An Enhanced Short Text Compression Scheme for Smart Devices

Md. Rafiqul Islam
Computer Science and Engineering Discipline, Khulna University, Khulna, Bangladesh.
Email: dmri1978@yahoo.com

S. A. Ahsan Rajon
Computer Science and Engineering Discipline, Khulna University, Khulna, Bangladesh.
Email: ahsan.rajon@gmail.com

Abstract — **Short Text Compression is a great concern for data engineering and management. The rapid use of small devices especially, mobile phones and wireless sensors have turned short text compression into a demand-of-the-time. In this paper, we propose an approach of compressing short English text for smart devices. The prime objective of this proposed technique is to establish a low-complexity lossless compression scheme suitable for smart devices like cellular phones and PDAs (Personal Digital Assistants) having small memory and relatively low processing speed. The main target is to compress short messages up to an optimal level, which requires optimal space, consumes less time and low overhead. Here a new static-statistical context model has been proposed to obtain the compression. We use character masking with space integration, syllable based dictionary matching and static coding in hierarchical steps to achieve low complexity lossless compression of short English text for low-powered electronic devices. We also propose an efficient probabilistic distribution based content-ranking scheme for training the statistical model. We analyze the performance of the proposed scheme as well as the other similar existing schemes with respect to compression ratio, computational complexity and compression-decompression time. The analysis shows that, the required number of operations for the proposed scheme is less than that of other existing systems. The experimental results of the implemented model give better compression for small text files using optimum resources. The obtained compression ratio indicates a satisfactory performance in terms of compression parameters including better compression ratio, lower compression and decompression time with reduced memory requirements and lower complexity. The compression time is also lower because of computational simplicity. In overall analysis, the simplicity of computational requirement encompasses the compression effective and efficient.**

Index Terms — **Short Text Compression, Syllable, Statistical Model, Text-ranking, Static Coding, Smart Devices.**

I. Introduction

Twenty-First century is the age of information and communication technology. Science through its marvelous achievements has converted this world into information and communication based global village. The prime aspect of present technology is to ensure a better way of communication throughout the world in a more convenient, easy and cost-effective way. With the aspects of cost, facility and reliability a new trend of introducing small sized devices with some sorts of computing and communicating power have established its place in the arena of research. With the voyage of introducing smart devices, the challenge of adorning them with greater and effective use has come into question. It is now a great concern to embed maximum applications within these smart devices where it is an extreme problem to provide a low-complex and low-memory consuming version for smart devices of some prime necessary applications like data compression, which generally requires large memory and greater processing speed. Mobile communication that is a great gift of modern technology introducing the era of digital communication also suffers from the same limitation. Though crossing the boundary of voice communication, short messages communication has established its robust place in the arena of digital communication, Short Message Service (SMS) providers (usually Telecommunication Companies) have a constraint that each message should be not more than of 160 characters. This constraint is really a great limitation for frequent communication using SMS. In order to overcome this limitation, compression of the short message is a well policy. That is why; our aim is to make "short" messages "shorter", expressing "larger" feelings in "smaller" expenses.

Here we introduce a scheme of compressing short English text for smart devices like cellular phones and wireless sensors having small memory and relatively low processing speed communicating with lower bandwidth *i.e.* channel capacity. We have employed a new statistical model with a novel approach of integrating text ranking or component categorization scheme for building the model. Modified syllable based dictionary matching and static coding is used to obtain the compression. Moreover, we have employed a new theoretical concept

Contact Author: S. A. Ahsan Rajon
E-Mail: ahsan.rajon@gmail.com

of choosing the multi-grams, which has facilitated us to obtain mentionable compression ratio using a small number of knowledgebase entries than other methods, consuming less resource. Besides of experimental results we have provided a comprehensive theoretical analysis of compression ratio of proposed scheme with similar existing scheme.

II. Related Literature

Though a number of researches have been performed regarding large-scale data compression, in the specific field of short text compression, the number of available research work is small. Business issue of mobile phone service providers may be indicated as a reason behind the unavailability of research material. The following sections give a glimpse of the most recent research developments on short text compression issues for small devices.

A. Compression of Short Text on Embedded Systems

The recent literature regarding short text compression titled "Compression of Short Text on Embedded Systems" by Rein *et al.* [1] proposes a low-complexity version of PPM (Prediction by Partial Matching). A hash table technique with one-at-a-time hash function is employed in this method to design the data structure for data and context model. They use statistical models with hash table lengths of 16384, 32768, 65536 and 131072 elements requiring two bytes for each element, which result an allocation of 32, 64, 128 and 256 Kbytes of RAM respectively. If this memory requirement may be substantially decreased, we may achieve more efficient compression and hence may make the scheme usable to even very low-quality cellular phones. Another concerned approach by Rein *et al.* is "Low Complexity Compression of Short Message" [2] with Low Complex and Power Efficient Text Compressor for Cellular and Sensor Networks [3] are variations of [1].

