Chapter 04
Query Languages

Outline

- Keyword-Based Querying
- Pattern Matching
- Structural Queries
- Query Protocols

Introduction

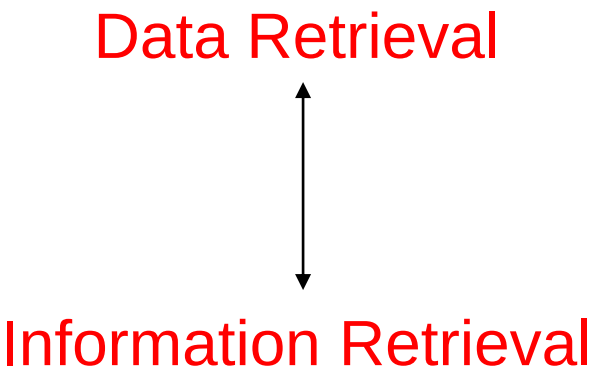
- Goals
  - Which queries can be formulated
  - How the formulation is related to underlying information retrieval models
- Query languages

Keyword-Based Querying

A query is formulation of a user information need

Keyword-based queries are popular

1. Single-Word Queries
2. Context Queries
3. Boolean Queries
4. Natural Language



Keyword-Based Querying

1.Queries = combinations of keywords

2. Queries →Search→ Document collection

3. Word queries are intuitive, easy to express and provide fast ranking.

- The concept of ***word*** must be defined.
  - A word is a sequence of ***letters*** terminated by a ***separator*** (period, comma, blank, etc).
  - Definition of ***letter*** and ***separator*** is flexible; e.g., **hyphen** could be defined as a letter or as a separator.
  - Usually, "***trivial words***"(such as "a", "the", or "of") are ignored.

Single-Word Queries

- A query is formulated by **a word**
- A document is formulated by long sequences of words
- A word is a sequence of letters surrounded by separators
- What are letters and separators? e.g,'**on-line**' The division of the text into words is not arbitrary

Single-Word Queries

Single-word queries:

- A query is a single word
- Simplest form of query.
- All documents that include this word are retrieved.
- Documents may be ranked by the ***frequency*** of this word in the document.

Boolean Queries

- Keywords combined with Boolean operators:
  - OR: (e_1 OR e_2)
  - AND: (e_1 AND e_2)
  - NOT:(NOT e2)
  - BUT: (e_1 BUT e_2) Satisfy e_1 but **not** e_2

A BUT B = A AND not B

Except

- Negation only allowed using BUT to allow efficient use of inverted index by filtering another efficiently retrievable set.
- Naïve users have trouble with Boolean logic.

Boolean queries (cont.)

Examples:

1.computer or server except mainframe

Select all documents that discuss computers, or documents that discuss servers but do not discuss mainframes.

2. (computer or server) except mainframe

Select all documents that discuss computers or servers, do not select any documents that discuss mainframes.

3. computer except (server or mainframe)

Select all documents that discuss computers, and do not discuss either servers or mainframes.

Boolean queries (cont.)

Example:

D1 = (0.3A, 0.5B, 0.6C)

D2 = (0.7A, 0.4B, 0.1C, 0.8D)

Query: (A **and** B) **or** (C **and** D)

	A and B	**C and D**	**Relevance**
D1	0.3	0.0	0.3
D2	0.4	0.1	0.4

Boolean queries (cont.)

Example:

D1 = (0.3A, 0.5B, 0.6C)

D2 = (0.7A, 0.4B, 0.1C, 0.8D)

Query: (A **and** ¬D)

	A and D	**A and ¬D**	**Relevance**
D1	0.0	0.3	0.3
D2	0.7	0.2	0.2

Boolean Queries (cont.)

Example:

D1 = (0.8A, 0.5B, 0.6C)

D2 = (0.4A, 0.4B, 0.1C, 0.8D)

Query: (0.5A **and** 0.2B) **or** (0.9C **and** 0.7D)

	0.5A and 0.2B	**0.9C and 0.7D**	**Relevance**
D1	**min(**0.5*0.8,0.2*0.5**)** **=0.10**	**min(**0.9*0.6,0.7*0.0**)** **=0.00**	**0.10**
D2	**min(**0.5*0.4,0.2*0.4**)** **=0.08**	**min(**0.9*0.1,0.7*0.8**)** **=0.09**	**0.09**

Boolean Queries (cont.)

