A Generalized Model Of A Classical Lempel-Ziv Parser

Jordan Genoff

Abstract : *A sequence parsing technique derived from the widely known classical Lempel-Ziv schema is presented. A comparative analysis of LZ77 and LZ78 is accomplished with aim to strictly determine their parsing properties and classify them as common ones and distinct ones. The latter are combined in a set of synthetic quasi-common properties, each of which coincides with any of its originators provided certain parameters or point of view. The original common properties and the synthetic common ones form a generalized Lempel-Ziv parser that can behave as both LZ77 and LZ78 in its ability to maintain a dictionary and to break an input sequence into sub-sequences.*

Key words : *sequence parsing, dictionary parsers, Lempel-Ziv parsers.*

INTRODUCTION

Soon after their promotion in the late 70-ies two amazing algorithms became a foundation of a new religion – the dynamic dictionary oriented sequence modelling and coding. The huge significance of LZ77 [1] and LZ78 [2] (the couple denoted below as LZ7X) has been undoubtedly confirmed in time by the enormous number of scientific and applied publications. Their properties have been deeply investigated and new techniques have been developed relying on these fundamental bases and again their properties have been analysed. Not to mention the fact that almost every general purpose data compression tool involves some LZ7X-like technique.

This paper concentrates on the source modelling part of these methods – the dictionary organization and maintenance. The reason is the specific Lempel-Ziv parsing approach and its ability to create a source model of more complex nature than just a probabilistic one. This feature makes these methods an appropriate solution to complex problems in sequence analysis, especially concerning sequential structure inference.

A need for a general model of this class of parsing algorithms arises because there should exist a way to evaluate their performance as a source modeling technique. The goal of this paper is to : first, precisely identify what is the same and what is different in the dictionary organization and maintenance of the two LZ7Xs and ; second, build a general model of operation that will encompass the properties of both original methods together.

CLASSICAL LEMPEL-ZIV PARSING

Initially devised and mathematically proven to be effective as data compression algorithms of the new for their time "phrase detecting/encoding" category, LZ7X appeared to be able to do much wider area of work. This is because of the very interesting way they construct and maintain the input source model or in LZ7X terms – the famous *dictionary*, after which this class of methods is named.

The dictionary is a set of phrases (sequences) of various lengths that have been identified (parsed) in the input stream from its beginning to the position which the processing has reached. The processed part of the stream itself is presented as a concatenation of parsed phrases. The way these sequences have been identified is a core mechanism of LZ7X. At each step the longest match between the beginning of the non-processed stream part and a dictionary phrase is searched. After the match is achieved at least one new phrase is appended to the dictionary – the concatenation of the identified prefix and the symbol right next to the end of the prefix.

The most important qualitative feature of the parsing is the dictionary adequacy – in what degree the set of phrases in the dictionary is relevant to the real properties of the input stream source. The major factors that influence the dictionary adequacy are the "greedy" nature of parsing and the specific dynamics of dictionary contents. They both result from the simplicity of operation and have always been a major motivation for further investigation and innovation.

Though having much in common as a basic principle of source modelling, LZ77 and LZ78 organize their dictionaries in rather different ways and hence they use rather different data structures and different techniques to manipulate and search their content.

COMPARATIVE ANALYSIS OF PROPERTIES

The original LZ77 and LZ78 will be presented by a comparative analysis of their features. The aim is to strictly determine their parsing properties and classify them as common ones and distinct ones.

The following Table 1 describes the similarities between LZ77 and LZ78 :

Common Features	
S01 :	Both algorithms operate on discrete input stream of symbols from finite alphabet.
S02 :	Both algorithms operate in discernable successive steps.
S03 :	There is a clear boundary (parsing boundary) between the already processed and still non-processed part of the input stream. After each operating step this boundary is unconditionally moved forwards – in a direction from the beginning to the end of the stream.
S04 :	Both are one-pass algorithms and no reprocessing of any part of the input stream occurs at any step.
S05 :	At each step the longest string at the beginning (the longest prefix) of the non-processed part of the input stream is identified as being present in a maintained set of strings (phrases) named "dictionary".
S06 :	The phrases in the dictionary are of various lengths. The dictionary always contains the empty phrase (zero-length phrase).
S07 :	At each step the dictionary is updated after it has been used for prefix identification. The update is performed by adding new phrases and/or removing existing phrases.
S08 :	At each step, when new phrases are added they depend in some way on : the current contents of the dictionary ; the identified prefix of the non-processed part of the stream ; the symbol next to the end of the identified prefix. The phrase, which is a concatenation of the identified prefix and the symbol next to it is always added to the dictionary.
S09 :	At the end of each step the parsing boundary is moved after the symbol, which is next to the end of the identified prefix.
S10 :	The dictionary always consists of phrases that can be found somewhere in the processed part of the stream.

Table 1. The similarities between LZ77 and LZ78.

