Scientific Research Publishing

Open Journal of Applied Sciences, 2024, 14, 1944-1962
https://www.scirp.org/journal/ojapps
ISSN Online: 2165-3925
ISSN Print: 2165-3917

Quantitative Comparative Study of the Performance of Lossless Compression Methods Based on a Text Data Model

Namogo Silué[1,2,3], Sié Ouattara[1,2,4], Mouhamadou Dosso[3], Alain Clément[5]

[1]Laboratoire des Sciences et Technologies de la Communication et de l'Information (LSTCI), Institut National Polytechnique Houphouët Boigny (INPHB), Yamoussoukro, Côte d'Ivoire
[2]Institut National Polytechnique Houphouët Boigny (INPHB), Yamoussoukro, Côte d'Ivoire
[3]UFR Mathématique-Informatique, Ecole Doctorale Sciences-Technologie et Agriculture Durable (ED STAD), Université Felix Houphouët-Boigny (UFHB), Abidjan, Côte d'Ivoire
[4]Ecole de Géomatique et du Territoire (EGT), Abidjan, Côte d'Ivoire
[5]LARIS, SFR MATHSTIC, Université d'Angers, Angers, France
Email: sie_ouat@yahoo.fr

How to cite this paper: Silué, N., Ouattara, S., Dosso, M. and Clément, A. (2024) Quantitative Comparative Study of the Performance of Lossless Compression Methods Based on a Text Data Model. *Open Journal of Applied Sciences*, **14**, 1944-1962.
https://doi.org/10.4236/ojapps.2024.147127

Received: May 28, 2024
Accepted: July 27, 2024
Published: July 30, 2024

Abstract

Data compression plays a key role in optimizing the use of memory storage space and also reducing latency in data transmission. In this paper, we are interested in lossless compression techniques because their performance is exploited with lossy compression techniques for images and videos generally using a mixed approach. To achieve our intended objective, which is to study the performance of lossless compression methods, we first carried out a literature review, a summary of which enabled us to select the most relevant, namely the following: arithmetic coding, LZW, Tunstall's algorithm, RLE, BWT, Huffman coding and Shannon-Fano. Secondly, we designed a purposive text dataset with a repeating pattern in order to test the behavior and effectiveness of the selected compression techniques. Thirdly, we designed the compression algorithms and developed the programs (scripts) in Matlab in order to test their performance. Finally, following the tests conducted on relevant data that we constructed according to a deliberate model, the results show that these methods presented in order of performance are very satisfactory:

- LZW
- Arithmetic coding
- Tunstall algorithm
- BWT + RLE

Likewise, it appears that on the one hand, the performance of certain techniques relative to others is strongly linked to the sequencing and/or recur-

rence of symbols that make up the message, and on the other hand, to the cumulative time of encoding and decoding.

Keywords

Arithmetic Coding, BWT, Compression Ratio, Comparative Study, Compression Techniques, Shannon-Fano, Huffman, Lossless Compression, LZW, Performance, Redundancy, RLE, Text Data, Tunstall

1. Introduction

With the complexity of processing and storage power increasing according to John Moore's law, the production of data outstrips the increase in the complexity of the hardware and transmission [1]. It is therefore important to make information compact before transmitting or archiving it. Compression techniques need to be used to reduce the size of the data to be transmitted or stored more efficiently. But this can only be achieved by choosing the best information compression methods. Compressed data may lose some information deemed non-essential, which is referred to as lossy compression. When the original data is recovered after reconstruction, it is said to have been losslessly compressed. The information to be compressed can be in the form of audio, text or images. Here, we are using text data with lossless compression. In [1], the author compares the RLE, LZW, arithmetic and Huffman techniques to find out which is the fastest and most efficient by choosing the following evaluation criteria: compression rate, compression time, size and complexity of the text files. He was able to rank them according to the difference in compression, starting with the best. It was not possible to evaluate the compression rate or compression time because it was found that each technique compressed the selected text files differently in terms of word count. According to the work in **[2]**, based on the RLE, LZW and Huffman methods, it was found that RLE gave the best results in terms of compression rate for text data. However, when it comes to HTML files, RLE is second only to LZW. RLE, Delta, Huffman and LZW were also compared in [3]. LZW presented a better result with a good compression ratio of 4:1 and also 76.9% memory space saving. And Delta encoding is better than RLE and Huffman.

The comparative study of lossless compression methods is a priority in a context where the amount of data to be stored or transmitted is constantly increasing. Given this growth, it is essential to determine which compression method offers the best compromise between reducing data size and preserving data integrity.

The problem of this study lies in identifying the relative performance of the different lossless compression methods available. It is to understand how these techniques react to different types of data, while taking into account evaluation criteria. For the purposes of this paper, we have chosen certain text data models

to achieve our objective.

In this paper, we carry out a comparative study of the most widely used and relevant lossless compression techniques. In the next section, we present the state-of-the-art in lossless compression methods; then in Section 3, we propose a text dataset and determine the performance models of the lossless compression methods selected. Sections 4 and 5 are devoted to results and discussions respectively. The final section provides a conclusion and outlook.

