DESIGNING INTEGRATED INFORMATION RETRIEVAL SYSTEM FOR FARSI TEXTS

S. M. KHALESIZADEH, Ph.D.
Industrial Engineering Department
Sharif University of Technology
email: khalessi@sharif.edu

A. RAHMANSETAYESH, M.S.
Industrial Engineering Department
Sharif University of Technology
email: setayesh@mehr.sharif.edu

Abstract - Extension of information together with the need for its use ine and appropriate time is one of the important goals in this century - the information century. The user query that is accessibility to the requested text information in a short time must be satisfied using effective techniques. This is performable under text compression and retrieval, which have been treated separately before. In this paper an algorithm, that considers the above two subjects as integrated (text integration), is introduced and implemented. From this point of view, the possibility of text optimization from two aspects can be applied. This article concentrates mainly on Farsi[1] texts.

Keywords - Text Retrieval, Compression, Search, Indexing, Farsi Texts, Information Retrieval.

INTRODUCTION

Nowadays, Information Technology is certainly one of the challenges affecting human community. The great and almost unbounded volume of available information causes its use and management to be performed in extensive and often complicated dimensions. In accordance with growth and extension of information volume parallel to necessity of handling information resources, accessibility of required information within a proper time is one of the most important and basic demands. In fact, to search and find desired information complying with user's query is of utmost importance.

Although the information supplied nowadays have adopted different formats of picture, sound, etc. the largest and mostly used information are non-structural texts. Meeting reports, letters, instructions, catalogs, etc. are some instances of miscellaneous text information available in organizational archive systems. Other examples are law maintenance, updating systems and email retrieval systems. So, information are saved in the computer in two formats including structural (database) and text as well as multimedia format. This article takes the text format information into consideration. According to the importance of this subject and the intensified needs in the country, this paper concentrates mainly on Farsi texts.

OBJECTIVES

As discussed before, this paper's objective is to present a method by which text integration (compression along with effective access) can undergo user's needs of fast retrieval together with less storage space consumption (optimization). Though this has been previously done for Latin languages, it is the first time to be implemented here in our country. Below definitions for the expressions used in the paper are given:

- Compression of text: the study of techniques for representing text in fewer bytes or bits.
- Coding: The substitution of text symbols by numeric codes with the aim of encrypting or compressing text.

Huffman coding: an algorithm for coding text in which the most frequent symbols are represented by the shortest codes.

- Index: a data structure built on the text to speed up searching.

The user's query for information retrieval may contain different goals that are summarized as follows:

a- Query for text classification.

b- Query for text summarization.

c- Query for text including a logic (Boolean) phrase.

d- Query for text including an exact specific pattern.

This article tries to present an algorithm and/or compression system to undergo cases(c) and specifically (d) of the above, in other words, to present an integrated system which has the two characteristics of text retrieval systems simultaneously.

IN PRACTICE

Compressing in text retrieval systems began with efforts done by [8]. In 1994 they defined a word-based compression model. Texts include some files and Huffman coding is used to compress them. Besides, the system uses full text retrieval for indexing, that is using all words in the text to get the key words list. After the key words are entered by the user, the system decodes the relevant texts and passes them to him. Because of using bits for compression, the decoding speed in this system is low [4].

Another system named Glimpse was developed in 1994 by Manber [3]. This system compresses the text, that is, it firstly classifies the text words within three groups: general, non-general and special. But the word list is independently used for indexing and compression. This system is capable of searching and retrieving texts, which include some key words from the user's query [3].

In 1997 Varadarajan & Chiueh developed another system called SASE [8]. This was the first system to use integrated compression and indexing, that is, word list for compression and indexing is the same. Indexing framework is like Glimpse system. The search method in this system is based upon key words.

In recent years, wide research in the field of compression and efficiency of text information retrieval systems, have been performed by Gonzalo Navaro & Ricardo Baeza-Yate. Their great achievement is the development of a system which, based on Huffman coding, compresses a text using on word list for indexing and compression [5]. The searching method is based on key increase in information volume. One of their new works is identifying people's names in texts [5,6,7]. We now explain what mentioned above in more details:

- INDEXING & SEARCHING

We use indexing to find user's query-based texts more rapidly in an Information Retrieval (IR) system. There are three methods:

- - FULL INVERTED INDEX:

A full inverted index is a word-oriented mechanism for indexing a text collection in order to speed up the searching task. The full inverted index is composed of two elements: the vocabulary and the occurrences. The vocabulary is the set of all different words in a text. For each such word a list of all text positions, where the word appears, is stored. The collection of all those lists is called the occurrences. In this method every word location is found and stored, so it will not be necessary to access the full text to define the answer set; however, using this method the system can identify phrases/patterns other than words while searching a text. That is why in this paper we use the some way for indexing Persian texts [1].

