Appl. Math. and Comp. Sci., 1995, vol.5, No.4, 717–742

UNIVERSAL DATA COMPRESSION BASED ON APPROXIMATE STRING MATCHING

ILAN SADEH*†

Two practical source coding schemes based on approximate string matching are proposed. One is an approximate fixed-length string matching data compression combined with a block-coder based on the empirical distribution, and the other is an LZ-type quasi-parsing method by approximate string matching. A lemma on approximate string matching, which is an extension of the Kac Lemma, is proved. It is shown, based on the lemma, that the former deterministic algorithm converts the stationary and ergodic source, u, into an output process v. On the assumption that v is a stationary process, after the scheme has run for the infinite time, it is shown that the optimal compression ratio $R(D)$ is achieved. This reduces the problem of the universal lossy coder to the proof of stationarity of the output process v in the proposed algorithm. A similar result holds for the latter algorithm in the limit of the long database size, which is the suffix of the infinite database generated by the former algorithm. The duality between the two algorithms is proved. The performance of the two algorithms and their sub-optimal versions are evaluated. The main advantages of the proposed methods are the asymptotic sequential behaviour of the encoder and the simplicity of the decoder.

1. Introduction

Approximate string matching is a technique with considerable theoretical merits and a practical promise. Our major motivation comes from data compression theory, although we believe that our results might have a wide range of applications in pattern recognition theory, artificial intelligence, biological research and, in particular, the Genome Project. Its algorithmic complexity and data structure were studied by Landau and Vishkin (1986), Galil and Giancarlo (1988) and Storer (1990). However, we focus our attention on a probabilistic aspect of the problem, i.e. universal source coding.

Universal source coding algorithms that achieve the theoretical bound $R(D)$ are still unknown. Such algorithms may be useful in image and voice compression, medical applications and DNA sequences compression. In our context, the most appropriate application is in multimedia, where voice and images are transmitted on the same

* Department of Electrical and Electronics Engineering, Center for Technological Education and Research, Holon, Affiliated with Tel Aviv University, P.O.Box 305, Holon, 58102 Israel, e-mail: sade@math.tau.ac.il

† Department of Math. and Computer Sciences, Ben-Gurion University, Beer Sheva 84120, Israel, e-mail: sade@ivory.bgu.ac.il

channels and the decoder in the universal receiver must be cheap. Consequently, bandwidth reduction and cheap decoding procedures are the necessary first step to bring multimedia closer to reality.

The crucial point is to choose a compression method which is universal for all image sources and realizable. We believe the algorithms described below are the best fitting for such an application. However, MPEG (see (Le Gall, 1991) for details) has chosen a different technology based on DCT (Discrete Cosine Transform) to compress the data rate and a cascaded two-dimensional variable length coding scheme. We think that our algorithms are superior to this strange combination of old technologies, which has no theoretical proof of optimality. Video coding is a promising application because almost surely there is an approximate "string" matching between a two-dimensional block in the current frame and a block in the previous frames in the spatial neighbourhood. Moreover, our methods are performed in such a way that the encoding and decoding procedures do not have to wait until the whole frame is transmitted. In this sense, it is suitable for ATM networks.

The first practical universal source coding were proposed by Lempel and Ziv (1977; 1978) for the compression of sequences by a lossless encoder. Their algorithms are based on a parsing procedure which creates a new phrase as soon as a prefix of the still unparsed part of the string differs from all preceding phrases. A fast implementation was given by Welch (1984). The asymptotic behaviour of this implementation through suffix trees was studied by Szpankowski (1993). A suffix tree construction algorithm was presented by McCreight (1976) and a simple algorithm for sorting the suffixes of a string was given by Manber and Myers (1993). Digital search trees were used in (Jacquet and Szpankowski, 1995) to obtain second-order properties of the LZ parsing scheme. The LZ parsing algorithms play a crucial role in such applications as efficient transmission of data (Lempel and Ziv, 1977; 1978), estimation of entropy (Ziv, 1978), discriminating between information sources (Gilbert and Kadota, 1992), testing randomness, estimating the statistical model of individual sequences (Plotnik *et al.*, 1992), etc.

