Indexing: The Most Vital Tool for Information Retrieval

Pooja Gupta
Maharaja Agarsain College,
New Delhi

A.K. Sharma
YMCA Institute of Engineering,
Faridabad

J.P. Gupta
JayPee University,
Noida

Abstract: *The rate of change in information over WWW is high, so in the current competitive environment, it has become essential for every enterprise/organization to take proper decision or to provide good services to the user by mean of accurate information as quickly as possible. Information kept in the local database should be organized in some structured form called index to facilitate the user with fast and accurate responses. Indexes to databases provide additional control to access or retrieval efficiency and optimizing the time and computing requirement. In this paper, some of the important indexing techniques is being discussed and a comparison, on the basis of various factors, has been done.*

Keywords: indexing, search engine, WWW, databases, information retrieval, algorithm, user query, Singature file, inverted index, PAT trees etc.

1. INTRODUCTION

The WWW [7] has become a universal repository of information in form of documents that allows its users to distribute or share the information accessible widely. The size of WWW is increasing exponentially, so each organization/business enterprise maintain its own database by extracting information from WWW according to its requirement; to later fulfil its clients/users with the relevant information retrieval for which they are interested, by applying some efficient search technique on the databases.

Information retrieval (IR) is a technique of searching information within documents, databases and WWW. Mainly, it deals with the representation, storage and organization of the documents to provide the user ease of accessing the information within the documents in which he/she is interested. With the competition in every area academic and enterprise; for each enterprise it has become essential to incorporate new and/or updated documents in the databases, thus, the need for retrieving valuable information [12] is increasing very rapidly. The storage of information in IR systems is carried in some structured format keeping in view the faster response to the user information retrieval query. The general model for the IR system [9] is shown in Fig. 1.

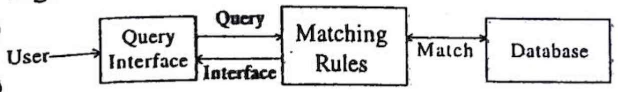

Fig. 1. General Model of Information Retrieval System

The process of information Retrieval involves a user forming a query with required keywords to specify the information to be retrieved and thereafter, posting it to the database. Based on the set of matching rules, database is searched and retrieved results are presented back to the user.

It has become a major responsibiliy of IR system or tool to systematically arranged information in order not only to respond to the query faster but also to get information relevant to the user. Indexing is one such technique that is used by various IR tools to optimize time and computing requirement to answer the user query. Search engine is one such tool that maintains the information downloaded from the WWW in local databases and indexes it to provide the specific and quick response to the user query.

Indexing increases document retrieval efficiency, thereby, optimizes the cost of retrieving the documents. Adding indexes to the databases provide additional, controlled ways to access information. Retrieval from the index is consistent and more accurate as it is based on controlled search domain. The following are some benefits of using indexes with the databases:

1. Increased accuracy and speed of information retrieval.
2. Improved response time and services to the user.
3. Quick additional and subsequent access of new documents to the user.

Indexed database search improves the performance by decreasing the search space, only the indexed is searched to find out a match for the user query and if a match is found, the corresponding records in the main database are accessed and returned to the user; thus reduces the response time. If the same search is carried out over unindexed data storage, say sequentially it take good amount of time to mine the text and to get a suitable match, whereas index is some part of the text maintain systematically if searched will take less time, hence increases the speed of retrieval.

2. INDEXING MODELS

There are many models available for the indexing:

1. Signature file index models
2. Inverted Index model
3. Pat Array model

2.1 Signature File Model

Signature file method seems to be a reasonable choice for dynamic environment that has large databases with large insertion rates but few deletions and updates as discussed by C. Faloutsos [1,2] and by S Buttcher. CLA Clarke [11]. This method keeps the whole text in one file known as the 'message file'. This file is divided into number of blocks, for each block an abstraction called 'signature' containing some of the information of the original text is created; usually using by applying some hash functions on the original text of the block and stored in the second file named 'signature file'.

The Structure signature file index is shown in Fig.2 and is comprises of Signature File over Message file as:

• Message file/Text	(the text divided into the number of blocks)
• Signature File	(signature for each block in message file)

This signature file acts as the index file and i searched to response the user query.

2.1.1 *Query Resolution in signature file Model*

To resolve a query, the signature file is scanne sequentially and many non-qualifying blocks are rejecte immediately. This method gurantees that all qualifyin blocks can be identified. At the same time this may b noted that some non-qualifying blocks will pass th signature test known as 'false drops'. To control th number of false drops the enough size for signature i used, generally 10% of the text file.

