COPYRIGHT © 2011 IJCIT, ISSN 2078-5828 (PRINT), ISSN 2218-5224 (ONLINE), VOLUME 01, ISSUE 02, MANUSCRIPT CODE: 110111

An Effective Approach for Compression of Bengali Text

S. A. Ahsan Rajon *and* Md. Rafiqul Islam

Abstract—In this paper, we propose an effective and efficient approach for compressing Bengali Text. This paper focuses on a methodical study on Bengali text compression techniques. The main target of this research is to provide a framework for Bengali text compression; which ensures a simple and computationally inexpensive effective scheme for Bengali text compression. The proposed Bengali text compression scheme is aimed to encompass the low-overhead data communication and management framework for battery powered, energy constrained devices. The current approaches of data compression and their correspondence, usage and efficiency for compressing Bengali text are also presented in this paper. The comparative analysis of existing compression techniques and proposed approach in terms of time and space complexity along with compression ratio has been integrated. We also present an effective scheme for constructing the training platform or knowledgebase for obtaining compression, as there is no specific tertiary dictionary based Bengali text compression scheme is available for research. The main aspect of the proposed scheme is the integration of string ranking scheme for indexing the source text to achieve hierarchical compression scheme. Static coding is also employed in the proposed scheme to encode the data for compression. This paper also incorporates power consumption analysis of proposed compression scheme along with the performance analysis in terms of compression ratio.

Index Terms—Bengali Text, Data Compression, Content or Component Ranking, Data Model.

1 INTRODUCTION

There are hundreds and thousands of languages in this world. However, Bengali is the only language in the pages of history, which has been established at a cost of valuable lives of martyrs. Though dignity of the language is saved through the terrible blood shedding and extreme sacrifice, still now, in establishing Bengali as a powerful language in the digital world, the steps are not note-worthy. The standardization of Bengali text representation techniques or encoding methods has got a unique base through a long debate. Consequently, enhanced Efficient Data Representation Schemes (EDRS) has not been developed smoothly. The aspects of Bengali Text Compression are still now mostly dependent on ordinary data compression techniques, which often results in expansion of Bengali Text.

Compression is the process of reducing the size of a file or data by identifying and removing redundancy in its structure. Data Compression offers an effective approach of reducing communication costs by using available bandwidth effectively. Data Compression technique is generally divided into two categories; namely, Lossless Data Compression and Lossy Data Compression [1]. For Lossless schemes, the recovery of data should be exact. Lossless compression algorithms are to some extent essential for all kinds of text processing, scientific and statistical databases applications, medical and biological image processing, DNA and other biological data management and so on. However, a lossy data compression technique does not ensure the exact recovery of data. For image compression and multimedia data compression, there is a great use of lossy data compression [3], [6]. Our aim is to develop a Lossless Compression technique for compressing Bengali text.

Here, we have employed a new statistical model with a novel approach of integrating text ranking or component categorization scheme for building the model. A new dictionary matching scheme and static coding are used to obtain the proposed compression. Moreover, we have used a new theoretical concept of choosing the knowledgebase entries, which has facilitated us to obtain mentionable compression ratio using a small number of knowledgebase entries than other methods consuming less resource.

The prime aspect of the proposed scheme is to ensure compression of small and moderate volume of text rather than compressing huge text, since we aim to develop the scheme for battery powered smart devices rather than for giant computers. Short text compression for battery powered small devices is much more challenging because, the devices posses' low computational capability with small memory and limited processing power which limits the applicable schemes to be simplistic and lower resource consuming. In terms of the communication channel usage, the devices should be efficient enough to put minimal traffic while communicating. Though there are a wide variety of devices (especially mobile phones) are providing data-services to communicate in Bengali texts, upto our knowledge best, there is no sophisticated compression scheme in this concern. This motivates us to develop such a Bengali small text compression scheme to be used for power-constrained devices.

