Using The Clustering for Improve of WLZ77 Compression

Jiří Dvorský, Jan Martinovič, Jan Platoš
VŠB - Technical University of Ostrava
Department of Computer Science
17. listopadu 15, Ostrava, Czech Republic
jiri.dvorsky@vsb.cz, jan.martinovic@vsb.cz, jan.platos@vsb.cz

Abstract

The volume of textual data still increase in these days, because many types of IRS are created a more and more documents are stored in them. In IRS are usually performed 2 actions. The first action is compression of the documents. One possibility for compression of textual data is word-based compression. Several algorithms for word-based compression algorithms based on Huffman encoding, LZW or BWT algorithm was proposed. In this paper, we describe word-based compression method based on LZ77 algorithm. The second action is cluster analysis of textual database to improve quality of answers to users' queries. The information retrieved from the clustering can be very helpful in compression. Word-based compression using information about cluster hierarchy is presented in this paper. Experimental results which are provided at the end of the paper were achieved not only using well-known word-based compression algorithms WBW and WLZW but also using quite new WLZ77 algorithm.

1. Introduction

Data compression is used more and more in these days, because larger amount of data require to be transferred or backed-up and capacity of media or speed of network lines increase slowly. Some data types, which still increase, are text documents like business materials, documentation, forms, contracts, emails and many others. For compression of textual data we usually use universal compression methods based on algorithms LZ77 and LZ78. However, there are also algorithms specially developed for text like PPM or Burrows-Wheeler transformation (BWT). An interesting approach to text compression is not taking this data as sequence of characters or bytes, but as sequence of words. These words may be real words from spoken language, but also sequences of characters, which fulfill some condition, e.g. character pairs. This approach is called word-based compression.The word-based compression is not a new algorithm, but only a revised approach to the text compression. In the past, word-based compression methods based on Huffman encoding, LZW or BWT were tested. This paper describes word-based compression methods based on the LZ77 algorithm. It is focused on different variants of algorithm itself, various implementations of the sliding window algorithm and on various possibilities of output encoding. Finally, many tests were performed to compare variants of our implementation as well as other word-based or classic compression algorithms.

2. Word-based compression methods

The first widely accessible description is that of Bentley et al. [2] (see also Ryabko [20]), who proposed that a dictionary of words parsed from the text should be coupled with codewords that correspond to MTF numbers. Moffat [13] also experimented with word-based models, and showed that for a range of data files the MTF transformation was less effective than a straightforward entropy code in those experiments, arithmetic coding. A similar word-based model is available as part of the arithmetic coding implementation of Moffat et al. [14].

Huffword compression method was designed by Moffat and Zobel in 1994 [22]. HuffWord is a compression method that is specialized in texts and uses a word-based alphabet. The compression is based on the so-called Huffman canonic coding. The authors of the HuffWord claim a compression ratio of about 30%.

The beginning of the *WLZW* method dates back to 1998 when its first variant and the first results were published [5, 8]. The *WBW* method is newer and its beginning dates back to 2001 [9].

2.1. WLZ77 method

The WLZ77 method is based on LZ77 algorithm developed in 1977 by Abraham Lempel and Jacob Ziv [23] and

it's modification LZSS published by J. Storer a T. Szymanski [21] in 1982 and practically implemented by T. C. Bell [1] in 1986.

LZ77 belongs to a group of dictionary compression algorithms, more precisely to the subgroup of algorithms with a sliding window. LZ77 algorithm uses a part of the recently encoded text which is stored in sliding window as a dictionary. This window is divided into two sections - the encoded section and the plain section. Compression itself consists of searching for the longest sequence in the encoded section, which is equal to the text at begin of the plain text section.

Speed of compression using LZ77 algorithm is mainly based on the implementation of sliding window. Several types of sliding window, based on binary trees or hash table, were proposed in the recent years. But this proposal was intended to character based algorithm. Word-based algorithms have other requirements than character based. Therefore, several variant of three, base types of sliding window: binary tree, hash tables and patricia tree [15] was tested.

Word-based variant of LZ77 algorithm was goal of master thesis of Jan Platoš [17] and was published in 2007 [18]. WLZ77 algorithm used in this paper is word-based variant of LZSS which uses patricia tree based sliding window implementation with Huffman encoding as a entropy encoder for position-length pairs and characters.