2. State-of-the-Art Lossless Compression Methods

In the digital domain, in order to compact information, there is a choice of several compression methods: lossy and lossless [3]. Lossy techniques lead to a higher compression ratio than lossless techniques [4]. Lossless compression techniques reduce the size of files so that the original data can be accurately recovered. Some research has focused on the comparison of lossless compression encodings. For example, [5] states that arithmetic coding has a better compression ratio than Huffman coding. In [6], a comparison of the Shannon-Fano, Huffman, RLE and Tunstall algorithms applied to sentences shows that the Huffman and Shannon-Fano compression rates are the highest, while the RLE compression rate is the lowest. Nagamani N. P. *et al.* used compression ratio and space gain to compare Huff-man, Shannon-Fano and LZW coding. The results show that the Shannon-Fano technique offers better efficiency than the other techniques when the text size is less than 50 Kb [7]. [8] follows suit, but compares LZW, RLE, Huffman, Shannon-Fano, arithmetic coding, LZ77 and LZ78. Here, LZW is the most efficient (81.31% space saving). In [9], Huffman and arithmetic coding were compared. It was found that the arithmetic method was the most reliable, but not as fast as the Huffman method. [3] carried out a comparison between RLE, Huffman, LZW and Delta coding, evaluating them in terms of compression ratio, space gain, compression factor and elapsed time. The conclusion drawn is that LZW has a better compression ratio (4:1) and space saved (76.9%) but this method is complex and takes time to compress. However, Delta's encoding is better than RLE and Huffman. For Rajput A. A. *et al.*, using the same evaluation criteria, compared the RLE, LZW, Shannon-Fano and Huffman techniques and found that Shannon-Fano coding was better than RLE and Huffman coding was slower than LZW [10]. In [11], LZW is compared with adaptive Huffman coding on English and Arabic texts. We note that for compression time, adaptive Huffman outperforms LZW and the opposite for decompression time and bit rate. In [12], the evaluation of arithmetic and adaptive Huffman coding indicates that the arithmetic method is preferable to the adaptive Huffman method, taking into account the greater number of bits required for adaptive coding than for arithmetic coding. Alif M. *et al.* compare LZW and Huffman according to space gain and compression time. The result shows that LZW is superior to Huffman [13]. But what we note is that this research does not focus on the characteristics of the data and does not present a

tipping point for better comparison.

There are several reversible compression methods, such as: RLE coding, BWT, Huffman, arithmetic coding, LZW, Tunstall coding, Shannon-Fano, Golomb coding, Prediction by Partial Matching (PPM), Delta encoding, Context Tree Weighting (CTW), Move-to-Front, and so on. But in the context of our work, we will only use seven of these methods, which we consider to be the most relevant and the most widely used in the literature for compressing text data.

In this work, we want to qualitatively compare the performances of certain lossless compression methods retained through performance quantities such as entropy, average code length and compression rate, using a text data model.

2.1. RLE Encoding Principle

The RLE (Run-Length Encoding) algorithm is a simple and effective compression technique for compressing data with repetitive sequences.

- It runs through the data sequence, character by character, from left to right.
- It then identifies sequences of consecutive characters that are identical. When a repetitive sequence is detected, the algorithm counts the number of repetitions.
- The repeated sequence is represented by the character followed by the number of repetitions. This representation is then added to the already compressed part.

2.2. Principle of Coding RLE + BWT

2.2.1. Principle of the Burrows-Wheeler Transform (BWT)

This is a technique that does not compress. It rearranges the characters in a string in order to exploit redundancy and therefore facilitate data compression using certain compression techniques [14]. The steps can bedistinguished as follows:

- All the rotations in the string under consideration are generated by iteratively moving the last character to the first position. This produces a rotation matrix.
- The next step is to sort these rotations lexicographically to obtain a sorted matrix.
- The expected result is given by the last column of the sorted matrix.

2.2.2. The Combination of RLE and BWT

The RLE algorithm is applied to the BWT result to obtain compression.

2.3. Principle of Shannon-Fano Coding

This is a lossless coding method for compressing data invented by Claude Shannon and Robert Fano in the 1940s. The basic principleis:

- Construct a table of symbol occurrence frequencies sorted in descending order.
- Divide this table into two parts. Each must have a sum of frequencies equal

(or practically equal) to that of the other.

- Assign the binary digit 0 to the lower half, with the upper half taking the value 1.
- Repeat operations 2 and 3 on both halves, until each symbol represents only one part of the table.

2.4. Principle of Huffman Coding

The Huffman compression method reduces the number of bits used in the code of a fragment of information to a minimum. The principle can bedescribed in severalstages:

- Messages from the leaves of a tree, each carrying a weight equal to the probability P of occurrence of the corresponding message.
- Join the 2 nodes with the lowest weights into a parent node to which we attach a weight equal to the sum of these 2 weights.
- Repeat step 2 until a single tree root is obtained.
- Assign codes 0 and 1 to the root's direct descendant nodes.
- Continue downwards by assigning codes to all the nodes, each pair of descendants receiving codes L0 and L1 where L designates the code associated with the parent.

✓***Example of Huffman encoding*** [15]

Consider T =< ab bd ac ab be bd ac>, T is compressed using Huffman coding, the letter frequencies are: freq (a) = 4, freq (b) = 5, freq (c) = 2, freq (d) = 2, freq (e) = 1 and freq (space) = 6.

Figure 1 shows the Huffman coding tree.

Text Compression Based on Letter's Prefix in the Word.

code (a) = 011, code (b) = 01, code (c) = 11111, code (d) = 0111, code (e) = 01111, code (space) = 0. Xo, the encoding of T will be 011010-0101110-011111110-011010-01011110-0101110-011111110.