Example:

D1 = (0.8A, 0.5B, 0.6C)

D2 = (0.4A, 0.4B, 0.1C, 0.8D)

Query: (0.5A **and** ⌐D)

	0.5A and 1D	**0.5A and ⌐1D**	**Relevance**
D1	**min(**0.5*0.8,1.0*0.0**)** =**0.00**	**min(**0.5*0.8,1.0*1.0**)** =**0.40**	**0.40**
D2	**min(**0.5*0.4,1.0*0.8**)** =**0.20**	**min(**0.5*0.4,1.0*0.2**)** =**0.20**	**0.20**

Boolean Retrieval with Inverted Indices

- Primitive keyword: Retrieve containing documents using the inverted index.
- OR: Recursively retrieve e_1 and e_2 and take **union** of results.
- AND: Recursively retrieve e_1 and e_2 and take **intersection** of results.
- NOT: Recursively retrieve e_1 and take **negative** of results.
- BUT: Recursively retrieve e_1 and e_2 and take set **difference** of results.

“Natural Language” Queries

- Full text queries as arbitrary strings.
- Typically just treated as a bag-of-words for a vector-space model.
- Typically processed using standard vector-space retrieval methods.

Natural language

***Example*:**

"*Find all the documents that discuss campaign finance reforms, including documents that discuss violations of campaign financing regulations.* ***Do not include*** *documents that discuss campaign contributions by the gun and the tobacco industries*"

- Natural language queries are converted to a formal language for processing against a set of documents.
- Such translation requires intelligence and is still a challenge

Natural language (cont.)

Pseudo NL processing:

System scans the prose and extracts recognized terms and Boolean connectors. The grammaticality of the text is *not important*.

***Problem*:**

Recognize the negation in the search statement ("**Do not include**...")

"campaign finance reforms" or "violations of campaign financing regulations"and not "campaign contributions by the gun and the tobacco industries".

Context Queries

- **Definition**

  - Search words in a given context

- **Types**
  - **Phrase**

    >a sequence of single-word queries

    >e.g, **enhance retrieval**
  - **Proximity**

    >a sequence of single words or phrases, and a **maximum allowed distance** between them are specified

    >e.g,within distance (enhance, retrieval, 4) will match ‘…enhance the power of retrieval…’

Phrasal Queries

- Retrieve documents with a specific phrase (ordered list of contiguous words)
  - "information theory"
- May allow intervening stop words and/or stemming.
  - "buy camera" matches:
    "buy a camera"
    "buying the cameras"
    etc.

Phrasal Queries

-a sequence of words → **A single unit**
-called "**literal string**" or "**exact phrase**"
-ignore → Separators (commas, colons, etc.) and "trivial words" (e.g., "a", "the", or "of")
-phrase in relavance document must appear ***in sequence***.
Example:
"Object Database System"
"United States of America"

Phrasal Retrieval with Inverted Indices

- Must have an inverted index that also stores ***positions*** of each keyword in a document.
- Retrieve documents and positions for each individual word, ***intersect*** documents, and then ***finally check*** for ordered contiguity of keyword positions.
- Best to start contiguity check with the least common word in the phrase.

Phrasal Search

Find set of documents *D* in which all keywords ($k_1 \ldots k_m$) in phrase occur (using AND query processing).

Intitialize empty set, *R*, of retrieved documents.

For each document, *d*, in *D*:

Get array, P_i ,of positions of occurrences for each k_i in *d*

Find shortest array P_s of the P_i's

For each position *p* of keyword k_s in P_s

For each keyword k_i except k_s

Use binary search to find a position ($p - s + i$) in the array P_i

If correct position for every keyword found, add *d* to *R*

Return *R*

Phrasal Search Example

Query : Object Database System

Object S1
8
15
19
34

Database S2
7
20
32

System S3
21
24
37
45
80
99

S=2 (Shortest Array)

Round 1

P=7

i=1→ P-S+i

→ 7-2+1 = 6 (Not found in S1)

Round 2

P=20

i=1→ P-S+i

→ 20-2+1 = 19 (Found in S1)

i=3→ P-S+i

→ 20-2+3 = 21 (Found in S3)

Add this document to Answer Set

Proximity Queries

- List of words with specific maximal distance constraints between terms.
- Example: “dogs” and “race” within 4 words match “…dogs will begin the race…”
- May also perform stemming and/or not count stop words.

