

# Using partial-matching approach with Sequitur for context-based coding

Gopal Lakhani, Radhakrishnan Sethuraman

Texas Tech University, Lubbock Texas, 79409-3104, USA

Sequitur (Nevill-Manning, DCC'94) is a powerful realization of the grammar-based text compression approach. First, it derives a context-free grammar, which generates the given input as its only language. Then it represents the grammar as a single sequence of symbols, where a symbol represents either a rule head or an alphabet, and finally it uses arithmetic coder to code the sequence. Sequitur encodes each symbol individually, i.e., it does not consider preceding symbols for predicting the next symbol. The reason is that Sequitur builds the grammar so that no pair of adjacent symbols can appear twice. We modified this requirement slightly. It allowed us to code a rule symbol in the context of the last alphabet of the preceding rule symbol. We followed the partial matching (PM) approach (Hoang, DCC'95) to realize code reduction. The crux of this approach is to maintain multiple dictionaries, each containing rules that begin with the same alphabet. Since such dictionaries are considerably smaller than the global dictionary of all rules, representing a rule using its index in an individual dictionary saves code bits.

Sequitur-PM builds the grammar like Sequitur, except for a small difference. It reads the next alphabet, 'b', from the input and appends it to the last symbol, 'A', of the current rule. If 'Ab' is not an existing digram in the grammar and the context-based coding is turned on, Sequitur-PM would reinsert the last alphabet, 'c', of 'A' and then append 'b'. We give an example. Let "cbabcbbbc\*bcxcbc" be the input; '\*' is used here just to denote an instance when the encoder decides to use context coding. Sequitur will generate the following grammar:  $\{S \rightarrow BaABCxC, C \rightarrow AA, A \rightarrow bc, B \rightarrow cb\}$ . Sequitur-PM will generate  $\{S \rightarrow BaABCxC, C \rightarrow AB_+c, A \rightarrow bc, B \rightarrow cb\}$ . Here,  $B_+$  denotes that B is coded in context of the preceding symbol. The decoder would have no difficulty removing the extra 'c'.

Experimental results to compare Sequitur (S) with Sequitur-PM (P) are given in the following table. Column 2 gives the input file size in bytes. Columns 3-4 give the number of rules generated by the two encoders. Column 5 gives the threshold on the size of individual dictionaries, when the context-based coding is commenced, and Column 6 gives the average of final dictionary sizes. Columns 7-8 gives the coding rates in bpc. Column 9 gives reduction in the code size in terms of percentage due to the PM approach. The average reduction is 6.71%.

1	2	3(S)	4(P)	5	6	7(S)	8(P)	9
book1	768771	27384	37687	60	251	2.89	2.645	8.48
book2	610856	23271	31901	65	100	2.54	2.39	5.91
news	377109	17537	23698	50	119	2.96	2.78	6.08
paper1	53161	3415	4561	20	25	3.01	2.874	4.52
paper2	82199	4448	5986	25	34	2.95	2.70	9.43
bib	111261	5535	7754	20	25	2.57	2.42	5.84