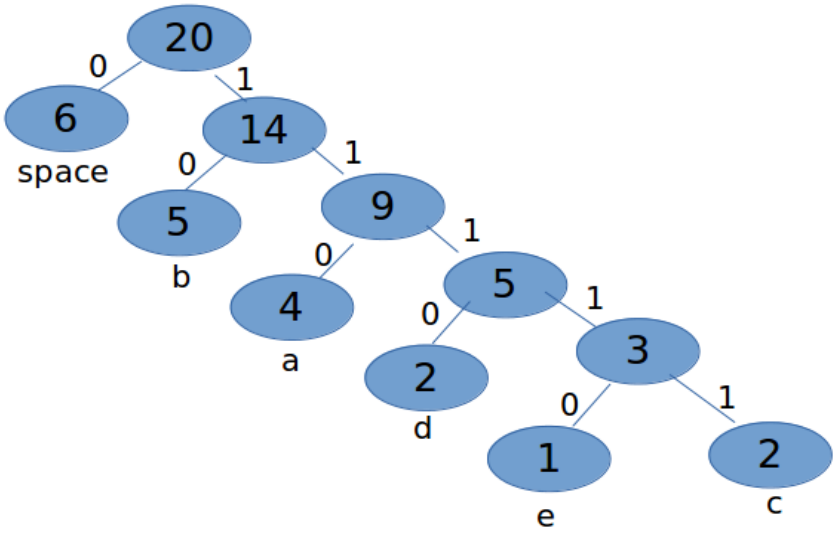

Figure 1. Huffman coding tree.

2.5. Arithmetic Coding Principle

Arithmetic coding is a compression technique used to represent a sequence of symbols by a single real number within a defined interval. The variouscoding stages are [16]:

- Let X be a sequence of m symbols with $X = x_1x_2....x_m$ which takes its values from a source $S = \{s_1, s_2, ...,s_n\}$,
- Calculate the probability associated with each symbol in source S, denote,
- p_i = Probability ($S = s_i$),
- Associate with each symbol sk a sub-interval $[L_{sk}, Hs_k[$ proportional to its probability, with $H_{sk} - L_{sk} = p_k$,
- Initialise the lower limit (L) of the working interval to the value 0 and the upper limit (H) to the value 1,
- As long as a symbol remains in the sequence to be coded:
  - o width =H – L,
  - o $L = L + width * L_{sk}$,
  - o $H = L + width * H_{sk}$,
- In the end, any value in the range [L, H[uniquely represents the input sequence.

In general, the lower limit (L) of the interval is chosen as the representative of the sequence, or the mean value ((L + H))/2 can be chosen. The value chosen to represent the sequence to be coded is called code(X). Remember that the code (X) is represented with a number of bits equal to the value $[-\log_2 (P(X))] + 1$.

Example of Arithmetic Coding [17]

Table 1 and Figure 2 illustrates an example arithmetic coding. Coding the word "COMCA" (Langdon, 1984).

Table 1. Example of statistics relating to the symbols in a message.

Symbol	Frequency	Probability	Interval
C	2	0.4	[0 - 0.4]
O	1	0.2	[0.4 - 0.6]
M	1	0.2	[0.6 - 0.8]
A	1	0.2	[0.8 - 1]
TOTAL	5	1	

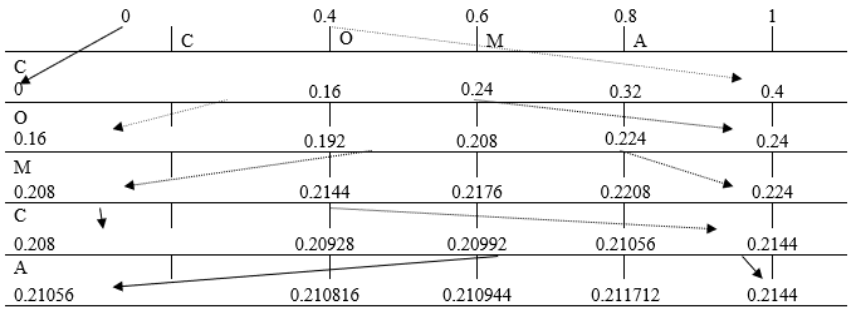

Figure2. Example of arithmetic coding.

2.6. Principle of Tunstall Coding

Tunstall coding uses fixed-length codes to represent whole blocks of symbols in a single operation. The principleis as follows [6]:

- n-bit code for a source S of an alphabet of size N;
- The number of code words is 2^n;
- The entry with the highest probability is taken and concatenated with the other letters of the alphabet;
- The size of the code increases from N to N + (N − 1);
- Pr (new entry) = Π Pr (letters);
- Repeat the same procedure;
- Each time the size of the code increases by N − 1;
- After K iterations the code size: N + K (N − 1);
- Stop condition: $N + K(N-1) \leq 2^n$.

2.7. Principle of Lempel Ziv Welch (LZW) Coding [18]

LZW coding is a compression technique which involves building a dictionary [1] as following:

- During data processing, character strings are placed one by one in the dictionary;
- When a string is already present in the dictionary, its usage frequency code is incremented;
- Character strings with high frequency codes are replaced by a "word" with the smallest possible number of characters, and the correspondence code is entered in the dictionary;

Lossless compression techniques are used in a number of areas where reducing data volume without degrading information is crucial since this reduction in the amount of data increases storage capacity and, above all, the capacity of the communication channel [19]. These methods offer a variety of approaches to achieving this objective.

3. Performance Evaluation of Selected Lossless Compression Methods

Evaluating the performance of lossless compression techniques is an essential process in choosing the most suitable and effective for a particular challenge. This requires the application of evaluation criteria such as compression ratio, average code length and entropy.

3.1. Our Model of Data

In order to gain a better understanding of the effectiveness of the compression techniques selected, we propose the following text data models for the various tests:

- Data 0:aaaaaaaa;
- Data 1:arararar;

- Data 2:abcabcabcabc;
- Data 3:abcdabcdcdabcdabcd.