The following Table 2 is a comparative list of differences between LZ77 and LZ78 :

	Distinct Features	
	LZ77	LZ78
D01 :	Upper limit for the identified prefix length :	
	There is a limit and it is one of the major parameters of the algorithm. It defines the maximum length of the prefix that the dictionary will be searched for. This length-constrained area at the beginning of the non-processed part of the input stream is named *buffer*.	There is not a limit. This means that the identified prefix could be as long as the longest matvhing phrase in the dictionary.
D02 :	Dictionary organization :	
	The dictionary is the ending sub-sequence of the processed part of the input stream. It is a linear sequence of a certain length and each of its sub-sequences with a length up to the length of the buffer is a phrase in the dictionary. This length-constrained area at the end of the processed part of the input stream is named *window*.	A collection of phrases organized as a digital search tree, according to the rules described in the next section (rules T1-T5).
D03 :	Dictionary update procedure at each step :	
	The adjacent buffer and window are moved forwards together with the parsing boundary (which is between them). Thus the phrase resulting from the concatenation of the identified prefix and the symbol next to its end enters the dictionary. Many other new phrases may enter the dictionary – these are sequences beginning somewhere at the end of the window and ending somewhere in the buffer. Many existing phrases may leave the dictionary – these are sequences beginning somewhere at the beginning of the window and ending nearby.	Only one new phrase is added to the dictionary and it is the concatenation of the identified prefix and the symbol next to its end. According to the dictionary organization this means that a new inheriting node is added to the tree and its parent is the node representing the identified prefix. No phrase is leaving the dictionary, which means that it is monotonically growing in size.
D04 :	Upper limit for the dictionary size :	
	The size of the dictionary is constrained by the length of the window. There is a maximum number of all phrases of length up to the length of the buffer, that can be placed in a window with a given length and given contents.	By definition, the size of the dictionary is unlimited, because the tree may grow without any limit.

D04 :	Dictionary contents 1 :	
	The dictionary may contain phrases that have never been identified as prefixes due to the possibility for putting them in it along with the identified prefixes (see D03). The dictionary may not contain phrases that have previously been identified as prefixes due to the possibility that such phrases may have left the dictionary (see D03).	The dictionary contains only phrases that have been identified as prefixes and besides, all of them for the whole processed part of the input stream.
D05 :	Dictionary contents 2 :	
	The dictionary may contain multiple instances of one and the same phrase.	The dictionary contains a single instance of each phrase.

Table 2. The differences between LZ77 and LZ78.

GENERALIZATION

The distinct properties of the two LZ7Xs as shown in Table 2 should be combined in a set of synthetic quasi-common properties, each of which coincides with any of its respective originators provided certain parameters or point of view. The original common properties and the synthetic common ones form a generalized Lempel-Ziv parser that can behave as both LZ77 and LZ78 in its ability to maintain a dictionary and to break an input sequence into sub-sequences.

Let $\{X_k\}_{k=-\infty}^{+\infty}$ be a discrete symbol stream that takes values in the finite alphabet $\mathcal{A}$ with cardinality A = $|\mathcal{A}|$. For $-\infty \le i \le j \le +\infty$ let X_i^j denote the sequence $(X_i, X_{i+1}, \ldots, X_j)$. Let X_1^L be the sequence of length L to be parsed.

Let b be the position of the first symbol of the non-processed part of X_1^L, which is the symbol next to the parsing boundary and thus is the beginning of a *buffer* $\mathrm{X}_b^{b+n_b-1}$ of length n_b. Let $\mathrm{X}_w^{w+n_w-1}$ be a *window* in the processed part of X_1^L with beginning at position w and of length of n_w. The concept of buffer and window is taken from LZ77.

Let the algorithm organize a dictionary D as a digital search tree (DST) according to the following rules :

T1 : Every node, except the root, contains an alphabet symbol.

T2 : Every node, except the leaves, may have one or more than one, but not more than A inheriting nodes.

T3 : All inheriting nodes of a given node contain different alphabet symbols.

T4 : The root represents the empty phrase.

T5 : Every node represents a phrase, produced as a concatenation of the symbols in the nodes on the path from the root to the given node.

The concept of dictionary organization as a DST is taken from LZ78. Such kind of dictionary is able to handle an arbitrary sequence with an arbitrary length and an arbitrary symbol configuration.

Let $\mathrm{X}_b^{b+n_p-1}$ of length n_p be the identified prefix in the buffer and X_{b+n_p} be the symbol next to the end of $\mathrm{X}_b^{b+n_p-1}$. Let the superscript (s) denote that the respective value or contents is for the s-th execution step while parsing X_1^L.

Several new descriptive elements are introduced as follows :

P1 : $b^{(s+1)} = \Phi_b(b^{(s)},\ldots)$ – a procedure to calculate the new position of the buffer beginning, which is the parsing boundary.