- ***S. A. Ahsan Rajon*** *is with the Computer Science and Engineering Discipline, Khulna University, Khulna-9208 and Faculty of Computer Science, KPCbd, Khulna-9100. Bangladesh. E-mail: ahsan.rajon@gmail.com .*
- ***Md. Rafiqul Islam*** *is with the Computer Science and Engineering Discipline, Khulna University, Khulna-9208, Bangladesh.*

2 BASIC CONCEPTSOVERVIEW

2.1 Overview of Text Compression

There are mainly two streams of text compression techniques, Statistical Modeling and Dictionary based techniques. The dictionary-based technique is the mostly adapted approach for English text compression. In Dictionary based compression, a text-base is formed in concern with specific text properties (i.e., frequency of text or syllable, pattern of frequent text etc.). A dictionary coder works by searching for a match between the text to be compressed and a string in the dictionary [6], [12]. Whenever a match is found, the text is substituted by a reference (i.e. pointer or index) to the string in the predefined dictionary. Examples of dictionary based compression schemes include Lempel-Ziv Algorithms (LZ, LZW).

In case of statistical modeling, the statistical encoder encodes each component separately taking their previous symbol (i.e. context) into consideration. LZW allows simplification of transmitting pointers of the phrasebook as initially one character strings are contained in the phrasebook [14] resulting a faster compression than Predictive coders. Context modeling is employed for text compression with integration of entropy coder. Generally saying, any compression technique comprises of two activities, construction of a model; that takes symbol probabilities into account and coding to form a minimal representation of each symbol or component. An efficient coder like Arithmetic Coder returns a binary stream for each set of symbol probabilities [12].

Compression techniques are in-generally characterized as static, semi adaptive and adaptive schemes. In semi adaptive approach, an initial pass over the source data is employed to collect statistics and in the next pass, the coding is performed. For this semi-adaptive approach, the sender first transmits the codebook and then sends encoded message. The static modeling proposes to use the same static statistics regarding symbol coding for both sender and receiver [15].

n-grams based text compression is an ideal example of semi static modeling. In this modeling, the total text to be compressed is considered as a series of strings of some length n [12].

Adaptive modeling is an approach of data compression where few hundred bytes are used for gathering the statistics. The main classes of Adaptive Coders include Lempel-Ziv Adaptive Dictionary Coders and Predictive Coders.

Predictive coders are concerned with probability of each symbol and the sequence of symbols preceding the current symbol (context). The number of preceding symbols (i.e. context) also determines the order of the model.

2.2 Characteristics of Bengali Text

Bengali Text Compression differs from English text compression from mainly two points of views. Firstly, the compression techniques involving pseudo-coding of uppercase (or lowercase) letters are not applicable for Bengali text. Secondly, in case of Bengali, we may employ specific mechanism of coding dependent vowel signs to remove redundancy, which is absent for the case of English. In Bengali, we have 91 distinct symbol units including independent vowels, constants, dependent vowel signs, two part independent vowel signs, additional consonants, various signs, additional signs and Bengali numerals etc. A detail list of Bengali symbols is available in [13]. Moreover, in Bengali we have a large involvement of conjuncts which also focuses a scope of redundancy removal.

Though English has got a fixed encoding base long ago, still now in practical applications, Bengali has not adapted unique encoding scheme. The use of Bengali Unicode has not yet got a massive use. This is really a great limitation for research in Bengali. Bengali text compression also suffers from the same problem.

3 Literature Review

Compression is an elementary concern of data engineering and management. Through the history of large-scale English text compression has already crossed its infancy, still now the total number of research activities on the specific field of Bengali text compression has not crossed any mentionable amount. The most recent study on text compression in [15] uses a context model with one at a time hashing based statistical model. Another fascinating research stated in [7] provides a concept of compressing small texts using syllables. The scheme provided in [11] makes use of greedy sequential grammar transform for text compression. A study regarding short text compression in [10] uses text-ranking scheme with syllable based dictionary matching for text compression.

The recent studies regarding text compression [6], [7] make use of syllables for compression of middle-sized files. They adapted well known Adaptive Huffman and Lempel-Ziv-Welch coding to use syllables instead of character. They provide elaborated definition of syllable emphasizing syllable as a sequence of sounds, which contains exactly one maximal subsequence of vowels. Words from non-letters are also considered as syllable. In order to improve a compression of alphabet of syllables or words, a database of frequent word-sets for each concerned language has been provided too. Initialization of compression algorithms are performed using words from the database. They code the words of source text from the defined database. The database of syllable form letters contains three thousand syllables approximately with condition to adding syllable to database was that, its frequency is greater than 1:65000. They do not employ any text ranking scheme for constructing the dictionary, and the number of initial entries can be reduced substantially by the use of proper substring weighting approaches.

