RESEARCH ARTICLE OPEN ACCESS

Performance Improvement Of Bengali Text Compression Using Transliteration And Huffman Principle

Md. Mamun Hossain[1], Ahsan Habib[2], Mohammad Shahidur Rahman[3]
[1]*Computer Science and Engineering, Bangladesh Army University of Science and Technology, Saidpur*
[2]*Computer Science and Engineering, Shahjalal University of Science and Technology, Sylhet, Bangladesh*
[3]*Computer Science and Engineering, Shahjalal University of Science and Technology, Sylhet, Bangladesh*

ABSTRACT
In this paper, we propose a new compression technique based on transliteration of Bengali text to English. Compared to Bengali, English is a less symbolic language. Thus transliteration of Bengali text to English reduces the number of characters to be coded. Huffman coding is well known for producing optimal compression. When Huffman principal is applied on transliterated text significant performance improvement is achieved in terms of decoding speed and space requirement compared to Unicode compression.

Keywords: Data compression; ASCII code; UNICODE; Huffman principle; Avro; Bengali text; English Text; Transliteration.

I. INTRODUCTION

Data compression is an important and essential research area in computer science. For faster transmission of information, text compression has become popular research area in recent years. The text compression algorithms can be broadly classified into two categories. The first category includes the dictionary based compression algorithms. These algorithms are generally of the Ziv-Lempel type and they replace a string with a pointer to an earlier occurrence of the same string. The second category includes the statistical compression algorithms. These algorithms are in general based on Huffman or arithmetic coding where they exploit the uneven frequency distribution of symbols, especially the dependence of symbols on their neighboring context [1, 2].

Most of natural language text compressors use general purpose data compression techniques and perform compression at the character level [3]. Some of the text compressors are word based that utlize words as the basic units for performing compression. The alphabet of some natural languages contains more symbols (e.g. Bengali, Chinese) than others (e.g. English, Arabic). The repetition of some symbols in same text will increase if more symbolic language is represented with less symbolic language. A number of works have been reported on compressing Bengali text. Islam and Rajon emphasized on designing a corpus for evaluation of Dictionary Based Bengali Text Compression Schemes [4]. Arif et al. worked on static data compression technique [5]. They attempted to balance between compression and decoding speed using static Huffman Coding for short message. In [6], the authors proposed a static Huffman coding system for different symbol of Bengali Text which published in 1990 before the Unicode Consortium was incorporated on January 3, 1991. A Huffman header is proposed in [7,8] using static Huffman coding. A dictionary based database compression technique is explained in [9] using variable length Huffman coding. We proposed a transliteration based compression technique using dynamic Huffman coding for achieving better performance. The work presented in this paper is based on the idea that Huffman principle could be used on transliterated text to achieve high compression ratios. This new approach may be used to improve the traditional compression technique [10, 11]. Experimental result shows that proposed technique achieves significant improvement in compression about 30% compared to Unicode, ASCII code and regular Huffman encoding.

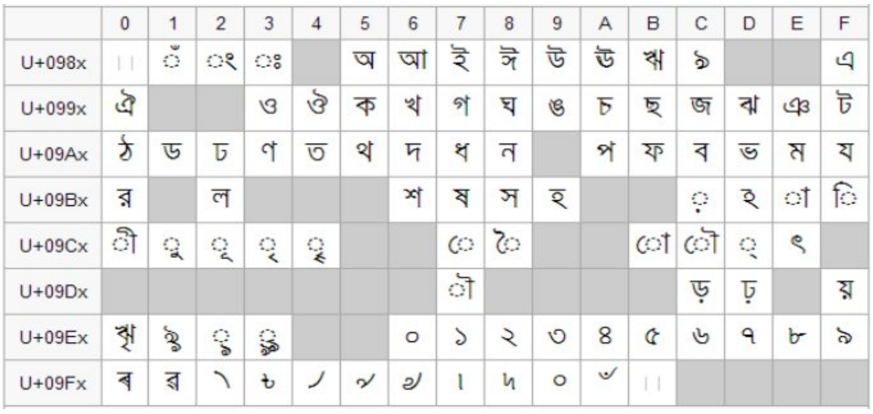
II. BACKGROUND STUDY

When we encode an English character in computer, an 8-bit ASCII code is assigned. Usually characters are repeated in same file. It therefore makes sense to assign shorter codes for more repeated characters [12, 13]. To encode Bengali characters, we assign each character a 16-bit code based on UNICODE chart which is double in size compared to English characters. To achieve better compression we may transliterate Bengali text to English and apply Huffman principle on transliterated text.

A. Structure of Bengali Alphabet

The Bengali alphabet is composed of 39 consonants, those are ক খ গ ঘ ঙ চ ছ য ঝ ঞ ট ঠ ড ঢ ণ ট থ দ ধ ন প ফ ব ভ ম য র ল শ ষ স হ ড় ঢ় য় ৎ ং ঃ

ঁ which are called “byanjonborrno” in Bengali, 11 vowels অ আ ই ঈ উ ঊ ঋ এ ঐ ও ঔ which are called “sorborno”, 10 vowel modifiers া ি ী ু ূ ৃ ?ে ?ৈ ?া ?ৌ which are called “kar” and 5 consonant modifiers ব - ব ফলা, - য ফলা, - র ফলা, - রেফ ও - হস? which are called “fola”. There are also some join or conjunct characters in Bengali alphabet ? ? ? ? ? ? ? ? ? ? ? ?.



