PPM(Prediction by Partial Match)

The Huffman and arithmetic coders are sometimes referred to as the ***entropy coder***. These methods normally use an order (0) model. If a good model with low entropy can be built external to the algorithms, these algorithms can generate the binary codes very efficiently. One of the most well known modeler is ***"prediction by partial match" (PPM)*** [ClW84; Moff90].

PPM uses a finite context Order (k) model where k is the maximum context that is specified ahead of execution of the algorithm. The program maintains all the previous occurrences of context at each level of k in a table or a trie-like data structure with associated probability values for each context.

Exclusion Principle: If a context at a lower level is a suffix of a context at a higher level, this context is excluded at the lower level.

Escape Character: At each level and for each distinct context at that level (except the level with $k = -1$), an *escape character* is defined whose frequency of occurrence is assumed to be equal to the number of distinct characters occurring for a given context encountered in the text so far encoded, at that context level for the purpose of calculating its probability. The escape character is required to handle the situation when the encoder encounters a new context never encountered before at any context level to give the decoder a signal that the context length has to be reduced by 1.

Algorithm: During the encoding process, the algorithm estimates the probability of the occurrence of some given *next character* in the text stream as follows: the algorithm tries to find the current context of maximum length k for this character in the context table or trie. If the context is not found, it passes the probability of the escape character at this level for this context and goes down one level to $k-1$ context table to find the current context of length $k-1$ and the process is repeated. If it continues to fail to find the context, it may go down ultimately to $k=-1$ level corresponding to equiprobable level for which the probability of any next character is $1/|A|$. If, on the other hand, a context of length q, $0<=q<=k$, is found, then the probability of this next character is estimated to be the product of probabilities of escape characters at levels $k, k-1, . . ., q+1$ multiplied by the probability of the context found at the qth level. This probability value is then passed to the backend entropy coder (arithmetic coder) to obtain the encoding. Note, at the beginning there is no context available so the algorithm assumes a model with $k = -1$. The context lengths are shorter at the early stage of the encoding when only a few contexts have been seen. As the encoding proceeds, longer and longer context become available.

Methods of Handling "Zero Frequency"

Method C :The method to assign probability to the escape character is called the method C and is as follows: at any level, with the current context, let the total number of symbols seen previously be n_t and let n_d be the total number of *distinct* context. Then the probability of the escape character is given by $n_d/(n_d + n_t)$. Any other character which appeared in this context n_c times will have a probability $n_c/(n_d + n_t)$. The intuitive explanation of this method, based on experimental evidence, is that if many distinct context are encountered, then the escape character will have higher probability but if these distinct context tend to appear too many times, then the probability of the escape character decreases. The PPM method using method C for probability estimation is called PPMC algorithm. There are a few other variations:

Method A: PPMA uses method A which simply assigns a count of 1 to escape character yielding its probability to be $1/(n_t +1)$ and the probability of the character which appeared n_c times is $n_c/(n_t+1)$.

PPMB PPMB is very similar to PPMC except that the probability of a symbol is $(n_c-1)/(n_d + n_t)$, that is 1 is subtracted from the count n_c If n_c is 1, since n_c-1 becomes 0, no probability is assigned to the symbol and the count for the escape character is increased by 1.

PPMD: In PPMD, the escape character gets a probability of $n_d/2n_t$ and the symbol gets a probability $(2n_c-1)/2n_t$. All of these methods have been proposed based on practical experience and have only some intuitive explanation but no theoretical basis. PPMD performs better than PPMC which is better than either PPMA or PPMB.

PPM*: In one version of PPM, called PPM*, an arbitrary length context is allowed which should give the optimal minimum entropy. In practice a model with $k = 5$ behaves as good as PPM* [ClTW95]. Although the PPM family of algorithms performs better than other compression algorithms in terms of high compression ratio or low BPC, it is very computation intensive and slow due to the enormous amount of computation that is needed as each character is processed for maintaining the context information and updating their probabilities.