Generalisation leads us to the data below:

- Data 0_G: $\underbrace{a \quad a}_{\text{ntimes}}$;
- Data 1_G: $\underbrace{ar \quad ar}_{\text{ntimes}}$;
- Data 2_G: $\underbrace{abc \quad abc}_{\text{ntimes}}$;
- Data 3_G: $\underbrace{abcd \quad abcd}_{\text{ntimes}}$;

Let's now calculate and present the fate of each of the generalized data sets, before and after the application of the selected compression techniques. Consider the following parameters: the initial data volume (IV), the compressed data volume (CV), the Compression ratio (CR), the average code length (L) and the source entropy (H).these parameters are calculated according to the following equations:

- The compression ratio measures the overall efficiency of compression [20].

$$\text{Compression ratio} = \frac{\text{Initial volume} - \text{Compressed volume}}{\text{Initial volume}} \tag{1}$$

- The average code length indicates the coding efficiency of the symbols

$$L = \sum_{k=1}^{N} P_k l_k \tag{2}$$

Source entropy defines the theoretical limit of lossless compression

$$H = \sum_{k=1}^{N} P_k \log_2\left(\frac{1}{Pk}\right) \tag{3}$$

The probabilities of the different characters in each of the data:

- *Data* 0_G: P(a) = 1;
- *Data* 1_G: P(a) = 1/2; P(r) = 1/2;
- *Data* 2_G: P(a) = 1/3; P(b) = 1/3; P(c) = 1/3;
- *Data* 3_G: P(a) = 1/4; P(b) = 1/4; P(c) = 1/4; P(d) = 1/4.

