Bangla Text Compression Based on Modified Lempel-Ziv-Welch Algorithm

Linkon Barua[1], Pranab Kumar Dhar[*1], Lamia Alam[1], and Isao Echizen[2]

[1]Department of CSE, Chittagong University of Engineering and Technology (CUET), Chittagong-4349, Bangladesh

[2]Digital Content and Media Sciences Research Division, National Institute of Informatics (NII), Tokyo 101-8430, Japan

Email: linkonb@gmail.com, pranabdhar81@gmail.com, lamiacse09@gmail.com, iechizen@nii.ac.jp

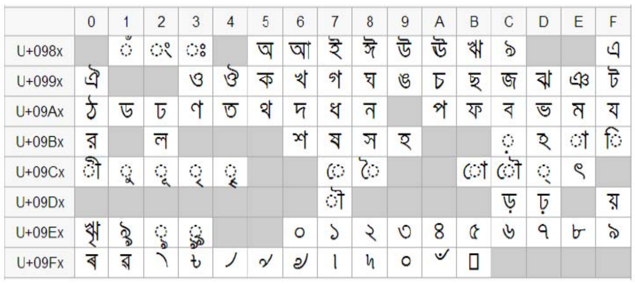
***Abstract*—Text compression algorithm performs compression at the character level. Bangla text has some unique features such as no distinct upper and lower case letter, consonant cluster (CC) and consonant with dependent vowel sign (CV) etc. The conventional Lempel-Ziv-Welch (LZW) algorithm is not suitable for compressing Bangle text. Therefore, in this paper, we propose a modified LZW (MLZW) algorithm which can compress Bangla text effectively and efficiently. In our proposed method, a dictionary with Unicode ranges from 1-90 is used for Bangla characters. The compression process is started with checking the input character. If input character is a part of CC or CV, then CC or CV is considered as a character and search it in the dictionary. If the character to be encoded is already in dictionary, encode it with the dictionary index. Otherwise, the character is added to the dictionary and is encoded with its corresponding dictionary index. Simulation results indicate that the proposed MLZW algorithm compresses Bangla text effectively and efficiently. We observed that the proposed MLZW provides higher compression rate approximately 3% for dictionary index and 33% for output sequence compared with LZW algorithm.**

Keywords—* *Bangla Text; ASCII code; LZW; Data Compression; Unicode; Consonant Cluster.

I. INTRODUCTION

Compression is the process of reducing the size of a file or data by identifying and removing redundancy in its structure. Statistical Modeling and Dictionary based techniques are mainly two streams of text compression techniques. The dictionary-based technique is the mostly adapted approach for English text compression. Lempel-Ziv-Welch (LZW) is the most well-known dictionary based text compression techniques. LZW is a universal lossless data compression algorithm and is good for text compression. A dictionary or code table based encoding algorithm can compress a large size of text file into half of its size [1].

Bangla is a native language of the people of Bangladesh and West Bengal of India. It is written using the Bangla alphabet and is the 6th most widely used writing system in the world [2]. In Bangla, we have about 90 distinct symbols including 11 independent vowels (অ আ ই ঈ উ ঊ ঋ এ ঐ ও ঔ), 39 consonants (ক খ গ ঘ ঙ চ ছ জ ঝ ঞ ট ঠ ড ঢ ণ ত থ দ ধ ন প ফ ব ভ ম য র ল শ ষ স হ ড় ঢ় য় ৎ ং ঃ ঁ), dependent vowel signs (া ি ী ু ূ ৃ ে ৈ), two part dependent vowel signs (ো ৌ) , various signs, additional signs (ব ফলা, য ফলা, র ফলা, রেফ , হসন্ত), and Bangla numerals (০ ১ ২ ৩ ৪ ৫ ৬ ৭ ৮ ৯) etc. [3] . The Unicode chart for Bangla character is shown in Fig. 1. Bangla alphabet also have join or conjunct characters such as ক্ষ , জ্ঞ ,হ্ম ন্ট etc. and dependent vowel signs which is absent in English text. Fig. 2 shows an example of consonant clusters (CC) and consonant with dependent vowel signs (CV). Because of these unique features, in this paper, we introduce a modified LZW (MLZW) algorithm for compressing Bangla text effectively and efficiently. To the best of our knowledge, this is the first Bangla test compression algorithm based on MLZW algorithm. Simulation results indicate that the MLZW algorithm compresses Bangla text effectively and efficiently. Comparison analysis shows that the proposed algorithm shows superior performance than the conventional LZW algorithm in terms of compression rate.