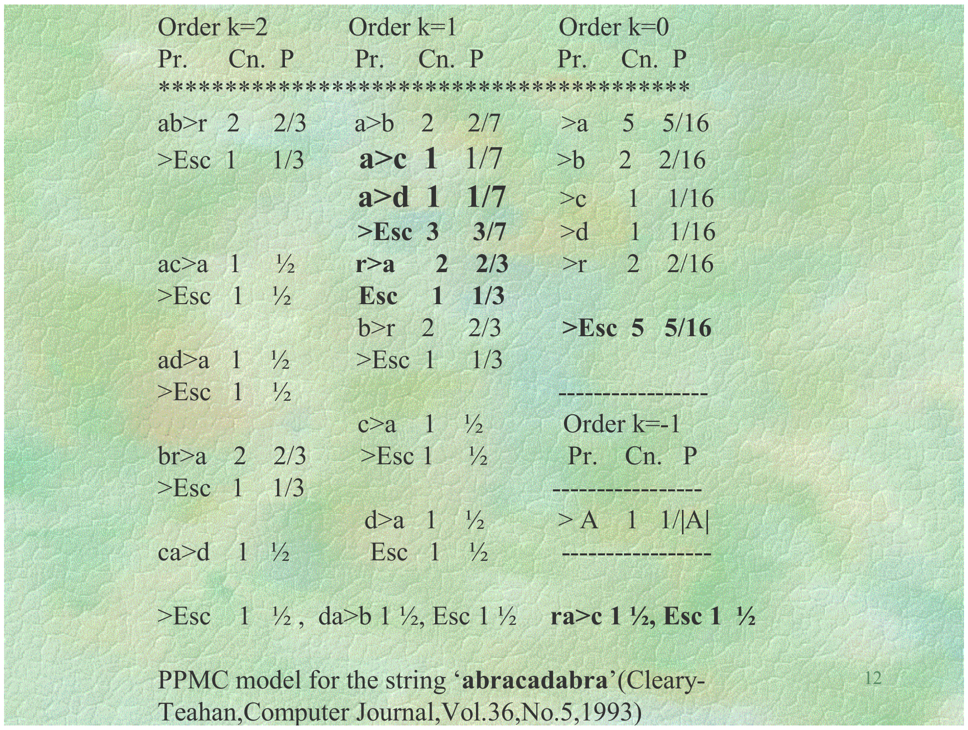
PPM* with Deterministic Context: The idea is that if a long context used only once, it is very unlikely that it will encounter a new symbol once. Contexts that are always followed by the same symbol are called **deterministic** context. The algorithm first looks for the longest deterministic context. If the symbol to be encoded does not appear in this context, an escape character is emitted and then the algorithm defaults to normal PPM*. It has been experimentally verified that a deterministic context length of 5 is as good as PPM* for all English text.

PPMD+ : It is PPMD with a training set**.**

PPMX: See “Managing Gigabytes”, p.67. Also Read Section 2.5 from this book.

Further reading from Salomon, Sayood and Moffat-Turpin.

An Example



Order k=2			Order k=1			Order k=0		
Pr.	Cn.	P	Pr.	Cn.	P	Pr.	Cn.	P
ab>r	2	2/3	a>b	2	2/7	>a	5	5/16
>Esc	1	1/3	**a>c**	**1**	1/7	>b	2	2/16
			a>d	**1**	**1/7**	>c	1	1/16
			>Esc	**3**	**3/7**	>d	1	1/16
ac>a	1	½	**r>a**	**2**	**2/3**	>r	2	2/16
>Esc	1	½	**Esc**	**1**	**1/3**			
			b>r	2	2/3	**>Esc**	**5**	**5/16**
ad>a	1	½	>Esc	1	1/3			
>Esc	1	½				-----------------		
			c>a	1	½	Order k=-1		
br>a	2	2/3	>Esc	1	½	Pr.	Cn.	P
>Esc	1	1/3				-----------------		
			d>a	1	½	> A	1	1/|A|
ca>d	1	½	Esc	1	½	-----------------		

>Esc 1 ½ , da>b 1 ½, Esc 1 ½ **ra>c 1 ½, Esc 1 ½**

PPMC model for the string '**abracadabra**'(Cleary-Teahan,Computer Journal,Vol.36,No.5,1993)

Char	Probabilities Without Exclusion	Probabilities With Exclusion	No. of bits
c	½	½	-log(1/2)=1 bit
d	½, 1/7	½, 1/6	-log(1/2*1/6)=3.6 bits
t	½,3/7,5/16,1/|A|	½,3/6,5/12,1/(|A|-5)	-log(1/2.3/6.5/12.1/251)=11.2 bits(?)

Note for ‘d’, for Order(1) model the probability is increased to 1/6 (rather than 1/7) with exclusion. This is because ‘c’ appeared in the context of ‘ra’ and therefore the context a->c will never be used at lower level and therefore should be excluded in estimating the probabilities at level 1. Similarly, the escape probability for t at Order (1) is reduced to 3/6 and so on. Since ‘b’,’c’ and ‘d’ appear in the context of ‘a->’, these are removed from Order(0) context, so that Esc probability for Order(0) is 5/12. In order (-1) , the Esc probability is 1/251 since 5 characters appear before (256-5=251). Note, Sayood descibes a slightly different method for Esc probability estimate.