Journal of Computer Science 6 (1): 24-28, 2010
ISSN 1549-3636

Arabic Short Text Compression

Iman Omer and Khalaf Khatatneh
Department of Information Technology,
Prince Abdullah Bin Ghazi Faculty of Science and Information Technology,
Al-Balqa Applied University, P.O. Box 19117, Al-Salt 19117, Jordan

Abstract: Problem statement: Text compression permits representing a document by using less space. This is useful not only to save disk space, but more importantly, to save disk transfer and network transmission time. With the continues increase in the number of Arabic short text messages sent by mobile phones, the use of a suitable compression scheme would allow users to use more characters than the default value specified by the provider. The development of an efficient compression scheme to compress short Arabic texts is not a straight forward task. **Approach:** This study combined the benefits of pre-processing, entropy reduction through splitting files and hybrid dynamic coding: A new technique proposed in this study that uses the fact that Arabic texts have single case letters. Experimental tests had been performed on short Arabic texts and a comparison with the well known plain Huffman compression was made to measure the performance of the proposed schema for Arabic short text. **Results:** The proposed schema can achieve a compression ratio around 4.6 bits byte^{-1} for very short Arabic text sequences of 15 bytes and around 4 bits byte^{-1} for 50 bytes text sequences, using only 8 Kbytes overhead of memory. **Conclusion:** Furthermore, a reasonable compression ratio can be achieved using less than 0.4 KB of memory overhead. We recommended the use of proposed schema to compress small Arabic text with recourses limited.

Key words: Short text compression, Huffman coding, Arabic language, dynamic hybrid coding

INTRODUCTION

Text compression permits representing a document by using less space. This is useful not only to save disk space, but more importantly, to save disk transfer and network transmission time (Brisaboa *et al*., 2007). Compression of short messages is a vital operation for low complexity entities, such as mobile phones, for achieving bandwidth savings and reducing costs, or even enabling the wireless medium to be shared by more devices (Rein *et al*., 2006b). With the continues increase in the number of Arabic short text messages sent by mobile phones, the use of a suitable compression scheme would allow users to use more characters than the default value specified by the provider.

There are two general approaches for text compression: Statistical coding and dictionary coding. Statistical coding schemes make use of the fact that different symbols usually occur at different frequencies and assign shorter codes to symbols that occur more frequently and longer codes to less frequently used symbols. They employ a statistical context model to compute the appropriate probabilities (Rein *et al*., 2006a). Then, the probabilities are coded with a proper encoding such as Huffman (1952), Arithmetic coding (Witten *et al*., 1987) or any entropy coder. Dictionary coding schemes make use of the fact that certain groups of consecutive characters (i.e., phrases) occur more than once and assign a code to a certain phrase. Most of the dictionary coders are based on LZ (Ziv and Lempel, 1977) and LZ78 (Ziv and Lempel, 1978) and are widely employed to compress computer data. The statistical coders give better compression performance than the dictionary coders. However, generally, they require large amounts memory (Rein *et al*., 2006a).

One way to enhance the performance of text compression schemes is to apply a lossless, reversible transformation to a source file prior to applying an existing compression algorithm. Therefore, it is easier to compress the source file. Decompression works in the reverse order (Fauzia and Mukherjee, 2001). BWT

Corresponding Author: Iman Omer, Department of Information Technology,
Prince Abdullah Bin Ghazi Faculty of Science and Information Technology, Al-Balqa Applied University,
P.O. Box 19117, Al-Salt 19117, Jordan Tel: 00962786374910 Fax: 962-5-3530462

is a well known algorithm that transforms the input text into another (reversible form) and apply some compression method on it. The compression is achieved if the transformation has a property that it enlarges the compressibility property of the text. In the decoding process which is broken into two steps. Firstly, the transformed text should be decoded and then the inverse transformation should be applied.

Generally, most text compression algorithms are designed for general natural language texts or designed for English text, as a result, they do not take advantage of special features of Arabic text. The problem with short text is that the redundancies within these texts are limited which makes regular compression messages inefficient. Few algorithms are designed for compressing short data messages used in devices with limited resources such as mobile phones and sensor network devices. However, they have restrictions on minimum length of the message to be compressed and they need a non-ignorable memory amount. Furthermore, no compression algorithm specifically designed for short text messages written in Arabic.

