International Journal of Research in Engineering and Science (IJRES)
ISSN (Online): 2320-9364, ISSN (Print): 2320-9356
www.ijres.org Volume 9 Issue 7 ǁ 2021 ǁ PP. 38-44

Comparative Analysis of Lossless Text Compression Methods with Novel Tamil Compression Technique

[1]B.Vijayalakshmi, [2]Dr.N.Sasirekha
[1]*Ph.D.Research Scholar, Department of Computer Science, Vidyasagar College of Arts and Science, Udumalpet, Tamilnadu, India*
[2] *Associate Professor, Department of Computer Science, Vidyasagar College of Arts and Science, Udumalpet, Tamilnadu, India*

Abstract
Data compression is a technique of representing the data in a condensed way. Data compression plays a vital role for both text and multimedia content. Many algorithms are available to compress the data which produce different file formats after compression. This paper provides an overview of novel lossless Tamil compression technique for Tamil text file and the experimental results extraction. These results are compared with statistical compression methods, dictionary based compression methods and some of the existing compression utility software. The efficiency of compression techniques are measured in terms of space required to store the compressed file and the time taken to compress and decompress the file. The algorithms such as Run length encoding, Huffman coding, Shannon Fano and Arithmetic coding in statistical compression are considered. In dictionary based compression technique LZ77, LZ78 and LZW are compared. The compression utility software like deflate, zip and gzip are also compared. The parameters like compression ratio, compression factor, percentage of compression and time taken to compress and decompress a file are used as a measure for efficiency of algorithms. The average result was extracted and compared with novel Tamil Compression Technique (TCT).

Keywords: *Static coding, dictionary encoding, text compression, decompression, Compression parameters.*

Date of Submission: 10-07-2021 Date of acceptance: 26-07-2021

I. INTRODUCTION

Data compression can be performed on text files, images, audio and video files. It is defined as minimizing the space required to store the data results in increase of speed to transmit the data. Text compression deals with representing the digital form of data with few bits than the original data bits. The decompression process involves reconstruction of original data with minimum requirement. Various compression algorithms are available that gives the result with different compression ratios [1].

Two classification of data compression are lossless and lossy compression. In lossless data compression the exact reconstruction of original data is built from the compressed form. Usually the text files follow lossless compression technique. The lossy compression algorithm results in loss of some part of data from the original. The decompression is performed by approximation method to compensate the lossy data. Almost all the multimedia files like images, audio and video are compressed using lossy compression techniques. The lossless text compression technique algorithm falls into either statistical coding method or dictionary based compression method. In this paper the popular statistical coding compression methods like Run length encoding, Huffman coding, Shannon Fano and Arithmetic coding are explained in section 3. Dictionary based compression techniques such as LZ77, LZ78 and LZW are presented in section 4. Compression utility software like deflate, zip and gzip are discussed in section 5. Section 6 briefly explains about a novel compression technique for Tamil document represented as Tamil Compression Technique (TCT). The parameters to measure the efficiency of compression technique are given in section 7. Section 8 detailed about the outcome of experimental results and analysis of statistical coding algorithms, dictionary based algorithms, compression utility software and Tamil compression technique. The parameters extracted to measure the efficiency of algorithms are compression factor, ratio, percentage, compression time and decompression time. The comparison of results was displayed in the form of tables and graphs.

II. RELATED WORKS

Arup Kumar, Tanumon and Saheb, authors of paper [4] represented that Lossless compression was broadly categorized into two types as entropy based encoding and dictionary based encoding. In paper [13] the author explained about the Huffman coding algorithm which helps to decrease the data length by replacing the

small variable length code word in the place of frequently occurring word. Author Shannon and Fano of paper [10] and [33] explained that Shannon Fano algorithm is used to compress the data by encoding the text data with the help of probabilities of occurrence of symbols. Pasco, Rissanen J and Langdon G.G. of paper [20], [24] and [25] developed an algorithm that generates single floating point value to replace the given input stream of text data. Ziv. J and Lempel A authors of paper [39] and [40] introduced LZ77 dictionary based compression by encoding the input text file of repeated symbols by a pointer. In paper [40] the working principle of LZ78 was explained. A dictionary has to maintain sequence of index with a pair of values to encode and decode the algorithm of compression. Welch T.A. of paper [38] presented the LZW dictionary based algorithm based on LZ78 with the initial dictionary has sequence of all symbols from the given input data. The input was encoded by the index from the dictionary. In Paper [34] the author Stay and Michael explains that ZIP is a compressor program developed by Phil Katz. It also acts as an archival compression program. The given input text file is converted to compressed file of ZIP file format. Carus, A., and A. Mesut author of paper [8] describes that GZIP uses deflate algorithm implemented in UNIX operating system. It compresses the text file given as input and produces the output as compressed file. Based on the nature of language, syllable, and phrase formation a specific compression technique was developed for different languages. Paper [26][27] and [31] a special compression method was developed for the language Malayalam, Gujarati and Czech.

Various lossless text compression algorithms were compared earlier with different algorithms. Paper [19], [21] and [29] has taken the parameters like compression ratio, compression factor, percentage of compression, time for compression and decompression as performance measure of comparison. Paper [18] compared various dictionary based compression algorithms and concluded that LZW as an efficiently performed among the other dictionary based compression algorithms.

