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

<term> → *<list of documents>*

e.g., vehicle → *D1, D3, D4* results in:

term	***docID***
vehicle	D1
vehicle	D3
vehicle	D4

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

DOCUMENT

docID	docname	headline	dateline
28	AP881214-0028	Stocks Up In Tokyo	TOKYO (AP)

INDEX

docID	termcnt	term
28	1	nikkei
28	2	stock
28	1	average
28	1	closed
28	2	points
28	1	up
28	1	tokyo
28	1	exchange
28	1	wednesday

TERM

term	idf
average	1.08
closed	1.08
exchange	1.00
nikkei	2.07
points	1.23
stock	1.00
tokyo	1.58
up	0.30
wednesday	0.60

Simplistic Models:

Keyword and Boolean Searches

Relational Approach:
Keyword Search

- Keyword search

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

- Keyword search with stop word list

```
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

```
     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
```

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

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
```

```
select docID
  from INDEX a, INDEX b, INDEX c, ... INDEX N
where   a.term = term1        AND
        b.term = term2        AND
        c.term = term3        AND
                ....
        n.term = termN        AND
        a.docID = b.docID     AND
        b.docID = c.docID     AND
                ....
        N-1.docID = N.docID
```


Fixed Join-Count AND Queries

Find all documents that contain <u>*all*</u> *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:

```
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)

DOCUMENT

docID	docname	headline	dateline
28	AP881214-0028	Stocks Up In Tokyo	TOKYO (AP)

INDEX

docID	termcnt	term
28	1	nikkei
28	2	stock
28	1	average
28	1	closed
28	2	points
28	1	up
28	1	tokyo
28	1	exchange
28	1	wednesday

TERM

term	idf
average	1.08
closed	1.08
exchange	1.00
nikkei	2.07
points	1.23
stock	1.00
tokyo	1.58
up	0.30
wednesday	0.60

Relational Query Representation

ORIGINAL QUERY:
"nikkei stock exchange american stock exchange"

QUERY

Term	tf
nikkei	1
Stock	2
exchange	2
american	1

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 $$\frac{\sum_{i=1}^{t} x_i y_i}{\sqrt{\sum_{i=1}^{t} x_i^2 \bullet \sum_{i=1}^{t} y_i^2}}$$

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.

*(Query Weight * Document Weight)*

SQL:

```
SELECT d.docID, d.docname, SUM(q.termcnt * t.idf * i.termcnt * t.idf)
  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
```


Relevance Ranking:
SQL for Probabilistic Retrieval

$$\sum_{i=1}^{num\_terms} \log\left(\frac{(numdocs - df_i) + .5}{(df_i + .5)}\right) * \left(\frac{2.2 * tf_{id}}{.3 + (.75 * doclength / avgdoclength) + tf_{id}}\right) * qtf$$

```
SELECT d.docID, d.docname, SUM(
   LOG(((NumDocs - t.df) + 0.5) / (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;
```

Relational Document Representation
(Phrase Processing)

DOCUMENT

docID	docname	headline	dateline
28	AP881214-0028	Stocks Up In Tokyo	TOKYO (AP)

INDEX

docID	termcnt	phrase
28	1	nikkei stock
28	1	stock average
28	1	average closed
28	1	points up
28	1	tokyo stock
28	1	stock exchange
28	1	exchange wednesday

PHRASE

phrase	idf
average closed	2.49
exchange Wednesday	3.33
nikkei stock	2.14
points up	2.61
stock average	2.14
stock exchange	1.34
tokyo stock	2.10

20

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

21

Relational Document Representation (Proximity Search)

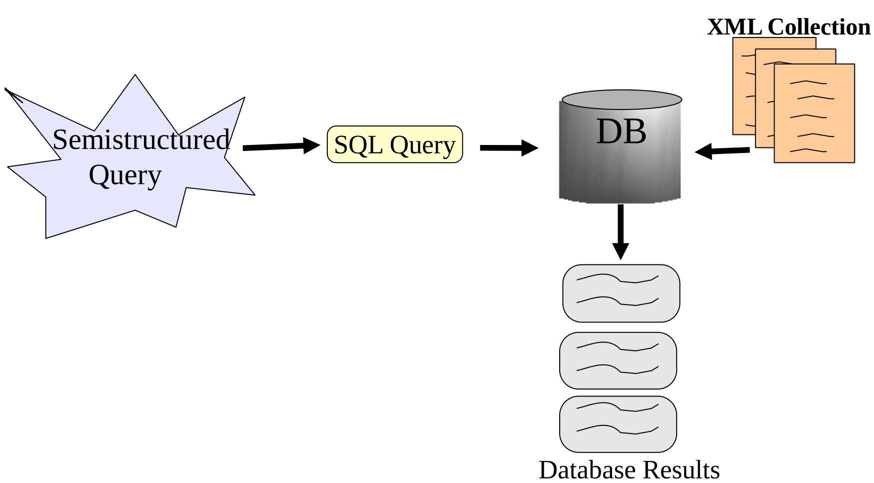
DOCUMENT

docID	docname	headline	dateline
28	AP881214-0028	Stocks Up In Tokyo	TOKYO (AP)

INDEX

docID	term	offset
28	nikkei	42
28	stock	43
28	average	44
28	closed	45
28	points	50
28	up	51
28	points	54
28	tokyo	57
28	stock	58
28	exchange	59
28	wednesday	60

Relational XML Approach: Architecture



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

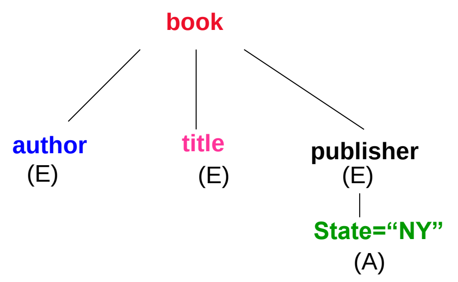
```
<purchase_order>
    <identifier> P-101 </identifier>
    <purchaser>  …. </purchaser>
    <itemlist>
       <item>
           <identifier> RS1 </identifier>
           <description> Atom powered rocket sled </description>
           <quantity> 2 </quantity>
           <price> 199.95 </price>
       </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

<book>
<author>
John Smith
</author>
<title>
Colonial America
</title>
<publisher state=”NY”>
NY Publishing
</publisher>
<book>



33

Relational XML Approach: Storage

<book>
<author>
John Smith
</author>
<title>
Colonial America
</title>
<publisher state=”NY”>
NY Publishing
</publisher>
<book>

pinndx

pinndxnum	parent	tagtype	tagpath	atrname	value
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

tagpath	value
1	[book]
2	[book,author]
3	[book,title]
4	[book,publisher]

tagpathtbl

atrname	value
1	NULL
2	state

atrnametbl

34