	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	৺	৻				

Fig. 1. The chart for Unicode Bangla characters

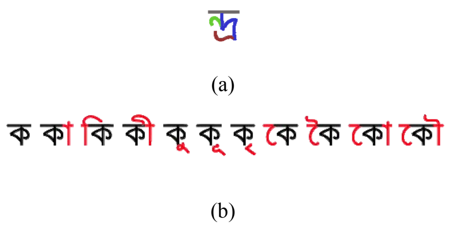


Fig. 2. Example of (a) consonant cluster (যুক্তবর্ণ) (b) Consonant with dependent vowel sign

II. RELATED WORK

A significant number of text compression techniques have been introduced in literature for English, Hebrew, and Arabic texts. However, Bangla text is quite different from these texts

[*]Corresponding author

978-1-5090-5627-9/17/$31.00 ©2017 IEEE

[4-6]. An effective approach for Bangla text compression was described in [7]. This method utilizes string ranking technique for indexing the source text to achieve hierarchical compression. Static coding is also employed in the proposed method to encode the data for compression. However, the compression rate of this method is little low. The authors of [8] proposed a compression method based on static Huffman coding. In addition, they also utilized WinZip for compression. In this technique, frequency distribution of the characters plays an important role for appropriate coding. But it takes more time for compressing and decompressing than WinZip. Hossain *et al.* [9] introduced a compression technique where Bangla text is first transliterated to English first and then Huffman encoding is applied on the transliterated text. However, the encoding process will be complex for long text message, indicating that it is not suitable for long text message. Marzan *et al.* [10] proposed a lossless innovative approach using text representation scheme. Here, Bangla character was represented by a unique 2-digit intermediate decimal value. Successive subtraction is performed on the word values for indexing and sorting which reduces the value of the numbers. In [11], authors designed a corpus for evaluation of dictionary based Bangla text compression. Mannan *et al.* [12] presented a compression technique which provides a tradeoff between compression rate and decoding speed for short message. In [13], a coding system for symbols of Bangla text was proposed. Arif *et al.* [14] described a Bangla text compression method based on Unicode standard. However, these method is not suitable for long text.

The conventional LZW algorithm was proposed by Lempel, Ziv, and Welch which first initializes all the characters of the alphabet into the dictionary. In the encoding process, the encoder takes characters one by one and accumulates into a string S. Enter a character for each alphabet, and continue inserting to the last of S, and then search string S in the dictionary as long as the S was found. Then at a certain point, add the next character x into string S, if it fails to find x in S. Then the encoder should do the following operation: (1) the output pointer points the string S into dictionaries, (2) the string S_x stored in the next available dictionary, (3) the string S presets for x.

In the decoding process, the decoder inputs a point, uses it to take back a dictionary entry S, and writes the S into the decoder output stream. Then the decoder indicates the next point, takes back the next dictionary entry J, and writes the J into the decoder output stream. Meanwhile, taking the first character x, when S_x is not in the dictionary put S_x into the next available dictionary entry, and preset S for J [15]. The conventional LZW algorithm is not suitable for Bangla text compression. Because it has some unique features such as no distinct upper and lower case letter, CC, and CV, etc.

III. Proposed Method

The MLZW algorithm creates a dictionary for Bangla characters whose Unicode values range from 1to 90. The flow chart of the MLZW algorithm is shown in Fig. 3. The encoding process of the proposed MLZW algorithm starts with checking the input if it is CC or CV. If input character is the part of CC or CV then we consider CC or CV as a character and search it in the dictionary. If the character to be encoded is already in dictionary, encode it with the dictionary index. Otherwise, the character is added to the dictionary and is encoded with its corresponding dictionary index. The MLZW encoding algorithm is shown in Fig. 4

Let us consider an example, a Bangla word 'ইন্টারনেট' (Internet) have one consonant cluster 'ন্ট' and two consonant with dependent vowel signs 'ন্টা' and 'নে'. The consonant cluster 'ন্ট' is created by two Bangla character 'ন' and 'ট'. Encoding and decoding of these word using our proposed MLZW is described below:

A. Encoding

Step 1: Initially, we create a dictionary for Bangla characters whose Unicode values range from 1to 90.

