UNITECH 2017 GABROVO

INTERNATIONAL SCIENTIFIC CONFERENCE
17-18 November 2017, GABROVO

STATIC MULTI-STREAM WORD-BASED COMPRESSION ALGORITHM

Emir ÖZTÜRK
emirozturk@trakya.edu.tr
Computer Engineering Department
Trakya University – Edirne / TURKEY

Altan MESUT
altanmesut@trakya.edu.tr
Computer Engineering Department
Trakya University – Edirne / TURKEY

Abstract

In this paper, we present a static lossless word based compression algorithm (MWCA-S) for text files. MWCA-S is the static version of MWCA, which is a semi-static word based compression algorithm. MWCA-S identifies the symbols to be compressed as words. The dictionary consists of words extracted from previously acquired languages. The MWCA-S splits the original file into 4 parts according to word frequencies similar to the MWCA. The language must be given as a parameter in compression state. MWCA-S obtains 4.32 bpc at best and 5.3 bpc on average. Although the compression ratios are worse than the MWCA, compression is 3.8 times faster than the MWCA on average in terms of time.

Keywords: Text Compression, Static compression, MWCA, MWCA-S.

INTRODUCTION

As the big data getting popular, processing data and reduction of data size has become an important field. With the help of compression, data can be stored using less disk space. Text compression can be defined as representing text data in less space using redundancy within [1]. Also, higher transfer rates can also be achieved with text compression.

Three different models can be used in data compression; static, semi-static and dynamic. Semi-static algorithms perform a first pass over the text to obtain information about the content, and then perform the encoding according to this information in the second pass. Unlike semi-static algorithms, static algorithms don't need a first-pass to obtain frequency information. Compression is made using the previously created dictionaries. These methods are faster but their compression ratios are worse than semi-static ones since they use dictionaries which are not source-specific. Dynamic model uses source-specific dictionary like semi-static model and perform compression in one pass like static model. These methods update the dictionary information while compressing.

To compress text files, general compression algorithms like LZ77 [2], LZ78 [3], LZW [4], LZMA [5], Deflate [6], PPM family (PPMa, PPMb, PPMd) [7] could be used. Word based semi-static algorithms like ETDC [8], SCDC [9] and dynamic versions of them, which are DETDC [10], DSCDC, DLETDC and DLSCDC [11] could also be used.

In this study, MWCA-S (Static Multi-Stream Word-Based Compression Algorithm) which is a word based compression algorithm based on MWCA [12] is proposed. Like MWCA, MWCA-S provides multi-stream structure and compressed matching. Created streams could be either stored as seperate files or a single file. Because MWCA-S is a static compression algorithm, it compresses the text in one pass. The file is divided into 4 different streams according to the codes given. In this study, the MWCA-S algorithm was tested for eight different languages and compared with the MWCA algorithm. The two-letter code (de, en, es, fr, it, pl, nl, tr) is given as a parameter for selecting the desired language.

MWCA static algorithm is explained in the second section, and the performance results are examined in the third section.

MWCA-S

The MWCA-S compression algorithm is a static version of the MWCA algorithm. The semi-static MWCA acquires the words and their frequencies in the first-pass, sorts these

words according to these frequencies in descending order and creates the D1 and D2 dictionaries containing the first 255 and the next 65536 words. In the second pass, coding is performed using these dictionaries and two different streams are generated (S1 and S2). If the encoded word is not in these dictionaries, it will be stored in S3 stream. It also uses a bit vector (BV) to help the decoder for choosing the appropriate dictionary. As a result, the output of MWCA has six different streams, including D1 and D2.

In the developed static MWCA version, the first pass operation is not performed and static dictionaries are used. The use of static dictionaries and the absence of the first pass phase increase the compression speed, but the compression ratio is expected to decrease because of the use of fixed dictionaries.

In order to create static dictionaries, the most common words were collected from the files retrieved from the internet in eight different languages. The first 255 + 65536 of these words were used to construct the D1 and D2 dictionaries for each language.

