Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 6.199

IJCSMC, Vol. 9, Issue. 3, March 2020, pg.100 – 105

Lossless Compression for Text Document Using Huffman Encoding, Run Length Encoding, and Lempel-Ziv Welch Coding Algorithms

Divya M S[1]; J. Chandrashekhara[2]; Vinay .S[3]; A. Ramadevi[4]
[1]Department of Studied and Research in Computer Science & Davanagere University, India
[2]Department of Studied and Research in Computer Science & Davanagere University, India
[3]Department of Studied and Research in Computer Science & Davanagere University, India
[4]Department of Studied and Research in Computer Science & Davanagere University, India
[1] divyams2346@gmail.com; [2] chandrashekharjk92@gmail.com; [3] vinaygnanesh@yahoo.com; [4] aramadevi86@gmail.com

Abstract- The main ideology of this concept is offering how text document is compressed by lossless compression schemes. One of the main issue while sending a data, data in the sense image, audio, video, or any legal document from source to the destination is the time taken for uploading, transmitting and downloading the data. If the size of the data is more it takes more time to upload at the sender side and download at the receiver side. They also consumes large amount of storage space. There arises a need to reduce the data size and storage space. Image Compression decreases the amount of data needs to be stored by removing redundant data present in the data. It also reduces the number of bits needed to represent the image, audio, video, text/program document. With these basic concept of image compression, in this concept we are mentioned the techniques to solve lossless compression of text documents.
Keywords- Compression, Symbols, Compression Ratio

I. INTRODUCTION

Comparing to the past era there has been a lot of changes in the way of communication. Multimedia way of communication is increasing, this demands internet to send the data from source to the destination. If the size of data is more it takes huge amount of storage space and also takes more time for uploading, transmitting and downloading, if the data contain redundant information this also leads to increase in the storage space and size of the file. Image compression decreases the amount of data needed to store and also reduces the size of file.

By this concept the data is encoded at the sender side and decoded at the receiver side. While compressing the data, it requires exact reproduction of the original data at the receiver side then not possible to loss of quality, clarity of the data so lossless compression scheme is used. This scheme involves no loss of information; the original data can be built accurately from the compressed data.

With respect to above mentioned the concept is introduced to get accuracy of data and information through these algorithms. These algorithms offers sending and storing less weighted data by removing the redundancy existing in the data, it also sends the data what the receiver is in need by removing the unnecessary information present in the data.

Lossless compression scheme uses three different techniques to compress the data i.e,

(i) Huffman Encoding
(ii) Run Length Encoding
(iii) Lempel-Ziv Welch

These three techniques works by encoding and decoding the data, each individual techniques have different algorithms.

Compression models have two distinct structures:

Encoder: Which takes the input and fed into the encoder which creates the symbol set.

Decoder: Which takes the output of the encoded data and reconstructs the output data which is same as input data

Lossless compression scheme takes the input data and converts to symbols by using the frequency values it gives the code word as a result, the efficiency of the encoder depends on the accuracy of the frequency value. Decoding is exactly the reverse process of encoding. Coding the data takes less space than the string compression.

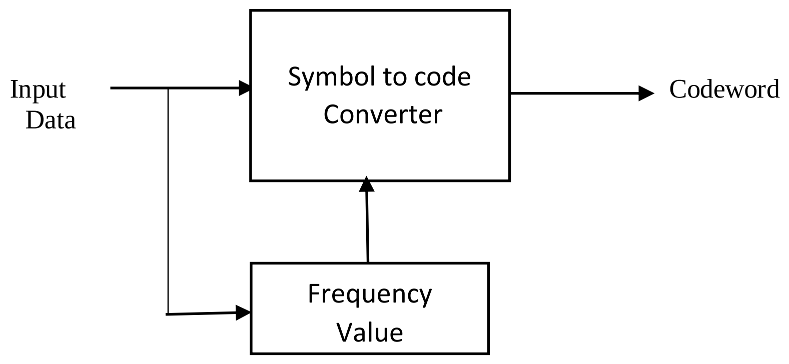


Fig: Lossless compression scheme general block diagram.