P2 : $n_b^{(s+1)} = \Phi_{n_b}(n_b^{(s)},\ldots)$ – a procedure to calculate the new length of the buffer.

P3 : $w^{(s+1)} = \Phi_w(w^{(s)},\ldots)$ – a procedure to calculate the new position of the window beginning.

P4 : $n_w^{(s+1)} = \Phi_{n_w}(n_w^{(s)},\ldots)$ – a procedure to calculate the new length of the window.

P5 : $D^{(s+1)} = \Phi_{DA}(D^{(s)}, \mathrm{X}_w^{w+n_w-1}, w^{(s+1)}, n_w^{(s+1)}, \mathrm{X}_b^{b+n_p-1}, X_{b+n_p},\ldots)$ – a procedure to append phrase(s) to the dictionary.

P6 : $D^{(s+1)} = \Phi_{DR}(D^{(s)}, \mathrm{X}_w^{w+n_w-1}, w^{(s+1)}, n_w^{(s+1)},\ldots)$ – a procedure to remove phrase(s) from the dictionary.

Combining the above described concepts taken from LZ77 and LZ78 and utilizing the framework P1-P6, a new representation of the differences between the two algorithms may be constructed and all distinct features may be represented as quasi-common features. This is shown in the following Table 3 :

	Synthetic (Quasi-Common) Features	
	LZ77	LZ78
G01 :	Dictionary organization :	
	The linear contents of the window is represented as a DST – every sub-sequence of $\mathrm{X}_w^{w+n_w-1}$ of length in $[1, n_b]$ is a node in the tree.	The dictionary is organized as a DST.
G02 :	Upper limit for the identified prefix length :	
	There is a buffer and it determines the upper limit, so $\max(n_p^{(s)}) = n_b$.	There is no buffer, which means the whole X_b^L plays role of a buffer, so $\max(n_p^{(s)}) \le n_b^{(s)} = L - b^{(s)} + 1$ or practically there is no limit. A limit may be enforced as a buffer of length n_b and the rules for LZ77 will apply.
G03 :	Upper limit for the dictionary size :	
	There is a window $\mathrm{X}_w^{w+n_w-1}$ of length n_w and it can contain a limited number of sub-sequences of length in $[1, n_b]$.	There is no window, which means the whole $\mathrm{X}_1^{b^{(s)}-1}$ plays role of a window, so practically there is no limit. A limit may be enforced as a window of length n_w and the rules for LZ77 will apply.

G04 :	Φ_b : Movement of the buffer beginning (parsing boundary) :	
	$b^{(s+1)} = b^{(s)} + n_p^{(s)} + 1$.	$b^{(s+1)} = b^{(s)} + n_p^{(s)} + 1$
D01 :	Φ_{n_b} : Modification of buffer length :	
	$n_b^{(s+1)} = n_b^{(s)} = n_b$	$n_b^{(s+1)} = n_b^{(s)} - n_p^{(s)} - 1$ if there is no explicit buffer, or as with LZ77 if there is one.
D02 :	Φ_w : Movement of window beginning :	
	$w^{(s+1)} = w^{(s)} + n_p^{(s)} + 1$	$w^{(s+1)} = w^{(s)} = 1$
D04 :	Φ_{n_w} : Modification of window length :	
	$n_w^{(s+1)} = n_w^{(s)} = n_w$	$n_w^{(s+1)} = n_w^{(s)} + n_p^{(s)} + 1$

Table 3. The synthetic quasi-common properties of LZ77 and LZ78.

The generalized model of LZ7X consists of bringing together three components : the concept of window and buffer from LZ77 ; the concept of DST organized dictionary from LZ78 ; the set of operators Φ that handle the differences between LZ77 and LZ78.

CONCLUSIONS AND FUTURE WORK

Though the subject of this paper may look like taken from the early days of the Lempel-Ziv era, it is a fact that no such kind of attempt for descriptive unification of the two algorithms was found during the comprehensive search that was carried out.

The generalized Lempel-Ziv parsing technique is a powerful tool with which to conduct full-scale performance estimation and parametric analysis of Lempel-Ziv parsing when used for sequential structure inference.

Future work will be directed at finding a proper way to represent the dictionary construction and maintenance in LZ7X derivative algorithms and thus joining them to this generalized model of dictionary organization.

REFERENCES

[1] Ziv, J., A. Lempel. A Universal Algorithm for Sequential Data Compression. IEEE Trans. Information Theory, vol. IT-23, no. 3, pp. 337-343, May 1977.

[2] Ziv, J., A. Lempel. Compression of Individual Sequences via Variable Rate Coding. IEEE Trans. Information Theory, vol. IT-24, no.5, pp. 530-536, Sep 1978.

ABOUT THE AUTHOR

Ass.Prof. Jordan Genoff, Department of Computer Systems, Technical University of Sofia at Plovdiv, Phone: +359 32 659 729, E-mail: jgenoff@tu-plovdiv.bg.