IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 11, Issue 6 (May. - Jun. 2013), PP 15-19
www.iosrjournals.org

Comparison Study of Lossless Data Compression Algorithms for Text Data

Arup Kumar Bhattacharjee[1], Tanumon Bej[2] , Saheb Agarwal[3]
[1, 2, 3]*(Dept. of MCA, RCC Institute of Information Technology, India)*

Abstract : *Data Compression is the technique through which, we can reduce the quantity of data, used to represent content without excessively reducing the quality of the content. This paper examines the performance of a set of lossless data compression algorithm, on different form of text data. A set of selected algorithms are implemented to evaluate the performance in compressing text data. A set of defined text file are used as test bed. The performance of different algorithms are measured on the basis of different parameter and tabulated in this article. The article is concluded by a comparison of these algorithms from different aspects.*
Keywords *- Encryption, Entropy Encoding, Dictionary Encoding, Compression Ratio, Compression time, Test Bed.*

I. Introduction:

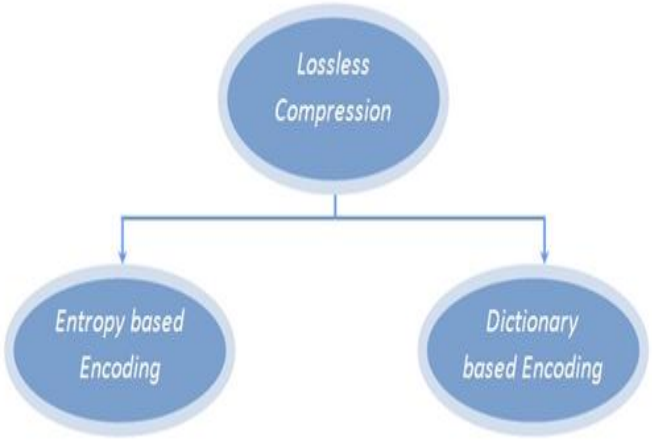
Compression is the art of representing information in a compact form rather than its original or uncompressed form [1]. The main objective of data compression is to find out the redundancy and eliminate them through different efficient methodology; so that the reduced data can save, space: to store the data, time: to transmit the data and cost: to maintain the data. To eliminate the redundancy, the original file is represented with some coded notation and this coded file is known as 'encrypted file'. For any efficient compression algorithm this file size must be less than the original file. To get back the original file we need to 'decrypt' the encoded file. In this paper we have selected 10 different file as test bed, and implemented 3 lossless compression algorithms, namely Huffman Compression Algorithm, Shannon Fano Compression Algorithm, Lempel Zev Welch (LZW) Compression Algorithm; and applied them on those test bed to evaluate the compression time, compression ratio, decompression time etc.

II. Types Of Compression:

Compression can be of two types: Lossless Compression, Lossy Compression.

2.1) Lossless Compression: In the process compression if no data is lost and the exact replica of the original file can be retrieved by decrypting the encrypted file then the compression is of lossless compression type. Text compression is generally of lossless type. In this type of compression generally the encrypted file is used for storing or transmitting data, for general purpose use we need to decrypt the file.

Lossless compression technique can be broadly categorized in to two classes:



i) Entropy Based Encoding: In this compression process the algorithm first counts the frequency of occurrence of each unique symbol in the document. Then the compression technique replaces the symbols with the algorithm generated symbol. These generated symbols are fixed for a certain symbol of the original document; and doesn't depend on the content of the document. The length of the generated symbols is variable and it varies on the frequency of the certain symbol in the original document.

ii) Dictionary Based Encoding: This encoding process is also known as substitution encoding. In this process the encoder maintain a data structure known as '**Dictionary**' [3]. This is basically a collection of string. The encoder matches the substrings chosen from the original text and finds it in the dictionary; if a successful match is found then the substring is replaced by a reference to the dictionary in the encoded file.

2.2) Lossy Compression: Lossy Compression is generally used for image, audio, video; where the compression process neglects some less important data. The exact replica of the original file can't be retrieved from the compressed file. To decompress the compressed data we can get a closer approximation of the original file.

III. Measurement Parameter:

Depending on the use of the compressed file the measurement parameter can differ. Space and time efficiency are two most important factors for a compression algorithm.