The existing literature involving Bengali text compression in [2] proposed by Hossain *et al.* presents a comparative analysis of Bengali text compression with WinZip. They employ static Huffman coding for achieving the compression. The test-bed was adapted from a collection of Bengali newspapers. This scheme achieves reduced transmission cost in terms of compression and decompression time. For the bandwidth-aware applications, this scheme is not applicable because it requires submitting the codebook along with the

source text, which essentially puts greater load on Bandwidth.

Mukherjee *et al.* [4, 5] proposed a dictionary based compression scheme titled star encoding. According to this scheme, words are replaced with sequence of * symbol accompanied with reference to an external dictionary. The dictionary is arranged according to the length of words and is known to both sender and receiver. Proper sub-dictionary is selected by the length of the sequence of * symbols. Length Index Preserving Transformation (*LIPT*) is a variation of the star encoding by the same authors. This algorithm improves the Prediction by Partial Matching (*PPM*), Burrows-Wheeler Coding Applications (*BWCA*) and Lempel-Ziv (*LZ*) based compression schemes. Another related literature known StarNT works with ternary search tree and is faster than the previous. The first 312 words of the dictionary are the most frequently used words of the English language. The remaining part of the dictionary is filled up by words sorted by their length first and then by their frequency. This scheme also does not take the use of substring weighting approach. Moreover selection of the substrings are made only from, the point of view of aggregated frequency.

Prediction by partial matching (*PPM*) is a major lossless data compression scheme, where each symbol is coded by taking account of the previous symbols. A context model is employed that gives statistical information about the symbol with its context. In order to signal the decoder on the context, specific symbols are used by the encoder. The model order in PPM is a vital parameter of compression performance. However, PPM is computationally more complex and the overhead too is greater.

Block Sorting is an innovative compression mechanism proposed by Burrows and Wheeler in 1994 [1]. Block Sorting woks in three steps, permuting the input block one at a time through the use of Burrows Wheeler Transform (*BWT*), applying a Move to Front Transform (*MFT*) to each of the permuted blocks, and then entropy coding the output with a Huffman or Arithmetic Coder. Run Length Coding is also often introduced prior to any or all of the three stages [16].

4 PROPOSED BENGALI TEXT COMPRESSION SCHEME

In this paper, we propose a new dictionary for Bengali text compression. To facilitate efficient searching of the text, we employ term weighting or string ranking in indexing the dictionary entries. The total compression scheme is divided into two stages:

Stage 1: Building the knowledgebase.

Stage 2: Apply proposed text ranking approach for compressing the source text.

The test-bed is formed from standard Bengali text collections from various sources. We consider a collection of texts of various categories and themes (like news documents, papers, essays, poems and advertising documents) as the test bed. The documents were chosen as the representative of the corresponding groups. The groups were selected to demonstrate the interrelation of compression performance for different themes and motives of text. Considerations of the changed sentence structure and wording approach with the evolution of smart devices especially mobile phones is also a reason of choosing the various groups of texts with varying size. It is necessary to mention that, though a few collections of field specific text collection are available, still now no sophisticated Bengali text compression evaluation test-bed is available. As data compression and especially dictionary based text compression greatly involves the structure, wording and context of texts, a collection of different types of texts is a must for evaluating the compression. In constructing the dictionary, we use the test-text-bed of 109 files varying from 4KB to 1800KB.

Construction of the Knowledgebase

The text-base is employed in two steps.

Firstly, we calculate the statistics of the text taking context and frequency of text into consideration and

Secondly, we insert the symbol(s) or text in the dictionary selected using collected statistical data with threshold.

