International Journal of Advance Engineering and Research Development (IJAERD)
Volume 1,Issue 5,May 2014, e-ISSN: 2348 - 4470 , print-ISSN:2348-6406

Reducing Text Fat with Lempel-ziv-welch Algorithm

Kalpesh A. Popat[1], Hardik K. Molia[2], Dr. Priyanka Sharma[3]

[1]*Faculty of Computer Applications, Marwadi Educatoin Foundation's Group of Institute, Rajkot, kppopat@gmail.com*
[2] *Computer Engineering, Atmiya Institute of Technology& Science, hardik.molia@gmail.com*
[3]*MCA Department, I-STAR College, V. V. Nagar, pspriyanka@yahoo.com*

Abstract— But dictionary-based compression algorithms use a completely different method to compress data as compared to statistical compression methods. This family of algorithms does not encode single symbols as variable length bit strings; it encodes variable-length strings of symbols as single tokens. The tokens form an index to a phrase dictionary. If the tokens are smaller than the phrases they replace, compression occurs. This paper describes a dictionary based compression algorithm called Lempel-ziv-welch Algorithm. The algorithm is suitable for lossless text compression.

Keywords- Lempel-ziv-welch Algorithm, Data Compression, Dictionary Based Compression

I. INTRODUCTION

Lempel-ziv-welch Algorithm is suitable for lossless text compression. It is required to create a dictionary before the compressed data can be transmitted. The created dictionary is also requiring to be sent along with the compressed data so that it can be decompressed. The algorithm is explained below.

The main component at the compression side are given below.

Dictionary:-

To compress and decompress, LZW uses a dictionary made of two rows or columns. The first row defines the code. The second row defines the string corresponding to that code.

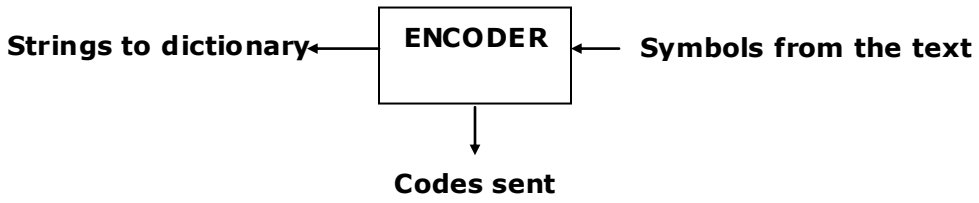
Before compression begins, the dictionary is initialized with only the set of alphabet characters in the text. If the text consists of only three symbols A,B,C, the dictionary originally has only 3 columns, but it becomes larger and larger as the text is being compressed or decompressed.

1	2	3		
A	B	C		

Original dictionary for a three-symbol text

Buffer:-

LZW uses a buffer. The symbols enter the buffer from one side one by one. The strings are transferred to the dictionary and the corresponding code is sent out according to the algorithm given below.



II. COMPRESSION ALGORITHM

Algorithm:-

1 – Initialize the dictionary with initial symbols.
2 – Read the next symbol and append it to the buffer.
3 – If there any match between buffer and dictionary then go to step 7.
4 – Send the code corresponding to the string in the buffer minus the last symbol.
5 – Add the whole buffer content to the dictionary.
6 – Purge the buffer except for the last symbol.
7 – If there is still unprocessed symbol then go to step 2.
8 – Send the code corresponding to the string in the buffer.

The algorithm tries to match the input to the longest string in the dictionary. It reads symbol one by one and stores them in the buffer. For each read it checks to see if the matching string can be found in the dictionary. If found it tries to read more symbols. If after reading one more symbols there is no match, it then adds the whole string to the dictionary. It purges the buffer except for the last symbol because the last symbol is not sent out yet. The last character remains in the buffer to become part of the next string.

Example:-

For simplicity assume that our input text contains only three letters, A,B,C. The input text, **BABACABA** is compressed to **214358** as follows,

- The dictionary is initialized to the three symbols(A,B and C). The buffer is empty.

