INTERNATIONAL JOURNAL OF INNOVATIVE TECHNOLOGY AND CREATIVE ENGINEERING (ISSN:2045-8711)
VOL.5 NO.8 AUGUST 2015

A COMPARATIVE ANALYSIS ON DIFFERENT TECHNIQUES IN TEXT COMPRESSION

Dr.S.Pannirselvam
Research Supervisor & Head
Department of Computer Science, Erode Arts & Science College (Autonomous), Erode, Tamil Nadu, India.
Email: pannirselvam08@gmail.com
D.Selvanayagi
Ph.D. Research Scholar, Department of Computer Science,
Erode Arts & Science College (Autonomous), Erode, Tamil Nadu, India.
Email: selvasubhika@gmail.com

Abstract— Digital image processing the term image refers to a digital image and its manipulation by means of processor. Data compression is the computing process to face the problems of the constraints in memory. Data compression is one of the important procedures to reduce the space occupied by a file, which normally leads to reduction of time taken to access the text file. The technique contains two types, they are lossy and lossless compression technique. This research work deals with lossless compression techniques. In this work describes a comparison of various techniques such as Run Length Encoding, Huffman Coding, Lempel Ziv Welch and Arithmetic Coding. The parameters that are considered to analyze text file are Compression Time, Compression Speed and Compression Ratio.

Keywords—Data Compression, Lossless compression, RLE, Huffman, LZW, Arithmetic.

1. INTRODUCTION

Digital image processing is a subset of the electronic domain where the image is converted to an array of small integers called pixels, representing a physical quantity such as scene radiance, stored in a digital memory and processed by computer or other digital hardware. A digital image compression is the method of image data rate reduction to save storage space and reduce transmission rate requirements. Compression is a system for reducing the quantity of facts needed to storage or transmission of information given like text, images, video, sound.Data compression algorithms[10] are classified into lossy and lossless facts compression algorithm. One of the lossy compressions [11], techniques are used to compress image, records documents for conversation or archives purposes.

Lossy compression [1] methods are categorized according to the type of data they are designed to compress. The methods, which may handle all binary input, are used to any form of data. However, most are unable to complete significant compression on data deal with a selected type. Lossless compression [2] methods to be categorized according their type of data are modeled to Compress. The algorithm that can handle all binary input can be used on any type of data. However, most are unable to achieve significant compression on data that is deal with a particular type.

2. LITERATURE REVIEW

Apoor et al [2] presented the comparative analysis between Huffman coding and Arithmetic coding. As arithmetic accommodates adaptive models easily and provide separation between model and coding. In arithmetic coding there is no need to translate each symbol into an integral number of bits, but it involves the large computation on the data like multiplication and division. The disadvantage of arithmetic coding is that it runs slowly, complicated to implement and it does not produce prefix code. Arithmetic Compression [2] is more suitable for small text when compared with Huffman compression and for large text Huffman compression is suitable.

Tanvi et al [4] analyzed lossless data compression techniques.Major focused 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 twomethods are much better than RLE. But, both Shannon Fanoand Huffman compression is almost same. Huffman is betterthan Shannon Fano method in a very small difference. Indictionary based method three methods are discussed uponwhich LZW works best in comparison to LZ77 and LZ78.

Kashfiaet al [12] suggested that after computing and comparing the compression ratio, average code length and standard deviation for Shannon Fano Coding, Huffman Coding, Repeated Huffman Coding, Run-Length Coding and Modified Run-Length Coding, an idea is generated about how much compression can be obtained by each technique. So, now the most effective algorithm can be used based on the input text file size, content type, available memory and execution time to get the best result. Future works can be carried on an efficient and optimal coding technique using mixture of two or more coding techniques for image file, exe file etc. to improve compression ratio and reduce average code length.

Altarawneh et al[13] presented various methods of data compression such as LZW, Huffman coding are using in text files. The authors have evaluated and test the algorithms on various sizes of text files and compared their performance on various parameters such as compression size, compression ratio, compression time.