Berger (1971), Blahut (1987), Gray (1975), Ornstein and Shields (1990), Sadeh (1995) and others proved the existence of optimal codes subject to a fidelity criterion. These works vary in the assumptions on the class of sources, the fidelity criterion and the type of convergence of the rates to the rate distortion function $R(D)$. In most cases, the construction of a code involves the following step: given a block of source symbols, find that code in an exponentially large class of codes, which performs best for this block. This task is prohibitively complex even for modest codebook sizes and makes these schemes impractical for implementation. Most of the schemes postulate that an *a priori* distribution of the source or some statistical information are known at both ends of the data link. Such postulates are inappropriate in the design of a universal machine.

We assume that the source process is ergodic and stationary. Its statistics is unknown, but we assume that the infinite time has elapsed since the beginning. So, actually, our convergence proofs are based on the fact that the source statistics is already known in the encoder but unknown in the decoder. We propose two practical source coding schemes based on approximate string matching. The first one is an

approximate fixed length string matching data compression method combined with a block coder based on the empirical distribution and the other is an LZ-type quasi parsing method by approximate string matching. The output process, which is a database, is also assumed to be stationary and ergodic. This assumption is explained and an informal proof is given. It is shown that in the former algorithm the compression rate converges to the theoretical bound $R(D)$ as the string length tends to infinity. We note that the decoding algorithm is very simple. The encoding algorithm is exponential at the beginning, but it is implementable. As time elapses, the encoding algorithm tends to be asymptotically linear with respect to the size of the database string. The algorithm of Wyner and Ziv (1989) is obtained as a special case of no distortion, i.e. $D = 0$.

A similar result holds for the latter algorithm in the limit of the long database generated by the former one. The algorithm of Lempel and Ziv is obtained as a special case of no distortion for the quasi parsing method by approximate string matching.

We adopt the terminology and some of the reasoning of Wyner and Ziv (1989), and Ornstein and Weiss (1993) in their work concerning lossless algorithms.

There is some literature on the probabilistic analysis of problems of approximate pattern matching (Arratia and Waterman, 1989; Arratia *et al.*, 1990; Luczak and Szpankowski, 1995; Sadeh, 1993a; Steinberg and Gutman, 1993). In particular, we would like to mention two references related to probabilistic analysis of pattern matching in the context of computer sciences (Knuth, 1981) and combinatorial probability (Pittel, 1985). The papers (Arratia and Waterman, 1989) and (Arratia *et al.*, 1990) explore the properties of the length of the longest substring that can be approximately recopied in context of genetics and biological research.

The paper is organized as follows. Section 2 is devoted to the principal lemma of approximate string matching, the extension of Kac's Lemma (Kac, 1947) and its new proof. Section 3 deals with the description and proof of convergence in probability of the algorithms. The section includes definitions and theorems about the duality of the minimal first repetition event and the longest first new block event, as well as the limits theorems. In Section 4 we give a short summary.

2. The Principal Lemma of Approximate String Matching

We present an extension of Kac's Lemma (Kac, 1947), based on ideas of L.H. Ozarow and A.D. Wyner as presented in the paper of Wyner and Ziv (1989).

Consider a finite valued stationary infinite sequence v defined on an alphabet V. Let v_i^j denote a sample sequence between positions i, j in the sequence v. Let B be a set of strings of length l taken from the space of all possible strings of length l, defined on V, i.e. $B \subseteq V^l$ such that $\Pr(B) > 0$. We denote by Y_n a string of length l starting from position n, $Y_n = v_n^{n+l-1}$, and we say that two strings Y_n, Y_m are approximately matched with respect to B if $Y_n \in B$, $Y_m \in B$. We denote the conditional probability that an approximate match with respect to B occurs for the first time at step k, by

$$Q_k(B) = \Pr\left\{Y_k \in B; \quad Y_j \notin B; \quad 1 \leq j \leq k-1 | Y_0 \in B\right\} \tag{1}$$

