International Journal of Engineering Research & Management Technology
ISSN: 2348-4039
Email: editor@ijermt.org November- 2014 Volume 1, Issue-6 www.ijermt.org

Exploration of Text Retrieval Algorithm

Navin Ahlawat
Head of Department
Computer Application
SRM University, NCR Campus
Ghaziabad

ABSTRACT
Web users usually suffer from the difficulties of finding desirable and accurate information on the Web due to two problems of low precision and low recall caused by significant and rapid growth in the amount of information and the number of users. Thus there is a need to develop more efficient and effective techniques to satisfy the increasing demands of Web users, such as retrieving the desirable and related information, creating good quality Web communities, extracting the informative knowledge from the available information, capturing the underlying usage pattern from the Web observation data, recommending user customized information to offer better Internet service, and furthermore mining valuable business information from the common or individual customer navigational behavior as well.

Key words: accurate information, web, business information

INTRODUCTION
We are in an age often referred to as the information age. In this information age, because we believe that information leads to power and success, and thanks to sophisticated technologies such as computers, satellites, etc., we have been collecting tremendous amounts of information. Initially, with the advent of computers and means for mass digital storage, we started collecting and storing all sorts of data, counting on the power of computers to help sort through this amalgam of information. Unfortunately, these massive collections of data stored on disparate structures very rapidly became overwhelming. This initial chaos has led to the creation of structured databases and database management systems (DBMS). The efficient database management systems have been very important assets for management of a large corpus of data and especially for effective and efficient retrieval of particular information from a large collection whenever needed. The proliferation of database management systems has also contributed to recent massive gathering of all sorts of information. Today, we have far more information than we can handle: from business transactions and scientific data, to satellite pictures, text reports and military intelligence. Information retrieval is simply not enough anymore for decision-making. Confronted with huge collections of data, we have now created new needs to help us make better managerial choices. These needs are automatic summarization of data, extraction of the “essence” of information stored, and the discovery of patterns in raw data.

Traditional Information Retrieval techniques become inadequate for the increasingly vast amounts of text data. In this paper, we show a method of query processing, which retrieve the documents containing not only the query terms but also documents having their synonyms. The method performs the query processing by retrieving and scanning the inverted index document list. We show that query response time for conjunctive Boolean queries can be dramatically reduced, at cost in terms of disk storage, by applying range partition feature of Oracle to reduce the primary memory storage space requirement for looking the inverted list. The proposed method is based on fuzzy relations and fuzzy reasoning to retrieve only top ranking documents from the database. To group similar documents Suffix tree clustering is used.

INVERTED INDEX FILES

User accesses the Information Retrieval system by submitting a query; the Information Retrieval system then tries to retrieve all documents that are "relevant" to the query. For this, the documents contained in the archive are analyzed to provide a formal representation of their contents through inverted indexing, where a surrogate describing the document is stored in an index, while the document itself is stored in the collection or archive. An index is a structure that is used to map from query-able entities to indexed items. For example, in a database system an index is used to map from entities such as name and bank account to records containing data about those entities. To gain the speed benefits of indexing at retrieval time, we have to build the inverted index in advance.
The major steps to build the inverted index are:

- Collect the documents to be indexed.
- Tokenize the text, turning each document into a list of tokens.
- Do linguistic preprocessing, producing a list of normalized tokens, treated.

The concept of inverted index file is explained through an example given below. Consider two documents of the document corpus – Doc 1 and Doc 2 having the text content shown below:

Doc 1
Mahatma Gandhi's birthday is celebrated on 2 Oct. He is father of India.
He is a freedom fighter.

Doc 2
Indira Gandhi is the mother of India. She is a politician of great personality.

Given below are the steps to be followed to create the inverted index file.

1. Collect the documents to be indexed

Mahatma Gandhi's birthday is	Indira Gandhi is the mother of

Doc 1 ***Doc 2***

2. Tokenize the text, turning each document into a list of tokens

Mahatma	Gandhi's	birthday	is	….