Khurana et al[7] developed a new compression technique that uses referencing through two byte numbers for the purpose of encoding has been presented. The technique is efficient in providing high compression ratio and faster search through the text. It should need extensive study of general sentence formats and scope for maximum compression. Another area of research would be two modify the compression scheme show that searching is even faster.

Singh at al[16] described the two phases encoding technique which compresses the shorted data more efficiently. The author provides a new way to enhance the compression techniques by merging RLE and incremental compression algorithms. In first phase the data is compressed by applying RLE algorithm, that compresses the frequent occur data bits by short bits. In second phase incremental compression algorithm stores the prefix of previous symbol from the current symbol and replaces with integer value. RLE technique can reduce the size of sorted data by 50% using two phases encoding techniques.

Mann et al[14] discussed and compared selected set of lossless data compression algorithms such as RLE, Huffman and Arithmetic coding. The author compares the performance of these algorithms on the basics of various parameters such as Compression ratio, Compression speed. The author has concluded that the compression speed of Huffman is better than the Arithmetic coding.

Even though lot of issues are available in the lossless Text compression techniques. So, there is need to develop a new efficient model for the Text Compression.

3. Methodology

3.1 Run Length Encoding

Data contains sequences of equal bytes. By changing the repeated byte sequences with the quantity of occurrences, a substantial reduction of data can be achieved and is called as Run Length Encoding. Run Length Encoding [3] is one of the simple data compression algorithm. 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. In this encoding technique, one after another the same characters are repeated in text file.

Algorithm

Step 1: Read the character from the input string.
Step 2: To store the repeated character.
Step 3: If the character is not repeated, write out the current Characters.
Step 4: The current character is does not match the previous character, set the previous character as current character and repeat the step 1.
Step 5: Those characters are encoded as 2 bytes.
Step 6: If the character is null to exit.

For example, the string ”ABABBBBBBC” 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 B 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.

3.2 Huffman Coding

Huffman coding [4] deals with data compression of ASCII characters. It follows top down approach the binary tree is built from the top down to generate achieve better result. In Huffman coding, the characters in a data file are converted to binary code. The maximum usual characters in the file have the shortest binary codes and which are least commonplace have the longest binary code. A Huffman code can be determined by means of successively constructing a binary tree, whereby the leaves represent the characters, which can be to be encoded. Each node carries the relative opportunity of prevalence of the characters belonging to the sub tree lower than the node. The edges are categorized with the bits 0 & 1.

Algorithm

Step 1: Parse the input and count the occurrence of each symbol
Step 2: Determine the probability of occurrence of each symbol using the symbol count.
Step 3: Sort the symbol according to their probability of occurrence with most probable first.
Step 4: Generate leaf nodes for each symbol and add them to a queue.
Step 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.

Then label the edges from each parent to its left child with the digit 0 and the edge to right child with one. Tracing down the tree yields to Huffman codes in which shortest codes are assigned to the character with greater frequency.

3.3 LZW

LZW would only send the index to the dictionary. Dictionary based[17] compression algorithms are based on a dictionary instead of a statistical model. A dictionary is a set of possible words of a language, and is stored in a table like structure and used the indexes of entries to represent larger and repeating dictionary words. The Lempel-Zev Welch algorithm or simply LZW algorithm[5] is one of such algorithms. In this method, a dictionary is used to store and index the previously seen string patterns. In the compression process, those index values are used instead of repeating string patterns. The dictionary is created dynamically in the compression process and no need to transfer it with the encoded message for decompressing. In the decompression process, the same dictionary is created dynamically. Therefore, this algorithm is an adaptive compression algorithm.

LZW is a general compression algorithm capable of working on almost any type of data. LZW compression creates a table of strings commonly occurring in the data being compressed, and replaces the actual data with references into the table. The table is formed during compression at the same time at which the data is encoded and during decompression at the same time as the data is decoded. The algorithm is surprisingly simple. LZW compression replaces strings of characters with single codes. It does not do any analysis of the incoming text. Instead, it just adds every new string of characters it sees to a table of strings. Compression occurs when a single code is output instead of a string of characters. It starts with a "dictionary" of all the single character with indexes 0...255. It then starts to expand the dictionary as information is sent through. Soon, redundant strings will be coded as a single bit, and compression has occurred. This means codes 0-255 refer to individual bytes, while codes 256-4095 refers to substrings.

