EX: $k = 200$ and $\alpha = 5$ and the set $Active\ Dims = \{1, 2, 3, 4, 5\}$. Suppose:

$$\mathbf{x} = (10, 5, 11, 13, 7, x_6, x_7, \ldots, x_{200}).$$
$$\mathbf{y} = (2, 4, 14, 8, 6, y_6, y_7, \ldots, y_{200}).$$

Then the active distance between $\mathbf{x}$ and $\mathbf{y}$ is given by:

$$\text{act\_dist}(\mathbf{x}, \mathbf{y}) = \sqrt{(10 - 2)^2 + (5 - 4)^2 + (11 - 14)^2 + (13 - 8)^2 + (7 -}$$
$$= \sqrt{100}$$
$$= 10.$$

- Node $N$ represents the region containing all points $\mathbf{x}$ such that the active distance (w.r.t. the active dimensions in $N.\ Active\ Dims$) between $\mathbf{x}$ and $N.Center$ is less than or equal to $N.Radius$. 

---

V.S. Subrahmanian — All Rights Reserved

These slides may not be duplicated without explicit written permission from Morgan Kaufmann Press.

Principles of Multimedia Database Systems  Morgan Kaufmann  Copyright ©1997
• EX: If we had a node $N$ with its center at

$$N.\text{Center} = (10, 5, 11, 13, 7, 0, 0, 0, \ldots, 0)$$

and $N.\text{ActiveDims} = \{1, 2, 3, 4, 5\}$, then this node represents the region consisting of all points $\mathbf{x}$ such that:

$$\sqrt{(x_1 - 10)^2 + (x_2 - 5)^2 + (x - 11)^2 + (x_4 - 13)^2 + (x_5 - 7)^2} \leq N.\text{Rad}$$

We use the notation $\text{Region}(N)$ to denote the region represented by a node $N$ in a TV-tree.

• $N$ also contains an array, $\text{Child}$ of $\text{NumChild}$ pointers to other nodes of the same type.
Properties of TV-Trees

- All data is stored at the leaf nodes;
- Each node in a TV-tree (except for the root and the leaves) must be at least half full, i.e. at least half the Child pointers must be non-NIL.
- If $N$ is a node, and $N_1, \ldots, N_r$ are all its children, then

$$Region(N) = \bigcup_{i=1}^{r} Region(N_i).$$
Insertion into TV-trees

There are three key steps.

1. **Branch Selection:** The first operation is called branch selection. When we insert a new vector into the TV-tree, and we are at node \( N \) (with children \( N_i \), for \( 1 \leq i \leq \text{NumChild} \)), we need to determine which of these children to insert the key into.

2. **Splitting:** The second approach is what to do, when we are at a leaf node that is full and cannot accommodate the vector \( v \) we are inserting. This causes a split in that node.

3. **Telescoping:** Suppose a node \( N \) is split into subnodes \( N_1, N_2 \). In this case, it may well turn out that the vectors in \( \text{Region}(N_1) \) all agree on not just the active dimensions of the parent \( N \), but a few more as well. The addition of these extra dimensions is called *telescoping*. Telescoping may also involve the *removal* of some active dimensions, as we shall see later.
Example

- 5-dimensional space.
- $TV(5, 3, 2)$.
- Space is a sphere centered at $(0, 0, 0, 0, 0)$ with radius 50.
- Initially, tree is empty.
- Insert $(5, 3, 20, 1, 5)$. This is handled straightforwardly by the creation of a root node with
  1. $Root.Center = (0, 0, 0, 0, 0)$;
  3. In this case, the root is also a leaf, with a pointer to the information relevant to the point $v_1 = (5, 3, 20, 1, 5)$.

See (a).

- Insert $v_2 = (0, 0, 18, 42, 4)$.
- Insert $v_3 = (0, 0, 19, 39, 6)$. At this stage, the root is “full”.
- Insert $v_4 = (9, 10, 2, 0, 16)$.