Proximity Retrieval with Inverted Index

- Use approach *similar to phrasal search* to find documents in which all keywords are found in a context that satisfies the proximity constraints.
- During binary search for positions of remaining keywords, find closest position of k_i to p and check that it is within *maximum allowed distance*.

Multiple-word queries

-A query is a set of words (or phrases)

-Two interpretations:

- *A document is retrieved if it includes **any** of the query words.*
- *A document is retrieved if it includes **each** of the query words.*

-Ranking:

- ***Numbers of keywords** in document .*
- *Documents containing **all** the query words are ranked **at the top**.*
- *Documents containing **only one** query word are ranked **at bottom**.*
- ***Frequency** counts may still be used to break ties among documents that contain the same query words.*

Pattern Matching

- Allow queries that **match strings** rather than word tokens.
- Requires more sophisticated data structures and algorithms than inverted indices to retrieve efficiently.

Simple Patterns

- Prefixes: Pattern that matches start of word.
  - "anti" matches "antiquity", "antibody", etc.
- Suffixes: Pattern that matches end of word:
  - "ix" matches "fix", "matrix", etc.
- Substrings: Pattern that matches arbitrary subsequence of characters.
  - "rapt" matches "enrapture", "velociraptor" etc.
- Ranges: Pair of strings that matches any word lexicographically (alphabetically) between them.
  - "tin" to "tix" matches "tip", "tire", "title", etc.

Allowing Errors

- What if query or document contains typos or misspellings?
- Judge similarity of words (or arbitrary strings) using:
  - **Edit distance (Levenstein distance)**
  - **Longest Common Subsequence (LCS)**
- Allow proximity search with bound on string similarity.

Edit (Levenstein) Distance

- Minimum number of character ***deletions, additions, or replacements*** needed to make two strings equivalent.
  - "misspell" to "mispell" is distance 1
  - "misspell" to "mistell" is distance 2
  - "misspell" to "misspelling" is distance 3

Longest Common Subsequence (LCS)

- Length of the longest ***subsequence of characters shared*** by two strings.
- A *subsequence* of a string is obtained by deleting zero or more characters.
- Examples:
  - "misspell" to "mispell" is 7
  - "misspelled" to "misinterpretted" is 7
    "mis…p…e…ed"

Simple Patterns

-Distance (***deletions, additions, or replacements***) between two strings.

(king, 2) matches *kin*, *kong*, *knig*, *kings*, *cling*, …

-Compensate for phonetic spelling errors.

(eight, 1) matches *ate*, *hate*.

Regular Expressions

- Language for composing complex patterns from simpler ones.
  - An individual character is a regex.
  - Union: If e_1 and e_2 are regexes, then ($\boldsymbol{e_1 \mid e_2}$) is a regex that matches whatever **either e_1 or e_2** matches.
  - Concatenation: If e_1 and e_2 are regexes, then $\boldsymbol{e_1\ e_2}$ is a regex that matches a string that consists of a substring that matches $\boldsymbol{e_1}$ **immediately followed by a substring that matches e_2**
  - Repetition (Kleene closure): If e_1 is a regex, then $\boldsymbol{e_1^*}$ is a regex that matches **a sequence of zero or more strings that match e_1**

Regular Expression Examples

- **(u|e)nabl(e|ing)** matches
  - unable
  - unabling
  - enable
  - enabling
- **(un|en)***able matches
  - able
  - unable
  - unenable
  - enununenable

Structured queries

Structured documents → more powerful queries

Example:
Retrieve documents that contain a page in which the phrase *"terrorist attack"* appears in the text and a photo whose caption contains the phrase *"World Trade Center".*

Query :
samepage("*terrorist attack*", **photo**(**caption**("*World Trade Center*"))).