3.2. Application of Compression Methods to Data

3.2.1. Case of RLE

- *Data* 0_G

IV = na = 8n; CV = 8 + $\log_2$(n); CR = $\frac{8n - 8 - \log_2(n)}{8n}$

L = 1xn = n bits/symbol

H = 1x$\log_2$ (1) = 0 bit/symbol

- *Data* 1_G

IV = 8xn; CV = 1n + 8n; CR = -1/8

L = 0.5xn + 0.5n = n bits/symbol

H = 1 bit/symbol

- *Data* 2_G

IV = 8xn; CV = 1xn + 8xn; CR = −1/8

L = n bit/symbol

H = 1.583 bit/symbol

- *Data* 3_G

IV = 8xn; CV = 1xn + 8xn; CR = −1/8

L = n bit/symbol

H = 2 bits/symbol

3.2.2. Case of RLE + BWT

- *Data* 0_G

IV = na = 8xn; CV = 8 +$\log_2$(n); CR = $\frac{8n-8-\log_2(n)}{8n}$

L = n bit/symbol

H = 0 bit/symbol

- *Data* 1_G

IV = na = 8xn; CV = 8 + $\log_2$(n); CR = $\frac{8n-8-\log_2(n)}{8n}$

L = n bit/symbol

H = 1 bit/symbol

- *Data* 2_G

IV = na = 8xn; CV = 8 + $\log_2$(n); CR = $\frac{8n-8-\log_2(n)}{8n}$

L = n bit/symbol

H = 1.583 bits/symbol

- *Data* 3_G

IV = na = 8xn; CV = 8 + $\log_2$(n); CR = $\frac{8n-8-\log_2(n)}{8n}$

L = n bit/symbol

H = 2 bits/symbol

3.2.3. Case of Shannon-Fano

- *Data* 0_G

IV = 8n; CV = n; CR = 7/8

L = 1 bit/symbol

H = 0bit/symbol

- *Data* 1_G

IV = 8xn; CV = 2xn; CR = 3/4

L = 1 bit/symbol

H = 1bit/symbol

- *Data* 2_G

IV = 8xn; CV = 5xn; CR = 3/8

L = 1.65 bit/symbol

H = 1.583bit/symbol

- *Data* 3_G

IV = 8xn; CV = 8xn; CR = 0

L = 2 bit/symbol

H = 2bit/symbol

3.2.4. Case of Huffman

- *Data* 0_G

IV = 8xn; CV = n; CR = 7/8

L = 1 bit/symbol

H = 0 bit/symbol

- *Data* 1_G

IV = 8xn; CV = n; CR = 7/8

L = 1 bit/symbol

H = 1 bit/symbol

- *Data* 2_G

IV = 8xn; CV = 5xn/3; CR = 19/24

L = 1.65 bit/symbol

H = 1.583 bit/symbol

- *Data* 3_G

IV = 8xn; CV = 4xn; CR = 1/2

L = 2 bit/symbol

H = 2 bit/symbol

3.2.5. Case of Arithmetic Coding

- *Data* 0_G

IV = 8xn; CV = 2 bits; CR = 1 − 1/4xn

L = 2 bit/symbol

H = 0 bit/symbol

- *Data* 1_G

IV = 8xn; CV = 9 bits; CR = 1 − 9/8xn

L = 4.5 bits/symbol

H = 1 bit/symbol

- *Data* 2_G

IV = 8xn; CV = 21 bits; CR = 1 − 21/8xn

L = 7 bits/symbol

H = 1.583 bits/symbol

- *Data* 3_G

IV = 8xn; CV = 21 bits; CR = 1 − 21/8xn

L = 5.25 bits/symbol

H = 2 bits/symbol

3.2.6. Case of Tunstall Coding

- *Data* 0_G

IV = 8xn; CV = 2 bits; CR = 1 − 2/8xn

L = 2 bits/symbol

H = 0 bit/symbol

- *Data* 1_G

IV = 8xn; CV = 8 bits; CR = 1 − 1/n

L = 4 bits/symbol

H = 1 bit/symbol

- *Data* 2_G

IV = 8xn; CV = 21 bits; CR = 1 − 21/8xn

L = 7 bits/symbol

H = 1.583 bits/symbol

- *Data* 3_G

IV = 8n; CV = 64 bits; CR = 1 − 8/n

L = 16 bits/symbol

H = 2 bits/symbol

3.2.7. Case of LZW

- *Data* 0_G

IV = na = 8xn; CV = 2 + $\log_2(n)$ bits ; CR = $\frac{8n-8-\log_2(n)}{8n}$

L = 27 bit/symbol

H = 1x$\log_2(1)$ = 0 bit/symbol

- *Data* 1_G

IV = 8xn; CV = 3 + $\log_2(n)$ bits; CR = $\frac{8n-3-\log_2(n)}{8n}$

L = 18 bits/symbol

H = 1 bit/symbol

- *Data* 2_G

IV = 8xn; CV = 4 + $\log_2(n)$ bits ;CR = $\frac{8n-4-\log_2(n)}{8n}$

L = 18 bits/symbol

H = 1 bit/symbol

- *Data* 3_G

IV = 8xn; CV = 6 + $\log_2(n)$ bits; CR = $\frac{8n-6-\log_2(n)}{8n}$

L = 20. 25 bits/symbol

H = 2 bits/symbol

4. Results

4.1. Summary Table

The quantities L, H, are expressed in bit/symbol and the Compression Rate (CR) to be expressed as a percentage is indicated in Table 2 relating to the data constructed.

Table 2. Summary of the calculation of the formulas of H, L and CR of the constructed data.

DATA		COMPRESSION METHODS						
		RLE	BWT + RLE	Shannon-Fano	Huffman	Arithmetic coding	Tunstall coding	LZW
Data 0_G	CR	$\frac{8n-8-\log_2(n)}{8n}$	$\frac{8n-8-\log_2(n)}{8n}$	7/8	7/8	$1-1/4n$	$1-1/4n$	$\frac{8n-8-\log_2(n)}{8n}$
	L	N	N	1	1	2	2	27
	H	0	0	0	0	0	0	0
Data 1_G	CR	−1/8	$\frac{8n-8-\log_2(n)}{8n}$	3/4	7/8	$1-9/8n$	$1-1/n$	$\frac{8n-3-\log_2(n)}{8n}$
	L	N	N	1	1	4.5	4	18
	H	1	1	1	1	1	1	1
Data 2_G	CR	−1/8	$\frac{8n-8-\log_2(n)}{8n}$	3/8	19/24	$1-21/8n$	$1-21/8n$	$\frac{8n-4-\log_2(n)}{8n}$
	L	N	N	1.65	1.65	7	7	18
	H	1.583	1.583	1.583	1.583	1.583	1.583	1
Data 3_G	CR	−1/8	$\frac{8n-8-\log_2(n)}{8n}$	0	1/2	$1-21/8n$	$1-8/n$	$\frac{8n-6-\log_2(n)}{8n}$
	L	N	N	2	2	5.25	16	20
	H	2	2	2	2	2	2	2

4.2. Construction of Compression Ratio Curves as a Function of *n* (Number of Character Repetitions) and Sign Tables for These Curves

Regarding Table 3, it shows the signs according to the tipping points between the different compression techniques, each expressed as a function (see Table 2). Equations are performed with these functions in pairs in order to find these tipping points (3.10 - 4, 1, 1.8622, 2, 11.5270). Hence, for Data 0_G we have the following signs:

- (RLE) − (BWT+RLE) = 0 on [0, 100].
- (RLE) − (Shannon-Fano) < 0 on [0, 11.527]; (RLE) − (Shannon-Fano) > 0 on [11.5270, 100].

And so on until the end of the table.

Indeed, Figure 3 presents a graphical representation showing the performance comparison of some compression methods on 0G data through the compression ratio (CR) illustrated in Table 3.

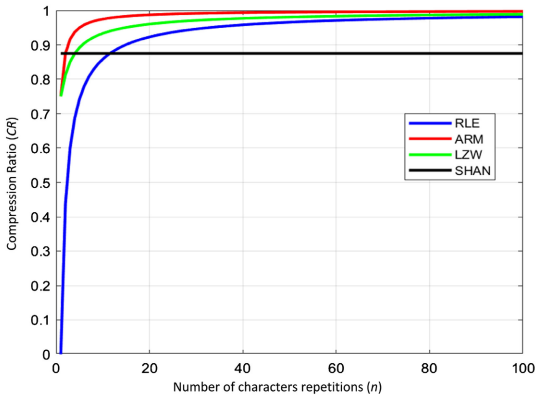

Figure 3. Compression Ratio (CR) for Data 0_G as a function of parameter *n*.

Table 3. Signs of Data 0_G.

n Name of courves	0	3.10^{-4}	1	1.8622	2	11.5270	100
RLE-BWT + RLE							
RLE-Shan			-			0	+
RLE-Huf			-			0	+
RLE-Arm		+		0		-	
RLE-Tuns		+		0		-	
Shan-Arm			+		0		
Shan-Tuns			+		0		
Shan-LZW	+	0			-		
Tuns-LZW		-	0		+		
Amr-Tuns							
Shan-Huf							
RLE-LZW				-			
Huf-Arm			+		0	-	
Huf-Tuns			+		0	-	
Huf-LZW	+	0			-		
BWT + RLE-Shan			-			0	+
BWT + RLE-Huf			-			0	+
BWT + RLE-Arm		+		0		-	
BWT + RLE-Tuns		+		0		-	
BWT + RLE-LZW				-			
Arm-LZW		-	0				

Regarding **Table 4**, the commentary remains the same as in **Table 3**, except that we are dealing with a different data (Data 1G). The tipping points are (0.86, 0.88, 1, 2, 4, 4.5, 5.18, 5.4, 8, 9, 11.5, 32, 64). The example of the meaning of the table is:

- [(RLE) − (BWT+RLE)] > 0 on [0, 0.86]; (RLE) − (BWT + RLE) < 0 on [0.86, 100].
- [(RLE) − (Shannon-Fano)] < 0 on [0, 100].