B. Compression of Small Text Files Using Syllables

"Compression of Small Text Files Using Syllables" proposed by Lansky *et al.* [4, 5] concerns on compressing small text files using syllables. To implement their concept they created database of frequent syllables. Here, condition for adding syllable to database is that, its frequency is greater than 1:65000. In this scheme, the primary knowledge-base size is more than 4000 entries initially. For low memory devices, it is obviously difficult to afford this amount of storage as well as to facilitate a well suited mechanism of searching; which leads our proposed scheme to redefine the knowledge-base span as short as possible and hence to reduce the scope of loosely choosing the syllables or n-grams. Moreover, in formation of the syllables, space is not considered with any special concern. But, as in any text document, it is a common assumption that, at least 20% of the total characters may be spaces, it may be a good idea to have specific consideration of syllable involving spaces. In [4, 5], all the training syllable entries are stored without any categorization. This often results for coding redundancy, which can be handled by integrating text ranking or component categorization scheme with syllable selection.

C. Modified Greedy Sequential Grammar Transform based Lossless data Compression

The model proposed by M. R. Islam *et al.* [6] uses the advantages of greedy sequential grammar transform with block sorting to compress data. However, this scheme is highly expensive in terms of memory consumption and thus not suitable for low memory devices.

D. Two-Level Directory Based Compression

Dictionary based text compression techniques are an important and mostly adapted data compression schemes. A dictionary based text preprocessing scheme titled TWRT (Two-level Word Replacing Transformation) has been proposed by P. Skibinski [7]. They use several dictionaries and divide files into various kinds, which improve the compression performance.

TWRT can use up to two dictionaries, which are dynamically selected before the actual preprocessing starts. For some types of data like programming languages, references etc. first level dictionaries (small dictionaries) are specified whereas second level dictionaries (large dictionaries) are specific for natural languages (e.g., English, Russian, French). While concerned with any source text, if no appropriate first level dictionary is found, then it is not used. Selection of the second level dictionary is analogous. When TWRT has selected only the one (the first or the second level) dictionary, it works like WRT (Word Replacing Transformation) [7]. If TWRT has selected both the first and the second level dictionaries, then the second level dictionary is appended after the first level dictionary. That is, the dictionaries are automatically merged. If the same word exists in the first and the second level dictionaries, then the second appearance of word is ignored to minimize length of code-words. Only the names of the dictionaries are written in the output file, so the decoder can use the same combination of the dictionaries.

TWRT preprocesses the input file step by step with all dictionaries and finally to choose the smallest output file. Nevertheless, this idea is very slow. They propose to read only the first *f* (*e.g.,* 250) words from each of *n* dictionaries (small and large) and create one joined dictionary which is completely impossible to afford for low-memory devices. If there are same words in different dictionaries, then all occurrences of this words are skipped which is too an extremely infeasible technique for smart device platform-aware compression. The main problem for TWRT is selection of the dictionaries before preprocessing, which hampers the processing time for concerned devices [3]. Moreover the dictionary length is too huge to be used for small memory devices. For small text compression it is also not feasible and to some extent efficient to have field-specific dictionary. Above all, being the code words are determined on the fly, it is great doubt to cope with the memory and energy constraints as well as time consumptions of the low-memory devices.

E. Other Schemes

H. Kruse and A. Mukherjee [8, 9] proposed a dictionary based compression scheme named 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 PPM, BWCA and LZ based compression schemes [9]. 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 lengths first and then by their frequencies. This scheme also does not take the use of substring weighting approach. Moreover, the scheme requires that, the dictionary should be transmitted first in order to set up the knowledge-base. It is completely in-feasible to be used for compressing small texts for low-powered and small memory devices.

Prediction by partial matching (PPM) is a major lossless data compression scheme, where each symbol is coded by taking account of the previous symbols [9]. 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 [1, 9, 12].

In [1] the compression starts for text sequences larger than 75 Bytes, and in [10] the starting point is 50 Bytes. If it is possible to make the lower threshold value into less than ten characters, the compression may really be a "very small text file" supported one that may place a new milestone in very small text file compression ensuring "short text gets shortest". Our prime aim is to design such an effective and efficient "very short text compression" scheme.