Structural Queries

- **Mixing contents and structure in queries**

  - contents: words, phrases, or patterns

  - structural constraints: containment, proximity, or other restrictions on structural elements

- **Three main structures**

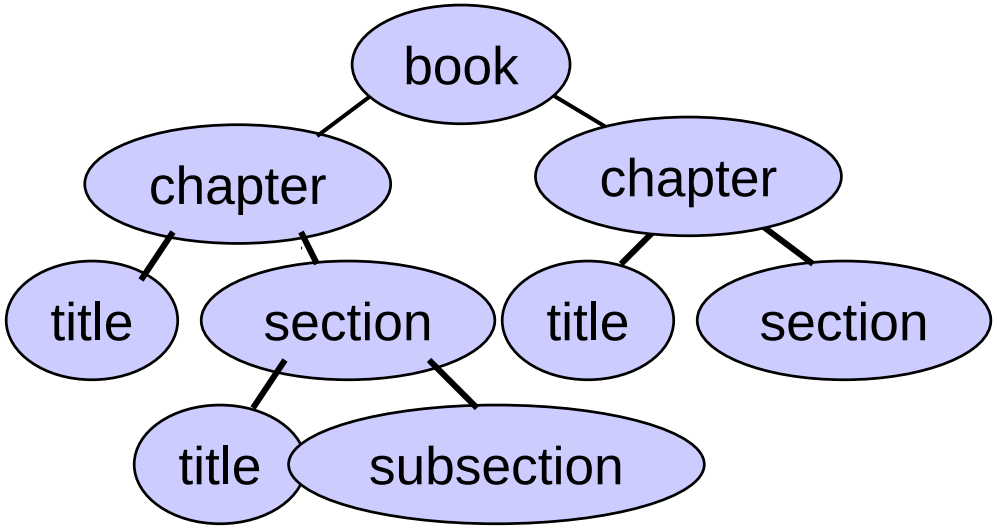
  - Fixed structure

  - Hypertext structure

  - Hierarchical structure

Fixed Structure

- Assumes documents have structure that can be exploited in search.
- Structure could be:
  - Fixed set of fields, e.g. title, author, abstract, etc.

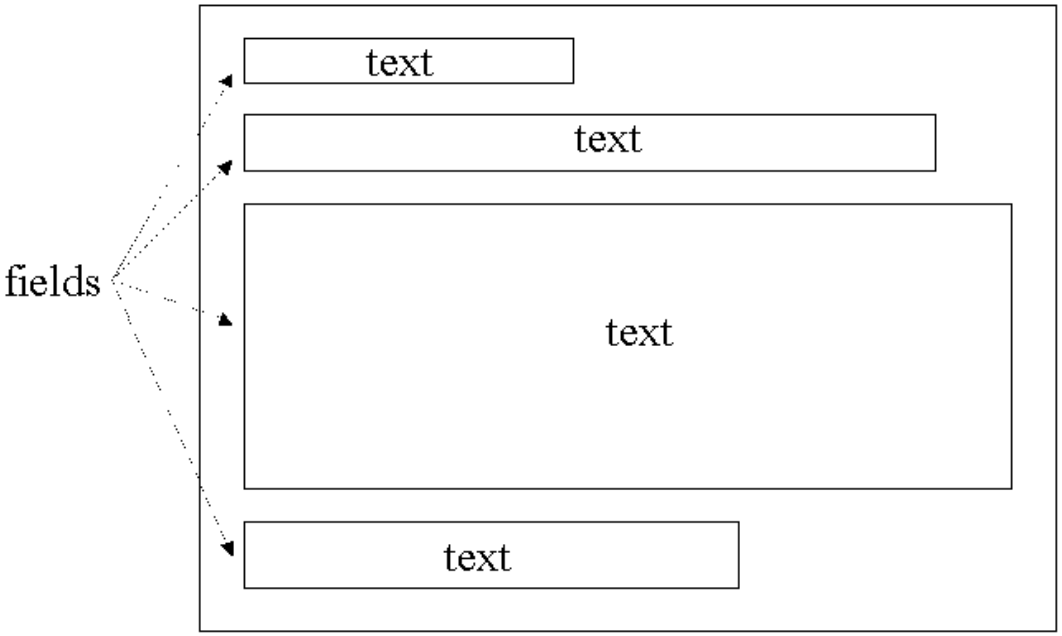


Fixed Structure

Document:a fixed set of fields

EX: a mail has a sender, a receiver, a date, a subject and a body field

Search for the mails sent to a given person with "**football**" in the Subject field



