International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

A Survey on Different text Data Compression Techniques

Apoorv Vikram Singh[1], Garima Singh[2]

[1]Department of Computer Science and Engineering, Motilal Nehru National Institute of Technology, Allahabad, Uttar Pradesh, India

[2] Department of Computer Science and Engineering, Gautam Buddha University, Greater Noida, Uttar Pradesh, India

Abstract: ***Data Compression refers to the process of reducing the data size and removing the excessive information. The main objective of data compression is to reduce the amount of redundant information in the stored or communicated data. Data compression is quite useful as it helps us to reduce the resources usage such as data storage space or transmission capacity. It finds its application in the area of file storage and distributed system because in distributed system we need to send data from and to all the systems. Data compression techniques are mainly used for speed and performance efficiency along with maintaining the cost of transmission. There are number of different data compression methodologies, which are used to compress different data formats like text, video, audio, image files. Data compression techniques can be broadly classified into two major categories, "lossy" and "lossless" data compression techniques. In this paper, reviews of different basic lossless data compression methods are considered and a conclusion is drawn on the basis of these methods.***

Keywords: Data Compression, Lossless data compression, Lossy data compression, encoding, coding

1. Introduction

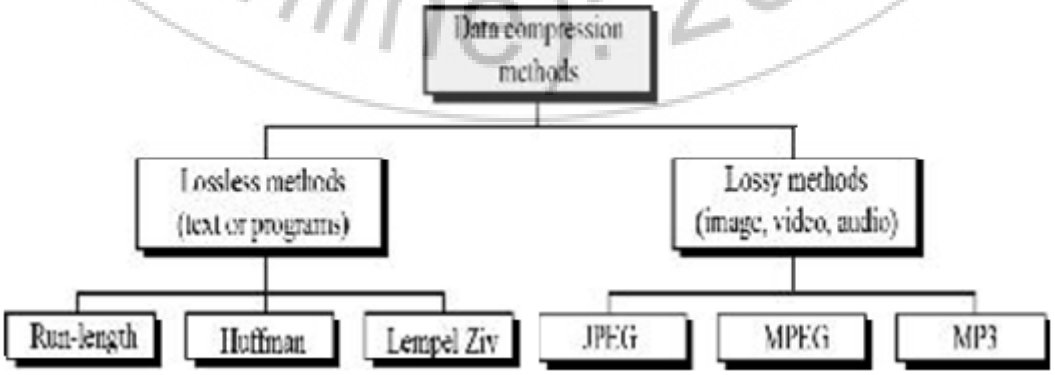
Data compression is a way to reduce the data size, remove excessive information and minimize storage cost by eliminating redundancies that happen in most files. Data compression is a common requirement for most of the computerized application. We find the use of data compression in the area of file storage and distributed systems. It also finds its application in network processing techniques in order to save energy because it reduces the amount of data in order to reduce data transmitted and/or decreases transfer time because the size of the data is reduced. Data compression is used in multimedia field, text documents and database tables as well. The most important criteria of classification is whether the compression algorithm removes some part of data which cannot be recovered during decompression. On the basis of this criterion, the data compression techniques are divided into two major categories, "lossy" data compression techniques and "lossless" data compression techniques.

2. Data Compression Techniques

Lossless: Lossless data compression algorithms are used to reduce the amount of source information to be transmitted in such a way that when compression information is decompressed, there is no loss of information. Lossless compression is possible because most real-world data have statistical redundancy and these algorithms exploit these statistical redundancies to represent data more concisely without losing information.

Lossy: Lossy data compression is contrasted with lossless data compression. Lossy data compression algorithms do not produce an exact copy of the information after decompression as was present before compression. In these schemes, some loss of information is acceptable. Lossy Compression reduce the file size by eliminating some redundant data that won't be recognized by humans after decoding. Applications of Lossy Data Compression techniques:

- Lossy image compression can be used in digital cameras, to increase storage capacities with minimal degradation of picture quality.
- Similarly, DVDs use the Lossy MPEG-2 Video codec for video compression.
- In Lossy audio compression, methods of psychoacoustics are used to remove non-audible (or less audible) components of the signal [3].



Tree representation of compression methods

3. Lossless Data Compression Techniques

There are several types of data compression techniques apart from the three mentioned in the figure. Some of them are:

3.1 Run Length Encoding or

RepetitiveSequence Suppression:

Run-length encoding is a very simple data compression technique whose basic principle is to count the number of consecutive data items and then use that count for compression. The main idea behind this approach is this: If any data item 'd' occurs 'k' times in an input stream, then instead of writing this data item k times we can replace it by 'kd'.

