Feasibility Study of Using Huffman Code Calculator in Learning Achievement of Data Compression

Zulhelman[1], Mohamad Fathurahman[2] and Endang Saepudin[3]

[1,2] Program Studi Broadband Multimedia, [3] Program Studi Elektronika Industri
Jurusan Teknik Elektro, Politeknik Negeri Jakarta
Jalan Prof GA. Siwabessy Kampus Universitas Indonesia Depok 16425

E-mail: zulhelman@elektro.pnj.ac.id

Abstract

In the world of education there is already a curriculum that must be carried out with learning outcomes that must be achieved by students, as well as lecturers who also have duties as facilitators in these achievements. With the Learning Outcomes, it makes students have the ability or competence in accordance with the requirements to meet the predefined study program profiles. One of the competencies that graduates of the Broadband Multimedia field must achieve is in the field of data processing including data compression. The most widely used data compression technique is the Huffman code. Some research on the Huffman code has been done a lot, and someone has made the Calculator application. The calculator is used to determine the code for each character. The code for each character is not fixed, depending on the frequency of appearance in a text. The more often the character appears, the shorter the code. In this study, it has proven that the feasibility of the Huffman Calculator as a tool for Learning Outcomes can be done where data compression with language variables, in this case English and Bahasa Indonesia to the compression ratio, produces the same compression ratio, which ranges from 1.51 to 1.64 for Bahasa Indonesia while English is 1.56 to 1.83, with the average entropy for English 4.35 and Bahasa Indonesia 4.41.

Keywords: competence , data compression, huffman code, learning outcomes.

1. Introduction

In computer science and information theory, Huffman coding is an entropy coding algorithm used for lossless data compression. The Huffman coding will result in a table of codes of varying lengths, these codes are derived in some way based on the estimated probability of occurrence for each of the possible values of the source symbol. The Huffman encoding is based on the frequency at which the character appears. With this code the characters that often occur will have a short code. The basis for this coding is the code tree according to Huffman, which assigns short code words to symbols. Characters that appear frequently in text will have a short code. Huffman tables are represented in the data stream. The algorithm for building coding follows this algorithm every symbol is a leaf and root. The flow chart of the Huffman algorithm is depicted in Figure 1. Huffman coding is very popular in the data compression area. Currently used in compressed data for wireless networks and, data mining[1]. Also found efficient for data compression on low resource Windows 7 systems. The use of Huffman code in word-based text compression is also very common. The Huffman principle generates optimal code using a Binary tree where most code lengths are smaller. However, the Huffman principle does not produce a balanced tree. For this reason, it takes more memory to store longer codewords, and it also takes more time to decode them from memory.

2. Literature Review and Hypothesis Development

The state of the art in the field of Information and Communication Technology, which is a very important part of today's information systems, is about Quality of Services (QoS). because QoS on the network is a requirement that must be met to run an application on the Internet network, so share information on companies, government sites, and science groups. This information can be in the form of documents, data folders, data shared and processed by various parties, and multimedia file streams are running well, including meeting large bandwidth requirements.

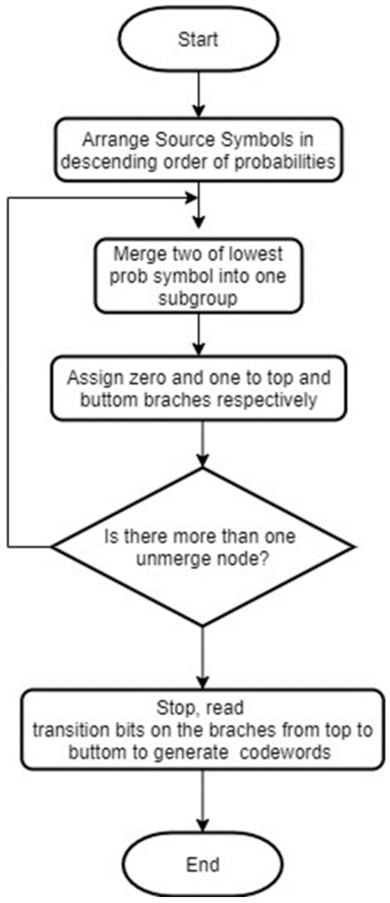
Start
Arrange Source Symbols in descending order of probabilities
Merge two of lowest prob symbol into one subgroup
Assign zero and one to top and buttom braches respectively
Is there more than one unmerge node?
Stop, read transition bits on the braches from top to buttom to generate codewords
End

