IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 19, Issue 4, Ver. I (Jul - Aug 2017), PP 39-43
www.iosrjournals.org

Comparison Between Dynamic And Static Blocks In Sequitur Algorithm

Andysah Putera Utama Siahaan[1],Solly Aryza[2],Robbi Rahim[3],
Andre Hasudungan Lubis[4]

[1]*Faculty of Computer Science, Universitas Pembanguan Panca Budi, Medan, Indonesia*
[2]*Faculty of Engineering, Universitas Pembanguan Panca Budi, Medan, Indonesia*
[3]*Faculty of Computer Science, Akademi Perekam Medik dan Infokes Imelda, Medan, Indonesia*
[4]*Faculty of Computer Science, Universitas Medan Area, Medan, Indonesia*
[1,2,3,4]*Student of Universiti Malysia Perlis, Kangar, Malaysia*

***Abstract**: In the Sequitur algorithm, the compression process is performed based on the number of characters having similar similarities. Similarities will be detected on every syllable in the entire text. Determination of the number of characters to be compared will affect the compression level of this algorithm. Using dynamic or static blocks can optimize the number of compressed characters and of course, it affects the speed of data transmission.*

***Keywords:** Sequitur, Compression, Algorithm, Security*

I. Introduction

The speed at which data is transmitted is of the utmost importance in the world of digital networks. To achieve the maximum speed, many algorithms have been applied in sending data processes [1][2]. There is no technique to accelerate the power of hardware to improve data transmission capabilities but many techniques can be used to shorten the messages to be sent [3][5].One of the most effective algorithms to apply to plain text is Sequitur. This algorithm is excellent for compressing text data such as letters and other language messages. On message delivery, there are many repeated syllables that are repeated in a single sent message. However, the repetitive fragment has several variations in the number. There are two, three, four and even more than ten characters. In this Sequitur algorithm itself has a difference on the level of compression. The problem here is which blocks have more optimal values, dynamic or static blocks.

II. Theories

2.1Data Compression

Data compression is the process of converting large data into smaller data [6][8]. Called as compression is because the output data has a smaller capacity than the input data. The input data is a file that has certain types. The files are text, sound, video, and others. Data compression is indispensable because of two things, capacity and speed. The human tendency to collect data and the need for a fast data transfer process is greatly evolving. Therefore the methods for compressing data are growing as well.

2.3Sequitur Algorithm

The Sequitur algorithm is one of the algorithms used for data compression. This algorithm works based on the concept of context-free grammar [10]. In this algorithm, there are several symbols known as nonterminal symbols and terminals. Both types of symbols are elements of the rules used to construct a sentence. A nonterminal symbol is placed on the left and a string of terminal and nonterminal symbols on the right. The nonterminal symbol on the left becomes the name of the string to the right [7][9].

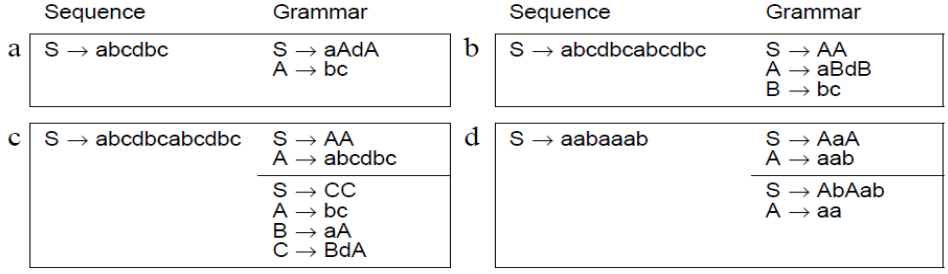

Figure 1: Sequitur reproduction

Figure 1 describes the example sequences and grammars that reproduce Sequitur result. In section (a), a sequence with one repetition; section (b) a sequence with a nested repetition; section (c) two grammars that violate the two constraints; section (d) two different grammars for the same sequence that obey the constraints [4].

