Query languages

Query languages

- The answers an IR system will provide depend on the retrieval model
- Information retrieval ≠ Data retrieval
  - Main difference: ranking
- A protocol is a lower level language, i.e., it's a language for the applications to communicate, but it is not a query language
- Languages should be able to use the semantics of the information as well as its structure
  - Query expansion, thesaurus and stemming may be used (more on this later)

Keyword-based Querying

- Single-word querying
  - Very popular on the WWW
  - Which of the following is a single term ?
    - on-line
    - http://www.acm.org
    - data bases
  - Depending on the model some terms may not be indexed

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.

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.

Boolean Querying

- Traditional operators:
  - AND, OR and NOT
  - Sub-expressions may be composed
  - Naive users may be confused
    - A and (B or C) ≠ A and B or C
- A possibility is to ignore the operators and only pay attention to the operands, basically OR’ing all of them. Ranking is based on how well the docs match the set of query terms. What about NOT though ?
- BUT instead of NOT
  - A but B= A and (not B) – first retrieve all documents satisfying A and then filter those not satisfying B

Pattern Matching

- Allow queries that match strings rather than word tokens.
- Requires more sophisticated data structures and algorithms 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
- Can be computed efficiently using *dynamic programming* in O(*mn*) time where *m* and *n* are the lengths of the two strings being compared.

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”

Regular Expressions

- Language for composing complex patterns from simpler ones.
  - An individual character is a regex.
  - Union: If *e1* and *e2* are regexes, then (*e1* | *e2*) is a regex that matches whatever either *e1* or *e2* matches.
  - Concatenation: If *e1* and *e2* are regexes, then *e1 e2* is a regex that matches a string that consists of a substring that matches *e1* immediately followed by a substring that matches *e2*
  - Repetition (Kleene closure): If *e1* is a regex, then *e1** is a regex that matches a sequence of zero or more strings that match *e1*

Regular Expression Examples

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

Structured Queries

- So far, we assumed documents that are entirely free of structure.
- *Structured* documents would allow more powerful queries.
- Queries could combine text queries with structural queries: queries that relate to the structure of the document.
- *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”.
- The corresponding query could be:**samepage**(“terrorist attack”, **photo**(**caption**(“World Trade Center”))).
- The three main structures:
  1. Form-like fixed structure
  2. Hypertext structure
  3. Hierarchical structure

Structured Queries *(cont.)*

Fixed Structure

- Document is divided to a fixed set of fields, much like a filled form.
- Fields may be associated with types, such as *date*.
- Each field has text
- Fields cannot nest or overlap.
- Queries (multiple-words, Boolean, proximity, patterns, etc.) are targeted at particular fields.
    - Suitable for documents such as mail messages, with fields for
    - Sender
    - Receiver
    - Date
    - Subject
    - Message body
- Lends itself to storage and manipulation in relational databases.
- However, there may not be any domain constraints for the fields – such constraints are important for a RDBMS to be efficient
- On the one hand it requires a well-defined structure, which is something we may not want to impose
- There is a on-going effort to extend RDBMS to incorporate full-text retrieval capabilities, though not in a full text retrieval but rather in a structured text retrieval setting (XML-ish stuff)

Structured Queries *(cont.)*

Hypertext.

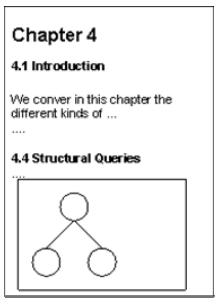
- The most general document structure.
- The term *hypertext* was coined by American computer scientist Ted Nelson in 1965 to describe textual information that could be accessed in a nonlinear way.
- The prefix *hyper* describes the speed and facility with which users could jump to and from related areas of text.
- Each document is divided into *regions* (*nodes*),where a region could be a section, a paragraph, or an entire document; regions may be nested.
- The nodes are connected with *directed links.* A link is *anchored* in a phrase or a word in one node and *leads* to another node.
- Result is a network of document parts.
- However, the WWW was used for browsing only, until the first search engines appeared
- Even today the searches are mostly based on HTML documents and do not explore (with some exceptions) the graph nature of a page.

