Indexing and Abstracting

Lecture 10 – Index Structure

Kuang-hua Chen
Department of Library and Information Science
National Taiwan University

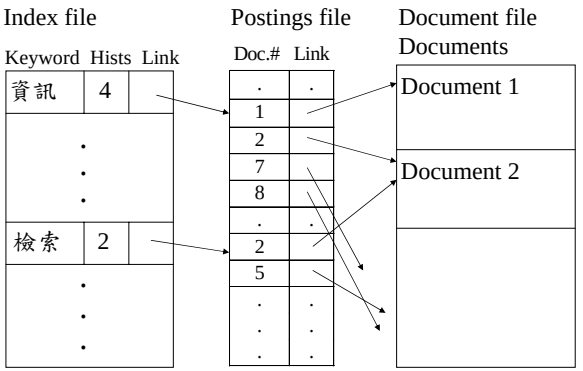
Structures for Index Files

- Inverted files
- Signature files
- PAT (Patricia Tree)

Inverted Files

- Each document is assigned a list of keywords or attributes.
- Each keyword (attribute) is associated with operational relevance weights.
- An inverted file is the sorted list of keywords (attributes), with each keyword having links to the documents containing that keyword.
- Penalty
  - the size of inverted files ranges from 10% to 300% of more of the size of the text itself
  - need to update the index as the data set changes

Sorted Array for Inverted Files

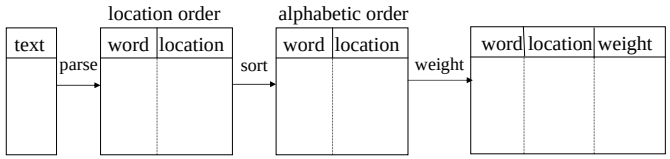


Structures for Inverted Files

- Sorted Arrays
  - store the list of keywords in a sorted array
  - using a standard binary search
  - advantage: easy to implement
  - disadvantage: updating the index is expensive
- Hashing Structures
- B-tree
- Combinations of these structures

Sorted Arrays

1. The input text is parsed into a list of words along with their location in the text. (time and storage consuming operation)
2. This list is inverted from a list of terms in location order to a list of terms in alphabetical order.
3. Add term weights, or reorganize or compress the files.



Inversion of Word

term	recon
pap	1
report	1
novel	1
technique	1
literat	1
result	1
technique	1
index	1
⋮	⋮
report	2
charact	2
human	2
being	2
ab	

sort →

term	recno
ab	2
being	2
charact	2
human	2
index	1
literat	1
novel	1
pap	1
report	1
report	2
result	1
technique	1
technique	1
⋮	

remove duplicates →

term	recno	freq
ab	2	1
being	2	1
charact	2	1
human	2	1
index	1	1
literat	1	1
novel	1	1
pap	1	1
report	1	1
report	2	1
result	1	1
technique	1	2
⋮		

Dictionary and Posting File

Dictionary

term	number of postings	offset
ab	1	.
being	1	.
charact	1	.
human	1	.
index	1	.
literat	1	.
novel	1	.
pap	1	.
report	2	.
result	1	.
technique	1	.
.		.
.		

Posting File

(document #, frequency)

2	1

1	1

1	1	2	1

1	2

Idea: the file to be searched should be as short as possible split a single file into two pieces

Signature Files

- basic idea: inexact filter
  - discard many of nonqualifying items
  - qualifying items definitely pass the test
  - “false hits” or “false drops” may also pass accidentally
- procedure
  - Documents are stored sequentially in “text file”.
  - Their signatures (hash-coded bit patterns) are stored in the “signature file”.
  - Scan the signature file, discard nonqualifying documents, and check the rest, when a query arrives.