Doc 1

3. Do linguistic preprocessing, producing a list of normalized tokens, treated as index terms

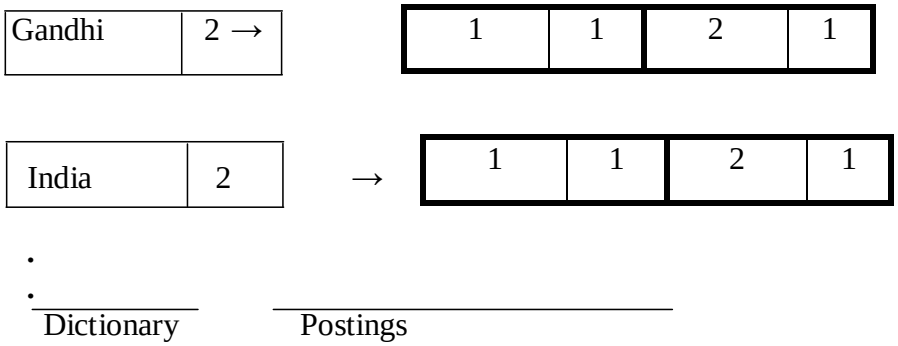
Mahatma	Gandhi	birthday	celebrate	….

4. Build an index by sorting and grouping

Index term	freq.
Mahatma	1

→

doc no.	Freq.
1	1



Inverted File Compression Techniques

Dictionary and the inverted index are the central data structures in Information Retrieval. A number of compression techniques for dictionary and inverted index are employed that are essential for efficient IR systems. There are two more subtle benefits of compression:

1) Efficient and maximum utilization of cache

Search systems use some parts of the dictionary and the index much more than others. For example, if the postings list of a frequently used query term t is cached in memory, then the computations necessary for responding to the one-term query t can be entirely done in memory. With compression, a lot more information can be fitted into main memory. Instead of having to expend a disk seek when processing a query with t, an access is made to its postings list in memory and decompress it. There are simple and efficient decompression methods, so that the penalty of having to decompress the postings list is small. As a result, the response time of the IR system can be decreased substantially.

2) Faster transfer of data from disk to memory

Efficient decompression algorithms run so fast on modern hardware that the total time of transferring a compressed chunk of data from disk and then decompressing it is usually less than transferring the same chunk of data in uncompressed form. For instance, it can reduce input/output (I/O) time by loading a much smaller compressed postings list, even when the cost of decompression is added. So, in most cases, the retrieval system runs faster on compressed postings lists than on uncompressed postings lists.

Compression algorithms for Integers

A number of compression algorithms for integers are discussed hereby.

a) Fixed, Linear - If the values are evenly distributed throughout the whole range, a direct binary representation is the optimal choice. The number of bits needed of course depends on the range. If the range is not a power of two, some tweaking can be done to the code to get nearer the theoretical optimum log2 (range) bits per value.

Table-1
Fixed Linear prefix adjusted codes with minimum average number of bits

Value	Binary	Adjusted	1&2	
0	000	00	000	H = 2.585 L = 2.666 (for flat distribution)
1	001	01	001	
2	010	100	010	
3	011	101	011	
4	100	110	10	
5	101	111	11	

Table 1 shows two different versions of how the adjustment could be done for a code that has to represent 6 different values with the minimum average number of bits. As can be seen, they are still both prefix codes, i.e. it is possible to (easily) decode them.

Table-2
Elias Delta Code Representation

Delta Code	Integer	Bits
1	1	1
010x	2-3	4
011xx	4-7	5
00100xxx	8-15	8
00101xxxx	16-31	9
00110xxxxx	32-63	10
00111xxxxxx	64-127	11

Table-3
Elias Gamma Code Representation

Gamma Code	Integer	Bits
1	1	1
01x	2-3	3
001xx	4-7	5
0001xxx	8-15	7
00001xxxx	16-31	9
000001xxxxx	32-63	11
0000001xxxxxx	64-127	13

The delta code is better than gamma code for big values, as it is asymptotically optimal (the expected codeword length approaches constant times entropy when entropy approaches infinity), which the gamma code is not. What this means is that the extra bits needed to indicate where the code ends become smaller and smaller proportion of the total bits as we encode bigger and bigger numbers. The gamma code is better for greatly skewed value distributions (a lot of small values).

