Query Languages

Berlin Chen 2003

Reference:
1. Modern Information Retrieval, chapter 4
The Kinds of Queries

• Data retrieval
  – Pattern-based querying
  – Retrieve docs that contains (or exactly match) the objects that satisfy the conditions clearly specified in the query
  – A single erroneous object implies failure!

• Information retrieval
  – Keyword-based querying
  – Retrieve relevant docs in response to the query (the formulation of a user information need)
  – Allow the answer to be ranked
The Kinds of Queries

• On-line databases or CD-ROM archives
  – High level software packages should be viewed as query languages
  – Named “protocols”

Different query languages are formulated and then used at different situations, by considering
- The underlying retrieval models
- The content (semantics) and structure (syntax) of the text

  Models: Boolean, vector-space, HMM ....
  Formulations/word-treating machineries: Stop-word list, stemming, query-expansion, ....
The Retrieval Units

• The retrieval unit: the basic element which can be retrieved as an answer to a query
  – A set of such basic elements with ranking information

• The retrieval unit can be a file, a doc, a Web page, a paragraph, a passage, or some other structural units

• Simply referred as “docs”
Keyword-based Querying

• Keywords
  – Those words can be used for retrieval by a query
  – A small set of words extracted from the docs
    • Preprocessing is needed

• Characteristics of keyword-based queries
  – A query composed of keywords and the docs containing such keywords are searching for
  – Intuitive, easy to express, and allowing for fast ranking
  – A query can be a single keyword or more complex combination of operation involving several keywords
Keyword-based Querying

• **Single-word queries**
  – **Query**: The elementary query is a word
  – **Docs**: The docs are long sequences of words
  – What is a **word** in **English**?
    • A word is a sequence of **letters** surrounded by **separators**
    • Some characters are not letters but do not split a word, e.g. the hyphen in ‘on-line’
    • Words possess **semantic/conceptual** information
Keyword-based Querying

• **Single-word queries** (cont.)
  – The use of word statistics for IR ranking
    • Word occurrences inside texts
      – Term frequency: number of times a word in a doc
      – Inverse document frequency: number of docs in which a word appears
  – Word positions in the docs
    • May be required, e.g., a interface highlighting each occurrence of a specific word
Keyword-based Querying

• **Context queries**
  – Complement single-word queries with ability to search words in a given context, i.e., near other words

  – Words appearing near each other may signal a higher likelihood of relevance than if they appear apart
    • **Phrases of words or words are proximal in the text**
Keyword-based Querying

- **Context queries** (cont.)
  - Two types of queries
    - **Phrase**
      - A sequence of single-word queries
      - \( Q: \) “enhance” and “retrieval”
      - \( D: \) “...enhance the retrieval....”

- **Proximity**
  - A relaxed version of the phrase query
  - A sequence of single words (or phrases) is given together with a maximum allowed distance between them
    - E.g., two keywords occur within four words
    - \( D: \) “...enhance the power of retrieval...”
Keyword-based Querying

• **Context queries** (cont.)
  – Ranking
    • Phrases: analogous to single words
    • Proximity queries: the same way if physical proximity is not used as a parameter in ranking
      – Just as a hard-limiter
      – But physical proximity has semantic value!
Keyword-based Querying

- **Boolean Queries**
  - Have a syntax composed of *atoms* (basic queries) that retrieve docs, and of Boolean operators which work on their operands

![Query Syntax Tree]

Leaves: basic queries
Internal nodes: operators

A query syntax tree.
Keyword-based Querying

• **Boolean Queries** (cont.)
  – Commonly used operators
    • OR, e.g. (e\_1 OR e\_2)
      – Select all docs which satisfy e\_1 or e\_2. Duplicates are eliminated
    • AND, e.g. (e\_1 AND e\_2)
      – Select all docs which satisfy both e\_1 and e\_2
    • BUT, e.g. (e\_1 BUT e\_2)
      – Select all docs which satisfy e\_1 but not e\_2

\[
\begin{array}{ccc}
e_1 & e_2 & e_1 \ OR e_2 \\
d_3 & d_4 & d_3 \\
d_7 & d_7 & d_4 \\
d_{10} & d_8 & d_7 \\
d_8 & & d_{10}
\end{array}
\]