III. Short Text Compression For Smart Devices

The prime concern of this paper is to implement a lossless compression of short English text for smart devices in a low complexity scheme. The idea behind this is to make it possible to communicate more efficiently by making utilization of minimum required bandwidth of wireless and sensor networks for small and embedded devices. More precisely saying, the main concern is to develop a low-complexity compression technique suitable for low-power consuming smart devices with small-memory; especially for wireless sensors and mobile phones. The proposed scheme is concerned with two parts. The first one consists of training the statistical model and the second provides a compression-decompression technique. Specifically, in analyzing step we count the frequency of represent-able ASCII characters. Then, the proposed scheme proceeds by identifying the syllables of length two, three, four and five. We consider <space> as a distinct vowel and include this while counting the frequency of syllables, that is, searches for <space> either at the beginning or ending of syllable. In the step of boosting the statistical model, the substrings (which were not grabbed in the phase concerning syllables) with length two to five are considered and the frequency of each are counted. In the second step, we employ the provided text-ranking scheme for each entry and calculate the entry index. For entries with same index, we simply sort them. In the phase of choosing entries, emphasis is give on probability distribution based text ranking, which is computed with the help of its neighbor characteristics.

As in most cases, it is unusual to have frequent match of substrings with length more than four characters, maximum of five (extra one for <space>) levels has been considered to train the statistical model. In each level, multiple entries with same weightage are simply sorted over. Resultant entries are assigned with non-conflicting binary stream. When we are to compress any text, the input text is successively compared with the statistical model and for any match, the binary stream is returned as output and for mismatch in any level, the level below it is forwarded.

The compression and decompression are expected to be performed in the following manner.

A. Compression Process

In the first step, we plan to employ Modified Multi-grams or syllable matching proposed by Lansky *et al.* [4, 5]. A static dictionary based compression scheme uses approximately the same concept as that of character masking. It reads input data and searches for symbols or groups of symbols that are previously coded or indexed in the dictionary. If a match is found, a pointer or index into the dictionary can be output instead of the code for the symbol. Compression occurs if the pointers or index requires lower space (in terms of memory usage) than the corresponding fragments [1, 12, 15]. Though it is the basic idea behind multi-grams, we use it in a slightly modified fashion. The Knowledgebase for the multi-grams is constructed with the help of the statistics obtained by analyzing the corpuses.

In the second step of compression process, static coding has been used. However, here the modification is made in such a way that, in spite of calculating on-stage codes predefined codes are employed in order to reduce the space and time complexity. The codes for dictionary entries are also predefined. The total message is encoded by a comparatively small number of bits and hence, we get a compressed outcome. The prime fact of static coding is that, in case of dynamic coding we are to calculate the frequencies in an elegant manner. Moreover, as the dynamic codes are determined and the lengths are varied through the frequencies of the symbols it is a must to submit the codes along with the compressed data. This is a limitation of dynamic coding if the total arena of compression is low spanned and the total number of symbols is not huge. Compression of short message also suffers in this context while using other similar semi-dynamic coding. Consequently, it motives to move towards static coding scheme to obtain compression of short messages. For this concerned study, we have analyzed the corpora- Bib, Book1, Book2, News, Paper1, Paper2, Paper4 and Paper5 from Calgary Corpus and Canterbury Corpus [16]. We have also analyzed 124 collections of small text for the same. A detail overview of the texts is presented in [10, 15].

B. Decompression Process

The decompression process is performed through the following steps:

Step 1: Grab the bit representation of the message,

Step 2: Identify the character representation,

Step 3: Display the decoded message.

As all the symbols are to be coded in such a fashion that, by looking ahead several symbols (Typically the maximum length of the code) we can distinguish each character (with the attribute of Static Coding). In step 1, the bit representation of the modified message is performed. It is simply analyzing the bitmaps. The second step involves recognition of each separate bit-patterns and indication of the characters or symbols indicated by each bit pattern. This recognition is performed on the basis of the information from fixed encoding table used at the time of encoding. The final step involves simply representing i.e. display of the characters recognized through decoding the received encoded message.

IV. Theoretical Aspect of Training The Proposed Statistical Model

The proposed scheme achieves better compression ratio with relatively low complexity by means of computational simplicity and effective expert model, which is used to train the statistical context. Firstly, the prime modification is performed in the syllable selection section of [4, 5] proposed by Lansky *et al.* In their papers, only syllables were considered by defining maximal subsequence of vowels including pseudo-vowel '*y*'. Here our proposal is to consider <*blank space*> as a *Prime Syllable*. Using <*space*> as a *prime syllable* may dramatically reduce the total number of characters needed to represent the message through sophisticated encoding. Secondly, the paper [4, 5] does not clearly express the criteria of choosing the training syllables for the model. The term used in [4, 5] to define the criteria of choosing syllables was simply "frequentness". However, we employ a new theoretical perspective on "Text Ranking or Component Ranking" method to choose the syllables for training our proposed model. We, in the first step count the frequency of each vowel from the standard Text Calgary Corpora. In the second step, we count the number of vowels having space either at *(i - 1)* or *(i + 1)* position with itself at position *(i)*. The reason of adding the with-space vowels is simply to increase the weightage of the corresponding vowels. These entries having vowels are also inserted in the knowledgebase as separate entity. In this step, we also count the statistics of consonants in order to build the dictionary or multi-grams entries. In the step of extending primary knowledgebase, proposed in papers [4, 5] the criteria was that, the frequency would be 1:65000, which results a knowledgebase size of more than 4000 entries initially. For low memory devices, it is obviously difficult to afford this amount of storage as well as to facilitate a well suited mechanism of searching, which leads our proposed scheme to redefine the knowledge base span as short as possible and hence to reduce the scope of loosely choosing the multi-grams. In the phase of choosing multi-grams, we give emphasis on Probability Distribution, which is computed with the help of its neighbor characteristics.