This study is an attempt to develop a compression schema for Arabic short text messages that is suitable for use in mobile phones.

MATERIALS AND METHODS

There exists a large amount of literature on lossless data compression and text compression. Most of the work is designed for compressing large size text and does not consider devices with limited resources. The researchers of (Rein *et al*., 2006a) indicated that many of the well known compression require memory from 0.5 to more than 100 Mbyte. On the other hand, most of the sensor networks' data compression techniques use statistical correlations between the typically larger data of multiple sensors as they all observe the same phenomenon which make them unsuitable for text compression.

One of the oldest and best known compression techniques is Huffman (1952) coding that maintains a forest of binary trees representing disjoint subsets of the alphabet with weight equal to the sum of probabilities of the elements in the subset. Various methods have been developed to improve Huffman coding such as, Dynamic Huffman coding (Vitter, 1987), Length-limited Huffman coding (Karp, 1961) and Huffman coding with unequal letter costs (Golin *et al*., 2002). Furthermore (Ghwanmeh *et al.*, 2006) used dynamic Huffman coding to compress Arabic text.

Arithmetic coding (Witten *et al*., 1987) is a form of variable-length entropy encoding that converts a string into another representation that represents frequently used characters using fewer bits and infrequently used characters using more bits, with the goal of using fewer bits in total. The basic idea of arithmetic coding scheme is to represent a text by an interval of real number between 0 and 1.

Most practical dictionary coding algorithms belong to a family of algorithms, known as Ziv-Lempel coding (abbreviated as LZ coding), derived from Ziv and Lempel's work. These algorithms are based on the idea of replacing the strings in the text with a pointer to where they have occurred earlier in the text. This family of algorithms are generally derived from one of the two approaches namely LZ77 and LZ78, respectively (Rein *et al*., 2006a).

Prediction by Partial Matching (PPM) is one of the most promising lossless data compression algorithms using Markov source model of order D. It uses a set of previous symbols in the source symbol stream to predict the next symbol in the stream. In the reality, the majority of symbols are encoded in inner nodes and the Marko v model becomes rather conventional. In spite of the fact that PPM algorithm achieves good results in comparison with others, it is used rarely in practical applications due to its high computational complexity (Shkarin, 2002).

The researchers of (Rein *et al*., 2006b) described a low-complexity scheme for lossless compression of short English text messages, a method which uses arithmetic coding and a specific statistical context model for prediction of single symbols. They used a simple yet effective approach for storing highly complex statistics in a succinct yet effective data model that can easily be trained by text data. They achieved a compression performance around 3.5 bits per byte for short text sequences larger than 75 bytes and required 128 kBytes of memory.

The researchers of (Rein *et al*., 2006a) used similar approach to the one they previously used in (Rein *et al*., 2006b) with a modified hash function in order to be able to compress short English text larger than 50 bytes. They achieved compression ranging from 4-3.2 bits byte^{-1} with memory requirements of 32-256 kBytes.

The researchers of (Lansky and Zemlicka, 2006) adapted well-known algorithms of adaptive Huffman coding and LZW to use syllables and words instead of characters for text compression of small or middle-sized English text files. However, their short text file should be larger than 3 kBytes.

That the entropy of Arabic text files can be highly reduced if the text is source mapped according to characters frequencies and then the resulting file is spitted into several sub-files each contains one or more bits from the code word of the mapped file (Abdel-

Rahman *et al*., 2006). This is a great method for large Arabic text but is does not work for short text.

In order to enhance the compression ratio, the researchers of (Abel and Teahan, 2005) proposed that preprocessing such as capital conversion, end of line encoding and token (diagrams, trigrams, word and phrases) replacement can reduce the size of compressed text. They proposed that the most frequently appearing tokens in a specific text are replaced by shorter code-words, then those token and their codes are inserted at the beginning of the compressed file to simplify decompression. However, because token frequency is limited within a single short text, inserting the frequently appearing tokens and their codes at the beginning of the compressed file will not be beneficial.