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
U+098x		◌ঁ	◌ং	◌ঃ		অ	আ	ই	ঈ	উ	ঊ	ঋ	ঌ			এ
U+099x	ঐ			ও	ঔ	ক	খ	গ	ঘ	ঙ	চ	ছ	জ	ঝ	ঞ	ট
U+09Ax	ঠ	ড	ঢ	ণ	ত	থ	দ	ধ	ন		প	ফ	ব	ভ	ম	য
U+09Bx	র		ল				শ	ষ	স	হ			◌়	ঽ	◌া	◌ি
U+09Cx	◌ী	◌ু	◌ূ	◌ৃ	◌ৄ			◌ে	◌ৈ			◌ো	◌ৌ	◌্	ৎ	
U+09Dx								◌ৗ					ড়	ঢ়		য়
U+09Ex	ৠ	ৡ	◌ৢ	◌ৣ			০	১	২	৩	৪	৫	৬	৭	৮	৯
U+09Fx	ৰ	ৱ	৲	৳	৴	৵	৶	৷	৸	৹	৺					

Figure 1. The UNICODE Bengali Characters chart

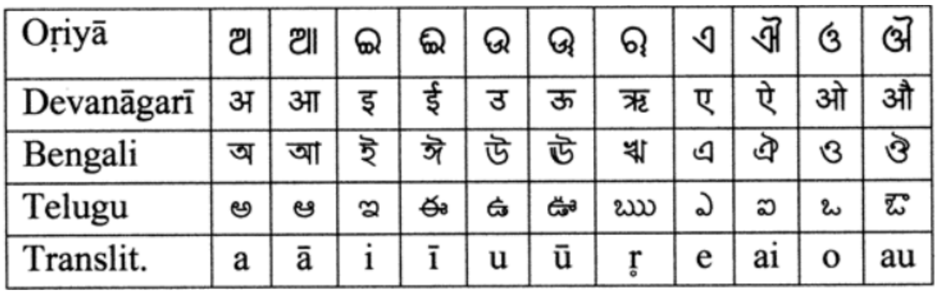
A detail list of Bengali symbols is available in [14]. In Figure 1, it is observed that there are some numeric characters 0-9 (০ ১ ২ ৩ ৪ ৫ ৬ ৭ ৮ ৯), Bengali currency mark (Taka sign), i.e.৳ and some less used characters, i.e. ঌ ঽ ৠ ৡ৲৺. It should be noted that there is no case sensitivity in Bengali language.

B. Structure of English Alphabet

The English alphabet is composed of 21 consonants and 5 vowels. English language is case sensitive. For upper and lower case, they have different value in ASCII character set.

C. Transliteration

Transliteration is the conversion of a text from one script to another. Transliteration is not concerned with representing the phonemics of the original: it only strives to represent the characters accurately which is Graphemic conversion. Transliteration is also different than translation. From an information-theoretical point of view, systematic transliteration is a mapping from one system of writing into another, word by word, or ideally letter by letter. Most transliteration systems are one-to-one, so a reader who knows the system can reconstruct the original spelling [15]. Figure 2 shows transliteration of some vowels Graphemes from one language to another. When Transliterating (decoding) from English to Bengali, ‘ph’ will transliterated as ‘ফ’ and ‘poh’ will transliterate ‘পহ’. If we place two consonant together it will make a conjunct in Bengali (kt as ?). If we want two consonant separately we need to append an extra ‘o’ (call null modifier, kot as কত).



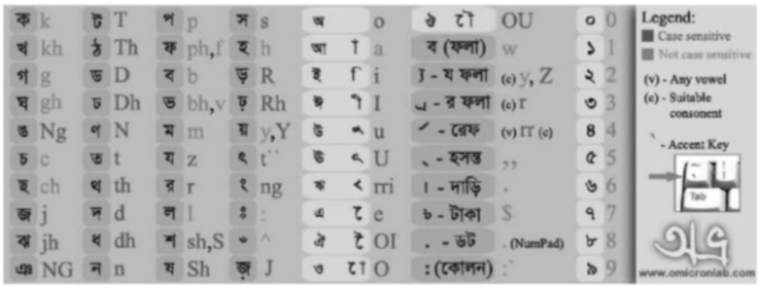
Oriyā	ଅ	ଆ	ଇ	ଈ	ଉ	ଊ	ଋ	ଏ	ଐ	ଓ	ଔ
Devanāgarī	अ	आ	इ	ई	उ	ऊ	ऋ	ए	ऐ	ओ	औ
Bengali	অ	আ	ই	ঈ	উ	ঊ	ঋ	এ	ঐ	ও	ঔ
Telugu	అ	ఆ	ఇ	ఈ	ఉ	ఊ	ఋ	ఎ	ఐ	ఒ	ఔ
Translit.	a	ā	i	ī	u	ū	r̥	e	ai	o	au

Figure 2. Transliteration of vowels Graphemes

III. DESCRIPTION OF THE PROPOSED METHOD

D. Mathematical Analysis

For transliterating Bengali to English, a Bengali alphabet can be transliterated using a uppercase or lowercase or conjunct of English characters. To express 65 Bengali symbols we need only 39 English. If we transliterate Bengali to English the 39 English character will be repeated and we achieve a better performance. The transliteration between Bengali and English symbols is shown in Figure 3.