Let, the test-bed contains a total of n documents. Again, assume that, document di contains tci number of distinct characters. The total number of distinct words in document di is assumed to be twi. The statistics also follows that, character ci occurs fci times and the number of occurrence of word wi is fwi times. At first, we rank the individual symbols in terms of their occurrence. Here, we use the term rank as the frequency of each character. We consider this total statistics involving the individual symbols as level one index. Then we proceed towards substrings with length greater than two. Each substring is considered from mutual directions. Firstly we consider the string for ranking from forward direction and then from backward direction. For words with length greater than seven symbols, it is partitioned into several partitioned-substrings with the length of multiplier of seven. Starting from the initial symbol, we proceed to rank each symbol, where for symbols positioned at p with respect to the concerned word of length u where $1< p \leq u \leq 7$. This ranking is expressed as the summation of ranks of previous symbols. We take the previous rank into concern because, if we simply consider the frequency of the substring, for discrete and sophisticated documents, the motive and sense of the document *i.e.*, repeated terms of the document would highly influence the ranking and fluctuate the overall ranking and indexing scheme. Nevertheless, considering the context of the symbol, that is, taking the rank of the symbols into consideration, which is embodied into the current symbol provides a cumulative statistics and context of the current substring and hence makes the dictionary an unbiased collection. In such a way, we completely rank the dictionary. It is noteworthy that, we consider seven as the threshold, based on the simple hypothesis that, the average length of a Bengali word may be seven characters.

The next step involves selecting 256 entries from the total selection. If there are a total of e entries in the resultant statistics where $e > 256$, and for levels $t=1, 2, 3, \ldots,7$ if there are

a total of ht entries, then the entries are sorted over for each level. For level t, (256 / e * ht) entries are selected in ascending order and in the temporary database the entries with their corresponding rank (in percentage with respect to total words for level > 1 and with respect to total symbols for level =1) and level id is stored.

In the same way, all the documents are ranked and stored into temporary database. The ending step of constructing the dictionary involves selecting 256 entries from the combined database by relational operations using the same criteria for choosing 256 entries from each temporary database.

The next step of the proposed scheme, *i.e.* application of proposed text ranking approach for compressing the source text constitutes of primarily two stages. In the first stage, the source text is successively compared with the entries of the knowledgebase starting from the maximum level. For any successful match, the substrings are marked with the corresponding words grabbed from the dictionary. If there is no successful match for that specific level, the substrings are converted to the level below it. By repeating this step, the total files are converted, since, level one is composed of single characters and symbols.

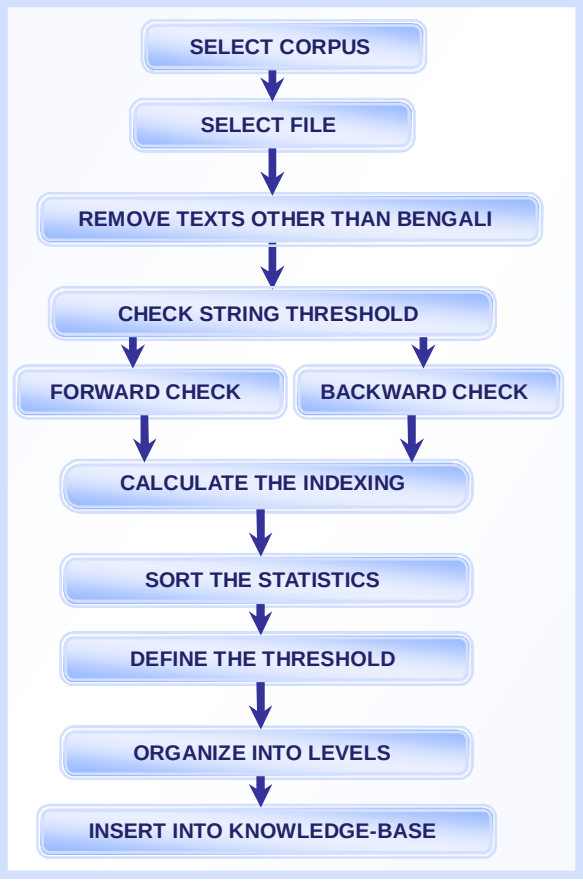

Fig. 1 Construction of the knowledgebase

5 Performance Analysis of Proposed Bengali Text Compression Scheme

Though it is a general idea that compression and decompression time should have an inter-relation, the proposed scheme demonstrated a little exception. The points behind that may be summarized through the following discussions.

5.1 Performance Analysis of Compression Process with respect to time

Let the total number of training entries for the statistical model be N_g , where N_g is a non-negative integer and the maximum level for statistical modeling is L. The first level of the statistical model will must contain the single characters, where the total number of distinct character is l1. For levels *1, 2, 3, …., n* the total number of distinct multi-gram entries are $l_1, l_2, l_3, \ldots\ldots, l_n$ respectively.