Figure 1. Huffman Algorithm Flowchart

One of the efforts to reduce the bandwidth requirement is by compressing the data. The results of current research and relevant to scientific journals related to this research are:

- Evaluation of Huffman and Arithmetic Algorithms for Multimedia Compression Standards which reduce the quantity of data without excessively reducing the quality of multimedia data. The transmission and storage of compressed multimedia data is much faster and more efficient than the original uncompressed multimedia data. There are various techniques and standards for multimedia data compression, especially for image compression such as the JPEG and JPEG2000 standards. This standard consists of various functions such as color space conversion and entropy coding. Arithmetic and Huffman Coding are typically used in the entropy coding phase. In this paper we try to answer the following questions. Entropy, arithmetic or Huffman coding, which is more suitable than others from compression ratio, performance, and implementation point of view? We have implemented and tested the Huffman Algorithm and arithmetic. The results that we apply show that the compression ratio of arithmetic coding is better than Huffman coding, while the performance of Huffman coding is higher than arithmetic coding. In addition, the implementation of Huffman coding is much easier than arithmetic coding[2].
- Two dynamic codes based on the Huffman algorithm, Octonary and Hexanary, are used for data compression. Faster encoding and decoding processes are very important in the area of data compression. Tribal-based (Octonary) and quadbit-based (Hexanary) and comparing the performance with existing single bit which are widely used (Binary) and recently introduced dibit (Quaternary) algorithms. The decoding algorithm for the proposed technique has also been described. After assessing all the results, it was found that the Octonary and the Hexanary technique performed better than the existing techniques in terms of encoding and decoding speed. Use MPLS-TE for multimedia applications[3].
- Data reduction is a data pre-processing technique that can be applied to obtain a reduced representation of a much smaller data set while maintaining the integrity of the original data. This data reduction is data compression, where in data compression, encoding or data transformation is applied to obtain a reduced or compressed representation of the original data. Huffman encoding was a compression method that was initially used successfully for text compression. Huffman's idea was, instead of using fixed-length code such as ASCII or DBCDIC which was expanded 8 bits for each symbol, to represent characters that often appear in sources with shorter codewords and to represent less frequent ones with longer codewords[4].

This research focuses on using quaternary trees instead of binary trees to speed up the decoding time for Huffman code. It's usually difficult to strike a balance between speed and memory usage using variable length binary Huffman code. Quaternary trees are used here to generate optimal passwords that speed up the search method. This research has analysed the performance of algorithms that have been made with Huffman-based techniques in terms of decoding speed and compression ratio. This study shows that the proposed decoding algorithm outperforms the Huffman-based technique in terms of speed while compression performance remains almost the same[3].

The frequency of letters in the English text, which is dominated by a small number of common words. Meanwhile, the frequency of letters in English vocabulary, regardless of word frequency, is based on an analysis of the letters that appear in the words listed in the main entry of the Oxford Concise Dictionary [5] and produces the following Table 1.

Table 1 Frequency of letters in the main entries of the Concise Oxford English Dictionary, 11th ed

Letters	Frequency	Proportion
E	11.1607%	56.88
A	8.4966%	43.31
R	7.5809%	38.64
I	7.5448%	38.45
O	7.1635%	36.51
T	6.9509%	35.43
N	6.6544%	33.92
S	5.7351%	29.23
L	5.4893%	27.98
C	4.5388%	23.13
U	3.6308%	18.51
D	3.3844%	17.25
P	3.1671%	16.14
M	3.0129%	15.36
H	3.0034%	15.31
G	2.4705%	12.59
B	2.0720%	10.56
F	1.8121%	9.24
Y	1.7779%	9.06
W	1.2899%	6.57
K	1.1016%	5.61
V	1.0074%	5.13
X	0.2902%	1.48
Z	0.2722%	1.39
J	0.1965%	1.00
Q	0.1962%	(1)

The third column represents proportions, taking the least common letter (q) as equal to 1. The letter E is over 56 times more common than Q in forming individual English words. The frequency of letters at the beginnings of words is different again. There are more English words beginning with the letter 's' than with any other letter. (This is mainly because clusters such as 'sc', 'sh', 'sp', and 'st' act almost like independent letters.) The letter 'e' only comes about halfway down the order, and the letter 'x' unsurprisingly comes last.