Example-
Input Stream
: AAAAAAABBBBCCCCCCCAAAAAADDDDD

Compressed Stream: 7A4B7C6A5D

RLE is mainly used to compress runs of same data byte. This method is used in the case when there is a lot of repetition of data items. Thus, RLE is often used to compress a bitmap image, especially the low bit one

Input	Rotations	Transformation Sorting All Rows in Alphabetical Order	Taking Last Column	Output Last Column
	^BANANA|	ANANA|^B	ANANA|^B	
	|^BANANA	ANA|^BAN	ANA|^BAN	
	A|^BANAN	A|^BANAN	A|^BANAN	
^BANANA|	NA|^BANA	BANANA|^	BANANA|^	BNN^AA|A
	AN	NANA|^BA	NANA|^BA	
	NANA|^BA	NA|^BANA	NA|^BANA	
	ANANA|^B	^BANANA|	^BANANA|	
	BANANA|^	|^BANANA	|^BANANA	

3.2 Burrows Wheeler Transform or Block Sorting Compression

Burrows Wheeler transform works in Block mode while others mostly work in streaming mode[2]. This algorithm is classified as transformation algorithm because it rearranges a character string into runs of similar characters. Now, these strings of similar characters can be used as an input stream for other algorithms like run-length encoding or move-to-front transform for achieving better compression ratios.

Example-
InputStream: AAAAAAABBBBCCCCCCCAAAAAADDDDD

Output Stream: 7A4B7C6A5D

The transform is done by sorting all rotations of the text in lexicographic order, then taking the last column.[6]. Since the BWT operates on data in memory, you may encounter files too big to process in one fell swoop. In these cases, the file must be split up and processed a block at a time [5]. One of the important feature of BWT is that, thistransformation is reversible because when a character string is transformed by BWT, the value of the character does not changes, it only permutes the order of characters.

3.3 Move-to-front Transform

Move-to-front Transform is another basic techniquefor data compression but the irony is that, it does not compress data, rather it helps to reduce redundancy sometimes. The main idea is that each symbol in the data is replaced by its index in the "stack of recently used symbols" thus, providing the symbol a smaller output number.

Example: Input Stream: banana

Output Stream: 1,1,13,1,1,1,0,0

Iteration	Sequence	List
Bananaaa	1	(abcdefghijklmnopqrstuvwxyz)
Bananaaa	1,1	(bacdefghijklmnopqrstuvwxyz)
Bananaaa	1,1,13	(abcdefghijklmnopqrstuvwxyz)
Bananaaa	1,1,13,1	(nabcdefghijklmopqrstuvwxyz)
Bananaaa	1,1,13,1,1	(anbcdefghijklmopqrstuvwxyz)
Bananaaa	1,1,13,1,1,1	(nabcdefghijklmopqrstuvwxyz)
Bananaaa	1,1,13,1,1,1,0	(anbcdefghijklmopqrstuvwxyz)
Bananaaa	1,1,13,1,1,1,0,	(anbcdefghijklmopqrstuvwxyz)

This is how input streams are transformed into output stream using Move-to-front Transformation. This technique is intended to be used as optimization for other algorithm likes Burrows-wheeler transform.**3.4 LZW (Lempel-Ziv Welch) compression**

LZWis one of the most popular method of data compression. The main steps for this technique are given below:-

- Firstly it will read the file and given a code to each character.
- If the same characters are found in a file then it will not assign the new code and then use the existing code from a dictionary.
- The process is continuous until the characters in a file are null.

3.5 Shannon Fano Coding

Shannon Fano Coding technique is used to encode data or messages depending upon their probability of occurrence. This technique involves following steps:

- For a given list of symbols, develop a probability table.
- Sorting the table according to the probability and placing the most probable element at the top of the list.
- The table is then divided into two parts, such that the sum of probabilities both the parts are as close as possible.
- The left half of the list is assigned '0' and the right half is

assigned '1'.

- Repeat the steps 3 and 4 for each of the two halves then further divide the groups and adding bits to the codes and stop the process when each symbol has a corresponding leaf on the tree.
-

Example:

Symbol	A	B	C	D	E
Frequency	15	7	6	6	5
Probabilities	0.3846	0.1795	0.1538	0.1538	0.1282

After going through all the steps mentioned above we get:

Symbol	A	B	C	D	E
Code	00	01	10	110	111

On calculating the average number of bits, we get it to be around 2.28 bits.

3.6 Huffman Coding

A Huffman Coding is more sophisticated and efficient lossless data compression technique. In Huffman Coding the characters in a data file are converted to binary code. And in this technique the most common characters in the file have the shortest binary codes, and the least common have the longest binary code [7].

1. Initialization: Put the elements in a list sorted according to their frequency counts.

2. Repeat the following steps until the sorted list has only one node left:

a) From the list pick two elements with the lowest frequency counts. Form a Huffman sub tree that has these two nodes as child nodes and create a parent node.

b) Assign the sum of the children's frequency to the parent node and now considering the parent node as one of the nodes of the list, again pick the lowest two frequency counts and form a Huffman sub tree.

3. In third step we do labelling the edges from each parent to its left child with the digit 0 and the edge to right child with 1. The code word for each source letter is the sequence of labels along the path from root to leaf node representing the letter.