MWCA-S uses the spaceless word model [13] which labels all non-alphanumeric characters as punctuation. A character group (word) consists of characters from the same type. The encoding algorithm generates a word until it encounters a character that is different from the characters being read. If the word is found in the dictionary, the frequency is increased. The space after an alphanumeric group is not encoded. When decompression, the algorithm looks for the word if it is "alphanumeric" or "punctuation". If two alphanumeric groups are read from the compressed streams sequentially, the decompression algorithm writes a space character between them when writing their uncompressed forms to the output file.

MWCA-S writes coded words in three different streams like MWCA. The first stream (S1) contains the one-byte code equivalents of the words in the first dictionary (D1), while the second stream (S2) contains the two-byte codes of the words in the second dictionary (D2). The third stream (S3) keeps words that are not found in the dictionaries. The encoder writes the length of the word before saving this word because there will not be a letter to use as a separator if the source file contains all the ASCII characters.

As we mentioned before, the encoding algorithm also generates a bit vector (BV) to indicate which dictionary must be used to decode the next word. In BV, "0" bit is used for words in D1, while "1" bit is used for words in D2.

When compressing files, the language of the files must be given as a parameter. MWCA-S compresses files according to these dictionaries. The pseudo-code of the MWCA-S encoding algorithm is given in Figure 1.

1. Get the first 255 words for the first dictionary
2. Get the next 65536 words in 255 words for the second dictionary
3. Read an alphanumeric or punctuation group from the file
4. Search C(W_i) code in dictionaries
5. If W_i is in one of the dictionaries
6. Write '0' or '1' to BV according to dictionary
7. Write C(W_i) to S1 or S2 according to dictionary
8. If Wi is not in one of the dictionaries
9. Write '0' bit to BV
10. Write "0" byte to S1
11. Write length(W_i) and W_i to S3
12. Repeat steps 3-11 until end of file

Fig. 1. *MWCA-S encoding algorithm*

In decompression, a bit (BV_i) is read from BV. If the BV_i is "0", one-byte code is read from S1. If $S1_i \neq$ "0", the word corresponding to the code is read from the dictionary and written to the output file. If $S1_i$ = "0" then the decompression algorithm accepts this as an escape situation. It reads a word from S3 and writes it to the output file. If BV_i is "1", a two-byte code is read from S2. Then the word corresponding to the code will be found in D2 and written to the output file. The opening phase continues until the end of BV. The pseudo-code of the MWCA-S decoding algorithm is given in Figure 2.

```
1. Define PW and CW as 'previous word' and 'next word'
2. PW = 0, CW = 0
3. Read a bit from BV
4. PW = CW
5. If BV is '0'
6.     Read a byte from S1
7.     If the byte is "0"
8.         Read a word from S3 (W)
9.     Else
10.        Find the corresponding word in D1 (W)
11. If BV is '1'
12.    Read 2 bytes from S2
13.    Find the corresponding word in D2 (W)
14. If (W) is alphanumeric, CW = 0
15. If (W) is punctuation, CW = 1
16. If CW = 0 and PW = 0 add a space character to the output file
17. Write W to output file
18. Repeat steps 3-17 until the end of BV
```

Fig. 2. *MWCA-S decoding algorithm*

COMPARISON AND PERFORMANCE RESULTS

For comparison, a corpus was prepared using Wikipedia articles collected in eight different languages. The languages and sizes of the Wikipedia articles used are given in the Table 1.

Table 1. *The files and sizes that Wikipedia articles store according to their language*

File Name	Size (Byte)
de (German)	5,030,246,082
en (English)	11,838,929,073
es (Spanish)	2,818,814,073
fr (French)	3,318,542,670
it (Italian)	2,367,208,876
nl (Dutch)	1,447,569,624
pl (Polish)	1,376,809,649
tr (Turkish)	2,130,928,293

The results for these languages are obtained in terms of compression ratios and compression/decompression speeds for both MWCA and MWCA-S. Compression ratios in bpc are given in the Table 2.

Table 2. *MWCA compression ratios (bpc)*

Files	MWCA	MWCA-S	Zlib
de	3.53	6.54	3,09
en	3.07	4.76	2,98
es	3.12	5.06	2,87
fr	3.20	5.36	2,86
it	3.12	4.32	2,98
nl	3.00	4.59	2,58
pl	3.59	5.31	3,09
tr	4.47	6.49	3,60
Average	3.39	5.30	3,01

As shown in Table 1, MWCA semi-static produces better compression ratios than the static version because it creates and uses different dictionaries for each file. The best result was obtained in “nl” with 3.00 bpc. MWCA-S gets the best result in “it” with 4.32 bpc. Zlib achieves best results in all languages. Compression and decompression speeds for MWCA and MWCA-S are given in the Table 3.

