International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 4 Issue 03, March-2015

Survey of Text Compression Algorithms

Tanvi Patel
Dept of Information Technology
SVMIT Engineering College
Bharuch, India

Judith Angela
Dept of Information Technology
SVMIT Engineering College
Bharuch, India

Kruti Dangarwala
Associate Professor
Dept of Information Technology
SVMIT Engineering College
Bharuch, India

Poonam Choudhary
Dept of Information Technology
SVMIT Engineering College
Bharuch, India

***Abstract*:- Data compression is now almost a common requirement for every applications as it is a means for saving the channel bandwidth and storage space. Data Compression is an art of allowing a technique to reduce the volume of data i.e. excess information, by maintaining the quality of data. There a number of algorithms available for compression of files of different formats. But, algorithm is to be such a chosen which reduces redundancy of data by consuming less time and providing more compression ratio as compared to other techniques. So, even for a single data type, numbers of approaches are available and to select among them the best one depending upon the need is very important and a difficult task. Compression methods are categorized as Lossy and Lossless but in this paper focus is only on Lossless text compression techniques. The methods which are discussed are Run Length Encoding, Shannon Fanon, Huffman, Arithmetic, LZ77, LZ78 and LZW with its performance.**

Keywords: Data Compression, Lossy, Lossless, Run Length Encoding,Huffman, Shannon Fano, Arithmetic, Lz77,Lz78, LZW.

I. INTRODUCTION

In past years there has been a remarkable blast of transmitting digital data via Internet, correspond to text, images, video, audio, computer programs, etc. [3]. With this tendency to continue, there is a need of developing algorithms that is capable of using network bandwidth effectively [1]. A text contains many words and in turn words contain many characters so to store a text we need to store all words and further to store words needs to store all characters. For this type of storage huge space is needed. So, there is a need of some technique to reduce the size of data so as to occupy less space [2].Data Compression is a technique which reduces the size of the data by removing redundancy and excessive information, for storing the data and to reduce time needed to transfer the data. So, it is a need of all computerized applications to reduce the cost by using the available bandwidth effectively [3]. It is easy to transfer the files on internet if it is more compressed because of which uploading and downloading becomes faster. More the information in file more the cost needed. So, the main goal of compression is to covert the source into digital form with as few bits as possible as the original file while maintaining the fidelity of the original file by less time and less storage space [4][5].

Data Compression has important applications in the area of file storage and distribution system as it requires a way to transmit and store different types of data such as text, audio, video, sound to reduce space and time [6].It is used in multimedia field, text documents, medical image and database table. Depending upon this the algorithm has been divided into two ways. The algorithm which removes some part of data is called lossy data compression and the algorithm which do not loss the data during compression and achieves the same back on decompression that is called lossless data compression [7]. The lossy data compression algorithm is mostly used when compression ratio need is higher than the quality of data after decompression. Lossless data compression is used when quality is the important factor i.e., original data needs to be obtained as such as original source after decompression.

II. TYPES OF COMPRESSION

Data Compression is divided into two types.

a. Lossless Compression

b. Lossy Compression

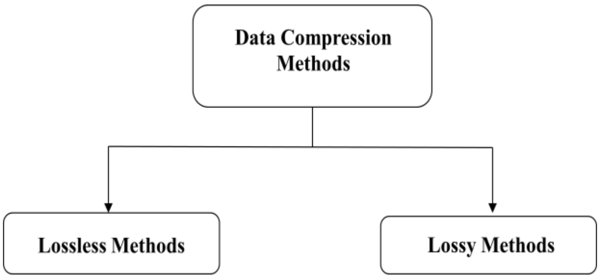

Fig. 1. Types of Data Compression Tehniques

A. *Lossless Compression*

The main aim of lossless compression technique is to compress the file by reducing the information in such a way that there is no loss when decrypting the file back into the original file. Text Compression is considered in Lossless type. One of the popular file format i.e., ZIP file format which is used for compression of data files is an application of lossless data compression.