III. STATISTICAL COMPRESSION TECHNIQUE

3.1 RUN LENGTH ENCODING:

The simplest method of data compression algorithm is Run length encoding. This algorithm takes a stream of characters of string as input and identifies the runs and non-runs. A run is consecutive sequence of characters and all the remaining are considered as non-runs. The runs are replaced by the symbol and length of the run. This runs are encoded in original file and results in compressed file. The decompression process replaces the length by repeated sequence of character.

Example:

Original data: aaabbaaaaacccc
Compressed data: 3a2b5a4c

3.2 HUFFMAN CODING:

David Huffman developed this algorithm in 1950. The Huffman coding algorithm reduces the length of data by finding the frequency of characters in the input and assigns the binary code for each character [13] [28]. This code is replaced in the original file to form a compressed file. Figure 1 shows the steps in Huffman coding algorithm. The binary numbers of a particular character act as encode and it was used to replace the characters in the original file to form a compressed file [5][15].

BEGIN: Huffman Coding

1. Read the input with streams of strings of characters or symbols.
2. Find the frequency of each character occurred using character count.
3. Sort the characters from highest to lowest count.
4. Built a Huffman binary tree and add the symbols in the leaf node.
5. Encode the character by binary numbers with left child as 0 and right child as 1.

END

Fig. 1: Huffman Coding Algorithm

3.3 SHANNON FANO CODING:

Claude Shannon and R.M. Fano designed and developed the Shannon Fano coding algorithm. It is the variation of static Huffman coding algorithm [17]. The variation is applied in the creation of code for symbol or character. In Shannon Fano algorithm the probability of symbol in the input are calculated similar to Huffman coding algorithm. Here a table is formed by placing the character based on the occurrences from the most frequent to the least frequent. This table is divided into two halves by placing as close as possible the probability of character count in the upper half and total frequency count in the bottom half. Assign '0' for upper half and '1' to the lower half. These binary digits act as an encoder to the characters in the input file. The compressed file is created by replacing the character by binary code from the table.

3.4 ARITHMETIC CODING:

The arithmetic encoding algorithm concept was developed by Elias and enhanced by Pasco, Langdon and Rissanen. This algorithm will not assign a binary bit code to the character similar to the one Huffman and Shannon Fano coding, instead it replace the entire stream of input by a single floating point value. Here each and every character is assigned by an interval with the starting value as '0' and '1'. These intervals are further divided to many subintervals and repeated till the last symbol. The compressed file will have a single floating point number which can be reversible to the original file.

IV. DICTIONARY CODE COMPRESSION TECHNIQUE

The basic idea behind the dictionary code compression technique is observing certain pattern of characters repeatedly occurred in the input text file. These patterns are placed in the dictionary as key and assigning a value to the key. Usually the size of value is smaller than the key pattern. The original file is substituted by the value of the corresponding key which forms a compressed file. This paper explains about three fundamental dictionary based compression techniques such as LZ77, LZ78 and LZW.

4.1 LZ77:

The name LZ77 is the abbreviation of the authors Jacob Ziv and Abraham Lempel, where 77 represents the year 1977 when the algorithm was developed. The LZ77 algorithm finds a word or phrase in the text file given as input which was frequently repeated. Each new word or phrase scanned in the input file was given a pointer as code. This pointer is added in the dictionary. If the same phrase or word again appeared in the file it was replaced by the corresponding pointer, which causes the file to be compressed. The examination of text file was carried out by moving a sliding window. The sliding window has two parts: search buffer and look ahead buffer. The look ahead buffer contains the string to be scanned and search buffer has the recently encoded pattern. The pointer was represented by a triplet code <o, l, c> where o, l and c are the offset, length and matching word or phrase respectively. Figure 2 shows an example for LZ77 processing. The compressed file will have the triplet codes.

4.2 LZ78:

Jacob Ziv and Abraham Lempel in 1978 created another dictionary based compression technique named as LZ78. In LZ77 the decompression process can be performed based on the pointer itself, where as in LZ78 a dictionary has to maintain separately to carry out the decompression process. Here the dictionary stores an index along with the matching pattern. To decompress a file the patterns are replaced by the index. A pair of values are used for encoding which was represented by <i, c>, where i is the index and c is the matching pattern symbol. Figure 3 shows the example of LZW compression.

4.3 LZW:

LZW was developed by Terry Welch based on LZ78 algorithm. LZW is the name of the authors Jacob Ziv, Abraham Lempel who developed LZ78 and Terry Welch. The LZW has to maintain each and every symbol and its index initially in the dictionary of the input text file. The dictionary contains the

index, pattern and the code. Figure 4 shows the example of LZW compression.