II. LOSSLESS COMPRESSION TECHNIQUES

Huffman Encoding: It is a lossless compression technique, and it is based on Binary-tree frequency sorting method, allows to encode message sequence into shorter encoded message.

Algorithm:

Step 1: Draw a table which consists of Symbol and Frequency columns. Enter each individual characters into the symbol column and the number of occurrences of that character into the frequency column.

Step 2: Sort the nodes in ascending order based on the frequency values.

Step 3: Create a Parent node by joining the 2 least frequency value nodes, then add these 2 frequency values and label it to the parent node.

Step 4: Repeat step 3 until all the nodes are created, then combine the resultant parent nodes to generate a new parent node, then add child node values and label it as parent node.

Step 5: if any node remains, then add to uppermost parent node, add the values and label it as root node.

Step 6: Label each left arrow as 0, and right arrow as 1.

Step 7: To encode the characters, start from the root node and trace the character by reading off 1's and 0's.

Step 8: To decode the characters reverse the steps

Note: Above Mentioned algorithm is applicable for encoding the text document. Huffman coding reduces the alterations and the length of the lengthiest character code along with the lossless compression.

Run Length Coding: It is the simplest data compression technique, in this sequence of same characters or numbers are stored as a single data values and counts rather than the earlier original sequence. This run length is applicable where there is a long sequence of same characters or numbers present in the data. It is a reversible process.

Algorithm:

Step 1: Count the number of same characters/numbers present in the data.

Step 2: To encode the data write the data value i.e. character or number with the count of data value do this for all the characters or numbers present in the data.

Step 3: For decoding, just expand number of the data value for the count of data value.

Note: Comparing to the length, before compression the count of data length is high, after compression the count of data length is less. Run Length Coding reduces the size of data.

Lempel-Ziv Welch coding.

Lempel-Ziv Welch in short LZW compression, it is worked by moving the source data stream into slices that are the shortest sequences not occurred previously.

This compression algorithm works by means of analysing the order of symbols, merging these symbols into strings later changing these strings into codes.

Algorithm:

For Encoding

Step 1: Write a dictionary table and name the columns index and string, check all single characters present in the string and add them to the string column and label the index values

Step 2: Create a table with a column name entry, index and encode. Check all single characters present in the string and add them to the entry column then assign index values to them.

Step 3: For all the remaining strings, compare each individual string with the entry column characters, if that string is not present then add that string to the column. And label the index value.

Step 4: For the encode column check the early occurrence of the string index value and assign that index value to the encode column.

Step 5: If that comparing string is already present in the entry column, then along with that string compare it to the next string and if it is not occurred then add it to the entry column and label the index value, then assign the encode value.

Step 6: Repeat step 2 and 3 until all the strings are compared.

Step 7: For the last single character check the entry in index column and assign that value to the encode column.

Algorithm

For Decoding

Step 1: By considering the earlier dictionary table, draw the table which consists of received, decode, index and entry columns.

Step 2: Add the dictionary values to the index and entry column.

Step 3: Then consider the encoded values, take the first value and write in the received column and in the decode column check the index column for the decode value and if their enter the entry value character in the decode column and leave the index and entry empty for the first occurrence.

Step 4: Then write the next encode value in the received column and check the occurrence of that value in the index, write the entry character of that index.

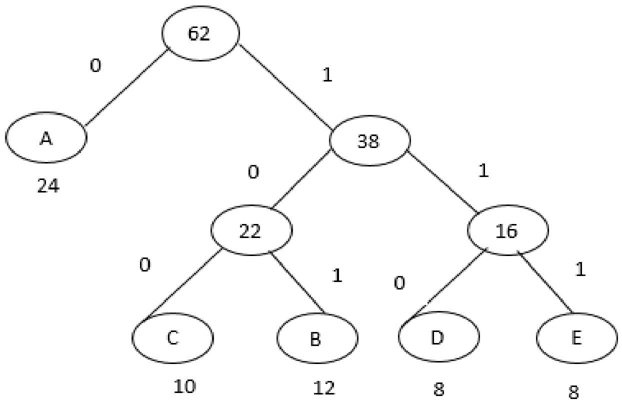
Step 5: Continue the index value, and in the entry column consider the above immediate before decode character and the first character of the current character fill the entry column.