III. Implementation

In this section, it will describe two techniques in the formation of the Sequitur algorithm. By experimenting with dynamic and static blocks, it will gain the better value of the compression rate. The first experiment was to use the phrase "LIKA-LIKU LAKI-LAKI TAK LAKU-LAKU"

3.1Dynamic Block

In dynamic blocks, word fragments are taken based on the resemblance of the longest character in a sentence. After that, the word fragment will be obtained from the next longest character until it goes to the similarity of the two characters.Assume the sentence is “LIKA-LIKU LAKI-LAKI TAK LAKU-LAKU”. The text above is 33 length.

L	I	K	A		L	I	K	U				
L	A	K	I	-	L	A	K	I				
T	A	K		L	A	K	U	-	L	A	K	U

From the word block above, there are two words that meet the condition. The first block is “LAKI“ and the last is “LAKU”. The word “LAKI” is repeated twice and so the word “LAKU”. The word “LAKI” is replaced by “a” and the word “LAKU” is replaced by “b”. The chosen replacing character can be derived from any ASCII code as long as the character is not listed in the previous sentence. For example, “a” and “b” are not in the word of “LIKA LIKU LAKI-LAKI TAK LAKU-LAKU”.

L	I	K	A		L	I	K	U	
a	-	a							
T	A	K		b	-	b			

The replacement words will be stored in a table.

a = LIKA
b = LAKU

The block above is the result of the first replacement. The length of the sentence is now 21 characters. There is no more four characters resemblance. The next character that has the loop is "LIK".

c	A		c	U		
a	-	a				
T	A	K		b	-	b

The block above is the result of the second replacement. The length of the sentence is now 16 characters. The replacement word will be stored in the previous table as well.

c = LIK

The final senteces is “cA cU a-a TAK b-b”. There is no more the repeated words.

CR = $\left(\frac{16}{33}\right) * 100\%$
= 0,4848 * 100%
= 48,48%

RD = $\left(\frac{33-16}{33}\right) * 100\%$
= $\left(\frac{17}{33}\right) * 100\%$
= 0,5152 * 100%
= 51,52%

The calculation above shows the compression rate and the redudancy. The algorithm made the previous sentence compressed up to twice smaller than the original text. It can be seen from the compression ratio below 50%.

3.2Static Block

In a static block, the user can freely specify the number of characters used in comparison in a sentence. The minimum number of characters allowed is two characters while there is no limit to the maximum number of characters. Each determination of the number of characters used will have different results. Below will explain the use of static blocks using the previous sentence.

Using four characters

L	I	K	A		L	I	K	U				
L	A	K	I	-	L	A	K	I				
T	A	K		L	A	K	U	-	L	A	K	U

It is similar from the previous calculation in dynamic blocks. There are two words can be compressed into a single character. The first block is "LAKI" and the last is "LAKU". The word "LAKI" is repeated twice and so the word "LAKU". The word "LAKI" is replaced by "a" and the word "LAKU" is replaced by "b".

L	I	K	A		L	I	K	U	
a	-	a							
T	A	K		b	-	b			

The replacement words will be stored in a table.

a = LIKA
b = LAKU

The block above is the result of the first replacement. The length of the sentence is now 21 characters. This is the final step since there is no other words that consists of four characters repeated more than once.

CR = $\left(\frac{21}{33}\right) * 100\%$
= 0,6363 * 100%
= 63,63%

RD = $\left(\frac{33-21}{33}\right) * 100\%$
= $\left(\frac{12}{33}\right) * 100\%$
= 0,3637 * 100%
= 36,37%

Using three characters

L	I	K	A		L	I	K	U				
L	A	K	I	-	L	A	K	I				
T	A	K		L	A	K	U	-	L	A	K	U

T here are two words can be compressed into a single character. The first block is "LIK" and the last is "LAK". The word "LIK" is repeated twice and the word "LAK" is repeated four times. The word "LIK" is replaced by "a" and the word "LAK" is replaced by "b".

a	A		a	U				
b	I	-	b	I				
T	A	K		b	U	-	b	U

The replacement words will be stored in a table.
A = LIK
b = LAK

The block above is the result of the first replacement. The length of the sentence is now 21 characters. This is the final step since there is no other words that consists of three characters repeated more than once.