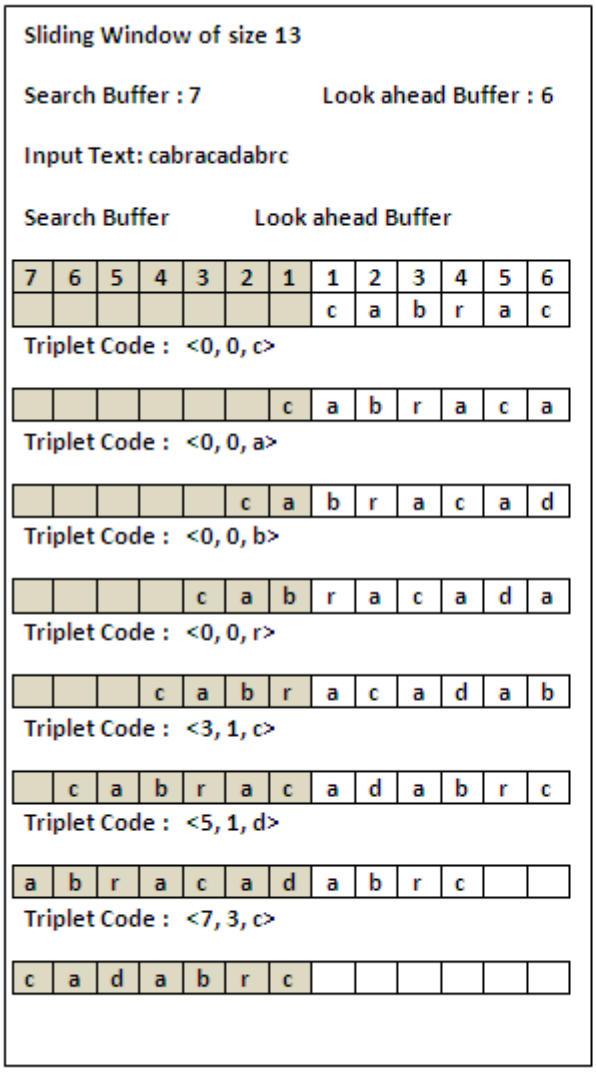

Figure 2: LZ77 Sliding Window Example

Input Text: abracadabra

DICTIONARY		OUTPUT CODE
INDEX	ENTRY	
1	a	< 0, a >
2	b	< 0, b >
3	r	< 0, r >
4	ac	< 1, c >
5	ad	< 1, d >
6	ab	< 1, b >
7	ra	< 3, a >

Figure 3: Example of LZ78 Compression

Input Text: ababbabc

DICTIONARY		OUTPUT CODE
INDEX	ENTRY	
1	a	-
2	b	-
3	c	-
4	ab	1
5	ba	2
6	abb	4
7	bab	5
8	bc	2
-	-	3

Figure 4: Example of LZW Compression

V. COMPRESSION UTILITY SOFTWARE

5.1 DEFLATE:

The compressed file has a specific format of data which can be decompressed to original file. Deflate is a most widely used file format in lossless data compression. Deflate performs the compression process by combining both LZSS and Huffman coding algorithm. LZSS is the algorithm derived from LZ77. Deflate is a computer program designed by Phil Katz. It was designed and developed for PKZIP archiving tool which is a file archive. Deflate file has streams made up of sequence of blocks. Each block has a header of 3 bits. The first bit has a '1' or '0', where '1' represents the last block of the stream and '0' represents many blocks are to be processed. The second and third bits are used for encoding method.

5.2 ZIP:

ZIP is utility software that does lossless compression and creates a zip file format. A ZIP can have one or more files and directories. The compressed file has .zip extension. The ZIP compression was created by Phil Katz who developed deflate compression. Originally ZIP was implemented in PKZIP and later it was included in many utility software [34]. The operating system Windows and Mac utility software can perform zip and unzip. ZIP compression uses combination of many compression algorithms in which deflate plays a major role. The zip file also acts as a base file format to other compression utility software.

5.3 GZIP:

GZIP is a compression program written in C programming language released in the year 1992. It was created by Jean Loup, Gailly and Mark Adler. The GZIP program was implemented in the UNIX operating system [8]. The file compressed using GZIP has the extension '.gz' which is a compressed file format. Deflate is the base for GZIP compression and also it is the combination of LZ77 and Huffman coding algorithm. GZIP can compress a single file only. The GZIP file format has a header of 10 bytes, optional header, and a body of deflate file and a footer of 8 bytes.

VI. TAMIL COMPRESSION TECHNIQUE

Lots of research was carried out in the field of compression to develop a compression process for specific languages based on its nature [3] [6]. Tamil Compression Technique (TCT) is a novel compression method to

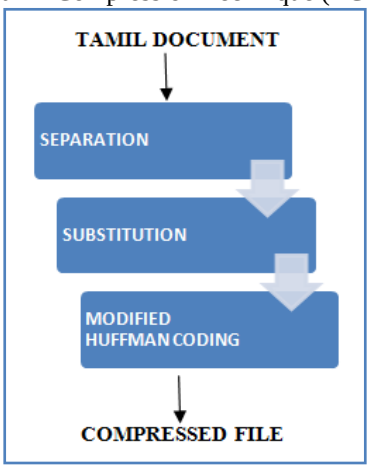

Figure 5: Steps in Tamil Compression Technique

compress Tamil documents. It is performed by a three step process. In this compression process a Tamil text document was given as input and the output is the compressed file. Tamil compression technique was shown in the figure 5. The first step called the separation process that separates the English alphabets from the Tamil words in the Tamil document using a pair of special symbols. Unicode is the universally accepted standards for data storage and transmission [11] and [14]. Almost all the documents of any languages were built using the text of Unicode characters [16] and [22]. All the characters of Tamil language are encoded as per the Universal Principle of Unicode. The Tamil characters are range from U+0B80 to U+0BFF in Unicode character set [2][30]. The second process is the substitution process where the Unicode Tamil characters are replaced by ASCII characters [37].