Indeed, as in **Figure 3**, **Figure 4** presents a graphical representation showing the performance comparison of some compression methods on 1_G data through the compression ratio (CR) illustrated in **Table 4**.

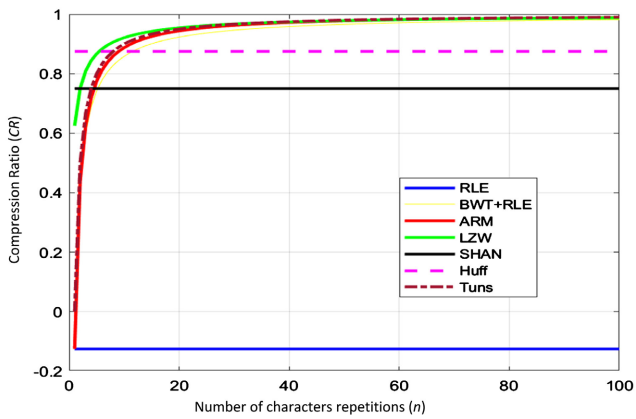

Figure 4. Compression ratio (CR) for Data 1_G as a function of parameter *n*.

Table 4. Signs of Data 1_G.

n Name of courves	$-\alpha$	0	0.86	0.88	1	2	4	4.5	5.18	5.4	8	9	11.5	32	64	00
RLE-BWT + RLE		+	0						-							
RLE-Shan									-							
RLE-Huf									-							
RLE-Arm			+		0						-					
RLE-Tuns				0						-						
RLE-LZW																
BWT + RLE-Shan					-				0				+			
BWT + RLE-Huf							-						0		+	
BWT + RLE-Arm			+			0					-					
BWT + RLE-Tuns			+		0						-					
BWT + RLE-LZW									-							
Shan-Huf									-							
Shan-Arm				+				0				-				
Shan-Tuns			+				0									
Shan-LZW		+				0										
Huf-Arm							+					0		-		
Huf-Tuns						+					0		-			
Huf-LZW					+					0		-				
Arm-Tuns									-							
Arm-LZW								+							0	-
Tuns-LZW							+							0	-	

Regarding Table 5, the comment remains the same as in Table 3, except that we are dealing with a different data (Data 2G). The tipping points are (0.69, 0.86, 1.76, 2.3, 3.47, 4, 4.2, 6.4, 100). The example of the meaning of the table is:

- [(RLE) − (BWT + RLE)] > 0 on [0, 0.86]; [(RLE) − (BWT + RLE)] <0 on [0.86, 100].
- [(RLE) − (Shannon − Fano)] < 0 on [0, 100].

Indeed, Figure 5 presents a graphical representation showing the performance comparison of some compression methods on 2_G data through the compression ratio (CR) illustrated in Table 5.

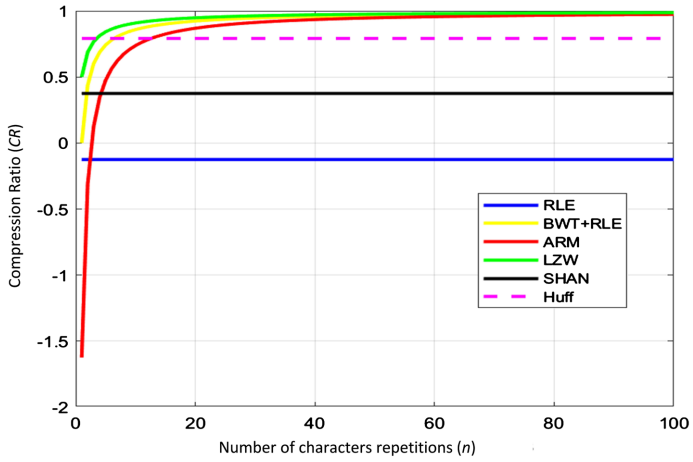

Figure 5. Compression Ratio (CR) for Data 2_G as a function of parameter n.

Table 5. Signs of Data 2_G.

n Name of courves	$-\alpha$	0	0.69	0.86	1.76	2.3	3.47	4	4.2	6.4	100
RLE-BWT + RLE			+	0			-			-	
RLE-Shan						-				-	
RLE-Huf						-				-	
RLE-Arm				+		0				-	
RLE-Tuns				+		0				-	
RLE-LZW						-				-	
BWT + RLE-Shan			-		0			+			
BWT + RLE-Huf									0	+	
BWT + RLE-Arm										-	+
BWT + RLE-Tuns										-	+
BWT + RLE-LZW										-	
Shan-Huf										-	
Shan-Arm					+				0	-	
Shan-Tuns					+				0	-	
Shan-LZW		+								-	
Huf-Arm					+				0	-	
Huf-Tuns					+				0	-	
Huf-LZW				+		0			-	-	
Arm-Tuns											
Arm-LZW										0	+
Tuns-LZW					+			0		-	

The commentary of **Table 6** remains the same as in **Table 3**, except that we are dealing with a different data (data 3_G). The tipping points are (0.57, 0.68, 0.86, 1, 1.68, 2.3, 2.33, 2.6, 5.25, 7.1, 8, 16, 100). The example of the meaning of the table is:

- [(RLE) − (BWT + RLE)] > 0 on [0, 0.86]; [(RLE) − (BWT + RLE)] < 0 on [0.86, 100].
- [(RLE) − (Shannon − Fano)]< 0 on [0, 100].

Indeed, **Figure 6** presents a graphical representation showing the performance comparison of some compression methods on 3_G data through the compression ratio (CR) illustrated in **Table 6**.

