Performance Analysis of Mobile Memory Optimization Techniques

Deepali D. Kayande
Asst. Professor
Computer Science & Engg. Department
Shri Ramdeobaba College of Engineering & Management, Nagpur, India

Jignyasa Sanghvi
Asst. Professor
Computer Science & Engg. Department
Shri Ramdeobaba College of Engineering & Management, Nagpur, India

ABSTRACT

SMSLingo, a hybrid compression technique is a conjunction of RLE with static dictionary encoders based on lossless reversible transformation. This will be used for text file compression to offer better compression ratio and allow better utilization of internal memory on Android platform. The idea behind the technique is of converting the normal English text into short-form words using static dictionary and then applying Run Length Encoding (RLE) technique on this converted short-form text. Experimental results are evaluated and comparison is made between 3 text compression techniques, Huffman Coding, RLE and SMS Lingo. Promising rise is seen in the achieved results using the SMS Lingo compression algorithm in comparison with other available methods. Also this application is reasonably smaller in size as compared to the size of default SMS application present on Android platforms and with the other SMS compressor applications present for the platform. The RAM and cache memory consumption is around 3 MB for this application where the default Messaging application requires around 8 MB. Thus the proposed hybrid technique is reasonable with respect to RAM as well as cache memory consumption.

General Terms

Mobile Operating System, Memory Optimization Techniques.

Keywords

SMS Compression, Internal Memory, Android Platform, Memory Optimization, Run Length Encoding, Dictionary Encoders.

1. INTRODUCTION

The increasing features and need of mobile phones in this era has made SMS a vibrant element for mobile users and an important technique for fast and easy communication. With the growing use of extended data cards, the importance of text compression is reducing these days. But even if extended data cards help in storing the data without hurdle, it is always better to compress the data in order to increase the internal storage space, which in turn can be used for some better applications. Compression techniques are widely categorized as Lossy techniques which take care of image compression and the second category is Lossless techniques which take care of text compression. Lossless techniques are broadly classified as Dictionary coder viz. LZ family algorithms [1, 6], Block Sorting Coders viz. Burrow Wheeler Transformation [17], Entropy Coders viz. Huffman Coders [3, 14], Information Inheritance viz. Prediction by Partial Matching [19] and Sequence Run Coders viz. Run Length Encoding [6].

Many algorithms have been proposed for lossless text compression. For SMS, dictionary coders are used to gain maximum compression ratio [12]. The main aim is to accommodate maximum information in the 160 character space available per short message supported by Global System for Mobile Communication (GSM). Thus if short forms are used in SMS writing, one text message can pass information of two messages. This will reduce the cost and bandwidth of per SMS transfer over the telecommunication network. SMSLingo, a hybrid compression technique for text files has been developed which offers better compression ratio and allows better utilization of internal memory on Android platform. The idea behind the technique is of converting the normal English text into short-form words using static dictionary and then applying Run Length Encoding (RLE) technique on this converted short-form text to achieve better compression ratio. Thus, this technique is a conjunction of RLE with static dictionary encoders based on lossless reversible transformation. Dictionary coders, in this technique consist of a static database having the rational short- form words used for general SMS texting, thus this technique will be called SMSLingo in all the future references. The database is created after collecting samples for the words from people of various age groups. There are around 500 words in the dictionary for testing purpose which are divided into various sub-categories like standard abbreviations viz *Dr. for Doctor*, standard acronyms viz *PFA for Please Find Attached*, standard greetings viz *GM for Good Morning* and general words viz *mrng for morning* [20, 21, 22]. This database can be further increased to accommodate more words. SMS Lingo technique is tested on SMS files for Android 2.1 platform.

2. WOKING MODEL OF PROPOSED EXPERIMENT

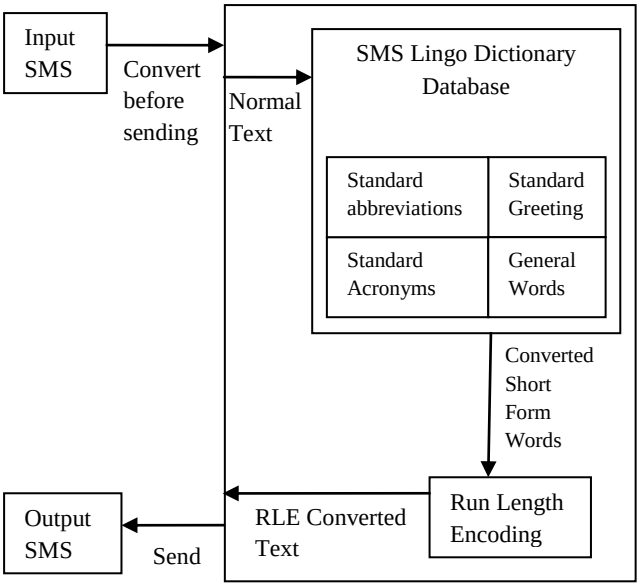

Fig1. Block Structure for Working of SMSLingo Application

Figure 1 shown above describes the block structure for working of SMSLingo.