  Content Of Buffer:- 'empty'

1	2	3
A	B	C

- The first symbol of the text (B) enters the buffer. This symbol is already in the dictionary, so the algorithm continues with the next iteration.

  Content Of Buffer:- 'B'

1	2	3
A	B	C

- The second symbol(A) enters the buffer. The string BA is not found in the dictionary, so string B is encoded as 2 and is sent. String(BA) is added to the dictionary and the buffer is purged of B(only A remains in the buffer).

  Content Of Buffer:- 'BA' Code Output:- 2

1	2	3	4
A	B	C	BA

- The third symbol(B) now enters the buffer. The string AB is not found in the dictionary, so string A is encoded as 1 and is sent. String AB is added to the dictionary and the buffer is purged of A(only B remains in the buffer).

Content Of Buffer:- 'AB' Code Output:- 21

1	2	3	4	5
A	B	C	BA	AB

- The forth symbol (A) now enters the buffer. The string BA is already in the dictionary, so the algorithm continues with the next iteration.

Content Of Buffer:- 'BA' Code Output:-21

1	2	3	4	5
A	B	C	BA	AB

- The fifth symbol(C) now enters. String BAC is not found in the dictionary, so BA is encoded as 4 and is sent. String BAC is added to the dictionary. The buffer is purged of what has been encoded and sent(BA). The only symbol left in the buffer is C.

Content Of Buffer:- 'BAC' Code Output:-214

1	2	3	4	5	6
A	B	C	BA	AB	BAC

- The sixth symbol(A) now enters. String CA is not in the dictionary, so C is encoded as 3 and is sent. String CA is added to the dictionary. The buffer is purged of what has been encoded and sent(C). the only symbol left in the buffer is A.

Content Of Buffer:- 'CA' Code Output:- 2143

1	2	3	4	5	6	7
A	B	C	BA	AB	BAC	CA

- The seventh symbol(B) now enters. String AB is in the dictionary.

Content Of Buffer:- 'AB' Code Output:- 2143

1	2	3	4	5	6	7
A	B	C	BA	AB	BAC	CA

- The eighth symbol(A) now enters. String ABA is not found in the dictionary. So AB is encoded as 5 and is sent. String ABA is added to the dictionary. The buffer is purged of what has been encoded and sent(AB). The only symbol left in the buffer is A.

Content Of Buffer:- 'ABA' Code Output:- 21435

1	2	3	4	5	6	7	8
A	B	C	BA	AB	BAC	CA	ABA

- The seventh symbol(B) now enters. String AB is in the dictionary.

  Content Of Buffer:- 'AB' Code Output:- 21435

1	2	3	4	5	6	7
A	B	C	BA	AB	BAC	CA

- The ninth symbol(B) now enters. String AB is in the dictionary.

  Content Of Buffer:- 'ABA' Code Output:- 214358

1	2	3	4	5	6	7	8
A	B	C	BA	AB	BAC	CA	ABA

- The tenth symbol(A) now enters. String ABA is in the dictionary, so the algorithm wants to start the next iteration but there are no more symbols. The algorithm terminates the loop and sends the code corresponding to the string ABA.

III. DECOMPRESSION ALGORITHM

Decompression:-

The decompression process uses the same component as the compression process. The sender does not send the dictionary created by the compression process. Instead the dictionary will be created at the receiver site and it is exact replica of the dictionary created at the sender site. the information in the dictionary is embedded in the codes transmitted.

Buffer:-

The decompression process uses two buffers. The arriving codes are decoded and the resulting symbols are kept in a temporary buffer before entering, one symbol at a time, the main buffer. Since each code may represent more than one symbol, the temporary buffer is needed to hold symbols before individual consumptions by the main buffer.

Decompression:-

1 – Initialize the dictionary with initial symbols.
2 – If there is no any unprocessed code then go to step 10.
3 – Decode the next code and send symbols to temp. buffer.
4 – If there is no any unprocessed symbol in temp buffer then go to step 2.
5 – Move the next symbol from temp buffer to the buffer.
6 – If there is any match between buffer and dictionary then go to step 4.
7 – Output the symbols in the buffer except the last one.
8 – Add the whole buffer content to the dictionary.
9 – Purge the buffer except for the last symbol and go to step 4.
10-Exit

The process of decompression is almost the same as compression. We use the same code sent by the sender in the previous example . The input code is **214358** decompressed to **BABACABA** as follows,

- The dictionary is initialized to the three symbols(A,B and C). The buffer is empty.

