Models for Retrieval and Browsing

- Structural Models and Browsing

Berlin Chen 2003

Reference:
1. Modern Information Retrieval, chapter 2
Structured Text Retrieval Models

• Structured Text Retrieval Models
  – Retrieval models which combine information on text content with information on the document structure
  – That is, the document structure is one additional piece of information which can be taken advantage

• E.g: Consider the following information need
  – Retrieve all docs which contain a page in which the string ‘atomic holocaust’ appears in italic in the text surrounding a Figure whose label contains the word earth

    Too many doc retrieved!

    • [‘atomic holocaust” and ‘earch’]
    • Or a structural (more complex) query instead

    data retrieval?
    • same-page( near( ‘atomic holocaust’, Figure( label( ‘earth’ )')))
Structured Text Retrieval Models

• Drawbacks
  – Difficult to specify the structural query
    • An advanced user interface is needed
  – Structured text retrieval models include no ranking
    (open research problem!)

• Tradeoffs
  – The more expressive the model, the less efficient is its query evaluation strategy

• Two structured text retrieval models are introduced here
  - Non-Overlapping Lists
  - Proximal Nodes
Basic Definitions

• **Match point**: the position in the text of a sequence of words that match the query
  – Query: “atomic holocaust in Hiroshima”
  – Doc $d_j$: contains 3 lines with this string
  – Then, doc $d_j$ contains 3 match points

• **Region**: a contiguous portion of the text

• **Node**: a structural component of the text such as a chapter, a section, a subsection, etc.
  – That is, a region with predefined topological properties
Non-Overlapping Lists

• **Idea**: divide the whole text of a document in non-overlapping text regions which are collected in a list
  
  – Multiple list generated
  
  • A list for chapters
  
  • A list for sections
  
  • A list for subsections

1. Kept as separate and distinct data structures

2. Text regions from distinct list might overlap!

$L_0$ — Chapter

$L_1$ — Sections

$L_2$ — SubSections

$L_3$ — SubSubSections
Non-Overlapping Lists

• Implementation:
  – A single inverted file build, in which each structural component stands as an entry in the index
  – Each entry has a list of text regions as a list occurrences
  – Such a list could be easily merged with the traditional inverted file

• Example types of queries
  – Select a region which contains a given word
  – Select a region A which does not contain any other region B
  – Select a region not contained within any other region
Inverted Files

• **Definition**
  – An inverted file is a word-oriented mechanism for indexing a text collection in order to speed up the searching task

• **Structure of inverted file**
  - *Vocabulary*: is the set of all distinct words in the text
  - *Occurrences*: lists containing all information necessary for each word of the vocabulary (text position, frequency, documents where the word appears, etc.)
Inverted Files

• Text:

That house has a garden. The garden has many flowers. The flowers are beautiful

• Inverted file

<table>
<thead>
<tr>
<th>Vocabulary</th>
<th>Occurrences</th>
</tr>
</thead>
<tbody>
<tr>
<td>beautiful</td>
<td>70</td>
</tr>
<tr>
<td>flowers</td>
<td>45, 58</td>
</tr>
<tr>
<td>garden</td>
<td>18, 29</td>
</tr>
<tr>
<td>House</td>
<td>6</td>
</tr>
<tr>
<td>....</td>
<td>....</td>
</tr>
</tbody>
</table>
Proximal Nodes

Navarro and Baeza-Yates, 1997

- **Idea**
  - Define a strict hierarchical index over the text. This enriches the previous model that used flat lists
  - Multiple index hierarchies might be defined
  - Two distinct index hierarchies might refer to text regions that overlap

- Each indexing structure is a strict hierarchy composed of
  - chapters, sections, subsections, paragraphs or lines

- Each of these components is called a node
  - Each node is associated with a text region
• **Features**
  - One node might be contained within another node
  - But, two nodes of a same hierarchy cannot overlap
  - The inverted list for words complements the hierarchical index
Proximal Nodes

• Query Language in regular expressions
  – Search for strings
  – References to structural components
  – Combination of these

• An example query: \[(*\text{section}) \text{ with } (“\text{holocaust”})]\]
  – Find the sections, the subsections, and the
    subsubsections that contain the word “holocaust”
Proximal Nodes

• Simple Query Processing
  – Traverse the inverted list for “holocaust” and determine all match points (all occurrence entries)
  – Use the match points to search in the hierarchical index for the structural components
Proximal Nodes

