IJMTST is UGC Approved Journal with Journal Id: 43137

International Journal for Modern Trends in Science and Technology, 8(S08):43-46, 2022

ISSN: 2455-3778 online
DOI: https://doi.org/10.46501/IJMTST08S0808
Available online at: https://www.ijmtst.com/vol8si08.html

Two Phase Text Data Compression Using New LZW Approach with Dynamic Bit Reduction

N Srinivasa Rao | M Praveen Kumar T | Mastanaih Naidu Y

Information Technology, Bapatla Engineering College, Bapatla, AP, India

To Cite this Article
N Srinivasa Rao, M Praveen Kumar T and Mastanaih Naidu Y. Two Phase Text Data Compression Using New LZW Approach with Dynamic Bit Reduction. International Journal for Modern Trends in Science and Technology 2022, 8(S08), pp. 43-46. https://doi.org/10.46501/IJMTST08S0808

Article Info
Received: 26 May 2022; Accepted: 24 June 2022; Published: 28 June 2022.

ABSTRACT

Text Data compression techniques are very useful in the present day scenario, as the world is rotating around data and information it has become a huge problem for both data storage and quick transmission of data. This could be achieved by compression techniques; lossless techniques can be used for compression and decompression without loss of any text data. Here, using the hybrid approach of combining two lossless compression techniques to compress text data and to store and transmit via networks without loss. The New approach in LZW with bit reduction can be used to achieve the goal.

KEYWORDS: ***Text Data Compression, Lzw, Bit Reduction***

1. INTRODUCTION

New approach in LZW[1] is good helpful which gives a better practical compression. This paper aims to decrease the compression enhancing existing LZW algorithm with bit reduction technique. Concentrate mainly on text compression[5] since text plays a vital role in the digital world. New approach in LZW is a dictionary based algorithm[8]. here compress and decompress the file using dictionary. In this dictionary first 256 codes are reserved for entire ASCII character set. After that LZW dictionary occupy strings and codes. New approach in LZW is appends some selective set of frequently encounter string patterns and bit reduction is reduce the bit size from 8bits to 5bits for each character, thus can reduce text file size.

2. LITERATURE SURVEY

The important criterion for compression evaluation is compression ratio which is expected to be raised. The data compression is of two types: Lossy and lossless[6]. Lossy[10] is preferable for audio, video, and images since it is bearable of having low quality. Whereas text compressions strongly recommend lossless because nobody wants to have some meaningless or even sometimes horrible messages instead of correct ones

2.1 Lossless vs Lossy compression

(1) The lossy[9] methods advantage is over lossless methods is that in some cases a lossy method can produce a much smaller compressed file than any known lossless method, while still meeting the requirements of the application.

(2) The lossless compression[7] schemes are reversible so that the original data can be reconstructed, while lossy schemes accept some loss of data in order to achieve higher compression.

2.2 New approach in LZW Data Compression

Lempel-Ziv-Welch (LZW[4]) is a universal lossless data compression algorithm created by Abraham Lempel, Jacob Ziv, and Terry Welch. Lempel- Ziv-Weltch (LZW) is one of the powerful existing compression algorithm. It finds in many important applications like win zip, 7zip and etc.

1. LZW is a fixed length coding algorithm. Uses 12bit unsigned codes. First 256 codes are the entire ASCII character set. Lateral entries in the LZW dictionary are strings and codes.

2. Every LZW code word is a reference to a string in the dictionary.

LZW compression replaces strings of characters with single codes. It does not do any analysis of the incoming text. Instead, it just adds every new string of characters it sees to a table of strings. Compression occurs when a single code is output instead of a string of characters.

Basic idea [3]:

(1) Replaces strings of characters with single integer codes.
(2) A table of string/code pairs is built as the compression algorithm reads the input file.
(3) The table is reconstructed as the decompression algorithm reads.

