Relational Approach

(COSC 488)

Nazli Goharian
nazli@cs.georgetown.edu

Slides are mostly based on Information Retrieval Algorithms and Heuristics, Grossman & Frieder

Problem Definition

• Three conceptual data types:
  – Structured:
    • Objective (absolute) correctness; perfect accuracy; efficiency only issue (e.g.; relational database)
  – Unstructured:
    • Subjective (relative) correctness; accuracy & efficiency trade-off (e.g.; documents, images, video, sound).
  – Semi-structured:
    • Combination of structured and unstructured (e.g.; XML)

• Problem:
  – Individually querying each type & merging answers.

• A Solution:
  – IR as an application of Relational Database Management System (RDBMS) – with typical IR Functionalities:
    • Boolean query
    • Relevance Ranking – multiple similarity measures
    • Proximity Search
Relational Inverted Index

All inverted index entries

\[ \text{term} \rightarrow \text{list of documents} \]

*e.g.*, \( \text{vehicle} \rightarrow \text{D1, D3, D4} \) results in:

<table>
<thead>
<tr>
<th>term</th>
<th>docID</th>
</tr>
</thead>
<tbody>
<tr>
<td>vehicle</td>
<td>D1</td>
</tr>
<tr>
<td>vehicle</td>
<td>D3</td>
</tr>
<tr>
<td>vehicle</td>
<td>D4</td>
</tr>
</tbody>
</table>

Text Retrieval Conference (TREC) Sample Document

<DOC>
<DOCNO> AP881214-0028 </DOCNO>
<FILEID>AP-NR-12-14-88 0117EST </FILEID>
<FIRST>u i BC-Japan-Stocks 12-14 0027 </FIRST>
<SECOND>BC-Japan-Stocks,0026 </SECOND>
<HEAD>Stocks Up In Tokyo </HEAD>
<DATELINE>TOKYO (AP) </DATELINE>
<TEXT>
The Nikkei Stock Average closed at 29,754.73 points up 156.92 points on the Tokyo Stock Exchange Wednesday. 
</TEXT>
</DOC>
# Relational Document Representation

<table>
<thead>
<tr>
<th>DOcID</th>
<th>docname</th>
<th>headline</th>
<th>dateline</th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>AP881214-0028</td>
<td>Stocks Up In Tokyo</td>
<td>TOKYO (AP)</td>
</tr>
</tbody>
</table>

## INDEX

<table>
<thead>
<tr>
<th>docID</th>
<th>termcnt</th>
<th>term</th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>1</td>
<td>nikkei</td>
</tr>
<tr>
<td>28</td>
<td>2</td>
<td>stock</td>
</tr>
<tr>
<td>28</td>
<td>1</td>
<td>average</td>
</tr>
<tr>
<td>28</td>
<td>1</td>
<td>closed</td>
</tr>
<tr>
<td>28</td>
<td>2</td>
<td>points</td>
</tr>
<tr>
<td>28</td>
<td>1</td>
<td>up</td>
</tr>
<tr>
<td>28</td>
<td>1</td>
<td>tokyo</td>
</tr>
<tr>
<td>28</td>
<td>1</td>
<td>exchange</td>
</tr>
<tr>
<td>28</td>
<td>1</td>
<td>wednesday</td>
</tr>
</tbody>
</table>

## TERM

<table>
<thead>
<tr>
<th>term</th>
<th>idf</th>
</tr>
</thead>
<tbody>
<tr>
<td>average</td>
<td>1.08</td>
</tr>
<tr>
<td>closed</td>
<td>1.08</td>
</tr>
<tr>
<td>exchange</td>
<td>1.00</td>
</tr>
<tr>
<td>nikkei</td>
<td>2.07</td>
</tr>
<tr>
<td>points</td>
<td>1.23</td>
</tr>
<tr>
<td>stock</td>
<td>1.00</td>
</tr>
<tr>
<td>tokyo</td>
<td>1.58</td>
</tr>
<tr>
<td>up</td>
<td>0.30</td>
</tr>
<tr>
<td>wednesday</td>
<td>0.60</td>
</tr>
</tbody>
</table>

---

## Simplistic Models:

### Keyword and Boolean Searches
Relational Approach:

**Keyword Search**

• Keyword search

```sql
select i.docID
from INDEX i, QUERY q
where i.term = q.term
```