  1. We must split the root.
  2. Take the four vectors involved and “group” them together into two groups, say. $v_1, v_4$ and $v_2, v_3$. See figure (d).
  3. Insert $v_5 = (18, 5, 27, 9, 9)$. Branch selection needed to determine how to branch. See Figure 2(a).
4. Insert $v_6 = (0, 0, 29, 0, 3)$. Again, we must perform branch selection, and this time, we may choose to branch right, as shown in Figure 2(b).
Figure 1

(a) 

(b) 

(c) 

(d) 

LEGEND

Center  R  ActiveD
child1  child2  child3
Figure 2
Branch Selection

- Consider node $N$ with $1 \leq j \leq \text{NumChild}$ children, denoted $N_1, \ldots, N_{\text{NumChild}}$.

- $\text{exp}_j(v)$ denotes the amount we must expand $N_j.Radius$ so that $v$’s active distance from $N_j.Center$ falls within this radius.

- First, select all $j$’s such that $\text{exp}_j(v)$ is minimized.
  
  EX: if we have nodes $N_1, \ldots, N_5$ with $exp$ values 10, 40, 19, 10, 32 respectively, the two candidates selected for possible insertion are $N_1$ and $N_4$. If a tie occurs, as in the above case, pick the node such that the distance from the center of that node to $v$ is minimized.
Splitting

- When we attempt to insert a vector \( \mathbf{v} \) into a leaf node \( N \) that is already full, then we need to split the node.

- We must create subnodes \( N_1, N_2 \), and each vector in node \( N \) must fall into one of the regions represented by these two subnodes.

- Split vectors in leaf \( N \) into two groups \( (G_1, H_1) \).

- We may be able to enclose all vectors in \( G_1 \) within a region with center \( c_1 \) and radius \( r_1 \) and all vectors in \( H_1 \) within a region with center \( c_2 \) and radius \( r_2 \).

- Many such splits are possible in general.

- Split \( (G_1, H_1) \) is finer than split \( (G_2, H_2) \) iff the sums of the radii, \( (r_1 + c'_1) \) is smaller than the sum of the radii \( (r_2 + r'_2) \).

- Still not enough to uniquely identify a “finest” split.

- If \( (G_1, H_1) \) and \( (G_2, H_2) \) are splits such that neither is finer than the other and no other split is finer than each of them, then we say that \( (G_1, H_1) \) is more conservative than \( (G_2, H_2) \) iff

\[
r_1 + r'_1 - \text{act\_dist}(c_1, c'_1) \leq r_2 + r'_2 - \text{act\_dist}(c_2, c'_2).
\]

- Split \( (G, H) \) is the selected split iff:

1. there is no split \( (G', H') \) that is finer than \( (G, H) \) and
2. there is no split \((G', H')\) that satisfies condition (1) (i.e. there exists no split \((G^*, H^*)\) finer than \((G', H')\)) and that is more conservative than \((G, H)\).
Suppose $N$ is the node into which we are to insert a vector $v$.

Insertion of $v$ may cause two types of changes to $N$: it may cause $N$ to be split into two subnodes $N_1, N_2$, or it may “modify” the set of active dimensions of node $N$ (e.g. if vector $v$ does not agree, on the active dimensions, with other vectors stored at node $N$).

When node $N$ gets split into two sub-nodes $N_1, N_2$, the set of vectors at either node $N_1$ or node $N_2$ (but not both) must be a subset of the set of vectors at node $N$ before the insertion.

Suppose $N_1$ has this property.

The vectors in $N_1$ may agree not only on the active dimensions of $N$, but on some other dimensions as well. In this case, we can *expand* the set of active dimensions of node $N$, by adding these new dimensions. See figure below.

