Journal of Computer Science 2012, 8 (12), 1940-1945
ISSN 1549-3636
© 2012 Science Publications
doi:10.3844/jcssp.2012.1940.1945 Published Online 8 (12) 2012 (http://www.thescipub.com/jcs.toc)

A Novel Multidictionary Based Text Compression

M. Baritha Begum and Y. Venkataramani

Department of Electronics and Communication,
Saranathan College of Engineering, Trichy-620012, Tamilnadu, India

Received 2012-02-24, Revised 2012-09-27; Accepted 2012-11-07

ABSTRACT

The amount of digital contents grows at a faster speed as a result does the demand for communicate them. On the other hand, the amount of storage and bandwidth increases at a slower rate. Thus powerful and efficient compression methods are required. The repetition of words and phrases cause the reordered text much more compressible than the original text. On the whole system is fast and achieves close to the best result on the test files. In this study a novel fast dictionary based text compression technique MBRH (Multidictionary with burrows wheeler transforms, Run length coding and Huffman coding) is proposed for the purpose of obtaining improved performance on various document sizes. MBRH algorithm comprises of two stages, the first stage is concerned with the conversion of input text into dictionary based compression .The second stage deals mainly with reduction of the redundancy in multidictionary based compression by using BWT, RLE and Huffman coding. Bib test files of input size of 111, 261 bytes achieves compression ratio of 0.192, bit rate of 1.538 and high speed using MBRH algorithm. The algorithm has attained a good compression ratio, reduction of bit rate and the increase in execution speed.

Keywords: Dictionary Based Encoding (DBE), Burrows-Wheeler Transform (BWT), Run Length Encoding (RLE)

1. INTRODUCTION

Data compression is the method representing information in a compact form. It decreases the number of bits required to represent a data. Similarly Data decompression restores compressed data back into an original form. A Bit is the most fundamental unit of information in computing and communications and it possess the value zero or one. The partial redundancy in uncompressed data paves way for compression; that is, the same information can be stored using fewer bits.

Generally compression algorithms require large execution time, memory size because of the presence of large number of alphabets in original source code (Carus and Mesut, 2010).Text compression coding can be categorized into two groups; statistical based coding and dictionary based coding.

Dictionary-based methods are popular in the data compression domain (Begum and Venkataramani, 2012; Mohan and Govindan, 2005; Sun *et al*., 2003). On contrary statistical methods use a statistical model of the data and encode the symbols using variable-size code words in accordance with their frequencies of occurrence, dictionary-based methods opt for strings of the symbols to set up a dictionary and then encode them into equal-size tokens using the dictionary (Li *et al*., 2003; Carus and Mesut, 2010; Bhadade and Trivedi, 2011). The dictionary is formed by the strings and it may be either static or dynamic (Mohan and Govindan, 2005). The static is permanent, occasionally allowing for the addition of strings but no deletions, whereas the latter holds strings formerly found in the input stream, allowing for additions and deletions of strings as a new input string is processed.

Huffman coding, Arithmetic coding and PPM are examples of statistics based coding. In this coding scheme the symbols are coded to variable lengths. The most well known dictionary based coding is LZ algorithm (Abel and Teahan, 2005).

Corresponding Author: M. Baritha Begum, Department of Electronics and Communication, Saranathan College of Engineering, Trichy-620012, Tamilnadu, India Tel: 919443677672

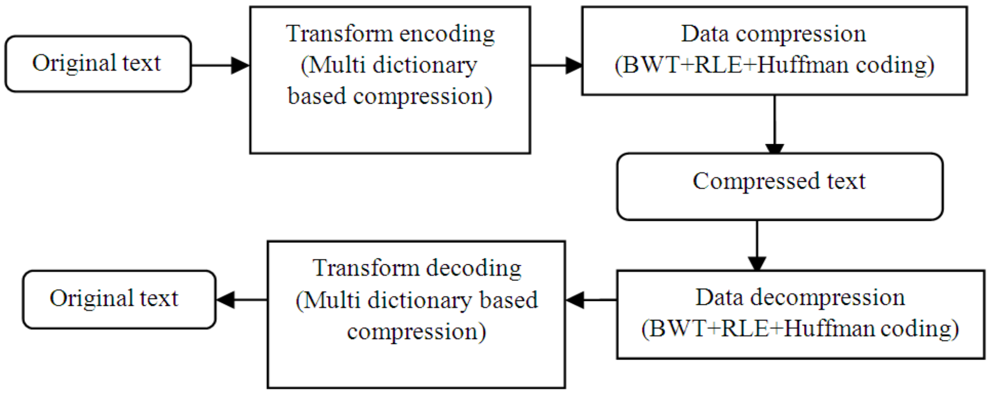
Original text
Transform encoding (Multi dictionary based compression)
Data compression (BWT+RLE+Huffman coding)
Compressed text
Data decompression (BWT+RLE+Huffman coding)
Transform decoding (Multi dictionary based compression)
Original text