When any text is to be compressed, it is hierarchically compared with each level of statistical model starting from the highest order. If there is any match, the corresponding static coding for multi-gram entry is assigned for the text. If the multi-gram entry is not found throughout the level, it is forwarded to the next level. This assignment uses efficient searching procedures. Let the code m is found at the *i*-th level with offset k resulting a search cost of $S_m (l j) + k_m$, where $k_m < l_i$ and, $j = L, L\text{-}1, \ldots\ldots, i\text{-}1$ with respect to search space. Here j limits from *L* to *(i-1)* instead of *i* in decreasing order because, as we find the code in somewhere of *i*-th level not requiring to search the whole element-space of the *i*-th level, rather searching through an offset value k for *i*-th level, the overall search-space is *L* to *(i-1)*. That is why, for the above consequences, the total searching appears search overhead for *(i-1)* number of levels with additional search overhead of *k* elements. Here the term "*search overhead*" means the complexity of searching as well as other related computational requirements. When the code matches, it is placed in output stream as character representation. This step requires padding the bit-stream and then conversion into character stream. Assume that, the process of overall conversion for each successful entry occurs with the overhead *B*. That is, for any multi-gram matching, the required overhead is,

$$C_1 = \sum_{j=L}^{i-1} (S_1(l_j)) + k_1 + B_1$$

Similarly,

$$C_2 = \sum_{j=L}^{i-1} (S_2(l_j)) + k_2 + B_2$$

And,

$$C_n = \sum_{j=L}^{i-1} (S_n(l_j)) + k_n + B_n$$

In such a way if a total of n multi-grams are identified and then encoded, the required resultant number of operations in compression process is:

$$T = \sum_{y=1}^{n} C_y = \sum_{y=1}^{n} \sum_{j=L}^{i-1} (S_y(l_j)) + \sum_{y=1}^{n} k_y + \sum_{y=1}^{n} B_y$$

5.2 Performance Analysis of Decompression Process with respect to time

For the decompression process, the text to be decompressed is converted into binary stream. If the largest code is of length c_{max} and the smallest code is of length c_{min} then the decompression process will start the searching with the c_{max} number of bits and search through the codes up to c_{min} bits by reducing one bit per step for unsuccessful match. It is necessary to mention that, the codes with same bit length does not essentially comprise any specific level. So, to reveal the character representation for each entry d if a switch of *h* levels are required, for $c_{min} \le h \le c_{max}$ where the maximum level is *p* with the matching offset for corresponding level k_d, and the assignment of the code with character representation for each successful match requires an overhead of B_d, then the overall requirement for comparing through the each level setting results (= Overhead of Searching through level + Overhead of Searching through offset + Overhead of Representation).

For detecting first character the overhead is,

$$E_1 = \sum_{q=p}^{h} (S'_1(l'_q)) + k'_1 + B'_1$$

.

Similarly, for detecting the second character, the level-wise overhead will be:

$$E_2 = \sum_{q=p}^{h} (S'_2(l'_q)) + k'_2 + B'_2$$

.

Similarly,

Here, *p* = maximum level, and *S/* is a function that denotes the overhead for searching in element space provided as parameter of the function and *h* = minimum level. The computation progresses through *p, p-1, p-2,……, h+2, h+1, h* . Here the subscript *n* is used to denote the level-wise overhead for detecting one character representation with respect to level.

In order to detect a single multi-grams *f*, the total search-overhead with respect to search space for level-wise calculation is, $\sum_{q=p}^{h} (S'_f(l'_q)) + k'_f$

because we are to start with maximum level *p* and then proceed decreasingly towards the downward levels *h* (as explained above). If *S/* is the search-overhead function, then searching from level *p* to *h* will result $\sum_{q=p}^{h} (S'_f(l'_q))$ where f is the multi-grams, which is being revealed. For the matching level, as only a partial number of elements are to be searched, the offset k is used to denote the offset for language *L*.

