International Journal of Advance Research in Engineering, Science & Technology(IJAREST),
ISSN(O):2393-9877, ISSN(P): 2394-2444,
Volume 2 , Issue 7 , July, 2015, Impact Factor: 2.125

Survey on LZW-Dictionary based Data compression Technique

Sonal M. Belani[1], Chintan K. Bhavsar[2]

[1]*Infromation Technology, MBICT- New V.V.Nagar, belanisonal@gmail.com*

[2] *Computer Engineering , GCET- V.V.Nagar, ck.bhavsar@gmail.com*

Abstract

Data compression techniques are used to reduce size of original data. Now a day's data transmission storage and processing are the integral parts of information systems and used in day to day life. Data transmission of large size of data over a network and also storage of large amount of data is critical task. Larger memory and increased bandwidth utilization is required by the system to store and transmit large amount of data over network. Due to this reason it increases the hardware and transmission cost. For this one solution can be that to reduce size of data before storage or transmission without affecting the information content of the data. To reduce size of data without affecting original data there are different data compression encoding techniques are available. In this paper, we survey on existing work which is used dictionary based data compression techniques and also give general introduction of different dictionary based data compression techniques

Keywords: *data compression, dictionary based compression,LZ77, LZ78,LZW*

I. INTRODUCTION

As per the situation of now a days internet users are rapidly grows so internet must requires to be capable for fast delivering text, voice, video and all other kind of multimedia information. To this end, the quest for faster and faster computing facilities (including CPU, memory, switching devices, and more) alone is insufficient; as one might expect, how to utilize the existing bandwidth effectively is also a key upon which the success of the Internet heavily relies [1].

In Statistical modeling it calculate probability of character reads in and encodes a single symbol at a time. Dictionary based modeling uses a single code to replace strings of symbols. In dictionary-based modeling, the coding problem is significantly reduced, leaving the model supremely important [2, 7]. Also, Adaptive models have been proposed in which, data does not have to be scanned once before coding in order to generate statistics.

Statistical compression techniques such as Shannon-Fano Coding, Huffman coding, Adaptive Huffman coding, Run Length Encoding and Arithmetic coding. The Dictionary based compression technique Lempel-Ziv scheme is divided into two families: those derived from LZ77 (LZ77, LZSS, LZH and LZB) and those derived from LZ78 (LZ78, LZW and LZFG).
Some Dictionary based data compression techniques are:

LZ77:

This dictionary-based scheme was presented by Jacob Ziv and Abraham Lempel in 1977 for lossless data compression. By their authors name and the year of implementation this technique is remembered.
In LZ77, the encoder uses a sliding window (implicit dictionary) over the text to search for previous occurrences of substrings [5].
LZ77 exploits the fact that words and phrases within a text file are likely to be repeated. When there is repetition, they can be encoded as a pointer to an earlier occurrence, with the pointer accompanied by the number of characters to be matched. No prior knowledge of the source and seems to require no assumptions about the characteristics of the source [5].
The encoder examines the input sequence through a sliding window which consists of two parts: a search buffer that contains a portion of the recently encoded sequence and a look-ahead buffer that contains the next portion of the sequence to be encoded. The algorithm searches the sliding window for the longest match with the beginning of the look-ahead buffer and outputs a reference (a pointer) to that match. In LZ77 the reference is always output as a triple <o,l,c>, where 'o' is an offset to the match, 'l' is length of the match, and 'c' is the next symbol after the match. If there is no match, the algorithm outputs a null-pointer (both the offset and the match length equal to 0) and the first symbol in the look-ahead buffer.
There are lots of ways that LZ77 scheme can be made more efficient and many of the improvements deal with the efficient encoding with the triples. There are several variations on LZ77 scheme, the best known are LZSS, LZH and LZB. LZSS which

was published by Storer and Szymanksi removes the requirement of mandatory inclusion of the next non-matching symbol
into each codeword. Their algorithm uses fixed length codewords consisting of offset and length to denote references. They propose to include an extra bit (a bit flag) at each coding step to indicate whether the output code represents a pair (a pointer and a match length) or a single symbol.

LZ78:

In 1978 Jacob Ziv and Abraham Lempel presented their dictionary based scheme [5], which is known as LZ78. It is a dictionary based compression algorithm that maintains an explicit dictionary. This dictionary has to be built both at the encoding and decoding side and they must follow the same rules to ensure that they use an identical dictionary. The code words output by the algorithm consists of two elements <i,c> where 'i' is an index referring to the longest matching dictionary entry and the first non-matching symbol. [5] In addition to outputting the codeword for storage / transmission the algorithm also adds the index and symbol pair to the dictionary. When a symbol that is not yet found in the dictionary, the codeword has the index value 0 and it is added to the dictionary as well. The algorithm gradually builds up a dictionary with this method.

LZW:

LZW is variant of LZ78, developed by Terry Welch in 1984. It basically applies the LZSS principle of not explicitly transmitting the next non-matching symbol to LZ78 algorithm. The dictionary has to be initialized with all possible symbols from the input alphabet [5].

An LZW token consists of just a pointer to the dictionary. The LZW method starts by initializing the dictionary to all the symbols in the alphabet. In the common case of 8-bit symbols, the first 256 entries of the dictionary (entries 0 through 255) are occupied before any data is input. Because the dictionary is initialized, the next input character will always be found in the dictionary. This is why an LZW token can consist of just a pointer and does not have to contain a character code as in LZ77 and LZ78.

