CODING AND TRANSMISSION OF NON UNIFORM ALPHABETS

Yohannes Kassahun and Hailu Ayele
Electrical Engineering Department
Addis Ababa University

ABSTRACT

Algorithms for Huffman encoding and decoding that would minimize the encoding and decoding times and also increase the transmission rate are developed and tested. The algorithms are employed for compression and decompression of text and image files and real time transmission of text files. It is demonstrated that the encoding produces average code lengths that are nearly equal to the theoretical limit represented by the entropy of the source and hence resulting in improved compression ratio and transmission rate.

INTRODUCTION

Various schemes of encoding of data have been developed over the years to facilitate improved transmission or storage in memory devices. [1, 2, 3]. When encoding is used for transmission purposes the main objective is to attain an error free transmission and where errors occur to be able to detect and correct them. Encoding is also employed to provide security so that unauthorized persons may not be able to read the data. Whatever the objectives of the encoding schemes are, it would be worthwhile, both in terms of saving transmission time or storage space, for the encoded data to be as small as possible so that better efficiency of transmission and storage is achieved

Information theory provides powerful schemes, through the use of discrete probability theory, for solving the problem of encoding for efficiency. [2,3]. There are two types of encoding that are frequently employed: Fixed Length encoding and Variable Length encoding. The Fixed Length encoding scheme produces code words of the same length for every symbols of a given source alphabet irrespective of the frequency of occurrence of these symbols in a given text written using the symbols of this alphabet. This type of coding is generally applied for error detection and correction. On the other, the variable Length encoding scheme produces code words of varying length for the symbols of the source alphabet such that most frequently occurring symbols are encoded with the shortest code words.

Data compression is best accomplished by the *Variable Length encoding* scheme. This scheme minimizes the average length of the encoded data. The source data may be encoded either using a single symbol at a time or a set of predefined blocks of symbols. In both cases what is important is not the actual symbols themselves but the probability distribution. For every probability distribution there is an associated quantity called *entropy*, which is a measure of the total "information" in the source. The goal of any encoding scheme is, therefore, to add as little additional "information" as possible beyond that given by the entropy of the source.

Among the *Variable Length encoding* methods, the Huffman encoding is a highly efficient instantaneous encoding scheme. [1, 2, 3]. Even though, Huffman coding results in highly efficient codes, the encoding and decoding processes can be lengthy and complex. In this work two algorithms are developed to implement Huffman encoding. The general schemes of both algorithms are basically similar and both have the same compression ratio when applied to a particular data set off-line. The first algorithm is not as "efficient" for coding and decoding as the second one as it is not optimized for speed and can only be applied for off-line data compression. The speed at which it compresses data is much less than the speed of the second algorithm.

The second algorithm is optimized for speed such that the time required by the computer to generate the Huffman codes, compress the data with the generated codes and transmit the data is minimized. The generation of the Huffman codes, compression and transmission are optimized at every stage by using computationally speed efficient algorithms. Only the results of this algorithm are presented in this paper.

Information Theory and Coding

Consider a source alphabet of q symbols $s_1, s_2, \ldots, s_q$ each with probability $p(s_1) = p_1, p(s_2) = p_2, \ldots, p(s_q) = p_q$. Assuming that the symbols in a given text occur independently, the average information over the alphabet is given by [1,2,3]

$$H_2(S) = -\sum_{i=1}^{q} p_i \log_2 \frac{1}{p_i} \qquad (1)$$

$H_2(S)$ is the entropy of the signaling system S having symbols s_i with corresponding probability p_i. The entropy function measures the amount of uncertainty, surprise or information that one gets from the outcome of a given situation such as the reception of a message or the outcome of a given experiment. It is therefore a function of the probabilities of the individual outcomes or events.

