On Arabic Texts Compression and Searching

Hassen Sallay
Imam Mohammad Ibn Saud Islamic University
College of Computer and Information Sciences
P.O.Box 84880 Riyadh 11681
Saudi Arabia
Hassen_sallay@ccis.imamu.edu.sa

Journal of Digital Information Management

ABSTRACT: *With the dramatic increasing of electronic Arabic content, the text compression techniques will play a major role in several domains and applications such as search engines, data archiving, searching and retrieval from huge databases. Mainly the combination of compression and indexing techniques allows the interesting possibility to work directly on the compressed textual files or databases, which results saving time and resources. The existing compression techniques and tools are generic and do not consider the specific characteristics of the Arabic language such as its derivative nature. Mainly compression techniques should be based on the morphology characteristics of the Arabic language, its grammatical characteristics, the texts subject, and their statistical characteristics. The paper surveys the state of the art of the Arabic texts compression techniques and tools and identifies some research tracks that should be explored in future. It presents also some dedicated Arabic text compression algorithms which save more physical space and speed up the data retrieval text files by searching in their compressed form.*

Categories and Subject Descriptors
H.3.3 [Information Search and Retrieval]: Search process; **I.2.7 [Natural Language Processing]:** *Text analysis*

General Terms: Information Retrieval; Text searching: Arabic language processing; Text compression; Morphology; Textual databases

Keywords: Data compression, Arabic language, Searching in compressed files

Received: 16 June 2010; Revised 18 August 2010; Accepted 21 August 2010

1. Introduction

Arabic language is the native language of more than 200 million people and is a sacred language for the entire Muslim world. It is one of the richest languages of the world thanks to its derivative nature. Furthermore, it is also a stable and mature language that does not change much with time.

Nowadays, the information revolution has largely increased the computers role, especially after the huge explosion in the amount of information transmission resulting form the emergence of Internet. Thus, data storage and exchange must be easier, faster, and cheaper. The amount of Arabic data electronically available, either on the web or on other forms is dramatically increasing daily. A report published by Madar Research Journal in the year 2005, which includes statistics and forecasts on Internet users in 17 Arab countries, estimated the size of the internet community in the Arab world in excess of 25 millions. Moreover, a study from the Research Unit of Internet Arab World magazine states that there are 1.9 million online websites in Arabic and that number is expected to double every year. Thus, data storage and exchange must be easier, faster, and cheaper. Data compression technique is one of the most modern methods dealing with these problems.savings resources and speeding up data transfer process. In recent years, data compression received a great attention from researchers in the field of data miming in compressed databases. The recent research studies show that specialized compression techniques for natural languages have a good performance with an average compression rates up to 35%. Moreover, these techniques increase the compression rates by a factor of up to 8 when compared to search in the original version [1]. The combination of compression techniques and indexing techniques allows the interesting possibility to directly work on the compressed text databases and indexes, which results in saving time and resources [2]. Furthermore, combining new techniques of database indexing, searching and retrieval with compression techniques is now a leading research tendency since they speed up the searching process by a factor of 8 [1].

There are many computer programs accepting Arabic texts and providing compressions for them, but they do not take into account the specific characteristics of the Arabic language. The Arabic language has interesting characteristics which can be very useful in increasing this compression rate and improve the searching and archiving and data transfer techniques. For example, by using morphological analysis due to the fact that Arabic language is a derivational language we can code the basic form of words (stems) followed by their suffix and prefix fixed code since these prefixes and suffixes are invariant for the stems. For example, the word "المستقبلون (the receivers)" will be segmented to "قبل (accept)" which is the stem and has its own code, and "المست" which is the prefix and "ون" which is the suffix that can be coded by a fixed code representing the word "many". If we represent each character by a byte it will cost 10 bytes (المستقبلون contains 10 arabic characters) without using morphological analysis and 5 bytes since "المست" will be represented by 1 character, "قبل" by 3 characters and "ون" by 2 characters and therefore we will get a ratio of 50% if we use Morphological analysis.

Providing Arabic users with a compression tool dealing with characteristics and specificities of the Arabic language to achieve high quality compression, and to improve the performance of the searching, archiving and network transfer of Arabic texts is an ultimate goal and does not need more justification.

