A Review on various Lossless Text Data Compression Techniques

Neha Sharma[1] sharma.neha243@gmail.com

Jasmeet Kaur[2] jasmeet.makkar10@gmail.com

Navmeet Kaur[3] navmeetbrar@gmail.com

Abstract: *Compression algorithms reduce the redundancy in data representation thus increasing effective data density. Data compression is a very useful technique that helps in reducing the size of text data and storing the same amount of data in relatively fewer bits resulting in reducing the data storage space, resource usage or transmission capacity. There are a number of techniques that have been used for text data compression which can be categorized as Lossy and Lossless data compression techniques. In this paper, various lossless data compression techniques have been reviewed that are in use such as Run Length Encoding, Burrows- wheeler transform, Shannon-Fano coding, Huffman coding, Arithmetic coding, Lempel-Ziv Welch and Bit-Reduction algorithm.*

Key-Words: *Text Data Compression, Lossless Compression, Huffman Coding, Arithmetic Coding, Bit reduction*

1. Introduction:

Data Compression is the process of encoding data so that it takes less storage space or less transmission time that it would if it were not compressed. Compression is possible because most of the real world data is very redundant. Data Compression is a technique that reduces the size of data by removing unnecessary information and duplicity. Data compression is the art of reducing the number of bits needed to store or transmit data.

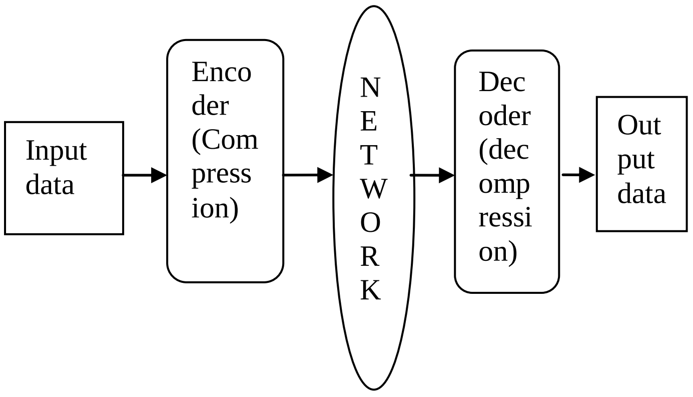

Fig 1: diagrammatic view of Compression

Mainly, two types of data compression techniques are there: Lossy and Lossless data compression.

A lossy data compression method is one where compressing data and then decompressing it retrieves data that may not be exactly same as the original, but is "close enough" to be useful for specific purpose. The examples of frequent use of Lossy data compression are on the Internet and especially in the streaming media and telephony applications. Most of the lossy data compression techniques suffer from generation loss which means decreasing the quality of text because of repeatedly compressing and decompressing the file. Lossy image compression can be used in digital cameras to increase storage capacities with minimal degradation of picture quality.

Lossless data compression is a technique that allows the use of data compression algorithms to compress the data and also allows the exact original data to be reconstructed from the compressed data. This is in contrary to the lossy data compression in which the exact original data cannot be reconstructed from the compressed data. The popular ZIP file format that is being used for the compression of data files is also an application of lossless data compression approach. Lossless compression is used when it is important that the original data and the decompressed data be identical. Lossless text data compression algorithms usually exploit statistical redundancy in such a way so as to represent the sender's data more concisely without any error or any sort of loss of important information contained within the text input data. Since most of the real-world data has statistical redundancy, therefore lossless data compression is possible. For instance, In English text, the letter 'a' is much more common than the letter 'z', and the probability that the letter't' will be followed by the letter 'z' is very small. So this type of redundancy can be removed using lossless compression. Lossless compression methods may be categorized according to the type of data they are designed to compress. Compression algorithms are basically used for the compression of text, images and sound. Most lossless compression programs use two different kinds of algorithms: one which generates a statistical model for the input data and another which maps the input data to bit strings using this model in such a way that frequently encountered data will produce shorter output than improbable(less frequent) data. The advantage of lossless methods over lossy methods is that Lossless compression results in a closer representation of the original input data.

