LIPT-DERIVED TRANSFORM METHODS USED IN LOSSLESS COMPRESSION OF TEXT FILES

Radu RĂDESCU[1]

Acest articol se ocupă de avantajele obţinute în compresia fără pierderi a datelor prin utilizarea metodelor de preprocesare, cu scopul de a exploata mai eficient redundanţa fişierului sursă. Contribuţiile originale ale lucrării se concentrează pe transformata ILPT (Length-Index Preserving Transform). ILPT face parte dintre cele trei proceduri derivate din transformata cu conservarea lungimii indexului, LIPT (Index Length Preserving Transform): ILPT, NIT şi LIT. Aceste transformări sunt aplicate fişierelor text, iar algoritmii aferenţi sunt prezentaţi pe scurt în lucrare, înainte de a se evidenţia efectele lor benefice asupra unui set de fişiere de test alese din corpusurile clasice de texte. Au fost obţinute câteva rezultate experimentale, din care au fost deduse concluzii interesante.

The present paper refers to the benefits provided in the lossless compression by using preprocessing methods in order to exploit better the redundancy of the source file. The following contributions focus on the Length-Index Preserving Transform (LIPT). The procedures are derived from LIPT and are known as ILPT, NIT and LIT. These transforms are applied to text files. The algorithms are briefly presented before emphasizing their positive effects on a set of test files chosen from the classical text corpora. Experimental results were performed and some interesting conclusions were driven on their basis.

Keywords: data compression, lossless algorithms, transform methods

1. Introduction

One of the new approaches in lossless text compression is to apply a reversible lossless transformation [5], [6] to a source file before applying any other existing compression algorithm. The transformation is meant to make the file compression easier [1], [2]. The original text is offered to the transformation input and its output is the transformed text, further applied to an existing compression algorithm [7], [8]. Decompression uses the same methods in the reverse order: decompression of the transformed text first and the inverse transform after that.

Several important remarks could be made regarding this model. The transformation has to be perfectly reversible, in order to keep the lossless feature of text compression [6]. The compression and decompression algorithms remain unchanged, thus they do not exploit the transformation-related information during

[1] Assoc. Prof., Applied Electronics and Information Engineering Department, University POLITEHNICA of Bucharest, Romania, e-mail: radu.radescu@upb.ro

the compression [11], [12]. The goal is to increase the compression ratio compared to the result obtained by using the compression algorithm only [3], [4].

The algorithms use a fixed amount of data stored in a dictionary [13], [16] specific to every work domain [15]. This dictionary should be known both by the sender and the receiver of the compressed files [9]. The average size of the English language dictionary is about 0.5 MB and it can be downloaded along with application files. If the compression algorithms are to be use repeatedly (a valid assumption in the practical cases), the dictionary size is negligible. The experimental results measuring the performances of the preprocessing methods are given using the Calgary Corpus [17] and some representative Romanian text files.

2. ILPT, NIT, and LIT Transforms

The three transform methods to be described here are all derived from the Length-Index Preserving Transform (LIPT), which is briefly presented in the following. LIPT uses a transformation dictionary of the corresponding language. Between the transformation dictionary and the specified language (English, Romanian, etc.), there is a one to one correspondence. The words that are not found in the dictionary are left unchanged in the coded text.

To create the LIPT dictionary, the language dictionary needs to be sorted depending on the word length, and every block of a specific length to be sorted in descending order of the frequency of the words. The encoding and decoding processes can be put together. We assume that both the compressor and the decompressor have access to the same *D* dictionary and its D_{LIPT} correspondent.

The **encoding steps** are:

1. The words from the input text are searched in the *D* dictionary.

2. If the input text is found in the *D* dictionary, then the position and the number of its length block are marked down and the adequate transformation is searched at the same position and same length block in the D_{LIPT} dictionary. This transformation is the encoding of the input word. It this is not found in the *D* dictionary that it is transferred unchanged.

3. Once the input text is transformed according to step 1 and 2, this is sent to a compressor (e.g., Bzip2, PPM, etc.)

The **decoding steps** are:

1. The compressed received text is first decompressed using the same compressor used in the coding phase the result being the LIPT transformed text.

2. To the decompressed text, an inverse transformation is applied. The words preceded by the "*" character are the transformed ones, and those without it are unchanged, so they do not need the inverse transformation. The transformed word is then interpreted like this: the length character indicates the length block where the decoded word is found, the next three symbols indicate

the *offset* where the word is situated in that block and there can also be a capitalization mask. The word is then searched in the D dictionary according to the above parameters. The transformed words are then replaced with the corresponding ones from the D dictionary.