2. Related work

Data compression methods can be divided into two categories: Lossless Compression methods without losing any part of the data and Lossy Compression methods. Lossless Compression methods that allow the full retrieval of the entire

information are used for important data such as some types of medical images, .exe files, and textual files like .txt, .doc… etc. Lossy Compression methods are used to obtain very high compression ratio when there is no necessary need to keep the full details of the original data. These compression methods are well suited to multimedia files, including images, sounds, and video files [3,4,5].

On the other hand, texts compression methods are divided into two main types: methods based on statistics and methods based on dictionaries or glossaries. Statistical methods count characters frequency based on probabilistic and statistical models. This modeling provides a text nature description based on the probability of characters frequency and also based on what came before and will follow the letters. The dictionaries-based methods use dictionaries containing words consisting of several characters with their coding. The similar words found in the text are replaced by the corresponding code. Statistics-based methods provide high compression ratios, but with costly computing time, while dictionaries-based methods, in the opposite ways, are fast but with smaller compression ratios.

Concerning the Arabic texts compression, there are some tools supporting the Arabic language without taking into account its characteristics, such as Run Length Encoding, Huffman Coding, Arithmetic Coding, LZ-77 Encoding [6,7], LZH, LZW Coding [3,8]. The most famous programs currently used for lossless data compression are Winzip, WinRAR, and the JBEG program used for Fax. All these algorithms are available on the market for windows, but they are not open-source. The most famous free and open-source programs are Compress, Gzip, and 7zip which work on Unix. The Winzip program is the fastest and the most famous. However, the WinRAR program provides better compression rates. All these programs integrate the Huffman algorithm with LZW algorithm. All of them do accept Arabic texts, but they do not take into account the Arabic language characteristics, especially the most important one: that it is a derivative language.

The second category is a few researches that focused on the characteristics of the Arabic language. In fact, in [9] Hufman algorithm was used in some Arabic texts. Also in [10] conjugational characteristics were used to compress dictionaries' data while in [11,12] there was an attempt to construct a library of general programmers but still incomplete and did not cover most known technical compression as it remained poor in adapting Arabic characteristics.

On the other side, the idea of compression through the topic was not tackled in the known compression software. The idea lies on the techniques of document's classification. In fact, classification of documents based on their contents maybe very interesting in the compression due to the fact that each subject and topic has its own characteristics and specificities that can be employed to improve the compression. The idea behind that consists in classifying the document first and then applies an appropriate compression method depending on its topic.

Typically, textual data analysis provides two types of topic-based classifications: the clustering methods and the scoring methods. Clustering methods divide the texts into different clusters (also called categories, classes or groups) providing a topic-based structure. The textual database clustering can be seen as a library design and arrangement. Then, scoring methods are used to discover and identify the appropriate cluster (i.e. the right subject, class, group or category) of a new text [13,14,15].

For Arabic content, classification methods have been used in several areas. There are some investigations on the on-line handwritten digit recognition of Arabic scripts [16,17]. Some researchers propose also an auto-indexing method for Arabic text based on the extraction of the keywords, necessary for the classification process [18]. However, only few papers investigate the use of classification methods for Arabic texts and their effectiveness remains limited [19,20,21,22]. To our best knowledge, there is no research that applies operations research algorithms to improve the performance of classification methods on the Arabic content. Also, the compression based-on the content has not been invoked and employed neither for English nor for Arabic.

Finally, Concerning the compression and extraction of data from data bases containing Arabic data, we have not encounter any research in this area as well as research in compressed Arabic texts. This is maybe due to the fact that compression of data has been recently introduced in this domain.

3. Arabic texts Compression Algorithms

3.1 Arabic text pre-processing

In our best knowledge, compression based on text topic is not used in any compression tool. The user determine the type of the text to compress (Religion, Sport, Engineering, Sociology, etc.) and some other features (contain foreign character or not, there are symbols varying depending on the grammar rules or not etc.). Based on statistics study of a large number of texts of each subject, we construct lists containing the most used words for each subject or topic. We use also the most used patterns in Arabic language. Firstly, we preprocess the Arabic texts by replacing the most used Arabic words in this subject by a letter of non Arabic alphabet (for example English alphabet) and some of ASCII codes not used in Arabic texts. For example the following Arabic sentence

يا أيها المستكثرون من الطعام أفعلتم أن الطيبات التي استمتعتم بها قبل ساعة أنكم مسؤولون عنها عند الله = 108 bytes

Will generate the following preprocessed sentence since each bold word is either a most used word or an Arabic pattern

q p n c مسؤول m ساعة 5% l k j متع i h طيب g f طعام أفعلتم A c كثر b a r bytes 64 =

So we will get just by this preprocessing a compression rate of 41%=108/(108-64). After that we run some of common compression used tools on the preprocessed file to get a greater compression. The limited number of character in foreign alphabet is a limitation of this algorithm, as a solution we can use a combination of more than one character to represent a word. Figure 1 shows the GUI offered to the user to choose the type and the feature of his text to compress.