Algorithm

Step 1: Read the input of first byte and stored as string
Step 2: To check whether the input string is available or not in dictionary.
Step 3: If the string is not available in the dictionary to add the string into dictionary
Step 4: If it is available, add character to the string until the last characters.
Step 5: To read the next character and Output the string.
Step 6: Repeat the Process until the character is null.

3.4 Arithmetic coding

The basic idea of arithmetic is to assign short code word to more probable events and longer code word to less probable events. Arithmetic coding[6] provides an effective mechanism for removing redundancy in the encoding of data. In arithmetic coding, an interval is assigned to each symbol. Starting with the interval [0,1] for each interval is divided in several sub intervals, which sizes are proportional to the current probability of the corresponding symbols of the alphabet. The subinterval from the coded symbol is then taken as the interval for the next symbol. The output is the interval of the last symbols.

Algorithm

Step 1: Read the input text file
Step 2: Calculate the number of unique symbols from input text file.
Step 3: Assign the values from each unique symbol in the order they appear.
Step 4: The symbols are replaced with the codes in the input.
Step 5: Convert the code to long fixed point binary number to preserving precision and store the length of the input string to generate decoding.

Arithmetic coding is change the method of replacing each bit with a code word. So it replaces a string of input data with a single floating point number as a output.

4. EXPERIMENTATION & RESULTS

In this work mainly focused on performance of various algorithms in with inputs. To evaluating the compression algorithms are based on Compression Time, Compression Ratio and Compression Speed.

a) Data Compression Ratio (CR) = Uncompressed Size/compressed Size
b) Compression time depends on size of the file to be compressed on particular time.
c) Compression Speed = Uncompressed Bits/Seconds to Compress.

Table 1: Run Length Encoding

Original Input File Size(kb)	Run length encoding		
	Compression Time (M/s)	Compression Ratio	Compression Speed
117	265	82.765	0.622
132	257	84.647	0.655
176	245	85.824	0.778
352	180	88.549	0.809

From the Table 1 show the Compression Time, Compression Speed, Compression ratio of the different sizes of input files applying Run Length Encoding.

Table 2: Huffman coding

Original Input file Size(Kb)	Huffman coding		
	Compression Time (M/s)	Compression Ratio	Compression Speed
117	250	86.291	0.772
132	231	88.725	0.783
176	223	89.334	0.818
352	136	92.908	0.876

From the Table 2 shows the Compression Time, Compression Speed, Compression Ratio of the different size of input files implemented on Huffman coding.

Table 3: LZW

Original Input file Size(Kb)	LZW		
	Compression Time (M/s)	Compression Ratio	Compression Speed
117	258	84.231	0.739
132	253	87.822	0.754
176	240	88.397	0.802
352	177	90.996	0.824

From the Table 3 shows the Compression Time Compression Speed, Compression ratio of the different input files applying LZW.

Table 4: Arithmetic Encoding

Original Input file Size(Kb)	Arithmetic Encoding		
	Compression Time (M/s)	Compression Ratio	Compressi on Speed
117	270	83.988	0.735
132	259	86.443	0.754
176	258	87.258	0.798
352	185	89.479	0.816

From the Table 4 shows the Compression Time, Compression Speed, Compression ratio of the different input files using Arithmetic Encoding.

Table 5: Overall Performance of Compression Ratio

Input File	117	352	176	352
Run Length Coding	82.765	83.988	84.231	86.291
Arithmetic Coding	84.647	86.443	87.822	88.725
LZW	85.824	87.258	88.397	89.334
Huffman Coding	88.549	89.479	90.996	92.908

From the table 5 shows the Compression Ratio of the different input files.