Lossless compression is used where we need the data the same when decrypting the source file. Lossless data compression most probably exploits statistical redundancy to express data more precisely without any loss in information [8].

Lossless compression can be divided into two categories:

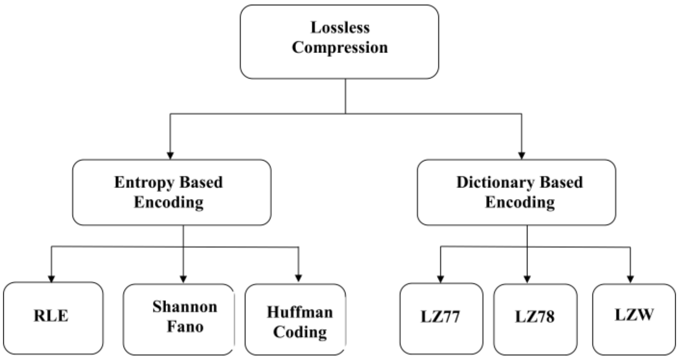

Fig. 2. Types of Lossless Compression

1) *Entropy based encoding*

This technique is not dependent on definite characteristics of medium. In this method the algorithm starts by first counting the frequency of each symbol according to its occurrence in the file. Then the original symbols are replaced with algorithm generated symbol by compression technique. For certain symbols of original file, these newly produced symbols are fixed and are not dependent on the content of file. The new symbols length is variable and it varies on the frequency of certain symbols of original file [9].

2) *Dictionary based encoding*

This algorithm does not encode single symbol but encodes a variable length string into a single token. This technique is also known as Substitution encoding. In this method a data structure is maintained known as dictionary [10]. The encoder finds the match of the string in dictionary from original text and if the match is found it replaces with its reference in the dictionary.

B. *Lossy Compression*

This technique does not produce the same original file i.e., the exact copy but gives output with some information lost. The original message cannot be reconstructed by decoding process, and it is called as irreversible compression [11]. So, this type of technique can't be applied to textual data but can be applied on video, audio, images etc.

III. COMPRESSION ALGORITHM

A. *Run Length Encoding*

RLE is one of the simple Data Compression Algorithm and also called "Run Length Limiting". The main aim of RLE algorithm is to pick out the runs of the source file and to report the symbol and the length of each run [11]. In this encoding technique, one after another the same characters are repeated in a text file.

Among the applications of RLE one of them most popularly known is Fax. For example, the string "XYXYYYYYYZ" is considered as a source to compress, taken the first 3 letters as a non-run is having a length 3, and the next 6 letters taken as a run having length 6, since symbol Y is repeated consequently. So, in this manner Run Length Encoding method compress the file or any type of document but it is not of much use because it cannot compress big files which may not have many repeated words or symbols.

B. *Shannon Fano Coding*

A coding process had been developed to create a binary code tree by Claude E. Shannon and Robert M. Fano in 1960. Shannon Fano coding is one of the easiest method to implement as compared to other methods.

Based on their probabilities it encodes messages. In highly probable character, less number of bits is used and in least occurring character, more number of bits is used [12]. The algorithm is as described below [13]:-

Step1:-
Find the probability/frequency count of the given list of symbol or character.
Step2:-
Sort the symbol/character according to frequency/probability in a descending order.
Step3:-
Divide the list into two parts according to the least difference between the total frequency counts of upper half and lower half.
Step4:-
Assign the value of upper half to be zero and lower half to be 1.
Step5:-
Apply the steps 3 and 4 recursively till the code is obtained for the entire symbol.

Fig. 3. Shannon Fano Algorithm

C. *Huffman Coding*

For text compression, Huffman Coding is most acknowledged method developed by David Huffman in 1950. Sometimes Huffman Coding performs better than the Shannon Fano Coding. In data file, the characters are converted to binary code and most frequent characters and rare characters are allocated by bits same as in Shannon Fano [14]. The algorithm is as follows [13]:-

The nodes are arranged in ascending order.

