International Journal of Computer and Communication Technology

Volume 5 | Issue 2 Article 3

April 2014

SMS Text Compression through IDBE (Intelligent Dictionary based Encoding) for Effective Mobile Storage Utilization

Parul Bhanarkar
Department of Information Technology, NRI Institute of Information Science &Technology, Bopal, India, parulbhanarkar@gmail.com

Nikhil Jha
Department of Information Technology, NRI Institute of Information Science &Technology, Bopal, India, Nikhil.jha.cse@gmail.com

SMS Text Compression through IDBE (Intelligent Dictionary based Encoding) for Effective Mobile Storage Utilization

Parul Bhanarkar & Nikhil Jha
Department of Information Technology, NRI Institute of Information Science &Technology, Bopal, India
Email: parulbhanarkar@gmail.com, Nikhil.jha.cse@gmail.com

Abstrac t- Effective storage utilization is the key concept for better working of any operating system. Even operating systems used for mobile phones are not an exception for this fact. This paper proposes a technique for maximizing the utilization of the storage space present in mobile phones. Thus it is important to utilize the space occupied by SMS files in phone's memory, which take maximum space. The objective involved is designing a semantic dictionary based on Intelligent Dictionary Based Encoding (IDBE) which provides a high text compression ratio to utilize the space in phone's memory. When SMS file will be received, English words present in the text will be replaced by the respective short words in the designed semantic dictionary. Thus replacing English words by the respective short forms reduces the space occupied by the SMS file. The paper describes the IDBE Compression Techniques for SMS Text Compression.

Keywords *- Space Utilization, Limpel Ziv Compression, Burrows-Wheeler-Transformation (BWT), Semantic Dictionary, IDBE (Intelligent Dictionary Based Encoding, High text Compression Ratio.*

I. INTRODUCTION

Mobile communication devices are bringing together multiple wireless networking technologies to support additional functionality and services. One of the most important developments that have taken place from communication technology is SMS. As noted, it was designed as part of Global System for Mobile communications (GSM), but is now available on a wide range of network standards. SMS has now become a popular means of communication by individual.

The most important issue of present technology is to ensure a better way of communication in a more convenient, easy and cost-effective way. The use of devices especially, mobile phones and smart phones have made short text compression crucial so as to save the memory space. The current SMS size allows not more than 160 characters. In this paper, we propose an algorithmic technique of compressing SMS text for mobile phones into short forms.

The prime objective of this proposed technique is to establish a cost effective and lossless compression scheme suitable for smart devices like cellular phones having small memory and relatively low processing speed. The IDBE (Intelligent Dictionary based Encoding) is used to compress short messages up to an optimal level, which requires optimal space, consumes less time and low overhead and reduces the communication costs. IDBE is used to design a semantic dictionary which will store compressed short forms.

Basic concept of the proposed Algorithm is detailed in Section 3 and the Algorithm is implemented in Section 4. The Running Example is discussed is Shown in Section 5. In Section 6 the Performance of the Algorithm is compared with other Compression Techniques. The Section 7 describes the dictionary management and Transfer protocol so as to share the dictionary with client and server in a secret manner without Intrusion. The paper is concluded in Section 8.

II. RELATED LITERATURE

Data compression offers an attractive approach for reducing communication costs by using available bandwidth effectively. Compression algorithms reduce the redundancy in data representation to decrease the storage required for that data. The following sections give a glimpse of the most recent research developments on text compression issues for mobile devices.

A. Data Compression Using Encrypted Text

A new algorithm has been devised for text compression that exploits the properties of the words in a dictionary to produce an encryption of given text. The basic idea of the algorithm is to define a unique encryption or signature of each word in the dictionary by replacing certain characters in the words by a special

character '*' and retain a few characters so that the word is still retrievable. This algorithm produces the best lossless compression rate reported to date in the literature: it beats the widely used UNIX "compress" method, which is based on the LZW algorithm. by about 7% and GNU-zip by about **5%.**

B. Search and Modification in Compressed Texts

This research by Stefan Böttcher, Alexander Bültmann, Rita Hartel describes about Indexed Reversible Transformation (IRT), which is a modified version of the Burrows-Wheeler-Transformation (BWT) that in combination with run length encoding (RLE) and wavelet trees (WT) allows for position based searching and updating substrings of compressed texts without prior decompression of the compressed text. As a result, IRT may be useful for a huge class of applications that due to space limitations prefer to search or to modify compressed texts instead of uncompressed texts. There is a wide variety of applications that benefit in terms of data access time or space consumption from using compressed texts, ranging from data transfer to data storage on disk to text processing in limited main memory.