Fig. 1. Multidictionary based text compression incorporating a lossless, reversible transformation

Huffman coding, Arithmetic coding and PPM are examples of statisticsl based coding (Sayood, 2012). In this coding scheme variable length of code used for symbols. These high redundant texts increase the performance of some text compression algorithms. Earlier researches (Carus and Mesut, 2010; Bhadade and Trivedi, 2011; Sun *et al*., 2003) use enormous dictionaries of words or phrases and their codes. The common approach employed is to use a coding scheme with high redundancy (Tadrat and Boonjing, 2008). On the other hand, their dictionary sizes make their works not suitable for embedding in compression algorithms. This study seeks for an optimal dictionary as well as a highly redundant coding scheme for such function. In this study different dictionaries with different coding schemes are experimentally investigated with various compression algorithms is apt for redundant texts (Al-Bahadili and Hussain, 2010; Martinez-Prieto *et al*., 2011; Kulekci, 2012). The performance of text compression is increased by text transformation.

The words in the input text are transformed with highly redundant codes by an approach known as multidictionary based text compression. By this approach the input file is first transformed into predefined codes, thereafter it is compressed using BWT, RLE and Huffman coding. On the receiver side it is decompressed using same algorithms and extracted from the compression method as shown in **Fig. 1**. The performance in terms of compression ratio is satisfactory.However a more efficient algorithm will give still better results.

2. MATERIALS AND METHODS

2.1. Dictionary Formation

- The words are extracted from the input test file and a table is formed. The first letter in the words which is in the upper case is converted into the lower case letter
- The frequency of occurrence of the word is calculated, sorted out and the words from the table are arranged in the descending order
- Each word is assigned with an ASCII code .The respective number (33-255) of the each ASCII character is assigned as code except small letters (a...z) and capital letters (A...Z) .So totally 170 character becomes as code
- ASCII character is assigned as code to every word. In table 170 single ASCII character is assigned as a code for first 170 words

!@#$%^&*()_+......... upto ASCII character of 255

- For the next 170 words the same 170 ASCII character with a prefix of character ’a’. Thus it becomes two character codes

a! a@ a# a$ a% a^ a& a* a(a) a_ a+……upto ASCII character of 255

- The remaining words will have the combination of (b….z) and the single 170 ASCII characters. The remaining words will have the combination of (A….Z) and the single 170 ASCII characters

b! b@ b# b$ b% b^ b& b* b(b) b_ b+……upto ASCII character of 255…………..

z! z@ z# z$ z% z^ z& z* z(z) z_ z+……upto ASCII character of 255.

A! A@ A# A$ A% A^ A& A* A (A) A_ A+……upto ASCII character of 255

B! B@ B# B$ B% B^ B& B* B (B) B_ B+……….upto ASCII character of 255................

Z! Z@ Z# Z$ Z% Z^ Z& Z* Z (Z) Z_ Z+ …….. ..Upto ASCII character of 255

- N*170 = Number of words assign as a code for two character combination.

N = Number of alphabetic characters [(a...z) + (A...Z) = 52]

- Further words will have the combination of 170*N codes with prefix of 'a', thus becoming three character code. Similarly each character of (b...z) is the prefix of two character combination.

aa! aa@ aa# aa$ aa% aa^ aa& aa* aa(aa) aa_ aa+……upto ASCII character of 255.

ab! ab@ ab# ab$ ab% ab^ ab& ab* ab(ab) ab_ ab+……upto ASCII character of 255.

AA! AA@ AA# AA% AA^ AA& AA* AA (AA) AA_ AA+ …. Upto ASCII character of 255

- Same coding format followed for (A...Z).

AA! AA@ AA# AA% AA^ AA& AA* AA (AA) AA_ AA+…. Upto ASCII character of 255
BB! BB@ BB# BB$ BB% BB^ BB& BB* BB (BB) BB_ BB+…. upto ASCII character of 255 ……
ZZ! ZZ@ ZZ# ZZ$ ZZ% ZZ^ ZZ& ZZ* ZZ (ZZ) ZZ_ ZZ+ ……. upto ASCII character of 255

- M*N*170 = Number of words assign as a code for three character combination.

M = N = Number of alphabetic characters [(a...z) + (A...Z) = 52].