3. The capitalization mask is applied.

It was noticed that the majority of words in the English language have a length ranging between 1 and 10 letters. Most of them have a length ranging between 2 and 5 letters. The length and the frequency of words were a solid base on creating the LIPT transform method [12], [17]. This can be considered a first phase of a compression algorithm with many stages, like the Bzip2, that includes the RLE encoding [6], the BWT transform [1] and the MTF [1] and Huffman encoding [6], [14], and [15]. LIPT can be used as an additional component in the Bzip2 algorithm, before the RLE encoding, or it can also replace it.

ILPT, NIT, and LIT do lossless reversible text transforms, and are based upon LIPT Transform [10], [14], and [20]. They do not offer a significant increase of the execution time performance, because they use the same method of loading a dictionary as LIPT does, and the static dictionary and the code dictionary remain the same.

The transform that preserves the initial letter (ILPT) is similar to LIPT [20] excepting that the dictionary is sorted in blocks according to the initial letters of the words. Then the words of each block of letters are sorted in descending order of frequencies of occurrence of these words. The character that for LIPT is the length of the coded word, in this case is the first letter of the coded word, that is instead of $*c_{len}[c][c][c]$, for ILPT is $*c_{init}[c][c][c]$, where c_{init} represents the first letter of the coded word. Besides that, everything remains as for LIPT. Numerical Index Transform (NIT) uses variable addresses based on numbers instead of letters of the alphabet. This method was tested using a simple linear address with numbers, giving the 59,951 words from English dictionary addresses from 0 to 59,950.

Using this method on English dictionary D, sorted first by word length, then the frequency of their appearance, offered a performance lower than LIPT. As a result, the dictionary was sorted globally in descending order of the frequency of appearance of the words. In the new dictionary, the sorting of the blocks was not used. The transformed words are represented by the character "*" followed by the corresponding code of the respective word. This way the first word is coded as "*0", the 1000th word is coded as "*999", and so on. Special characters are treated the same way as for LIPT.

Combining the method used for NIT with the use of letters to specifying the *offset* was discovered another transform which is similar to the NIT, except that now, for the specification of the linear address of a word in the dictionary, the alphabet letters [a-z; A-Z] are used instead of numbers. This is called the Literal Index Transform (LIT).

The size of the dictionary of transformation is variable depending on the individual transform. For example, the text book1.txt, which had 767 KB, the size of the dictionaries of transformation obtained for ILPT, NIT and LIT are 294 KB, 344 KB, and 311 KB. It is noted that ILPT has the dictionary of transformation with the smallest size.

The frequency of the repeated words remains the same in the original text file and the transformed one, only the frequency of the characters changes. This factor, together with reducing the file size, contributes to a better compression by using these transforms. For all transforms for generating transformation, dictionary words have been sorted according to their frequency of occurrence. For ILPT words of the vector are sorted in descending order of frequency of their occurrence. For NIT the entire dictionary is considered a single block, and is sorted in descending order of frequency of occurrence of words. LIT uses the same structure as NIT. Sorting words according to frequency plays an important role in determining the size of the transformed file and its entropy. Arranging words in the decreasing order of their frequencies leads to shorter codes for more often used words and longer codes for less used words. This leads to obtain the file size smaller.

The coding scheme for the three transforms is shown in Fig. 1.