When encoding a source S, it certainly seems reasonable to expect at least as many bits of information in the encoding as there are in the source. However, additional bits may be required if the encoding scheme has to result in a given level of compression efficiency. Since the entropy of S measures the amount of information in S, it can be shown that the minimum average code word length of any form of encoding of S should be as large as the entropy of S. [1,2] This is the essence of the *Noiseless Coding theorem*, which is usually stated as *if S is an information source, then its entropy H (S) is the minimum average code word length among all the uniquely decodable encoding schemes of S.* [1,2]

Variable Length Code Words

A variable code is uniquely decodable if for each source sequence of finite length, the sequence of the code of symbols corresponding to that source sequence is different from the sequence of code symbols corresponding to any other source sequence.

A prefix condition code is a variable code in which no code word is a prefix of any other code. Prefix condition codes are distinguished from the other uniquely decodable codes by the fact that the end of a code word is always recognizable so that decoding can be accomplished without a delay for observing subsequent code words. For this reason, prefix condition codes are called *instantaneous* codes. [1,2]

Kraft Inequality [1,3]

Consider a code having code word lengths $n_1, n_2, \ldots, n_k$ and D symbols in the code alphabet. A prefix condition of alphabet size D exists with the integers $n_1, n_2, \ldots, n_k$ as code word lengths provided the following inequality, known as Kraft inequality, is satisfied.

$$\sum_{k=1}^{K} D^{-nk} \leq 1 \qquad (2)$$

In other words, if a code is to be uniquily decodable then the Kraft inequality must be satisfied.

A Source Coding Theorem [1,2,3]

Given a finite source symbols S with entropy $H(S)$ and a code alphabet of D symbols, it is possible to assign code words to the source symbols in such a way that the prefix condition is satisfied and the average length of the code words, $\bar{n}$, satisfies the inequality

$$n < \frac{H(S)}{logD} + 1 \qquad (3)$$

Furthermore, for any uniqnely decodable set of code words

$$\bar{n} \geq \frac{H(S)}{logD} \qquad (4)$$

Alternately, it is also possible to asssign code words to sequences of L source symbols in such a way that the prefix condition is satisfied and the average lenghth of the code words per source symbols, $\bar{n}$, satisfies the following inequality

$$\frac{H(S)}{logD} \leq \bar{n} \leq \frac{H(S)}{logD} + \frac{1}{L} \qquad (5)$$

If L is taken to be arbitrarily large, it can be shown, applying the law of large numbers to a long string of sequences each of L source symbols, that equations (3) and (4) are special cases of equation (5).

Variable Length Encoding Procedure

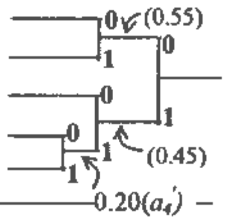
D.A. Huffman proposed what is now commonly referred to as the Huffman Coding Scheme which produces optimum codes. The codes are optimum in the sense that no other uniquely decodahle set of code words have smaller average code word length than this set.

For any given source with K source symbols ($K \geq 2$) , an optimum binary code exists in which the two least likely code words, x_k and x_{k-1}, have the same length and differ in only the last digit, . x_k ending in 1 and x_{k-1} in 0. [1]

A systematic procedure for carrying out the above operation is demonstarted in Table 1. First, the two least probable messages, a_4, and a_5 in this case are tied together, assigning 0 as the last digit for a_4 and 1 as the last digit for a_5. At this point, the two least likely messages are a_3 and a_4'. On the next stage, the two least likely messages are a_1 and a_2 and here after only two messages remain. Looking at the resulting figure, it can be seen that a code tree has been constructed for a sourse S, starting at the outermost branches and working down to the trunk. The code words are read off the tree from right to left.

Table 1: An Example of Huffman Encoding procedure.

Code Word	Message	$P(a_k)$
00	a_1	0.3
01	a_2	0.25
10	a_3	0.25
110	a_4	0.10
111	a_5	0.10



Compression [4]

Data compression may be expressed either in terms of compression ratio, or compression efficeincy. An original data stream processed according to one or more compression algorithms results in a more compression algorithms results in a compressed data stream and the efficiency of compression can be measured by the compresion ratio

$$\text{Compression ratio} = \frac{\text{Size of original data}}{\text{Size of compressed data}} \quad (6)$$

The compression efficiency, which is one minus the reciprocal of compression ratio, is also used to measure the degree of reduction of the size of the original data by the compression process.