Step 2: The encoder first encounters the letter 'ই' . This letter is in the dictionary. Therefore, encode 'ই' with its dictionary index 7.

Step 3: Now concatenate the next letter 'ন' with 'ই', forming the pattern 'ইন'. The Character 'ন' is not independent. It is a consonant cluster. Therefore, concatenate this consonant clusters 'ন্ট' with previous pattern, forming the pattern 'ইন্ট'. Again, 'ন্ট' have dependent vowel sign ◌া, forming the new pattern 'ইন্টা'. This pattern is not in the dictionary. For this reason, we have added the pattern 'ইন্টা' and consonant cluster 'ন্ট' to the dictionary as the new initial element of the dictionary and rebuild the dictionary table .We encode 'ন্টা' with its dictionary index 91.

Step 4: Begin a new pattern starting with the consonant cluster 'ন্টা'. Concatenate the next element 'র' to form the pattern 'ন্টার'. This pattern is not in the dictionary, therefore encode 'র' with its dictionary index value 43, add the pattern 'ন্টার' to the dictionary as the next element of the dictionary.

Step 5: Concatenate the next letter 'ন' with 'র', forming the pattern 'রন'. The Character 'ন' is not independent. Because, it has dependent vowel sign '◌ে' which forms new pattern 'রনে'. This pattern is not in the dictionary, therefore, the pattern 'রনে' is added to the dictionary and 'নে' is the new element of the dictionary. Rebuild the dictionary table and we encode 'নে' with its dictionary index 92**.**

Step 6: Now begin a new pattern starting with the letter 'নে' . Concatenate the next element 'নে' to form the pattern 'ট'. This pattern is not in the dictionary, so encode 'ট' with its dictionary index value 27, add the pattern 'নেট' to the dictionary as the next element of the dictionary.

The dictionary and output sequence is given in Table I. From this table, we observed that, in case of LZW and MLZW algorithms, the total numbers of dictionary indices are 98 and 96, respectively, and total numbers of output sequences are 9 and 5, respectively for the word 'ইন্টারনেট'.