A new text weighting or component ranking scheme has been employed to select the multi-grams that facilitates us to efficiently construct the knowledgebase [11, 15]. This developed novel text weighting scheme is employed with an aim to get the knowledgebase. The proposed text weighting or component ranking is obtained through the following equation:

Suppose we choose a multi-grams D consisting of characters $C_1\ C_2\ C_3 \ldots\ C_n$ of length n. Thus,

$$\partial(D_{C_1, C_2, C_3, \ldots\ldots, C_n}) =$$

$$\sum_{i=1}^{n} (\partial(D_{C_i}) - 1)\ \partial(D_{C_1, C_2, C_3, \ldots\ldots, C_{n-1}}) + \lambda_{C_1, C_2, C_3, \ldots\ldots, C_n}$$

for $i > 1$ and,

$\partial(D_{C_i}) = \lambda(C_i)$ for $i = 1$, where $\lambda(C_i)$ is the frequency of the character C_i with assumption that each character of the alphabet must exist in the training data. And the value of $\partial(D_\varphi)$ indicates the multi-grams index of character D_φ.. Here, we refer the multi-gram index obtained from $\partial(D_{C_1, C_2, C_3, \ldots\ldots, C_n})$ as the resultant text-weightage for the text $C_1\ C_2\ C_3 \ldots\ C_n$.

For example, we want to have the probability distribution of multi-grams "*and*".
The steps will be:
Let $\partial(a) = x_1$,
$\partial(n) = x_2$,
$\partial(d) = x_3$,
$\partial(an) = x_4$ and,
$\lambda(C_{and})$ = *Frequency of 'and' in the training document = x5.*
Therefore, $\partial(and) = (\partial(a)-1) + (\partial(n)-1) + (\partial(d)-1) — \partial(an) + \lambda(C_{and}) = x_1 + x_2 + x_3 — x_4 + x_5$

Static coding is defined as a set $S = \{B_1, B_2, B_3 ......, B_{n-1}, B_n\}$ of binary streams, where each element of the set $B_1, B_2, B_3 ......, B_{n-1}, B_n$ are uniquely identifiable providing that, $B_1, B_2, B_3 ......, B_{n-1}, B_n$ are not necessarily to be of equal bit lengths. As all the elements of the sets supports our three basic points of concern- identifiability, uniqueness and variability of any encoding scheme, we are interested to use static coding. This theoretical aspect let us do not consider updating the model throughout the compression, because, firstly, updating a larger knowledgebase using a very short data is not computationally affordable and effective, secondly, a larger portion of the source code may be saved if the update is avoided resulting faster execution [1, 3, 11] and finally, if the training statistical data are not permitted to be updated as well as to be expanded, a constant and consistent memory optimization for the overall compression process may ensured. That is, there is no possibility to expand the knowledgebase arbitrarily and thus there is no risk of arising the overloaded memory problem or out-of-memory problem. As the knowledgebase is not updated, the use of static coding is also perfect for the same.

V. Performance Analysis

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.

A. Performance Analysis of Compression Process

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 must contain the single characters, where the total number of distinct character is l_1. For levels $1, 2, 3, ..., n-1, n$ the total number of distinct multi-gram entries are $l_1, l_2, l_3, ..., l_{n-1}, l_n$ 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, $j = L, L-1, ..., i-1$ with respect to search space. Here j limits from L to $(i-1)$ instead of i 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" stands for the search space complexity as well as other related computational requirements like time and power consumptions. 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 n multi-grams are identified and then encoded, the required resultant number of operations in compression process is:

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

B. Performance Analysis of Decompression Process

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 in each step for unsuccessful match. It is necessary to mention that, the codes with same bit length do not essentially comprise any specific level. So, to reveal the character representation for each entry d if a switch of h levels are required, where $c_{min} \le h \le c_{max}$ 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 settings results (= overhead of searching through level + overhead for searching through offset + overhead of representation).
For detecting first character the overhead is,

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

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

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

Hence,

$$E_n = \sum_{q=p}^{h} (S'_n(l'_q)) + k'_n + B'_n \qquad (2)$$

Here, p = *m*aximum level, and S' is a function that denotes the search-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.

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 descending 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, $C'_1 = \sum_{b=d}^{g}(E_b)$ where d and g are maximum and minimum bit phases respectively and $d \geq g$.