C. Block Sorting Lossless Data Compression Algorithm

The algorithm designed by M. Burrows and D.Wheeler works by applying a reversible transformation to a block of input text. The transformation compresses the data, and reorders it to make it easy to compress with simple algorithms such as move-to-front coding.

The algorithm achieves speed comparable to algorithms based on the techniques of Lempel and Ziv, and obtains compression close to the best statistical modeling techniques. The size of the input block must be large (a few kilobytes) to achieve good compression.

Algorithm : Compression transformation

This algorithm takes as input a string S of N characters S[0], ……., S[N-1] selected from an ordered alphabet X of characters. An illustration of the technique is given in (1) & (2), using the string S = 'abraca', N x 6, and the alphabet

$$X = \{'a','b','c','r'\}. \qquad (1)$$

C1. [sort rotations] form a conceptual N x N matrix M whose elements are characters, and whose rows are the rotations (cyclic shifts) of S, sorted in lexicographical order. At least one of the rows of M contains the original string S. Let I be the index of the first such row, numbering from zero In our example, the index I x 1 and the matrix M is shown in (2).

Row		
0	aabrac	
1	abraca	
2	acaabr	
3	bracaa	(2)
4	caabra	
5	racaab	

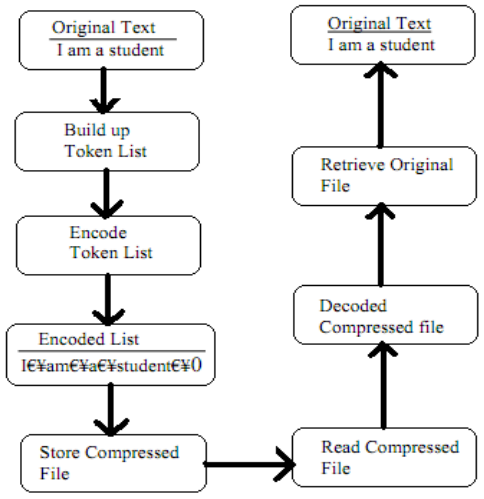
C2. [find last characters of rotations]

Let the string L be the last column of M, with characters L[0],… ,L[N-1] (equal to M[0,N- 1],…,M[N-1,N-1]). The output of the transformation is the pair (L,I). In our example, L = 'caraab' and I = 1 (from step C1).

D. Position Index Preserving Compression for Text

In this paper, a new Compression technique called Position Index preserving Compression (PIPC) is proposed which compresses the text when natural redundancy occurs in the Text. This Technique first scan the entire text line by line due to which the Time Complexity for encoding is more. It is based on the Sophisticated Compression Techniques. The comparison of this technique is shown below in Section 6. The Basic idea is given below:

Fig. 1. PIPC Compression Technique



III. USING IDBE FOR TEXT COMPRESSION

The most widely used data compression schemes /algorithms are based on the sequential data compressors of Lempel and Ziv. These Statistical modeling techniques produce superior compression, but are significantly slower. In this paper, we present a technique that achieves higher compression ratio as compared to that achieved by statistical modeling techniques, and at speeds comparable to those of algorithms based on Lempel and Ziv's.

The main issue in this paper is to implement a lossless and low complexity compression of short English text for low power consuming smart devices like cell phones with small memory. The proposed scheme is concerned with two parts. The first one consists of designing the semantic dictionary and the second provides a compression / decompression technique. The basic idea is to first convert the text in SMS to an intermediate pattern by pre-processing it, and then compress the output to generate compressed Text.

A strategy called Intelligent Dictionary Based Encoding (IDBE) is discussed below to achieve this. It has been observed that a preprocessing of the text before compression will improve the compression efficiency.