Table 1: Size of memory occupied by the word 'முகநூல்'

Tamil Alphabet	Combination of Unicode Characters	Size in memory (Bytes)
மு	ம + ு	4
க	-	2
நூ	ந + ூ	4
ல்	ல + ்	4
	Total:	**14**

This replacement reduce the size of memory to half, since Tamil Unicode character takes 2 bytes of memory where as ASCII character takes 1 byte of memory.

Tamil characters are the combination of vowels and consonants [9][23]. All the 'Uirmaiezuthu' is a combination of prefix or suffix or both of consonants [12]. Table 1 shows the size of memory occupied by a Tamil word. After the substitution process of ASCII character in the place of Tamil characters the memory needed is 7 bytes. In the third step a modified Huffman coding process is carried out by building the Huffman tree based on frequency of words. The output of the third step is the compressed file. The algorithm for Tamil compression is given in the figure 6.

BEGIN

1. A Tamil document file is given as input.

2. **BEGIN: SEPARATION PROCESS**

   i. Scan the entire file to find any English alphabets or words in the input.
   ii. If English alphabets found, separate it from Tamil alphabets by placing a '~' symbol before and after.

   END

3. **BEGIN: SUBSTITUTION PROCESS**

   i. The output of step 2 is given as input to step 3.
   ii. Replace the Tamil alphabets by basic ASCII characters using static dictionaries.
   iii. The result is the document with collection of ASCII characters which is the intermediate form of compression process.

   END

4. **BEGIN: APPLY MODIFIED HUFFMAN ENCODING PROCESS**

   i. The intermediate file obtained in step 3 is given as input.
   ii. Select the words from the input based on number of characters and repetition of words.
   iii. Build a dynamic Huffman tree with leaf node has selected words from left to right in decreasing order.
   iv. The leaf node words are encoded in the input file.
   v. The encoded file is the output stored in the binary form.

   END

5. The result of step 4 is the compressed file.

END

Figure 6: Algorithm for Tamil Compression Technique

VII. COMPRESSION MEASUREMENT PARAMETER

Efficiency of compression can be measured by calculating the parameters for space and time. The efficiency of space can be finding out by compression ratio, compression factor and percentage of compression [7].

The compression ratio is the ratio between size of compressed file and the original file represented in formula 1.

$$\textbf{Compression Ratio} = \frac{\textbf{Compressed File Size}}{\textbf{Original File Size}} \quad (1)$$

The compression factor is the reverse of compression ratio. It is shown in the formula 2. It gives the ratio between original file size and the compressed file size.

$$\textbf{Compression Factor} = \frac{\textbf{Original File Size}}{\textbf{Compressed File Size}} \quad (2)$$

The percentage of compression is also called as saving percentage; because it gives the percentage of reduce in memory size from the original file size [35]. The compression percentage is calculated by subtracting the size of original file and compressed file and then divided it by original file size using the formula 3.

$$\textbf{Percentage of Compression} = \frac{\textbf{Bits Before Compression} - \textbf{Bits After Compression}}{\textbf{Bits Before Compression}} * \mathbf{100} \quad (3)$$

The parameters like compression ratio, compression factor and percentage of compression are used to find the efficiency of space. The formula in 1, 2 and 3 is used to find the storage of memory needed for a compressed file. The efficiency of time for a compression algorithm is calculated by the time taken to compress the file and decompress the file.

VIII. EXPERIMENTAL ANALYSIS AND RESULTS

In this section the parameters like percentage of compression using the formula 3 and time taken to compress and decompress the algorithms are calculated. The parameters for statistical compression techniques such as Run length encoding, Huffman coding, Shannon Fano and Arithmetic coding are compared followed by dictionary based compression technique LZ77, LZ78 and LZW. Then the comparison of utility programs like deflate, zip and gzip are performed.

Table 2 shows the comparison of space and time efficiency of statistical coding algorithms such as Run Length Encoding (RLE), Huffman coding, Shannon Fano coding and Arithmetic coding algorithms. In table 2, the space efficiency is calculated by the percentage of compression using the formula 3. The best result was given by Huffman Coding as the average percentage is 40.23. It was followed by Shannon Fano coding, Arithmetic Coding and Run length encoding with 37.92%, 37.34% and 0.87% respectfully.

Table 2: Comparison of space and time efficiency of statistical coding algorithms

File Name	File Size (Bytes)	Space Efficiency – Percentage of Compression				Time Efficiency – Compression Time (Seconds)			
		RLE	Huffman Coding	Shannon Fano Coding	Arithmetic Coding	RLE	Huffman Coding	Shannon Fano Coding	Arithmetic Coding
Bharathi	14746	0.28	39.49	39.04	38.98	0.015	0.16	0.2	0.21
Chennaithagaval	4260	0.16	38.39	37.6	36.83	0.01	0.08	0.22	0.34
Ettuthokkai	3072	1.5	39.32	35.05	33.49	0.008	0.07	0.24	0.26
Nambikkai	6124	1.5	39.43	37.64	37.44	0.01	0.11	0.19	0.23
Natrrinai	5634	0.48	35.99	31.71	31.67	0.01	0.05	0.18	0.17
Pathittrupathu	13074	0.75	44.26	40.72	39.93	0.014	0.12	0.23	0.21
Story1	3625	0.36	42.64	40.61	40.21	0.009	0.16	0.26	0.29
Story2	11167	0.23	42.22	40.51	39.99	0.013	0.78	0.94	0.92
Tamil_text	1065	3.38	37.93	36.03	35.53	0.008	0.04	0.17	0.16
Vairamuthuvaralaru	26222	0.09	42.62	40.33	39.32	0.018	0.29	0.41	0.43
Average		**0.87**	**40.23**	**37.92**	**37.34**	**0.012**	**0.19**	**0.30**	**0.32**