Approach is appending frequently encountered string patterns to the dictionary:

The words having the high probability of occurrence in the general text will be added to the dictionary selectively. For example, words like as, at, an, in, on and etc. Using this we can reduce the number of bits required.

Compression Algorithm:
Algorithm1 New approach in LZW Compression Algorithm
1: if (STR = get input character) = EOF then
2: while there are still input characters do
3: CHAR = get input character
4: if STR+CHAR is in the String table then
5: STR = STR+CHAR
6: else
7: output code for STR
8: add STR + CHAR into the String table
9: STR = CHAR
10: end if
11: end while
12: Output the code for STR
13: end if

Decompression Algorithm:
Algorithm2 New approach in LZW Decompression Algorithm
1: Read OC = OLD CODE
2: if OC is not EOF then
3: output OC
4: CHARACTER = OC
5: while there are still input characters do
6: Read NC = NEW CODE
7: if NC is in not DICTIONARY then
8: STRING = get translation of OC
9: STRING = STRING + CHARACTER
10: else
11: STRING = get translation of NC
12: end if
13: output STRING
14: CHARACTER = first character in STRING
15: add OC + CHARACTER into the DICTIONARY
16: OC = NC
17: end while
18: Output string for code
19: end if

Bit Reduction Algorithm[2]:
Data compression is always useful for encoding information using lesser number of bits than the original representation it would use. There are many applications where the size of information would be critical. In data communication, the size of data can affect the storage cost. This algorithm was originally implemented for use

in a text file like message communication application. The idea in is this program reduces the standard 8-bit encoding to some application using specific 5-bit encoding system and then pack into a byte array. This method will reduce the size of a string considerably when the string is lengthy and the compression ratio is not affected by the content of the string. The Algorithm

1. Compression:

Let's assume that we have a input string with 8 characters. If we put this on a byte array, we get a byte array with the size of 8. A single character will need 8 bits if the characters are represented with ASCII values. A set of 8 bits can represent 2^8 different characters. But if we consider the application, a simple text data might be included only around 26 different characters. Therefore it is need to have 5-bit encoding which can give up to 2^5 different characters to represent. For converting into the new 5-bit encoding, we assign new values to the alphabet characters like | p=1 | q=2 | r=3 | s=4 | t=5 | u=6 | v=7 | w=8 |. If we look more closely at the new byte array, it will look like the following (the values of characters are in binary representation). 00000001 | 00000010 | 00000011 | 00000100 | 00000 101 | 00000110 | 00000111 | 00001000| But we use 8 bytes for storing the 8 characters. In the next step, we will cut three bits from the position of 3rd bit from the left side and extract the 5 least significant bits. The result will be shows as follows:
|00001|00010|00011|00100|00101|00110|00111|01000|.
Now we have reduced 8 bytes to 5 bytes. The next step shows how these 5 bytes convert to the 8 bytes and we get the original information.

2. Decompression:

When an array of bytes is shown, each character should be represented in the binary form. Then all the 1's and 0's should be arranged as their index values and all data split to the sets of five bits. After splitting data, it will be as follows: Code |00001 000|10 00011 0|0100 0010|1 0011000|111 01000| then these sets converted to decimal values represent the characters that we have compressed. Code |00001 = 1(a) 000|10 = 2 (b) 00011 = 3(c) 0|0100 = 4 (d) 0010|1 = 5 (e) 00110 = 6 (f) 00|111 = 7 (g) 01000| = 8 (h). Then the information will be shown in original form as "abcdefgh".

3. DESIGN

In the design we can achieve better compression combining the both techniques one after one at the time compressing the text data. First apply the new approach in LZW algorithm to compress the text data, and second use the data produced by first technique encoded information as input and apply this technique.

