Modern information retrieval

Chapter 04: Query Languages
Outline

• Keyword-based querying
• Queries in weighted systems
• Pattern matching
• Natural language
• Structured queries
• Query protocols
Keyword-based querying

- Queries are combinations of words.
- The document collection is searched for documents that contain these words.
- 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.
Basic 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.
Basic queries (cont.)

Phrase queries:
- A query is a sequence of words treated as a single unit.
- Also called “literal string” or “exact phrase” query.
- Phrase is usually surrounded by quotation marks.
- All documents that include this phrase are retrieved.
- Usually, separators (commas, colons, etc.) and “trivial words” (e.g., “a”, “the”, or “of”) in the phrase are ignored.
- In effect, this query is for a set of words that must appear in sequence.
- Allows users to specify a context and thus gain precision.
- Example: “United States of America”.

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.

• Documents may be ranked by the number of query words they contain:
  • A document containing \( n \) query words is ranked higher than a document containing \( m < n \) query words.
  • 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.

• Example:
  • The phrase “Venetian blind” finds documents that discuss Venetian blinds.
  • The set (Venetian, blind) finds in addition documents that discuss blind Venetians.
Proximity queries: Restrict the distance within a document between two search terms.

• Important for large documents in which the two search words may appear in different contexts.
• Proximity specifications limit the acceptable occurrences and hence increase the precision of the search.
• General Format: \textit{Word}_1 \textit{within } m \textit{ units of } \textit{Word}_2.
• Unit may be character, word, paragraph, etc.
• Examples:
  • \textit{united within 5 words of } \textit{american}: Finds documents that discuss “United Airlines and American Airlines” but not “United States of America and the American dream”.
  • \textit{nuclear within 0 paragraphs of cleanup}: Finds documents that discuss
Boolean queries. Describe the information needed by relating multiple words with Boolean operators.

- **Operators:** and, or, except
- **except** corresponds to **and not**
- **Semantics:** For each query word $w$ a corresponding set $D_w$ is constructed that includes the documents that contain $w$.
- The Boolean expression is then interpreted as an expression on the corresponding document sets with corresponding set operators:
  - **and** $\rightarrow$ intersection
  - **or** $\rightarrow$ union
  - **except** $\rightarrow$ difference
• The use of **except** prevents creation of very large answers: **not** $B$ computes all the documents that do not include $B$(complement), whereas $A$ **except** $B$ limits the universe to the documents that include $A$.

• Precedence: **except**, **and**, **or**; use parentheses to override; process left-to-right among operators with the same precedence.

• **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.)

- Classical Boolean systems do not rank documents:
  - A document either satisfies the query (and is retrieved)
  - or it does not satisfy the query (and is not retrieved).
- The Boolean formalism is not simple for users without training in mathematics.
Weighted queries

*Weighted multiple-word queries.* Each of the words is assigned a different *weight*, expressing the relative importance of the word within the request.

- A query is then a set of word-weight pairs: \((k_1, w_1), \ldots, (k_n, w_n)\).
- The ranking of a document is the *sum* of the weights for the query words that it satisfies.

**Example:**

- Query: \((A,0.8,), (B,0.5), (C,0.3)\)
- Document 1: \((A, B, D)\)
- Document 2: \((A, C, D)\)
- Ranking of Document 1: 0.8+0.5 = 1.3
- Ranking of Document 2: 0.8+0.3 = 1.1
- Each document includes two words from the query, but Document 1 is ranked higher because it includes more important words.
Weighted queries (cont.)

Weighted Boolean queries. Each word in a Boolean query is associated with a weight.

- **Example**: \( A^{0.9} \text{ and } (B^{0.8} \text{ or } C^{0.3}) \).
- A document with \( A \) and \( B \) satisfies this query better than a document with \( A \) and \( C \) (without such weights, both documents satisfy the query equally).
- Possible approach:
  - or
    - \( A^{1} \lor B^{1} \) includes all the documents in \( DA \cup DB \).
    - \( A^{1} \lor B^{0} \) includes all the documents in \( DA \).
    - As the weight of \( B \) changes from 0 to 1, documents from \( DB \setminus DA \) are added to \( DA \).
  - and
    - \( A^{1} \land B^{1} \) includes all the documents in \( DA \cap DB \).
    - \( A^{1} \land B^{0} \) includes all the documents in \( DA \).
    - As the weight of \( B \) changes from 1 to 0, documents from \( DA \setminus DB \) are added to \( DA \cap DB \).
Weighted queries (cont.)

- except
- \( A_1 \land \neg B_1 \) includes all the documents in \( DA - DB \).
- \( A_1 \land \neg B_0 \) includes all the documents in \( DA \).
- As the weight of B changes from 1 to 0, documents from \( DA \cap DB \) are added to \( DA - DB \).
Queries in weighted systems

- Until now we assumed that a document is a simple set of words.
- Assume now that the words of each document are *weighted* according to their importance within the document.
- A document is a set of word-weight pairs \((k_1, w_1), \ldots, (k_n, w_n)\).
- If a global vector of all the possible words is known \((k_1, \ldots, k_t)\), then a document can be represented by a corresponding vector of weights \((w_1, \ldots, w_t)\), with 0 weight for words not in the document.
- The four cases of queries are now revisited:
  1. Multiple-word queries
  2. Weighted multiple-word queries.
  4. Weighted Boolean queries.
• *Weighted multiple-word queries.* Each of the words is assigned a different *weight*, expressing the relative importance of the word within the request.
• A query is then a set of word-weight pairs, and assuming a global vector of all the possible words, a query can be represented by a corresponding vector of weights \((w_1, \ldots, w_t)\), with 0 weight for words not in the query.
• This is the *vector model*, and queries are ranked by a similarity measure.
• *Non-weighted multiple-word queries* can be handled as a special case in which the query weights are 0 or 1.
Queries in Weighted Systems (cont.)