ক	k	ট	T	প	p	স	s	অ	o	ঔ ৌ	OU	০	0
খ	kh	ঠ	Th	ফ	ph,f	হ	h	আ া	a	ব (ফলা)	w	১	1
গ	g	ড	D	ব	b	ড়	R	ই ি	i	্য - য ফলা	(c) y, Z	২	2
ঘ	gh	ঢ	Dh	ভ	bh,v	ঢ়	Rh	ঈ ী	I	্র - র ফলা	(c) r	৩	3
ঙ	Ng	ণ	N	ম	m	য়	y,Y	উ ু	u	র্ - রেফ	(v) rr (c)	৪	4
চ	c	ত	t	য	z	ৎ	t``	ঊ ূ	U	্ - হসন্ত	,,	৫	5
ছ	ch	থ	th	র	r	ং	ng	ঋ ৃ	rri	। - দাড়ি	.	৬	6
জ	j	দ	d	ল	l	ঃ	:	এ ে	e	৳ - টাকা	$	৭	7
ঝ	jh	ধ	dh	শ	sh,S	ঁ	^	ঐ ৈ	OI	. - ডট	. (NumPad)	৮	8
ঞ	NG	ন	n	ষ	Sh	জ়	J	ও ো	O	: (কোলন)	:`	৯	9

Legend:
▨ Case sensitive
▨ Not case sensitive
(v) - Any vowel
(c) - Suitable consonent
` - Accent Key
www.omicronlab.com

Figure 3. Avro Phonetic layout

A.1. The length (height) of Huffman tree

With 39 English symbols, the length of the Huffman tree will be shorter than the length with 65 Bengali symbols. For Bengali alphabet, to represent 65 Bengali symbols, the bit requirement can be calculated as following way:

Total number of node, $N_T = N_I + N_E$ = 64+65 =129; where, N_E= Number of External node=65; N_I= Number of Internal node =N_E -1 =65-1=64.

The depth or height of the tree can be calculated by the following equation D_n = Floor (Log2 n +1) = Floor (Log2 129 +1) = Floor (7.01+1) = 8; where n is the total number of nodes [16].

Numbers of bit required to represent each symbol = Largest level number = Depth -1=8-1=7

To represent 65 symbol maximum number of bits require for Bengali, N_B =65*7= 455

Whereas English alphabet, to represent the same 65 Bengali symbols we need only 39 English Symbols. The bit requirement can be calculated follows:

Total no. of node, $N_T = N_I + N_E$ = 38+39 =77; where N_E is number of External node and N_I is number of Internal node.

The depth or height of the tree can be calculated by the following equation: D_n = Floor (Log_2 n +1) = Floor (Log_2 77 +1) = Floor (6.27+1) = 7.

Maximum number of bit required to represent each symbol = Largest level number = Depth -1=7-1=6

To represent 39 symbol total number of bits required for English, N_E= 39*6= 234

The compression ratio (r) can be calculated as, r = F (N_B, N_E) = ((N_B - N_E)/ N_B)*100%

=((Number of bits required to represent Bengali Text, N_B - No. of bits require to represent English Text, N_E)/ No. of bits require to represent Bengali Text, N_B))*100% = ((455-234)/455)*100% = 48.57%

A.2. Further Compression Using Huffman Principle

With 39 in English symbols instead of 65 Bengali symbols, a considerable variation will occur within the Huffman tree. The Huffman tree will converse to extended binary tree or 2-tree from complete binary tree and the weighted path length will be minimum. The weighted path length of the tree can be calculated using the formula, P = W_1L_1 + W_1L_1 + … … … W_nL_n; P=weighted path length, W and L denote the weight and the path length respectively [16].

In Figure 4, the weighted path length P_1 of the tree T_1, P_{1T1} = 2*2+ 2*2 +2*2+ 2*2 =16 whereas the weighted path length of P_2 of the tree T_2, P_{2T2} = 1*3+1 *3+2 *2+1*4 =14 and the weighted path of P_3 of the tree T_3, P_{3T3}= 1*3+1 *3+1 *2+5*1 =13.

A.3.Conjunct Letters in Bengali

Almost every Bengali sentence contains one or more conjuncts. Whenever we join two or more Bengali letters we need an “hosont” (- হস?). ? is represented by only two symbols, k and t in English. So to represent every joint word in Bengali we need one more symbol than English which increase the bit requirement but positive effect in transliterated English text.

***A.4.* Frequent symbols in Bengali texts**

There are only 17 Bengali symbols that required two or more English symbols during transliteration.

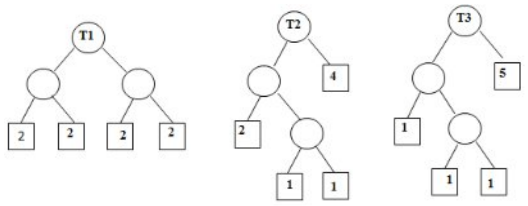
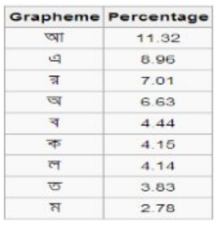