LZW Algorithm steps:

1. Initialize table with single character strings
2. P = first input character
3. WHILE not end of input stream
4. C = next input character
5. IF P + C is in the string table
6. P = P + C
7. ELSE
8. output the code for P
9. add P + C to the string table
10. P = C
11. END WHILE
12. output code for P

Example of LZW:
ABABBABBB

ENCODER OUTPUT		**String Table**	
output code	**representing**	**codeword**	**String**
65	**A**	**256**	**AB**
66	**B**	**257**	**BA**
256	**AB**	**258**	**ABB**
257	**BA**	**259**	**BAB**
66	**B**	**260**	**BB**
260	**BB**		

II. LITERATURE SURVEY:

A COMPARATIVE STUDY OF TEXT COMPRESSION ALGORITHMS

Senthil and Et.all [5] compared lossless compression algorithms using Statistical compression techniques and Dictionary based compression techniques were on text data and analyzed LZB outperforms LZ77,
LZSS and LZH to show a marked compression, which is 19.85% improvement over LZ77, 6.33% improvement over LZSS and 3.42% improvement over LZH, amongst the LZ77 family. LZFG shows a significant result in the average BPC compared to LZ78 and LZW. From the result it is evident that LZFG has outperformed the other two with an improvement of 32.16% over LZ78 and 41.02% over LZW.

DATA COMPRESSION ON COLUMNAR-DATABASE USING HYBRID APPROACH (HUFFMAN AND LEMPEL-ZIV WELCH (LZW) ALGORITHM)

Dalvir and Et.all [8] provide Hybrid approach on Columnar Database using Lossless Huffman Coding and Lempel-Ziv Welch Algorithm Features. They had discussed only the Huffman and LZW coding decoding on data images and show the image compression using MATLAB software. Compression is achieved by removing one or more of the three basic data redundancies:
1) Coding redundancy, which is presented when less than optimal code words are used;
2) Inter pixel redundancy, which results from correlations between the pixels of an image;
3) Psycho visual redundancy, which is due to data that are ignored by the human visual system [13]
They conclude Hybrid approach based on Huffman coding and Lempel Ziv coding is very efficient technique for compressing the data image.

OPTIMIZATION OF LZW ALGORITHM TO REDUCE TIME COMPLEXITY FOR DICTIONARY CREATION IN ENCODING AND DECODING

Nishad & R. Et.all [9] discussed a methodology to reduce time complexity by combining binary search with LZW. They proposed a new approach using the simple binary search to overcome the problems of child node insertion using LZW with the Binary Search Tree or the Self Balanced binary tree, by using binary search, the sorted table (Dictionary) is constructed. Therefore the complexity of using the Binary Search Tree (BST) and the self balanced binary search tree gradually decreases the complexity in time. The proposed method also reduces the algorithmic complexity. To reduce comparison ratio using the binary search tree the sorted table (Dictionary) is generated using binary search. As a result their proposed methodology reduces the complexity in time with Binary search tree. The experimental result shows 94.21 % improvement on Compression and 93.34% improvement on Decompression.

A HIGH-PERFORMANCE REVERSIBLE DATA-HIDING SCHEME FOR LZW CODES

Zhi-Hui and Et.all [11] proposed a high-performance, data-hiding Lempel–Ziv–Welch(HPDH-LZW) scheme, which reversibly embeds data in LZW compression codes by modifying the value of the compression codes, where the value of the LZW code either remains unchanged or is changed to the original value of the LZW code plus the LZW dictionary size according to the data to be embedded Compared to other information-hiding schemes based on LZW compression codes, their proposed scheme achieves better hiding capacity by increasing the number of symbols available to hide secrets and also achieves faster hiding and extracting speeds due to the lower computation requirements. Their results with the proposed scheme have confirmed both its high embedding capacity and its high speed when hiding and extracting data.

III. Conclusion

LZW is a universal lossless data compression algorithm, In this paper, we have surveyed existing work done on LZW. We also give general guide line about LZW. We conclude that all there are different lossless data compression algorithms used to compress the data but LZW had better compression ratio among basic data compression algorithm.

References:

[1] Khalid Sayood, "Introduction to Data Compression", 2nd EditionSan Francisco, CA, Morgan Kaufmann, 2000.

[2] Welch T.A., "A technique for high-performance data compression", IEEE Computer, 17, pp. 8–19, 1984.

[3] Ziv. J and Lempel A., "A Universal Algorithm for Sequential Data Compression", IEEE Transactions on Information Theory 23 (3), pp. 337–342, May 1977.

[4] Ziv. J and Lempel A., "Compression of Individual Sequences via Variable-Rate Coding", IEEE Transactions on Information Theory 24 (5), pp. 530–536, September 1978.

[5] S.Shanmugasundaram and R. Lourdusamy "A Comparative Study Of Text Compression Algorithms", International Journal of Wisdom Based Computing, Vol. 1 (3), December 2011, PP-68-76

[6] C. Saravanan, M. Surender "Enhancing Efficiency of Huffman Coding using Lempel Ziv Coding for Image Compression" International Journal of Soft Computing and Engineering, ISSN: 2231-2307, Volume-2, Issue-6, January 2013

[7] E.Guy Blelloch "Introduction to Data Compression "Computer Science Department Carnegie Mellon University January 31, 2013.

[8] Dalvir Kaur and Kamaljeet Kaur, "Data Compression on Columnar-Database Using Hybrid Approach (Huffman and Lempel-Ziv Welch Algorithm)", Valume-3, Issue-5, May-2013.

[9] Nishad P M and R. Manicka Chezian, "Optimization of LZW algorithm to reduce time complexity for dictionary creation in encoding and decoding" , Asian Journal of Computer Science and Information Technology, 2012.

[10] 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

[11] Zhi-Hui Wang, Hai-Rui Yang, Ting-Fang Cheng and Chin-Chen Chang, "A high-performance Reversible data-hiding scheme for LZW codes", The Journal of Systems and Software, 2013, 2771– 2778