Because of the limited redundancy within short Arabic text, using traditional dictionary based or statistical compression do not work. The size of the dictionary is too large and the encoding tree or table for statistical encoding may not be compensated. In this study, we propose the use of static Huffman tree in a way similar to the method used for fax machines and Morris codes where frequently appearing characters in the Arabic language are given short codes and less frequently appearing characters are given longer codes. The code will, generally, reduce the size of the text, even that the code may not be optimal for a specific text.

In order to enhance the compression ratio, we use a technique similar to token replacement, described in (Abel and Teahan, 2005). The authors of (Abel and Teahan, 2005) showed that compression of large text files can be improved using dictionaries built "on-the-fly" and storing them within the compressed file. However, embedding a dictionary within a compressed file, can be considered as extra overhead that may not compensated, and it may reduce the overall compression ratio.

Alternatively, we discover frequently appearing diagrams and trigrams in the language and assign them code-words in the alphabet. Using too many diagrams and trigrams may be counterproductive because it reduce redundancy when used with statistical coding. On the other hand, using too few diagrams may have little effect on the overall performance of the compression schema As a result; we only used the 10 most frequent diagrams and trigrams.

The authors of (Abdel-Rahman *et al*., 2006) proved that the entropy of the least significant bits of Arabic text files is very high and the entropy of the most significant bits of Arabic text files is very low, such that, the entropy of the most significant bit is 0, 0.0097 for the second and 0.3867 for the third most significant bits. On the other hand the entropy of the least significant bit is 0.9855, 0.9787 for the second, 0.9213 for the third, 0.8464 for the fourth and 0.7127 for the fifth least significant bits.

Based on the previous two observations and on the fact that Arabic character can be encoded with 7 bits only (less than 128 characters), we propose the use of source mapping in a way similar to the one used in (Abdel-Rahman *et al*., 2006) were the most frequently appearing character is assigned the numerical 0 and the next frequently appearing character is assigned the numerical 1 code and so on. Then the resulting source mapped text is to be split into two sub-files of unequal length, one that contains the most significant bit (i.e. The 7th bit) of each character and the other file contains the remaining 6 least significant bits of each character. We deviated from the equal length sub files proposed in (Abdel-Rahman *et al*., 2006) because of the high entropy of least significant bits, which makes their compression very difficult especially for short Arabic texts.

It is well known that the Arabic language do not has things like capital letters and small letters as English language. Furthermore, lots of the non basic characters as fatha, are used rarely, especially with informal Arabic texts used in forums, emails and short messages in mobiles and even in many memorandums and books, simply because their meaning can easily be figured out from the context .

Furthermore, we proposes the use of what we call, Dynamic Hybrid Encoding (DHE). The technique dynamically determines the length of each character, however the length of each character can only be 6 or 7 bits. The technique works as follows, the frequencies of characters are calculated and numerical codes between 0 and 127 are assigned to each character according to its frequency as indicated by (Abdel-Rahman *et al*., 2006). Then, the most frequent diagrams and trigrams are add to the alphabet and assigned unique numerical values. Each token is scanned before being encoded, if it contain a character whose numerical value greater than or equal to 64, then each characters in that token has to be encoded using 7 bits code. On the other hand, if all characters in the token have numerical values less than or equal 63, then each character is encoded using 6 bits code.

In other way, (Abdel-Rahman *et al*., 2006) proved that all Arabic characters can be encoded with 7 bits only, and most Arabic characters can be encoded using only 6 bits, so we propose the use of 6 bits only to encode each character, and 7 bits when necessary. However, in order to inform the decompression algorithm witch to use 6 bit or 7 bits, we use a code for the space when

followed by a token that need to be encoded using 7 bit for each character, and a different code for the space when followed by a token that can to be encoded using 6 bit for each character.

In order to avoid the overhead of embedding Huffman trees within compressed files, one can use dynamic Huffman. However, with short texts, dynamic Huffman do not have good chances for compression, because of low redundancy of characters. Alternatively, Static Huffman and be used in a way similar to Huffman coding used for fax machines and the Morris codes where frequently appearing characters in the language-rather than in individual files, are given short codes and less frequently appearing characters are given longer codes. The code will, generally, reduce the size of source text, even that the code may not be optimal for specific texts. This approach is used in this study, and a static Huffman tree is used .