Compression methods are generally classified into two groups: Lossless and Lossy compression techniques.

Lossless compression techniques are fully reproducible and are primarily restricted to operations on data. This technique is usually applied in areas like data consisting of accounts receivable, payroll and health insurance claims and so on.

Lossy compression techniques may or may not be fully reproducble and are primarily restricted to operations on images, video and audio. Although the result of decompression may not provide an exact reproduction of the original data, the difference between the original and reconstructed may be so minor as to be not discernable. The compression ratio obtained from Lossy compression can significantly exceed the compression ratio obtained by the use of lossless compression.

Data Compression and Information Transfer [4]

Data transferred between terminals encounters several delay factors which cummulatively affect the information transfer rate. Data transmission over a transmission medium must be converted into an acceptable format for that medium. When digital data is transmitted over analog lines, modems are employed to convert the digital signal into a modulated signal acceptable for transmission on the analog circuit. The time between the first bit entering modem and the first modulated signal produced by the device is the modem's internal delay time. [6]

Another source of delay is the processing time at the reciever end. Once data is received at the distant end it must be acted upon, resulting in a propagation delay which is a function of the computer or terminal employed as well as the quantity of received data.

Each time the direction of transmission changes in a typical half duplex protocol, control signals change at the associated modem to computer and modem to terminal interfaces. The time required to switch the control signals to change the direction of transmission is the line turn around time.

If real time compression and decompression are employed on both the transmitter and receiver sides, there is an additional encoding and decoding delay on the transimission and reception of the data. All these delays have a major effect on information transfer rate.

The information transfer rate (ITR) increases in proportion to the compression ratio achieved by the encoding sceme. Accordingly a compression ratio of *2* will result in doubling the information transfer rate.

The effective information transfer rate (*EITR*) can be obtained by the following formula

$$EITR = \frac{B_{IC} D_1 (1-P)}{T_R \left[T_C (D_2 + C) + 2(T_{PA} + T_L + T_P) + \frac{A B_C}{T_R} \right]} \qquad (7)$$

Where

D_1	=	Origial data block size in characters perior to compression
D_2	=	Compressed data block size in characters to inlude special compression indication characters
B_{IC}	=	Information bits per character
B_C	=	Total bits per character
A	=	Characters in the acknowledgement message.
C	=	Control characters per message blocks
T_R	=	Data transfer rate(bps)
T_C	=	Transmission time per character
T_{PA}	=	Processing and acknowledgment time
T_L	=	Line turn around time
T_P	=	Propagation delay time
P	=	Probability of one or more errors in block

The effective information transfer ratio can be approximated by

$$EITR = ITR * CR \qquad (8)$$

Where *ITR* is information transfer rate and *CR* is compression ratio. Eqs. (7) and (8) are used for a communication medium with full duplex model.

If a *'stop and wait ARQ'* error control procedure is applied where there is a return channel available for the transmission of acknowledgments, then the transfer ratio becomes

$$ITR = \frac{B_{IC} D_1 (1-P)}{T_R \left[T_C (D_2 + C) + 2(T_{PA} + T_P) + \frac{A B_C}{T_S} \right]} \qquad (9)$$

Since an error free line is not something a transmission engineer can reasonably expect, a maximum block size will exist beyond which our line efficiency will decrease. At this point, only data compression will result in additional transmision efficiencies. In addition from a physical standpoint, the buffer area of some devices may prohibit block sizes exceeding a certain number of characters. So data compression can become an effective mechanisim for increasing transmission effeciency while keeping data buffer requirments to an acceptable size.