The time efficiency of compression is given in seconds in table 2. The time taken to compression is best for Run length encoding as the coding was simple. The next best result was given by Huffman coding with 0.19 seconds of compression time. It was followed by Arithmetic coding and Shannon Fano coding algorithm. So in statistical coding technique Huffman gives the overall best result.

The same parameters are calculated and compared for dictionary based encoding technique. Here the comparison was made between LZ77, LZ88 and LZW methods and shown in table 3.

Table 3: Space and time efficiency of dictionary based algorithms

File Name	File Size (Bytes)	Space Efficiency – Percentage of Compression			Time Efficiency – Compression Time (Seconds)		
		LZ77	LZ88	LZW	LZ77	LZ78	LZW
Bharathi	14746	46.25	50	52.5	0.6	0.12	0.7
Chennaithagaval	4260	41.24	45	51.24	0.09	0.2	0.17
Ettuthokkai	3072	46.26	50	53.74	0.09	0.07	0.12
Nambikkai	6124	40.01	43.75	51.26	0.05	0.2	0.52
Natrrinai	5634	46.25	50	52.5	0.08	0.09	0.49
Pathittrupathu	13074	41.25	47.5	55	0.7	0.2	0.68
Story1	3625	46.26	49.99	51.26	0.38	0.3	0.41
Story2	11167	38.75	45	53.75	0.04	0.5	0.58
Tamil_text	1065	46.29	49.95	52.49	0.08	0.2	0.3
Vairamuthuvaralaru	26222	46.25	50	51.25	0.79	0.3	0.8
Average		**43.88**	**48.12**	**52.5**	**0.29**	**0.06**	**0.48**

In which the LZ77 has a pointer that points the compressed data value as shown in figure 2, where as the algorithm LZ78 and LZW both has to maintain the compressed data and dictionary of entry and its index. The percentage of compression and compression time was taken to analysis. The result shows that the memory needed to store LZW is low when compared to LZ77 and LZ78.

.

Table 4: Space and Time efficiency of utility compression software

File Name	File Size (Bytes)	Space Efficiency – Percentage of Compression			Time Efficiency – Compression Time (Seconds)		
		Deflate	ZIP	GZIP	Deflate	ZIP	GZIP
Bharathi	14746	69.6	70.76	70.21	2	0.9	4
Chennaithagaval	4260	60.35	64.67	64.48	0.5	0.21	1
Ettuthokkai	3072	64.29	67.19	65.63	0.2	0.08	0.2
Nambikkai	6124	67.46	68.57	68.03	0.4	0.09	0.5
Natrrinai	5634	83.9	83.07	84.26	0.1	0.16	0.2
Pathittrupathu	13074	79.6	80.73	78.73	0.5	0.24	0.3
Story1	3625	64.36	65.54	64.44	1.4	0.9	0.9
Story2	11167	73.72	74.42	74	0.19	0.1	0.16
Tamil_text	1065	23.85	33.9	24.98	4	0.66	2.5
Vairamuthuvaralaru	26222	72.97	73.88	74.12	1.3	0.18	0.6
Average		**66.01**	**68.27**	**66.89**	**1.06**	**0.35**	**1.04**

The comparison of space and time efficiency of utility compression software is shown in table 4. The comparison was made between the compression file formats generated by the utility software such as deflate, ZIP and GZIP. The space efficiency was calculated using the formula 3 of percentage of compression. The average compression percentage of deflate, ZIP and GZIP was 66.01%, 68.27% and 66.89% respectively. It shows that ZIP algorithm gives the best result to store the compressed file. The time efficiency of the utility compression programs are also shown in the table 4. The ZIP gives the best compression time of average 0.35 seconds. It was followed by GZIP and deflate compression with average compression time 1.04 and 1.06 seconds respectively. The table 4 shows that ZIP compression gives the better result when compared to deflate and GZIP compression.

Table 5: Decompression time of static, dictionary and utility based compression Technique

File Name	STATISTICAL COMPRESSION ALGORITHM				DICTIONARY BASED COMPRESSION ALGORITHM			COMPRESSION UTILITY SOFTWARE		
	RLE	Huffman Coding	Shannon Fano Coding	Arithmetic Coding	LZ77	LZ78	LZW	Deflate	ZIP	GZIP
Bharathi	1.03	0.04	0.5	3.2	0.04	0.12	0.3	0.02	0.12	0.2
Chennaithagaval	2.01	0.006	0.56	2.5	0.008	0.2	0.09	0.02	0.1	0.03
Ettuthokkai	0.98	0.007	0.9	2.8	0.01	0.07	0.2	0.02	0.14	0.04
Nambikkai	0.12	0.004	0.8	3	0.009	0.2	0.3	0.03	0.18	0.05
Natrrinai	0.23	0.007	0.2	2.42	0.02	0.09	0.1	0.19	0.11	0.06
Pathittrupathu	0.5	0.03	0.82	1.86	0.08	0.2	0.45	0.3	0.24	0.3
Story1	1.09	0.006	0.3	1.56	0.006	0.3	0.5	0.16	0.16	0.12
Story2	2.03	0.05	1.43	2.13	0.1	0.5	0.7	1.02	1.02	0.25
Tamil_text	1.07	0.007	0.36	1.63	0.03	0.2	0.3	0.09	0.09	0.13
Vairamuthuvaralaru	3.14	0.002	2.43	2.16	0.05	0.3	0.5	0.12	0.16	0.13
Average	**1.22**	**0.02**	**0.83**	**2.33**	**0.04**	**0.22**	**3.44**	**0.197**	**0.23**	**0.13**