• Keyword search with stop word list

```sql
select i.docID
from INDEX i, QUERY q, STOPLIST s
where (i.term = q.term) and (i.term <> s.term)
```

---

Relational Approach:

**Boolean Search OR query**

```sql
select docID
from INDEX
where term = term1
union
select docID
from INDEX
where term = term2
union
select docID
from INDEX
where term = term3
union
....
select docID
from INDEX
where term = termN
```

© David Grossman & Ophir Frieder
Relational Approach:

Boolean Search AND query

```
select docID from INDEX where term = term1 intersect select docID from INDEX where term = term2 intersect select docID from INDEX where term = term3 .... intersect select docID from INDEX where term = termN
```

Fixed Join-Count AND Queries

Find all documents that contain all of the terms found in the QUERY relation:

```
select i.docID from INDEX i, QUERY q where i.term = q.term group by i.docID having count (distinct (i.term)) = select count(distinct (term)) from QUERY
```
TAND Queries

Find all documents that contain at least $X$ of the terms found in the QUERY relation:

```sql
select i.docID
  from INDEX i, QUERY q
where i.term = q.term
group by i.docID
having count (distinct (i.term)) >= X
```

Relevance Ranking:

Vector Space and Probabilistic Models
### Relational Document Representation

**Single Term Processing**

<table>
<thead>
<tr>
<th>docID</th>
<th>docname</th>
<th>headline</th>
<th>dateline</th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>AP881214-0028</td>
<td>Stocks Up In Tokyo</td>
<td>TOKYO (AP)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Index</th>
<th>Term Count</th>
<th>Term</th>
<th>Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>1</td>
<td>nikkei</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>2</td>
<td>stock</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>1</td>
<td>average</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>1</td>
<td>closed</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>2</td>
<td>points</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>1</td>
<td>up</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>1</td>
<td>tokyo</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>1</td>
<td>exchange</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>1</td>
<td>wednesday</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Term</th>
<th>IDF</th>
</tr>
</thead>
<tbody>
<tr>
<td>average</td>
<td>1.08</td>
</tr>
<tr>
<td>closed</td>
<td>1.08</td>
</tr>
<tr>
<td>exchange</td>
<td>1.00</td>
</tr>
<tr>
<td>nikkei</td>
<td>2.07</td>
</tr>
<tr>
<td>points</td>
<td>1.23</td>
</tr>
<tr>
<td>stock</td>
<td>1.00</td>
</tr>
<tr>
<td>tokyo</td>
<td>1.58</td>
</tr>
<tr>
<td>up</td>
<td>0.30</td>
</tr>
<tr>
<td>wednesday</td>
<td>0.60</td>
</tr>
</tbody>
</table>

### Relational Query Representation

**Original Query:**
```
“nikkei stock exchange
american stock exchange”
```

**Query**

<table>
<thead>
<tr>
<th>Term</th>
<th>TF</th>
</tr>
</thead>
<tbody>
<tr>
<td>nikkei</td>
<td>1</td>
</tr>
<tr>
<td>stock</td>
<td>2</td>
</tr>
<tr>
<td>exchange</td>
<td>2</td>
</tr>
<tr>
<td>american</td>
<td>1</td>
</tr>
</tbody>
</table>

© David Grossman & Ophir Frieder
Vector Space Model

- Term Frequency ($tf_{ik}$):
  - number of occurrences of term $t_k$ in document $i$
- Document Frequency ($df_j$):
  - number of documents which contain $t_j$
- Inverse Document Frequency ($idf_j$):
  - $\log(d/df_j)$ where $d$ is the total number of documents

Notes:
- $idf$ is a measure of uniqueness of a term across the collection
- $tf$ is the frequency of a term in a given document

Similarity Coefficients

- Several similarity coefficients based on the query vector $X$ and the document vector $Y$ are defined:

  Inner Product $\sum_{i=1}^{t} x_i \cdot y_i$

  Cosine Coefficient $\frac{\sum_{i=1}^{t} x_i y_i}{\sqrt{\sum_{i=1}^{t} x_i^2 \cdot \sum_{i=1}^{t} y_i^2}}$
Relevance Ranking:
SQL for VSM dot product

List all documents in the order of their similarity coefficient where the coefficient is computed using the dot product.

\[(\text{Query Weight} \times \text{Document Weight})\]