The average recurrence time assigned to B is defined by

$$\mu(B) = \sum_{k=1}^{\infty} kQ_k(B) \tag{2}$$

The event that in the realizations of v we can find members of B, is

$$A = \Big\{Y_n \in B \text{ for some } n, \quad -\infty < n < \infty\Big\}$$

Lemma 1. *(The Extended Kac Lemma) Under the above assumptions,*

$$Pr\Big\{A\Big\} = \Pr\Big\{Y_0 \in B\Big\}\mu(B) \tag{3}$$

In particular, for stationary ergodic processes, we have

$$1 = \Pr\{Y_0 \in B\}\mu(B) = \Pr\{B\}\mu(B) \tag{4}$$

Proof. We slice the event A as follows:

$$A_+ = \Big\{Y_n \in B \text{ for some } n, \quad 0 \le n < \infty\Big\}$$

$$A_- = \Big\{Y_n \in B \text{ for some } n, \quad -\infty < n \le -1\Big\}$$

Then

$$A = A_- \cup A_+ = A_+A_- + A_+A_-^c + A_+^cA_-$$

where $+$ denotes the disjoint union. We first show that

$$\Pr\Big\{A_+A_-^c\Big\} = \Pr\Big\{A_+^cA_-\Big\} = 0$$

Intuitively, it is clear that in the ergodic case, in any time slot of infinite length an event must occur. However, for the general non-ergodic case, if the event A_+ occurs, then there is a smallest $j \ge 0$ such that $Y_j \in B$. Thus

$$\Pr(A_+A_-^c) = \sum_{j=0}^{\infty} \Pr\Big\{Y_n \notin B, \quad -\infty \le n < j; \quad Y_j \in B\Big\}$$

But since the sequence $\{Y_n\}$ is stationary, the summand does not depend on j and therefore it must vanish. Similarly, $\Pr\Big\{A_+A_-^c\Big\} = \Pr\Big\{A_+^cA_-\Big\} = 0$.

Now, if the two events A_+ and A_- occur, there must be the smallest $j \geq 0$ such that $Y_j \in B$, and the smallest $k > 0$ such that $Y_{-k} \in B$. Thus

$$\begin{aligned}\Pr(A) = \Pr(A_+A_-) &= \sum_{k=1}^{\infty}\sum_{j=0}^{\infty} \Pr\Big\{Y_n \notin B, \quad -k+1 \leq n < j; \quad Y_{-k} \in B; \quad Y_j \in B\Big\} \\ &= \sum_{k=1}^{\infty}\sum_{j=0}^{\infty} \Pr\Big\{Y_{-k} \in B\Big\} \Pr\Big\{Y_n \notin B, \quad -k+1 \leq n < j; \quad Y_j \in B \Big| Y_{-k} \in B\Big\} \\ &= \sum_{k=1}^{\infty}\sum_{j=0}^{\infty} \Pr\Big\{Y_0 \in B\Big\} Q_{j+k}(B)\end{aligned}$$

where the last step follows from the stationarity of Y_n. For $k \geq 1$, $Q_i(B)$ appears in the last summation exactly i times, i.e. for (j,k) in $(0,i),(1,i-1),\ldots,(i-1,1)$. Thus

$$\Pr(A) = \Pr\Big\{Y_0 \in B\Big\} \sum_{i=0}^{\infty} iQ_i(B) = \Pr\Big\{Y_0 \in B\Big\}\mu(B)$$

In particular, for stationary ergodic sources, we have,

$$1 = \Pr\{Y_0 \in B\}\mu(B) = \Pr\{B\}\mu(B)$$

■

3. Algorithms and Convergence to $R(D)$

3.1. Definitions

Consider an ergodic and stationary finite valued source u, defined on a finite alphabet U. Let $\bar{u}$ denote a sample sequence or a block in the sequence u. The sequence u_{-n}^{-1} is our data base.

Definition 1. We define $L_n(u)$ to be the first index such that the string $u_0 \ldots u_{L-1}$ is not a substring of the database u_{-n}^{-1}. That is, for $n = 1, 2, \ldots$ $L_n(\bar{u})$ is the smallest integer $L > 0$ such that

$$u_0^{L-1} \neq u_{-m}^{-m+L-1}, \qquad L \leq m \leq n$$

$$L_n(u) = \inf\big(L > 0 : \quad u_0^{L-1} \neq u_{-m}^{-m+L-1}, \quad L \leq m \leq n\big)$$

Definition 2. The random variable $N_l(\bar{u})$ for $l > 0$ is the smallest integer $N \geq l$ such that

$$u_0^{l-1} = u_{-N}^{l-1-N}$$

In other words,

$$N_l(\bar{u}) = \inf\big(N \geq l : u_0^{l-1} = u_{-N}^{l-1-N}\big)$$

Given alphabets U and V, a distortion measure is any function $d : |U \times V| \to \mathbb{R}^+$. The function d measures the distortion (cost, penalty, loss etc.) suffers each time when the source produces letters $u \in U$ and the letters $v \in V$ are presented to the user. Usually, the range of d is finite and without loss of generality we may assume that, for all k, $\min_l d(u_k, v_l) = 0$.