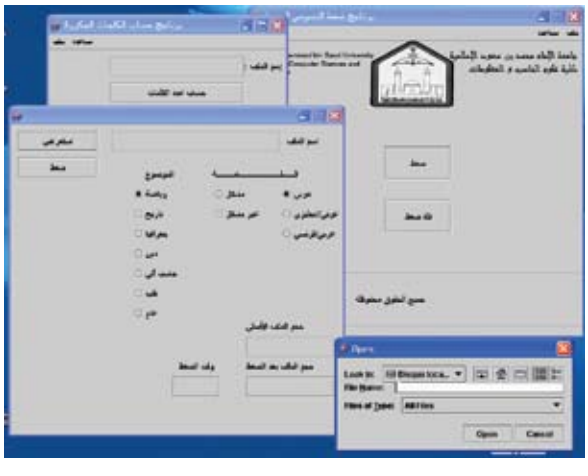

Figure 1. GUI to choose the text type and features

Based on deep statistics on religious books dataset, we identified the most used words in this subject. We included also a list of 200 items of Arabic suffixes and prefixes. We preprocessed the dataset files and the resulting preprocessed files are passed to winrar compression tool. We measured the compression gain and the time needed when we apply this algorithm. Fig 2 shows the compression rates we got for four scenarios (1) compression rate by using only preprocessing (2) compression rate using only winrar without preprocessing (3) compression rate using preprocessing and winrar which is our algorithm and (4) the impact of using preprocessing on winrar compression. We observe that the algorithm, for the dataset we used, reaches an average compression rate of 28% and a maximum rate of 52%. It increases also the winrar efficiency by 18% in average and 41% in the best case and it increases also the winrar compression rate by 5%. Obviously if include more and more words and Arabic suffixes and prefixes the compression gain will increase more and more.

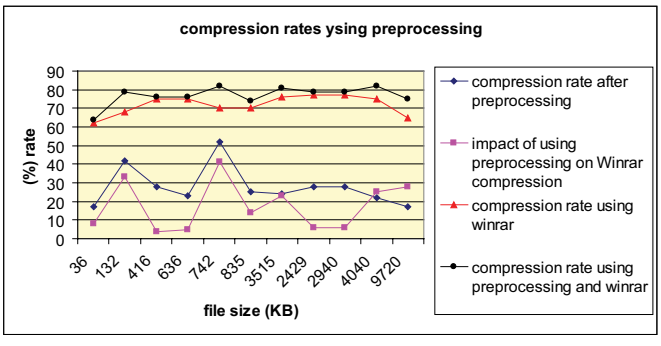

Figure 2. Compression rates by using preprocessing

Regarding the compression time, the preprocessing generates an overhead and increases clearly the compression time. For example we need only 13 seconds to compress a 10mb text file without preprocessing and 49 seconds by using preprocessing. Thus this algorithm is more dedicated to the archiving application where the compression rate is the big issue rather than the compression time. In all the cases getting greater compression rates will be very beneficial even in searching time if we search in the compressed file version. Fig3 shows the impact of preprocessing on the compression time. This impact can be reduced by implementation the algorithm with C language rather than JAVA language which we used in our current implementation.

3.2 Arabic text oriented compression

As we mentioned before, texts compression methods are divided into two main types: methods based on statistics and methods based on dictionaries or glossaries. Statistics-based methods provide high compression ratios, but with costly computing time, while dictionaries-based methods are fast but with smaller compression ratios. Huffman and LZW algorithms are respectively typical examples of these two classes. In this section we will describe some compression algorithms enhancing the Arabic text compression by combining the two aforementioned classes to give more interesting results.