In order to build a generic Huffman tree, a large amount of short Arabic texts is gathered from different sources such as newspapers, forums web sites and web portals. The frequencies of letters and tokens are calculated and numerical codes between 0 and 127 are assigned to each character as described previously. Then the gathered short Arabic texts are source mapped according to the assigned numerical codes, and encoded using the dynamic hybrid coding technique described previously. The least significant 6 bits of each character are stored in a single sub-file and the remaining bit- if exists- is stored in a single sub-file. After that, for each sub-file, the well known Huffman algorithm is used to construct a generic encoding tree for the sub file the contains 6 bits of each character. The resulting generic tree can be used later to compress and decompress short Arabic texts. After that, for each sub file, the well known Huffman algorithm is used to construct a generic encoding tree that assign a single code for each 12 bits or 6 bits. The resulting generic tree can be used later to compress and decompress short Arabic texts.

RESULTS

In order to study the performance of the proposed schema, the schema applied on short text files gathered from different sources, then the proposed schema and the traditional Huffman are applied to texts of different lengths for 6 and 12 bit ectentions. All experiments were done on Intel Celeron 1 MHz processor, 1 G RAM, running Windows XP. Java language is used to implement booth algorithms. Arabic texts used to setup the Huffman tree were gathered from different sources such as e-newspapers, forums web sites and web portals. Arabic short texts used to measure the performance of the proposed schema also gathered is the same way.

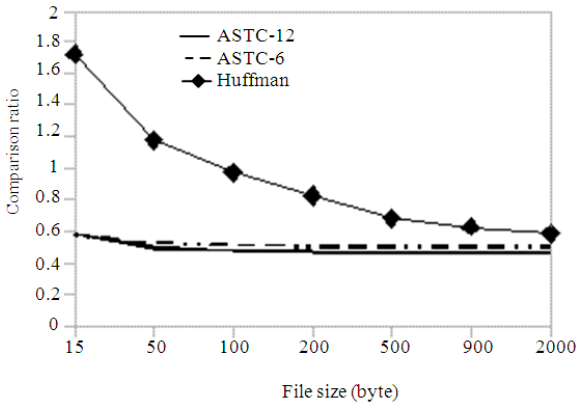

Fig. 1: Compression ratio of ASTC-12 and ASTC-6 compared with Huffman

In order to avoid extreme conditions, the average of compressing hundred different short texts of the same size is considered. Those hundred texts are constructed as follows; all gathered Arabic short texts were placed in a single file, then a continuous sequence of the required length is extracted from a random location and place in a different file. The process of selecting text and placing it in a file is repeated hundred times. A simple Java program is used to perform this task .

Because no previous work specifically investigated the compression of short Arabic texts, this we compared the compression ration achieved by the proposed schema, with the well known two pass Huffman compression. Figure 1 shows the compression ratio of the proposed schema, the 6 bits extension (ASTC-6) and the 12 bits extensions ASTC-12, compared with the Huffman compression for different text lengths.

DISCUSSION

As the length of text increases, the redundancy of characters within the text increases, and the chance to reduce the size of the text increases. Furthermore, the possibility that popular diagrams and trigrams appear in the text also increase, as the length of the text increases. However, it is also true that as the length of text increase, the possibility that rarely used characters appear in the text also increase, which negatively affect the compression ratio.

We can see that at very small text length, the proposed schema outperforms the two pass Huffman compression because the two pass Huffman compression embed the encoding trees into the compressed files. With very small files, such overhead may not paid back by the size reduction achieved by the compression. On the other hand, larger text files, this

overhead can be compensated because of the higher characters redundancy .

From Fig. 1, it is also clear that the compression ratio of the proposed schema when using 12 bits extension, is only slightly better than that when using only 6 bits extinction. However, the size of the generic Huffman tree is about 8 KB when using 12 bit extension compared to less than 0.4 KB when using 6 bits only.