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 denote 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:

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

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

$$T' = \sum_{y=1}^{n} \sum_{b=d}^{g} \sum_{q=p}^{h} (S'_{y,b}(l'_q)) + \sum_{y=1}^{n} \sum_{b=d}^{g} k'_{y,b} + \sum_{y=1}^{n} \sum_{b=d}^{g} B'_{y,b} \quad (3)$$

TABLE 1:

COMPLEXITIES OF COMPRESSION AND DECOMPRESSION PROCESSES

Compression Complexity of Proposed Scheme
$T = \sum_{y=1}^{n} (\sum_{j=L}^{i-1}(S_y(l_j)) + k_y + B_y)$
Decompression Complexity of Proposed Scheme
$T' = \sum_{y=1}^{n}(\sum_{b=d}^{g} \sum_{q=p}^{h} (S'_{y,b}(l'_q)) + \sum_{b=d}^{g} k'_{y,b} + \sum_{b=d}^{g} B'_{y,b})$

C. Performance Analysis with respect to Compression Ratio

Compression ratio may be defined as the ratio of total number of bits to represent the compressed information and the total number of bits in original text.

Let the training knowledgebase contains n items. The items may contain any of the total s symbols of the source language. Again, the knowledgebase items vary from one to c characters. For traditional encoding, required overhead (*i.e.* total number of required bits) to represent each character of knowledgebase entries is *log(s)* in average. Therefore, if we categorize the knowledgebase items into k_n categories, (where the categorization aspect is total number of characters in each knowledgebase entry) and there are $e_1, e_2, e_3, …, e_n$ elements in $k_1, k_2, k_3, ..., k_n$ categories respectively, then we may easily calculate the overhead for coding the total knowledgebase.

For category k_1, we need a total of $log(s) * k_1$ bits is needed to code each knowledgebase items.

As there are a total of e_1 elements, total bit requirement for coding all the elements of category k_1 is,

$log(s) * k_1 * e_1$.

Here, $log(s)$, k_1 and e_1 are non-negative integer values.

That is,

$O_{k1} = log(s) * k_1 * e_1$

$O_{k1} = k_1 \ e_1 \ log(s)$

Where O_{k1} indicates the total bit requirement for coding all the elements of category k_1.

For category k_2, we need a total of $log(s) * k_2$ bits to code each knowledgebase entries.

As there are e_2 elements, total bit requirement for coding all the elements of category k_2 is, $log(s) * k_2 * e_2$. Here, $log(s)$, k_2 and e_2 too are non-negative integer values.

That is, $O_{k2} = log(s) * k_2 * e_2$

$O_{k2} = k_2 \ e_2 \ log(s)$

Where O_{k2} indicates the total bit requirement for coding all the elements of category k_2.

Similarly, For category k_n, we can write ,

$O_{kn} = log(s) * k_n * e_n$

$O_{kn} = k_n \ e_n \ log(s)$

Where O_{kn} indicates the total bit requirement for coding all the elements of category k_2.

Symbolically, the total bit requirement for representation of the knowledgebase entries is:

$$O_t = O_{k1} + O_{k2} + O_{k3} + \dots\dots\dots + O_{kn}$$
$$= log(s) * k_1 * e_1 + log(s) * k_2 * e_2 + \dots + log(s) * k_n * e_n$$
$$= k_1 e_1 \ log(s) + k_2 e_2 \ log(s) + \dots + k_n e_n \ log(s)$$
$$= log(s) \, (k_1 e_1 + k_2 \ e_2 + \dots + k_n \ e_n)$$

That is, $$O_t = log(s) \sum_{i=1}^{n} (k_i . e_i) \qquad (4)$$

Since, $1 \le e_i \le log(n)$, and s being the total number of symbol unit in source language, in worst case,

$$O_t = log(s) \, .n \, . \, log \, n \sum_{i=1}^{n} (k_i) \qquad (5)$$

Now let us consider our proposed scheme, where components of knowledgebase entries are grouped into several levels and each entry in the level is chosen by means of effective statistical entity. Because of using hierarchical statistics, any element of certain level possess greater probability of occurrence than the element placed at any position below that. However the levels are placed in descending order to facilitate that, higher-gram texts or knowledgebase components are coded in advance to any lower-gram texts irrespective of probability distribution [11]. Here, it is noteworthy that, though the probability distribution or statistical context was not taken into consideration to organize the levels, the total structure resulted an automatic statistical distribution, because, the text-ranking scheme used to build the knowledgebase followed hierarchical steps that inherently inferred statistics from lower groups. Consequently, whenever we are formulating the statistical entries, any subgroup from its upper group will have lower values and specific coding schemes may be employed taking this criteria into consideration.