1. Text Data Compression Algorithm

Step I: Input the text data to be compressed.
Step II: Find the number of unique words in the input text data and assign the symbols thatare not in the input.
Step III: Now find the unique characters from the step II.
Step IV: Find the number of bits required to assign bit code to the characters.
Step V: Assign the numeric code to the unique characters found in the step II according to the number of bits calculated in step IV.
Step VI: Starting from first symbol in the input find the binary code corresponding to that symbols from assigned numerical codes and concatenate them to obtain binary output.
Step VII: Add number of 0's in MSB of Binary output until it is divisible by 8.
Step VIII: Generate the ASCII code for every 8 bits for the binary output obtained in step VI and concatenate them to create input for second phase.
[Step VI is the result of dynamic bit Reduction method in ASCII format]
Step IX: Compressed data is obtained.

2. Text Data Decompression Algorithm

Step I: Input the Final output from compressed phase.
Step II: Calculate the binary code corresponding to the ASCII values of input obtained in Step I.
Step III: Remove the extra bits from the binary output added in the compression phase.
Step IV: Calculate the numeric code for every 8 bits obtained in the Step IV.
Step V: For every numeric value obtained in the step V, find the corresponding symbol to get the final decompressed data.
Step VI: Concatenate the data symbols obtained in the step VI and obtain the final output.

Step VII: Display the final result to the user

4. CONCLUSION & FUTURE WORK

New approach in LZW algorithm with bit reduction techniques are presented in this paper. It concludes that we can achieve the goal to reduce the actual file size into a better compressed file. This indicates that Combining two techniques have performed better than the existing LZW algorithm and one time compression in terms of compressed file size. Limitations of this work include dictionary overflow with large files and increased searching time.

The suggested future work is to reduce the size of the character output of the LZW and make better use of the dictionary

Conflict of interest statement

Authors declare that they do not have any conflict of interest.

REFERENCES

[1] Srinivasa Rao Namburi, Praveen Kumar Muvva, A New Approach To Increase Lzw Algorithm Compression Ratio, IJEAST, Vol. 4 Issue 10, pg. 141-144, 2020.

[2] Rajinder Kaur, Er. Monica Goyal, An Algorithm For Lossless Text Data Compression, IJERT, vol. 2 Issue 7, 2013.

[3] David Solomon. Data compression: The complete references book, Pub-SV 3rd Edition, 2004.

[4] Michael Dipperstein. Lempel-Ziv-Welch (LZW) Encoding Discussion, http://michael.dipperstein.com/lzw/.

[5] Ezhilarasu P,Karthik Kumar P, LZW Lossless Text Data Compression Algorithm – A Review International Journal of Computer Science & Engineering Technology (IJCSET, Vol. 6 No. 11 Nov 2015.

[6] Simrandeep kaur, V.Sulochana Verma, Design and Implementation of LZW Data Compression Algorithm, International Journal of Information Sciences and Techniques (IJIST) Vol.2, No.4, July 2012

[7] Sawsan A. Abu Taleb Hossam M.J. Musafa Asma'a M. Khtoom Islah K. Gharaybih, Improving LZW Image Compression, European Journal of Scientific Research, Vol.44 No.3 (2010), pp.502-509.

[8] Sawsan A. Abu Taleb , Hossam M.J. Musafa , Asma'a M. Khtoom Improving LZW Image Compression, European Journal of Scientific Research 1450-216X Vol.44 No.3 (2010), pp.502-509

[9] Evon Abu-Taieh1, Issam AlHadid, A New Lossless Compression Algorithm, Modern Applied Science, Canadian Center of Science and Education, Vol. 12, No. 11, 2018.

[10] Restu Maulunida, Achmad Solichin, Optimization of LZW Compression Algorithm With Modification of Dictionary Formation, Indonesian Journal of Computing and Cybernetics Systems) Vol.12, No.1, January 2018, pp. 73~82.

[11] J. Uthayakumar, T. Vengattaraman, P. Dhavachelva, A survey on data compression techniques: From the perspective of data quality, coding schemes, data type and applications, Journal of King Saud University - Computer and Information Sciences, 2018.