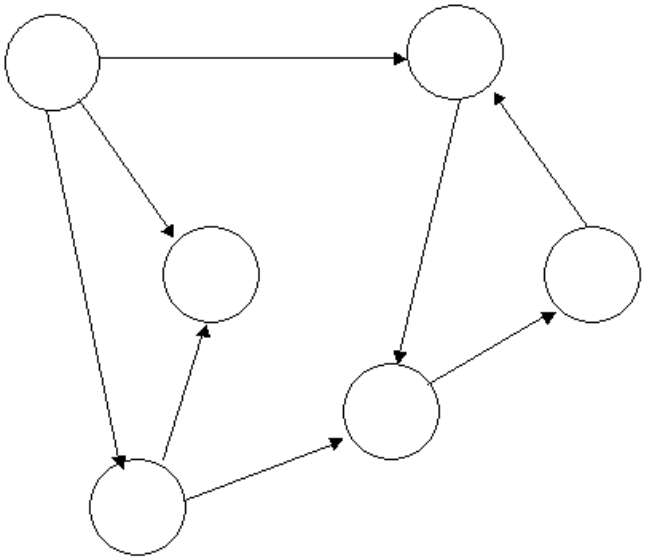
Queries with Structure

- Allow queries for text appearing in specific fields:
  - "nuclear fusion" appearing in a chapter title
- SFQL: Relational database query language SQL enhanced with "full text" search.
  - Select abstract from journal.papers where author contains "Teller" and title contains "nuclear fusion" and date < 1/1/1950

Hypertext

A hypertext is a **directed graph** where nodes hold some text (text contents)

the links represent connections between nodes or between positions inside nodes (structural connectivity)



Hierarchical Structure

Chapter 4

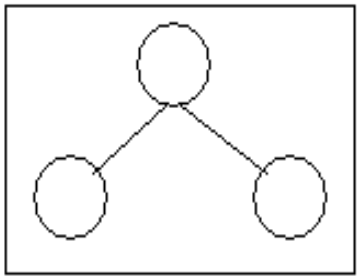
4.1 Introduction

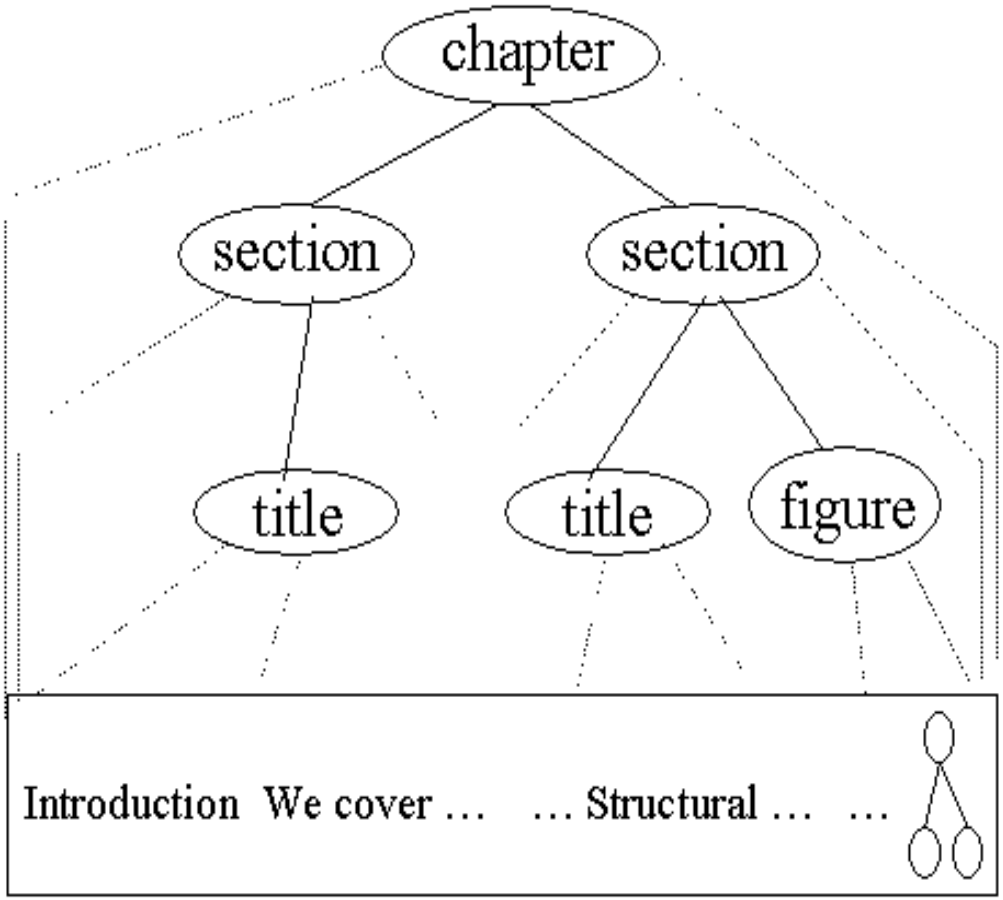
We cover in this chapter the different kinds of …