Performance of the compression algorithm largely depends on the redundancy on the source data. So to generalize the test platform we used same test files for all the algorithms. The parameters were as follows:

Compression Ratio: The ratio between the compressed file and the original file.

$$\text{Compression Ratio} = \frac{Compressed\ file\ size}{Original\ File\ size}$$

Compression Factor: The ratio between the original file and the compressed file.
This is basically the inverse of the Compression Ratio.

$$\text{Compression Factor} = \frac{1}{Compression\ \ Ratio} \text{ OR } \frac{Original\ \ File\ size}{Compr\ essed\ \ file\ size}$$

Saving Percentage: The percentage of the size reduction of the file, after the compression.

$$\text{Saving Percentage} = \frac{Original\ file\ size - Compressed\ file\ size}{Original\ file\ size}\%$$

Compression Time: The time taken by the algorithm to compress the file. Calculated in milliseconds (ms).

Decompression Time: The time taken by the algorithm to decompress and retrieve the original file from compressed file. It is also calculated in milliseconds.

The compression time and decompression time is important in case of the applications where the algorithms are used to transmit the data, or to store the data in a secondary storage and retrieve it as required.

IV. Compression Algorithms:

Three lossless algorithms (two of Entropy encoding type: Huffman Compression & Shannon Fano Compression and one Dictionary encoding type: LZW Compression) was implemented using JAVA.

4.1) Huffman Compression: Huffman coding is used for lossless data compression. The term refers to the use of a variable length code for encoding a source symbol (such as a character in a file).The variable-length code table has been derived in a particular way, based on the estimated probability of occurrence for each possible value of the source symbol. In this compression technique a table is created incorporating the no of occurrences of an individual symbol, this table is known as **frequency table.** This table is arranged in a certain order and then a tree is generated from that table, in this tree high frequency symbols are assigned codes which have fewer bits, and less frequent symbols are assigned codes with many bits. In this way the **code table** is generated.

4.2) Shannon Fano Compression: Named after Claude Shannon and Robert Fano, variable length code for encoding a source symbol, lossless data compression scheme, Entropy encoding, According to Shannon's source coding theorem, the optimal code length for a symbol is $-\log_b P$, where b is the number of symbols used to make output codes and P is the probability of the input symbol [5, 6]. Similar to the Huffman coding, initially a frequency table is generated, then a particular procedure is followed to produce the code table from frequency

4.3) LZW Compression: Named after Abraham Lampel , Jacob Zev and Terry Welch, it is dictionary coder or substitution coder, which means a dynamic dictionary is created depending upon the presence of substring chosen from the original file. Then the substring is matched with the Dictionary, if the string is found then a reference of the dictionary is mentioned in the encoded file, if the string is not found then a new dictionary entry is made with a new reference [8].

In all algorithms the encoded file contains the code table/ Dictionary and the encoded text; the encoder matches the codes with the directory (code table/ dictionary) and retrieves the original text iteratively.

V. Test Bed:

For test bed we have selected 10 files to test the algorithms. The descriptions of the files are as follows:

Test beds	Size(kb)	type	Extension	Content
1.	93.5	Rich text document	.doc	Content of book, less white space, less repetition of symbol
2.	50	Spreadsheet document	.xls	List document, repetition of same type of word, less white space
3.	33.5	Rich text document	.doc	Content of database query, contains lots of white space and tab space
4.	516	Database file	.mdb	Collection of tables, less white space & null value, repetition of same type of word
5.	208	Database file	.mdb	No null value, fewer table data
6.	6.11	Text document	.txt	Same content repeated multiple time, contains white space
7.	99.5	Spreadsheet document	.xls	Repetition of same type of word, less null value
8.	44	Rich text document	.doc	Contains programs, lots of special character, lots of white space, repetition of same character as well as same word
9.	135	Rich text document	.docx	Output sequence, contains lots of white space
10.	45.1	Text document	.txt	Contains programs, lots of tab space, and same contain repeated multiple time

VI. Result:

The result found by implementing the algorithms over the test files we get the following results:

6.1) Huffman Compression:

Original File		Huffman Compression & Decompression					
S.No	File Size (Bytes)	Comp File Size (Bytes)	Compression Ratio	Compression Factor	Saving Percentage	Comp Time (mS)	Decomp Time (mS)
1	96256	54272	0.56383	1.773585	0.43617	180	1110
2	51200	26624	0.52	1.923077	0.48	90	550
3	34816	13312	0.382353	2.615385	0.617647	60	300
4	528384	139264	0.263566	3.794118	0.736434	250	2990
5	212992	44032	0.206731	4.837209	0.793269	160	1110
6	7168	5120	0.714286	1.4	0.285714	20	100
7	102400	61440	0.6	1.666667	0.4	260	1240
8	45056	20480	0.454545	2.2	0.545455	100	440
9	139264	80896	0.580882	1.721519	0.419118	320	1890
10	47104	29696	0.630435	1.586207	0.369565	130	500