The shortest code is assigned to most frequently used words. The longest code is assigned to less frequently used words.

2.2. Multidictionary Generation

Multidictionary method helps in extracting the words in the dictionary rapidly and easily:

- Words starting with the letter 'a' are converted into a dictionary similarly remaining alphabets are converted into respective dictionaries
- Codes are assigned to the words in various dictionaries.
- The characters apart from the alphabets such as words starting with ASCII characters are also grouped into separate dictionary

2.3. Encoding Algorithm

2.2.1. BWT Algorithm

The data encoded by multidictionary method is given as input to the BWT transform algorithm:

- The block of data is taken and is rearranged by sorting algorithm. The output of the BWT block will contain the same number of data element; however the order may be different
- In the reverse transform the original order will be sorted without the loss of data
- The BWT is performed on the entire block of data elements at once
- The lossless compressions algorithms operate in streaming mode, reading single byte or few bytes at a time. BWT transform operates on large chunks of data. It further operates on data in the memory and encounters files that are too big to process in one swoop. In those cases the file must be split up and processed as blocks and termed as parallel BWT transform
- The files that are divided into n number of blocks are given as input to BWT transform
- Finally on the other side the output blocks are combined and obtained as single output block. This parallel BWT transform method increases the speed is rapidly. The output of this method is given as input to the run length coding (Sayood, 2012)

2.4. Run Length Coding

- Run length coding is widely used data compression algorithm. The main feature of the algorithm is to replace the long sequence of the same symbol by a short sequence. The output of the BWT generally has runs
- Runs refer to the continual occurring of same symbols. The RLE has a specific role in conversion of such long sequences into a short sequence by substituting the number of repetition of that particular symbol before the special character '@' and the repeated symbol is followed by this special character (Salomon, 2007)

2.5. Huffman Coding

- The output of the RLE coding is a given as an input to the Huffman coding
- The number of occurrence is determined and a code is generated using Huffman coding. This leads to further compression of the input file. Huffman coding has a unique method for choosing the representation for each symbol, resulting in a prefix code. No other mapping of individual source symbols to unique strings of bits will produce a smaller average output size. Huffman coding is an extensive method for creating prefix codes (Sayood, 2012)

2.6. Decoding Algorithm

- Huffman decoding algorithm decodes the binary code from the encoded output. This Huffman decoding output is given as input to the RLE algorithm (Sayood, 2012)
- The RLE decoded output converts short sequence symbol into a long sequence symbol (Salomon, 2007)
- After this conversion the output is given as an input to the BWT reverse transform which rearranges the data into an original order
- The combination of (a...z) or (A...Z) with ASCII character is considered as a code and the equivalent word of the code is searched in the corresponding dictionary
- Similarly with this combination if two consecutive ASCII character occurs, it is extensively considered as a separate code and searched in the respective dictionary. Finally the words are collected in the output file

3. RESULTS

We performed experiments on the MBRH transformation algorithms using standard Calgary Corpus text file collection and compared with some standard existing compression algorithm Eq. 1 and 2:

$$\text{Compression ratio} = \frac{\text{Output file size}}{\text{Input file size}} \tag{1}$$

$$\text{Bit per characer (BPC)} = \frac{\text{Output file size}}{\text{Input file size}} * 8 \tag{2}$$