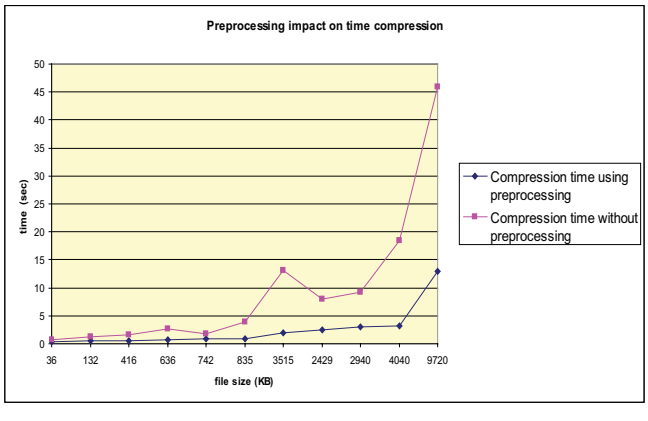

Figure 3. Preprocessing impact on the compression time

The first algorithm uses two fixed tables one is for the most used words in Arabic and the second for Arabic characters. The most 30 used Arabic words are calculated by Tim Backwalter web site (qamus.org) by gathering the dataset from Google. By applying Huffman compression which focuses on the character/word distribution redundancy and uses a variable length coding by giving the letters/words having high frequency smaller codes, we generate the word's coded table. The second table is the letters coded table generated upon statistics performed on several variant Arabic texts. Since these two tables are static and do not change from a text to another they are included directly in the code and do not affect the size of the compressed file since we are not obliged to include these tables code in the compressed file. The main advantage of using a merge algorithm between words and letters is the space saving we get every time a word in the coded table is found in the file. For example if the word "الذي" is found it will be given a code "11001" rather than "01111000100011101010101001" since we have (see [12])

11001	الذي

And for the character we have

011	ا
110	ل
001000111010	ذ
101001	ي

So we save 19 bits every time the word "الذي" occurs in the text. We notice that more frequent and longer the word found larger the saving will be.

The algorithm works as follow:

For the compression phase

While the there are words in the file do
Read a word
If it exists in the words table write 0 concatenated to the words code Else Split the word into characters and write 1 concatenated to character code for each character of the word.

There is no need to code the spaces character since we can add it in decompression phase when we identify a word.

The bit 1 or 0 is used in decompression phase to identify which table should we use to reconstruct the word.

For the decompression phase

Reconstruct the two Huffman trees of words and characters

Read a bit if it is 0 read bit by bit and search in the words tree to identify the word else if it is 1 then read bit by bit and search in the character tree to identify the character. Once a word (splitted or not) was identified write it concatenated with a space character in the decompression file and repeat the aforementioned operation until the file ends.

If we extend now the words table to contain more than 30 words so the compression rate will be grater but in contrast the

Huffman code will be larger. One enhancement is to divide the words table to several sub-tables. Each sub-table contains a specific number of words and an identifying number needed to identify the sub-table in both compression and decompression phases. It is obvious that smaller the table size is, smaller the word code will be. But there is a trade-off between the code size of the word and the number of table and switching between these sub-tables. In the following paragraph a formalization of the problem under an optimization problem form is given.

Dividing table which has (m) lines into many small table(n),each table(i), $1 \le i \le n$, contains k_i lines and one switch symbol. Thus

$$m = \sum_{i=1}^{n} k_i - m, \text{ with } k_i < m, 1 \le i \le n.$$

m is fixed number ($m \in N^*$)

n is the number of switches symbol which corresponds to the number of tables.

Switch symbol is one bit (0 or 1) one means the read out word in same table then not need to add number of table in encoding. Zero means the read out word in different table then must adding number of table in encoding.

$k_i = p$ then $m = n*p-n=n*(p-1)$.

Denoting by b_s the number of bits in the line(s) ($s \le m$) and by b_{ij}^N the number of bits in the line(j) in the table (i), $1 \le j \le k_i$.

Then we have:

First case: $b_{ij}^N < b_s$ for each ($s \le m$) for all $1 \le j \le k_i$ and $1 \le i \le n$.

Second case: $(\sum_{1 \le i \le n} \sum_{i \le j \le n} b_j^N - \sum_{i=1}^{n} b_i \#) < \sum_{s \le m} b_s$

So we have a minimization problem represented by the following equation $\text{Min}_{1 \le n \le m} (\sum_{1 \le i \le n} \sum_{i \le j \le n} b_j^N - \sum_{i=1}^{n} b_i \#)$

We encode the text by using these different tables by including a switch symbol to identify the specific table we should use in decompression process.