Step1:-
Two nodes with the lowest frequency is located.
Step2:-
The two least node is added and an internal node of this two node is created and the added sum of the two node is given as it's weight.
Step3:-
The internal node is now added to the list and the two node as it's child.
Step4:-
One of the child node is assigned 1 and other as 0 during coding.
Step5:-
Previous steps are repeated till there is no other node left in the tree. The free node is root of the tree.

Fig. 4. Huffman Algoritm

D. LZ77

Huffman and Arithmetic Coding don't capture the higher order relationship between words and phrase. There is an another technique which is more effective for compressing text known as LZ77 developed by Jacob Ziv and Abraham Lempel in 1977. It use Sliding Window concept [15,16]. Encoding-Pseudo code algorithms is as follows [13]:-

Step1:-
Check look-ahead buffer is empty or not
Step2:-
If not empty, look for the longest match in search buffer.
Step3:-
If match is found print output as (offset from window boundary, length of match, next symbol in lookahead buffer) and shift window by length+1 else print output as (0,0,first symbol in look-ahead buffer) and shift window by 1.
Step4:-
Loop until the look ahead buffer is empty.

Fig. 5. LZ77 Algoritm

E. LZ78

It is dictionary based compression algorithm developed by Jacob Ziv and Abraham Lempel in 1978. In this, encoding and decoding both side has to create a dictionary, it is necessary that both sides dictionary are identical. It maintains an explicit dictionary[17].

The compression algorithm is as follows [13]:-

```
Dictionary = empty; Prefix = empty;
DictionaryIndex = 1;
Loop
while(characterStream is not empty)
{
   Take Char as a next character in characterStream;
   If (Prefix+Char exists in the dictionary)
      Assign Prefix =  Prefix+char;
      Else
      {
        If (Prefix is empty)
        Assign CodeWordForPrefix = 0;
       Else
        Assign CodeWordForPrefix = DictionaryIndex
for Prefix;
        Print Output: (CodeWordForPrefix, Char);
        insertInDictionary((DictionaryIndex,
Prefix+char));
         DictionaryIndex++;
         Prefix = empty;
          }
       }
If(Prefix is not empty)
   Assign CodeWordForPrefix=DictionaryIndex for
Prefix;
  And print Output: (CodeWordForPrefix);
```

Fig. 6. LZ78 Algorithm

F. LZW

LZW is denoted by the name Lempel–Ziv–Welch developed by Abraham Lampel , Jacob Zev and Terry Welch in 1984 and is based on LZ78. In LZW, only the index is send to the dictionary[19]. Based on the presence of substring chosen from the original file, dynamic dictionary is obtained. When a string is matched from the dictionary then the reference of that string is used to encode it and if the match of that string is not found then a new entry is made in the dictionary [18]. LZW algorithm records the string in dictionary. The first 255 entries contains the value of ASCII therefore the actual allocation of index to the string starts from index 256. The Dictionary is built to store all the possible combination of string from the message, starting from two character and so on.

The algorithm is as follows [13]:-

```
Set w=NIL
  loop
      read a character k
      if wk exists in the dictionary
            w=k
else
    Output the code for w add wk to the dictionary
w=k
end loop
```

Fig. 7. LZW Algoritm

IV. MEASURING COMPRESSION PERFOMANCE

A. *Compression Ratio*

Compression ratio is the ratio between the original size of the file and the compressed size of the file it is calculated as [13] :

$$\text{Compression Ratio} = \frac{\text{Original Size}}{\text{Compressed Size}} \qquad (1)$$

B. *Compression Time*

Time taken for compression and decompression must be taken into consideration as in some cases decompression time and in some cases compression time to be considered is necessary and in some cases both of them are necessary.