Figure 4. Variation in Huffman Tree



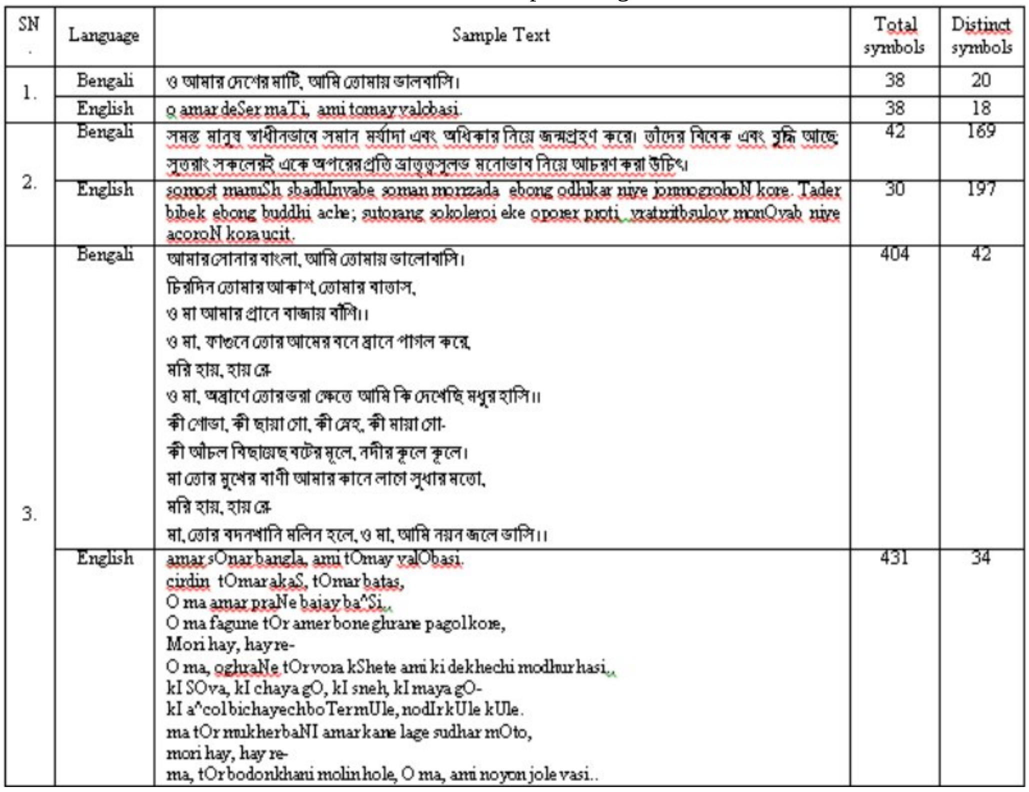
Grapheme	Percentage
আ	11.32
এ	8.96
র	7.01
অ	6.63
ব	4.44
ক	4.15
ল	4.14
ত	3.83
ন	2.78

Figure 5. Frequent symbols in Bengali

The transliterated English symbols for this 17 Bengali symbols are also used to represent other Bengali symbols. For example, kh is require to represent খ but ক is also represent by k and হ is also represent by h. The frequency of k and h is increase though kh is used to represent খ anywhere in the text. According to the Bengali philosopher Munir Chowdhury, the 9 graphemes in Figure 5, are the most frequent in Bengali texts [17]. To represent these most frequent Bengali symbols we need only one English symbol. That means the Bengali symbol which requires two or more symbols in English has very negligible effect to achieve better compression.

E. Data Analysis

Table I: Sample strings



SN.	Language	Sample Text	Total symbols	Distinct symbols
1.	Bengali	ও আমার দেশের মাটি, আমি তোমায় ভালবাসি।	38	20
	English	o amar deSer maTi, ami tomay valobasi.	38	18
2.	Bengali	সমস্ত মানুষ স্বাধীনভাবে সমান মর্যাদা এবং অধিকার নিয়ে জন্মগ্রহণ করে। তাঁদের বিবেক এবং বুদ্ধি আছে; সুতরাং সকলেরই একে অপরের প্রতি ভ্রাতৃত্বসুলভ মনোভাব নিয়ে আচরণ করা উচিৎ।	42	169
	English	somost manuSh sbadhInvabe soman morzada ebong odhikar niye jonmogrohoN kore. Tader bibek ebong buddhi ache; sutorang sokoleroi eke oporer proti vratritbsulov monOvab niye acoroN kora ucit.	30	197
3.	Bengali	আমার সোনার বাংলা, আমি তোমায় ভালোবাসি। চিরদিন তোমার আকাশ, তোমার বাতাস, ও মা আমার প্রানে বাজায় বাঁশি।। ও মা, ফাগুনে তোর আমের বনে ঘ্রানে পাগল করে, মরি হায়, হায় রে ও মা, অঘ্রাণে তোর ভরা ক্ষেতে আমি কি দেখেছি মধুর হাসি।। কী শোভা, কী ছায়া গো, কী স্নেহ, কী মায়া গো- কী আঁচল বিছায়েছ বটের মূলে, নদীর কূলে কূলে। মা তোর মুখের বাণী আমার কানে লাগে সুধার মতো, মরি হায়, হায় রে মা, তোর বদনখানি মলিন হলে, ও মা, আমি নয়ন জলে ভাসি।।	404	42
	English	amar sOnar bangla, ami tOmay valObasi. cirdin tOmar akaS, tOmar batas, O ma amar praNe bajay ba^Si.. O ma fagune tOr amer bone ghrane pagol kore, Mori hay, hay re- O ma, oghraNe tOr vora kShete ami ki dekhechi modhur hasi.. kI SOva, kI chaya gO, kI sneh, kI maya gO- kI a^col bichayechbo TermUle, nodIr kUle kUle. ma tOr mukher baNI amar kane lage sudhar mOto, mori hay, hay re- ma, tOr bodonkhani molin hole, O ma, ami noyon jole vasi..	431	34

We consider some sample Bengali text and transliterated to English. We count their frequencies in terms of number of distinct symbols as well as number of total symbols. Details are shown in Table I.