Scheme of WLZW, WBW and WLZ77 compression algorithms: These compression methods can be roughly divided into two parts that are named *front end* and *back end*. These compression methods process document texts in two passes. The division of compression methods into two parts corresponds with those passes. Some parts are not active at all in the individual passes or their activity is different. Two algorithm phases can be distinguished according to the order of passing through document texts in all compression algorithms:

- *First phase* - corresponds to the *first pass* of the compression algorithm. A word-based alphabet is created in this phase. Individual tokens are extracted from documents through the process of lexical analysis, which is implemented by the front end part. This phase is shared with document indexing in the textual database.
- *Second phase* - corresponds to the *second pass* of the compression algorithm. A complete word-based alphabet is available upon the completion of the first phase and the actual document compression can begin. A lexical analysis is again performed and the token sequence that is being created is compressed by a chosen algorithm. Both phases of the compression algorithm, the front end and the back end, are already active in this phase.

The division of the compression algorithm into two relatively independent parts made it possible to separate two different compression algorithm phases, i.e. the creation of a word-based alphabet and the actual compression. Naturally, this separation has simplified the algorithm design, it has made the implementation more transparent, etc.

3. The Topic Evolution

3.1. Ultrametric

The triangular inequality holds for a metric space: $d(x,z) \leq d(x,y) + d(x,z)$ for any triplet of points x, y, z. In addition the properties of symmetry and positive definiteness are respected.

Definition 3.1 *A metric space (X,d) is called ultrametric if for all $x,y,z \in X$ we have $d(x,z) \leq \max\{d(x,y), d(y,z)\}$ [4].*

Definition of *the hierarchical agglomerative clustering* method outgoing from [11], which compute ultrametric tree from the distance matrix.

3.2. Aglomerative Clustering Method and Ultrametric

Finding of groups of objects with the same or similar features within given set of objects is the goal of cluster analysis [3]. These groups are called *clusters*. In our case objects are equal to documents that will be stored in textual base, and clusters are equal to groups of similar documents. First of all the distance of two documents and distance matrix C for each pair of documents should be defined. Our approach of cluster analysis is based on ultrametric tree [11].

As is well known, in clustering a bijection is defined between a rooted, binary, ranked, indexed tree, called a dendrogram, and a set of ultrametric distances [16, 11].

3.3. TOPIC-CA

Our research concerns with the topics undergoing an evolution. Let's assume the document from the collection of documents, that describes the same topic. It is clear, that there are some other documents in the collection that describe the same topic, but they use different words to characterize the topic. The difference can be caused by many reasons. The first document focused on the topic uses a set of words and next documents may use synonyms or for example exploration of new circumstances, facts, or political situation etc. [7, 12]. The result of searching the evolution of topic is to engaged query finding the lists of documents related by thematic with engaged query. We mean the query

Algorithm 1 TOPIC-CA

$U_{Tree} \leftarrow$ Nodes from ultrametric Tree

function TOPIC_CA($node \in U_{Tree} \cup null$)
  $L \leftarrow$ Empty list
  if $node \neq null$ **then**
    AddNodeToEnd(L, $node$)
    while $node \neq null$ **do**
      $sibling \leftarrow$ SIBLING($node$)
      $L \leftarrow$ SUB($sibling$, L)
      $node \leftarrow$ PARENT($node$)
    end while
  end if
  return L
end function

function SUB($node \in U_{Tree} \cup null$, list L)
  if $node = null$ **then**
    return L
  end if
  $sibling \leftarrow$ SIBLING($node$)
  if $node \in$ leaf nodes of U_{Tree} **then**
    AddNodeToEnd(L, $node$)
  else if $sibling \neq null$ **then**
    $siblingLeft \leftarrow$ LEFTCHILD($sibling$)
    $siblingRight \leftarrow$ RIGHTCHILD($sibling$)
    $sim_{Left} \leftarrow$ SIM($node$, $siblingLeft$)
    $sim_{Right} \leftarrow$ SIM($node$, $siblingRight$)
    if $Sim_{Right} \leq Sim_{Left}$ **then**
      $L \leftarrow$ SUB($siblingLeft$, L)
      $L \leftarrow$ SUB($siblingRight$, L)
    else
      $L \leftarrow$ SUB($siblingRight$, L)
      $L \leftarrow$ SUB($siblingLeft$, L)
    end if
  end if
  return L
end function

function SIM($n_1 \in U_{Tree} \cup null$, $n_2 \in U_{Tree} \cup null$)
  if $node_1 = null \vee node_2 = null$ **then**
    return 0
  end if
  $c_{n_1} \leftarrow$ centroid created from all leafs nodes in n_1
  $c_{n_2} \leftarrow$ centroid created from all leafs nodes in n_2
  $sim \leftarrow$ similarity between c_{n_1} and c_{n_2}
  return sim
end function

as query sets by terms or as document which is set as relevant.