The research is planned to be carried out in two stages, namely research 1 and 2, for Research 1 it is planned to be carried out in 2020. To conduct research 1 above a preliminary study has been carried out which is to study the calculator tools to measure the performance of the calculator, and the results that have been achieved are:
Calculating the Compression ratio of text in English Text is compressed as follows:
The Goose with the Golden Eggs
Once upon a time, a man and his wife had the good fortune to have a goose which laid a golden egg every day. Lucky though they were, they soon began to think they were not getting rich fast enough. They imagined that if the bird must be able to lay golden eggs, its insides must be made of gold. And they thought that if they could get all that precious metal at once, they would get mighty rich very soon. So, the man and his wife decided to kill the bird. However, upon cutting the goose open, they were shocked to find that its innards were like that of any other goose!
The test results obtained a Huffman symbol for each character as shown in Table 2.

Table 2 Test Huffman Symbols for Each Character

No.	Character	Value	Binary Symbol
1.	SPACE	110	00
2.	E	59	011
3.	I	29	0100
4.	R	15	01010
5.	Y	15	01011
6.	H	35	1001
7.	S	18	10100
8.	M	9	100010
9.	V	4	1000000
10.	ENTER	4	1000001
11.	O	38	1011
12.	W	9	100011
13.	F	9	101010
14.	C	11	101011
15.	.	5	1000011
16.	T	1	11100011110
17.	A	1	11100011111
18.	S	1	11100011100
19.	H	1	11100011101
20.	t	49	1101
21.	d	23	11001
22.	g	22	11000
23.	n	27	11110
24.	u	13	111010
25.	,	6	1110001
26.	k	5	1110000
27.	a	28	11111
28.	l	14	111011
29.	b	6	1110010
30.	p	4	1000010
31.	!	1	11100011010
32.	O	1	11100011011
33.	L	1	111001100

Based on the calculator, the initial number of bits is 4824 (603 characters) while after being compressed as many as 2438 bits, the Binary Symbol is obtained from the Huffman Tree as shown in Figure 3.

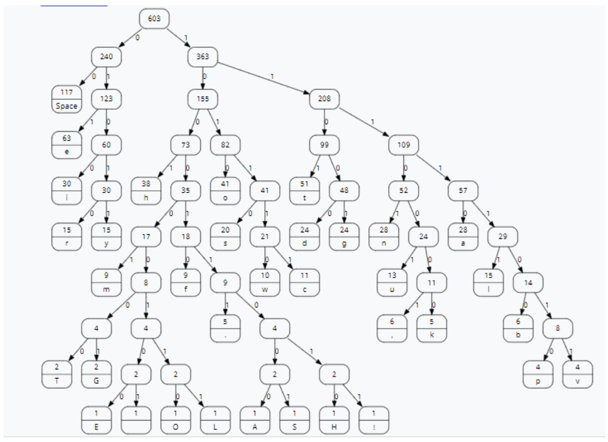

Figure 3 Obtained Huffman Tree

3. Methodology

Huffman coding is a lossless data compression algorithm. The idea is to assign variable-length codes to input characters, lengths of the assigned codes are based on the frequencies of corresponding characters. The most frequent character gets the smallest code and the least frequent character gets the largest code.

The variable-length codes assigned to input characters are Prefix Codes, means the codes (bit sequences) are assigned in such a way that the code assigned to one character is not the prefix of code assigned to any other character. This is how Huffman Coding makes sure that there is no ambiguity when decoding the generated bitstream.

Let us understand prefix codes with a counter example. Let there be four characters a, b, c and d, and their corresponding variable length codes be 00, 01, 0 and 1. This coding leads to ambiguity because code assigned to c is the prefix of codes assigned to a and b. If the compressed bit stream is 0001, the de-compressed output may be "cccd" or "ccb" or "acd" or "ab".

There are mainly two major parts in Huffman Coding
- Build a Huffman Tree from input characters.
- Traverse the Huffman Tree and assign codes to characters.

Steps to build Huffman Tree
Input is an array of unique characters along with their frequency of occurrences and output is Huffman Tree.

1. Create a leaf node for each unique character and build a min heap of all leaf nodes (Min Heap is used as a priority queue. The value of frequency field is used to compare two nodes in min heap. Initially, the least frequent character is at root)
2. Extract two nodes with the minimum frequency from the min heap.
3. Create a new internal node with a frequency equal to the sum of the two nodes frequencies. Make the first extracted node as its left child and the other extracted node as its right child. Add this node to the min heap.
4. Repeat steps#2 and #3 until the heap contains only one node. The remaining node is the root node and the tree is complete.