After checking through the levels, the procedure follows searching through the bit-wise statistics for any unsuccessful match in level-wise statistics. If there are a total of *u* bit-phases, we are to perform searching through the search-space consisting of starting from the maximum bit phase to the minimum bit phase in decreasing order. Because of any unsuccessful match in any bit-phase, a bit switch is performed and level wise calculation for that level is forwarded. That is, an overhead of $\sum_{b=d}^{g} (E_b)$ is incurred for each level-wise analysis. Consequently, the overhead of unit step will be, $C'_1 = \sum_{b=d}^{g} (E_b)$

Here, d and g are maximum and minimum bit phases respectively and $d \ge g$. Substituting the value of E_b, we get,

$$C'_1 = \sum_{b=d}^{g} \sum_{q=p}^{h} (S'_{1,b}(l'_q)) + \sum_{b=d}^{g} k'_{1,b} + \sum_{b=d}^{g} B'_{1,b}$$

Similarly, we get,

$$C'_2 = \sum_{b=d}^{g} \sum_{q=p}^{h} (S'_{2,b}(l'_q)) + \sum_{b=d}^{g} k'_{2,b} + \sum_{b=d}^{g} B'_{2,b}$$

And,

$$C'_n = \sum_{b=d}^{g} \sum_{q=p}^{h} (S'_{n,b}(l'_q)) + \sum_{b=d}^{g} k'_{n,b} + \sum_{b=d}^{g} B'_{n,b}$$

Here we use the subscript *1* with *k* and *B* in order to mean that, the calculations are for detecting unit code only where the calculation is performed starting from *d* to *g* in decreasing order, that is, in the order of *d, (d-1), (d-2), …… , (g+2), (g+1), g*.

If we are to reveal n number of codes, then the total overhead becomes:

$$T' = \sum_{y=1}^{n} C'_y$$

As for each bit wise overhead calculation, we must include level-wise calculations; we may omit the subscript notation for search overhead function for simplicity,

$$T' = \sum_{y=1}^{n} \sum_{b=d}^{g} \sum_{q=p}^{h} (S'_{y,b}(l'_q)) + \sum_{y=1}^{n} \sum_{b=d}^{g} k'_{y,b} + \sum_{y=1}^{n} \sum_{b=d}^{g} B'_{y,b}$$

6 Experimental Results and Discussions

The proposed compression is basically a multi-stage compression technique. In this scheme, we use a new knowledgebase for Bengali text compression. This knowledgebase is formed by analyzing the test bed discussed in section III. The knowledgebase is considered as a static base.

In the initial step, the source text (Bengali) is passed into Unicode converter. For the traditionally used encoding-based Bengali texts, it is converted into Unicode, whereas Unicoded source Bengali text is simply passed over the next step. The second step involves matching the proposed dictionary (*i.e.* knowledgebase). We employ a text-base of only 256 entries. As the proposed dictionary contains small number of entries in the knowledgebase, the search-space is

greatly reduced resulting a faster retrieval of index data from the dictionary for the querying string or substring. Consequently, we get a stream of integers representing the source text, which may be reduced at this step. It is then passed as the input for Arithmetic Coder after re-converting the stream of integers into corresponding character (and symbol) representation.

The compression ratio is a metric to describe how many compressed units are required to describe one unit of data. The lower the presented value shows better compression. A general observation is that higher modes lead to better compression ratios even if the difference with higher orders becomes smaller.

We have also analyzed the performance of our proposed scheme with existing domain independent text compaction scheme like "A modification of Greedy Sequential Grammar Transform based Universal Lossless data Compression" (*mGSGT*) by R. Islam *et al.* [8] and "Word-Based Block - Sorting Text Compression" (*WBBSTC*) by Isal and Moffat [16]. The performance analysis was performed in a quad-core 2.0 GHz personal computer with 1.0 GB RAM with threading support. Object Oriented Programming Language JAVA was used to simulate the total scheme.