Merits of Signature Files

- faster than full text scanning
  - 1 or 2 orders of magnitude faster
- modest space overhead
  - 10-20% vs. 10-300% (inversion)
- insertions can be handled more easily than inverted file
  - append only
  - no reorganization or rewriting

Basic Concepts

- Use superimposed coding to create signature
- Each document is divided into logical blocks
- A block contains D distinct non-common words
- Each word yields “word signature”
- A word signature is a F-bit pattern, with m 1-bit
- The word signatures are OR’ed to form block signature
- Block signatures are concatenated to form the document signature

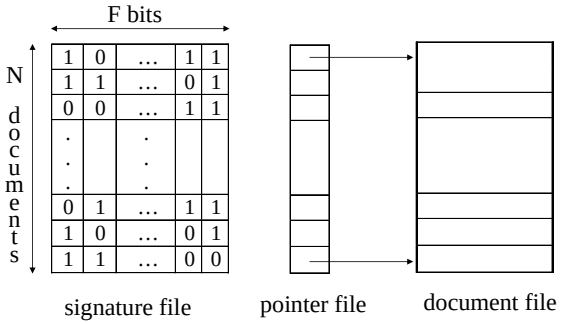
Basic Concepts (Continued)

- Example (D=2, F=12, m=4)

word	signature			
資訊	001	000	110	010
檢索	000	010	101	001
block signature	001	010	111	011

- Search
  - Use hash function to determine the *m* 1-bit positions.
  - Examine each block signature for 1’s bit positions that the signature of the search word has a 1.

Sequential Signature File (SSF)



Bit-block Compression (BC)

- Data structures
  - The sparse vector is divided into groups of consecutive bits (bit-blocks).
  - Each bit block is encoded individually.

Bit-block Compression (BC)

Part I. It is one bit long, and it indicates whether there are any "1"s in the bit-block (1) or the bit-block is (0). In the latter case, the bit-block signature stops here.

0000 1001 0000 0010 1000

0 1 0 1 1

Part II. It indicates the number *s* of "1"s in the bit-block. It consists of *s*-1 "1" and a terminating zero.

10 0 0

Part III. It contains the offsets of the "1"s from the beginning of the bit-block.

0011 10 00

說明：b = 4, 距離爲 0, 1, 2, 3, 編碼爲 00, 01, 10, 11

block signature: 01011 | 10 00 | 00 11 10 00

Search in BC

- Search "資訊"
  - 資訊 ==> 0000 0000 0000 0010 0000
  - the #3 bit-block
  - signature 01011 | 10 0 0 | 00 11 10 00
  - OK, there is at least one setting in the #3 bit-block
  - Check furthermore. "0" tells us there is only one setting in the #3 bit-clock. Is it the #2 bit?
  - Yes, "10" confirms the result

PAT Trees and PAT Arrays

- Problems of tradition IR models
  - Documents and words are assumed
  - Keywords must be extracted from the text (indexing)
  - Queries are restricted to keywords
- New indices for text
  - A text is regarded as a long string
  - Each position corresponds to a semi-infinite string (*sistring*)
  - no keywords

Semi-Infinite Strings (Sistring)

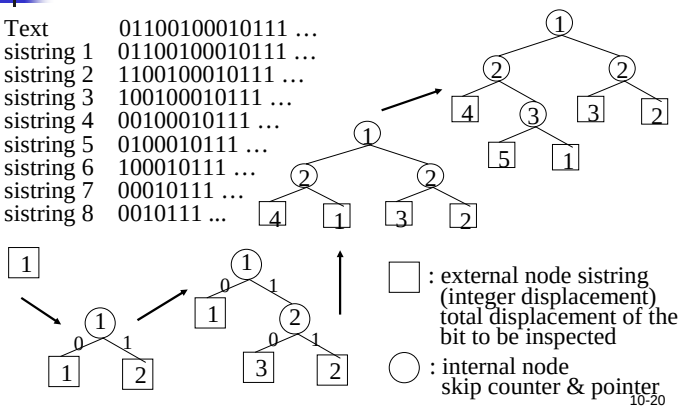
- Example
  是寂寞　慢慢佔領我的心　長夜裡愈來愈冷清　回憶裡愈來愈孤寂　是後悔　慢慢侵蝕我的心　抹去了最後的淚滴　從今後　不再爲誰哭泣
- sistring 1 是寂寞　慢慢佔領我的心 ...
  sistring 2 寂寞　慢慢佔領我的心　...
  sistring 4　慢慢佔領我的心　長夜 ...
  sistring 8　領我的心　長夜裡愈來愈 ...
  sistring 22　回憶裡愈來愈孤寂　是後 ...