Encoder and Decoder Cost for Huffman Codes [5]

There are various implemntations for Huffman Codes. For all of them whether they are implemented in hardware or software, there is a cost incurred in terms of storage space. Although no attempt was made to estimate the actual cost for the different kind of implementations in this work, the major cost components are the following.

Assumng restricted length Huffman codes, the encoder requires the following storages:

- A look up table for the code words. For a code with N code words, this table requires N times the size of the longest code in bytes.
- A look up table for the lengths of each code. For a code with N code words, this table requires N Bytes. (Assuming that the length of the longest code is not greatert than 255).

- Storage for code tree. Even though it depends on the type of implementation, the maximum size of the code tree description is twice that of code words.

- A storage for a heap where sorting is mainly performed. This is the same as that of the code words.

On the decoder side the following storage is required:

- A look up table for code words

- A look up table for code word lengths

- Storage for code tree.

For large N the look up table will become predominant in both the encoder and decoder. The major problem is selecting and/or minimizing the code words. This in turn leads to the need for selecting appropraite number of symbols. So if one selects an optimal number of symbols, then it is possible to reduce the cost of implementaion

Moreover, the algorithms used in encoding and decoding have a major influence on the cost. Some algorithms require a lot of hardware and/ or software commnads while others require mimimum hardware and/or software commands. The amount of hardware or software required to implement these algorithms will add to the cost.

Another cost is the encoding and decoding speed. However, not all applications require speedy decoding. This is case in communication applications where the data processing bandwidth at the transmitter and receiver is much larger than the bandwidth of the communication channel

RESULTS

The two algorithms mentioned above are implemented and tested for file compression and real time text transmission over a telephone line.

a. Result from compression of word processor documents

A sample result for the algorithm that has beeen optimized for speed as applied to a word processer files is shown in Fig. 1.

It can be easily seen that the average percentage savings obtained for word processor documents is around 50 percent. Various block sizes were used to investigate the effect of block size on the compression efficiency of the encoding scheme.

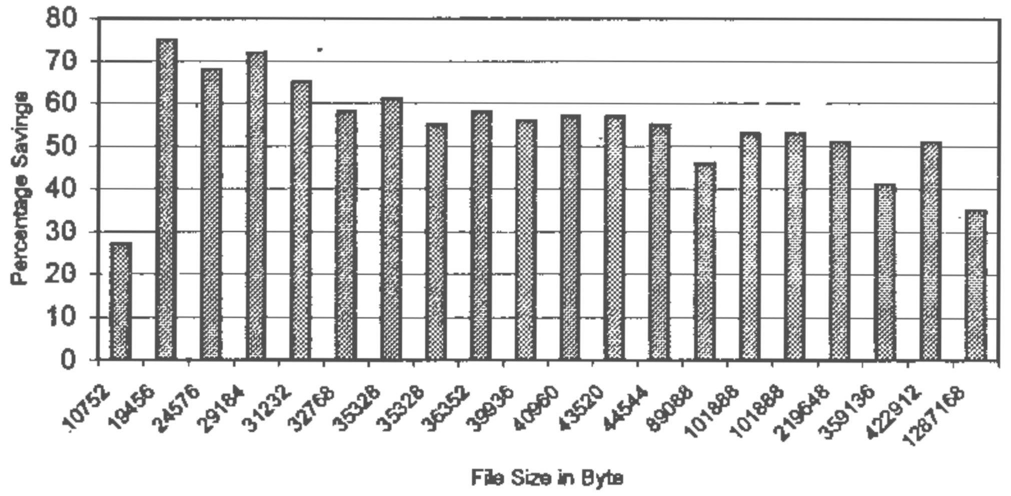


Figure 1 Bar graph obtained for word processor documents

The result indicates that the optimal block size is around 3.5 Kbytes.