Step 6: Repeat step 4 and 5 until all the encode values are completed.

Step 7: Read the decode column characters from the top to get decoded string.

❖ **Example for Huffman Coding:**
Consider the string: A B C D E

Symbol	Frequency	Code
A	24	0
B	12	101
C	10	100
D	08	110
E	08	111



So the text will be encoded as **0101100110111**

❖ **Example for Run Length Encoding**

If there is a space between strings then considered it as 0

❖ **Example 1: Consider the String: BBBBBBBBBAAAAANGGMMM**

B=09 A=05 N=01 G=02 M=03

Before compression the length of string is 9+5+1+2+3=20

RLE for the above string: B9A5N1G2M3

After compression the length of string is 10

❖ **Example for Lempel-Ziv Welch (LZW) Coding.**

Consider a string **ABABBABCABABBA**

DICTIONARY TABLE

Index	String
1	A
2	B
3	C

ENCODING TABLE

Encoded Output	Index	Entry
....	1	A
....	2	B
....	3	C
1	4	AB
2	5	BA
4	6	ABB
5	7	BAB
2	8	BC
3	9	CA
4	10	ABA
6	11	ABBA
1	.....	.....

Encoded Text: 124523461

DECODING TABLE
FEED THE ENCODED TEXT: 124523461

Received	Decoded	Index	Entry
....	....	1	A
....	....	2	B
....	....	3	C
1	A	....	....
2	B	4	AB
4	AB	5	BA
5	BA	6	ABB
2	B	7	BAB
3	C	8	BC
4	AB	9	CA
6	ABB	10	ABA
1	A	11	ABBA

Decoded Text: ABABBABCABABBA

III. CONCLUSION

In present directions there is lot of issues occurred while compressing the text. And also it very difficult, not secured. This paper is taking action to overcome above mentioned issue. By using Huffman encoding, Run Length encoding and Lempel-Ziv Welch coding algorithms user get good interaction to manage compression problems. Each algorithms having distinct way of compressions, so that this paper conclude much better. Above algorithms will gives security for data by encoding and decoding that enables to user to interaction better.

REFERENCES

[1]. Mrs.Bhumika Gupta, "Study Of Various Lossless Image Compression Technique", IJETTCS, volume 2, issue 4, July-August 2013.

[2]. Harpreet Kaur, Rupinder Kaur, Navdeep Kumar, "Review of Various Techniques for Medical Image Compression", International Journal of Computer Applications, Volume 123, No.4, August 2015.

[3]. Bhonde Nilesh, Shinde Sachin, Nagmode Pradip, D.B. Rane, "Image Compression Using Discrete Wavelet Transform", IJCTEE, Volume 3, March-April 2013.

[4]. Malwinder Kaur, Navdeep Kaur, "A Litreature Survey on Lossless Image Compression", International Journal of Advanced Research in Computer and Communication Engineering, Vol. 4, Issue 3, March 2015.

[5]. A. Alarabeyyat, S. Al-Hashemi, T. Khdour, M. Hjouj Btoush, S. Bani-Ahmed, R. Al-Hashemi, "Lossless Image Compression Technique Using Combination Methods", Journal of Software Engineering and Applications, 2012.

[6]. Richa Goyal, Jasmeen Jaura, "A Review of Various Image Compression Techniques", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 4, Issue 7, July 2014.

[7]. Dalvir Kaur, Kamaljit Kaur, "Huffman Based LZW Lossless Image Compression Using Retinex Algorithm", International Journal of Advanced Research in Computer and Communication Engineering, Vol. 2, Issue 8, August 2013.

[8]. Mridul Kumar Mathur, Seema Loonker, Dr. Dheeraj Saxena, "Lossless Huffman Coding Technique For Image Compression And Reconstruction Using Binary Trees", IJCTA, Vol 3 , Jan-Feb 2012.