Quick text retrieval algorithm

In this section, we will describe the algorithm for grouping the retrieved ranked documents that is based on the structure outlined in section 3.5.1.

Given below are explained, the steps of the algorithm to be performed.

- Consider a query *Q*. Ignore all the stop words from the search query. Final query is represented by set *A*.
- For each query term *t*,
  - if target partition of term table is not in main memory then bring to memory from disk
  - search the partition term table to locate *t*
  - Record the posting list (synonyms with their fuzzy thesaurus association) for term *t*, in the final list *X*
- Identify the fuzzy thesaurus *T* for each pair of index terms *(xi,xk) ε X*, where *t(xi,xk)* expresses the degree of association of *xi* with *xk*.
- Compute $B \leftarrow A \; o \; T$
- For each term $t \; \varepsilon \; X$
  - if target partition of word document table is not in main memory then bring to memory from disk
  - Search the partition word document table to get word_id for term *t*
  - Record the posting list (for each term *t*, get Elias γ codes representing document_id along with their term frequency value)
  - Decompress to get the actual document_ids
  - Record the document ids to the final relation *R* along with their term_ frequency
- Compute $D \leftarrow B \; o \; R$
- Inspect only those document_ids captured by some α-cut of *D*
- Bring the document table in memory
- For each $d \; \varepsilon \; D$
  - Look up the address of document d in document table b.
  - retrieve document *d*

CONCLUSION

Compressed, partitioned, in-memory Inverted Index to store text documents index terms is proposed which enables fast searching requiring less main memory. Through the use of compression and range partitioning, a small, faster representation for sorted lexicon of interest is achieved. Application of range partition feature on proposed data structure enables few relevant partitions against the given query (2-5 words) to be retrieved from the proposed data structure and hence saves time to decompress the small relevant data of interest. Sorted lexicon also permits rapid binary search for matching strings. Conjunctive queries are easily handled through the concept of fuzzy logic in retrieving the documents having high value of □-cut (threshold limit) in fuzzy set D for AND operation and non-zero value of □-cut for OR operation. The proposed method also retrieves the documents containing synonyms of the search query terms in the document.

REFERENCES

1. **[Agaian, 2012]** S. S. Agaian, K. A. Panetta, S. C. Nercessian and E. E. Danahy, "Boolean Derivatives With Application to Edge Detection for Imaging Systems," in *IEEE* Transactions *on Systems, Man, And Cybernetics*—Part B: Cybernetics, Vol. 40, No. 2, April 2010.
2. **[Cheng, 2007]** S. Cheng, W. Huang, Y. Liao and D. Wu, "A Parallel CBIR Implementation Using Perceptual Grouping Of Block-based Visual Patterns," in *IEEE International Conference on Image Processing – ICIP*, 2007, pp. V - 161 - V - 164.
3. **[Geusebroek, 2010]** J. M. Geusebroek, G. J. Burghouts, and A. W. M. Smeulders, "The Amsterdam library of object images," in *International Journal of Computer Vision*, 61(1), 103-112, January, 2005 .
4. **[Kekre, 2013]** Dr. H. B. Kekre, S. Thepade, P. Mukherjee, M. Kakaiya, S. Wadhwa, and S. Singh, "Image Retrieval with Shape Features Extracted using Gradient Operators and Slope Magnitude Technique with BTC," in *International Journal of Computer Applications (0975 – 8887) Volume 6– No.8, September 2010.*
5. **[Pratikakis, 2012]** I. Pratikakis, I. Vanhamel, H. Sahli, B. Gatos and S.J. Perantonis, "Unsupervised watershed-driven region-based image retrieval," in *IEE Proceedings on Vision, Image and Signal Processing*, Vol. 153, No. 3 , June 2006 pp. 313-322.
6. **[Wang, 2001]** J. Z. Wang, J. Li and G. Wiederhold, "SIMPLIcity: Semantics-Sensitive Integrated Matching for Picture Libraries," in *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Volume 23, No 9, Sept 2001, pp. 947-963.
7. **[V. V. Raghavan, 2013],** "Content-based image retrieval systems," in *IEEE Computer* 28(9), 1995, pp. 18-22.