ISSN: 2348 - 2117

Available Online at www.ijetcr.org

International Journal of Engineering Technology and Computer Research (IJETCR) 2 (1) 2014, 16-19

Analysis and Performance Comparison of Lossless Compression Techniques for Text Data

Koush Rastogi[1], Kanchan Sengar[2]

[1] M.Tech Scholar, JECRC University, Jaipur, India

[2] Assistant Professor, JECRC University, Jaipur, India

Received 25 January 2014; Accepted 10 February 2014

ABSTRACT:
For most of the computerized application used presently data. Compression is an essential requirement. Compression is the process which saves memory space by reducing the number of bits used to represent certain message. Different data compression techniques exist for compression of different data types. Data compression can be classified as lossless or lossy compression. Run Length Encoding, Huffman Coding, and Shannon Fano Coding techniques for lossless data compression are discussed in this paper. The article is concluded by doing a comparison of these techniques.

Key Word: *Lossless compression, Lossy compression, Run Length Encoding, Huffman Coding, Shannon Fano Coding*

INTRODUCTION:

The science and art of representing data in a more compact form so that it requires less memory space compared to original uncompressed form is called as data compression [1]. Hence data compression make task of processing, storing or transferring a huge file easier. The primary goal when transmitting any data is to achieve high speed. Number of bits sent, time consumed by encoder to generate coded information and the time required by the decoder to reconstruct the original message are factors which determine speed of transmission. In data storage, amount of compression is big factors. Hence for both tasks of transmission and storage data compression is an essential step. Data compression can be performed either in lossy or lossless ways. Lossless compression techniques are able to reconstruct data from compressed message in exactly same form as original message. Their application are in medical images, text and images stored for legal matters, executable file of computer programs [2]. On the contrary, lossy compression techniques are irreversible techniques in which it is not possible to reconstruct message in exactly same form as original and some distortion occurs while recreating original message from compressed form [3]. The approximate reconstruction techniques of lossy compression achieve more compression ratio and their common application includes the compression of multimedia images, audio and video files.

Compression ratio = B_1/B_0*100% (1)

B_0 = no of bits before compression

B_1 = no of bits after compression

2. RUN LENGTH ENCODING:

RLE is the simplest algorithm for lossless compression and its main aim is to check for redundancy in the consecutive sequence of symbols. The repeating sequences of symbols are identified as runs and the non repeating patterns are defined as non runs by this algorithm. For example, if text ABCCCAAABB is given as an input, then first two letters are identified as non run with length 2 and next three letters are identified as a run with length 3 since they are a repetition of symbol C. The main task of this algorithm is to store the symbol and length of each run. The algorithm then uses these runs to compress the original text whereas non runs are not used in compression.

A. Algorithm:

The algorithm for RLE consists of following steps
Step 1. Choose the character from source message
Step 2. Append the character to the compressed string
Step 3. Count the repeating occurrences of the choosen character and transfer the count to compressed string
Step 4. Choose next character and repeat steps 2,3,4 till the string is finished.
Run Length Coding is not much effective in English text. For example, the sentence

*Corresponding author: Koush Rastogi

Laxman_is_brother_of_Ram has no repeated sequences of any alphabet, hence there will be no space savings.

3. HUFFMAN CODING:

Huffman codes were invented by David A. Huffman[4]. Huffman code is minimum length code because no other encoding has a shorter average length of compressed data. Huffman coding is a greedy algorithm since it makes an optimal selection of symbols at each stage of compression to find most optimal solution.

A Huffman encoder takes a block of input characters with fixed length and produces a block of output bits which are of variable length. It is a fixed-to-variable length code. It is entropy based coding; hence the basic task in Huffman coding is to assign short codewords to those input symbols with high probabilities of occurrence and long codewords to those with low probabilities. Huffman coding is a static compression algorithm, hence encoding table is built by analyzing the entire document before the compression step.Huffman encoding is also classified into two types which are Static Huffman Coding and Adaptive Huffman coding. In Static Huffman Algorithm, first statistical table is generated by calculating frequencies and then same tree is used for both decompression and compression process. Hence there is an overhead because Huffman tree must be saved and transferred with compressed file. On the contrary, in Adaptive Huffman coding two different trees for compression and decompression process are used.

A .Properties:

1. Prefix property - No code word is prefix of any other code word. Hence, decoding is unambiguous.
2. If prior statistical table is built accurately, then Huffman coding gives best results.
3. The best case and most optimal solution for Huffman coding occur when probability of each symbol is negative power of 2.
4. When probability of a symbol exceeds 0.5 then it results in worst case for Huffman encoding.

B. Algorithm:

Step 1. Calculate the probability for each character.
Step 2. Sort the list in ascending order.
Step 3. Create a new node whose left child is at lowest position in the sorted list and right child
is at second lowest position in sorted list.
Step 4. The probabilities of these two elements are added to give probability of new node and
they are replaced by new node formed above.
Step 5. Sort the list in ascending order with the new node.
Step 6. Repeat steps 2,3,4,5 until only one node called parent node is left.
Step 7. Create the coding tree Step 8. Calculate entropy.