Table 3. *MWCA compression - decompression speeds (Mbps)*

Files	Semi-Static		Static		Zlib	
	Comp.	Decomp.	Comp.	Decomp.	Comp.	Decomp.
de	16.68	76.83	59.14	91.36	12,38	180,84
en	16.84	82.69	65.50	103.74	12,12	190,23
es	17.11	86.32	64.59	102.55	10,25	195,85
fr	17.64	91.47	62.94	99.87	11,19	198,52
it	16.62	88.99	65.63	107.76	12,26	193,16
nl	16.98	96.76	69.21	113.08	16,32	190,73
pl	16.89	89.35	62.11	102.22	10,78	199,26
tr	14.90	74.21	49.10	78.68	12,19	163,71
Avg.	16.71	85.83	62.28	99.91	12,19	189,04

Since the semi-static MWCA algorithm has two passes, it performs slower compression than the static MWCA method. The MWCA-S method does not include the first pass, which

contains word and frequency extraction, code assignment to words, and dynamic memory management operations used to store these words. Since the same operations are performed in the decompression algorithm, no significant difference is obtained. Both semi-static and static MWCA implementations are faster than Zlib at compression.

CONCLUSIONS

The MWCA algorithm can be used as semi-static (MWCA) and static (MWCA-S). The static algorithm provides faster compression because it performs a single pass, while the semi-static algorithm achieves better compression ratios by creating a file-specific dictionary.

In the static algorithm, dictionary selection requires a parameter to determine the language of the current file. The MWCA-S offers 46% space savings at best. This gain was 34% on average. MWCA-S can be used if speed is required and MWCA can be used to save more space. Zlib is better than MWCA at compression, but lacks of compressed matching offered by MWCA.

REFERENCES

[1] Bell, T.C., Cleary, J.G., Witten, I.H.: Text compression. Prentice-Hall, Inc. (1990)

[2] J.Ziv, A.Lempel: A Universal Algorithm for Data Compression. IEEE Trans. Inf. Theory. I, (1977)

[3] Ziv, J., Lempel, A.: Compression of Individual Sequences via Variable-Rate Coding. IEEE Trans. Inf. Theory. 24, 530–536 (1978). doi:10.1109/TIT.1978.1055934

[4] Welch, T.A.: A techinique for high-performance data compression. IEEE Comput. (1984)

[5] LZMA Specification, https://github.com/jljusten/LZMA-SDK/blob/master/DOC/lzma-specification.txt

[6] Deutsch, L.P.: DEFLATE compressed data format specification version 1.3. (1996)

[7] Cleary, J., Witten, I.: Data compression using adaptive coding and partial string matching. IEEE Trans. Commun. 32, 396–402 (1984)

[8] Navarro, G., Brisaboa, N.R.: ETDC.

[9] Brisaboa, N.R., Fariña, A., Navarro, G., Esteller, M.F.: (S,C)-Dense Coding: An Optimized Compression Code for Natural Language Text Databases. 10th Int. Symp. String Process. Inf. Retr. 122–136 (2003). doi:10.1007/978-3-540-39984-1_10

[10] Brisaboa, N.R., Farina, A., Navarro, G., Parama, J.R., Fariña, A., Navarro, G., Parama, J.R.: New adaptive compressors for natural language text. Softw. Pract. Exp. 38, 1429–1450 (2008). doi:10.1002/spe.882

[11] Brisaboa, N., Farina, A., Navarro, G., Paramá, J., Fariña, A., Navarro, G., Paramá, J.: Dynamic lightweight text compression. ACM Trans. Inf. Syst. 28, 1–32 (2010). doi:10.1145/1777432.1777433

[12] Öztürk, E., Mesut, A., Diri, B.: Multi-Stream Word-Based Compression Algorithm. In: UBMK2017 (2017)

[13] Moffat, A.: Word-based text compression. Softw. Pract. Exp. 19, 185–198 (1989)