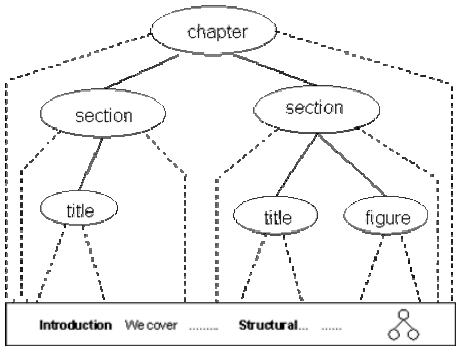
Structured Queries *(cont.)*

Hierarchical structure.

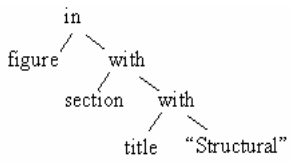
- Intermediate model between fixed structure and hypertext.
- The “anarchic”hypertext network is restricted to a hierarchical structure.
- The model allows recursive decomposition of documents.
- Queries may combine
- Regular text queries, which are targeted at particular areas (the target area is defined by a “path expression”).
- Queries on the structure itself; for example “retrieve documents with at least 5 sections”

Structured Queries *(cont.)*





Query example



Query Protocols

Protocol vs. Language: S/W Program vs. Human

Query Protocol: a query language that is used automatically by software applications to query text databases

Standards for ▪ Querying CD-ROMs ▪ Querying Library systems

Z39.50

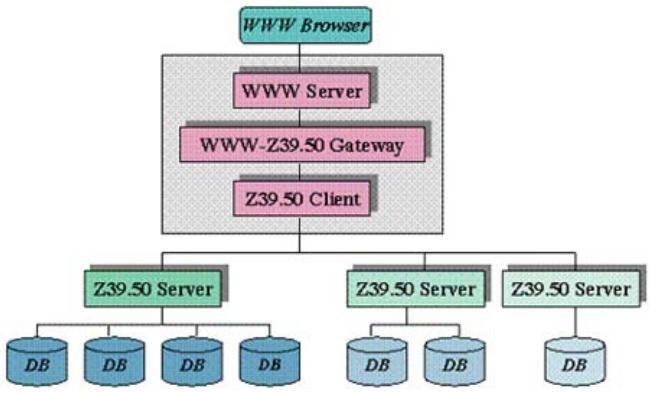
Information Retrieval (Z39.50);

Application Service Definition and Protocol Specification,

- American National Standards Institute (ANSI)
- National Information Standards Organization (NISO)

- A computer protocol that can be implemented on any platform, defines a standard way for two computers to communicate for the purpose of information retrieval
- A Z39.50 implementation enables one interface to access multiple systems providing the end-user with nearly transparent access to other systems
- Specifies data structures and interchange rules that allow a client machine to search databases on a server machine and retrieve records that are identified as a result of such a search
- A large and complex standard
- Widely used in bibliographic and digital library applications

Z39.50 - Web



WAIS (Wide Area Information Servers)

- Searching of databases on the Internet.
- In response to a word or words entered by a user, WAIS displays a list of names of documents on the server computer that match the query.
- Documents containing numerous uses of the keywords appear at the top of the list; those with only a single reference appear at the bottom of the list
- WAIS servers are specialized, each dealing with a specific subject, such as astronomy, physics, cooking, or political issues.

SFQL (Structured Full-text Query Language)

- Document retrieval language based on SQL.
- Merging of database and information retrieval technologies.
- Documents are stored in relations.
- Each document is a row.
- Documents are assumed to be marked ("tagged") by a standard markup language, such as SGML.
- There are columns for "tagged"regions of the documents; for example,
  - Date
  - Abstract
  - The full text
- The familiar SELECT statement is used to express queries. It consists of three basic clauses:
- The **from** clause lists the document collections.
- The **where** clause specifies the criteria for including documents (records)in the result.
- The **select** clause specifies a list of tag-fields to be returned from matched documents (records).
- *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";

Conclusion

- The truth is there is not really a standard IR query language, nor interface …
- Main reason (IMHO): if one cannot precisely state what information is needed, how to come up with a precise language to express that ?