The second algorithm extends the QR algorithm by considering the topic of the text to compress and the Arabic languages characteristics. The QR Code is a matrix code (or two-dimensional bar code) created by Japanese corporation Denso-Wave in 1994. The "QR" is derived from "Quick Response", as the code to allow its contents to be decoded at high speed. QR Codes are common in Japan, where they are currently the most popular type of two dimensional codes.
So we define k two dimension tables (I, J) indexed by k. In our case k=8.Table1 contains the common used words per text subject or topic (literature, history …). Table2 and 3 contain Arabic language patterns. Table 4 contains Arabic letters, Arabic particles and Arabic prefix. Table 5 and 6 contain Arabic suffix. Table 7 contains English letters and finally table 8 contains the special characters.

In the compression phase we indicate a switching between these different tables as following: when we change the table we change k, I and J. If we are in the same table we check if row (I) is changed and column (J) is changed then we include the symbol 10 to indicate this switching. Else if only row (I) is changed then we preserve J and we include 10 symbol, else if column (J) is being changed we preserve I and include 00 symbol. So we have two possibilities if we are in the same row we change only the column and we preserve the row else we change the row and preserve the column.

For the compression phase

Read a word from the text to compress

Check if that word is in common used words or Arabic particles then encode it.

Else if it is in Arabic patterns then extract root and encode it.

Else split it into three part: prefix (if there is prefix in this word), suffix (if there is suffix in this word) and stem. Split the stem and for each letter get its code form table of letter.

For the decompression phase

Read bit if this bit is "0" then

Take two bits after if these bits are

"00" then this is a space character.

"01"(01 for Arabic letters) take Huffman code.

"10"(10 for Arabic prefix) take Huffman code.

"11"(11 for Arabic particles) take Huffman code.

Else if this character is "1" then

Take three characters for no. of table (K), take three characters for row (I) and take three characters for column(this process is for (common used Arabic words),(Arabic suffix),(Arabic patterns),(English letters) and(Special characters)).

4. Searching in Arabic Compressed files

As aforementioned before, data miming in compressed databases is promising more and more. Combining new techniques of database indexing, searching and retrieval with compression techniques is now a leading research tendency. They speed up the searching process by a factor of 8 [1] since the searching is performed in compressed file which has small size than the original file. Such saving in searching time is very beneficial for several applications such those developed for mobiles and PDA devices where the resource are limited (CPU, Memory and battery liveness). Moreover speed up the searching is also crucial in the archiving and the retrieval of data from huge database like search engine databases. Responding the user quickly is a primordial goal for web search engines. Combining indexing and compression is a good mean to reach this goal. We present in the following sections a performance study of two algorithms applied in Arabic texts more specifically Quranic Texts just for proof of the concept.

4.1 Indexed Huffmann Compression

This algorithm adds a new dimension to Huffman compression algorithm by including indexes of the verses of Al-Quran into the compressed file. Text of sacred Quran contains 320439 letters, 14960 non repeated words and consists of 114 chapters containing 6236 verses. In compression phase we add some bits to index the verses. Once the Huffman code table of each letters is generated, we compress each verse by writing the code of its characters in the compressed file. We add an index which is a counter of the used bits to compress this verse. We include the index code in the compressed file and we save it also in the index file used in the searching phase to extract the Quranic verse.

Now if we want to search a specific word, we compress it according to the same letters code table. We perform then the searching in the compressed file and when we find an occurrence we use the index file to identify the Quranic verse (see Fig. 4).