IV. IMPLEMENTATION OF IDBE ALGORITHM

A. Dictionary Creation Algorithm

START

Create Dictionary with source file as input

1. Extract words from the input files one by one and check whether it is already available in the table. If it is already available, increment the number of occurrences by one, otherwise add it and set the number of occurrence to one.
2. Sort the table by length of the words in the Ascending order (Two letter words, Three letter words and so on).
3. Again sort the table by frequency of occurrences in descending order according to the length of the word.
4. Start assigning codes with the following method:
   - Assign the first 52 (Two letter) words the ASCII characters 65 – 90 and 97 –122 as the code.
   - Now assign each of the remaining words permutation of two of the ASCII characters in the range of 65 – 90 and 97 – 122 taken in order.
   - If any words remain without assigning ASCII characters assign each of them permutation of three of the ASCII characters and finally, if required, permutation of four of the ASCII characters.
5. Repeat the above procedure for three letter words, four letter words and so on up to Twenty two letter words because the maximum length of an English word is 22[15].
6. The created file which consists of only words and their codes serves as the dictionary file.

STOP

B. Compression Algorithm

Start encode with argument input file **infile**

A. Read the dictionary and store all words and their codes in a table

B. While **infile** is not empty

1. Read the characters from **infile** and form tokens.

2. If the token is longer than 1 character, then

   1. Search for the token in the table

   2. If it is not found,

   Write the token as such in to the output file.

   Else

a. Find the length of the code for the word.

b. The actual code consists of the length concatenated with the code in the table, the length serves as a marker while decoding and is represented by the ASCII characters 251 to 254 with 251 representing a code of length 1, 252 .

c. Write the actual code into the output file.

d. Read the next character and neglect theit if it is a space. If it is any other character, make it the first character of the next token and go back to B, after inserting a marker character (ASCII 255) to indicate the absence of a space.

Else

1. Write the 1 character token

2. If the character is one of the ASCII characters 251 - 255, write the character once more so as to show that it is part of the text and not a marker End if End (While)

C. Stop.

V. EXAMPLE RUN OF THE ABOVE IMPLEMENTATION

Consider the following text shown below:

"In the beginning God created the heaven and the earth. And the earth was without form, and void; and darkness was upon the face of the deep. And the Spirit of God moved upon the face of the waters. And God said, Let here be light: and there was light. And God saw the light that it was good and God divided the light from the darkness. And God called the light Day, and the darkness he called Night. And the evening and the morning were the first day.…"

A. Running the text through the Intelligent Dictionary Based Encoder (IDBE) yields the following text

"û©û!ü%;ûNü'.û!ü".û"û!û.ÿ. û*û!û.û5ü"8ü"}ÿ, û"ü2Óÿ; û"ü%Lû5ûYû!ü" nû#û!ü&. ÿ.û*û!ü

%Ìû#ûNü&ÇûYû!ü"nû#û! ü#Éÿ.û*ûNûAÿ, ü"¿û]û.ü".ÿ: û"û]û5ü".ÿ.û*ûNü"Qû!ü".ÿ, û'û1û5û²ÿ: û"ûNü(Rû!ü".û;û!ü%Lÿ…."

It is clear from the above sample output that the encoded text provide a better compression.

VI. PERFORMANCE ANALYSIS

The performance issues such as Bits Per Character (BPC) and conversion time are compared for the three cases i.e., simple BWT, BWT with Star encoding and BWT with Intelligent Dictionary Based Encoding (IDBE).

The results are shown graphically and prove that BWT with IDBE out performs all other techniques in compression ratio, speed of compression (conversion time) and have higher level of security.

Table.1 shows the comparison between the compression ratios obtained for BWT algorithm, BWT with *encode algorithm and BWT with IDBE.

Table 1. BPC & Conversion time comparison of transform with BWT, BWT with *Encoding and BWT with IDBE for Calgary corpus files.

Calgary Corpuses Files							
		BWT		BWT with *encode		BWT with IDBE	
File Size	File size (KB)	BPC	Time (secs	BPC	Time (secs)	BPC	Time (secs)
bib	108.7	2.11	1	1.93	6	1.69	3
book1	750.8	2.85	11	2.74	18	2.36	10
book2	596.5	2.43	9	2.33	14	2.02	9
geo	100	4.84	2	4.84	6	5.18	4
news	368.3	2.83	6	2.65	10	2.37	6
paper1	51.9	2.65	1	1.59	5	2.26	2
paper2	80.3	2.61	2	2.45	5	2.14	3
paper3	45.4	2.91	2	2.6	6	2.27	2
paper4	13	3.32	2	2.79	5	2.8	2
paper5	11.7	3.41	1	3	4	2.38	2
paper6	37.2	2.73	1	2.54	5	2.44	2
progc	38.7	2.67	2	2.54	5	1.7	3
prog1	70	1.88	1	1.78	5	1.46	3
trans	91.5	1.63	2	1.53	5	1.46	3