- - INVERTED FILE INDEX

In this method, names of the files containing any word is set as a linked list for any word so that phrase searching or approximate searching would be impossible.

- - BLOCK ADDRESSING INDEX

In this method, text is divided into equal blocks, then for every word the block number of the word occurrence is stored so that approximate searching is possible besides word-based search.

- COMPRESSION AND DECODING

The coding system used is an influential factor in compression. There are different coding methods of which many are not suitable for IR systems. The Ziv-Lempel (Dictionary-based) Method, for instance, is one of the text compression methods used in most of operating systems, for it is fast and needs a small storage space. But for use in a text retrieval environment this method yields two basic obstacles [1]:

a) Decoding must start at the beginning of the file.

b) It would be difficult to search compressed files without decoding.

But one of the privileges is that in this method no lexicographical storage (Character Table) is necessary. However, for text retrieval systems, in which lexicon is a key need for index and search, this cannot be considered as privilege.

In coordination with retrieval systems, we usually use two methods for coding:

a) Pattern substitution coding (constant).

b) Word-based of Huffman coding (variable).

The two methods are generally similar. But they have some advantages/disadvantages over each other. Pattern substitution coding performs faster in decoding compressed files. However, Huffman coding method needs less storage space in compression. They only alter in codes; Huffman coding method uses variable length codes, while pattern substitution method uses numeric assignment. This paper follows pattern substitution method for compression.

IMPLEMENTATION/DEVELOPMENT

Here comes the implementation of adopted algorithm.

- BASES OF THE ALGORITHM

As discussed before, the approach used to reach a faster information retrieval and search, is to organize an index for text database. For indexing we need a lexicon containing different words of each text. Also, for compressing a text file we need a lexicon to identify the relations between codes and actual text phrases. The important point here is what phrases to code. If the text words are selected, the compression lexicon would be the same as that of indexing so that a unique lexicon is used to perform two goals (compression and Indexing). Hence, the base of the discussed algorithm, is to use one lexicon uniquely in two case [8].

- COMPRESSION

we need a lexicon to compress the text. This lexicon contains words and their codes which identify the act of compression. Table 1, as an example, demonstrates a concise lexicon including some words and their corresponding codes.
Sample text: " رضا و علی با هم دوست هستند."

Table 1: Compression.

word	code
رضا	1
و	2
علی	3
با	4
هم	5
دوست	6
هستند	7

This Algorithm works as follows: At first the input text is broken down to words. These words are then compared individually with those in the lexicon. On case it is found in the lexicon, its particular compression code is selected and substituted. Otherwise, that word is appended at the end of lexicon and a next code is assigned to it. Figure 1 simply shows encoding the sample text:
"رضا و علی با هم دوست هستند. رضا و علی با هم به گردش رفتند."

word	رضا	و	علی	با	هم	دوست	هستند	.	به	گردش	رفتند
Code	1	2	3	4	5	6	7	8	9	10	11

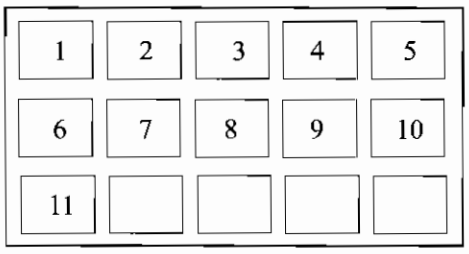


Figure 1: Compressed file as shown in the figure, for data compression, to each word a number is assigned. This number is saved when the corresponding word occurs instead of the word itself.

- DECODING

In most of compression methods, decoding algorithm is considered to be inverse of compression algorithms. This is also true with our algorithm, i.e., codes are read from the compressed file, the corresponding word is figured out in the lexicon, and finally retrieved

in a new file. We keep on repeating this procedure up to the end of that compressed file. One of the advantages of this method is that codes are incrementally stored, so that the speed of information retrieval is considerably high unlike the Huffman coding method.

- INDEXING ALGORITHM

As we are intended to retrieve those texts including a specific pattern, we need to locate and store all text words in our index. That's why in this algorithm, the lexicon must contain text words, a linked list that portraits location of word occurrence, and the number of those occurrences in the text. In this algorithm, the input text file is read, then words and their locations are registered. If the word happens to exist in the lexicon, the word's next location in the text is added to the linked list, and we increment the number of word occurrences in the text where it is stored. Otherwise, that word is appended to the lexicon and its location is added to the linked list. The above algorithm for our sample text yields what is shown in Figure 2:

Sample text: " رضا و علی با هم دوست هستند. رضا و علی با هم به گردش رفتند."