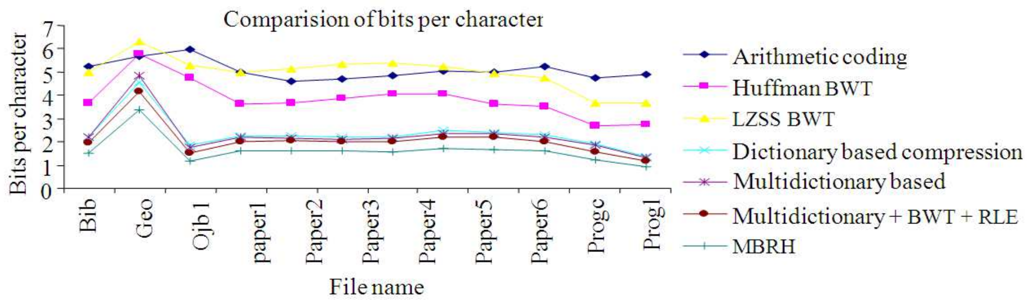

Fig. 2. Comparison of BPC

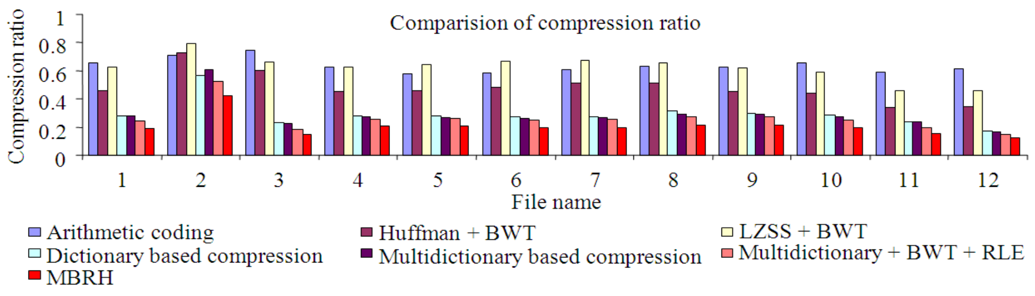

Fig. 3. Comparison of compression ratio

Table 1. List of files used in experiments

File name	Size (byte)	Description
Bib	111,261	Bibliography
Geo	102,400	Geological seismic data
Obj1	21,504	VAX object program
paper1	53,161	Technical Paper
Paper2	82,199	Technical Paper
Paper3	46,526	Technical Paper
Paper4	13,286	Technical Paper
Paper5	11,954	Technical Paper
Paper6	38105	Technical Paper
Progc	39,611	Source Code in “C”
Progl	71,646	Source Code in “Pascal”
Progp	49,379	Text: English Text

4. DISCUSSION

The test files specified in **Table 1** are programmed by Matlab for implementation of MBRH and are compared with various compression algorithms such as arithmetic coding, Huffman with BWT, LZSS with BWT and Dictionary Based Encoding (DBE) and multidictionary based compression, multidictionary BWT with RLE and MBRH. By using equations (1, 2), compression ratio and bits per character are calculated. The comparison is shown in **Table 2 and 3**.The results are shown graphically in **Figure 2 and 3**. They show that MBRH out performs the other techniques in terms of compression ratio and Bits Per Character (BPC). **Table 4** shows compression time of input file size to compression code for each algorithm. Compression ratio is increased in MBRH compared with other dictionary based compression. MBRH achieves less transmission time.

Table 2. Comparison of BPC

	Bits per character						
File Name	Arithmetic coding	Huffman BWT	LZSS BWT	Dictionary based compression	Multidictionary based compression	Multidictionary +BWT +RLE	Multidictionary+ BWT + RLE + Huffman coding(MBRH)
Bib	5.232	3.656	5.016	2.224	2.219	1.955	1.538
Geo	5.656	5.800	6.304	4.560	4.857	4.168	3.386
Obj1	5.968	4.768	5.288	1.856	1.765	1.495	1.187
paper1	4.984	3.616	4.976	2.256	2.183	2.028	1.615
paper2	4.624	3.680	5.136	2.256	2.161	2.077	1.630
paper3	4.712	3.856	5.336	2.200	2.097	2.009	1.596
paper4	4.824	4.064	5.376	2.212	2.136	2.018	1.562
paper5	5.064	4.056	5.256	2.480	2.350	2.194	1.700
paper6	5.008	3.632	4.952	2.408	2.349	2.186	1.686
Progc	5.240	3.504	4.728	2.288	2.206	1.987	1.596
Progl	4.760	2.680	3.648	1.896	1.882	1.560	1.243
Progp	4.896	2.760	3.688	1.392	1.317	1.153	0.950

Table 3. Comparison of compression ratio

		Compression ratio						
File Name	Input file size (byte)	Arithmetic coding	Huffman + BWT	LZSS + BWT	Dictionary based compression	Multidictionary based compression	Multidictionary+ BWT+RLE	Multidictionary+ BWT + RLE + Huffman coding (MBRH)
Bib	111261	0.654	0.457	0.627	0.2780	0.277	0.244	0.192
Geo	102400	0.707	0.725	0.788	0.5700	0.607	0.521	0.423
Obj1	215040	0.746	0.596	0.661	0.2320	0.221	0.187	0.148
paper1	531601	0.623	0.452	0.622	0.2820	0.273	0.254	0.202
paper2	821990	0.578	0.460	0.642	0.2820	0.270	0.260	0.204
paper3	465260	0.589	0.482	0.667	0.2750	0.262	0.251	0.199
paper4	132860	0.603	0.508	0.672	0.2765	0.267	0.252	0.195
paper5	119540	0.633	0.507	0.657	0.3100	0.294	0.274	0.213
paper6	381050	0.626	0.454	0.619	0.3010	0.294	0.273	0.211
progc	396110	0.655	0.438	0.591	0.2860	0.276	0.248	0.200
progl	716460	0.595	0.335	0.456	0.2370	0.235	0.195	0.155
progp	493790	0.612	0.345	0.461	0.1740	0.165	0.144	0.119