As we are using static coding in order to encode the total knowledgebase and the knowledgebase is hierarchically grouped, the resultant outcome leads our proposed scheme into a low-bit consuming one. In our proposed scheme, symbolically, any knowledgebase entry varies from 1 to r characters. That is, the levels are $r, (r - 1), (r - 2), \dots\dots\dots, 2, 1$. Formation of levels start from single characters and proceed incrementally. Any entry in the knowledgebase containing any substring from its successor level will have greater multi-gram index because of being inferred from the previously encountered entry. This aspect results in a sustainable knowledgebase architecture if we sort the knowledgebase in any order considering multi-gram index as the primary criteria. Since we are calculating the multi-gram-index or multi-gram-weight through a comprehensive text ranking scheme we may consider the underlying text elements as a single unit. Even though, we are considering the overall knowledgebase a single unit, it has been clarified earlier that, the architecture of the knowledgebase will provide us elementary grouping facilities. This coherence model provides us the opportunity to use static coding. If we have a total of m entries in the knowledgebase, coding of those values using binary stream may vary from 1 to $log(m)$ bits. Let it be y in an average, where $1 \le y \le log(m)$.

Again, as we are using multi-grams as single component, if the multi-gram consists of even k characters where $1 \le k \le r$, it will be turned into a single one. Consequently, the average requirement may be specified as

$$O_p = y_1 + y_2 + y_3 + \dots\dots\dots\dots + y_n$$

$$O_p = \sum_{i=1}^{n} y_i \qquad (6)$$

where, $1 \le y_i \le \log(m)$ for $1 \le i \le n$.

For worst case $O_p = n \log(n)$ (7)

where $y_i = \log(n)$ *for* $1 \le i \le n$.

Even if, the total number of components n is same for both the cases, since $1 \le y \le \log(n)$ (because here m=n, both indicating the total number of elements) and $1 \le e_i \le \log(n)$ multiplying any value (category index k) with e_i will definitely result much more than that of y_i. Consequently, we may deduce that,

$O_p \le O_t$

Here, n is total number of elements in the knowledgebase and y ($1 \le y \le \log(m)$) refers the total bits needed to code component i. It is inherently clear that, $O_p \le O_t$. That is, the total number of bits needed to encode the same source symbols using our proposed scheme is less than that of the traditional schemes. As, compression ratio *(R)* is the ratio of compressed bit and source bits, we get that,

$$R_t = O_t / n_t$$

and,

$$R_p = O_p / n_t$$

As, $O_p \le O_t$ we get that,

$R_p \le R_t$.

Even for the worst case, we get that,

$$O_t = \log(s) \ . n \ . \log n \sum_{i=1}^{n} (k_i)$$

$$O_t = \log(s) \ . O_p \sum_{i=1}^{n} (k_i) \qquad (8)$$

Hence we may deduce that, even for worst case,

$O_p \le O_t$

The lower the value of compression ratio, the better the compression. Consequently ,we may conclude that, the compression efficiency of proposed scheme is better than that of traditional dictionary based compression schemes.

Our proposed scheme requires less space because we have constrained the growth of the knowledgebase. The facility to use a couple of levels will ensure greater flexibility in memory management. As, the search space is minimized, the computational complexity will also be reduced. It is of no doubt that, lower computational complexity will ensure faster performance.

VI. Experimental Results And Discussions

The performance evaluation is performed on the basis of the file “book1”, “book2”, “paper4”, “paper5” and “paper6” from “Calgary Corpus”. As the prime aspect of our proposed Compression Scheme is not to compress huge amount of text rather to compress texts with limited size affordable by the mobile devices i.e. embedded systems, we took blocks of texts less than five hundred characters chosen randomly from those files ignoring binary files and other non-text files and performed the efficiency evaluation.

The most recent study involving compression of text data are:

1. “Low Complex and Power Efficient Text Compressor for Cellular and Sensor Networks” (Mode 1) by Rein *et al.* [1, 2, 3] and,
2. “A modification Of Greedy Sequential Grammar Transform based data Compression" by Islam *et al* [4]

We denote the above two methods as DCM-1 and DCM-2 respectively, where DCM stands for Data Compression Method.

The simulation was performed in a 2.0 GHz Personal Computer with 112 MB of RAM with the object oriented programming language Java. The average compression ratio for three random execution results for different size of blocks of text is as follows

TABLE 2

Compression Ratio

Corpus	Number of Characters Considered	Compressi on Ratio (%) for DCM-1	Compressi on Ratio (%) for DCM-2	Compressi on Ratio (%) for proposed scheme
paper 4	108	44.01	44.03	42.94
paper 5	061	44.98	45.51	44.02
paper 6	032	45.22	45.96	44.30
book 1	191	46.84	48.11	45.97
book 2	104	43.95	46.69	45.27

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.