Content Of Buffer :- 'empty'
Content Of temp. :- 'empty'
Output Symbol :- 'empty'

1	2	3
A	B	C

- The first code(2) is decoded and the corresponding (B) enters the temporary buffer. The buffer reads the symbol. The string is in the dictionary.

Content Of Buffer :- 'B'
Content Of temp. :- 'B'
Output Symbol :- 'empty'

1	2	3
A	B	C

- The second code(1) is decoded and the corresponding symbol (A) enters the temporary buffer. The buffer reads it. Now the string BA is in the buffer. The symbol is printed. The whole string BA is added to the dictionary, the buffer is purged of B, the only symbol left is A.

Content Of Buffer :- 'BA'
Content Of temp. :- 'A'
Output Symbol :- 'B'

1	2	3	4
A	B	C	BA

- The third code(4) is decoded and the corresponding symbol (BA) enters the temporary buffer. The buffer reads it. The whole string AB is added to the dictionary.

Content Of Buffer :- 'A'
Content Of temp. :- 'BA'
Output Symbol :- 'BA'

1	2	3	4	5
A	B	C	BA	AB

- The second symbol of temp buffer A is processed. The string BA is already in the dictionary, so the algorithm continues with the next iteration.

Content Of Buffer :- 'BA'

Content Of temp. :- 'A'
Output Symbol :- 'BA'

1	2	3	4	5
A	B	C	BA	AB

- The forth symbol(3) now enters. String BAC is not found in the dictionary, so BA is taken out. String BAC is added to the dictionary.

Content Of Buffer :- 'BAC'
Content Of temp. :- 'C'
Output Symbol :- 'BABA'

1	2	3	4	5	6
A	B	C	BA	AB	BAC

- The fifth symbol(5) now enters. String CA is not in the dictionary, String CA is added to the dictionary.

Content Of Buffer :- 'CA'
Content Of temp. :- 'AB'
Output Symbol :- 'BABAC'

1	2	3	4	5	6	7
A	B	C	BA	AB	BAC	CA

- The sixth symbol(8) is not available so we use symbol 1. String AB is in the dictionary.
Content Of Buffer :- 'AB'
Content Of temp. :- 'A'
Output Symbol :- 'BABAC'

1	2	3	4	5	6	7
A	B	C	BA	AB	BAC	CA

- The string ABA is added and it is also moved to temp. buffer for later use.

Content Of Buffer :- 'ABA'
Content Of temp. :- 'ABA'
Output Symbol :- 'BABACAB'

1	2	3	4	5	6	7	8
A	B	C	BA	AB	BAC	CA	ABA

- The next symbol of temp buffer B is processed in this step.

Content Of Buffer :- 'AB'
Content Of temp. :- 'BA'
Output Symbol :- 'BABACAB'

1	2	3	4	5	6	7	8
A	B	C	BA	AB	BAC	CA	ABA

- The last symbol of temp. buffer A is processed/

Content Of Buffer :- 'ABA'
Content Of temp. :- 'A'
Output Symbol :- 'BABACABABA'

1	2	3	4	5	6	7	8
A	B	C	BA	AB	BAC	CA	ABA

- Finally whatever output contains is the decompression.

IV. CONCLUSION

Lempel-ziv-welch Algorithm is easy and simple to implement for small dictionaries. It requires less search and memory requirement too. The algorithm may get better, smaller compressed file for large directories but it becomes complex and time consuming to manage and transfer the dictionary.

References

[1] Mark Nelson, "The Data Compression Book", John Wiley & Sons; 2nd Edition edition.

[2] Adam Drozdek,Bonnie L. Drewniany,David R. Anderson, "Elements of Data Compression"