For block sizes below 3.5 Kbytes, the frequency of occurences of the individual source symbols may not on the average correspond to the assumed standard frequency distribution of the symbols. This will result in sub- optimal codes. For block sizes above 3.5 Kbytes, it appears that the frequency distribution of the symbols tend to equalize.

in the program. The first code is generated based on the standard frequency distribution of English alphabets and the second is generated by studying

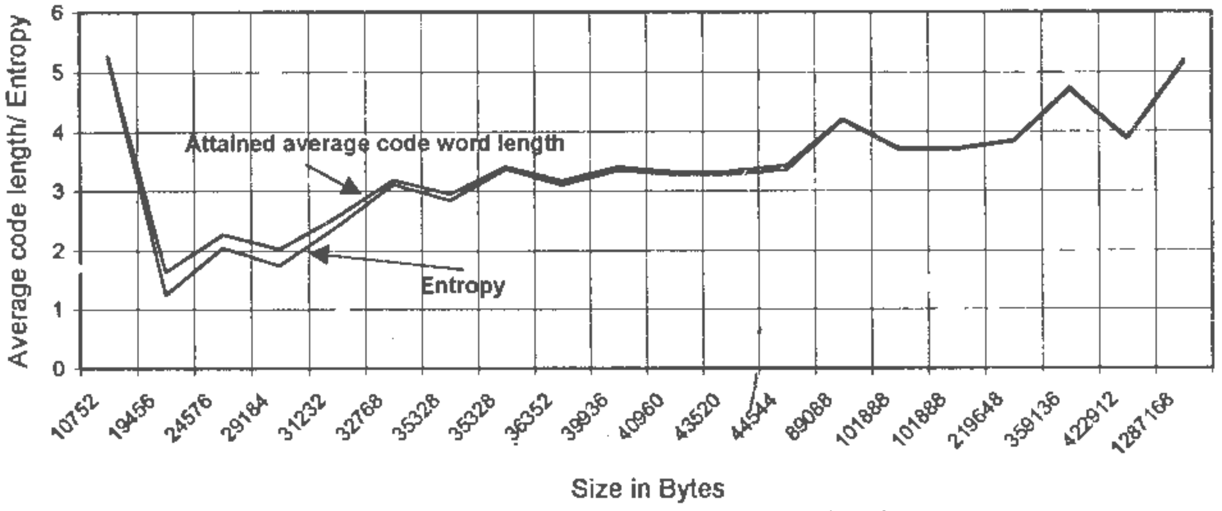


Figure 2 Comparison between the attained average code length and Entropy

The graph in figure 2 shows a comparison between the average code word length obtained through encoding using the second algorithm and the entropy of a given set of files of varying sizes.

b. Real Time Text Transmission over Telephone Line

A communication program is developed and tested for different text files for a modem using a transmission rate of 9600 bps. Two codes are used different English text samples assuming the optimal block size fo 3.5 Kbytes.

As can easily be seen from the graph the information transfer rate has been increased from 9600 bps to an average speed of 15000 bps. This shows an increase in the transmission rate by about 5000 bps. This in other words means that the time needed to transmit uncompressed text document is reduced by 36 percent if one uses this compression technique.