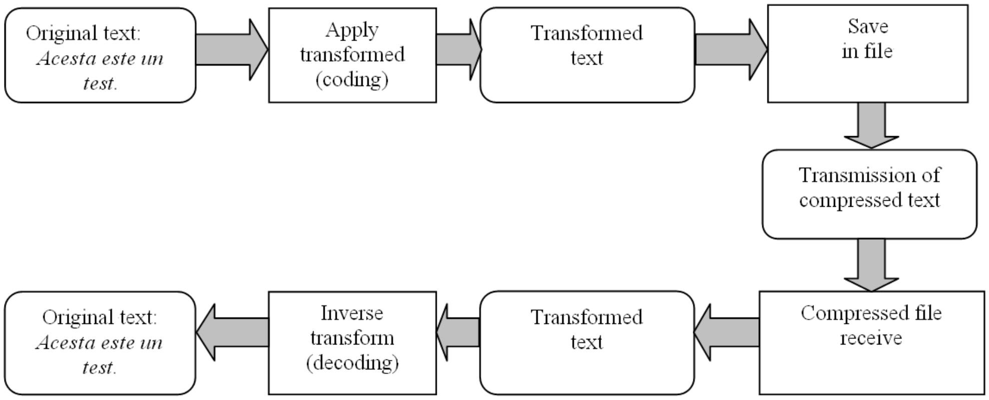

Fig. 1. Coding scheme for ILPT, NIT and LIT transforms

3. Compression Tests

Software compression [18]–[20] results are presented for the following files using Initial Letter Preservation Transform (ILPT), Literal Index Transform (LIT) and Numerical Index Transform (NIT) with the classic archiver WinRar (see Tables 1-3). There have been used some representatives Romanian text files and two test files, called "book1" and "book2", taken from the set of evaluation of lossless compression algorithms Calgary Corpus [17].

Table 1

ILPT Transform

File Name	File Size	ILPT Compression	WinRar Compression	
book1.txt	767	596	273	261
book2	612	456	179	161
creierul o enigma	494	328	139	122
regulament ordine interioara	84	50	14	10
tarzan of the apes	497	380	165	144
Yserver	260	158	14	12

Table 2

NIT Transform

File Name	File Size	NIT Compression	WinRar Compression	
book1.txt	767	697	273	233
book2	612	516	179	157
creierul o enigma	494	374	139	118
regulament ordine interioara	84	55	14	10
tarzan of the apes	497	434	165	141
Yserver	260	137	14	10

Table 3

LIT Transform

File Name	File Size	LIT Compression	WinRar Compression	
book1.txt	767	571	273	231
book2	612	431	179	154
creierul o enigma	494	310	139	117
regulament ordine interioara	84	55	14	10
tarzan of the apes	497	364	165	140
Yserver	260	124	14	9

4. Experimental Results

The following experimental results were obtained (see Figs. 2–7).

There can be noticed improvements in compression without classic archivers from 34 KB for 80 KB files up to 171 KB for 767 KB files.

For example, a file of 170 KB has about 1200 lines, which is not negligible. Transformation algorithm is based on the exploitation of redundancy, so the larger the file the better the compression.