C. *Entropy*

Entropy is the measurement of the amount of information in your file. This method is used when compression algorithm is based on the statistical information of the original file. The self-information can be calculated by equation [13].

$$I(a1) = \log_2 1/\,p(i) \qquad (2)$$

Or $$I(a1) = -\log_2 p(i) \qquad (3)$$

The Entropy value H of a compression algorithm can be evaluated by the following equation [13].

$$H = \sum_{i=0}^{n} p(i) I(a1) \qquad (4)$$

V.CONCLUSION

In this paper in which situations lossy and lossless compression methods can be used are discussed. Different compression techniques are discussed in detail. Major focus in this paper is made on various data text compression methods like dictionary based and entropy based dictionary. In entropy based technique Run length encoding is not used much as that of Shannon Fano and Huffman. This two methods are much better than RLE. But, both Shannon Fano and Huffman compression is almost same. Huffman is better than Shannon Fano method in a very small difference. In dictionary based method three methods are discussed upon which LZW works best in comparison to LZ77 and LZ78.

REFERENCES

[1] Dr. V.K.Govindan and B.S. Mohan“An Intelligent Text Data Encryption and Compression for High Speed and Secure Data Transmission Over Internet” NIT Calicut, Kerala.

[2] Md. A. Kalam Azad, Rezwana S., Shabbir Ahmed and S. M. Kamruzzaman “An Efficient Technique for Text Compression” 1st International Conference on Information Management and Business (IMB2005).

[3] Introduction to Data Compression, Khalid Sayood, Ed Fox (Editor), March 2000.

[4] R.S. Brar and B. Singh, “A survey on different compression techniques and bit reduction Algorithm for compression of text data” International Journal of Advanced Research In Computer Science and Software Engineering (IJARCSSE) Volume 3, Issue 3, March 2013.

[5] Amandeep Singh Sidhu and Er. Meenakshi Garg “Research Paper on Text Data Compression Algorithm using Hybrid Approach” IJCSMC, Vol. 3, Issue. 12, December 2014.

[6] Elabdalla, A.R. and Irshid, M. I., “An efficient bitwise Huffman coding technique based on source mapping”. Computer and Electrical Engineering 27 (2001) 265 – 272.

[7] Burrows M., and Wheeler, D. J. 1994. “A Block-Sorting Lossless Data Compression Algorithm”. SRC Research Report 124, Digital Systems Research Center.

[8] Shrusti Porwal, Yashi Chaudhary, Jitendra Joshi, Manish Jain “Data Compression Methodologies for Lossless Data and Comparison between Algorithms” International Journal of Engineering Science and Innovative Technology (IJESIT) Volume 2, Issue 2, March 2013.

[9] Arup Kumar Bhattacharjee, Tanumon Bej, Saheb Agarwal “Comparison Study of Lossless Data Compression Algorithms for Text Data” IOSR Journal of Computer Engineering (IOSR-JCE).

[10] Kesheng, W., J. Otoo and S. Arie, 2006. Optimizing bitmap indices with efficient compression, ACM Trans. Database Systems, 31: 1-38.

[11] S.R. Kodituwakku and U.S. AmaraSinghe “Compression of Lossless Data Compression Algorithms for Text Data” Indian Journal of Computer Science and Engineering Vol 1 No 4 416-425.

[12] Fano R.M., “The Transmission of Information”, Technical Report No.65, Research Laboratory of Electronics, M.I.T., Cambridge, Mass.; 1949.

[13] Mark Nelson, Jean-Loup Gailly, “The Data Compression book” 2nd Edition

[14] Huffman D.A., “A method for the construction of minimum redundancy codes”, Proceedings of the Institute of Radio Engineers, 40 (9), pp. 1098–1101, September 1952.

[15] The MPEG-4 Book.

[16] Data Compression Conference (DCC '00), March 28-30, 2000, Snowbird, Utah.

[17] M. Pal Singh and N. Singh, “A Study of Various Standards for Text Compression Techniques”.

[18] Ziv. J and Lempel A., “Compression of Individual Sequences viaVariable-Rate Coding”, IEEE Transactions on Information Theory 24 (5), pp. 530–536, September 1978.

[19] Welch T.A., “A technique for high-performance data compression”, IEEE Computer, 17, pp. 8–19, 1984.