Example: Using the same frequency as Shannon Fano above:

Symbol	A	B	C	D	E
Frequency	15	7	6	6	5
Probabilities	0.3846	0.1795	0.1538	0.1538	0.1282

After going through all the steps mentioned above, we get:

Symbol	A	B	C	D	E
Code	0	100	101	110	111

On calculating the average number of bits, we get it to be around 2.23 bits.

3.7 Arithmetic Coding Technique

Arithmetic coding is the most powerful coding technique. This method is different from other compression techniques as it does not replace each bit with a codeword as other methods instead it replaces a stream of input data with a floating number as output.

a) In the first step, we calculate the frequency count of different symbols.

b) In second step we encode the string by dividing up the interval [0, 1] and allocate each letter an interval whose size depends on how often it comes in the string.

c) In third step we consider the next letter, so now we subdivide the interval of that letter in the same way. We carry on through the message….And, continuing in this way, we obtain the required interval.

A message is represented by a half-open interval [*a*, *b*) where *a* and *b* are real numbers between 0 and 1. Initially, the interval is [0, 1). When the message becomes longer, the length of the interval shorts and the number of bits needed to represent the interval increases.[4]

4. Measuring Compression Performances

Performance measure is use to find which technique is good according to some criteria. The performance of the compression algorithm can be measured on the basis of different criteria depending upon the nature of the application. The most important thing we should keep in mind while measuring performance is **space efficiency**. Time efficiency is also an important factor. Since the compression behavior depends on the redundancy of symbols in the source file, it is difficult to measure performance of compression algorithm in general. The performance of data compression depends on the type of data and structure of input source. The compression behavior depends on the category of the compression algorithm: lossy or lossless [1]. Following are some measurements use to calculate the performances of lossless algorithms.

- ***Compression Ratio*:** Compression Ratio is the ratio between the size of the file after compression and the size of the file before compression.
- **Compression Ratio** = Size after compression/size before compression
- ***Compression Factor*:** Compression Factor is the inverse of compression ratio. It is the ratio between the size of the file before compression and size of the file after compression.
- **Compression Factor** = Size before compression/size after compression

5. Conclusion

In this paper, we have talked about the need of data compressions and the situations in which lossy and lossless data compressions are useful. Several algorithms used for lossless compression are described in brief and various conclusions are drawn. Compression techniques like BWT(Burrows Wheeler Transform) and MFT(Move-to-front Transform) are the algorithms which does not compress data, they just transform the input stream and these transformed input stream act as input for better compression techniques. Run Length Encoding is a good compression technique but it is effective only in the case when there is a consecutive repetition of symbols or data. Thus, when such repetitions are not present, then this compression does not

work effectively. Huffman coding is a better compression technique than Shannon Fano coding but Arithmetic Coding is the most effective compression technique among all the above mentioned compression techniques. Compression speed of Huffman and Shannon Fano coding is faster than Arithmetic Coding but the compression ratio of Arithmetic Coding is far better than the other two. And furthermore arithmetic encoding reduces channel bandwidth and transmission time.

Compression Ratio of any technique can further be improved by applying two techniques on the same data or message. For instance, we can firstly apply BWT on any data and then we can apply any of the compression techniques like RLE or Huffman Coding. This type of combination improves the compression ratio. Future work can be done on implementing the compression schemes so that the searching and compression is faster.

References

[1] "Data Compression Methodologies for Lossless Data and Comparison between Algorithms", International Journal of Engineering Science and Innovative Technology, Vol 2, March 2013.
[2] I Made Agus Dwi Suarjaya, "A New Algorithm for Data Compression Optimization", International Journal of Advanced Computer Science and Applications, VOl 3, 2012.
[3] A Survey on Different Compression Techniques and Bit Reduction Algorithm for Compression of Text/Lossless Data", International Journal of Advanced Research in Computer Science and Software Engineering, Vol 3, March 2013
[4] "A Survey on the different text data compression techniques", International Journal of Advanced Research in Computer Science and Technology, Vol 2,Feb 2013.
[5] M. 1996. Data compression with Burrows-Wheeler Transform. Dr. Dobb's Journal.
[6] Ken Huffman. Profile: David A. Huffman, Scientific American, September 1991, pp. 54–58.
[7] Mark Daniel Ward, "Exploring Data Compression via Binary Trees1," *International Journal of Advanced Computer Science and Applications (IJACSA)*, Vol. 3, No.8, 2012.
[8] Blelloch, E., 2002. Introduction to Data Compression, Computer Science Department, Carnegie Mellon University.

Author Profile

Apoorv Vikram Singh is currently enrolled in 4th year of his B.Tech programme (2011-2015) from Motilal Nehru National Institute of Technology(MNNIT). He is an ace programmer and has developed many android applications. In 2013, he received the title of "Mr. Avishkar" in the Technical Festival organised by his college. Besides programming, he has interests in playing football and listening to music.

Garima Singh is presently enrolled in the 4thyearof her Integrated M.Tech programme (2011-2016) in Computer Science Engineering from Gautam Buddha University. She has already written papers in 3rd year but this one is her first to be published. She has interests in web development and has developed a website for her college fest. Besides this she loves to read novels, painting and listening to music.