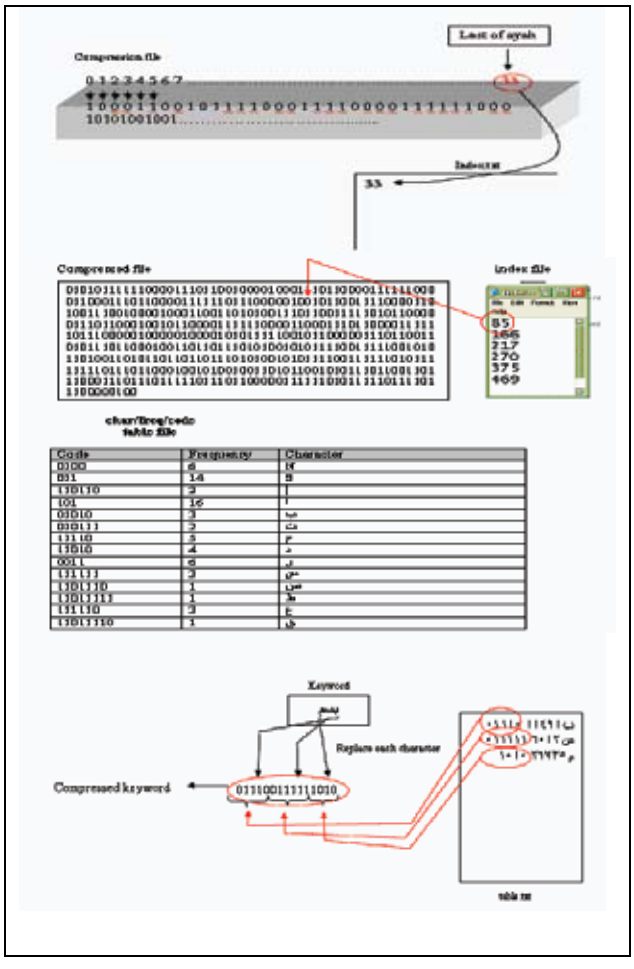

Figure 4. Indexed Huffman Compression

4.2 Indexed Quick Response Code Compression Algorithm

The Quick Response Code is a matrix code or two-dimensional bar code indexed by (I,J) allowing to code text contents at high speed. The QR Codes are currently the most popular type of two dimensional codes. In our case it is the perfect code since the Quranic texts contain a fixed number of non repeated words equal to 14960 words. So a matrix of 130*130 is enough to code all non repeated words. Each word will be indexed by two characters which is its code independently of the word's length. We use this matrix to compress the whole Quran. When we search a specific Quranic word we construct its code from this matrix and we perform a searching and extract the according verses based on the included indexing in the Quran compressed version (see Fig.5)

We implemented the two algorithms with a GUI for Quran searching application using JAVA for portability purposes, JSP for the Web version and Java Micro Edition (Mobile Java Applications) to run the application on NOKIA mobiles (see Fig.6)

We studied the performance of the four algorithms by evaluating the searching time in both original and compressed texts. The first is the linear search algorithm for the searching of the original text. The second is an algorithm which indexes all non repeated words by the number of the verse they belong to. This algorithm does not use compression and it is used to search the original text. The third algorithm is the indexed Huffman algorithm and the fourth is the indexed quick response algorithm. Fig7 shows the searching time comparison between these algorithms for the searching of a set of words chosen according to its frequency repetition (high, average and low) in the Quranic texts.

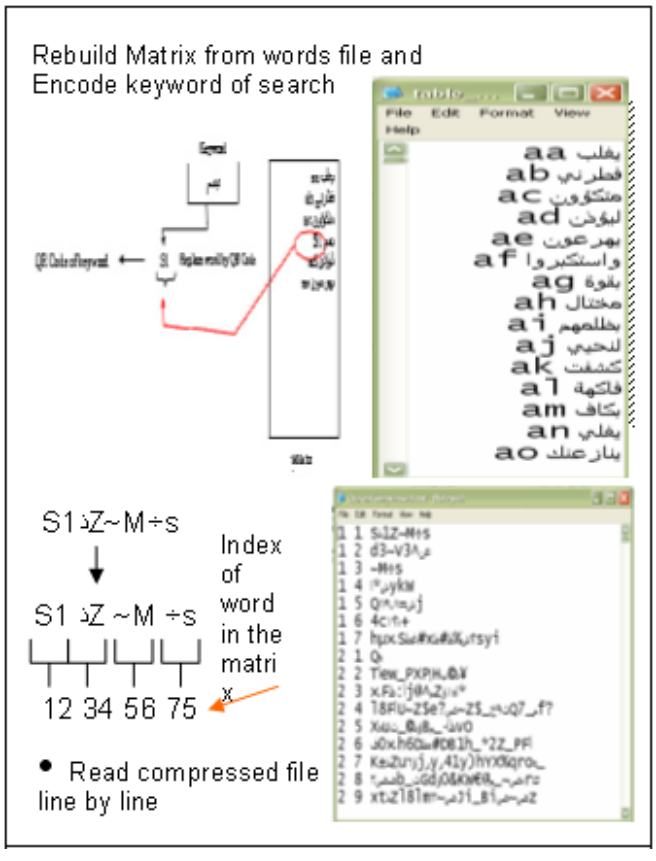

Figure 5. Indexed Quick Response Code Compression

We observe that in all the cases the searching time in the compressed file is less than the time in original file. This saving increases with the increasing of the word frequency. For example the needed time to search the very frequent word "الله" using linear search is largely greater than the searching time using indexed quick response algorithm. Fig7 shows also that the indexed Huffman algorithm is less efficient than the indexed quick response one but this difference decreases according to the word frequency on which Huffman algorithm bases its words code assignment.