An algorithm TOPIC-CA [6] which is described in the algorithm 1. This algorithm for getting the evolution of the topic from the clusters hierarchy, which uses the count of documents in evolution as the binding condition.

Definition 3.2 *TOPIC-CA is defined in the ultrametric tree X as a list $S_T = TOPIC\_CA(d_q)$. Where d_q is the node of ultrametric tree for which the topic is generated.*

4. Compression with clustering support

Ordering of input documents was not taken in consideration in general description of word-base compression methods. The compression method works correctly for any ordering of documents. Probably the simplest ordering of input documents is time ordering, i.e. the documents are compressed in the same order as they are added to textual database. Seeing that compression methods are based on searching of repeated parts of texts, it is easy to see, that this ordering is not necessary the best possible. Improvement of compression performance can be achieved by reordering of input documents. Better ordering of input documents moves similar documents to one another.

Similar documents are grouped together using cluster analysis. Of course cluster analysis is very time consuming so that it is counterproductive to perform the analysis only to enhance compression performance. But when compression method for IRS is developed, results of cluster analysis can be used in query processing [7, 12] and vice versa, cluster analysis originally devoted to query processing can be incorporated to compression.

To group similar documents together, agglomerative clustering algorithm was used. But the question how to convert hierarchical tree structure of clusters to linear list of documents still remains. The ultramnetric tree was created during clustering. We can used this fact and for list of documents L_X for compression used ultrametric ball query: $B_X(x, r)$, where r is maximal distance in ultrametric tree. L_X be sorted before compression aided distance $d(x, z)$ where $z \in L_X$.

Two strategies were used to reorder collection of documents entering the compression process:

Most Similar Left (MSL) – x in $B_X(x, r)$ is leftmost document in the ultrametric tree.

Most Similar Right (MSR) – x in $B_X(x, r)$ is rightmost document in the ultrametric tree.

5. Experimental Results

Some experiments were done to test impact clustering on word-based compression methods. Both compression methods were used in our tests. Two large text files were used for our tests: latimes.txt coming from TREC corpus [10], and enron.txt, which consists of emails from Enron email corpus[1]. In file latimes.txt individual documents are represented by each newspapers article and ordering is determined by date of publication. Each individual email represents document in file enron.txt, and ordering is defined as alphabetical ordering of users in Enron corpus. Results for this type of ordering without ordering is provided in Table 1.

The following notation will be used to describe results of experiments:

- CS is the size of compressed file
- CS_α, where $\alpha \in \{WLZW, WBW, GZIP, BZIP2\}$ is the size of compressed file without clustering, see Table 1
- $\Delta_{CS} = \frac{CS_\alpha - CS}{CS_\alpha} \times 100\%$
- $CR = \frac{CS}{S_0} \times 100\%$ is compression ratio
- $\Delta_{CR} = CR_\alpha - CR$, where $\alpha \in \{WLZW, WBW, WLZ77, GZIP, BZIP2\}$

Δ values represents difference between given value and corresponding value in compression without clustering. Positive Δ value means that given value is worse than original value, negative value means than new value is better than original one.

The first experiments was focused on comparison between three types of word-based compression methods and two commonly-used programs - GZip and BZip. Result of this experiment are depicted in Table 1. As can be seen, the best result was achieved by algorithms WBW for latimes.txt file and WLZ77 for enron.txt file. Other algorithms were much worst than WLZ77.

The second experiments are focused on compression of clustered files. Both files are relatively large. The size of documents (newspapers articles, emails) varies from houndreds of bytes to eight kilobytes. Compression with clustering and five random permutations were tested.

It is easy to see from Table 2, that clustering brings positive results in terms of compression ratio. The size of the compressed text for latimes.txt file is about 4% less than the original size in the WLZW methods, about 5% smaller than the original one in the WBW method and about 3.5% smaller than the original size in the WLZ77 method. The compression ratio improves to cca 1.2% with respect to original values in all cases. Better results were achieved for file enron.txt, see Table 2. The improvement of compression ratio is more than 2 % with respect to the original compressed size in the WLZW and WLZ77 methods, and cca 4 % in the WBW method.