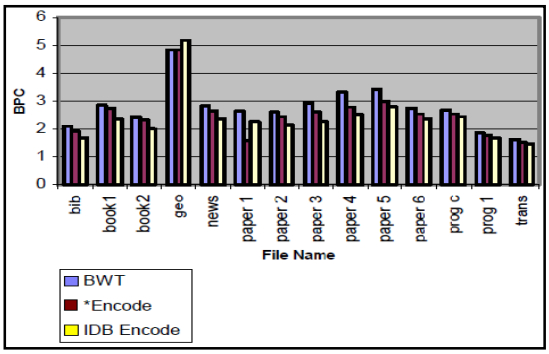


Fig. Conversion time comparison of BWT, BWT with * encode and BWT with IDBE for Calgary Corpus Files.

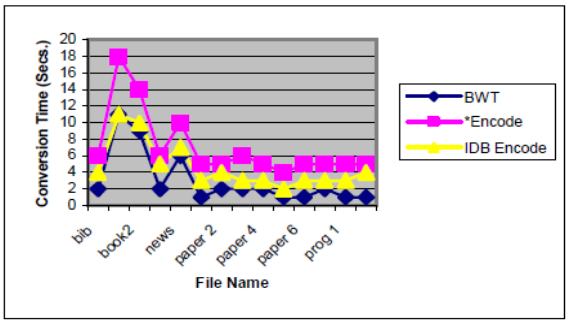


Fig. Conversion time comparison of BWT, BWT with * encode and BWT with IDBE for Calgary Corpus Files.

VII. DICTIONARY MANAGEMENT AND TRANSFER PROTOCOL

In order to make the system least vulnerable to possible attacks by hackers, a suitable dictionary management and transfer protocol can be devised.

One suggested method for dictionary transfer between server and client can be as per SSL (Secure Socket Layer) Record Protocol, which provides basic security services to various higher-level protocols such as HyperText Transport Protocol (HTTP). A typical strategy can be accepted as follows:

The first step is to fragment the dictionary in to chunks of suitable size, say 16KB. Then an optional compression can be applied. The next step is to compute a message authentication code (MAC) over the compressed data. A secret key can be used for this purpose. Cryptographic hash algorithm such as SHA-1 or MD5 can be used for the calculation.

The compressed dictionary fragment and the MAC are encrypted using symmetric encryption such as IDEA, DES or Fortezza. The final process is to prepend the encrypted dictionary fragment with the header and send it to the required destination.

VIII. CONCLUSION

The proposed technique provides an excellent method of data compression to increases the message transfer rate during data communication. In this technique we can reduce the size of data using the IDBE data compression technique. The Present network scenario demands exchange of information with reduction in both space requirement for data storage and time for data transmission along with security. Our proposed technique fulfils all such requirements as this technique use the concept of data compression and space utilization.

REFERENCES

[1]. [BuWh94] M. Burrows and D. J. Wheeler. .A Block -sorting Lossless Data Compression Algorithm., SRC Research Report 124, Digital Systems Research Center.

[2]. [ZiLe77] J. Ziv and A. Lempel. .A Universal Algorithm for Sequential Data Compression., IEEE Trans. Information Theory, IT-23, pp.237-243.

[3]. 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.

[4]. A. Ahsan Rajon and Anonda Podder, “Lossles Compression of Short English Text for Low-Powered Devices”- A non-published undergraduate thesis, CSE Discipline, Khulna University, Khulna, Bangladesh.

[5]. M. Burrows, and D.J. Wheeler, “A Block-sorting Lossless Data Compression Algorithm,” Digital Systems Research Center Research Report 124, 1994

[6]. F. Awan, N. Zhang, N. Motgi, R. Iqbal, and A. Mukherjee, “LIPT: A Reversible Lossless Text Transform to Improve Compression Performance,” Proceedings IEEE Data Compression Conference, pp. 481-210, 2001.

[7]. B. S. Shajeemohan, Dr. V. K. Govindan, “Compression Scheme for Faster and Secure Data Transmission Over Networks”, International Conference on Mobile Business (ICMB), 2005.

[8]. Jan L´ansk´y, Michal ˇZemliˇcka, “Compression Of Small Text Files using Syllables”, Data Compression Conférence (DCC), 2009.

◈◈◈