There are various signature extraction methods:

• Word Signature	(signatures are generate for each different word i the text)
• Superimposed coding	(text is in the form o block containing non common words an signature is generated fo each block)

2.2 Inverted File Model

According to the available literature on inverte text model by Navarro [3] and Ricardo and Charles i [4,5], the inverted file model treats the text file as sequence of words. This model scans the whole tex word by word and builds a table that contains al different words known as 'vocabulary' and store th every occurrence of each words on a list. This list c occurrence of each word is kept in order by position i the text. Thus, the structure has 2 elements as show Fig. 3 and has two parts:

• Vocabulary	(the set of all different worc in the text)
• Occurrence	(it is the list of position of tex

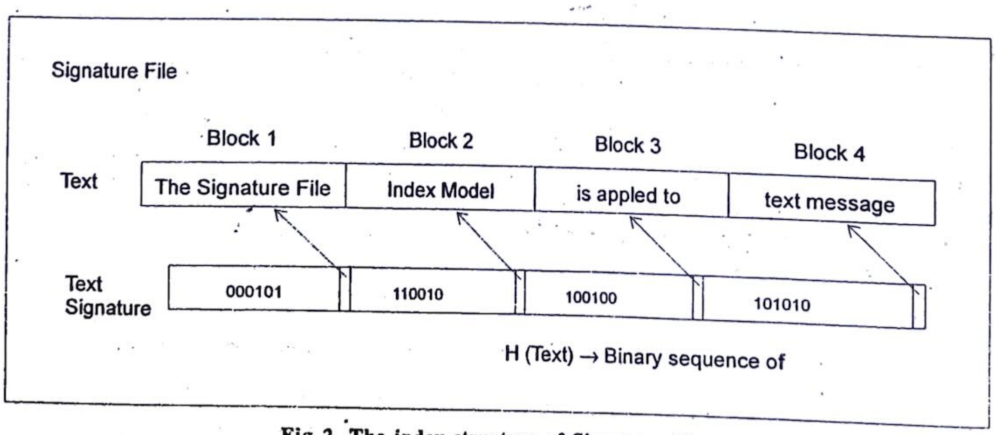

Fig 2. The index structure of Signature file

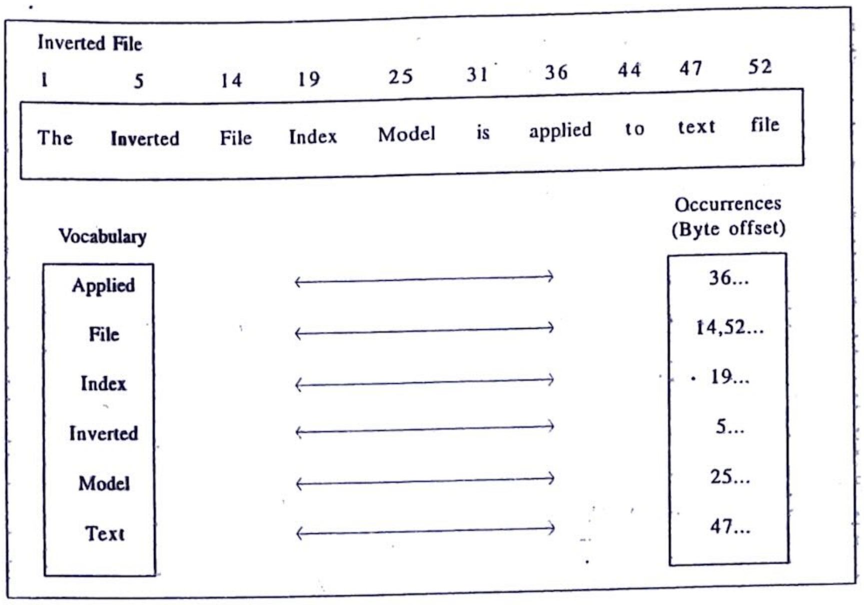

Fig. 3. The index structure of Inverted File

Inverted file index model maintains the index in main memory and if the size of the index exhausts the partial index is kept in the main memory known as 'dump'. The merging of dumps is done periodically that concatenate the lists of occurrences of each world in linear time. Partial dumps are merged until the complete index is obtained. The text address always increases as the scanning goes on, so the insertion in the list of occurrance always done at the end of the list. Insertion takes O(n) CPU time, same is for the deletion.

2.2.1 *Query Resolution in Inverted File Model*

To resolve a query occurrence list respective to each word is inserted in the index. Index is scanned sequentially to get a match and corresponding access from the vocabulary is done. For more complex queries that are form of number of words, number of occurrence lists are in result; the smallest list is selected and then the next large list is intersected with it and the resultant with next be intersected with the next occurrence list and finally the intersection of all the list will be the answer to the query.