• Sophisticated query processing
  – Get the first entry in the inverted list for “holocaust”
  – Use this match point to search in the hierarchical index for the structural components until innermost matching component (the smallest one) found
  – Check if innermost matching component includes the second entry in the inverted list for “holocaust”
  – If it does, check the third entry and so on. If not, traverse up to higher nodes then traverse down ....
  – This allows matching efficiently the nearby (or proximal) nodes
Proximal Nodes

• **Conclusions**
  – Model allows formulating queries that are more sophisticated than those allowed by non-overlapping lists
  – To speed up query processing, nearby nodes are inspected
  – Types of queries that can be asked are somewhat limited (all nodes in the answer must come from a same index hierarchy!)
  – Model is a compromise between efficiency and expressiveness
Models for Browsing

• **Premise**: the user is usually interested in browsing the documents instead of searching (specifying the queries)
  - However, the goal of a searching task is clearer in the mind of the user than the goal of a browsing task

• Three types of browsing discussed here
  - Flat Browsing
  - Structure Guided Browsing
  - The Hypertext Model
Flat Browsing

• Documents represented as dots in
  – A two-dimensional plane
  – A one-dimensional plane (list)

• **Features**
  – Glance here and there looking for information within documents visited
    • Correlations among neighbor documents
  – Add keywords of interest into original query
    • Relevance feedback or query expansion
  – Also, explore a single document in a flat manner (like a web page)

• **Drawbacks**
  – No indication about the context where the user is
Structure Guided Browsing

• Documents organized in a structure as a directory
  – Directories are hierarchies of classes which group documents covering related topics
  – E.g.: “Yahoo!”

• Same idea applied to a single document
  – Chapter level, section level, etc.
  – The last level is the text itself (flat!)
  - **A good UI needed** for keeping track of the context

• Additional facilities provided when searching
  - **A history map** identifies classes recently visited
  - **Occurrences** (of terms) in a global context
The Hypertext Model

• **Premise**: communication between writer and user
  – A sequenced organizational structure lies underneath most written text
  – The reader should not expect to fully understand the message conveyed by the writer by randomly reading pieces of text here and there
  – Sometimes, we even can’t capture the information through sequential reading of the whole text
  • E.g.: a book about “the history of the wars” is organized chronologically, but we only interested in “the regional wars in Europe”
    – Wars fought by each European country
    – War fought in Europe in chronological order

Rewrite the book?  
Or defining a new structure?
The Hypertext Model

• **Hypertext**
  – A high level interactive navigational structure allowing users to browse text non-sequentially
  – Consist of *nodes* correlated by directed links in a graph structure
    • A *node* could be a chapter in a book, a section in an article, or a web page
    • Links are attached to specific strings inside the nodes

• Hypertexts provide the basis for HTML and HTTP
  – HTML: hypertext markup language
  – HTTP: hypertext transfer protocol
The Hypertext Model

• **Features**
  - The process of navigating the hypertext is like a traversal of a directed graph

• **Drawbacks**
  - *Loose in hyperspace*: the user will lose track of the organizational structure of the hypertext when it is large
    - A hypertext map shows where the user is at all times (graphical user interface design)
  - But, the user is restricted to the intended flow of information previously convinced by the hypertext designer
    - Should take into account the needs of potential users

Analyzing before implementation

Guiding tools needed (hypertext map)
Trends and Research Issues

• Three main types of IR related products and systems
  - Library systems
  - Specialized retrieval systems
  - The Web

• Library systems
  – Much interest in cognitive and behavioral issues
    • Oriented particularly at a better understanding of which criteria the users adopt to judge relevance (most systems here adopt Boolean model)
      – Ranking strategies
      – User interface design
  – How to implement
Trends and Research Issues

• **Specialized retrieval systems**
  – E.g. LEXIS-NEXIS: a system to access a very large collection of legal and business documents
  – How to retrieve almost all relevant documents without retrieving a large number of unrelated documents
    • Sophisticated ranking algorithms are desirable
Trends and Research Issues

• **The Web**
  
  – User does not know what he wants or has great difficulty in properly formulating his request
  
  – Study how the paradigm adopted for the user interface affects the ranking
  
  – The indexes maintained by various Web search engine are almost disjoint
    
    • The intersection corresponds to less than 2% of the total number of page indexed
  
  • **Meta-search**
    
    • Search engines which work by fusing the ranking generated by other search engines