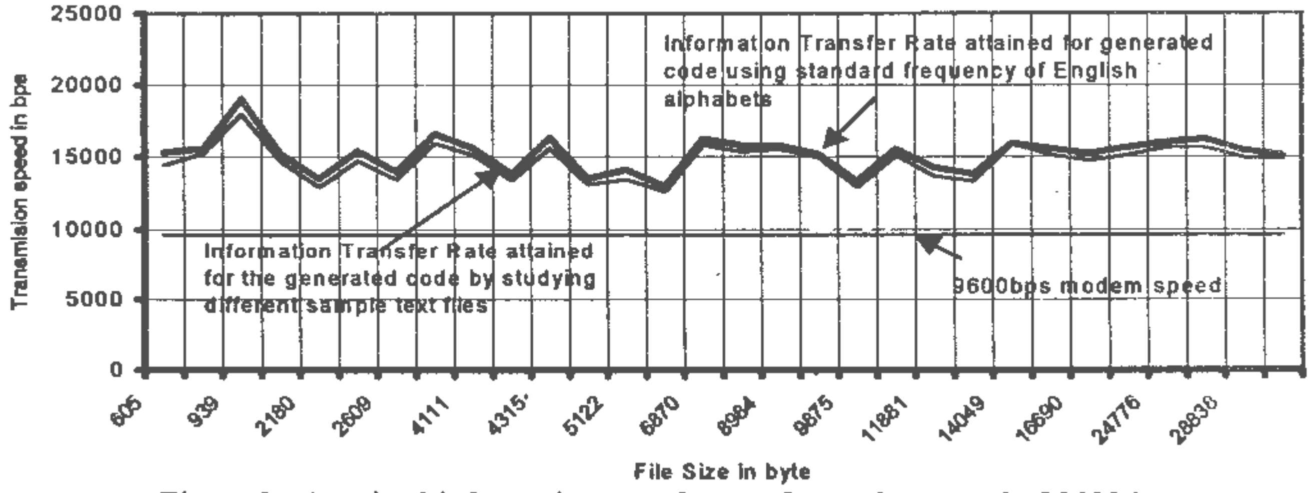


Figure 3. Atttained information transfer rate for modem speed of 9600 bps.

CONCLUSIONS

The average code word length obtained is very near the entropy of a given source supporting the assertion that the codes generated by the encoding scheme are optimal.

The second algorithm is also found to be fast enough that one can apply it for real time communication applications. This algorithm is tested for serial communication upto a maximum speed of 128 Kbps.

The algorithms that were developed can also be used to test the compression efficiency of other compression algorithms. The is true because codes generated by the Huffman coding schemes are optimal that a source compressed with any other algorithm can further be compressed by the these algorithms. The efficeincy of the compression algorithm to be tested is measured by the compression efficency achieved by compressing the source data compressed by the algorithm to be tested. Larger compression ratio means lower efficiency and lower compression ratio means larger or higher efficency of the algorithm to be tested. Table 2 shows the result obtained on two standard video compression and decompression codecs. As can be seen from table 2 *Cinpack Codec* is more efficient than the *Microsoft Vidoe 1 Codec* in terms of compression ratio.

It is recommended that one has to use Huffman encoding as last step of compression because every other type of compression algorithm will not produce an optimal code for a given source.

Applying adaptive Huffman coding can further expand this work. This type of coding scheme uses Markov process to model the soures. If the source is ergodic, the Markov structure of the source can be used to improve the encoding of the source. For each state in the Markov system, one can use appropriate Huffman code obtained from the corresponding transition probabililties for leaving that state. Depending on how variable are the probabilities for each state, the encoding will gain a lot or not.

It would be interesting to develop an optimal code for transmisssion of Amharic texts using the technique discussed above. This would obviously require the development of a standard frequency distribution for the Amharic characters.

Table 2: Comparison between two standard video codecs.

	Original Bitmap Image Size in byte	*Compression achieved by each encoder*	*Percentage saving on the compressed file after Huffman encoding is applied*
Microsoft Video 1 Codec	1,179,702	113768	11.46%
Cinpack Codec Version 1.10	1,179,702	50476	2.73%

REFERENCES

[1] Robert G. Galler, *Informotion Theory and Relaible Communication*, John Wiley & Sones, 1968.

[2] Steven Roman, *Introduction to Coding and Informotion Theory*, Springer, 1997.

[3] Richard W. Hamming, *Coding and Information Theory*, Printce-Hall, Inc., Englewood, N.J. 07632, 1980.

[4] Gilbert Held, *Data and Image Compression Tools and Techiniques*, John Wilkey & Sons, 1996.

[5] Ulrich Gunthere and Radu Nicolescu, *A Low Cost Decocder for Arbitrary Binary Varaiable Length Codes*, Compter Science Department of the University of Auckland CITR at Tamaki Campus, September 1997

[6] Robert L. Hummel, *Data and Fax Communications*, Ziff-Davis ZD Press