Let $\rho_l(\bar{u}; \bar{v})$ denote the average value of the 'per letter' distortions for the letters that comprise the block $\bar{u}$, that is

$$\rho_l(\bar{u}; \bar{v}) = \frac{1}{l} \sum_{k=1}^{l} d(\bar{u}_k; \bar{v}_k) \tag{5}$$

We omit the subscript l. The pair $(\bar{u}_k; \bar{v}_k)$ denotes the letters at the k-th position at the source and the user, respectively. The distortion is assumed to be memoryless, that is, independent of the neighbouring letters.

Definition 3. For each sample sequence $\bar{u}$ of length l, taken from the sequence u, we define the set

$$D\text{-Ball}(\bar{u}) = \left\{ \bar{v} |\ \rho(\bar{u}, \bar{v}) \leq D \right\}$$

We generalize Definitions 1, 2 to the lossy case by using the sequence v_{-n}^{-1} as our data base.

Definition 4. For each sample sequence $\bar{u}$ we define the random variable

$$DL_n(\bar{u}, v_{-n}^{-1}) = \max_{\bar{v}:\rho(\bar{u},\bar{v})\leq D} L_n(\bar{v}, v_{-n}^{-1})$$

In other words, using the sequence v_{-n}^{-1} as our data base, we continue the sequence $u_0, u_1, \ldots$ until there is no string $\bar{v}$ of length L, in the D-Ball surrounding the string $\bar{u} = u_0 \ldots u_{L-1}$ such that $\bar{v}$ is a substring of the database v_{-n}^{-1}. We define $DL_n(\bar{u}, v_{-n}^{-1}) = L$.

Definition 5. For each sample sequence $\bar{u}$ we define the random variable

$$DN_l(\bar{u}, v_{-k}^{-1}) = \min_{\bar{v}:\rho(\bar{u},\bar{v})\leq D} N_l(\bar{v}, v_{-k}^{-1})$$

In other words, we choose all the strings of length l taken from V^l that are neighbours of $u_0^{l-l} = \bar{u}$. From this ensemble we select the string $\bar{v}$ with the smallest $N_l(\bar{v}, v_{-k}^{-1})$. The selected $\bar{v}$ is the element in D-Ball$(\bar{u})$ with minimal first repetition.

3.2. Duality Lemma

We present the duality between the two random variables of Definitions 4 and 5. In what follows we will prove that this duality implies that between the proposed algorithms.

Lemma 2. *(Duality Lemma) For each sample sequence $\bar{u}$, for every database v_{-k}^{-1}, and for any positive integers $n \leq k$ and $l \leq n$,*

$$\left\{DN_l(\bar{u}, v_{-k}^{-1}) > n\right\} = \left\{DL_n(\bar{u}, v_{-n}^{-1}) \leq l\right\} \tag{6}$$

Proof. For given positive integers n, l,

$$\begin{aligned}\left\{DN_l(\bar{u}, v_{-k}^{-1}) > n\right\} &= \left\{\min_{\bar{v}:\rho(\bar{u},\bar{v})\leq D} N_l(\bar{v}, v_{-k}^{-1}) > n\right\} \\ &= \left\{\forall \bar{v} \quad \rho(\bar{u}, \bar{v}) \leq D, \quad N_l(\bar{v}) > n\right\}\end{aligned}$$

As in (Wyner and Ziv, 1989), for positive integers n, l,

$$\left\{N_l(\bar{v}, v_{-k}^{-1}) > n\right\} = \left\{v_0^{l-1} \neq v_{-m}^{l-1-m}, \ l \leq m \leq n\right\} = \left\{L_n(\bar{v}, v_{-n}^{-1}) \leq l\right\}$$