The output of innovation targeted in this research is the result of a feasibility study on the use of the Huffman Code Calculator in the achievement of learning about Data Compression, thus contributing to the superior scientific development of the Broadband Multimedia study program. This research examines the implementation of the data compression calculator work system, which is one of the tools to test one of the important issues in Multimedia computing and communication, and cannot be separated from the State of the art in the field of Information and Communication Technology, namely about Quality of Services (QoS). QoS on the Communication Network is a requirement that must be met to run an application on the Internet Network. With the fulfilment of this QoS information sharing on companies, Government Sites, and Science Groups can be done quickly and efficiently. This information can be in the form of documents or data folders, and data is shared and processed by various parties, and streams of multimedia files work well. QoS can be provided to users, including meeting large bandwidth requirements. One of the efforts to reduce the bandwidth requirement is by compressing the data. The problem is whether the language in the text affects the compression ratio, because the Huffman code is determined by the frequency of the appearance of the alphabet in the text.

4. Result and Discussion

In this study, a comparative analysis of text data compression in Indonesian and English was carried out. Are there any influence on the level of compression based on the language of the written text using the Huffman code calculator located on the web page [6][7]. Based on the research flow, the following results are obtained: Compression 100 words text file for English text: Translation history will soon be available when you are signed in and will be centrally managed, the compressed Huffman tree from 100 English words of text is shown in Figure 4.

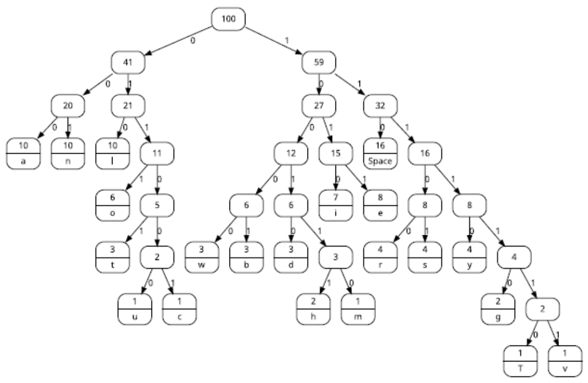

Figure 4 Huffman Tree for Text Compression 100 English words

Figure 4 shows the result of 100 characters of text in English. Where the results from the Huffman Generator for text with a total of 100 characters. It can be seen that the shortest code is letter a, 3 bit 000, while the longest T is 7 bits, 1111110. Next, the probability calculation is carried out, the results can be seen in Table 3, the number of characters is 100 text characters in above is,

Table 3 Probability of Appearing Characters in English Text

Character	Total Text	Probability (%)
T	1	1
Space	16	16
a	10	10
b	3	3
c	1	1
d	3	3
e	8	8
g	2	2
h	2	2
i	7	7
l	10	10
m	1	1
n	10	10
o	6	6
r	4	4
s	4	4
t	3	3
u	1	1
v	1	1
w	3	3
y	4	4
Total	100	100

After obtaining the Probability of each Character, then enter it in Online Calculator with the link: https://planetcalc.com/2480. After entering the probability value, the Shannon Entropy data is obtained and the compression ratio is calculated using the formula,

Compression Ratio = Total of Characters/Entropy (1)

Furthermore, the same method is carried out for English and Bahasa Indonesia texts with 200, 300, 400, 500 and 1000 characters and the data are presented in Table 4.

Table 4 Shannon Entropy and Compression Ratio

Total Characters	Shannon Entropy		Compression Ratio	
	English	Bahasa Indonesia	English	Bahasa Indonesia
100	3,82	4,65	1.83	1.51
200	4.38	4,4	1.59	1.60
300	4.42	4,31	1.58	1.62
400	4,50	4,27	1.56	1.64
500	4,49	4,45	1.56	1.57
1000	4,51	4,42	1.56	1.58
Average	4.35	4.42	1.61	1.59

If the graph of the number of characters is plotted against the Shannon Entropy value as shown in Figure 5. It can be seen that the Shannon Entropy value between English and Indonesian texts is not much different, especially with the increasing number of characters, the Shannon Entropy score is relatively similar, namely with the average value for English text of 4.35 and Indonesian text of 4 , 42. This shows that the language differences in the text files do not have a significant effect on the Shannon entropy value. As for the graph of the compression value for the number of characters, it can be seen in Figure 6. It can be seen that the values range between 1.51 and 1.64 for Bahasa Indonesia. Meanwhile, for English it ranges from 1.56 to 1.83.