Random permutations deteriorate compression in all cases (see Table 3). These negative results mean that clustering has measurable impact on compression performance, and the positive results of regarding cluster supported compression are not coincidental.

The results of standard GZip and BZip2 compression utilities provide data for comparison with our proposed word-based compression methods. As can be seen from tables, character of these results is very close to our methods; therefore clustering has serious impact on compression regardless of selected compression method.

[1] Duplicate emails were deleted before processing.

6. Conclusion and future works

Word-based variant of LZ77 compression algorithm is at least so good as other word-based compression algorithms and in many cases is much better than other. In comparison with GZIP or BZIP program is WLZ77 algorithm better of about several percent of compression ratio. Word-based compression methods combined with cluster analysis of input document have been presented in this paper. These compression methods are suitable especially for IRS. Experimental results prove that clustering has a positive impact on the compression ratio.

References

[1] T. C. Bell. Better opm/l text compression. *IEEE Trans. Commun.*, COM-34:1176–1182, 1986.

[2] J. L. Bentley, D. D. Sleator, R. E. Tarjan, and V. K. Wei. A locally adaptive data compression scheme. *Communications of the ACM*, 29(4):320–330, 1986.

[3] M. W. Berry and M. Browne. *Understanding Search Engines*. SIAM Society for Industrial and Applied Mathematics, Philadelphia, 1999.

[4] N. Brodskiy, J. Dydak, J. Higes, and A. Mitra. Dimension zero at all scales. *ArXiv Mathematics e-prints*, July 2006.

[5] J. Dvorský. Word-based compression methods for text retrieval systems. In *WDS'98*, Praha, 1998. ISBN 80-85863-29-4.

[6] J. Dvorský and J. Martinovič. Document classification based on the topic evaluation and its usage

Table 1. Compression without clustering

		latimes.txt	**enron.txt**
Original size [bytes]	S_0	498,360,166	886,993,953
WLZW method			
Compressed size [bytes]	CS_{WLZW}	158,017,940	207,908,560
Compression ratio [%]	CR_{WLZW}	31.708	23.440
WBW method			
Compressed size [bytes]	CS_{WBW}	110,246,524	167,099,129
Compression ratio [%]	CR_{WBW}	22.122	18.839
WLZ77 method			
Compressed size [bytes]	CS_{WLZ77}	113,185,477	113,394,015
Compression ratio [%]	CR_{WLZ77}	22.712	12.784
Gzip			
Compressed size [bytes]	CS_{GZIP}	175,864,812	228,953,895
Compression ratio [%]	CR_{GZIP}	35.289	25.812
BZip2			
Compressed size [bytes]	CS_{BZIP2}	131,371,338	164,720,382
Compression ratio [%]	CR_{BZIP2}	26.361	18.571

Table 2. Impact of clustering on compression

	WLZW method				
Cluster strategy on file	CS [bytes]	$CS_\alpha - CS$ [bytes]	Δ_{CS} [%]	CR [%]	Δ_{CR} [%]
MSL latimes.txt	151,869,588	-6,148,352	-3.891	30.474	-1.234
MSR latimes.txt	151,973,800	-6,044,140	-3.825	30.495	-1.213
MSL enron.txt	187,951,820	-19,956,740	-9.599	21.19	-2.25
	WBW method				
Cluster strategy on file	CS [bytes]	$CS_\alpha - CS$ [bytes]	Δ_{CS} [%]	CR [%]	Δ_{CR} [%]
MSL latimes.txt	104,701,332	-5,545,192	-5.03	21.009	-1.113
MSR latimes.txt	104,706,446	-5,540,078	-5.025	21.01	-1.112
MSL enron.txt	132,707,295	-34,391,834	-20.582	14.961	-3.877
	WLZ77 method				
Cluster strategy on file	CS [bytes]	$CS_\alpha - CS$ [bytes]	Δ_{CS} [%]	CR [%]	Δ_{CR} [%]
MSL latimes.txt	109,102,809	-4,082,668	-3.741	21.892	-0.814
MSR latimes.txt	109,127,221	-4,058,256	-3.502	21.897	-0.819
MSL enron.txt	92,094,979	-21,299,036	-18.783	10.383	-2.401
	GZip method				
Cluster strategy on file	CS [bytes]	$CS_\alpha - CS$ [bytes]	Δ_{CS} [%]	CR [%]	Δ_{CR} [%]
MSL latimes.txt	164,298,043	-11,566,769	-6.577	32.968	-2.321
MSR latimes.txt	164,322,641	-11,542,171	-6.563	32.973	-2.316
MSL enron.txt	153,765,189	-75,188,706	-32.84	17.336	-8.477
	Bzip2 method				
Cluster strategy on file	CS [bytes]	$CS_\alpha - CS$ [bytes]	Δ_{CS} [%]	CR [%]	Δ_{CR} [%]
MSL latimes.txt	120,149,683	-11,221,655	-8.542	24.109	-2.252
MSR latimes.txt	120,154,853	-11,216,485	-8.538	24.11	-2.251
MSL enron.txt	122,024,594	-42,695,788	-25.92	13.757	-4.814