Thus

$$\begin{aligned}\left\{DN_l(\bar{u}, v_{-k}^{-1}) > n\right\} &= \left\{\forall \bar{v} \quad \rho(\bar{u}, \bar{v}) \leq D, \quad L_n(\bar{v}) \leq l\right\} \\ &= \left\{\max_{\bar{v}:\rho(\bar{u},\bar{v})\leq D} L_n(\bar{v}) \leq l\right\} = \left\{DL_n(\bar{u}, v_{-n}^{-1}) \leq l\right\}\end{aligned}$$

■

3.3. Approximate Fixed-Length String Matching Data Compression

The following scheme is actually based on two machines that work in parallel; one is an approximate string matching and the other is a blockcoder that generates *on-line* a Codebook. At the beginning most of the codewords are created via blockcoding, and after a sufficient period of time the approximate string matchings dominates the scheme. More and more approximate string matchings occur as the size of the database increases, and then the machine becomes sequential. As the Codebook is gradually constructed, the optimal acceptable partition is also created and updated as the empirical source distribution goes closer to the exact source distribution. In this algorithm we do not need to send the entire Codebook as in blockcoding. We transmit only the *necessary* codewords that comprise the database for string matching. In comparison with blockcoding this reduces the traffic in the channel thus rendering the algorithm feasible. Since the decoder adds a new string from either the database or the input to this database, its complexity is very low. A typical application is a multimedia system that requires image compression, where the transmitter can afford an expensive computer but the television sets must be cheap and therefore should be based on decoders of low complexity.

Let l be a fixed integer known for all terminals (the encoder and decoder). The sequence $u_0, u_1, \ldots, u_{l-1}$ is a new block. The data compression scheme is as follows. Assume that the encoder and decoder have the same database, $v_{-n} \ldots v_{-1}$, defined on the alphabet V which has been produced in previous steps. Without loss of generality, we assume that the initial database is empty.

We assume that the encoder has an algorithm that constructs *on-line* a codebook based on observations of a sufficiently long prefix of source symbols. Such a construction is the "bootstrap" stage of our algorithm. We do not require this huge Codebook to be transmitted to the decoder. We only require the encoder to transmit to the decoder only the necessary codewords and that these codewords comprise the database at both sides of the channel. Furthermore, the decoding procedures are very simple and can be very easily implemented. Recall that blockcoding requires the same computational power at both sides of the channel, or transmission of the huge Codebook to the decoder. We will show that asymptotically, as $l \to \infty$, the compression ratio of this algorithm converges to $R(D)$. The most appropriate encoding algorithm is that finding first the optimal acceptable partition in the sense of the minimal l-th order induced entropy on codewords, and next choosing the most probable codewords to Codebook (Sadeh, 1993a). In the following Theorem 1, we assume (Step 5) that the l-th order entropy induced on codewords is minimal among all the possible acceptable partitions. In the sequel, we describe a more practical algorithm based on (Ornstein and Shields, 1990). Blockcoding algorithms are discussed in (Ornstein and Shields, 1990; Sadeh, 1993b, 1995).

We apply a deterministic partition of the sourceword space. Such a partition, or a similar one as described in (Ornstein and Shields, 1990; Sadeh, 1993b, 1995), induces probabilities and the l-th order entropy on the process v, such that the Codebook set creates the best blockcoding. Optimality is considered with respect to the minimal error probability and/or the minimum rate. However, it is known (Ornstein and Shields, 1990; Sadeh, 1995) that the optimal blockcoder induces the l-th order entropy on the codewords such that it tends to $R(D)$ as $l \to \infty$. The probabilities are induced on the codewords by acceptable partitions. An *acceptable partition* of blocklength l is a partition on the space of sourcewords of length l such that for all $\bar{v}$ the associated subset $\boldsymbol{A}(\bar{v})$ satisfies $\boldsymbol{A}(\bar{v}) \subseteq \Upsilon(\bar{v})$ which is the D-Ball around $\bar{v}$, and that a limit of the induced l-th order entropy $\lim_{l\to\infty} H_v(l)$ exists.