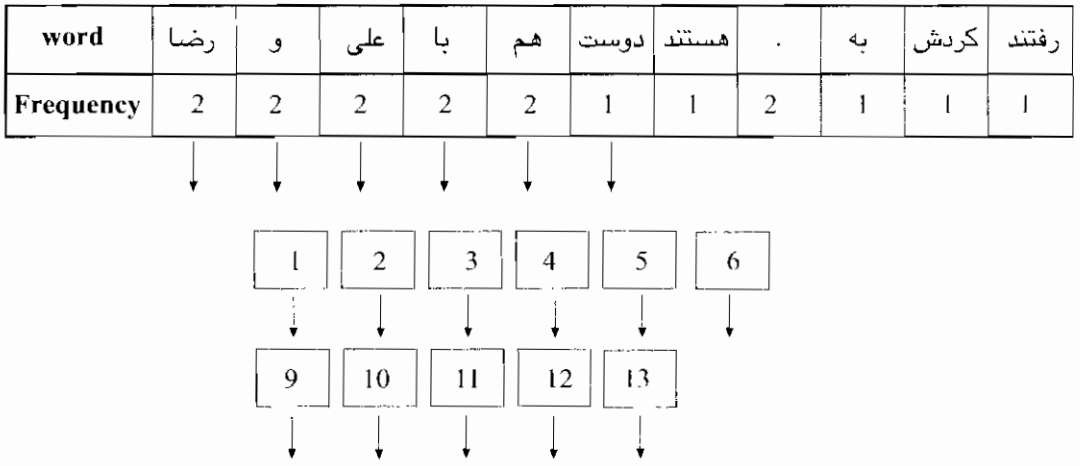


Figure 2: Indexing.

The indexing part consists of a lexicon which includes non-similar words so that the number of word occurrences and the next place of word occurrence for each word is specified. The pointers in the figure show the location of the words in reference to each other in the text.

As observed, both algorithms of compression and indexing use the same common lexicon. The use of similar lexicons for two separate goals saves storage space. We now continue with combination/integration and the privileges and efficiencies of the algorithm.

- COMBINATION OF INDEXING AND COMPRESSION ALGORITHM

We observed, in our indexing algorithm, that besides saving the words together with their occurrences each word's location is also stored. So, in order to combine these two algorithms (compression & indexing) it would be just enough if we store one code in respect to/with

any given word, and put the linked list inside the compressed file. To clarify this point, let us consider the example given below:

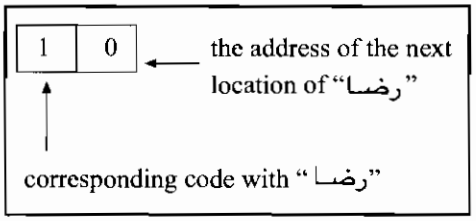
Example: Consider the text, "رضا و علی با هم دوست هستند. رضا و علی با هم به گردش رفتند."

Our system first identifies the existing words. Words are distinguished by spaces or other special characters. In the above example the word distinguished first, is "رضا". Since this word dose not exist in the lexicon, we add it. Then a code is assigned to it and its occurrence address is stored. Table 2 presents the process.

Table 2: Making lexicon

word	رضـا
code	1
location	1
frequency	1

Then the main file needs to be compressed. In respect to the main text we will have:



After " رضا" is distinguished and the required data is stored, the next word will be distinguished and so forth. During execution of the algorithm only the value of code remains constant while their values vary. When we come to "رضا" for the second time, in our example, the information to be stored is the way presented in Figure 3.

word	رضـا	و	علی	با	هم	دوست	هستند
code	1	2	3	4	5	6	7
location	1	2	3	4	5	6	7
frequency	2	1	1	1	1	1	1

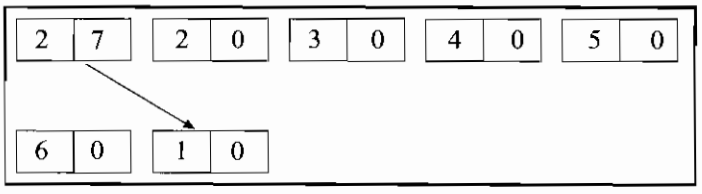


Figure 3: As shown in the above figure, code 1 is assigned to " رضا", the first word in the text. In the compressed file the numbers 1,7 are saved instead of the word " رضـا " so that 1 implies " رضـا " and 7 implies the next place of " رضـا " in the text.

When the information is saved, we should be able to achieve user's queries.