TABLE 1: COMPARISON OF COMPRESSION RATIO

File Name	Proposed Scheme	mGSGT	WBBSTC
Article	3.748	3.89	3.96
Poem	4.014	4.48	4.51
Advertise	3.928	4.36	4.34
Speech	3.624	3.81	3.88
News	3.818	4.11	3.87
SMS	3.416	4.76	4.59
Email	3.718	3.87	4.01
Particulars	3.941	4.21	4.04
Story	3.371	3.59	3.77
Report	3.749	3.76	3.79

Our proposed scheme for knowledgebase formation is also applicable for uni-language text compression of languages other than Bengali. Since the proposed scheme builds the knowledge-base in a hierarchical manner with provision to define the levels of hierarchy, re-defining the span of levels may optimize the compression effectiveness. Since, the proposed scheme demonstrates a unified way

Though the proposed scheme demonstrates better performance for compression of Bengali text, the scheme is not efficient for compression of multi-lingual text. Future works may be dedicated for attaining a multilingual text compression scheme including Bengali by applying the core concept of proposed scheme. Let the source text comprise of language l1 and l2 with total character set of a1 and a2. In order to code any source text containing components from both languages, the knowledgebase will must comprise of all the unit components of l1 and l2 to facilitate the coding-ability to the unit components. This requires a minimum knowledge-base space of a1+a2. Again, level 2 to level n for each language will also require certain spaces for formation of the knowledge-base. If the summation of spaces exceeds the threshold value for optimal coding using static-coding scheme, there may be negative fluctuation between the expected performance and actual performance.

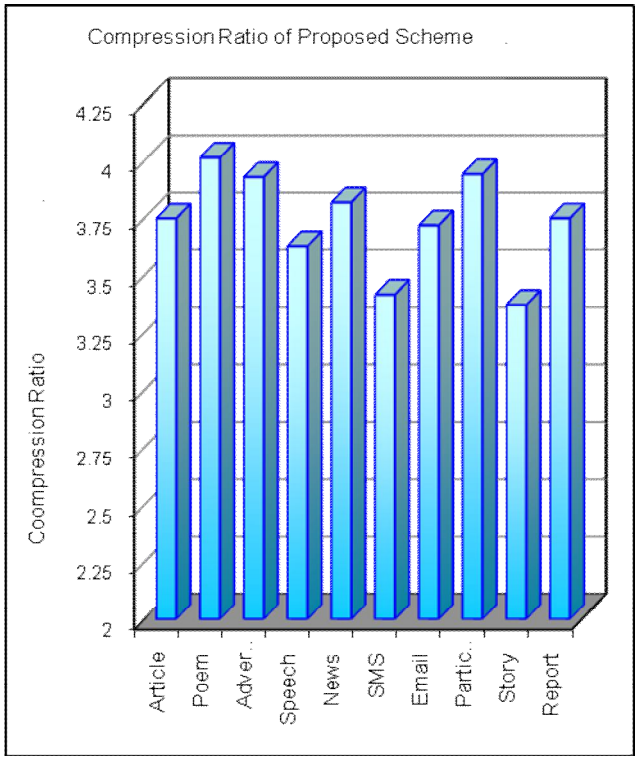

Fig. 2 Compression Ratio of the proposed scheme

The proposed scheme achieves better compression ratio for Bengali text compression by means of efficient dictionary-mapper. The proposed dictionary is completely different form conventional dictionaries as it employs a modified scheme of selection of dictionary entries with enhanced ranking criteria. Such adaptation is a first for Bengali text. The constraint of making the dictionary span fixed ensures optimal search space. Moreover, our proposed scheme is designated for small to medium sized text files, which is of the most necessary span for widespread use.

7 CONCLUSIONS AND RECOMMENDATIONS

The proposed scheme is one of the initiating steps of Enhanced Text Representation Scheme for Bengali Text. In this step, a novel approach of constructing data compression dictionary has been proposed which is also an innovative approach of Bengali text compression. We have impressive outcomes of the proposed approach in terms of compression time, compression ratio and overall overhead requirements. As the proposed scheme is adapted for both conventional encoding and Unicode standard, it may be employed very easily for any Bengali text compression. Being a low-memory consuming one, the proposed approach may also be adapted for text compaction in small memory devices.

Communication of Bengali Small Text Message may also be immensely facilitated with the presented approach of Bengali text compression. The proposed scheme is also to some extent an initialization of Bengali text compression approaches with a few pathfinders. .

REFERENCES

[1] M. Burrows and D. J .Wheeler. A block sorting lossless data compression algorithm. Technical report, Digital Equipment Corporation, Palo Alto, CA, 1994.

[2] Md. Sazzad Hossain and R.C. Debnath, "A Comparative Study of Bangla Text Compression with Winzip", Information Technology Journal, 3 (1): 93-94, 2004.