6.2) Shannon Fano Compression:

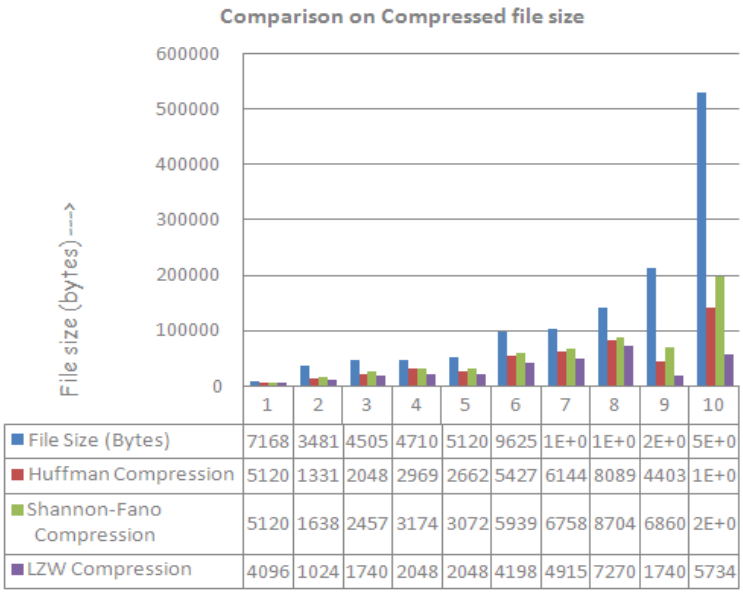
Original File		Shannon-Fano Compression & Decompression					
S.No	File Size (Bytes)	Comp File Size (Bytes)	Compression Ratio	Compression Factor	Saving Percentage	Comp Time (mS)	Decomp Time (mS)
1	96256	59393	0.617032	1.620662	0.382968	260	1550
2	51200	30720	0.6	1.666667	0.4	130	620
3	34816	16384	0.470588	2.125	0.529412	90	400
4	528384	196608	0.372093	2.6875	0.627907	390	4160
5	212992	68608	0.322115	3.104478	0.677885	250	1460
6	7168	5120	0.714286	1.4	0.285714	30	80
7	102400	67584	0.66	1.515152	0.34	270	1320
8	45056	24576	0.545455	1.833333	0.454545	50	590
9	139264	87040	0.625	1.6	0.375	270	1700
10	47104	31744	0.673913	1.483871	0.326087	60	510

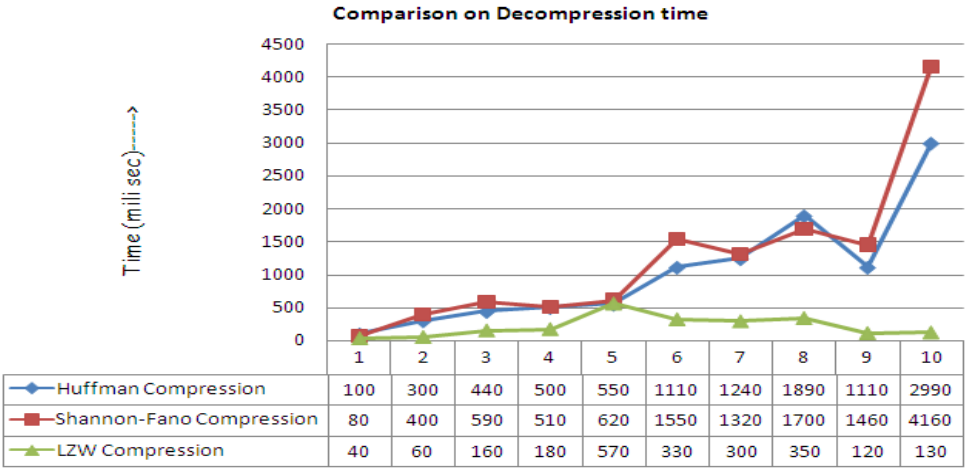
6.3) LZW Compression:

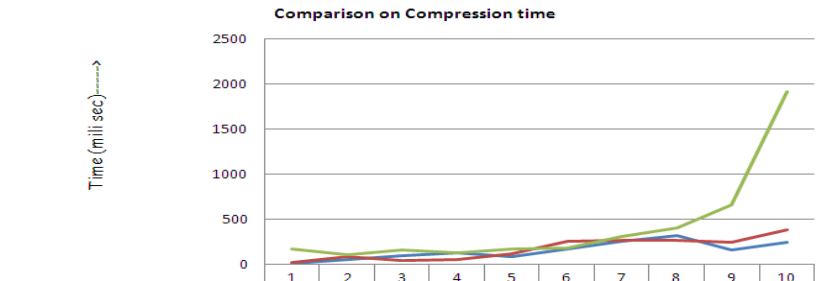
Original File		LZW Compression & Decompression					
S.No	File Size (Bytes)	Comp File Size (Bytes)	Compression Ratio	Compression Factor	Saving Percentage	Comp Time (mS)	Decomp Time (mS)
1	96256	41984	0.43617	2.292683	0.56383	180	330
2	51200	20480	0.4	2.5	0.6	170	570
3	34816	10240	0.294118	3.4	0.705882	110	60
4	528384	57344	0.108527	9.214286	0.891473	1910	130
5	212992	17408	0.081731	12.23529	0.918269	660	120
6	7168	4096	0.571429	1.75	0.428571	170	40
7	102400	49152	0.48	2.083333	0.52	310	300
8	45056	17408	0.386364	2.588235	0.613636	160	160
9	139264	72704	0.522059	1.915493	0.477941	410	350
10	47104	20480	0.434783	2.3	0.565217	130	180

VII. Comparison:

From the data presented in the above tables it can be concluded that LZW provide better result than other two methods. The values drawn from the three algorithms on the basis of compressed file size, compression time and decompression time are compared through graphical representation. For the cardinality of the graph the data were sorted according to the ascending order of the file size.







	1	2	3	4	5	6	7	8	9	10
Huffman Compression	20	60	100	130	90	180	260	320	160	250
Shannon-Fano Compression	30	90	50	60	130	260	270	270	250	390
LZW Compression	170	110	160	130	170	180	310	410	660	1910

VIII. Conclution and Future Scope:

From the compression and decompression time we can conclude that LZW take much more time to compress a file than the other two algorithms whereas decompression of a file is much faster than other two algorithms. The average decompression time of Shannon Fano algorithm is higher than the Huffman Decompression. In case of decompressed file size LZW give better result than other two algorithms, however it can also be concluded that depending on the content of the original file, the performance of the algorithm varies.

In this paper we have compared three lossless data compression algorithm and our text bed was limited in text data, In future, more compression algorithms(both lossless and lossy) can be implemented over a larger test bed which includes audio, video and image data. And then a system can be implemented which will detect the file type and then depending on that it will choose the appropriate compression technique for the file.

Reference:

[1] Pu, I.M., 2006, *Fundamental Data Compression*, Elsevier, Britain.
[2] Kaufman, K. and T. Shmuel, 2005. *Semi-lossless text compression*, Intl. J. Foundations of Computer Sci., 16: 1167-1178.
[3] Kesheng, W., J. Otoo and S. Arie, 2006. *Optimizing bitmap indices with efficient compression*, ACM Trans. Database Systems, 31: 1-38.
[4] Introduction to Data Compression, Khalid Sayood, Ed Fox(Editor), March 2000
[5] Shannon, C.E. (July 1948). "A Mathematical Theory of Communication". Bell System Technical Journal 27: 379–423.
[6] Fano, R.M. (1949). "The transmission of information". Technical Report No. 65 (Cambridge (Mass.), USA: Research Laboratory of Electronics at MIT).
[7] Terry Welch, "A Technique for High-Performance Data Compression", IEEE Computer, June 1984, p. 8–19.
[8] Jacob Ziv and Abraham Lempel; Compression of Individual Sequences Via Variable-Rate Coding, IEEE Transactions on Information Theory, September 1978.
[9] Improved wordaligned binary compression for text indexing, Vo Ngoc and M. Alistair, 2006. IEEE Trans. Knowledge & Data Engineering, 18: 857-861.
[10] Data compression using dynamic Markov modeling, Cormak, V. and S. Horspool, 1987. Comput. J., 30: 541–550.
[11] Bounds on the redundancy of Huffman codes, Capocelli, M., R. Giancarlo and J. Taneja, 1986. IEEE Trans. Inf. Theory, 32: 854–857.
[12] Data compression for estimation of the physical parameters of stable and unstable linear systems, Gawthrop, J. and W. Liuping, 2005. Automatica, 41: 1313-1321.