At the sender's side input SMS is written by the user in English language/ SMS lingo, which consists of the short form words, broadly used by the texters these days. Before sending the texted SMS it is converted in the compressed form. During the compression process, first the text enters the SMSLingo Dictionary. This dictionary has a *key-value* set of English words where normal English word is the key and its rational short-form word is the value. The database is divided in 4 sub categories viz: standard abbreviations, standard acronyms, standard greetings and general words.

After the normal text in SMS is matched and replaced with the SMSLingo dictionary, an intermediate output of SMS comprising of the short-form words is created. Later, Run Length Encoding (RLE) is applied on this intermediate output. Thus this 2 step compression yields better results in achieving text compression ratio.

This compressed SMS is then sent to the recipient by the sender. Thus, the received SMS is in compressed form which requires less internal space for storage thus helping in memory optimization. The optimized memory in return can be used for better purpose.

Here, SMS files are used as a specimen for tesing this application.

Figure 2 below explains the flowchart for the application. Step by step flow of the working modules is shown in the flowchart.

On the recipient side, the Broadcast Receiver receives SMS from the server stores it in a buffer, processes it and applies the compression algorithm. Thus, before the SMS gets stored in the inbox, it passes through this application. If the SMS is in normal text it gets converted in the compressed form and then is stored in the inbox.

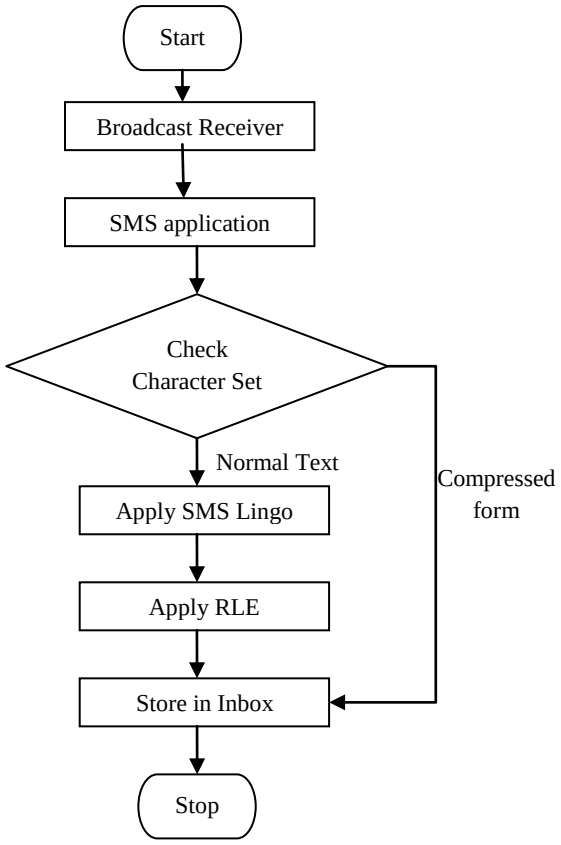

Fig 2. Flowchart for the working of SMSLingo Application

3. KEY TECHNIQUES

SMSLingo is a hybrid technique which is a conjunction of RLE with static dictionary encoders based on lossless reversible transformation.

A. Run Length Encoding (RLE)

Run Length Encoding (RLE) falls under the category of sequence run coders which are used to perform text compression. RLE is based on the count of the run for every sequence of data given in the input string [6]. Repetitive symbols of data are replaced with the respective count of that data in RLE.

The algorithm which has been proposed in this paper, rules out the dependency on entropy of the particular input string. Instead of taking into consideration the entropy of each and every string, static dictionary will be created which will store the compressed word for the usual English word used for SMS file creation.

B. Static Dictionary Encoders

In this category of encoders, a set of predefined database is maintained. This database can vary according to the application or the scenario in which this database will be used. The input words are matched with the data present in this predefined dictionary, if the match is found the respective value for the word is replaced from the database in place of the actual input word. This type of encoder is useful when the text or the set of text to be encoded is of fixed style.