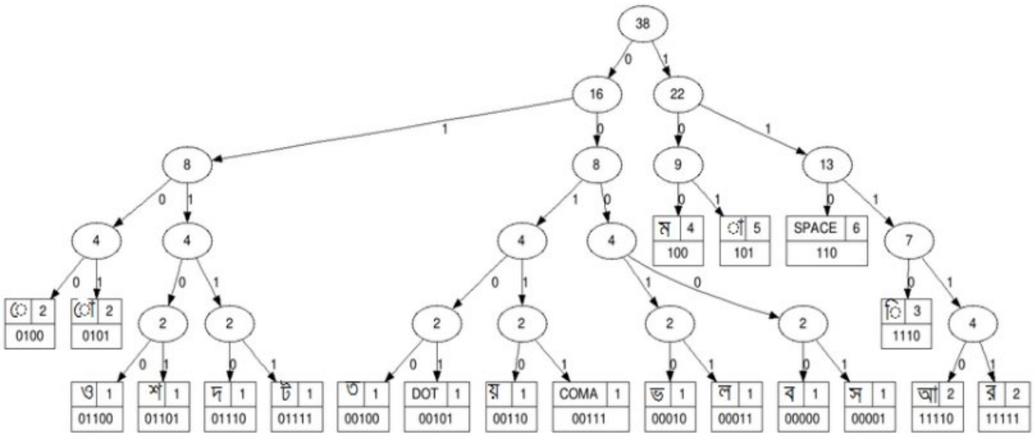


Figure 6(a). Huffman Tree for sample string 1 (Bengali)

For sample Bengali text 1, the constructed Huffman tree are shown in Figure 6(a). The codeword and some other parameter are also shown in Table II (a).

Table II (a): Huffman code and Unicode for Sample String 1 (Bengali)

SN	Symbol	Freq.(F_i)	Unicode Codeword	Huffman Codeword	X_i	Fi*Xi
0	space	6	0000 0000 0010 0000	110	3	18
1	া	5	0000 1001 1011 1110	101	3	15
2	ম	4	0000 1001 1010 1100	100	3	12
3	ি	3	0000 1001 1011 1111	1110	4	12
4	ে	2	0000 1001110 00111	0100	4	8
5	ো	2	0000 1001 1100 1011	0101	4	8
6	আ	2	0000 1001 1000 0110	11110	5	10
7	র	2	0000 100110110000	11111	5	10
8	ব	1	0000 1001 1010 1100	00000	5	5
9	স	1	0000 1001 1011 1000	00001	5	5
10	ভ	1	0000 1001 1010 1101	00010	5	5
11	ল	1	0000 1001 1011 0010	00011	5	5
12	ত	1	0000 100110100100	00100	5	5
13	য়	1	0000 1001 1101 1111	00110	5	5
14	ও	1	0000 1001 1001 0011	01100	5	5
15	শ	1	0000 100110110110	01101	5	5
16	দ	1	0000 1001 1010 0110	01110	5	5
17	ট	1	0000 10011001 1111	01111	5	5
18	,	1	0000 0000 00101100	00111	5	5
19	।	1	0000 1001 1111 0111	00101	5	5
Total=20		$\sum F_i$ =38	Total=20*16=320		$\sum X_i$ =91	$\sum F_i*X_i$=153

The Huffman tree for transliterated Bengali text-1 are shown in Figure 6(b). The required bit, code word for each symbols and some other parameter are shown in Table II(b).

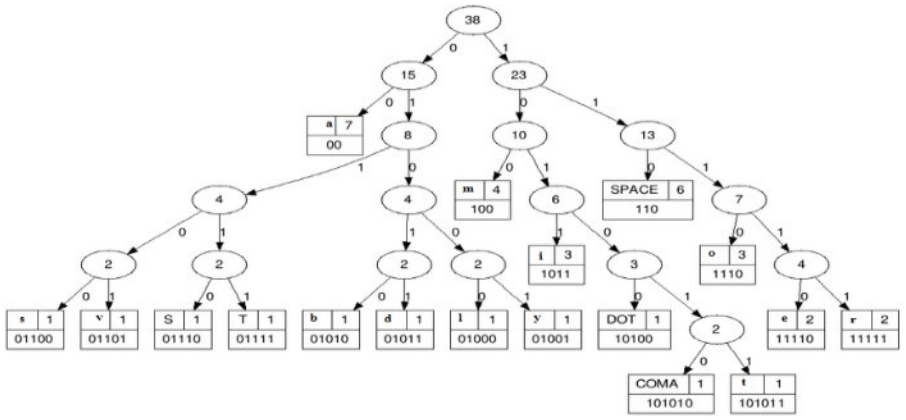


Figure 6(b). Huffman Tree for sample string 1 (English)

Table II (b): Huffman code and ASCII for sample 1 (English)

Seq. no.	Chars.	ASCII (Decimal)	Frequency (Fi)	ASCII (Binary)	Huffman	BitperFreq (X_i)	Fi*X_i
0	'a'	97	7	01100001	00	2	14
1	' '	32	6	00100000	110	3	18
2	'm'	109	4	01101101	100	3	12
3	'i'	105	3	01101001	1011	4	12
4	'o'	111	3	01101111	1110	4	12
5	'r'	114	2	01110010	11111	5	10
6	'e'	101	2	01100101	11110	5	10
7	'.'	46	1	00101110	10100	5	5
8	'S'	83	1	01010011	01110	5	5
9	'T'	84	1	01010100	01111	5	5
10	'b'	98	1	01100010	01010	5	5
11	'd'	100	1	01100100	01011	5	5
12	'l'	108	1	01101100	01000	5	5
13	's'	115	1	01110011	01100	5	5
14	'v'	118	1	01110110	01101	5	5
15	'y'	121	1	01111001	01001	5	5
16	','	44	1	00101100	101010	6	6
17	't'	116	1	01110100	101011	6	6
Total=18			$\sum$Fi=38	Total=18*8=144		$\sum X_i$=**83**	$\sum$Fi*X_i=**145**

***B.1. UNICODE representation** for sample text 1*

The total number of bits required to encode “sample text 1” using unicode encoding = No. of Unicode code bits for a symbol * No. of symbols in the text = 16*38 = 608. The thirty eight Unicode symbols of the sample string-1 are encoded by placing the 16- bit sequence symbols one after another. There is no overhead expense here. Decoding is petty straight forward. The decoder will treat every 16 bits as a unique symbol and assign its corresponding UICODE symbol against that 16 bit. To encode this way, it require a great amount of space but noticeable benefit is decoding is simple.