- **Boolean Queries.** Each word in a Boolean query is associated with a weight.
- **Possible approach:**
  - Transform the query to *disjunctive normal form*:
    - Sets of conjuncts of the form $T_1 \land T_2 \land T_3 \ldots$
    - Connected by a $\lor$ operators.
  - Given a document $d$,
    - First, its relevance to each conjunct is computed as the *minimum* weight of any term that appears in the conjunct.
    - Then, the document relevance for the complete query is the *maximum* of the weights of the conjuncts.
- **Weighted Boolean queries.** We do not offer a solution for this case. Of course, the weights in either the query or the documents can be converted to 0 and 1, using a threshold, thus reverting this case to previous
Queries in Weighted Systems (cont.)

- Example: Two documents indexed by four terms:
  - Document 1 = A0.2, B0.5, C0.6, D0
  - Document 2 = A0.7, B0.4, C0.1, D0.8
  - Query: (A and B) or (C and D)

<table>
<thead>
<tr>
<th></th>
<th>A and B</th>
<th>C and D</th>
<th>Relevance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document 1</td>
<td>0.2</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>Document 2</td>
<td>0.4</td>
<td>0.1</td>
<td>0.4</td>
</tr>
</tbody>
</table>
Queries in Weighted Systems (cont.)

• **Final note:** When interpreting queries, some models *demote* documents that include keywords that were not requested
  • The vector model with the cosine measure
  • The Bayesian network model
• **Example:** Assume the vector model with the cosine measure and the simple case that both documents and queries use binary values. Consider these two documents and a query:
  • $d_1 = (0,1,0,1,0)$, $d_2 = (0,1,1,1,0)$, $q = (0,1,0,1,0)$
  • $sim(q, d_1) = 1.0$, $sim(q, d_2) = 0.82$
  • $d_2$ is demoted because it includes an extra keyword not requested by $q$.
• In contrast, the Boolean model does not “penalize” documents with extra (non-


Pattern queries

*Pattern*: An expression that defines a set of words, possibly infinite.

- *Pattern matching*: A word matches a pattern if it is equal to *one* of the words defined by the pattern.
- In other words, the semantics are of *disjunction*: A pattern $P$ that defines $w_1$, $w_2$, ..., $w_n$ is interpreted as $w_1 \lor w_2 \lor \ldots \lor w_n$.
- Patterns may be included in queries.
  - When used by itself a pattern $P$ is a disjunctive query
    
    $$w_1 \lor w_2 \lor \ldots \lor w_n.$$ 
  - A proximity query $P \text{near} A$ is interpreted as
    
    $$(w_1 \text{near} A) - (w_2 \text{near} A) - \ldots - (w_n \text{near} A).$$
  - A Boolean query $P \text{and} A \text{or} B$ is interpreted as
    
    $$(w_1 \text{and} A \text{or} B) - (w_2 \text{and} A \text{or} B) - \ldots - (w_n \text{and} A \text{or} B).$$
Pattern queries (cont.)

*Simple pattern.* Specifies a string that restricts the words that are defined.
  - *Word* pattern defines a single word (the given string).
    - The word pattern `computer` defines `computer`.
  - *Prefix* pattern defines the set of all words that *begin* with the given string.
    - The prefix pattern `comput` defines `computer`, `computation`, `computing`, …
  - *Suffix* pattern defines the set of all words that *end* with the given string.
    - The suffix pattern `load` defines `load`, `unload`, `download`, `upload`, …
  - *Substring* pattern defines the set of all words that *embed* the given string.
    - The substring pattern `tal` defines `coastal`, `talk`, `metallic`, …

*Rangepattern.* Specifies two strings (and a total order on all words); defines all the words *between* the two strings.
  - Lexicographical order: The range pattern `(kind, knife)` defines words such as `king`, `kiss`.
  - Date order: The range pattern `(01/01/2001, 12/31/2001)` defines dates such as `01/20/2001`, `09/11/2001`
Pattern queries (cont.)

Similarity pattern. Specifies a string and a radius; defines all the words whose distance from the string is within the radius.

- Assume the distance between two strings is measured by the number of one-character changes (insertions, deletions, replacements or transpositions) required to transform one string into the other.
  - The similarity pattern \((king, 2)\) defines \(kin, kong, knig, kings, cling, \ldots\)
  - Useful to compensate for typing or scanning (OCR) errors.
- A different distance measure may be used to compensate for phonetic spelling errors.
  - The similarity pattern \((eight, 1)\) defines \(ate, hate\).

Regular expressions. A general pattern language; used frequently in Unix tools.

- The regular expression \([a-z] *[0-9]\) defines arbitrary-length strings that begin with a letter and end with a digit.
Using natural language for querying is very attractive.

- **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.

- Often used by WWW search engines.
- **Problem:** Recognize the negation in the search statement (“Do not include…”).

**Compromise:** Users enter natural language _clauses_ connected with Boolean operators.

- In the above example: “campaign finance reforms” or “violations of campaign financing regulations” _and not_ “campaign contributions by the gun and the tobacco industries”.

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.
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.
• Hypertext lends itself more to browsing than to querying.
• Search techniques for hypertext will be discussed later in the course.
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.)

Example:

Typical query:
- Retrieve documents with a section whose title contains “simpsons” and whose text contains “Homer”.

The Simpsons  The Simpson family has five members... A powerful figure
Query protocols

Over the years, several standards and protocols for query languages have emerged. Two examples:

• **WAIS (Wide Area Information Servers)** allows searches 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) is a 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”;