ADVANTAGES OF DATA COMPRESSION

- Less disk space
- Faster writing and reading
- Faster File transfer
- Variable dynamic range
- Byte order independent

DISADVANTAGES OF DATA COMPRESSION

- Effects of errors in transmission
- Added Complication
- Slower for sophisticated methods
- Unknown byte/pixel relationship
- Need to decompress all previous data

2. EXISTING WORK:

R.S. Brar and B.Singh, "A survey on different compression techniques and bit reduction algorithm for compression of text data"
This paper provides a survey of different basic lossless and lossy data compression techniques. On the basis of these techniques a bit reduction algorithm for compression of text data has been proposed by the authors based on number theory system and file differential technique which is a simple compression and decompression technique free from time complexity. This compression algorithm takes O(n) time and n is the total number of characters in the file and the total computation time required for this algorithm is proportional to O(nlogn). Future work can be done on coding of special character which are not specified on key-board to revise better results.

A.D.Suarjaya, "A new algorithm for data compression optimization"
This paper propose a new lossless data compression algorithm called J-bit encoding(JBE) that manipulates each bit of data inside file to minimize the size without losing any data after decoding. This basic algorithm is intended to be combining with other data compression algorithms to optimize the compression ratio. The performance of this algorithm is measured by comparing combination of different data compression algorithms. The main idea of this algorithm is to split the input data into two data where the first data will contain original nonzero byte and the second data will contain bit value explaining position of nonzero and zero bytes. Both of the data can be separately compressed with other data compression algorithm to achieve maximum compression ratio.5 combination algorithms have been tested and compared in this paper. The ability to recognize the hybrid contents (text, audio, video, binary) of a file regardless the file type is potential for future research to achieve better compression ratio.

A.Singh and Y.Bhatnagar, "Enhancement of data compression using Incremental Encoding"
This paper describes the two phase encoding technique which compresses the sorted data more efficiently. This research paper provides a way to enhance the compression technique by merging RLE compression algorithm and incremental compression algorithm. In first phase the data is compressed by applying RLE algorithm that compresses the frequent occur data bits by short bits. In the second phase incremental compression algorithm stores the prefix of previous symbol from the current symbol and replaces with integer value. This technique can reduce the size of sorted data by 50% using two phase encoding technique.

S. Shanmugasundaram and R. Lourdusamy, "A Comparative Study of Text Compression Algorithms"
There are lot of data compression algorithms which are available to compress files of different formats. This paper provides a survey of different basic lossless data compression algorithms. Experimental results and comparisons of the lossless compression algorithms using Statistical compression techniques and Dictionary based compression techniques were performed on text data. Among the statistical coding techniques the algorithms such as Shannon-Fano Coding, Huffman coding, Adaptive Huffman coding, Run Length Encoding and Arithmetic coding are considered. Lempel Ziv scheme which is a dictionary based technique is divided into two families: those derived from LZ77 (LZ77, LZSS, LZH and LZB) and those derived from LZ78 (LZ78, LZW and LZFG). A set of interesting conclusions are derived on their basis. Lossy algorithms achieve better compression effectiveness than lossless

algorithms, but lossy compression is limited to audio, images, and video, where some loss is acceptable. The question of the better technique of the two, "lossless" or "lossy" is pointless as each has its own uses with lossless techniques better in some cases and lossy technique better in others.

S.Porwal, Y.Chaudhary, J.Joshi, M. Jain, "Data Compression Methodologies for Lossless Data and Comparison between Algorithms"
This research paper provides lossless data compression methodologies and compares their performance. Huffman and arithmetic coding are compared according to their performances. In this paper the author has found that arithmetic encoding methodology is powerful as compared to Huffman encoding methodology. By comparing the two techniques the author has concluded that the compression ratio of arithmetic encoding is better and furthermore arithmetic encoding reduces channel bandwidth and transmission time also.

U. Khurana and A.Koul, "Text Compression And Superfast Searching" A new compression technique that uses referencing through two-byte numbers (indices) for the purpose of encoding has been presented. The technique is efficient in providing high compression ratios and faster search through the text. It leaves a good scope for further research for actually incorporating phase 3 of the given algorithm. The same should need extensive study of general sentence formats and scope for maximum compression. Another area of research would be to modify the compression scheme so that searching is even faster. Incorporating indexing so as to achieve the same is yet another challenge.