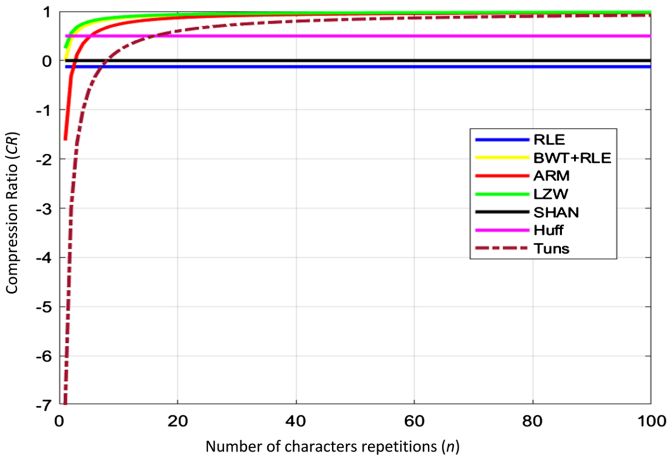

Figure 6. Compression ratio (CR) for Data 3_G as a function of parameter n.

Table 6. Signs of Data 3_G.

n Name of courves	0	0.57	0.68	0.86	1	1.68	2.3	2.33	2.6	5.25	7.1	8	16	100
RLE-BWT + RLE		+		0					-					
RLE-Shan								-						
RLE-Huf								-						
RLE-Arm					+			0				-		
RLE-Tuns						+					0	-		
RLE-LZW	+	0						-						
BWT + RLE-Shan			-		0					+				
BWT + RLE-Huf							0				+			
BWT + RLE-Arm								+						-
BWT + RLE-Tuns								+						-
BWT + RLE-LZW								-				-		
Shan-Huf														
Shan-Arm					+				0		-			
Shan-Tuns										+		-		
Shan-LZW	+											-		
Huf-Arm								+				-		
Huf-Tuns											+		-	
Huf-LZW				+								-		
Arm-Tuns														
Arm-LZW													0	+
Tuns-LZW													0	+

Applying the selected compression techniques to our selected text data provides relevant results for analysis.

5. Discussion

✓ **Data 0_G**

-The techniques: (RLE, BWT + RLE), (Shannon-Fano coding, Huffman), (arithmetic coding, Tunstall) both offer the same performance.

-Above the value of 11.5270, Shannon-Fano performs less well than all the other techniques selected. But before the values 3.10 - 4, 2, and 11.5270, Shannon-Fano is better than the other techniques.

- Apart from Shannon-Fano, and both before and after 11.5270, RLE is not as good as arithmetic and LZW coding.

- From value 1, arithmetic coding is better than LZW. And it is the best-performing technique in this data.

✓**Data 1_G**

- RLE is a poor encoder here with a negative CR and is inferior to the other compression techniques selected.

- In the same order of increasing performance, Shannon-Fano comes after RLE. It is above 5.18 that Shannon-Fano coding is worse than the others. But before this value, this coding is better than BWT + RLE, as well as arithmetic coding before 4.5, Tunstall before 4, and LZW before 2.

- Then we have Huffman coding, which performs less well than BWT + RLE, above 11.5, arithmetic coding, above 9, Tunstall, above 8 and LZW, above 5.4.

- LZW, Tunstall, arithmetic, BWT+RLE have similar performances, but LZW has the best result.

✓**Data 2_G**

- RLE is still the least efficient of the other techniques, from 2.3 upwards. But this technique is better arithmetically before this value where the CR is negative. And for others such as BWT + RLE and LZW, when RLE is more important, it is with an insignificant CR.

- After RLE, Shannon-Fano shows a poor performance compared to Huffman, arithmetic, BWT + RLE, and LZW, from 4.2.

- Huffman performs less well than arithmetic coding, BWT + RLE and LZW, above 4.2. But this technique is better than arithmetic coding before this value, BWT + RLE, before 6.4 and LZW, before 3.47.

- LZW has the best result in this data just after 3.47 compared to Huffman and the others.

✓**Data 3_G**

*- RLE appears to have the lowest performance, with a negative CR after a value of 7.1. However, before this value, this coding performed better than Tunstall, the arithmetic coding, before 2.*33, BWT + RLE, before 0.86 and LZW, before 0.57.

- Shannon-Fano performs less well than the others after 8. But before this value, this compression technique is better than Tunstall, arithmetic coding, before 2.3, BWT + RLE, before 1 and LZW, before 0.68.

- Still in the direction of increasing performance, Huffman comes after Shannon-Fano. This performs less well than LZW, arithmetic coding, Tunstall after the value 16. But before this value, Huffman performs better than Tunstall, arithmetic coding, before 5.25, BWT + RLE, before 2.3 and LZW, before 1.68.

- LZW is the best technique here, just ahead of BWT + RLE and arithmetic coding respectively.

6. Conclusions and Perspectives

In this paper, we have carried out a comparative study in terms of the performance of certain lossless compression techniques, namely: RLE, arithmetic coding, Huffman, BWT + RLE, Shannon-Fano, Tunstall, LZW and made proposals for text data models in order to achieve our goal. Our evaluations based on compression ratio, average code length and entropy have enabled us to develop underlying analytical models. We note that the decompression headers have not been considered in the models developed. This could reduce the performance of certain methods.

The results obtained, have indicated that it makes sense to consider tipping points when comparing lossless compression techniques applied to text data in order to be strictly accurate. Apart from 0_G data, RLE is inefficient. On the other hand, arithmetic, Tunstall, LZW and BWT + RLE coding give very good results, starting from the toggle values. The LZW coding performed very well in most of

our data.