Table 1: Sample SMSLingo Dictionary Entries

SMS Lingo Dictionary Entries			
As per standard dictionary abbreviations	**As per standard dictionary acronyms**	**As per standard dictionary greetings**	**Other words**
Dr for Doctor *Er* for Engineer *Ar* for Architect *HOD* for Head of Department	*PFA* for Please Find Attached *ASAP* for As Soon As Possible	*GM* for Good Morning *GN* for Goon Night *HBD* for Happy Birth Day *TY* for Thank You	*Mrng* for Morning *Fl* for Feel *Brk* for Break *Gr8* for Great

4. ALGORITHM USED

Proposed SMSLingo algorithm works as follows:

1. Start the SMS application
2. Register incoming SMS
3. SMS received by BroadcastReceiver
4. Check SMS for the character set
   if (character set == compressed format) then store in inbox
   else apply SMS Lingo Algorithm

The steps that will be performed in this algorithm are divided in four categories:

1. Creating SMS Lingo Dictionary using the standards available and then using a supervised word suggestion tool.
2. Match-and-Replace Algorithm
3. Run Length Encoding.
4. Send the SMS file.

A. Steps for creating the SMS Lingo Dictionary involve:

a. Identify the standard greetings used in English language.
   Ex: GM for Good Morning

b. Identify the standard abbreviations used in English language.
   Ex: Dr for Doctor

c. Identify the standard acronyms used in English Business Language.
   Ex: PFA for Please Find Attached.

d. For the other words, a supervised short-form word suggestion will occur.

B. Steps for creating the supervised short-form suggestion:

a. From the usual words, delete all the vowels
   Ex: Mrnng for Morning

b. Delete the repetitive alphabets, if any
   Ex: Mrng for Mrnng

c. If the above steps change the meaning of the original word then short form of the word will be created explicitly.
   Say, depending on pronunciation
   Ex: U for You

C. Match-and-Replace Algorithm:

a. Scan the input SMS file

b. Match the words from input SMS file with the words in the SMS Lingo Dictionary

c. Replace the input word with the word present in dictionary

D. Run Length Encoding

After the English words are matched and replaced with the words from SMS Lingo dictionary, Run Length Encoding will be applied on this message to achieve better compression ratio.

E. Sending the SMS

After implementation of the above 3 steps, the SMS is ready to be sent on the recipient's number.

5. R ESULTS AND DISCUSSION

Various samples of text files are tested on Huffman Encoding, Run Length Encoding and SMSLingo Algorithm to analyze the compression ratio. As compared to the already existing text compression algorithms, SMSLingo Algorithm is giving approximately 8-12% rise in the compression ratio. Table II below gives the detailed description for few sample SMS files and the complete comparative analysis of the three discussed text compression algorithms. The table entry makes it clearly visible that SMSLingo Algorithm is giving the most promising results in comparison with the already existing text compression algorithms. Following is the formula used for calculating the compression ratio of text files.

Figure 3 shows the graphical comparison for the compression ratio achieved on the text files using all three text compression. The values for the compression ratios are taken from Table II. The graphical analysis is done in order to show the improvement attained after using the SMSLingo Algorithm for text compression. The graph is plotted as per the values attained after testing the sample files.

$$\text{Compression Ratio} = \frac{A - B}{A} * 100$$

Where, A is Original size of $\left(\frac{\text{SMS}}{\phantom{\text{SMS}}}\right)$
B is Compressed size of SMS

Figure 4 below is showing a linear analysis using the Line Chart in which it is clearly visible that the results attained for the compression ratio using the SMSLingo algorithm is consistently above the results attained by other a vailable text compression algorithms. Thus using the new developed algorithm is giving better results as compared to the existing algorithms.

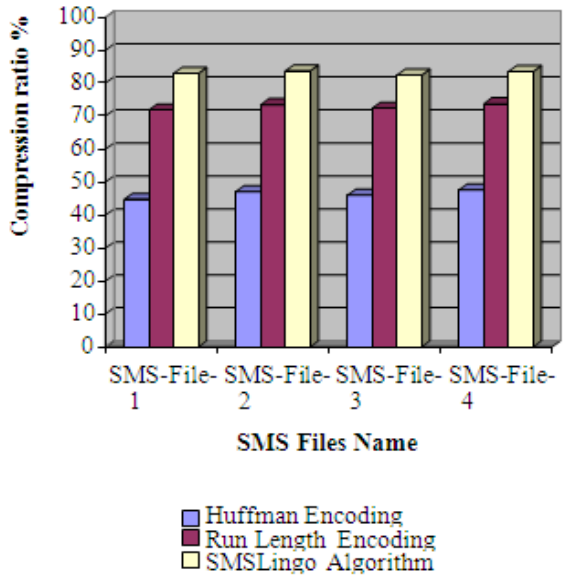

Fig. 3 Graphical Analysis of Compression Ratio Achieved using Huffman Encoding, Run Length Encoding and SMSLingo Algorithm