3. Existing Lossless data compression Techniques:

3.1 Run length Encoding: Run-Length Encoding is a simple data compression algorithm which is supported by bitmap file formats such as BMP. RLE basically compresses the data by reducing the physical size of a repeating string of characters. This repeating string is called a *run* which is typically encoded into two bytes where the first byte represents the total number of characters in the run and is called the *run count* and it replaces runs of two or more of the same character with a number which represents the length of the run which will be followed by the original character and single characters are coded as runs of 1. RLE is useful where redundancy of data is high or ii can also be used in combination with other compression techniques also. Here is an example of RLE:

Input: YYYBBCCCCDEEEEEERRRRRRRRRR

Output: 3Y2B4C1D6E10R

The drawback of RLE algorithm is that it cannot achieve the high compression ratios as compared to another advanced compression methods, but the advantage of RLE is that it is easy to implement and quick to execute thus making it a good alternative for a complex compression algorithm.

3.2 Burrows-Wheeler Transform: Burrows-Wheeler transform does not compress a message, but rather transform it into a form that is more amenable to compression i.e. it can be more efficiently coded by a Run-Length Encoder or other secondary compression technique. The transform rearranges the characters in the input so that that there are lots of clusters with repeated characters, but in a way so that it is still possible to recover the original input. Burrows-wheeler transform (BWT) works in block mode while others mostly work in streaming mode.

The algorithm for a BWT is:

1. The first step is to create a string array.
2. Then generate all the possible rotations of the input string and store each in the array.
3. Sort the array in alphabetical order.
4. Return the last column of the array.

BWT usually works best on long inputs with many alternating identical characters. The working of BWT can be explained by the example given below where & represents the End Of File character.

Fig 2: Example of BWT Algorithm

Because of its alternating identical characters, this input generates an optimal result on performing BWT algorithm and then it could be further compressed using another compression algorithm such as RLE which would results in "3E&3A". But the drawback of BWT is that it does not generate optimal results on most real-world data.

3.3 Shannon-Fano coding: This is one of an earliest technique for data compression that was invented by Claude Shannon and Robert Fano in 1949.In this technique, a binary tree is generated that represent the probabilities of each symbol occurring. The symbols are ordered in a way such that the most frequent symbols appear at the top of the tree and the least likely symbols appear at the bottom.

The algorithm for Shanon- Fano coding is:

1. Parse the input and count the occurrence of each symbol.
2. Determine the probability of occurrence of each symbol using the symbol count.
3. Sort the symbols according to their probability of occurrence, with the most probable first.
4. Then generate leaf nodes for each symbol.
5. Divide the list in two while keeping the probability of the left branch roughly equal to those on the right branch.
6. Prepend 0 to the left node and 1 to the right node codes.
7. Recursively apply steps 5 and 6 to the left and right sub trees until each node is a leaf in the tree.

Generally, Shannon-Fano coding does not guarantee the generation of an optimal code. Shannon – Fano algorithm is more efficient when the probabilities are closer to inverses of powers of 2.

3.4 Huffman Coding: Huffman coding deals with data compression of ASCII characters. Its working is very similar to Shannon-Fano Coding but it follows top down approach means the binary tree is built from the top down to generate an optimal result. In Huffman Coding the characters in a data file are converted to binary code. In Huffman coding, the most common characters in the file have the shortest binary codes, and the characters which are least common have the longest binary code.

Input	Rotations	Alpha-sorted rotations	Output
EAEA EA&	EAEAEA&	AEAEA&E	EEE&AA A
	&EAEAEA	AEA&EAE	
	A&EAEAE	A&EAEAE	
	EA&EAEA	EAEAEA&	
	AEA&EAE	EAEA&EA	
	EAEA&EA	EA&EAEA	
	AEAEA&E	&EAEAEA	

The initial steps of Huffman coding algorithm are similar to Shanon- Fano coding.

The algorithm to generate Huffman code is:

1. Parse the input and count the occurrence of each symbol.
2. Determine the probability of occurrence of each symbol using the symbol count.
3. Sort the symbols according to their probability of occurrence, with the most probable first.
4. Then generate leaf nodes for each symbol and add them to a queue.
5. Take two least frequent characters and then logically group them together to obtain their combined frequency that leads to the construction of a binary tree structure.
6. Repeat step 5 until all elements are reached and there remains only one parent for all nodes which is known as root.
7. Then label the edges from each parent to its left child with the digit 0 and the edge to right child with 1.
8. Tracing down the tree yields to "Huffman codes" in which shortest codes are assigned to the character with greater frequency.