***B.2. Bengali Text Compression Using Huffman Principles** for sample text-1*

At the time of encoding the message, we have to take care about the decoding phase and send the corresponding header information with the data sequences. The message is composed with data sequence and header sequence.

Data sequence for sample text-1: According to Table II (a) the sequence is follows:

01100110111101001011111111001110010001101
01001111111010010101111111000111110111101
00111011000100010010010100110110000101010
001101000000010100001111000101

The total numbers of bit require to encode the message can be calculated by the following formula.

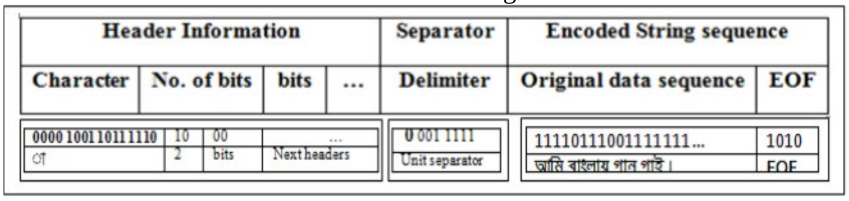
Number of bits for Huffman encoding =∑ (No. of Huffman bits for a symbol * frequency of the symbol) =
$\sum X_i*F_i$ =5*16+4*7+3*15=80+28+45=153

The average code length for Huffman Bengali text can be calculated as- Total number of bits required to encode the message/ Number of distinct symbols =153/38 =4.0263. Whereas the average code length is 16 for UNICODE representation. It indicates that proposed technique will be faster than Unicode technique.

Header sequence for sample text 1: For decoding the message it require the information of original symbols and their corresponding huffman bits. To form the header, initially sixteen bit Unicode is added, then a constant 4 bit code which represent a number to indicate how many next bits are represent the Unicode symbol, and finally the Huffman code of the symbol is added to make a header information for a symbol. The procedure is shown in Table III.

Number of bits required of header information for sample text 1 = 16*Number of distinct symbols + Number of Huffman bits to represent the distinct symbols + Huffman bits of the symbol = 20*16+20*4+91=491.

Table III: Overall Message instance



Header Information				Separator	Encoded String sequence	
Character	No. of bits	bits	…	Delimiter	Original data sequence	EOF
0000 1001 1011 1110	10	00	…	0 001 1111	11110111001111111…	1010
া	2	bits	Next headers	Unit separator	আমি বাংলায় গান গাই।	EOF

The total number of bits to process the information is depicted in Table III and can be calculated as: Header bits + Number of bits for delimiter (Separator) + bits for data sequence =491+8+153=652.

Using Huffman encoding it require few bit to represent data than Unicode encoding. If we wish to gain more compression ratio than regular Huffman encoding technique we may transliterate Bengali text into English and then apply Huffman principle to encode the transliterated text.

***B.3. Bengali Text Compression Using Transliteration and Huffman Principles** for sample text-1*

Data sequence for sample text-1: After applying Huffman principle on transliterated sample string -1 (English) we get Figure 6(b) as Huffman tree and Table II(b) as Huffman code word. Using these data the compress message will be as follows.

11101100010000111111001011111001110111
01111111010000011111011101010110001001011
11010101111101000001001110011010001000111
0010100001100101110100

Number of bits for Transliterated Huffman encoding = ∑ (Number of Huffman bits for a symbol * frequency of the symbol) = ∑Xi*Fi = 6*2 +5*13 +4*6+3*10+2*7 = 12+65+24+30+14 =145

The average code length for transliterated English text can be calculated as-

Total number of bits required to encode the message/ Number of distinct symbols =145/38 =3.8157 which indicates a faster search than those of Unicode and Huffman Bengali text.

Number of header bits = 8*Number of distinct symbols +Number of Huffman bits to represent the distinct symbols+ Huffman bits of the symbol = 8*18+4*18+83 =299

Total number of bits to process the information=Number of bits to encode the message + numbers of bit for the separator + numbers of bits require to encode the header information= 299+8+145= 452.

If we transliterate the original Bengali text to English and process the English string than the original Bengali one, we need few memory to represent data. Though the amount of memory compression achieved is moderate for the small text. It will increase sharply with the increase of the text size.

B.4 *Performance* ***analysis for large text.***
We have considered the sample text 2 and 3 in Table I. we have constructed Huffman tree with the symbols of the samples and listed the outcome in Table IV and V repectively.

Table IV: Medium size text sample (string 2)

	Bengali	**English**
Frequency	া=4; space=3; গ=2; ং=1;ল=1; য়=1; ন=1; আ=1; ম=1; ি=1; ব=1;ই=1; ।=1; & EOF=1	a=5; g=3; space=3; i=2; n=2; l=1; y=1; .=1; m=1 ; b=1; & EOF=1
Distinct Char	42	30
Total character	169	189
Fixed Encoding	Unicode = 169*16=2704	ASCII Code = 189*8=1512
Number of Header bits	42*16+42*4+246=1086	30*8+4*8+169=529
Separator	8	8
Number of bits for sequence	=∑ (No. of Huffman code bits for a symbol * frequency of the symbol) =785	=∑ (No. of Huffman code bits for a symbol * frequency of the symbol) =804
Total Number of bits	Header +Separator +sequence =1086+8+785=1871	Header +Separator +sequence =529+8+804=1333
Average code length	785/169=4.6450	802/189=4.2437

For sample text 2 and 3 in Table IV and V, we have listed the frequencies against each symbol in terms of number of distinct and total symbols. From these data we have calculated the number of bits require to encode Bengali and transliterated English text. For Bengali, we multiply the total symbols by 16, as 16 bit is required to encode each Bengali character using Unicode encoding and in the same way for English, we multiply the total symbols of the transliterated text by 8, as 8 bit can represent each English character according to ASCII chart. To calculated the bit requirement for proposed technique we have calculated the number of bit requre to process header as well as encode the original text. Than we have sumup the header bit with the bit require to process the data sequence.