Table 4. Comparison of compression time

File Name	Dictionary based compression (sec)	Multidictionary based compression (sec)	BWT (sec)	Runlength coding (sec)	Huffman coding (sec)
Bib	638	85.770	9.0	1.4	1.5
Geo	1620	409.500	10.0	2.0	1.7
Obj1	180	26.330	4.7	0.3	0.8
paper1	207	41.340	3.2	0.7	1.1
paper2	264	48.490	6.5	0.9	1.2
paper3	182	30.320	1.8	0.6	1.0
paper4	42	3.896	2.5	0.4	1.3
paper5	47	4.719	2.4	0.3	0.9
paper6	118	29.790	0.7	0.6	1.0
Progc	142	33.060	0.2	0.7	1.0
Progl	167	53.950	3.8	0.8	1.1
Progp	125	21.160	0.3	0.6	0.9

5. CONCLUSION

This study proposes a new method of text transformation using Multidictionary based encoding. In a channel, the amount of compression paves way for reduction in transmission time. The input text is replaced by variable length codes, the size of input text can be reduced by using Multidictionary based compression. MBRH compression algorithm attains good compression ratio, reduces bits per character and conversion time.

6. REFERENCES

1. Al-Bahadili, H. and S.M. Hussain, 2010. A bit-level text compression scheme based on the ACW algorithm. Int. J. Automat. Comput., 7: 123-131. DOI: 10.1007/s11633-010-0123-6
2. Abel, J. and W. Teahan, 2005. Universal text preprocessing for data compression. IEEE Trans. Comput., 54: 497-507. DOI: 10.1109/TC.2005.85
3. Begum, M.B. and Y. Venkataramani, 2012. LSB based audio steganography based on text compression. Proc Eng., 30: 703-710. DOI: 10.1016/j.proeng.2012.01.917
4. Bhadade, S.U. and A.I. Trivedi, 2011. Lossless text compression using dictionaries. Int. J. Comput. Appli., 13: 27-340. http://www.ijcaonline.org/volume13/number8/pxc3871767.pdf
5. Carus, A. and A. Mesut, 2010. Fast text compression using multiple static dictionaries. Inform. Technol. J., 9: 1013-1021. http://altanmesut.trakya.edu.tr/pubs/1013-1021.pdf
6. Kulekci, M.O., 2012. On scrambling the burrows-wheeler transform to provide privacy in lossless compression. Comput. Security, 31: 26-32. DOI: 10.1016/j.cose.2011.11.005
7. Li, L., K. Chakrabarty and N.A. Touba, 2003. Test data compression using dictionaries with selective entries and fixed-length indices. ACM Trans. Design Automat. Elect. Syst., 8: 470-490. DOI: 10.1145/944027.944032
8. Mohan, B.S. and V.K. Govindan, 2005. Compression scheme for faster and secure data transmission over networks. Proceedings of the International Conference on Mobile Business, Jul. 11-13, pp: 678-681. DOI: 10.1109/ICMB.2005.29
9. Martinez-Prieto, M.A., J. Adiego and P.D.L. Fuente, 2011. Natural language compression on edge-guided text preprocessing. Inform. Sci., 181: 5387-5411. DOI: 10.1016/j.ins.2011.07.039
10. Salomon, D., 2007. Data Compression: The Complete Reference. 4th Edn., Springer, London, ISBN-10: 9781846286032, pp: 1092.
11. Sun, W., N. Zhang and A. Mukherjee, 2003. A dictionary-based multi-corpora text compression system. Proceedings of the 2003 IEEE Data Compression Conference, Mar. 27-27, IEEE Xplore Press, USA. DOI: 10.1109/DCC.2003.1194067
12. Sayood, K., 2012. Introduction to Data Compression. 4th Edn., Morgan Kaufmann, Waltham, ISBN-10: 9780124157965, pp: 768.
13. Sun, W., Z. Zhang and N., Mukherjee, 2003. A dictionary-based fast transform for text compression. Proceedings of the International Conference on Information Technology, Computers and Communications, Apr. 28-30, IEEE Xplore Press, pp: 176-182. DOI: 10.1109/ITCC.2003.1197522
14. Tadrat, J. and V. Boonjing, 2008. An experiment study on text transformation for compression using stoplists and frequent words. Proceedings of the 5th International Conference on Information Technology: New Generations, Apri 4-9, IEEE Xplore Press, DC, USA., pp: 709-713. DOI: 10.1109/ITNG.2008.178