We used a combination of text preprocessing, entropy reduction, our technique and Huffman coding to achieve a compression performance around 4.6 bits per byte for very short text sequences of 15 Bytes and a round 4 bits per byte for 50 Bytes text sequences, using only 8 kBytes overhead of memory. Furthermore, a reasonable compression ratio can be achieved using less than 0.4 KB of memory overhead. Our further investigations will concern the use of a more efficient offline context models, and the use of effective dynamic coding schema.

REFERENCES

Abdel-Rahman, M.J., M.I. Irshid and T.T. Nassar, 2006. Entropy reduction of Arabic text files. Asian J. Inform. Technol., 5: 578-583. http://medwelljournals.com/fulltext/ajit/2006/578-583.pdf

Abel, J. and W. Teahan, 2005. Universal text preprocessing for data compression. IEEE Trans. Comput., 54: 497-507. DOI: 10.1109/TC.2005.85

Brisaboa, N.R., A. Fariña, G. Navarro and J.R. Paramá, 2007. Lightweight natural language text compression. Inform. Retriev., 10: 1-33. http://portal.acm.org/citation.cfm?id=1210461

Fauzia, S.A. and A. Mukherjee, 2001. LIPT: A lossless text transform to improve compression. Proceedings of the International Conference on Information Technology: Coding and Computing, Apr. 2-4, IEEE Xplore Press, Las Vegas, NV., USA., pp: 452-460. DOI: 10.1109/ITCC.2001.918838

Ghwanmeh, S., R. Al-Shalabi and G. Kanaan, 2006. Efficient data compression scheme using dynamic Huffman code applied on Arabic language. J. Comput. Sci., 2: 885-888. http://www.scipub.org/fulltext/jcs/jcs212885-888.pdf

Golin, M.J., C. Kenyon and N.E. Young, 2002. Huffman coding with unequal letter costs. Proceedings of the 34th Annual ACM Symposium on Theory of Computing, May 19-21, ACM Press, Montréal, Québec, Canada, pp: 785-791. http://portal.acm.org/citation.cfm?id=509907.510020

Huffman, D.A., 1952. A method for the construction of minimum-redundancy codes. Proc. Inst. Radio Eng., 40: 1098-1101. DOI: 10.1109/JRPROC.1952.273898

Karp, R., 1961. Minimum-redundancy coding for the discrete noiseless channel. IRE Trans. Inform. Theor., 7: 27-38. DOI: 10.1109/TIT.1961.1057615

Lansky, J. and M. Zemlicka, 2006. Compression of small text files using syllables. Proceedings of the Data Compression Conference, Mar. 28-30, IEEE Xplore Press, Snowbird, UT., USA., pp: 458. DOI: 10.1109/DCC.2006.16

Rein, S., C. Gühmann and F. Fitzek, 2006a. Compression of short text on embedded systems. J. Comput., 1: 1-10. http://www.academypublisher.com/jcp/vol01/no06/jcp01060110.pdf

Rein, S., C. Gühmann and F.H.P. Fitzek, 2006b. Low-complexity compression of short messages. Proceeding of the Data Compression Conference, Mar. 28-30, IEEE Xplore Press, Snowbird, UT., pp: 123-132. DOI: 10.1109/DCC.2006.45

Shkarin, D., 2002. PPM: One Step to practicality. Proceeding of the Data Compression Conference, (DCC'02), IEEE Xplore Press, USA., pp: 202-211. DOI: 10.1109/DCC.2002.999958

Vitter, J.S., 1987. Design and analysis of dynamic Huffman codes. J. ACM, 34: 825-845. http://portal.acm.org/citation.cfm?id=31846.42227

Witten, I.H., R.M. Neal and J.G. Cleary, 1987. Arithmetic coding for data compression. Commun. ACM, 30: 520-540. http://portal.acm.org/citation.cfm?id=214762.214771

Ziv, J. and A. Lempel, 1977. A universal algorithm for sequential data compression. IEEE Trans. Inform. Theor., 23: 337-343. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?tp=&arnumber=1055714&isnumber=22696

Ziv, J. and A. Lempel, 1978. Compression of Individual sequences via variable-rate coding. IEEE Trans. Inform. Theor., 24: 530-536. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1055934