Fast Codes for Large Alphabet Sources and Its Application to Block Encoding

Boris Ryabko[1]
Siberian State University of Telecommunication and Computer Science; Novosibirsk, 630102, Russia
e-mail: ryabko@adm.ict.nsc.ru

Jaakko Astola
Tampere International Center for Signal Processing at the Technical University of Tampere, Finland
e-mail: Jaakko.Astola@tut.fi

The computational efficiency of lossless data compression for large alphabets has attracted attention of researches for ages due to its great importance in practice. The point is that, on the one hand, often a source alphabet is very large or even infinite (see, for example, [2]) and, on the other hand, for many adaptive codes the speed of coding depends substantially on the alphabet size. Thus, the number of operations of an obvious (or naive) method of updating the cumulative probabilities is proportional to the alphabet size N. Jones [1] and Ryabko [4] have independently suggested two different algorithms of updating, which perform all the necessary transitions between individual and cumulative probabilities in $O(\log N)$ operations. Later many such algorithms have been developed and investigated in numerous papers, see for a review, for example, [3].

In this paper we suggest a method for speeding up codes based on the following main idea. Letters of the alphabet are put in order according to their probabilities (or frequencies of occurrence), and the letters with probabilities close to each other are grouped in subsets (as new super letters), which contain letters with small probabilities. The key point is the following: equal probability is ascribed to all letters in one subset, and, consequently, their codewords have the same length. This gives a possibility to encode and decode them much faster than if they are different. Then each subset of the grouped letters is treated as one letter in the new alphabet, whose size is much smaller than the original alphabet. Such a grouping can increase the redundancy of the code. It turns out, however, that a large decrease in the alphabet size may cause a relatively small increase in the redundancy. Since the frequencies are changing after coding of each message letter, the order should be updated. Now there exist algorithms and data structures, which give a possibility to carry out the updating using a few operations per message letter, see [5, 3].

Let us give some definitions. Let $A = \{a_1, a_2, \dots, a_N\}$ be an alphabet with a probability distribution $\bar{p} = \{p_1, p_2, \dots, p_N\}$ where $p_1 \geq p_2 \geq \dots \geq p_N, N \geq 1$. The distribution can be either known a priory or estimated on the basis of statistics. Let letters from the alphabet A be grouped as follows : $A_1 = \{a_1, a_2, \dots, a_{n_1}\}$, $A_2 = \{a_{n_1+1}, a_{n_1+2}, \dots, a_{n_2}\}, \dots, A_s = \{a_{n_{s-1}+1}, a_{n_{s-1}+2}, \dots, a_{n_s}\}$ where $n_s = N, s \geq 1$. We define the probability distribution π and the vector $\bar{m} = (m_1, m_2, ..., m_s)$ by $\pi_i = \sum_{a_j \in A_i} p_j$ and $m_i = (n_i - n_{i-1}), n_0 = 0, i = 1, 2, \dots, s$, correspondingly. We intend to encode all letters from one subset A_i by codewords of the same length. For this purpose we ascribe equal probabilities to the letters from A_i by $\hat{p}_j = \pi_i / m_i$. Such an encoding causes some redundancy, which is $r(\bar{p}, \bar{m}) = \sum_{i=1}^{N} p_i \log(p_i/\hat{p}_i)$.

The suggested method of grouping implies that the probabilities (or their estimates) are ordered. That is why we are interested in an upper bound for the redundancy, given by $R(\bar{m}) = \sup_{\bar{p} \in \bar{P}_N} r(\bar{p}, \bar{m})$, $\bar{P}_N = \{p_1, p_2, \dots, p_N : p_1 \geq p_2 \geq \dots \geq p_N\}$. The following theorems gives a possibility to calculate the redundancy and to find its asymptotic estimation.

Theorem. *For each $\bar{m} = (m_1, m_2, ..., m_s)$ the following equality for the redundancy is valid:* $R(\bar{m}) = \max_{i=1,\dots,s} \max_{l=1,\dots,m_i} l \log(m_i/l)/(n_i + l)$, *where, as before,* $\bar{m} = (m_1, m_2, ..., m_s)$, $n_i = \sum_{j=1}^{i} m_j$, $i = 1, ..., s$.

Corollary. *If we denote the extra redundancy by δ and apply the proposed scheme to the arithmetic code and N-symbol alphabet, we obtain the time of encoding and decoding $c(\log\log N + \log(1/\delta)) + c_1$ instead of $c \log N + c_2$ for a usual code, $N \to \infty$.*

The practically interesting question is how to find a grouping which minimizes the number of groups for a given upper bound of the redundancy δ. The theorem can be used as the basis for such an algorithm. One of such algorithms is implemented and can be used for practical needs, see http://www.ict.nsc.ru/∼ryabko/ GroupYourAlphabet.html.

The suggested method of grouping is applied to block coding of stationary ergodic sources with unknown statistics. The main problem of block encoding is that the number of blocks grows exponentially when the block length grows. In fact, it means exponential increasing of the input alphabet. In order to surmount this obstacles we suggest applying the described method of grouping to the set of all possible blocks in such a way that the redundancy caused by grouping is relatively small whereas the size of new alphabet is much less than the number of the possible blocks. As a result, the average number of operations per a source letter will be exponentially less than for usual block coding.

[1]Supported by Tampere International Center for Signal Processing at the Technical University of Tampere, Finland, and by INTAS under Grant no. 00-738. The part of results was presented at International workshop "Trends and recent achievements in information technology", 16-18 May 2002, Cluj-Napoca, Romania.

References

[1] D. W. Jones. "Application of splay trees to data compression", *Communications of the ACM*, v 31, n. 8,1988, pp.996-1007.

[2] J.C. Kieffer, E.H. Yang. "Grammar-based codes: a new class of universal lossless source codes", *IEEE Trans. Inform. Theory*, v.46 (2000), no. 3, 737–754.

[3] A. Moffat and A. Turpin. *Compression and Coding Algorithms.* Kluwer Academic Publishers, 2002.

[4] B. Ya. Ryabko. "A fast sequential code", *Dokl. Akad. Nauk SSSR* v.306 (1989), no. 3, pp.548–552 (Russian); translation in *Soviet Math. Dokl.*, v. 39 (1989), no. 3, pp. 533–537.

[5] B.Ryabko, J.Rissanen. " Fast adaptive arithmetic code for large alphabet sources with asymmetrical distributions", *IEEE Communications Letters,* 2003, n.1. pp. 33- 35.