Table 6: Space and Time efficiency of Tamil Compression Technique

File Name	File Size (Bytes)	Compressed File Size (Bytes)	Space Efficiency – Percentage of Compression			Time Efficiency (Seconds)	
			Compre-ssion Ratio	Compression Factor	Percentage of compression	Compression Time	Decompres-sion Time
Bharathi	14746	4035	0.274	3.655	72.64	0.19	0.09
Chennaithagaval	4260	1383	0.325	3.08	67.54	0.13	0.05
Ettuthokkai	3072	904	0.294	3.398	70.57	0.1	0.05
Nambikkai	6124	1782	0.291	3.437	70.9	0.13	0.05
Natrrinai	5634	834	0.148	6.755	85.2	0.08	0.04
Pathittrupathu	13074	2324	0.178	5.626	82.22	0.14	0.11
Story1	3625	1178	0.325	3.077	67.5	0.18	0.04
Story2	11167	2704	0.242	4.13	75.79	0.87	0.18
Tamil_text	1065	501	0.47	2.126	52.96	0.05	0.04
Vairamuthuvaralaru	26222	6444	0.246	4.069	75.43	0.36	0.06
Average			**68.27**	**66.89**	**72.08**	**0.22**	**0.07**

The efficiency of Tamil Compression Technique was also measured using the parameters such as compression ratio, Compression factor and compression percentage using the formula 1, 2 and 3. Followed by time taken for compression and decompression was extracted. The TCT algorithm was developed and executed in ASP.NET. The result was displayed in the table 6 [36]. The average percentage of compression is 72.08%. The average compression and decompression for TCT algorithm is given in seconds as 0.36 seconds for compression and 0.06 seconds for decompression. It shows that decompression takes less time than compression.

The time efficiency of compression algorithm measured by both compression and decompression time parameter. The compression time was given in the table 2, 3 and 4 for the statistical compression algorithms, dictionary based compression methods and compression utility software. The comparison was made between all the algorithms. Similarly the time taken for decompress a file was also considered as efficiency measure of compression algorithms. The decompression time in seconds was shown in the table 5. It contains all the files listed in the Table 2, 3 and 4. The least time taken for decompression by algorithms in static based compression technique was Huffman coding with 0.02 seconds. In dictionary based compression the LZ77algorithm results the least decompression time as 0.04 seconds. The decompression time for GZIP compression is 0.13 seconds which results the best when compared to deflate and ZIP compressor.

The table 2, 3, 4 and 5 shows the values of parameters such as percentage of compression, time taken to compress and decompress the files using static, dictionary and compressor utility programs. It also shows the average values for all the parameters of the algorithms. Table 6 shows the result of parameters taken after executing the novel TCT algorithm. The comparison was performed by taking the best average result of static based compression, dictionary based compression and utility compressor such as Huffman coding, LZW and ZIP respectively based on space efficiency. The average value of the parameters are taken for these algorithms and compared with the average values of TCT algorithms shown in table 6. The percentage of compression for Huffman coding, LZW and ZIP was compared and analyzed in the figure 7.

The percentage of compression was considered as one of the measurement parameter for space efficiency of compression algorithms. Increase in percentage of compression decreases the memory storage capacity. The average values of percentage of compression for Huffman coding, LZW, ZIP and TCT was compared in figure 7. It shows clearly that TCT gives the best result of compression percentage as 72.08.

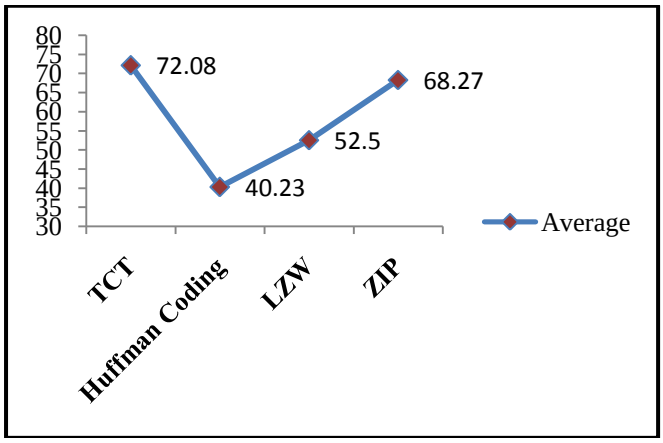

Fig 7: overall comparison of compression percentage

After comparing the space efficiency of the algorithms, the time efficiency was also analyzed. The performance of time efficiency was carried out for the Huffman coding, LZW, ZIP and TCT algorithms.