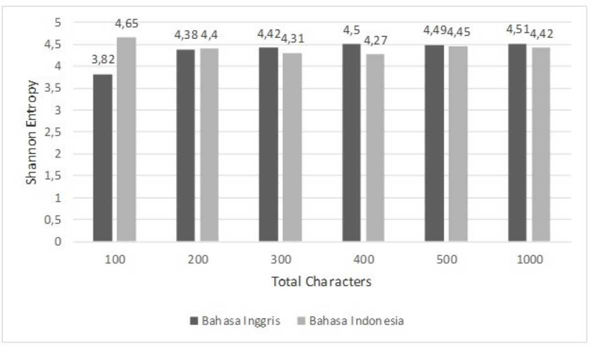

Figure 5 Graph of the number of characters in English and Bahasa Indonesia text against the Shannon Entropy value.

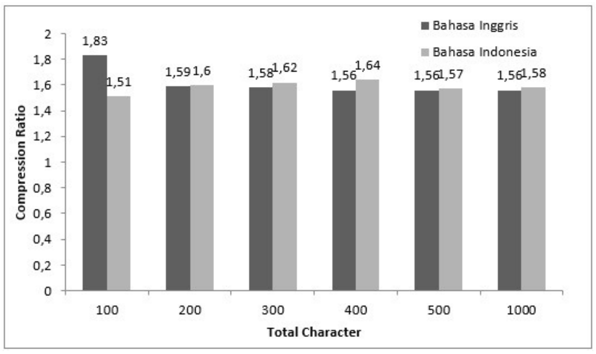

Figure 6 Graph of Compression Ratio Value to Number of Characters

5. Conclusion

Based on the discussion of the data obtained, it is concluded that this calculator can be used for practical facilities, but it still works manually, it is necessary to continue further research so that it works faster by developing its application software. Meanwhile, the calculation results show that the value of entropy and data compression is not much different between English and Bahasa Indonesia texts, namely the average entropy value in English is 4.35 while Bahasa Indonesia is 4.42. While the average Compression ratio for English is 1.61, Bahasa Indonesia is 1.59. From the results of this study, it can be seen that this calculator can be used for online data compression practice, where data is compressed between English and Bahasa Indonesia, the value is almost close. This proves that language only affects the frequency of the appearance of characters in text, although the data processing still needs to be improved so that manual work can be developed using software applications.

Acknowledgments

This research was funded by the Politeknik Negeri Jakarta through the *Penelitian Penugasan* scheme with funds from PNPB.

References

[1] D. HUFFMAN, "Huffman_1952_Minimum-Redundancy-Codes," *Proc. I.R.E.*, pp. 1–4, 2002, [Online]. Available: papers2://publication/uuid/8EC008F4-3BE7-42B5-B30D-36AECD1DD81F.

[2] A. Shahbahrami, R. Bahrampour, M. S. Rostami, and Mostafa Ayoubi, "Evaluation of Huffman and Arithmetic Algorithms for Multimedia Compression Standards," *Int. J. Comput. Sci. Eng. Appl.*, vol. 1, no. 4, pp. 34–47, 2011, doi: 10.5121/ijcsea.2011.1404.

[3] A. Habib, M. Jahirul Islam, and M. Shahidur Rahman, "Huffman based code generation algorithms: Data compression perspectives," *J. Comput. Sci.*, vol. 14, no. 12, pp. 1599–1610, 2018, doi: 10.3844/jcssp.2018.1599.1610.

[4] D. J. Shoba and S. Sivakumar, "A Study on Data Compression Using Huffman Coding Algorithms," vol. 5, no. 1, pp. 58–63, 2017.

[5] Oxford University Press, "Which letters in the alphabet are used most often?," 2014. https://www.lexico.com/explore/which-letters-are-used-most.

[6] C. S. E. R. G. University of Cantenbury, "Huffman Tree Generator." https://csfieldguide.org.nz/en/interactives/huffman-tree/.

[7] PlanetCalc, "Huffman Coding," 2020. https://planetcalc.com/2481/.