An Algorithm For Multilingual Text Compression

Md. Rezwan Salam, Md. Shahidul Islam
Department of Computer Science & Engineering
Chittagong University of Engineering & Technology.
e-mail: tanvircuet03@gmail.com, sicuet@gmail.com

Abstract: *Multilingual means use of multiple languages. A multilingual text contains text from multiple languages. Multilingual text is essential if we have to show some text to users of multiple languages. Multilingual text use Unicode base character set. Unicode provide unique bit pattern for every character of all languages. In Unicode the bit length of each character is 16 bit. So the text size of a multilingual text increases very much. As a result when a user wants to send a multilingual text through some paid channel he has to pay more than usual. So compression of multilingual text is essential. This paper proposes an efficient algorithm for multilingual text compression, so that any multilingual text can be transferred through any paid channel reducing the communication cost.*

KEY WORDS: Text compression, multilingual, unicode, paid channel.

1. INTRODUCTION

A text document may contain text from one language or more than one language. Generally to represent text in computers, communication equipment, and other devices that work with text we use 7 bit ASCII [1] character set which is based on the English alphabet. But if we want to add text from multiple languages in the same document we have to identify every character distinctly which is not possible with ASCII character set. For this purpose we have to use Unicode [2] base character set.

Unicode is an internationally standard character set. It uses 16 bit to identify each character of every language distinctly. Unicode provides us flexibility to write multilingual text, but it also increase the text size a lot which is really costly when we are transferring this text through some paid channel.

For an example in short messaging service (sms) in cellular networks we have 160 characters per sms packet when we use 7 bit ASCII character set. But if we use unicode for multilingual text we will have only 70 characters for each sms packet. For this reason compression of multilingual text is needed.

We found some works on multilingual text compression. Conley, E.S., Klein, S.T. [3] showed compression of multilingual aligned text by mapping one language into another language which will be much complex while we use more language simultaneously. We also found some text compression algorithm like Huffman coding [4], LZ77 [5] etc. They both build a runtime dictionary for compression which has to be added with the compressed text while sending data. These are not efficient while working with multilingual text and small text data. We found another approach of compression which is done by maintaining word dictionary [6]. But maintaining word dictionary for every language is not suitable and possible.

In this thesis our vision is to reduce the communication cost of multilingual text effectively by designing an efficient algorithm for multilingual text compression. This proposed algorithm is designed to handle any unicode base language easily.

2. ALGORITHM

2.1 Compression

The work flow diagram of the proposed multilingual text compression algorithm is as below:

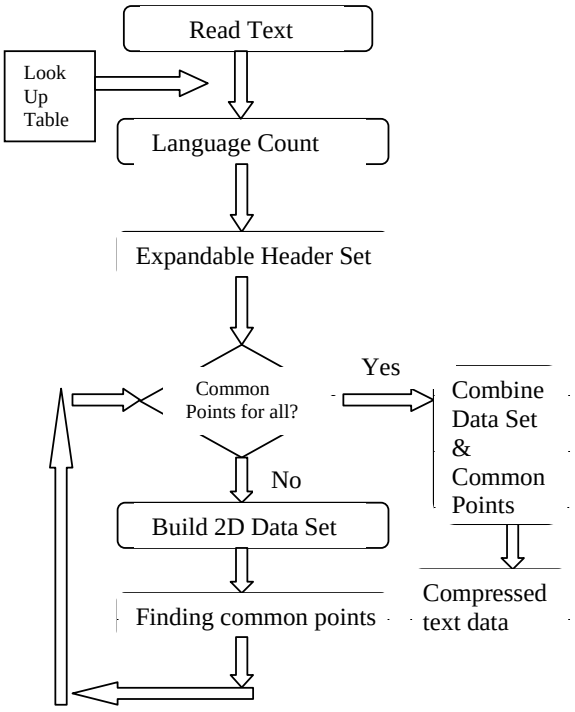

Figure 1: Flow chart of the proposed compression algorithm

ISBN 984-300-002131-3

We divide the whole compression procedure in the following parts:

2.1.1 Look up Table
2.1.2 Expandable Header Set
2.1.3 Common Points Iteration
2.1.4 Final Compressed Data

2.1.1 Look up Table: This is a table which will consist the language range. We can easily maintain this kind of table for all language easily. This look up table will contain the Unicode number of every character with their corresponding language range.

2.1.2 Expandable Header Set: An expandable header set will create with the help of look up table to distinguish the languages used in the text so that the receiver can decompress the data.

- Total Language – 4 Fixed bit
- Language IDs – 4 bit each

2.1.3 Common Points Iteration: This is a technique to reduce the text sometimes to a large extent. We will generate a 2D table/graph from the numbering of character set used. From this table we will get a common point for every adjacent character pair. This will reduce the text depending on the character position in the look up table. Now we will have a list of common points. We can also apply the 2D table on these common points to find the common points of the common point. By this we will iterate the process until we found a grand common point for all the letters in the block.

57	58	59	60	61	62	63	64
49	50	51	52	53	54	55	56
41	42	43	44	45	46	47	48
33	34	35	36	37	38	39	40
25	26	27	28	29	30	31	32
17	18	19	20	21	22	23	24
9	10	11	12	13	14	15	16
1	2	3	4	5	6	7	8
1	2	3	4	5	6	7	8

Table 1: 2D table for 8 characters

This common point is surely a very big number but this number will be represented by fewer bits then the actual numbers. Sometimes this decrement will be very much.