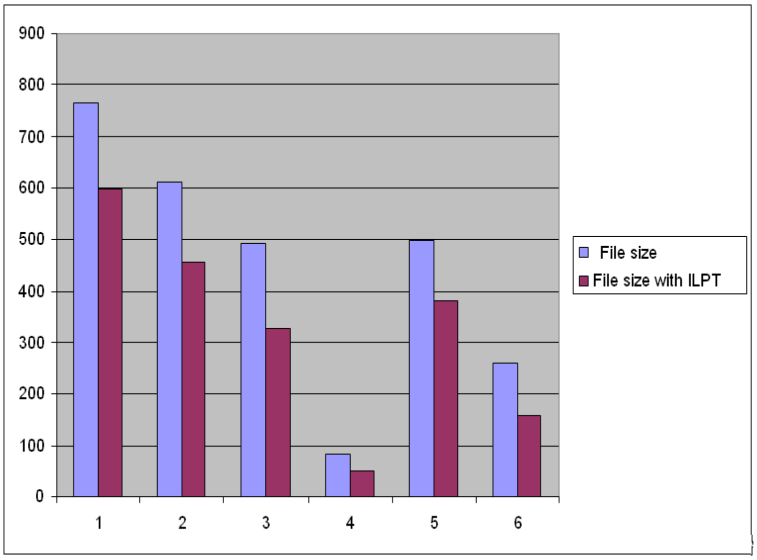

Fig. 2. The original file and ILPT transformed file

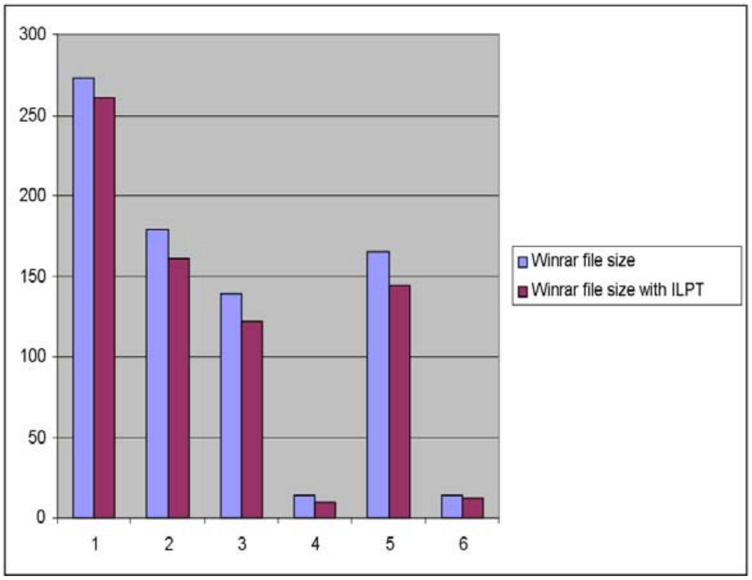

Fig. 3. WinRar archived file with and without ILPT

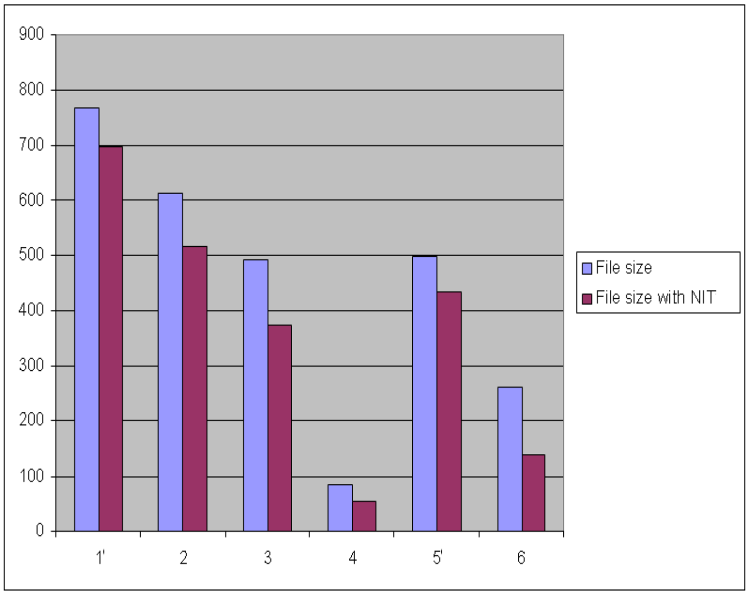

Fig. 4. The file with and without NIT

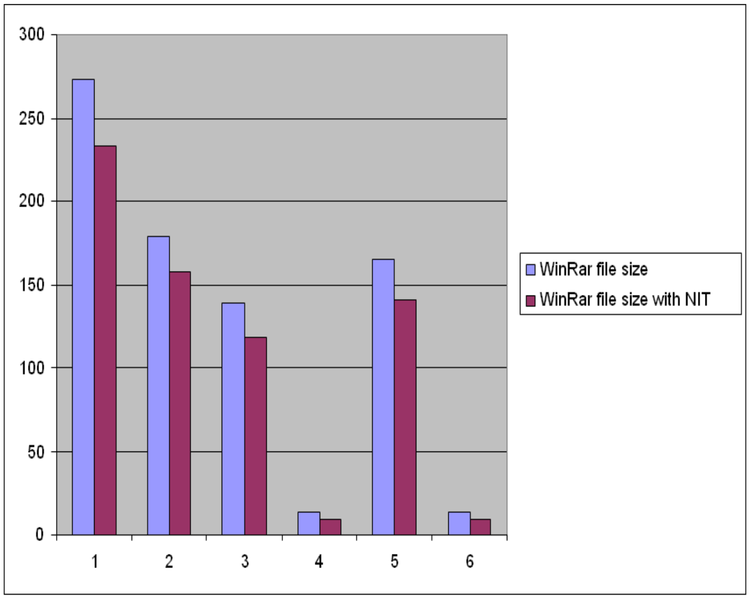

Fig. 5. WinRar archived file with and without NIT

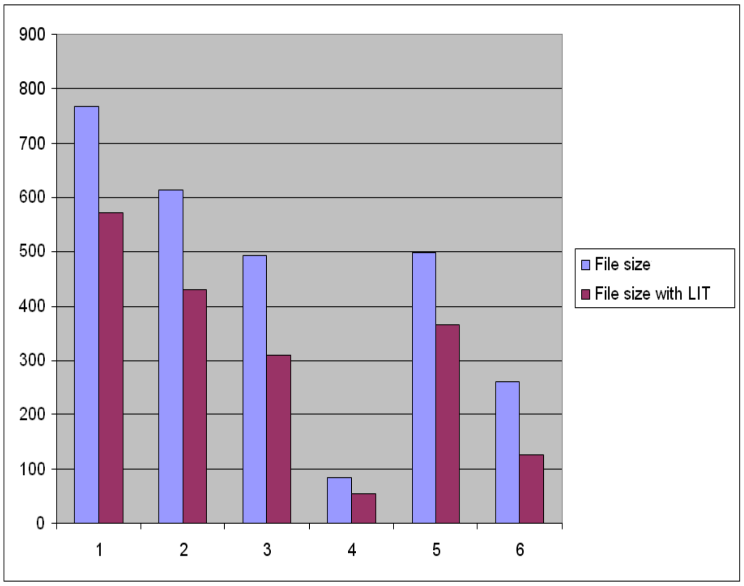

Fig. 6. File with and without LIT

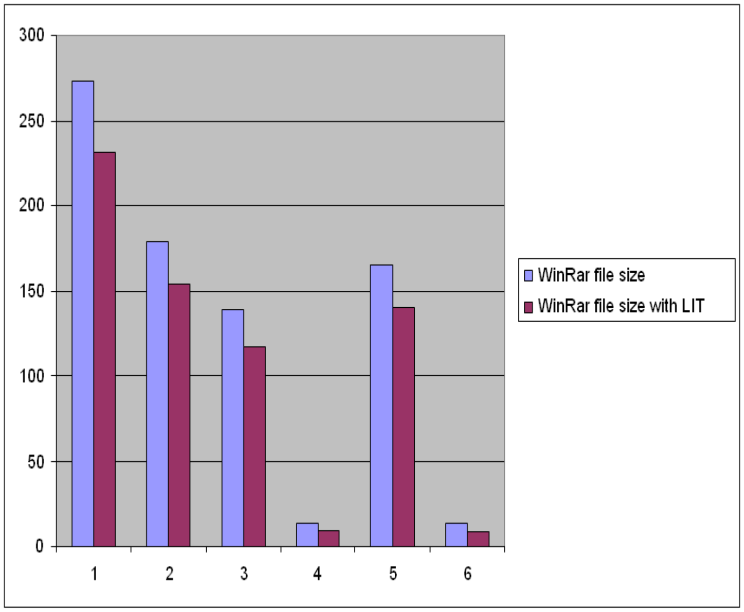

Fig. 7. WinRar archived file with and without LIT

Tables 4-6 are given in percentage of the rate of compression for archiving performed.

Table 4

ILPT Transform

File Name	ILPT Compression (%)	WinRar Compression (%)
book1.txt	22.29465	4.395604
book2	25.4902	10.05587
creierul o enigma	33.60324	12.23022
regulament ordine interioara	40.47619	28.57143
tarzan of the apes	23.54125	12.72727
Yserver	39.23077	14.28571

Table 5

NIT Transform

File Name	ILPT Compression (%)	WinRar Compression (%)
book1.txt	9.126467	14.65201
book2	15.68627	12.2905
creierul o enigma	24.2915	15.10791
regulament ordine interioara	34.52381	28.57143
tarzan of the apes	12.67606	14.54545
Yserver	47.30769	28.57143

Table 6

LIT Transform

File Name	ILPT Compression (%)	WinRar Compression (%)
book1.txt	25.55411	15.38462
book2	29.57516	13.96648
creierul o enigma	37.24696	15.82734
regulament ordine interioara	34.52381	28.57143
tarzan of the apes	26.76056	15.15152
Yserver	52.30769	35.71429

5. Conclusions

This paper presents important results in pre-compression processing for lossless algorithms using a set of transforms for different text files. If Burrows-Wheeler, Star, and Length-Index Preserving Transforms are already well-known and widely used for this purpose, the derived transforms (ILPT, NIT, and LIT) are more recent and their capabilities still not enough exploited. The present contribution focuses on the compression performances obtained using the three transform methods in the particular case of the Romanian text files. All three transforms present significant improvements over the original files in terms of compression rate. There is not a distinction to be seen between transforms, no one can say that one is better than the other.

As optimization of the compression process, some suggestions can be made:

- one can work with a database that retains all dictionary words at a time;
- one can define filters for words, for example, for an e-mail address;

- when the dictionary is formed, it may be lexicographical ordered, and not by the frequency of occurrence and then a "divide et impera" algorithm to search for that word can be used.