PAT Tree

- PAT Tree
  - Patricia tree constructed over all the possible sistrings of a text
- Patricia tree
  - a digital tree where the individual bits of the keys are used to decide on the branching
  - each internal node indicates which bit of the query is used for branching
    - absolute bit position
    - a count of the number of bits to skip
  - each external node is a sistring

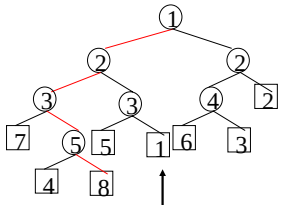
Example

Text 01100100010111 …
sistring 1 01100100010111 …
sistring 2 1100100010111 …
sistring 3 100100010111 …
sistring 4 00100010111 …
sistring 5 0100010111 …
sistring 6 100010111 …
sistring 7 00010111 …
sistring 8 0010111 ...

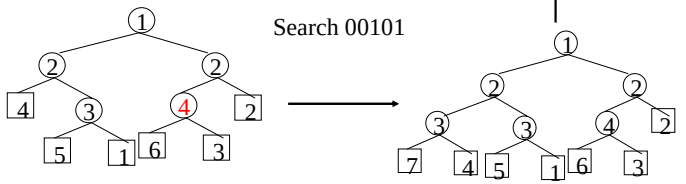


Example (continued)

Text	01100100010111 …
sistring 1	01100100010111 …
sistring 2	1100100010111 …
sistring 3	100100010111 …
sistring 4	00100010111 …
sistring 5	0100010111 …
sistring 6	100010111 …
sistring 7	00010111 …
sistring 8	0010111 ...



Search 00101

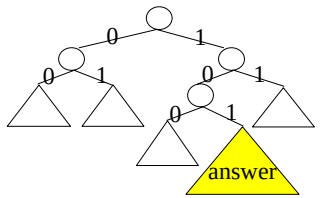


Indexing Points

- The above example assumes every position in the text is indexed.
  n external nodes, one for each position in the text
- word and phrase searches
  sistrings that are at the beginning of words are necessary
- trade-off between size of the index and search requirements

Prefix searching

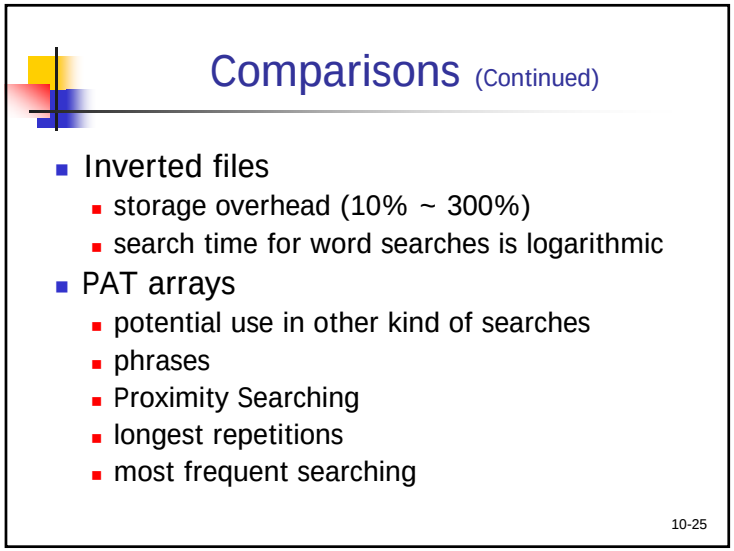
- Idea
  every subtree of the PAT tree has all the sistrings with a given prefix
- Search
  - proportional to the query length
  - exhaust the prefix or up to external node



Search for the prefix "101" and its answer

Comparisons

- Signature files
  - Use hashing techniques to produce an index
  - Advantage
    - storage overhead is small (10%-20%)
  - Disadvantages
    - the search time on the index is linear
    - some answers may not match the query, thus filtering must be done



Comparisons (Continued)

- Inverted files
  - storage overhead (10% ~ 300%)
  - search time for word searches is logarithmic
- PAT arrays
  - potential use in other kind of searches
  - phrases
  - Proximity Searching
  - longest repetitions
  - most frequent searching