Table V: Using large sample (string 3)

	Bengali	**English**
Frequency	া=4; space=3; গ=2; ং=1;ল=1; য়=1; ন=1; আ=1; ম=1; ি=1; ব=1;ই=1; ।=1; & EOF=1	a=5; g=3; space=3; i=2; n=2; l=1; y=1; .=1; m=1; b=1; & EOF=1
Distinct Char	42	34
Total character	404	431
Fixed Encoding	Unicode : 6464 bit	ASCII code: 3448 bits
Number of Header bits	42*16+42*4+282=1122	34*8+34*4+206=614
Separator	8	8
Number of bits for sequence	=∑ (No. of Huffman code bits for a symbol * frequency of the symbol) =1835	=∑ (No. of Huffman code bits for a symbol * frequency of the symbol) =1756
Total Number of bits	Header +Separator +sequence =1122 + 8 +1835 = 2965	Header +Separator +sequence = 1756 + 8 + 614 = 2378
Average code length:	1835/404=4.5421	1756/431=4.0742

We can calculated the average number of bit require to encode the entire message by dividing the total number of bit with total number of symbols in the corresponding text in Bengali and transliterated English text.

B.5. Implementation

The Algorithm for encoding process is given bellow:

Algorithm: Encoding Technique
Step 1: Input a Bengali Text.
Step 2: Transliterate the Bengali Text into English as C.
Step 3: Call HUFFMAN (C) to construct a Huffman Tree (HT) from the transliterated Text of step 2.
Step 4: Evaluate the Huffman codeword from the HT of step 3.
Step 5: Create Data sequence from the Huffman codeword of step 4.

The Algorithm for encoding process is given bellow:
Algorithm: Decoding Technique
Step 1: Input the data Sequence.
Step 2: Call Decode Tree to get the symbols of the English Text.
Step 3: Reconvert English text into Bengali using transliteration.

The popular Huffman Algorithm is also given bellow:
Huffman invented a greedy algorithm that constructs an optimal prefix code called a Huffman code. Here is the function, HUFFMAN (C) that we use in encoding phase.

HUFFMAN(C)
```
1    n =|C|
2    Q = C
3    for i =1 to n -1
4    allocate a new node ´
5    z.left = x = EXTRACT-MIN(Q)
6    z.:right = y = EXTRACT-MIN(Q)
7    z. freq = x.freq + y.freq
8    INSERT (Q, z )
9     return EXTRACT-MIN (Q )
```

In the pseudo code that follows, we assume that C is a set of n characters and that each character c € C is an object with an attribute c:freq giving its frequency. The algorithm builds the tree T corresponding to the optimal code in a bottom-up manner. It begins with a set of |C| leaves and performs a sequence of |C| -1 “merging” operations to create the final tree. The algorithm uses a min-priority queue Q, keyed on the freq attribute, to identify the two least-frequent objects to merge together. When we merge two objects, the result is a new object whose frequency is the sum of the frequencies of the two objects that were merged [18].

IV. PERFORMANCE ANALYSIS

The Table VI shows that, for sample string 1, the number of bits requires to encode the Bengali text using Unicode is exactly twice than those of transliterated English text using ASCII code. The number of bit require to encode the Bengali text using regular Huffman and proposed technique is 652 bit and 452 bit respectively. Which achieve almost 31% compression ratio for transliterated text than its counterpart regular Huffman. For sample string 2, the number of bits requires for Bengali text in Unicode is 2704 and for transliterated English text in ASCII code is 1881, which is almost twice more than those of transliterated text. And the number of bit require to encode the Bengali text using regular Huffman and applying Huffman principal on transliterated English text is 1871 bit and 1333 bit respectively, which achieve about 30% compression ratio. For sample string -3, Unicode representation require almost twice as much as those of ASCII representation and the number are 6464 and 3448 for UNICODE and ASCII code respectively. Applying Huffman principal on Bengali text and transliterated English text the number is 2965 and 2378 respectively, which achieve almost 20% compression ratio.

Table VI: Overall compression comparison

S.N.	**UNICODE**	**ASCII Code**	**Regular Huffman (RH)**	**Transliterated Huffman (TH)**	**Compression Ratio (%) =((RH-TH)/ RH)*100**
1	608	304	652	452	30.67%
2	2704	1512	1871	1333	28.75%
3	6464	3448	2965	2378	19.79%

In the bar diagram of Figure 7, the four bar Cornflower Blue, Firebrick, Olive, Slate Blue Drab respectively represent the total number of bit require to represent using UNICODE, ASCII code, regular Huffman and transliterated Huffman for each of the sample string 1, 2 and 3. It has been shown that for every sample, transliterated compression performance is better than regular Huffman technique.