Figure 6. GUI of Quran Searching Application

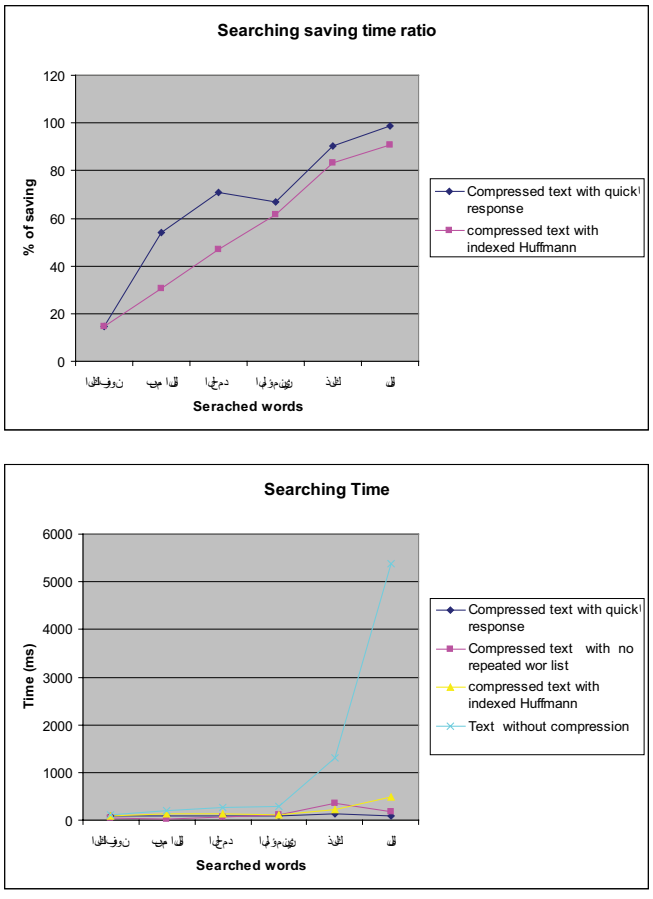

Figure 7. Searching time and the saving ratio

5. Conclusion and Future Work

This paper presents some adapted algorithms for the Arabic texts compression, based on the morphology characteristics of the Arabic language, its grammatical characteristics, and the texts subject, and their statistical characteristics. The first performance studies encourages us to continue to explore other data compression algorithms based on the compression fields and purposes, such as to speed up Arabic texts transfer, archiving databases, the rapid data retrieval from their compression form, as well as the quick search through the compressed version. As future work we envisage investigating statistics-based algorithms. We will focus on choosing a statistical model appropriate for this problem and collecting the data needed for its manipulation. A Hidden Markov Model with a specific architecture can be tested and an n-gram model applied to the Arabic characters will be explored for different values: unigram, bi-gram, tri-gram, etc. Characteristics of the Arabic will be largely treated there. Another direction is to investigate different approaches for building the dictionary according to three methods which are: static dictionary, semi-adaptative depending on the text, and fully-dynamic. After that, a coding scheme will be established. Explore an approach that combines advantages of the two previous ones is also an important research issue. Finally we believe that the compression based on the topic of the texts is a fertile research subject. In this context we will investigate the problem of classification of documents based on their contents. Different approaches of the topic-based classification will be explored, but the main focus will be on the use of operations research algorithms. Many Arabic texts in an electronic format have to be collected and used with convenient methods to extract lists of keywords needed for the classification methods. Once the classification approach is fixed, an appropriate compression strategy could be selected for the document based on its topic. All these algorithms will be incorporated in an integrated computing environment providing all the services mentioned above.

References

[1] Moffat, Alistair., Turpin, Andrew (2002). Compression and Coding Algorithms. Springer.

[2] Silva de Moura, Edleno., Navarro, Gonzalo., Ziviani, Nivio., Baeza-Yates, Ricardo (2000). Fast and flexible word searching on compressed text, ACM Transactions on Information Systems, 18 (2) 113-139.

[3] Cormack, G. V., Horspool, R. N. (1984). Algorithms for Adaptive Huffman Codes, Inform. Process. Lett. 18 (3) 159-165.