Table 3. File enron.txt: random permutations, WLZW method

Permutation	CS [bytes]	$CS_\alpha - CS$ [bytes]	Δ_{CS} [%]	CR [%]	Δ_{CR} [%]
1	242,459,136	34,550,576	16.618	27.335	3.895
2	249,122,668	41,214,108	19.823	28.086	4.646
3	250,203,876	42,295,316	20.343	28.208	4.768
4	250,342,664	42,434,104	20.41	28.224	4.784
5	250,511,920	42,603,360	20.491	28.243	4.803
Average	248,528,052	40,619,492	19.537	28.019	4.579

in data compression. In *IEEE / WIC / ACM International Conference on Web Intelligence WI/IAT'07*, pages 204–207, 2007.

[7] J. Dvorský, J. Martinovič, and V. Snášel. Query expansion and evolution of topic in information retrieval systems. In V. Snásel, J. Pokorný, and K. Richta, editors, *DATESO*, volume 98 of *CEUR Workshop Proceedings*, pages 117–127. CEUR-WS.org, 2004.

[8] J. Dvorský, J. Pokorný, and V. Snášel. Word-based compression methods with empty words and non-words for text retrieval systems. In *Datasem'98*, Brno, 1998.

[9] J. Dvorský and V. Snášel. Modifications in Burrows-Wheeler compression algorithm. In *Proceedings of ISM 2001*, pages 29–35, Ostrava, 2001. ISBN 80-85988-51-8.

[10] D. Harman, editor. *The Forth REtrieval Conference (TREC-4)*. National Inst. of Standards and Technology, Gaithersburg, USA, 1997.

[11] S. Johnson. Hierarchical clustering schemes. *Psychometrika*, 32:241–254, 1967.

[12] J. Martinovic and P. Gajdos. Vector model improvement by fca and topic evolution. In Richta et al. [19], pages 46–57.

[13] A. Moffat. Word-based text compression. *Software-Practice and Experience*, 19(2):185–198, 1989.

[14] A. Moffat, R. M. Neal, and I. Witten. Arithmethic coding revisited. *ACM Transactions on Information Systems*, 16(3):256–294, 1998.

[15] D. R. Morrison. PATRICIA - Practical Algorithm To Retrieve Information Coded in Alphanumeric. *J. ACM*, 15(4):514–534, 1968.

[16] F. Murtagh. On ultrametricity, data coding, and computation. *Journal of Classification*, Volume 21, Number 2 / September:167–184, 2004.

[17] J. Platoš. Word-based text compression. Master's thesis, Department of Computer Science, Faculty of Electrical Engineering and Computer Science, VSB-Technical University Ostrava, Ostrava (Czech Republic), 2006.

[18] J. Platoš and J. Dvorský. Word-based text compression. In *Proceedings of the DCCA 2007*, 2007.

[19] K. Richta, V. Snásel, and J. Pokorný, editors. *Proceedings of the Dateso 2005 Annual International Workshop on DAtabases, TExts, Specifications and Objects, Desna, Czech Republic, April 13-15, 2005*, volume 129 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2005.

[20] B. Y. Ryabko. Technical correspondence: A locally adaptive data compression scheme. *Communications of the ACM*, 30(9):792, 1987.

[21] J. A. Storer and T. G. Szymanski. Data compression via textual substitution. *Journal of the ACM*, 26(10/82):928–951, 1982.

[22] I. Witten, A. Moffat, and T. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Van Nostrand Reinhold, 1994.

[23] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, IT-23(3):337–343, 1977.