3.5 Arithmetic Coding: Arithmetic coding is an optimal entropy coding technique as it provides best compression ratio and usually achieves better results than Huffman Coding. It is quite complicated as compared to the other coding techniques. When a string is converted in to arithmetic encoding, the characters having maximum probability of occurrence will be stored with fewer bits and the characters that do not occur so frequently will be stored with more bits, resulting in fewer bits used overall. Arithmetic coding converts the entire input data into a single floating point number.

Here is an algorithm to generate the arithmetic code:

1. Calculate the number of unique symbols in the input. This number represents the base b (e.g. base 2 is binary) of the arithmetic code.
2. Assign values from 0 to b to each unique symbol in the order they appear.
3. Using the values from step 2, the symbols are replaced with their codes in the input.
4. Convert the result from step 3 from base b to a sufficiently long fixed-point binary number to preserve precision.
5. Record the length of the input string somewhere in the result as it is needed for decoding.

Here is an example of an encode operation, given the input “XYZTXXYT”:

1. There are 4 unique symbols in input, therefore base = 4. Length = 8
2. Values assigned to symbols: X=0, Y=1, Z=2, T=3
3. Replaced input with values assigned: “0.01230013$_4$”.
4. Convert “0.01231123$_4$” from base 4 to base 2: “0.01101100000111$_2$” i.e. output.

In the above example, the input is 64 bits long assuming 8 bits per character but after performing arithmetic coding the size is reduced to just 15 bits thus resulting in an excellent compression ratio.

3.6 Bit Reduction algorithm: This algorithm was originally implemented for use in a short messaging service application. The main idea behind this program is to reduce the standard 7-bit encoding to some application specific 5-bit encoding system and then pack into a byte array. This method reduces the size of a string considerably when the string is lengthy and the compression ratio is not affected by the content of the string.

Bit Reduction Algorithm in steps-

1. Take a plain text or file which contain the text.

2. Apply Bit Reduction algorithm.

3. Steps to be followed in Bit reduction algorithm-

- Select the frequently occurring characters from the text file which are to be encoded and obtain their corresponding ASCII code.
- Obtain the corresponding binary code of these ASCII key codes for each character.
- Then put these binary numbers into an array of byte (8bit array).
- Remove extra bits from binary no like extra 3 bits from the front.
- Then rearrange these into array of byte and maintain the array.
- Final text will be encoded and as well as compression will be achieved.
- Now decompression will be achieved in reverse order at the client-side.

3.7 Lempel-Ziv-Welch (LZW) Algorithm:

The Lempel-Ziv-Welch algorithm was created in 1984 by Terry Welch. It removes redundant characters in the output and includes every character in the dictionary before starting compression and employs other techniques to improve compression.

The algorithm for Lempel-Ziv-Welch is:

```
w = NIL;
while ( read a character k )
   {
     if wk exists in the dictionary
      w = wk;
     else
       add wk to the dictionary;
       output the code for w;
       w = k;
   }
```

Conclusion:

In this paper, author presents the review on various lossless text data compression techniques. Authors represent literature survey of various latest papers for analysis. Authors conclude that text data can be compressed more than that of by the existing methods and better compression ratio can be achieved in the new proposed method.

References:

[1] 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

[2] A.D. Suarjaya, "A new algorithm for data compression optimization" International Journal of Advanced Computer Science and Applications, Vol. 3, No.8, 2012

[3] A. Singh and Y. Bhatnagar, "Enhancement of data compression using Incremental Encoding" International Journal of Scientific & Engineering Research, Volume 3, Issue 5, May-2012

[4] S. Shanmugasundaram and R. Lourdusamy, "A Comparative Study Of Text Compression Algorithms" International Journal of Wisdom Based Computing, Vol. 1 (3), December 2011

[5] S. Porwal, Y. Chaudhary, J. Joshi, M. 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

[6] D.E.Asuquo, E.O.Nwachukwu and N.J.Ekong, "Application of Run-length Encoding Technique in Data Compression

[7] U .Khurana and A.Koul, "Text Compression And Superfast Searching", Thapar Institute Of Engineering and Technology, Patiala, Punjab, India-147004

[8] V.K. Govindan and B.S. Shajee mohan,"IDBE - An Intelligent Dictionary Based Encoding Algorithm for Text Data Compression for High Speed Data Transmission over Internet "

[9]P.G. Howard and J.C. Vitter, Fellow IEEE," Arithmetic Coding For Data Compression".

[10] H. Altarawneh and M. Altarawneh , "Data Compression Techniques on Text Files: A Comparison Study" International Journal of Computer Applications, Volume 26– No.5, July 2011