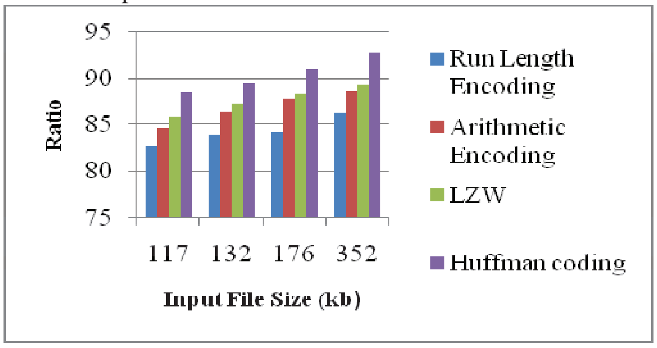

Fig.1 Overall Compression Ratio

From the Fig 1 shows the Compression Ratio of the different size of input files. In that Huffman Coding Ratio is better while compare to other techniques.

5. CONCLUSION

Four different types of lossless compression techniques are focused in this work. Text data are compressed and decompressed by using these different techniques. From the overall performance we observed that Huffman technique is better for compression ratio while compare to other techniques. Huffman technique gives better results for compression time and itis best suited for the compression speed. Even though, in future there will be need to develop a lossless text compression algorithm that can compress the text data in the efficient way that can also be used in various real time applications where compression on text data is required.

6. REFERENCES

[1] Sashikala, Melwin.Y., Arunodhayansam Solomon,, M.N.Nachappa, "A survey of compression techniques",International journal of recent technology and Engineering, vol-2,issue-1,March 2013.

[2] Apoorv Vikram Singh and Garima Singh, "A survey on Different text Data Compression Techniques", International Journal of Science and Research,Vol 3,July 2014.

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

[4] Tanvi Patel and Judith Angela, kurtiDangarwala, " Survey Of Text Compression Algorithms", International Journal Of Engineering Research and Technology, issue 3, march 2015.

[5] S.Senthil and L.Robert "Text compression Algorithms A Comparative Study" , journal of communication technology, vol 1, December 2011.

[6] Ming-Bo Lin Jang-Feng Lee; Gene Eu Jan, "A Lossless Data Compression and Decompression Algorithm and its Hardware Architecture", Very Large Scale Integration Systems, IEEE Transactions on Vol 14, Issue:9, Pages: 925-936, October 2006.

[7] U.Khurana and A.Koul, "Text compression and superfast searching", Thapar institute of engineering technology, Patiala, Punjab, India.

[8] Ming-Bo Lin Jang-Feng Lee; Gene Eu Jan, "A Lossless Data Compression and Decompression Algorithm and its Hardware Architecture", Very Large Scale Integration Systems, IEEE Transactions on Vol 14, Issue:9, Pages: 925-936, October 2006.

[9] S.Porwal, Y.Chaudhary, J.Joshi and .Jain, "Data Compression Methodologies for Lossless Data and Comparison between Algorithms", International Journal of Engineering Science and Innovative Technology, Vol 2, March 2013.

[10] R.Kaur and M.Goyal, "An Algorithm for Lossless Text Data Compression", International Journal of Engineering Research & Technology, Vol 2, July 2013.

[11] AmarjitKaur, Navdeep Singh Sethi, Harinderpal Singh, "A Review on Data Compression Techniques", International Journal of Advanced Research in Computer Science and Software Engineering, Vol 5, January 2015.

[12] KashfiaSailunaz, Mohammed RokibulAlamKotwal, Mohammad Nurul Huda, "Data Compression Considering Text Files", International Journal of Computer Applications (0975 – 8887) Volume 90 – No 11, March 2014.

[13] H.Altarawneh and M.Altarawneh, "Data compression Techniques on text files: A Comparative study" International Journal of Computer Applications, Volume 26 No.5, July 2011.

[14] A.J Mann, "Analysis and comparison of Algorithms for Lossless data Compression" International Journal Of Information and Computation Technology, ISSN:0974-2239.pp. 139-146.

[15] Nathanael Jacob, Priyanka Somvanshi, Rupali Tornekar "Comparative Analysis of Lossless Text Compression Techniques" International Journal of Computer Applications Volume 56- No.3, October 2012.

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

[17] Amitjain, kamaljit, I.Lakhtaria "Comparative Study of Dictionary Based Compression Algorithms on Text Data", International Journal of Computer Engineering and Applications, Volume 5, Issue 2, May14.