Table III shows the comparison of SMSLingo application with the default messaging service and other third party applications is shown in the above table. Parameters related to memory configuration are taken for comparison. Application size (.apk), data size and RAM requirement are the parameters which are tested. Among all the applications, SMSLingo has shown the most promising results. Thus taking into consideration the measly RAM behavior in Android phones, SMSLingo application can be a promising replacement for the SMS texting application. Results obtained for memory optimization are better using SMSLingo because; the SMS is in readable format even after compressing thus the need for decompressing the SMS lessens after using SMSLingo application. Thus the internal memory required for storing the SMS is optimised promisingly using SMSLingo application. And the saved internal memory can be used for other better applications in return.

6. CONCLUSION

Android device supports UTF-8 encoding for the text in SMS. This encoding consumes double number of bytes as compared to the regular character encoding which is supported by rest of the mobile operating systems. Thus the space required in storing the SMS in internal memory is doubled and thus it reduces the space which can rather be used for other applications. As per the observations in Table III, SMSLingo helps in optimizing the internal storage usage and RAM consumption on the Android device.

Entries in Table II shows that as compared to Huffman Encoding which gives 40 to 45% of compression ratio for the tested text files, SMSLingo gives 80 to 85% of compression ratio for the same files. Thus a promising rise of around 40% is achieved in the compression ratio. Also, as compared to Run Length Encoding, SMSLingo Algorithm gives promising improvement of about 10-12% in compression ratio on the tested text samples. Thus, it is observed that when semantic dictionary is combined with the sequence run encoders, better compression ratio is achieved.

Apart from the text files, this algorithm can be improved to work for large files. Compressing the SMS files and accommodating the multiple length SMS data in a single length SMS helps in economizing the application, thus making it cost effective as well. Further this algorithm can be enriched for compressing the attachments which can be shared via Multi Media Messages viz., picture files, audio files, video files, etc. Also it can be useful in SMS applications that support Rich Text Formatting. Also when larger files are sent in a compressed form, it helps in better communication.

Making the text data brief, helps in better utilization of internal storage space available in mobile phones. More information can be accommodated in the 160 character space provided per short message. This in turn will reduce the cost and bandwidth of data transfer across the telecommunication network Instead of using short-form with RLE as the compression technique, the same algorithm can be extended using other compression techniques and SMS can be sent in an encrypted fashion in order to achieve data security.

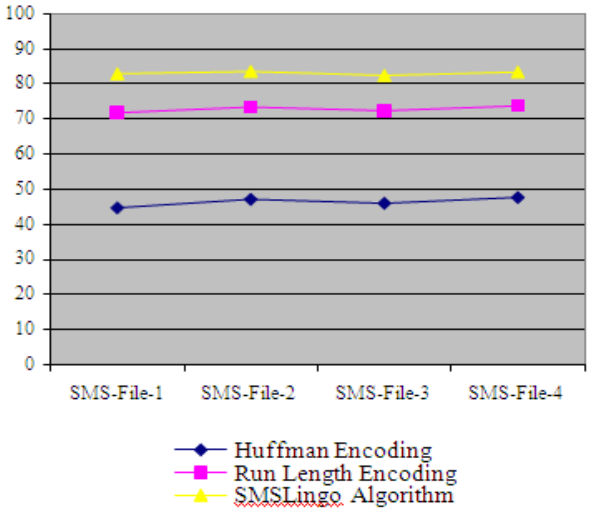

Fig.4 Line Chart showing the Linear Analysis of Compression Ratio Achieved using Huffman Encoding, Run Length Encoding and SMSLingo Algorithm

Table 2: Comparison of SMSLingo Algorithm with Run Length Encoding and Huffman Encoding

File Name	Original Size (in Bytes)	Compressed Size (in Bytes)			Compression Ratio		
		Huffman Encoding	Run Length Encoding	SMS Lingo Algorithm	Huffman Encoding	Run Length Encoding	SMS Lingo Algorithm
SMS-File-1	246	136	70	42	44.76%	71.84%	82.92%
SMS-File-2	373	197	99	61	47.14%	73.42%	83.64%
SMS-File-3	337	182	93	59	46.03%	72.40%	82.49%
SMS-File-4	339	177	89	56	47.71%	73.78%	83.48%

Table 3: Comparison of Memory Statistics of SMSLingo application with the default messaging service in an Android Device and also with few third party applications for SMS texting.