For future work, we will address the following issues:

- The optimization of existing compression techniques for improved performance and reduced complexity can be explored;
- The adaptation of compression methods to specific types of data may be the subject of further research.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Baidoo, P.K. (2023) Comparative Analysis of the Compression of Text Data Using Huffman, Arithmetic, Run-Length, and Lempel Ziv Welch Coding Algorithms. *Journal of Advances in Mathematics and Computer Science*, **38**, 144-156. https://doi.org/10.9734/jamcs/2023/v38i91812

[2] Amoah, G.A. and Wahab, E.R.A. (2023) Comparative Analysis of Multimedia Compression Algorithms. *Asian Journal of Mathematics and Computer Research*, **30**, 26-37. https://doi.org/10.56557/ajomcor/2023/v30i28265

[3] Gopinath, A. and Ravisankar, M. (2020) Comparison of Lossless Data Compression Techniques. 2020 *International Conference on Inventive Computation Technologies* (*ICICT*), Coimbatore, 26-28 February 2020, 628-633. https://doi.org/10.1109/icict48043.2020.9112516

[4] Shanmugasundaram, S. and Lourdusamy, R. (2011) A Comparative Study of Text Compression Algorithms. *International Journal of Wisdom Based Computing*, **1**, 68-76.

[5] Porwal, S., Chaudhary, Y., Joshi, J. and Jain, M. (2013) Data Compression Methodologies for Lossless Data and Comparison between Algorithms. *International Journal of Engineering Science and Innovative Technology*, **2**, 142-147. https://www.ijesit.com/Volume%202/Issue%202/IJESIT201302_23.pdf

[6] Dea, A.R., Tito, W.P. and Anggunmeka, L.P. (2017) Comparison of Text Data Compression Using Huffman, Shannon-Fano, Run Length Encoding, and Tunstall Methods. *International Journal of Applied Engineering Research*, **12**, 13618-13622. https://www.ripublication.com/ijaer17/ijaerv12n23_80.pdf

[7] Nagamani, N.P., Sushruth, N., Sagar, J., *et al*. (2022) Comparative Analysis of Lossless Data Compression Techniques. 8*th National Conference on Advancements in Information Technology*, *NCAIT*-2022, Bengaluru, 16-17 June 2022, 176-180. https://www.ijesi.org/

[8] Gupta, A. and Nigam, S. (2021) A Review on Different Types of Lossless Data Compression Techniques. *International Journal of Scientific Research in Computer Science*, *Engineering and Information Technology*, **7**, 50-56. https://www.ijsrcseit.com/ https://doi.org/10.32628/cseit217113

[9] Prakash, F., Singh, V. and Saxena, A.K. (2024) An Evaluation of Arithmetic and Huffman Coding in Data Compression & Source Coding. 2024 *International Conference on Optimization Computing and Wireless Communication* (*ICOCWC*), Debre Tabor, 29-30 January 2024, 1-6.

https://doi.org/10.1109/icocwc60930.2024.10470768

[10] Anup, A., Ashok, R. and Raundale, P. (2019) Comparative Study of Data Compression Techniques. *International Journal of Computer Applications*, **178**, 15-19. https://doi.org/10.5120/ijca2019919104

[11] Kadhim, D.J., Mosleh, M.F. and Abed, F.A. (2024) Exploring Text Data Compression: A Comparative Study of Adaptive Huffman and LZW Approaches. *BIO Web of Conferences*, **97**, Article 00035. https://doi.org/10.1051/bioconf/20249700035

[12] Devi Kotha, H., Tummanapally, M. and Upadhyay, V.K. (2019) Review on Lossless Compression Techniques. *Journal of Physics: Conference Series*, **1228**, Article 012007. https://doi.org/10.1088/1742-6596/1228/1/012007

[13] Fauzan, M.N., Alif, M. and Prianto, C. (2022) Comparison of Huffman Algorithm and Lempel Ziv Welch Algorithm in Text File Compression. *IT Journal Research and Development*, **7**, 155-169. https://doi.org/10.25299/itjrd.2023.10437

[14] Effros, M., Visweswariah, K., Kulkarni, S.R. and Verdú, S. (2002) Universal Lossless Source Coding with the Burrows Wheeler Transform. *IEEE Transactions on Information Theory*, **48**, 1061-1081. https://doi.org/10.1109/18.995542

[15] Masmoudi, A. and Bouhlel, M.S. (2007) Un nouvel algorithme de compression exploitant le codage arithmétique en lui introduisant de nouveaux paramètres de codage. 12*ème journées d'étude et d'échange CORESA*'07 (*COmpression et REprésentation des Signaux Audiovisuels*), Montpellier. https://www.researchgate.net/publication/266577645

[16] AbuSafiya, M. (2020) Text Compression Based on Letter's Prefix in the Word. *Computers, Materials & Continua*, **64**, 17-30. https://doi.org/10.32604/cmc.2020.09282

[17] Taieh, E.A. (2018) The Pillars of Lossless Compression Algorithms a Road Map and Genealogy Tree. *International Journal of Applied Engineering Research*, **13**, 3296-3414. https://www.researchgate.net/publication/323883375_The_Pillars_of_Lossless_Compression_Algorithms_a_Road_Map_and_Genealogy_Tree

[18] Basma, E.D. (2023) Data Compression Algorithms and Their Applications. *Academy Journal for Basic and Applied Sciences* (*AJBAS*).

[19] Lal, E.M. and Rasheed, E.S. (2020) A Review on Data Compression Techniques. *International Journal of Advance Research and Innovative Ideas in Education*, **6**, 590-597.

[20] Tariq, J., Mosleh, M.F., Abdulameer, M., Obeidat, H.A. and Obeidat, O.A. (2023) Hybrid Lossless Compression Techniques for English Text. *Journal of Techniques*, **5**, 52-57. https://doi.org/10.51173/jt.v5i1.1059