MODULE 2

STRUCTURED TEXT RETRIEVAL MODELS

STRUCTURED TEXT RETRIEVAL MODELS

- Consider a user with high visual memory. Such a user might then recall that the specific document he is interested in contains a page in which the string “atomic holocaust” appears in italic in the text surrounding a figure whose label contains the word “earth”.
- With a classic information retrieval model, this query could be expressed as ‘atomic holocaust ’ and ‘earth’ which retrieves all the documents that contains both strings.
- However this answer contains many more documents than desired by this user.
- In this particular case, the user would like to express his query through a richer expression such as

  same-page (near (‘*atomic holocaust*,’ Figure (label (‘earth’))))

- Which conveys the details in his visual recollection.

- This example illustrates the appeal of a query language which allows us to combine the specification of strings with the specification of structural components of the document.
- Retrieval models which combine information on text content with information on the document structure are called structured text retrieval models.
- For a query such as one illustrated above, a structured text retrieval system searches for all the documents which satisfy the query.
- Thus there is no notion of relevance attached to the retrieval task.

- The retrieval system could search for documents which match query conditions only partially.
- In this situation matching would be approximate and some ranking would have to be used for ordering the approximate answers.
- Types of structured text retrieval models:

I. Non-overlapping lists

II. Proximal nodes

- Match point:refer to the position in the text of a sequence of words which matches the user query.
- Thus if the user specifies query: 'atomic holocaust in Hiroshima’ and this string appears in 3 positions in the text of document dj. Then we can say that document dj contains 3 match points.
- Region : refer to a contiguous portion of the text.
- Node : refer to a structural component of the document such as chapter, a section , subsection.
- Thus a node is a region with predefined topological properties which are known both to the author of document and to the user who searches the document system.

MODEL BASED ON NON-OVERLAPPING LISTS

- Divides the whole text of each document in non-overlapping text regions which are collected in a list.
- Since there are multiple ways to divide a text in non-overlapping regions,multiple lists are generated.
- For eg: list of chapters in the document , list of all sections in the document ,list of all subsections in the document.
- These lists are kept as separate and distinct data structures.

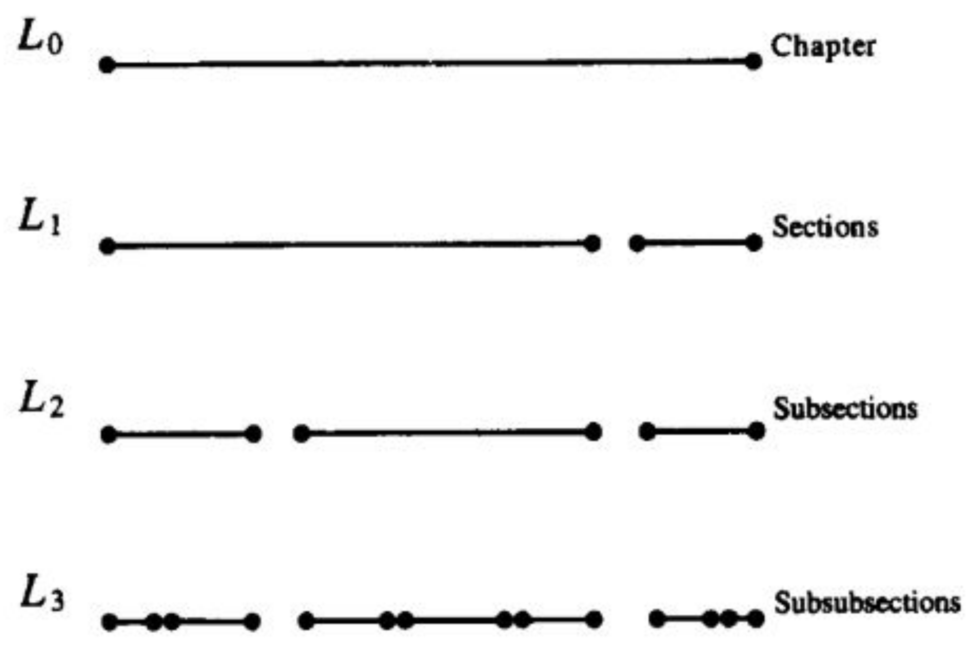

Figure 2.11 Representation of the structure in the text of a document through four separate (flat) indexing lists.

To allow searching for index terms and for text regions, a single inverted file (see Chapter 8 for a definition of inverted files) is built in which each structural component stands as an entry in the index. Associated with each entry, there is a list of text regions as a list of occurrences. Moreover, such a list could be easily merged with the traditional inverted file for the words in the text. Since the text regions are non-overlapping, the types of queries which can be asked are simple: (a) select a region which contains a given word (and does not contain other regions); (b) select a region A which does not contain any other region B (where B belongs to a list distinct from the list for A); (c) select a region not contained within any other region, etc.