No partial matching between a doc and a query
No ranking of retrieved docs are provided!
Keyword-based Querying

- **Boolean Queries** (cont.)
  - A relaxed version: a “fuzzy Boolean” set of operators
    - The meaning of AND and OR can be relaxed
      - all: the AND operator
      - one: the OR operator (at least one)
      - some: retrieval elements appearing in more operands than the OR
    - Docs are ranked higher when having a larger number of elements in common with the query
Keyword-based Querying

- **Natural language**
  - Push the fuzzy Boolean model even further
    - The distinction between AND and OR are complete blurred
  - A query is an enumeration of words and context queries
  - All the documents matching a portion of the user query are retrieved
    - Docs matching more parts of the query assigned a higher ranking
  - Negation also can be handled by penalizing the ranking score
    - E.g. some words are not desired
Pattern Matching

• Pattern matching: allow the retrieval of doc based on some patterns
  – A pattern is a set of syntactic features must occur in a text segments
    • Segments satisfying the pattern specifications are said to “match the pattern”
    • E.g. the prefix of a word
  – A kind of data retrieval

• Pattern matching (data retrieval) can be viewed as an enhanced tool for information retrieval
  – Require more sophisticated data structures and algorithms to retrieve efficiently
Pattern Matching

• Types of patterns
  – **Words**
  – **Prefixes**: a string from the beginning of a text word
    • E.g. ‘comput’: ‘computer’, ‘computation’,…
  – **Suffixes**: a string from the termination of a text word
    • E.g. ‘ters’: ‘computers’, ‘testers’, ‘painters’,…
  – **Substrings**: A string within a text word
    • E.g. ‘tal’: ‘coastal’, ‘talk’, ‘metallic’, …
  – **Ranges**: a pair of strings matching any words lying between them in lexicographic order
    • E.g. between ‘held’ and ‘hold’: ‘hoax’ and ‘hissing’,…
Pattern Matching

- **Allowing errors**: a word together with an error threshold
  - Useful for when query or doc contains typos or misspelling
  - Retrieve all text words which are ‘similar’ to the given word
  - **edit (or Levenshtein) distance**: the minimum number of character insertions, deletions, and replacements needed to make two strings equal
    - E.g. ‘flower’ and ‘flo wer’
  - **maximum allowed edit distance**: query specifies the maximum number of allowed errors for a word to match the pattern
Pattern Matching

– Regular Expressions
  • General patterns are built up by simple strings and several operations
  • **union**: if $e_1$ and $e_2$ are regular expressions, then $(e_1 | e_2)$ matches what $e_1$ or $e_2$ matches
  • **concatenation**: if $e_1$ and $e_2$ are regular expressions, the occurrences of $(e_1 e_2)$ are formed by the occurrences of $e_1$ immediately followed by those of $e_2$
  • **repetition** (Kleene closure): if $e$ is a regular expression, then $(e^*)$ matches a sequence of zero or more contiguous occurrence of $e$
  • Example:
    – ‘pro (blem | tein) (s | $\varepsilon$ ) (0 | 1 | 2)*’ matches words ‘problem2’, ‘proteins’, etc.
Pattern Matching

– Extended Patterns
  • Subsets of the regular expressions expressed with a simpler syntax
  • System can convert extended patterns into regular expressions, or search them with specific algorithms
  • E.g.: classes of characters:

<table>
<thead>
<tr>
<th>RE</th>
<th>Expansion</th>
<th>Match</th>
<th>Example Patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>\d</td>
<td>[0-9]</td>
<td>any digit</td>
<td>Party_of_5</td>
</tr>
<tr>
<td>\D</td>
<td>[^0-9]</td>
<td>any non-digit</td>
<td>Blue_moon</td>
</tr>
<tr>
<td>\w</td>
<td>[a-zA-Z0-9_]</td>
<td>any alphanumeric or space</td>
<td>Daiyu</td>
</tr>
<tr>
<td>\W</td>
<td>[^\w]</td>
<td>a non-alphanumeric</td>
<td>!!!!</td>
</tr>
<tr>
<td>\s</td>
<td>[\w\r\t\n\f]</td>
<td>whitespace (space, tab)</td>
<td>in_Concord</td>
</tr>
<tr>
<td>\S</td>
<td>[^\s]</td>
<td>Non-whitespace</td>
<td></td>
</tr>
</tbody>
</table>
Structural Queries

- Docs are allowed to be queried with respect to both their text content and structural constraints
  - **Text content**: words, phrases, or patterns
  - **Structural constraints**: containment, proximity, or other restrictions on the structural elements (e.g., chapters, sections, etc.)

*Mixing contents and structures in queries*
Structural Queries

• Three main structures discussed here
  – Form-like fixed structure
  – Hierarchical structure
  – Hypertext structure

What structure a text may have?
What can be queried about that structure? (the query model)
How to rank docs?
Form-like Fixed Structure

- Docs have a fixed set of **fields**, much like a filled form
  - Each field has some text inside
  - Some fields are not presented in all docs
  - Text has to be classified into a field
  - Fields are not allow to nest or overlap
  - A given pattern only can be associated with a specified filed
  - E.g., a **mail achieve** (sender, receiver, date, subject, body ..)
    - Search for the mail sent to a given person with “football” in the subject field

- Compared with the relational database systems
  - Different fields with different data types
Hypertext Structure

• A hypertext is a directed graph where
  – Nodes hold some text (content)
  – The links represent connection (structural connectivity) between nodes or between positions inside the nodes

• Retrieval from a hypertext began as a merely navigational activity
  – Manually traverse the hypertext nodes following links to search what one wanted
  – It’s still not possible to query the hypertext based on its structure

• An interesting proposal to combine browsing and searching on the web → WebGlimpse
Hierarchical Structure

- Represent a recursive decomposition of the text and is a natural model for many text collections
  - E.g., books, articles, legal documents,…
Issues of Hierarchical Structure

• Static or dynamic structure
  – Statistic: one or more explicit hierarchies can be queried, e.g., by ancestry
  – Dynamic: not really a hierarchy, the required elements are built on the fly
    • Implemented over a normal text index

• Restrictions on the structure
  – The text or the answers may have restrictions about nesting and/or overlapping for efficiency reasons
  – In other cases, the query language is restricted to avoid restricting the structure
Issues of Hierarchical Structure

• Integration with text
  – Effective Integration of queries on text content with queries on text structure
  – From perspectives of classical IR models and structural models, respectively

• Query language
  – Some features for queries on structure including selection of areas that
    • Contain (or not) other areas
    • Are contained (or not) in other areas
    • Follow (or are followed by) other areas
    • Are close to other areas
  – Also including set manipulation

Classical model: primary -> text
  secondary->structure

Structural model: primary -> structure
  secondary->text
Query Protocols

• The query languages used automatically by software applications to query text databases
  – Standards for querying CD-ROMs
  – Or, intermediate languages to query library systems

• Important query protocols
  – Z39.50
    • For bibliographical information systems
    • Protocols for not only the query language but also the client-server connection
  – WAIS (Wide Area Information Service)
    • A networking publishing protocol
    • For querying database through the Internet
Query Protocols

• CD-ROM publishing protocols
  – Provide “disk interchangeability”: flexibility in data communication between primary information providers and end users
  – Some example protocols
    • CCL (Common Command Language)
    • CD-RDx (Compact Disk Read only Data exchange)
    • SFQL (Structured Full-text Query Languages)
Trends and Research Issues

• Types of queries and how they are structured