Table 7: Average Time taken to compress and decompress in seconds

Process	TCT	Huffman Coding	LZW	ZIP
Compression Time	0.22	0.19	0.48	0.35
Decompression Time	0.07	0.02	3.44	0.23

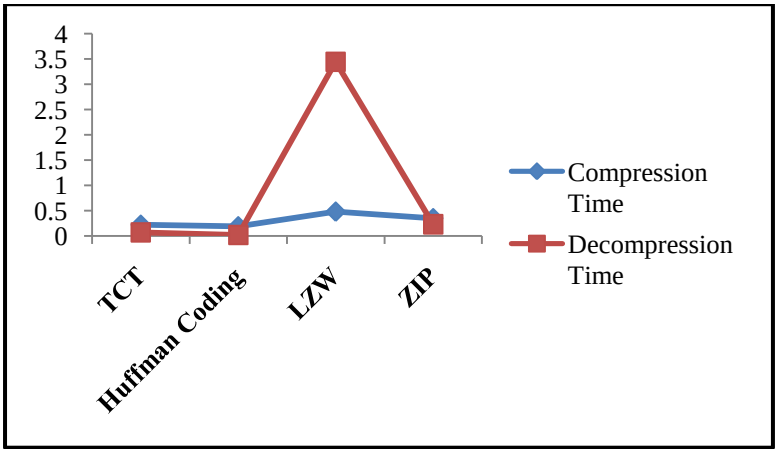

Fig 8: Overall comparison of compression and decompression time in seconds

Time taken for compressing and decompressing was taken for analysis which was shown in figure 8. Table 7 shows the average compression and decompression time for TCT, Huffman coding, LZW and ZIP taken from the table 5 and 6. From the figure 8 it shows that, Huffman coding takes less average time to compress and decompress the file as 0.19 seconds and 0.02 seconds respectively. TCT gives the next best result followed by Huffman coding algorithm as 0.22 seconds for compression and 0.07 seconds for decompression.

IX. CONCLUSION

An experimental comparison and analysis was carried out on different lossless compression algorithms of Tamil text documents. Several text compression algorithms from statistical coding compression, dictionary based encoding and compression utility programs performance was compared with proposed Tamil Compression Technique (TCT) to find its efficiency. The average percentage of compression was calculated and the best resulted algorithm was selected. Huffman coding, LZW and ZIP techniques were selected from statistical coding algorithms, dictionary based algorithms and utility compressor programs which has 40.23%, 52.5% and 68.27% average compression percentage. The result was compared with TCT average compression

percentage which gives best result with an improvement of 31.85%, 19.58% and 3.81% over Huffman coding, LZW and ZIP respectively. TCT also gives the best result in compression and decompression time as 0.22 seconds and 0.07 seconds after Huffman coding. While considering the overall performance of all the algorithms and the parameters like compression percentage, time taken to compression and decompression TCT gives the better result.

REFERENCES

[1]. Apoorv Vikram Singh and Garima Singh, "A survey on different text Data Compression Techniques", International Journal of Wisdom Based Computing, Vol.1, December 2011.

[2]. Apte, Akshay, and Harshad Gado. "Tamil character recognition using structural features." (2010).

[3]. Arafat Awajan and Enas Abu Jrai, "Hybrid Techniques for Arabic Text Compression", Global Journal of Computer Science and Technology:C Software and Data Engineering, Vol 15 Issue 1 Version 1.0 2015, Print ISSN: 0975-4350 (2015).

[4]. Arup Kumar Bhattacharjee, Tanumon Bej and Saheb Agarwal, "Comparison study of Lossless Data Compression Algorithms for Text Data". IOSR Journal of Computer Engineering, Vol-11, issue-6, Jun 2013.

[5]. Awan, Fauzia S., et al. "LIPT: A Reversible Lossless Text Transform to Improve Compression Performance." Data Compression Conference. 2001.

[6]. Barua, Linkon, et al. "Bangla text compression based on modified Lempel-Ziv-Welch algorithm." Electrical, Computer and Communication Engineering (ECCE), International Conference on. IEEE, 2017.

[7]. Bhattacharjee, Arup Kumar, Tanumon Bej, and Saheb Agarwal. "Comparison study of lossless data compression algorithms for text data." IOSR Journal of Computer Engineering (IOSR-JCE) 11.6 (2013): 15-19.

[8]. Carus, A., and A. Mesut. "Fast text compression using multiple static dictionaries." Information Technology Journal 9.5 (2010): 1013-1021.

[9]. Dr.J.Venkatesh and C.Sureshkumar, "Tamil Handwritten Character Recognition Using Kohonon's Self Organizing Map", International Journal of Computer Science and Network Security, Vol. 9 No. 12, December 2009.

[10]. Fano R.M, "The Transmission of Information", Technical Report No. 65, Research Laboratory of Electronics, M.I.T., Cambridge, Mass, 1949

[11]. Gleave, Adam, and Christian Steinruecken. "Making compression algorithms for Unicode text." arXiv preprint arXiv:1701.04047 (2017).

[12]. Hewavitharana, S., and H. C. Fernando. "A two stage classification approach to Tamil handwriting recognition." Proc. TI (2002).