MODELS BASED ON PROXIMAL NODES

- A model which allows the definition of independent hierarchical indexing structures over the same document.
- Each of these indexing structures is a strict hierarchy composed of chapters,sections,paragraphs, pages and lines which are called nodes.
- To each of these nodes is associated a text region . Two distinct hierarchies might refer to overlapping text regions.
- Given a user query which refers to distinct hierarchies the compiled answer is formed by nodes which all come from only one of them.
- Thus an answer cannot be composed of nodes which come from two distinct hierarchies.

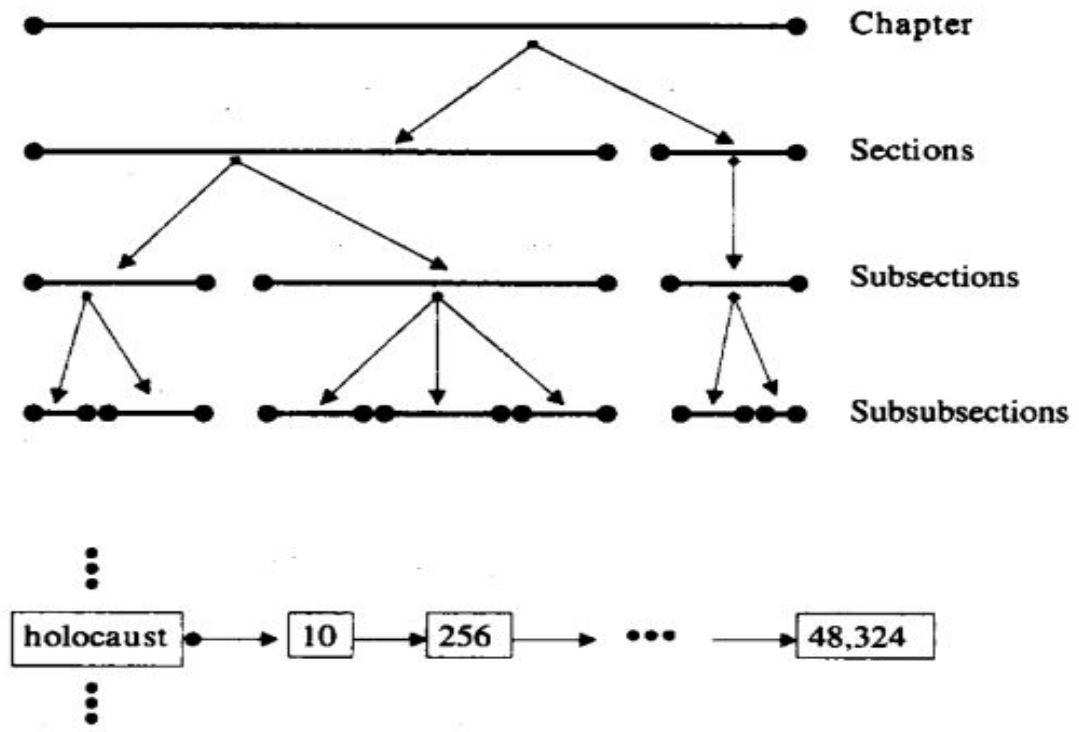

Figure 2.12 Hierarchical index for structural components and flat index for words.

- The figure illustrates a hierarchical indexing structure composed of 4 levels (corresponding to a chapter, sections ,subsections and subsubsections of same document) and an inverted list for the word ‘holocaust’.
- The entries in this inverted list indicate all the positions in the text of the documents in which the word ‘holocaust’ occurs.
- In the hierarchy each node indicates the position in the text of its associated structural component
- The query language allows the specification of regular expressions (to search strings) the reference to structural components by name (to search for chapter for instance) and a combination of these.

- In this sense, model can be viewed as a compromise between expressiveness and efficiency.
- The expressiveness of the query language allows efficient query processing by first searching for the components which match the strings specified in the query and subsequently evaluating which of these components satisfy the structure part of the query.
- For eg: the query [(*section) with (‘holocaust’)] which searches for sections , subsections or subsubsections which contain the word ‘holocaust’.
- A simple query processing strategy is to traverse the inverted list for the term ‘holocaust’ and for each entry in the list (which indicates an occurrence of the term ‘holocaust’ in the text)

- Search the hierarchical index looking for sections, subsections and subsubsections containing the occurrence of the term.
- A more sophisticated query processing strategy as follows:
- For the first entry in the list for ‘holocaust’ ,search the hierarchical index as before.
- This implies traversing down the hierarchy until no more successful matches occur.ie the bottom of the hierarchy is reached.
- Let the last matching structural component be reffered to as the innermost matching component.

- Once this first search is concluded, do not start all over again for the following entry in the inverted list. Instead verify whether the innermost matching component also matches the second entry in the list.
- If it does ,immediately conclude that the larger structural component above it also do.
- Proceed then to the third entry in the list and so on.
- The query processing is accelerated because only the nearby nodes in the list need to be searched at each time. This is the reason for the label proximal nodes.

- The model based on proximal nodes allows us to formulate queries which are more complex than those which can be formulated in the model based on non-overlapping lists.
- To speed up query processing only nearby (proximal)nodes are looked at which impose restrictions on the answer set retrieved.