2.3 PAT Tree Model

The Gonnet [6] discussed that PAT array or PAT trees are suitable for more complex inquiry. PAT tree treats text of a document as an array of characters. It does not use the concept of words and need not to know about the structure of the text; is a shallowest tree structure that allows ~~very~~ efficient searching with pre-processing.

This technique use the text as a single array of characters, numbered sequentially from one onwards. It is not relevant if same part has been repeated in the text. It extracts semi-infinite string as a subsequence of characters from the fully array, starting from a given point but moving to the right if necessary. If this semi-infinite string goes beyond the end of the actual text, special null characters different from the text is appended to the end. These strings are added to the PAT tree. Every position in the text is indexed and PAT tree has 'n' external nodes, one for corresponding to each position in the text. The PAT tree stores they key values at the external nodes whereas the internal nodes have the skip counter and pointers to the subtress but no key values. For a text size of 'n', there will be 'n' external nodes and 'n-1' internal nodes. Every subtree has all these semi-infinite strings with a given prefix.

PAT tree is a digital tree where the individual bits of the keys are used to take branching decision. The bit '0' branch to the left subtree and '1' branch to the right subtree, i.e. PAT tree is binary digital tree.

2.3.1 *Construction of PAT tree*

To construct a PAT tree first the Patricia trie is build. In Patricia trie [10] every node has a 'bit index' i.e. a number associated with node and all comparisons are done at this place. If a node 'X' has a bit index '2' so while searching the comparison on bit 2 of the key is done and branch left if key at place is '0' and right if

'1'. The ○ . represents the internal node □ and represent the external node.

e.e.: lets the text 'TO' be added to the Patricia trie

INDEX	:	4 3 2 1 0
T	:	0 0 1 0 0
O	:	0 1 1 1 1

Inserting 'T' : Since this is the first alphabet so new trie is building:

1. find the bit index i.e. the index of leftmost '1' i.e. '2' in this case

2. build links-for new trie node leftmost link is unconnected and the right link always leads back up to itself.

The trie after insertion is shown in Fig. 4.

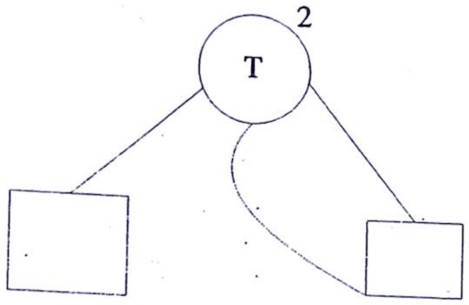

Fig. 4. The trie after inserting 'T'

Inserting 'O': (0 1 1 1 1)

1. search for 'O' on existing trie starts at 'T' that has bit index '2' so the bit '2' of 'O' is taken i.e. '1'; so branch to right which lead to 'T' that implies 'T' is the closest node.

2. find the lcftmost bit where 'T' and 'O' differ i.e. at bit '3', the bit index of will be '3' greater than the bit index of 'T'.

3. as the bit index is greater than "T" so 'O' must be inserted above 'T'.

4. now, at bit index '3' O has 1, so right link point back to 'O' and left will to 'T' as shown in Fig. 5.

On the similar basis complete trie is build on text and then compact it to PAT tree by removing the similar node to different semi-infinite strings.

2.3.2 *Query Resolution in PAT Tree Model*

Various type of query processing can be done by traversing through PAT tress.

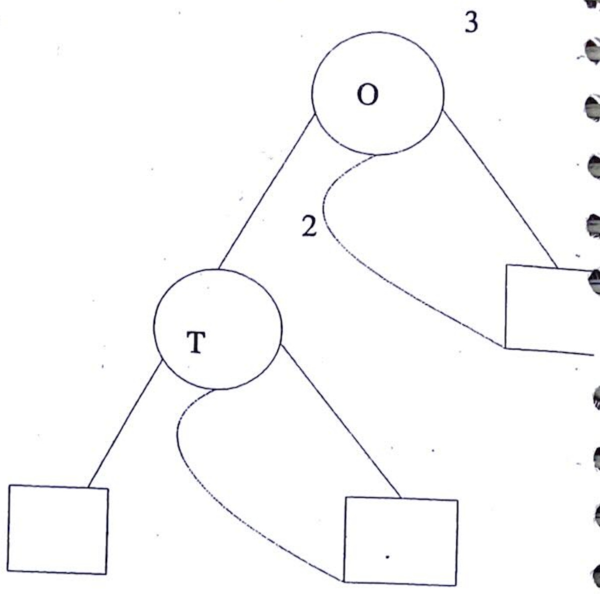

Fig. 5. Trie after insertion of 'O'

Prefix Searching	(searching of prefix in the thre up to point where we reach a external node or exhaust)
Proximity Searching	(finding all places where a strin 'sl' is at most a fix number c characters away from anothe string say 's2')
Range Searching	(searching for all the string within a certain range of value i.e.. in lexicographical range)

Similarly the search can be carried out to find th "Longest repetition searching", "Most significan searching" and "Regular expression searching' Searching is done in tree by traversing from root t leaf, if a leaf is searched the search is successful, if n corresponding pointers is found from the internal nod search is fail. To improve the efficiency PAT trees ar represented in array form as PAT arrays.