SEARCHING

By analyzing all words used in the text and their prepared data, the iteration frequency and location of words will be available. This system is capable of satisfying user's query in three ways:

a- Retrieving texts containing a specific pattern

b- Retrieving texts relevant to a number of specific words (queries)

c- Distinguishing homogeneous texts

- RETRIEVING TEXTS CONTAINING A SPECIFIC PATTERN

As every word's location is available, retrieval of texts that contain a specific pattern would be fast and convenient. To distinguish texts containing such phrases we don't need to decode the compressed text. However, searching is performed on compressed text and we only decode texts which are part of the answer set.

To do this, we just need to find the first location of the word occurrence in our word list, and then check the existence or non-existence of the rest of the phrase. In case we observe the complete phrase occurrence, the searched text is considered as desired and presented to the user.

- RETRIEVING TEXTS RELEVANT TO A NUMBER OF SPECIFIC WORDS

To retrieve texts relevant to a query, we adopt the extended Boolean model to discover texts and rank them. So, to weigh the text words the method of parameter TF-IDF (Term Frequency - Inverse Document Frequency) is employed [1].

The speed of decoding compressed files in retrieval operation is one of the deficiencies of text retrieval systems. Indexing method helps us a lot to overcome this. Since text word's location is available, it would be possible in a text retrieval including a specific pattern to retrieval only the adjacent context of that pattern. For instance, suppose a 100 page book containing the desired phrase. If the user only needs one page on which the phrase occurs, the retrieval (decoding) time is just 1/100 of the time needed for the whole book. This means that one of the advantages of this system, is the ability to free-access text pieces.

EXECUTION

To execute the above algorithm a software package under Visual Basic 6.0 has been designed. After an input text file with word pad format is read through the discussed algorithm, and executed, it compresses and indexes the text, in which word lists are processed using a common lexicon. Then, the user is capable of performing queries for

specific patterns on key words. In this package, the query for key words is done employing extended Boolean method. We can summarize the package features as follows:

- Compressed text storage.
- Using integrated word list (lexicon) for indexing and compression.
- Ability of searching text with constant patterns/phrases.
- Ability of searching text relevant to word-based query.
- Retrieval of vicinities to user's search phrases.
- Statistical information of texts such as total word numbers, different word numbers, number of sentences, number of paragraphs, iteration frequency of any desired word.
- Using extended Boolean method to search.
- Finding out homogenous texts.

The system evaluation is different for different Farsi texts. One application of this engine was successfully, 80% success, tested on Golestan Sa'di[2] with cooperation of professional Persion linguists. It mentioned to what corresponding Bob[3] of Golestan Sa'di the requested text belonged [2].

SUGGESTIONS

a) Increasing system power using audio/video data.

b) Using Huffman word-based coding method for compression.

c) Inter communicating of this system and other word-processing software.

ENDNOTS

1. Persian language
2. Golestan sa'di is written by sa'di Persion poet of the 14th century.
3. Each of the 8 drivisions of Golestan.

REFERENCES

[1] Baeza-Yates, R. and Ribeiro- Neto, B., *Modern Information Retrieval,* Addison Wesley Longman, Reading, Information Mass., 1999.

[2] Khalessizadeh, S. M. and others, "An Investigation in Farsi Texts and Literature as well as Case Study by Means of Investigator Software", *Journal of the Literature and Human Sciences*, 2005.

[3] Manber, U., Sun Wu; "GLIMPSE: A Tool to Search Through Entire File Systems", Proceedings of the Winter 1994 USENIX Conference, San Francisco, CA, USA, pp. 23- 32, 17-21, January 1994.

[4] Moffat, A. and Zobel, J., "Information Retrieval Systems for Large Document Collections", *Text Retrieval Conference (TREC- 3)*, Gaithersburg, MD, USA, pp. 85-93, 2- 4, November 1994.

[5] Navarro, G. and Baeza-Yates, R., "A Flexible Approximate Matching Tool for Searching Proper Names", *Journal of the American Society for Information Science and Technology*, vol. 54, pp. 3-15, 2003.

[6] Navarro, G., et al., "Adding Compression to Block Addressing Inverted Indexes," Information Retrieval, vol. 3, no. 1, pp. 49- 77, 2000.

[7] Navarro, G., Ziviani, N. and Edleno Silva, "Compression: A Key for Next-Generation Text Retrieval Systems", *IEEE Computer* 33(11):37-44, November 2000.

[8] Varadarajan, S. and Tzi- cker Chiueh, "SASE: Implementation of a Compressed Text Search Engine", Usenix Symposium on Internet Technologies and Systems, November 1997.