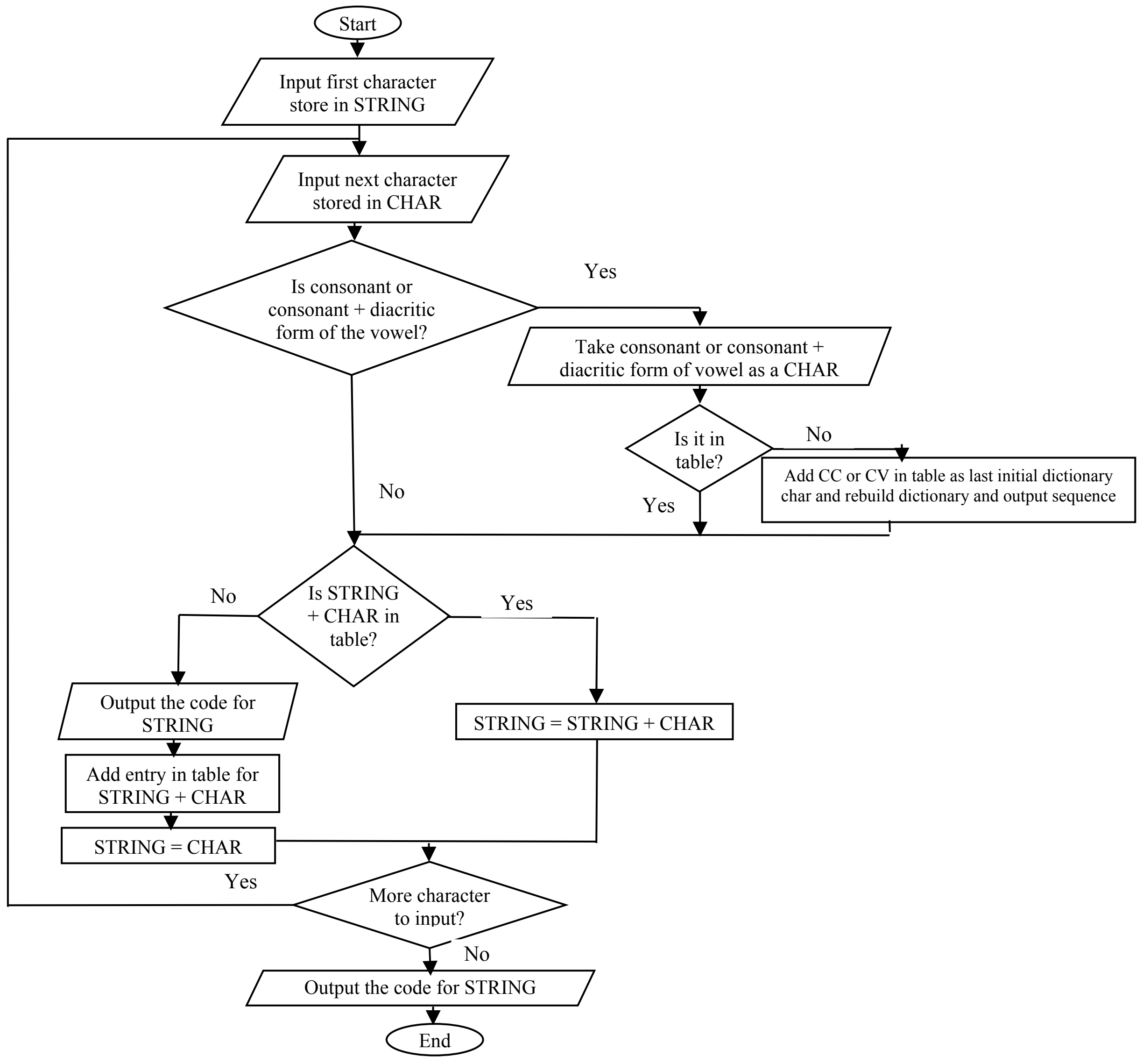

Start
Input first character store in STRING
Input next character stored in CHAR
Is consonant or consonant + diacritic form of the vowel?
Yes
Take consonant or consonant + diacritic form of vowel as a CHAR
Is it in table?
No
Add CC or CV in table as last initial dictionary char and rebuild dictionary and output sequence
Yes
No
Is STRING + CHAR in table?
No
Yes
Output the code for STRING
STRING = STRING + CHAR
Add entry in table for STRING + CHAR
STRING = CHAR
Yes
More character to input?
No
Output the code for STRING
End

Fig. 3. Flow chart of MLZW compression algorithm

```
STRING = get input character
WHILE there are still input characters DO
CHAR = get input character
IF CHAR is CC or CV  then
CHAR = get input CC or CV
      IF CHAR is not in the string table then
      add CHAR to the string table and Rebuild string        table
      and output the code for STRING
      END of IF
END of IF
IF STRING+CHAR is in the string table then
STRING = STRING+CHAR
ELSE
output the code for STRING
add STRING+CHAR to the string table
STRING = CHAR
END of IF
END of WHILE
```

Fig. 4. MLZW encoding algorithm

TABLE I. LZW & MLZW ENCODING PROCESS FOR THE BANGLA WORD 'ইন্টারনেট'

Dictionary for the word ' ইন্টারনেট ' (Internet)				
LZW			*MLZW*	
Index	*Entry*		*Index*	*Entry*
1-90	All Bangla Character		1-90	All Bangla Character
91	ইন		91	ন্টা
92	ন্		92	নে
93	্ট		93	ইন্টা
94	টা		94	ন্টার
95	ার		95	রনে
96	রন		96	নেট
97	নে			
98	েট			
The encoder Output Sequence				
7,36,61,27,50,43,36,57,27			7,91,43,92,27	

B. Decoding

The decoding process starts with reading a value from the encoded input and generating the corresponding string from the initialized dictionary. In order to rebuild the dictionary, it follows the same way as it was built during encoding. It also obtains the next value from the input and adds it to the dictionary. The concatenation of the current string and the first character of the string obtained by decoding the next input value, or the first character of the string just output if the next value cannot be decoded. The decoder then proceeds to the next input value and repeats the process until there is no more input, at which point the final input value is decoded without any other additions to the dictionary. For Bangla word 'ইন্টারনেট' (Internet) the decoding process is given below:

Step 1: The decoder starts with the initial dictionary of the encoder. (Dictionary Index 1-92).

Step 2: The decoder first encounters the code 7. This code is for letter 'ই' which exists in the dictionary; therefore, we do not add it to the dictionary and continue with the decoding process.

Step3: The next decoder input is 91, which is the index corresponding to the pattern 'ন্টা'. We decode 'ন্টা' and concatenate it with our current pattern to form the pattern ইন্টা. As this does not exist in the dictionary, we add it as the next element of the dictionary.