SQL:
```
SELECT d.docID, d.docname, SUM(q.termcnt * t.tfidf * i.termcnt * t.tfidf)
FROM QUERY q, INDEX i, TERM t, DOCUMENT d
WHERE q.term = i.term AND
q.term = t.term AND
i.docID = d.docID
GROUP BY d.docID, docname
ORDER BY 3 DESC
```

© David Grossman & Ophir Frieder

Relevance Ranking:
SQL for Probabilistic Retrieval

\[
\sum_{i=1}^{\text{num_terms}} \log \left( \frac{(\text{numdocs} - df_i) + .5}{(df_i + .5)} \right) \times \frac{2.2 \times i.tf}{.3 + (.75 \times \text{doclength} / \text{avgdoclength}) + t.tfidf} \times q.tf
\]

SELECT d.docID, d.docname, SUM(
    LOG((numDocs - t.df) / (t.df + 0.5)) * 
    ((2.2*i.tf) / (.3 + (.75 * d.DocLen)/AvgDocLen) + i.tf)) * q.termcnt 
FROM INDEX i, TERM t, DOCUMENT d, QUERY q
WHERE i.term = t.term
AND i.docID = d.docID
AND  t.term = q.term
GROUP BY d.docID, d.docname
ORDER BY 3;

© David Grossman & Ophir Frieder
Relational Document Representation  
(Phrase Processing)

**DOCUMENT**

<table>
<thead>
<tr>
<th>docID</th>
<th>docname</th>
<th>headline</th>
<th>dateline</th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>AP881214-0028</td>
<td>Stocks Up In Tokyo</td>
<td>TOKYO (AP)</td>
</tr>
</tbody>
</table>

**INDEX**

<table>
<thead>
<tr>
<th>docID</th>
<th>termcnt</th>
<th>phrase</th>
<th>docID</th>
<th>termcnt</th>
<th>phrase</th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>1</td>
<td>nikkei stock</td>
<td>28</td>
<td>1</td>
<td>stock average</td>
</tr>
<tr>
<td>28</td>
<td>1</td>
<td>average closed</td>
<td>28</td>
<td>1</td>
<td>points up</td>
</tr>
<tr>
<td>28</td>
<td>1</td>
<td>tokyo stock</td>
<td>28</td>
<td>1</td>
<td>stock exchange</td>
</tr>
<tr>
<td>28</td>
<td>1</td>
<td>exchange Wednesday</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**PHRASE**

<table>
<thead>
<tr>
<th>phrase</th>
<th>idf</th>
</tr>
</thead>
<tbody>
<tr>
<td>average closed</td>
<td>2.49</td>
</tr>
<tr>
<td>exchange Wednesday</td>
<td>3.33</td>
</tr>
<tr>
<td>nikkei stock</td>
<td>2.14</td>
</tr>
<tr>
<td>points up</td>
<td>2.61</td>
</tr>
<tr>
<td>stock average</td>
<td>2.14</td>
</tr>
<tr>
<td>stock exchange</td>
<td>1.34</td>
</tr>
<tr>
<td>tokyo stock</td>
<td>2.10</td>
</tr>
</tbody>
</table>

Simple Phrase Parsing

- Simple phrase parser with the following rules
  - Phrases do not include stop terms
  - Phrases do not span across symbols

*Example:* The Nikkei Stock Average closed at 29,754.73 points up 156.92 points, on the Tokyo Stock Exchange Wednesday.

**Phrases:**

- nikkei stock
- stock average
- average closed
- points up
- tokyo stock
- stock exchange
- exchange wednesday
Relational Document Representation
(Proximity Search)

<table>
<thead>
<tr>
<th>docID</th>
<th>docname</th>
<th>headline</th>
<th>dateline</th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>AP881214-0028</td>
<td>Stocks Up In Tokyo</td>
<td>TOKYO (AP)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>docID</th>
<th>term</th>
<th>offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>nikkei</td>
<td>42</td>
</tr>
<tr>
<td>28</td>
<td>stock</td>
<td>43</td>
</tr>
<tr>
<td>28</td>
<td>average</td>
<td>44</td>
</tr>
<tr>
<td>28</td>
<td>closed</td>
<td>45</td>
</tr>
<tr>
<td>28</td>
<td>points</td>
<td>50</td>
</tr>
<tr>
<td>28</td>
<td>up</td>
<td>51</td>
</tr>
<tr>
<td>28</td>
<td>points</td>
<td>54</td>
</tr>
<tr>
<td>28</td>
<td>tokyo</td>
<td>57</td>
</tr>
<tr>
<td>28</td>
<td>stock</td>
<td>58</td>
</tr>
<tr>
<td>28</td>
<td>exchange</td>
<td>59</td>
</tr>
<tr>
<td>28</td>
<td>wednesday</td>
<td>60</td>
</tr>
</tbody>
</table>

Relational XML Approach: Architecture

Semistructured Query → SQL Query → DB → XML Collection

Database Results
XML Search

• XML provides “tags” that allow both structured and unstructured data to be represented in the same XML document.