The Performance of any dictionary based or knowledge-inferred compression scheme greatly varies with the architecture of dictionary construction and knowledge inference. When the test bed is considered as the same of the knowledgebase inferring one or dictionary forming one, the performance gain will be higher because of the greater match with the dictionary entries or knowledgebase components. Our proposed scheme has worse performance because, the training scheme that we have provided makes an effective use of text ranking scheme for several corpora and choice of the knowledgebase entries are unbiased towards any specific corpus. It is the novelty of our approach which ensures a greater distribution of ranking and makes the developed scheme uniformly usable for text data compression. But, other schemes which are trained with specific files of any corpora, there are greater matching with the knowledgebase and the hence performance evaluation demonstrates better performance for that corpus or that specific file than that of our proposed scheme.

The prime achievement of our proposed scheme is, it is even capable to compress source texts consisting with an average of only five characters. Though the compression ratio is deteriorated for that case, it is an evolutionary step for small text compression. This achievement may be greatly helpful for compression of sensor network concerned data and even in asynchronous data transmission management for web applications having the glimpse of real time computation.

Besides of the evaluation scheme that has been presented earlier in this section, we analyze the performance of the proposed scheme in terms of compression ratio with respect to the text presented in [3].

TITLE	*TEXT*
TEST_A	The international system of units consists of a set of the units together with a set of the prefixes. The units of the SI can be divided into two subsets. There are seven base units. Each of these base units are dimensionally independent. From the seven base units all other units are derived.
TEST_B	However, few believed that SMS would be used as the means of sending text messages from a mobile to the another. One factor in the takeup of the SMS was that operators were slow to eliminate billing fraud which was possibly by changing SMSC setting on individual handsets to the SMSC's of other operators.
TEST_C	Alice is a fictional character in the books of the Alice's adventures in the Wonderland and its sequel through the Looking-Glass, which were written by Charles Dodgson under the pen name Lewis Caroll. The character is based on Alice Liddel, a child friend of Dodgson's. The pictures, however do not depict Alice Liddel, as the illustrator never met her. She is seen as a logical girl, sometimes being pedantic, especially with Humpty Dumpty in the second book.

The performance of proposed compression scheme (in terms of compression ratio) for the above texts in comparison with [3] (indicated as DCM-1) is given below.

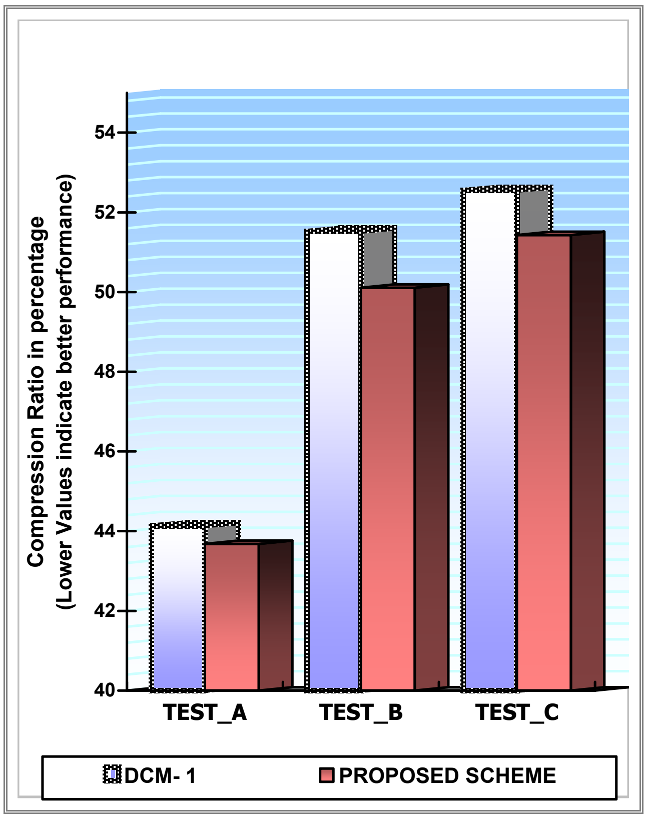

Figure 1. Performance Comparison for example text.

From the figure we find that the compression ratio is lower for all the cases for the text presented in [11].

It is necessary to mention here that In order to implement our proposed scheme no additional hardware is necessary. Rather it is possible to use even in any low-powered and low-memory devices. Basically the aspect of ensuring an affordable text compression scheme for a greater variety of smart devices we emphasis on static coding in lieu of on-the-fly coding. Besides the results presented in this section, detailed theoretical analysis on the space and time requirements, computational complexity i.e. overall computational overhead is already presented in section V.