…

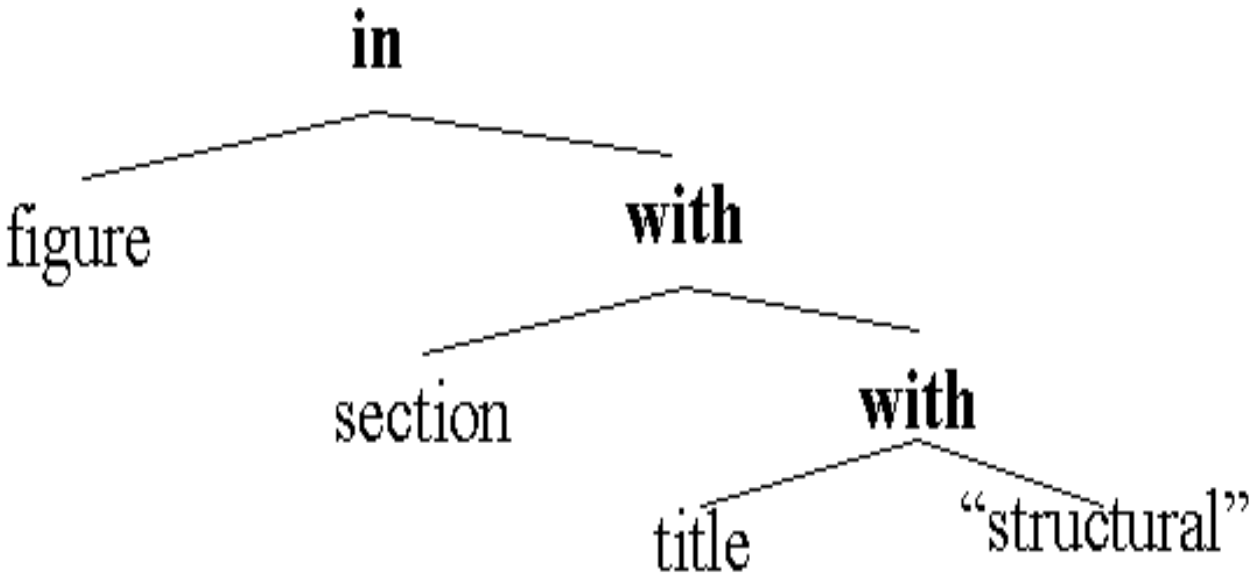
4.4 Structural Queries

…





Hierarchical Structure



Query Protocols

- Z39.50

  Client,Server, Database

- WAIS (Wide Area Information Service)

  query database through internet

Query protocols (cont.)

SFQL (structured Full-text Query Language) is a document retrieval language based on SQL.

Example:

Select author
from Washington-Post
union Washington-Times
where title **contains** “Michael Jordan”
    and date > 10/1/01
    and article **contains** “return”
within 3 words of “game”;