[4] Fraenkel, A. S., Mor, M., Perl, Y. (1983). Is Text Compression by Prefixes and Suffixes Practical? Acta Inf. 20, 4 (Dec.), 371-375. Wilkins, L. C., and Wintz, P. A. 1971.

[5] de Moura, E., Navarro, G., Ziviani, N. (1997). Indexing compressed text. In: R. Baeza-Yates, ed., Proc. of the Fourth South American Workshop on String Processing, Carleton University Press International Informatics Series, v.8, p. 95-111.

[6]. Shannon, C.E (1981). A mathematical theory of Communication, Bell System Technical Journal, 27. 379-423.

[7] J. Rissanen and G.G. Langdon. Universal modeling and coding, IEEE Transactions on Information Theory, IT-27:12-23.

[8] Ziv, J., Lempel, A. (1977). A Universal Algorithm for Sequential Data Compression, IEEE Trans. Inform. Theory 23 (3) 337-343.

[9] Mohamed, Y., Al-Habib, Osman and Mohammed (1991). Arabic Text Compression Using Huffman Code, The Arabian Journal for Science and Engineering, 16 (4B).

[10] El-Affendi, M A (1992). Arabic dictionary compression using an invertible integer transformation and a bitmap representation. J KING SAUD UNIV ENG SCI. 4 (1) 105-125.

[11] Kacem, Afef (1998). Intelligent and lossless data compression for Arabic documents, IASTD.

[12] Sallay, H., Alhadj, Yahya (2007). An Arabic Compression Tool, In: International Conference of Information technologies and Arabic language, Riyadh.

[13] Busygin, Stanislav., Prokopyev, Oleg., Pardalos, Panos M. (2008). Biclustering in data mining, Computers & Operations Research, 35 (9) 2964-2987.

[14] Kao, Yi-Tung., Zahara, Erwie., Kao, I-Wei (2008). A hybridized approach to data clustering, Expert Systems with Applications, 34 (3) 1754-1762.

[15] Güngör, Zülal., Ünler, Alper (2008). K-Harmonic means data clustering with tabu-search method, Applied Mathematical Modelling, 32 (6) 1115-1125.

[16] Khorsheed, M.S. (2007). Offline recognition of omnifont Arabic text using the HMM ToolKit (HTK), Pattern Recognition Letters, 28 (12) 1563-1571.

[17] Kherallah, Monji., Haddad, Lobna., Alimi, Adel M., Mitiche, Amar (2008). On-line handwritten digit recognition based on trajectory and velocity modeling, Pattern Recognition Letters, 29 (5) 580-594.

[18] Mansour, Nashat., Haraty, Ramzi A., Daher, Walid.,

Houri, Manal (2008). An auto-indexing method for Arabic text, Information Processing & Management, In Press, Corrected Proof.

[19] Duwairi R., An eager k-nearest-neighbor classifier for arabic text categorization, In: Proceedings of the International Conference on Data Mining (DMIN), Nevada, USA, p.187–192, 2005.

[20] El-Halees A. (2007). Arabic text classification using maximum entropy, The Islamic University Journal (Series of Natural Studies and Engineering) 15 (1) 157–167.

[21] Khreisat L (2006). Arabic text classification using N-Gram frequency statistics. A comparative study, In: Proceedings of the international conference on data mining (DMIN), Nevada, USA, p. 78–82.

[22] Hmeidi, Ismail., Hawashin, Bilal.,El-Qawasmeh, Eyas (2008). Performance of KNN and SVM classifiers on full word Arabic articles. Advanced Engineering Informatics, 22 (1) 106-111.

[23] Jensen, (1996). An Introduction to Bayesian Networks, UCL Press Limited, London.

Author Biography

Hassen Sallay received the B.E degree from national college of computer sciences of Tunisia in 1999. He received the M.E. degree and Dr. Eng. degree from Henri Poincarre Univ.- France in 2000 and 2004, respectively. After working as a teaching assistant (from 2000) in Ecole de Mines in France, he has joined as assistant professor (from 2004) in the Dept. of Computer and Information Sciences, Imam Mohammad ibn Saud Univ. His research interest includes Network and information Security Management, Network Management and Multicast services. He is also interested to the Arabization research fields. He is the leader of AMAN research group and a principal investigator in two research projects funded by KACST in Network Security field. He is also a steering committee member of Arabization IT research program.