CSA4020

Multimedia Systems:
Adaptive Hypermedia Systems

Lecture 2:
Indexing and Inverted Files

Indexing on Relational Databases

- Index structures:

  hashing, dynamic and linear hashing, B-tree, B+-tree

- Index whole attribute

- Search whole attribute value (partial match possible, but expensive)

Indexing on (unstructured) documents

- Index structures:

  hashing, dynamic and linear hashing, B-tree, B+-tree, tries

- Index word within document

- Must support partial matching and wildcard matches

Major difference between IR & RDBMS

- SQL returns all and only those results which totally satisfy the query
- IR can return partially satisfactory results!

What to index?

- Documents are seen as containing **terms** which convey "meaning"
- Possible to extract a set of terms that occur in each document (similar to a book index)
- Now we would have records composed of document IDs and a list of terms
- We can "invert" the index, so that the index is ordered by a unique list of terms, and each term is associated with list of documents which contain the term

The Inverted Index

- Structure created to store searchable index of terms and pointers to documents containing those terms
- Can implement index file in any efficient disk file structure (ideally, index file will be small enough to fit in RAM)
- The **vocabulary** of the **document collection** is the number of unique index terms

Boolean Retrieval

- Boolean Retrieval is based on set theory and Boolean algebra

- If we think of the postings list as a set of documents which contain the index term…

- … we can apply set operations to the document sets…

- A Boolean query in composed of terms connected by Boolean operators

((computer AND science) OR
(informatics)) NOT (arts)

Advantages of Inverted Indices

- Fast retrieval from disk

  overall efficiency is dependent on no. of terms in query and the matching algorithm used

- Flexibility

  different kinds of info can be stored along with the terms, eg., frequency info, positional info, etc.

- Support for complex retrieval operations

  if enough info is stored, can create new operators, eg, NEAR

Disadvantages of Inverted Indices

- Large storage overhead
- High maintenance costs on updates, insertions and deletions
- Processing cost increases with number of Boolean operators
- Do terms really capture semantics?

Extensions on Inverted Indices

- Distance constraints

  Adjacency conditions often needed

  "computer" immediately followed by "science"
  "science" no more than 2 words before "computing"
  "computer" and "science" in the same sentence

  Keep location information in index

  Can lead to structural data, eg, term occurs in heading, especially if document is HTML or related

- Term weights

  Occurrence info can be kept

  Count the frequency of occurrence of each term in each document
  More accurate if the frequency count is weighted, by incorporating document frequency information as well

Extensions (contd.)

- Synonyms

  Increase coverage of a query
  Terms which are synonyms can point to the same postings list

- Term truncation

  Suffix truncation is simple form of stemming

  For English and other languages that support it

  “Computer”, “computers”, “computing”…

  Document terms must match query terms to be relevant…

  Comput* = computer, computers, computing

  Increases chance of relevant document being retrieved, but…

  Can easily be handled on an inverted index: 2 ways…

  Languages often support prefix and postfix operators too…

Conclusion

You can only get out of the index what you put into it!

Creating the index is often the most expensive (time, computation, space) part of Information Indexing and Retrieval

Features which appear to improve the retrieval of relevant documents, may inadvertently increase the chances of retrieving non-relevant documents

Know what your “client” expects!