[3] A. Moffat, R. M. Neal and I. H. Witten "Arithmetic coding revisited". ACM Transactions on Information Systems, 16:256–294, 1998.

[4] F. Awan and A. Mukherjee, "LIPT: A Lossless Text Transform to improve compression", Proceedings of International Conference on Information and Theory : Coding and Computing, IEEE Computer Society, Las Vegas Nevada, 2001.

[5] H. Kruse and A. Mukherjee, "Preprocessing Text to Improve Compression Ratios", Proceedings of Data Compression Conference, IEEE Computer Society, Snowbird Utah, 1998, pp. 556.

[6] Lansky, J, Zemlicka, M., "Compression of a Dictionary". Proceedings of the DATESO 2006 Annual International Workshop on DAtabases, TExts, Specifications and Objects. CEUR-WS,Vol.176,(2006)11-20.

[7] Lansky, J, Zemlicka, M.: "Compression of Small Text Files Using Syllables". IEEE Data Compression Conference-2006, IEEE CS Press, Los Alamitos, CA, USA (2006) 458.

[8] Md. Rafiqul Islam, Sajib Kumar Saha, Mrinal Kanti Baowaly. "A modification of Greedy Sequential Grammar Transform based Universal Lossless data Compression". Published in Proceedings of 9th International Conference on Computer and Information Technology (ICCIT 2006), 28-30 December, 2006, Dhaka, Bangladesh.

[9] S. Rein and C. Guhmann, "Arithmetic coding–a short tutorial", Wavelet Application Group, Technical Report, April 2005.

[10] Md. Rafiqul Islam, S. A. Ahsan Rajon and Anonda Podder, "Small Text Compression for Smart Devices", In the proceedings of International Conference on Computer and Communication Technology, ICCIT' 2008, Khulna, Bangladesh.

[11] E. H. Yang and J. C. Kieffer. "Efficient universal lossless data compression algorithms based on a greedy sequential grammar transform. Part one: Without context models". IEEE Transactions on Information Theory ,46(3): 755–777, 2000.

[12] J. Able "Universal text preprocessing for data compression", IEEE 2005.

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

[14] Phil Vines and Justin Zobel: "Compression Techniques for Chinese text", Department of Computer Science, RMIT, Melbourne, Australia.

[15] Stephan Rein, Clemens Guhmann, Frank H. P. Fitzek: "Compression of Short Text on Embedded Systems", Journal of Computers: Volume 1, No: 06, September 2006.

[16] R. Yugo Kartono Isal and Alistair Moffat, "Word - Based Block - Sorting Text Compression", Proc. of 24th Australian Computer Science Conference, Gold Coast, Australia, pp. 92-99.

[17] Md. Rafiqul Islam and S. A. Ahsan Rajon, "An Enhanced Short Text Compression Scheme for smart devices", Journal of Computers, Vol. 5, No. 1, January 2010, pp. 49-58.

Prof. Dr. Md. Rafiqul Islam obtained Master of Science (M. S.) in Engineering (Computers) from Azerbaijan Polytechnic Institute (Azerbaijan Technical University at present) in 1987 and Ph.D. in Computer Science from Universiti Teknologi Malaysia (UTM) in 1999. His research areas include design and analysis of algorithms and Information Security. Dr. Islam has around 75 papers related to these areas published in national and international journals as well as in referred conference proceedings. He is currently working as a professor of Computer Science and Engineering Discipline, Khulna University, Bangladesh.

S. A. Ahsan Rajon is currently working as a Senior Lecturer of Department of Computer Science, KPCbd, Khulna-9100, Bangladesh. Engr. Rajon is also an Adjunct Faculty of Computer Science and Engineering Discipline, Khulna University, Bangladesh. After completion of B.Sc.Engineering from CSE discipline, Science, Engineering and Technology School, Khulna University, Bangladesh in April 2008, he was appointed in his native discipline. Rajon has made three journal and eight conference publications in International conferences and Journals. He is also a reviewer of Journal of Engineering and Technology Research and International Journal of Information Systems. His research interest includes data engineering and management, information systems and ubiquitous computing. Currently he is working on robotics. He is a member of Institute of Engineers, Bangladesh (IEB). For more information about Rajon, please visit: http://sites.google.com/site/ahsanrajon