REFERENCES

1. *M. Burrows, D.J. Wheeler*, "A Block-Sorting Lossless Data Compression Algorithm", 1994, report available at: http://gatekeeper.dec.com/pub/DEC/SRC/research-reports/abstracts/src-rr-124.html
2. *M. Nelson*, "Data Compression with the Burrows-Wheeler Transform", September 1996, available at: http://dogma.net/markn/articles/bwt/bwt.htm
3. *M.A. Maniscalco*, "A Run Length Encoding Scheme for Block Sort Transformed Data", 2000, available at: http://www.geocities.com/m99datacompression/papers/rle/rle.html
4. *P. M. Fenwick*, "Block Sorting Text Compression", 1996, available at: ftp.cs.auckland.ac.nz
5. *T.C. Tell*, J.G. Cleary and I.H. Witten, Text Compression, Prentice Hall, Englewood Cliffs, NJ, 1990.
6. *R. Franceschini*, *H. Kruse, N. Zhang, R. Iqbal, A. Mukherjee*, "Lossless, Reversible Transformations that Improve Text Compression Ratios", Preprint of the M5 Lab, University of Central Florida, 2000.
7. *R. Franceschini, A. Mukherjee*, "Data compression using text encryption", Proceedings of the Third Forum on Research and Technology, Advances on Digital Libraries, ADL, 1996.
8. *H. Kruse, A. Mukherjee*, "Data compression using text encryption", Proceedings of Data Compression Conference, IEEE Comput. Soc., Los Alamitos, CA, 1997.
9. *M. R. Nelson,* "Star Encoding", Dr. Dobb's Journal, August, 2002.
10. *F.S. Awan, N. Zhang, N. Motgi, R.T. Iqbal, A. Mukherjee,* "LIPT: A Reversible Lossless Text Transform to Improve Compression Performance", Proc. of Data Compression Conf., Snowbird, UT, 2001.
11. *R. Rădescu,* Compresia fără pierderi – algoritmi şi aplicații, MatrixRom, Press Bucharest, 2003.
12. *R. Rădescu*, I. Bălăşan, "Recent Results în Lossless Text Compression Using the Burrows-Wheeler Transform (BWT)", Proceedings of IEEE International Conference on Communications 2004 (COMM04), pp. 105-110, Bucharest, Romania, 3-5 June 2004.
13. *R. Rădescu, G. Liculescu*, "Table Look-Up Lossless Compression Using Index Archiving", Proceedings of the Fifth International Workshop on Optimal Codes and Related Topics OC2007, Balchik, Bulgaria, 16-22 June 2007, pp. 216-221.
14. *R. Rădescu*, "Lossless Text Compression Using the LIPT Transform", Proceedings of the 7th International Conference Communications 2008 (COMM2008), pp. 59-62, Bucharest, Romania, 5-7 June 2008.
15. *R. Rădescu*, "Lossless Compression Tool for Medical Imaging", Proceedings of the 6th International Symposium on Advanced Topics in Electrical Engineering, ELTH & AIEER, pp. 265-267, November 20-21, 2008, Bucharest, Romania, Printech Press.
16. *R. Rădescu, C. Bontaş*, "Design and Implementation of a Dictionary-Based Archiver", Sci. Bull., Electrical Engineering Series C, University Politehnica of Bucharest, vol. 70, nr. 3, pp. 21-28, 2008.
17. The Calgary Corpus can be found at: ftp://ftp.cpsc.ucalgary.ca/pub/projects/text.compression.corpus/
18. Derived classes MyTabCtrl: http://www.codersource.net/mfc_ctabctrl.html
19. Ostream libraries: http://www.cplusplus.com/reference/iostream/ostream/
20. *Radu Rădescu*, "Transform Methods Used in Lossless Compression of Text Files", Romanian Journal of Information Science and Technology (ROMJIST), Publishing House of the Romanian Academy, Bucharest, vol. 12, nr. 1, pp. 101-115, 2009, ISSN 1453-8245.

http://www.agilemodeling.com/artifacts/classDiagram.htm

http://gatekeeper.dec.com/pub/DEC/SRC/research-reports/abstracts/src-rr-124.html

http://www.dogma.net/markn/articles/Star/

http://www.codeguru.com/forum/

http://www.codersource.net/codersource_mfc_prog.html

http://www.y0da.cjb.net