- The other case when telescoping occurs is when a vector is added to a node, $N$, but no split occurs. If $N$ originally...
contained the vectors \( \mathbf{v}_1, \ldots, \mathbf{v}_r \) and \( \mathbf{v} \) is the vector being added, even though vectors \( \mathbf{v}_1, \ldots, \mathbf{v}_r \) originally agreed on the active dimensions \( d_1, \ldots, d_s \) of node \( N \), they only agree now on a subset (for example, \( d_2, \ldots, d_s \)) and hence, the set of active dimensions of node \( N \) must be *contracted* to reflect this fact.
Searching in TV-Trees

Algorithm 2 Search(T,v)

if Leaf(T) then Return (T.Center = v); Halt 
else
  if v ∈ Region(T) then
    Return ∩_{i=1}^{NumChild} Search(T.Child[i], v)
  
end
Nearest Neighbor Retrievals in TV-Trees

\[
\begin{align*}
\min(N, v) &= \begin{cases} 
0 & \text{if } v \in \text{Region}(N) \\
\text{act}_\text{dist}(v, N.\text{Center}) - N.\text{Radius} & \text{otherwise}
\end{cases} \\
\max(N, v) &= \text{act}_\text{dist}(v, N.\text{Center}) + N.\text{Radius}.
\end{align*}
\]

- Maintain an array \( SOL \) of length \( p \), i.e. with indices running 1 through \( p \).
- The algorithm \( NNSearch \) uses a routine called \( Insert \) that takes as input, a vector \( vec \), and an array \( SOL \) maintained in non-descending order of active distance from \( vec \).
- \( Insert \) returns as output, the array \( SOL \) with \( vec \) inserted in it, at the right place, and with the \( p \)'th element of \( SOL \) eliminated.
Nearest Neighbor Retrievals in TV-Trees (Contd.)

Algorithm 3  NNSearch(T,v,p)
  for i = 1 to p do SOL[i] = ∞;
     NNSearch1(T,v,p);
end (* end of program NNSearch *)

procedure NNSearch1(T,v,p);
if Leaf(T) & act.dist(T.val,v) < SOL[p] then
  Insert T.val into SOL;
else
  
  if Leaf(T) then r = 0;
  else 
    Let N₁, ..., N_r be the children of T;
    Order the N_i’s in ascending order w.r.t. min(N_i,v);
    Let Nₙ(1), ..., Nₙ(r) be the resulting order;
  
  done = false; i = 1;
  while ((i ≤ r) ∧ ¬done) do
    
    NNSearch(Nₙ(i),v,p);
    if SOL[p] < min(Nₙ(i+1),v) then
      done = true;
      i = i + 1;
    
  }; (* end of while *)
} (* end of else *)
Return SOL;
end proc (* end of subroutine NNSearch1 *)
Other Retrieval Techniques: Inverted Indices

- A document record contains two fields – a `doc_id` and a `postings_list`.
- Postings list is a list of terms (or pointers to terms) that occur in the document. Sorted using a suitable relevance measure.
- A term record consists of two similar fields: a `term` field (string), and a `postings_list`.
- Two hash tables are maintained: a `DocTable` and a `TermTable`. The `DocTable` is constructed by hashing on the `doc_id` key, while the `TermTable` is obtained by hashing on the `term` key.
- To find all documents associated with a term, we merely return the postings list.
- Used in many commercial systems such as MEDLARS and DIALOG.
Other Retrieval Techniques: Signature Files

- Associate a signature $s_d$, with each document $d$.
- A signature is a representation of an ordered list of terms that describe the document.
- The list of terms from which $s_d$ is derived is obtained after performing a frequency analysis, stemming, and usage of stop lists.
- If signature list consists of the ordered list of words $w_1, w_2, \ldots, w_r$.
- This means that word $w_1$ is most important when describing the document, word $w_2$ is second most important, and so forth.
- Signature of the document $d$ is a bit-representation of this list, usually obtained by encoding the list after using hashing, and then superimposing a coding scheme.