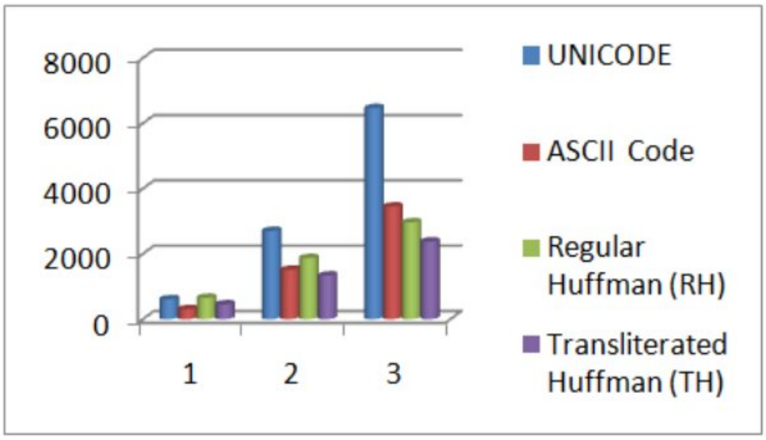


Figure 7. Overall Compression Comparison

It is also noticeable that for small text, like sample string 1 which consists of very few symbols, is always expensive in terms of data compression. But it is always cost-effective if we apply Huffman principle on the transliterated text. This is because of the Unicode header information. If the text is very small where the number of total symbols is nearly equal to the number of distinct symbols, then regular Huffman encoding requires more bits than Unicode technique or ASCII encoding. In regular Huffman encoding technique the text information has to process using Unicode information. On the other hand, transliterated text allows processing the information using ASCII value which minimizes the space requirement. Thus application of Huffman technique on transliterated text yields better compression ratio than the existing regular Huffman technique, ASCII code or UNICODE representation.

V. CONCLUSION

In this paper, we have proposed a lossless data compression technique for more symbolic language like Bengali. Bengali text is first transliterated to English and then Huffman encoding is applied on the transliterated text. The resulting compression ratios are compared with UNICODE, ASCII and regular Huffman encoding. Experimental result shows that the proposed method achieved about 30% enhancement in compression. Although the model has been developed and applied to only Bengali texts, the underlying idea could be investigated for other languages as well, especially for languages that have more symbols than English.

REFERENCES

[1]. P. Fenwick, “Differential Ziv-Lempel Text Compression,” Computer Journal of Universal Computer Science, vol. 1, no. 8, pp. 591-602, 1995.

[2]. A. Lelewer, and S. Hirschberg, “Data Compression,” Computer Journal of ACM Computing Surveys, vol. 19, no. 3, pp. 261-296, 1987.

[3]. M. Long, I. Natsev, and S. Vitter, “Text Compression via Alphabet Re-Representation,” Computer Journal of Neural Networks, vol. 12, no. 4, pp. 755-776, 1999.

[4]. M. R. Islam, and S. A. Rajon, “On the design of an effective corpus for evaluation of Bengali Text Compression Schemes,” Proceeding of ICCIT 2008, pp.236-241.

[5]. A. S. M. Arif, A. Mahamud and R. Islam, “ an enhanced static data compression scheme of bengali short message,” International Journal of Computer Science and Information Security,Vol.4, No-1& 2,2009.

[6]. S.M.Humayun, S.H. Rahman and M. Kaykobad, “Static huffman code for Bangla text,” Proceedings of the 15th Annual Conference of BAAS, Section III, (ACBS'90), AERE, Savar, pp: 5-8, 1990.

[7]. M.A.Mannan, R.A. Chowdhury and M. Kaykobad, “A Storage Efficient Header for Huffman Coding,” Proceeding of ICCIT 2001, Dhaka, pp: 57-59.

[8]. C.K. Roy, M.M. Masud, M.M. Asaduzzaman and H.M. Hasan, 2001, “ A modification of huffman header, “Proceeding of ICCIT 2001, pp: 62-65.

[9]. Ahsan Habib, A.S.M. Latiful Haque, Md. Russel Hossain, H-HIBASE: Compression Enhancement of HIBASE Technique Using Huffman Coding, *Journal of Computers*, Vol 8, No 5 (2013), 1175-1183, May 2013.

[10]. I. Akman, “A New Text Compression Technique Based on Language Structured,” Computer Journal of Information Science, vol. 21, no. 2, pp. 87-94, 1995.

[11]. B. Diri, “A Text Compression System Based on the Morphology of Turkish Language,” *in Proceedings of the 15th International Symposium on Computer and Information Sciences*, Turkey,pp. 156-159, 2000.

[12]. Y. Wiseman, and I. Gefner, “Conjugation-based Compression for Hebrew Texts,” Computer Journal of ACM Transactions on AsianLanguage Information Processing, vol. 6, no. 1, pp. 1-10, 2007.

[13]. J. Yaghi, R. Titchener, and S. Yagi , “T-Code Compression for Arabic Computational Morphology,” *in Proceedings of the Australasian Language Technology Workshop*, Australia, pp. 425-465, 2003.

[14]. Unicode List of Bengali: Available at Official website of Unicode Standard 5.0, Unicode Inc. http://www.unicode.org. Retrieved on November 02, 2009.

[15]. N. S. Kharusi, &, A. Salman, (2011) The English Transliteration of Place Names in Oman. Journal of Academic and Applied Studies Vol. 1(3) September 2011, pp. 1–27 Available online at www.academians.org

[16]. S. Lipschutz, and G.A.V. Pai, “Data Structures”,Chapter-7, Page 4, 65-66, 2006

[17]. M. Chowdhury, (1963), "Shahitto, shônkhatôtto o bhashatôtto (Literature, statistics and linguistics)", Bangla Academy Potrika (Dhaka) 6 (4): 65–76

[18]. Thomas H. Cormen, Charles E. Leiserson, R. Rivest, and C. Stein, Introduction to Algorithms, 2009, Massachusetts Institute of Technology, The MIT Press Cambridge, Massachusetts London, England.