Name / Parameters	**Messaging** Service (provided by Android)	**Pack SMS** (third party SMS compressor application)	**Kysh Mysh** (third party SMS compressor application)	**SMS Lingo** (developed SMS compressor application for testing)
Total Memory	520 KB	420 KB	1.79 MB	88 KB
Application size (.apk)	512 KB	392 KB	1.78 MB	76 KB
Data size	12 KB	28 KB	16.00 KB	12 KB
RAM requirement (approx)	8.88 MB	15.5 MB	8.66 MB	3.49 MB
Test compression	No	Yes	Yes	Yes
SMS readability (without decompression)	No	No	No	Yes
Memory optimization	No	Yes	Yes	Yes

7. REFERENCES

[1] Deepali Kayande, Urmila Shrawankar, "SMS Lingo: A hybrid text compression technique for mobile storage utilization*", International Conference on Advances in Modeling, Optimization and Computing 2011*

[2] Ningde Xie, et al., "Using Lossless Data Compression in Data Storage Systems", *IEEE Transactions on Computers*, Vol. 60, No. 3, March 2011.

[3] Mo yuanbin, et al. , "A data compression algorithm based on adaptive Huffman code for Wireless Sensor Networks", *Fourth International Conference on Intelligent Computation Technology and Automation 2011.*

[4] Ahmad Affandi, et al., "The Application of Text Compression to Short Message Service Using Huffman Table", *Jurnal Generic* Indonesia 2011.

[5] Stefan Böttcher, et al., 2011 , Search and Modification in Compressed Texts, *Data Compression Conference,*

[6] S.R. Kodituwakku and U. S.Amarasinghe, 2010, Comparison of Lossless Data Compression Algorithms for Text Data, *Indian Journal of Computer Science and Engineering*, Vol 1 No 4.

[7] Rui Yang, Hong Cai, 2009, Research and Design of Short Message Service System Based on ARM and GPRS, *Second International Symposium on Computational Intelligence and Design.*

[8] Liu Pu, 2009, Performance Analysis of Short Message Service, *International Symposium on Intelligent Ubiquitous Computing and Education.*

[9] L. Robert and R. Nadarajan, 2009, Simple lossless preprocessing algorithms for text compression, *IET Softw.*, Vol. 3, Iss. 1, pp. 37–45.

[10] Xibo Wang, Yanting Yang, 2009, Method and Implementation of Sending and Receiving Mobile Phone Messages, *International Forum on Computer Science-Technology and Applications.*

[11] Mauro Teófilo, et al., 2009, Ulmo: A system to Enable Mobile Applications Personalization by Binary SMS, *Fourth International Multi- Conference on Computing in the Global Information Technology*.

[12] Stephan Rein et al., 2006, Low-Complexity Compression of Short Messages, *Data Compression Conference (DCC).*

[13] Jan L´ansk´y, Michal ˇZemliˇcka, 2009, Compression of Small Text Files using Syllables, *Data Compression Conference (DCC)*.

[14] Ming-Bo Lin, et al., 2006, A Lossless Data Compression and Decompression Algorithm and Its Hardware Architecture, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 14, No.

9, September.[15] B.S.Shajeemohan, Dr. V.K.Govindan, 2005, Compression Scheme for Faster and Secure Data Transmission over Networks, *International Conference on Mobile Business (ICMB).*

[16] Timm Euler, 2006, Tailoring Text Using Topic Words: Selection and Compression, *13th International Workshop on Database and Expert Systems Applications (DEXA).*

[17] Michelle Effros, et al., 2002, Universal Lossless Source Coding With the Burrows Wheeler Transform, *IEEE Transactions on Information Theory*, Vol. 48, No. 5.

[18] S. Kwong and Y. F. Ho, 2001, A Statistical Lempel-Ziv Compression Algorithm for Personal Digital Assistant (PDA), *IEEE Transactions on Consumer Electronics*, Vol. 47, No. 1.

[19] David Salomon, *Data Compression- The Complete Reference*, Third Edition.

[20] The Oxford English Dictionary, http://www.indiana.edu/~letrs/help-services/QuickGuides/oed-abbr.html

[21] Standard Word Abbreviations, http://www.acs.utah.edu/acs/qa_standards/psstd02a.htm

[22] Business English, http://www.learn-english-today.com/business-english/abbreviations.html