• Frequently used as a common representation for a variety of document formats.

Introduction

• XML: Extensible Markup Language
• Defined by the WWW Consortium (W3C)
• Derived from SGML (Standard Generalized Markup Language), but simpler to use than SGML
• Documents have tags giving extra information about sections of the document
  – E.g. <title> XML </title> <slide> Introduction … </slide>
• Extensible, unlike HTML
  – Users can add new tags, and separately specify how the tag should be handled for display

• A wide variety of tools is available for parsing, browsing and querying XML documents/data
XML Introduction (Cont.)

• Tags make data (relatively) self-documenting
  – E.g.
    <university>
      <department>
        <dept_name> Comp. Sci. </dept_name>
        <building> Taylor </building>
        <budget> 100000 </budget>
      </department>
      <course>
        <course_id> CS-101 </course_id>
        <title> Intro. to Computer Science </title>
        <dept_name> Comp. Sci </dept_name>
        <credits> 4 </credits>
      </course>
    </university>

Structure of XML Data

• **Tag**: label for a section of data
• **Element**: section of data beginning with `<tagname>` and ending with matching `</tagname>`
• Elements must be properly nested
  – Proper nesting
    • `<course> ... <title> ... </title> </course>`
  – Improper nesting
    • `<course> ... <title> ... </course> </title>`

  – Formally: every start tag must have a unique matching end tag, that is in the context of the same parent element.
• Every document must have a single top-level element
Example of Nested Elements

```xml
<purchase_order>
  <identifier> P-101 </identifier>
  <purchaser> .... </purchaser>
  <itemlist>
    <item>
      <identifier> RS1 </identifier>
      <description> Atom powered rocket sled </description>
      <quantity> 2 </quantity>
      <price> 199.95 </price>
    </item>
    <item>
      <identifier> SG2 </identifier>
      <description> Superb glue </description>
      <quantity> 1 </quantity>
      <unit-of-measure> liter </unit-of-measure>
      <price> 29.95 </price>
    </item>
  </itemlist>
</purchase_order>
```

Tree Model of XML Data

- An XML document is modeled as a tree, with nodes corresponding to elements and attributes
  - Element nodes have child nodes, which can be attributes or subelements
  - Text in an element is modeled as a text node child of the element
  - Children of a node are ordered according to their order in the XML document
  - Element and attribute nodes (except for the root node) have a single parent, which is an element node
  - The root node has a single child, which is the root element of the document
Relational XML Approach: Storage

\[\begin{array}{c}
\text{book} \\
\text{author} \quad (E) \\
\text{title} \quad (E) \\
\text{publisher} \quad (E) \\
\text{State} = "NY" \quad (A)
\end{array}\]

\[\begin{array}{c|c|c|c|c|c}
\text{pinndx} & \text{pinndxnum} & \text{parent} & \text{tagtype} & \text{tagpath} & \text{atname} & \text{value} \\
\hline
1 & 0 & E & 1 & 1 & NULL \\
2 & 1 & E & 2 & 1 & John Smith \\
3 & 1 & E & 3 & 1 & Colonial America \\
4 & 1 & E & 4 & 1 & NY Publishing \\
5 & 4 & A & 4 & 2 & NY \\
\end{array}\]

\[\begin{array}{c|c}
\text{tagpath} & \text{value} \\
\hline
1 & \text{[book]} \\
2 & \text{[book,author]} \\
3 & \text{[book,title]} \\
4 & \text{[book,publisher]} \\
\end{array}\]

\[\begin{array}{c|c|c}
\text{atname} & \text{value} \\
\hline
1 & \text{NULL} \\
2 & \text{state} \\
\end{array}\]

\[\begin{array}{c}
tagpathtbl
\end{array}\]