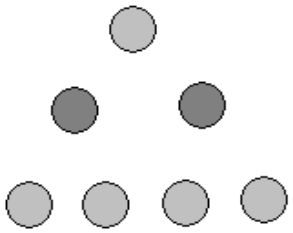

Figure 2: Building Common Points

The efficiency of the common point technique depends on the design of the look up table.

2.1.4 Final Compressed Data: We will now merge the data in following order to achieve the final compressed text:

- Header Set
- Common point
  - Block Length
  - Depth of Iteration
  - Data Block

2.2 Decompression

The receiver of the compressed data has a decompression algorithm to display the text to the receiver. The work flow diagram of the proposed multilingual text decompression algorithm is as below:

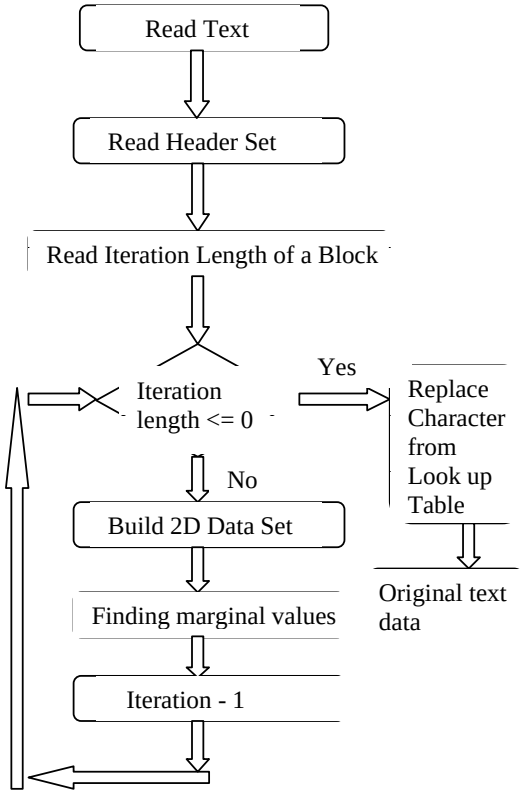

Figure 3: Flow chart of the proposed compression algorithm

We divide the whole compression procedure in the following parts:

2.2.1 Look up Table
2.2.2 Header Set Read
2.2.3 Block recovery

2.2.1 Look up Table: It is similar to the compression technique. The sender and the receiver must have the same look up table.

2.2.2 Header Set Read: Here we will collect the header set to know the languages used in the text and to make the 2D table/graph.

2.2.2 Block Recovery: The block recovery stage will use the backward iteration technique to find the marginal values from the common points until the depth of iteration of the block is reached.

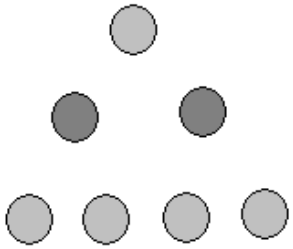

Figure 4: Recovering Marginal Values

We will perform this block recovery state to all blocks. Then we will simply replace the numbers with the characters from the look up table to have the original text.

3. RESULTS

We use java language to simulate this algorithm and got following results. We use two language Bangla & English for the simulation.

Multilingual Text	Comparative size
Multilingual text is a text contains more than one language in the same content.	Original Size: 2048 bit 128charcter(16bit/charac)
	Compress Size: 498 bit
	Compression Ratio: 76 %
	Original Size: : 2048 bit 128charcter(16bit/charac)
	Compress Size: 454 bit
	Compression Ratio: 78%

Table 2: Simulation Result

From the result set we get from the simulation software we found that in average case the compression is more than 70%.
From some more results we found the compression ratio like this:

	Compression Ratio
Best Case	Above 80%
Average Case	Above 70%
Worst Case	Above 20%

Table 3: Comparison table

Here we consider the worst case as if we are using all the characters of all the languages supported by the Unicode in the same text document.

4. CONCLUSIONS

The algorithm for multilingual text compression proposed here compresses the text significantly and make it suitable for sending the text data through any paid channel. As this algorithm does not depend on any dictionary or any semantic analysis or any partial text matching technique, this algorithm can work similar to all the language independently.
We work on some small data set specially intended for reducing the size of the packet data sending through paid channel, this algorithm also handle for large text.
This algorithm can also be used for multilingual encryption-decryption technique by improving the 2D data set used in the proposed algorithm for finding the common points.

References

[1] www.en.wikipedia.org/wiki/ASCII
[2] www.unicode.org
[3] Conley E.S., Klein S.T. ”Compression of multilingual aligned texts”, Proceedings of the Data Compression Conference, 2006.
[4] D.A. Huffman, “A Method for the Construction of Minimum-Redundancy Codes”, Proceedings of the I.R.E., September 1952, pp 1098-1102

[5] Ziv, J. and Lempel, A. “A universal algorithm for sequential data compression” IEEE; Trans. On inf. Th., IT-23:337-343, 1977.
[6] Jeehong Yang, Serap A. Savari. “Dictionary-based English text compression using word endings”, Proceedings of the Data Compression Conference, 2007, Page 410.