Table 1: Huffman Coding

Symbol	Frequency	Huffman Code	No. of bits used
A	7	00	14
B	5	01	10
C	4	10	8
D	4	11	8

Results of Huffman Coding

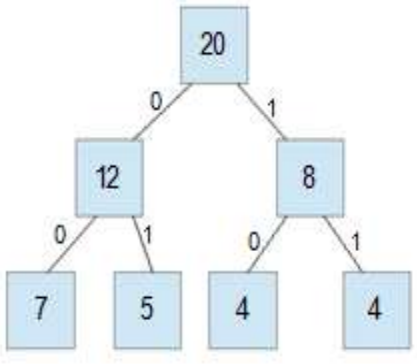

Figure 1: Huffman Coding Tree

Analysis of Shannon Fano compression

Total number of bits used = 40
Before compression = 20 * 8 = 160
Compression percentage = 40//160 *100 = 25% from original source.
Hence applying Huffman compression has resulted in 75% of space savings.

4. SHANNON FANO CODING:

It was proposed by Claude Shannon [5] and Robert Fano [6] in late 1940s. The main difference between the two methods is that Shannon - Fano constructs its codes from top to bottom and the bits of every codeword are constructed from left to right , while in Huffman coding the code tree is constructed bottom up and bits of each codeword are constructed from right to left. To create a code tree according to Shannon and Fano an ordered statistical table is required providing the frequency of any symbol. Each part of the table will be divided into two segments in each iteration and the algorithm has to ensure that the upper and the lower part of the segment have same or nearly the same sum of frequencies. This procedure will be repeated until only single symbols are left.

A.Algorithm:

Step 1. Create statistical table providing frequencies / probability of each symbol.
Step 2. Sort symbols according to probability of each symbol in descending order.
Step 3. Divide the table recursively into two parts such that lower and upper part have same or nearly same number of frequency.
Step 4. Append a binary 0 to the code of upper part and a binary 1 to the code for lower part.
Step 5. Repeat steps 3 and 4 for each part containing more than two symbols.
Step 6. Create the coding tree.
Step 7. Calculate entropy.

Table 2: Shannon Fano Coding

Symbol	Frequency	Huffman Code	No. of bits used
A	10	00	20
B	8	01	16
C	7	10	14
D	4	110	12
E	3	111	9

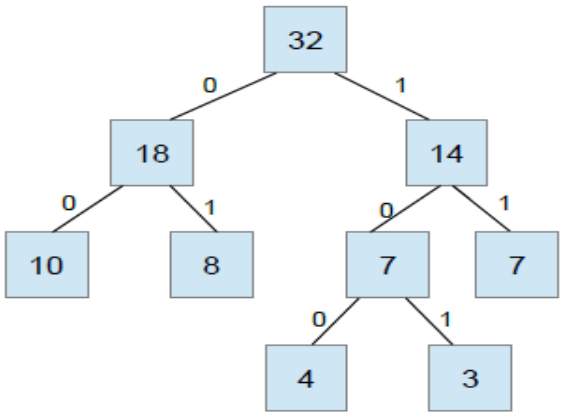

Figure 2: Shannon Fano Coding Tree

Analysis of Shannon Fano compression
Total number of bits used = 71
Before compression = 32 * 8 = 256
Compression percentage = 71/256 *100 = 27.73% from original source.
This means 72 % of savings in memory.

Table 3.Comparison of Lossless Compression Algorithms

Algorithm	Run Length Encoding	Huffman	Shannon Fano
Advantages	1.SimpleImplementation 2.Execution is fast 3.Losslesscompression 4. Does not require statistical table.	1.Easy Implementation 2.Lossless compression 3. Produces an optimal and compact code. 4.It is free to use	1.Produces compact code 2.Lossless compression 3. There is no patent, hence free to use.
Disadvantage	1.Compression ratio is very low compared to other algorithms	1.Relatively slow 2. Statistical table is required to implement. 3. Different code length make difficult to decompress. 4.Huffman tree is required which creates overhead	1. Statistical Table is required to implement. 2. Lower code efficiency compared to Huffman.
Applications	Used mostly for TIFF, BMP And PCX files.	Used in JPEG.	Used in IMPLODE compression in ZIP format.

6. CONCLUSION:

Data compression techniques are most essential in modern times to easily handle the task of storage and transmission. Lossless compression techniques give less compression ratio but produce exactly same data. Huffman technique is most optimal for Lossless compression.

7. REFERENCES:

1. Pu, I.M., 2006, Fundamental Data Compression, Elsevier, Britain.
2. Blelloch, E., 2002. Introduction to Data Compression, Computer Science Department, Carnegie Mellon University.
3. Kesheng, W., J. Otoo and S. Arie, 2006. Optimizing bitmap indices with efficient compression, ACM Trans. Database Systems, 31: 1-38
4. Huffman, David A. " A Method for the construction of Minimum-Redudancy Codes", Proceedings of the IRE , 40:1098-1101(September 1952).
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.