3. COMPARATIVE ANALYSIS OF INDEXIN TECHNIQUES

The comparative analysis of three indexing mod has been done on the basis of various parameters liste in Table 1.

4. CONCLUSION

It has been concluded that above discusse indexing models have their usage in different types c application and some advantages and disadvantages ove each other. The insertion in the signature file is easy a for each new message its signature is created an

Table 1. Comparative analysis of Indexing Techniques

Paramter	Signature file index	Inverted Index Model	PAT Array Model
Structure of the index	World-oriented index structure based on hashing functions on hashing functions	Character position oriented index that facilitate direct access to matching text positions	Data structure has indexes for all character of text called index point
Space requirement	Extra space is required to store signatures	Space required for vocabulary is small but for occurrence is more i.e. O(n)	Extra space required for nodes, for nodes, for internal nodes 3-4 words and for external nodes I word
Space overhead %	10-20%	30-40%	120%-240%
Searching techniques	Using Brute Force or can be using Knuth-Morris-Pratt (KMP) and Boyer-Moore (BM)	Single word queries by hashing or B-trees and prefix or range queries by binary search or B-trees	Binary Tree Traversal
Search time	Linear O(n)	Logarithmic O(log n) for single word and for longer phrases it is O (√ n)	Using PAT array logarithmic O(n log n)
Performance over text size	Not good for large texts as search time is linear	Not good for large text	Good for large text
Results/usage	Chances to get false drop as result i.e. answers do not match the query. Used in offices generally to index messages in form of text files	Matched results respective to query, use to solve phrases, proximity or Boolean operation.	Relevant results and potentially used for various queries such as searching for phrases, regular expression searching, approximate string searching, longest repetition and most frequent searching

appended at the end of the signature file whereas in inverted index model insertion require re-writing of the index. Based on the aforesaid argument the inverted index technique is expensive for optical disc whereas in the signature index model scanning is much faster as it is much smaller.

PAT tree perform much better than Signature file and Inverted index only if it is kept in the main memory otherwise falls drastically. The PAT tree can not be used for the text documents and to structure documents where searching is done on word basis as it treats the text as combination of characters.

References:

1. C. Faloutsos and S. Christodoulakis, "Description and performance analysis of signature file methods," ACMTOIS, 5(3), pp. 237-257, 1987.

2. C. Faloutsos and R. Chan, "Text access methods for optical and large magnetic disks design and performance comparison," In Proc. *QFPXDB'88*, pages 280-293, Los Angeles, CA, USA, 1988.

3. M. Araujo, G. Navarro, and N. Ziviani, "Large text searching allowing error," *In Pro. MP '97, pages 2-20*; Vafparaiso, Chile, 1997. Carleton University Press.

4. Ricardo Baeza-Yates, *Modern information Renieval*, New York, ACM Press, 1999, pp. 191-198.

5. Charles T. *Meadow, Text Information Retriewul Sysrems (2nd edjfion)*. San Diego: Academic Press, 2000, pp. 13 1-32.

6. Gonnet, G.H. et al, "New indices for text: FAT trees and PAT arrays," *Information Retrievd: Data Structure and Algorilhms (Frakes. W.E. and Bueza-Yates, R.A. (edss.))*, Prentice-Hall, New Jersey, pp. 66-82, 1992.

7. I. Elizabeth Shanthi and R. Nadarajan, "An Index Structure for Fast Query Retrieval in Object Oriented Data Bases Using Signature Weight Declustering", *Information technology Journal*, vol-8, issue-3, pp. 275-283, 2009.

8. Internet:http://www.google.co.in/

9. The URL http://people.ischool.berkeley.edu/~buckland/paper/analysis/node2.html.

10. CS209-HOW TO Series: goanna.cs.rmit.edu.au/~stbird/Tutorials/patricia.pdf.

11. S. Buttcher, CLA Clarke, "Indexing time vs. query time: trade-offs in dynamic information retrieval systems", Proceedings of the 14th ACM international in 2005.

12. Tzi-cker Chiueh, Lan Huang, "Efficient Real-Time Index Updates in Text Retrieval Systems" in (1999).

❏