CR = $\left(\frac{21}{33}\right) * 100\%$
= 0,6363 * 100%
= 63,63 %

RD = $\left(\frac{33-21}{33}\right) * 100\%$
= $\left(\frac{12}{33}\right) * 100\%$
= 0,3637 * 100%
= 36,37%

Using two characters

L	I	K	A		L	I	K	U				
L	A	K	I	-	L	A	K	I				
T	A	K		L	A	K	U	-	L	A	K	U

There are several words can be compressed into a single character. The blocksare “LI“, “KA”, “KU”, “LA”, and “KI”. The word “LI” is replaced by “a”, “KA” is replaced by “b”, “KU” is replaced by “c”, “LA” is replaced by “d” and the word “KI” is replaced by “e”.

a	b		a	c				
d	e	-	d	e				
T	A	K		d	c	-	d	c

The replacement words will be stored in a table.
a = LI
b = KA
c = KU
d = LA
e = KI

The block above is the result of the first replacement. The length of the sentence is now 21 characters. There are two words more can be compressed into a single character. The blocks are “de“ and “dc”. The word “de” is replaced by “f” and the word “dc” is replaced by “g”.

a	b		a	c		
f	-	f				
T	A	K		g	-	g

The block above is the result of the second replacement. The length of the sentence is now 17 characters. This is the final step since there is no other words that consists of two characters repeated more than once.

CR = $\left(\frac{17}{33}\right) * 100\%$
= 0,5152 * 100%
= 51,52 %

RD = $\left(\frac{33-17}{33}\right) * 100\%$
= $\left(\frac{16}{33}\right) * 100\%$
= 0,4848 * 100%
= 48,48%

IV. Conclusion

Sequitur algorithm has two techniques in doing data compression. In a static block, we can specify the number of characters to be compared in similarity to a particular sentence. The disadvantage of a static block is that the compression level can not reach the maximum point because there may be some actual blocks that can still be compressed but in the static block, it is declared complete. A dynamic block is a better technique used to achieve high compression level. This block will automatically search up to the smallest number of characters to compare in a sentence.

References

[1]. A. S. Sidhu and M. Garg, "Research Paper on Text Data Compression Algorithm using Hybrid Approach," IJCSMC, vol. 3, no. 12, pp. 1-10, 2014.
[2]. H. Al-Bahadili and S. M. Hussain, "A Bit-level Text Compression Scheme Based on the ACW Algorithm," International Journal of Automation and Computing, pp. 123-131, 2010.
[3]. M. Schindler, "Practical Huffman coding," 1998. [Online]. Available: http://www.compressconsult.com/huffman/.
[4]. C. G. Nevill-Manning and I. H. Witten, "Identifying Hierarchical Structure in Sequences: A Linear-time Algorithm," International Arab Journal of Information Technology, vol. 7, no. 1, pp. 67-82, 1997.
[5]. U. Mahajan and P. C. S. R., "Algorithms for Data Compression in Wireless Computing Systems," International Journal of Computer Science, vol. 10, no. 5, pp. 71-77, 2013.
[6]. Suherman and A. P. U. Siahaan, "Huffman Text Compression Technique," International Journal of Computer Science and Engineering, vol. 3, no. 8, pp. 103-108, 2016.
[7]. A. Dwivedi, D. B. Ojha, A. Sharma, R. Singh and A. Mishra, "Transmission of Large Information through KAP Using Non-Abelian Group," Journal of Global Research in Computer Science, vol. 2, no. 4, pp. 120-125, 2011.
[8]. R. Singh, S. Tyagi, A. Mishra, A. Tyagi and D. B. Ojha, "A Proficient mode to Transmit Secure Large Size Data with Authentication & Integrity using Double –EHDES over Teeming Channel," Journal of Global Research in Computer Science, vol. 2, no. 1, pp. 48-54, 2011.
[9]. A. Kaur and N. S. Sethi, "Approach for Lossless Text data Compression using Advanced Bit Reduction Algorithm," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 5, no. 7, pp. 1172-1176, 2015.
[10]. A. S. Sidhu and E. M. Garg, "Research Paper on Text Data Compression Algorithm using Hybrid Approach," International Journal of Computer Science and Mobile Computing, vol. 3, no. 12, pp. 1-10, 2014.