[13]. Huffman D.A., "A method for the construction of minimum-redundancy codes", Proceedings of the Institute of Radio Engineers, Vol. 40, No.9, pp. 1098–1101, 1952.

[14]. Juul, Svend, and Morten Frydenberg. "UNICODE2ASCII: Stata modules to translate between Unicode and ASCII." Statistical Software Components (2016).

[15]. Kodituwakku, S. R., and U. S. Amarasinghe. "Comparison of lossless data compression algorithms for text data." Indian journal of computer science and engineering 1.4 (2010): 416-425.

[16]. Kuo, Shihjong. "Processors, methods, systems, and instructions to transcode variable length code points of unicode characters." U.S. Patent No. 9,626,184. 18 Apr. 2017.

[17]. Langdon G.G., "An introduction to arithmetic coding", IBM Journal of Research and Development, Vol. 28, No. 2, pp. 135–149, 1984.

[18]. Nishad, R. Manicka Chezian, "A Survey on Lossless Dictionary Based Data Compression Algorithms", Internation Journal of Science, Engineering and Technology Research, vol 2, issue 2, February 2013.

[19]. Pannirselvam, S., and D. Selvanayagi. "A Comparative Analysis On Different Techniques In Text Compression." International Journal of Innovative Technology and Creative Engineering, ISSN:2045-8711, Vol.5 No.8 (August 2015).

[20]. Pasco.R., "Source coding algorithms for fast data compression", Ph.D thesis, Department of Electrical Engineering, Stanford University, 1976.

[21]. Porwal, Shrusti, et al. "Data compression methodologies for lossless data and comparison between algorithms." International Journal of Engineering Science and Innovative Technology (IJESIT) Volume 2 (2013): 142-147.

[22]. Raja, J. Nelson, P. Jaganathan, and S. Domnic. "A new variable-length integer code for integer representation and its application to text compression." Indian Journal of Science and Technology 8.24 (2015).

[23]. Ramachandran, Raj, and Ashik Ali. "Social challenges faced by technology in developing countries: focus on Tamil Nadu State." (2017).

[24]. Rissanen J and Langdon G.G., "Arithmetic coding", IBM Journal of Research and Development, Vol. 23, No. 2, pp. 149–162, 1979.

[25]. Rissanen J., "Generalised Kraft inequality and arithmetic coding", IBM Journal of Research and Development, Vol. 20, No. 3, pp. 198–203, 1976.

[26]. Sajilal Divakaran, Biji C.L., Anjali. C, Achuthsankar s. Nair, "Malayalam Text Compression", International Journal of Information Systems and Engineering, Vol 1, No. 1, April 2013.

[27]. Sandip V Maniya, MJ Sheth, K Lad - pdfs.semanticscholar.org, "Compression Technique based on Dictionary approach for Gujarati Text", International Journal of Engineering Research and Development eISSN : 2278-067X, pISSN : 2278-800X, www.ijerd.com Volume 4, Issue 8, PP. 101-108 (November 2012).

[28]. Sangwan, Nigam. "Text encryption with huffman compression." International Journal of Computer Applications 54.6 (2012).

[29]. 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

[30]. Seethalakshmi.R, Sreeranjani.T.R, Balachandaar.T, "Optical Character Recognition for Printed Tamil Text using Unicode", Journal of Zhejiang University Science, ISSN 1009-3095, 2005 6A(11):1297-1305 (2005).

[31]. Ševčík, Jiří, and Jiří Dvorský. "Techniques of Czech Language Lossless Text Compression." IFIP International Conference on Computer Information Systems and Industrial Management. Springer International Publishing, 2016.

[32]. Shannon C.E., "A mathematical theory of communication," The Bell System Technical Journal, Vol. 27, pp. 398-403, 1948.

[33]. Siva Jyothi Chandra, Ashlesha Pandhare, Mamatha Vani, "Multilingual Font Creation by Mapping Unicode to Ascii", International Journal of Advanced Research in Computer Science and Software Engineering, Vol 5, Issue 9, Sep 2015, ISSN: 2277 128X (2015).

[34]. Stay, Michael. "ZIP attacks with reduced known plaintext." International Workshop on Fast Software Encryption. Springer, Berlin, Heidelberg, 2001.
[35]. Tanvi Patel and Judith Angela, KurtiDangarwala. " Survey of Text Compression Algorithms", International Journal of Engineering Research and Technology, issue-3, March 2015.
[36]. Vijayalakshmi, B., and N. Sasirekha "Lossless Tamil Compression using ASCII substitution and Modified Huffman Encoding Technique", International Journal of Recent Technology and Engineering, ISSN:2277-3878,Vol 8, Issue 6, March 2020
[37]. Vijayalakshmi, B., and N. Sasirekha. "Lossless Text Compression For Unicode Tamil Documents." ICTACT Journal on Soft Computing 8.2 (2018).
[38]. Welch T.A., "A technique for high-performance data compression", IEEE Computer, Vol. 17, No. 6, pp. 8–19, 1984.
[39]. Ziv. J and Lempel A., "A Universal Algorithm for Sequential Data Compression", IEEE Transactions on Information Theory, Vol. 23, No. 3, pp. 337–342, 1977.
[40]. Ziv. J and Lempel A., "Compression of Individual Sequences via Variable-Rate Coding", IEEE Transactions on Information Theory, Vol. 24, No. 5, pp. 530–536, 1978.