Step 4: Now start a new pattern beginning with the letter 'ন্টা' and decoder input is 43, which is the index of the pattern 'র' and new concatenated pattern is 'ন্টার' which is not exist in the dictionary, therefore, new element is added to the dictionary.

Step 5: Similarly, next two inputs 92, 27 correspond to the letters 'নে' and 'ট'. Table II shows the MLZW dcoding process for the Bangla Word 'ইন্টারনেট.

In this study, the compression rate is calculated in terms of dictionary index and output sequence. It is defined by the ratio of the total number of dictionary index of MLZW and the total number of dictionary index of LZW.

TABLE II. MLZW DCODING PROCESS FOR THE BANGLA WORD 'ইন্টারনেট'

Encoder Output Sequence		
7, 91, 43, 92, 27		
	Index	*Entry*
Dictionary for 'ইন্টারনেট' (Internet)	1-90	All Bangla Character
	91	ন্টা
	92	নে
After Decode 7,91	93	ইন্টা
After Decode 7,91,43	94	ন্টার
After Decode 7,91,43,92	95	রনে
After Decode 7,91,43,92,27	96	নেট

TABLE III. ENCODING RESULTS OF LZW AND MLZW ALGORITHMS FOR SOME BANGLA SENTENCES

Input Bangla Text	**Total Word**	**Total Dictionary Index**		**Total Output Sequence**	
		LZW	***MLZW***	***LZW***	***MLZW***
আমার সোনার বাংলা আমি তোমায় ভালোভাসি	6	120	122	31	23
চট্টগ্রাম প্রকৌশল ও প্রযুক্তি বিশ্ববিদ্যালয়ের ক্যাম্পাস খুবই সুন্দর	8	150	138	61	34
পদ্মার ঢেউ রে মোর শূন্য হৃদয়ে পদ্ম নিয়ে যা , যা রে	12	135	132	42	28

IV. EXPERIMENTAL RESULTS AND DISCUSSION

In this section, we have provided a comparison between the conventional LZW coding and proposed MLZW coding processes and the results are shown in Table III for different length of Bangla sentences. We observed that for MLZW the size of dictionary decreases and also the encoder output sequence is less then LZW for all Bangla sentences. Compression results in terms of dictionary index (DI) and encoded sequence obtained from different length of sample Bangla sentences are given in the Table IV. Fig. 5 and Fig. 6 show the graphical representations of compression ratios between the LZW and proposed MLZW coding techniques for different number of Bangla character in terms of dictionary index and encoded sequence, respectively. We observed that the proposed MLZW provides higher compression rate approximately 3% for dictionary index and 33% for output sequence compared with LZW algorithm. From these results, we can conclude that for every sentence or word, MLZW algorithm outperforms LZW in terms of compression for dictionary index and output sequence.

TABLE IV. ENCODING RESULTS OF LZW AND MLZW ALGORITHMS FOR DIFFERENT BANGLA WORDS

Total Input Word (Bangla)	LZW		MLZW		Compression Rate	
	Total DI	Total Output Sequenc e	Total DI	Total Output Sequence	Dictionary Index	Output Sequence

109	562	461	544	346	1.03	1.33
143	679	574	660	451	1.03	1.27
141	648	540	608	410	1.07	1.32
194	864	761	819	585	1.05	1.30
217	874	762	831	600	1.05	1.27
175	776	670	713	519	1.09	1.29
199	825	724	798	567	1.03	1.28
184	881	758	753	522	1.17	1.45
265	987	885	936	715	1.05	1.24
260	1027	916	979	739	1.05	1.24

.