VII. Conclusion And Recommendation

We have presented an effective and efficient approach of compressing short English text message for low-powered embedded devices. Here modified syllable based dictionary matching and static coding have been employed to obtain the compression. Moreover, a new theoretical concept of choosing the multi-grams is used, which has facilitated us to obtain mentionable compression ratio using a small number of knowledgebase entries than other methods consuming less resource. The overall strategy of computational simplicity has also ensured the reduced time complexity for the proposed compression and decompression process. The main aspect of our proposed scheme resides in the text ranking based knowledge-base construction with space integration that initiates a new arena of text compression methodology. A consistent and relevant mathematical analysis of the overall performance also establishes a strong technical basis of the proposed scheme. Moreover, the prime achievement is in the scale of starting threshold of text compression; that we have reduced to less than five characters. With limited knowledge-base size, the achieved compression is of no doubt efficient and effective. As the knowledge base is not accepted to be grown through the continuous applications, we may keep out the low-memory system from the risk of expanding its knowledgebase crossing optimal memory size and thus, the applicability of the proposed system even in any very low memory devices is ensured.

REFERENCES

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

[2] S. Rein, C. Guhmann, and F. Fitzek , “Low Complexity Compression of Short Messages,” *Proceedings of the IEEE Data Compression Conference (DCC’06)*, March 2006, pp.123–132.

[3] Stephan. Rein, F. Fitzek, M. P. G. Perucci, T. Schneider and C. Guhmann, “Low Complex and Power Efficient Text Compressor for Cellular And Sensor Networks”, In *15th IST Mobile and Wireless Communication Summit, June 2006.*

[4] J. Lansky , M. Zemlicka , “Compression Of Small Text Files Using Syllables*”. Proceedings of Data Compression Conference (DCC’06) , Los Alamitos, CA, USA, 2006.*

[5] J. Lansky and M. Zemlicka. “*Text Compression: Syllables*”. In *Annual International Workshop on DAtabases, TExts, Specifications and Objects (DATESO)*, Volume 129 of *CEUR Workshop Proceedings*, pp. 32–45. 2005.. *CEUR-WS.*

[6] Md. Rafiqul Islam, Sajib Kumar Saha, Mrinal Kanti Baowaly. “A modification of Greedy Sequential Grammar Transform based Universal Lossless data Compression”. *Proceedings of 9th International Conference on Computer and Information Technology (ICCIT 06)*, 28-30 December, 2006, Dhaka, Bangladesh.

[7] Przemysław Skibiński, “Two-Level Dictionary Based Compression”, *Proceedings of the IEEE Data Compression Conference (DCC’05),* page 481.

[8] 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.

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

[10] S. A. Ahsan Rajon, “A Study on Text Corpora for Evaluating Data Compression Schemes: Summary of Findings and Recommendations”, *Research Report, Computer Science and Engineering Discipline, Khulna University, Khulna, Bangladesh,* December, 2008.

[11] Md. Rafiqul Islam, S. A. Ahsan Rajon and Anonda Podder, “Short Text Compression for Smart Devices", *Proceedings of 11th International Conference on Computer and Information Technology (ICCIT 2008), 25-27 December, 2008, Khulna, Bangladesh,* pp. 453-558.

[12] S. Rein and C. Guhmann, “Arithmetic Coding–A Short Tutorial”, Wavelet Application Group, *Technical Report*, April 2005.

[13] David Hertz: “Secure Text Communication for the Tiger XS”. *Master of Science Thesis, Department of Electrical Engineering, Linköpings University, Linköping, Sweden.*

[14] S. A. Ahsan Rajon and Anonda Podder, “Lossless Compression of Short English Text for Low-Powered Devices”- *Undergraduate thesis, CSE Discipline, Khulna University, Khulna, Bangladesh, March, 2008.*

[15] Md. Rafiqul Islam, S. A. Ahsan Rajon, Anonda Podder, “Lossless Compression of Short English Text for Low-Powered Deices”, in the *proceedings of International Conference on Data Engineering and Management (ICDEM 2008)* Tiruchirappalli, Tamil Nadu, India. February 9, 2008.

[16] Ross Arnold, and Tim Bell, “A Corpus for the evaluation pf lossless compression algorithms”, *Data Compression Conference*, pp. 201-210, IEEE Computer Society Press, 1997.

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 got a number of papers related to these areas published in national and international journals as well as in referred conference proceedings.

He is currently working as the Head of Computer Science and Engineering Discipline, Khulna University, Bangladesh.

S. A. Ahsan Rajon received his B.Sc. Engineering degree from Computer Science and Engineering Discipline, Khulna University, Khulna in April 2008. He is now a postgraduate student of Business Administration Discipline under Management and Business Administration School of same university. Rajon is also working as an Adjunct Faculty of Computer Science and Engineering Discipline, Khulna University, Bangladesh. Rajon has made several publications in International conferences. His research interests include data engineering and management, electronic commerce and ubiquitous computing. Currently he is working on robotics.

He is a member of Institute of Engineers, Bangladesh (IEB).