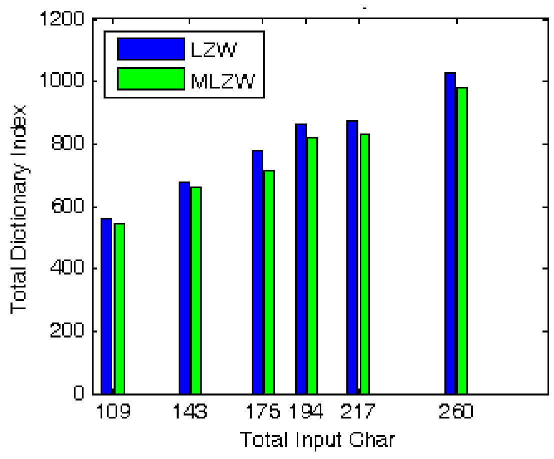


Fig. 5. Comparison of dictionary index between LZW and MLZW algorithms

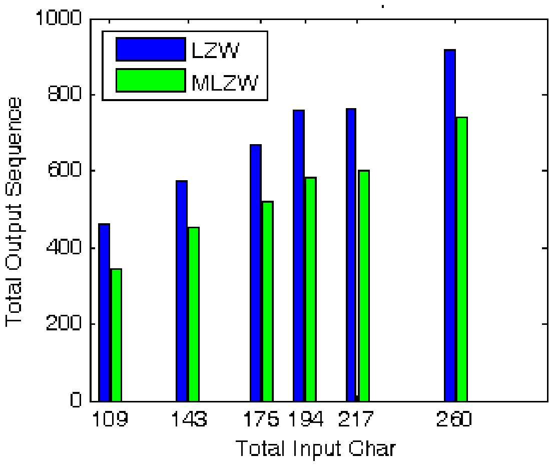


Fig. 6. Comparison of encoding sequences between LZW & MLZW algorithms

V. Conclusion

In this paper, we have presented a MLZW algorithm for Bangla text compression. The rate of compression depends on frequency of join or conjunct characters and dependent vowel signs. Experimental results show that the proposed MLZW algorithm improves the compression rate effectively than the conventional LZW algorithm. As the proposed algorithm is adapted for Unicode standard, it may be used very easily for any Bangla text compression. The proposed algorithm can also be used in small memory devices and text communication. These results verify that the proposed MLZW algorithm can be a suitable candidate for Bangla text compression. The proposed MLZW algorithm can be applicable to other languages that have similar characteristics like Bangla texts.

References

[1] K. Sayood, Introduction to Data Compression, 3rd Edition, Morgan Kaufmann Publishers, 2006,.

[2] https://en. wikipedia.org/wiki/Bangla_alphabet.

[3] https://www.unicode.org.

[4] M. A. K. Azad, R. Sharmeen, S. Ahmad and S. M. Kamruzzaman, "An Efficient Technique for Text Compression," *in Proc. 1st International Conference on Information Management and Business* (IMB), 2005.

[5] Y. Wiseman, and I. Gefner, "Conjugation-based Compression for Hebrew Texts", *ACM Transactions on Asian Language Information Processing*, vol. 6, no. 1, pp. 1-10, 2007.

[6] J. Yaghi, R. Titchener, and S. Yagi , "T-Code Compression for Arabic Computational Morphology," *in Proc. Australasian Language Technology Workshop*, Australia, pp. 425-465, 2003.

[7] S. A. A. Rajon and M. R. Islam, "An Effective Approach for Compression of Bangla Text," International Journal of Computer Engineering and Techlology (IJCIT), 2011, vol. 01, no. 02.

[8] M. S. Hossain and R.C. Debnath, "A Comparative Study of Bangla Text Compression with Winzip," Information Technology Journal, 2004, vol3, pp. 93-94.

[9] M. M. Hossain,A. Habib and M. S. Rahman, "Transliteration Based Bangla Text Compression Using Huffman Principle," *in Proc 3rd International Conference on Informatics, Electronics & Vision,* 2014.

[10] M. A. Marjan, M. P. Uddin, M. I. Afjal and M. D. Haque, "Developing an efficient algorithm for representation and compression of large Bengali text," in Proc. *9th International Forum on Strategic Technology (IFOST),* Cox's Bazar, pp. 22-25, 2014.

[11] S. M. Humayun, S.H. Rahman and M. Kaykobad, "Static huffman code for Bangla text," *in Proc. 15th Annual Conference of BAAS, Section III, (ACBS '90)*, Savar, pp: 5-8, 1990.

[12] M. A. Mannan, R.A. Chowdhury and M. Kaykobad, "A Storage Efficient Header for Huffman Coding," *in Proc.International Conference on Computer and Information Technology*, Dhaka, pp: 57-59, 2001.

[13] C.K. Roy, M. M. Masud, M.M. Asaduzzaman and H.M. Hasan, "A modification of huffman header," *in Proc. International Conference on Computer and Information Technology*, pp: 62-65, 2001.

[14] 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, 2009.

[15] J. Weimin, "Application Research of the LZW Algorithm in Data Communications," *Computer Engineering & Science Journal*, vol. 26, no.5, pp. 46-48, 2004.