The partition algorithm described and used here assumes a fixed l and a source with a known probability structure. The probabilities are obtained according to the empirical distribution of u in the past. As time elapses, the empirical distribution converges to the real one. Hence also the induced entropy given an empirical distribution converges to the entropy based on the real source distribution. The details are included in (Ornstein and Shields, 1990; Sadeh, 1993b, 1995) where it is proved that such an *acceptable partition* attains the minimum entropy $R(D)$ as the blocklength tends to infinity. In general, the blockcoding algorithm applies only to communications situations in which both the encoder and decoder exactly know the statistics of the source. Once the sorted lists on both sides are prepared, the transmitted information is the index of the selected codeword. However, universal coding means dropping this assumption and it implies performing the blockcoding only at the encoder size

and transmitting the selected codeword itself without compression.

For the sake of clarity, we repeat some basics concerning blockcoding. We describe one algorithm, but it is not crucial to use a specific one because we deal with the convergence issue. All the optimal acceptable partitions induce the entropy that converges to $R(D)$.

3.3.1. The Partition Algorithm

We define spheres around all the possible codewords $\bar{v}$,

$$\Upsilon(\bar{v}) = \left\{ \bar{u} \middle| \, \rho(\bar{u}, \bar{v}) \leq D \right\}$$

The coding is a deterministic process which maps each sourceword to exactly one codeword. We construct an algorithm that defines this mapping.

Denote by $\boldsymbol{A}(\bar{v})$ the set of sourcewords that are mapped to the codeword $\bar{v}$. Clearly, $\mathbf{A}(\bar{v}) \subseteq \Upsilon(\bar{v})$. In each step we create the most probable subset which is mutually exclusive with all the sets that are already selected. Here $\bar{v}^j$ denotes the j-th codeword in the list. The procedure is as follows:

Initialize:

1. Define $\Upsilon(\bar{v})$ for all the possible codewords.
2. Set $m = 1$.
3. Set $\boldsymbol{A}(\bar{v}) = \Upsilon(\bar{v})$ for all $\bar{v}$.

Loop:

4. Find an index k, $k > m$, which satisfies

$$\Pr\left(\boldsymbol{A}(\bar{v}^k)\right) = \max_{j \geq m} \Pr\left(\boldsymbol{A}(\bar{v}^j)\right)$$

5. Swap $\bar{v}^m$ for $\bar{v}^k$.
6. Update all the remaining subsets

$$\boldsymbol{A}(\bar{v}^j) = \boldsymbol{A}(\bar{v}^j) - \boldsymbol{A}(\bar{v}^j) \cap \boldsymbol{A}(\bar{v}^m), \qquad \forall j > m$$

7. Set $m = m + 1$.
8. If $m \leq |V|^l$, then goto Loop else stop.

After the algorithm has been executed, all the codewords are sorted such that to each codeword we assign the set of sourcewords mapped to it. The probability of each codeword is defined to be the probability of the set of sourcewords assigned to it. (Actually, the Codebook is constructed in accordance with the empirical source distribution.) In other words,

$$\Pr(\bar{v}^m) = \Pr\left(\boldsymbol{A}(\bar{v}^m)\right)$$

Then, the subsets $\boldsymbol{A}(\bar{v}^m)$ form a full partition of the set of all sourcewords. The following properties are valid:

$$\boldsymbol{A}(\bar{v}^j) \cap \boldsymbol{A}(\bar{v}^m) = \emptyset, \qquad \forall j \neq m$$

$$\Pr\left(\boldsymbol{A}(\bar{v}^j)\right) \geq \Pr\left(\boldsymbol{A}(\bar{v}^m)\right), \quad \forall j < m$$

$$\Pr(\bar{v}^j) \geq \Pr(\bar{v}^m), \qquad \forall j < m$$

The "Codebook" set $C_l(D, \delta)$ is defined as the set of all the codewords whose probability exceed a threshold p_t,

$$p_t = e^{-l(R(D)+\delta)}$$

The Codebook is

$$C_l = \left\{\bar{v} \middle| \Pr(\bar{v}) \geq p_t\right\}$$

and the cardinality of the set is $\mid C_l \mid = e^{lr}$. We define a function $\delta(r, l)$ such that $\delta = -\frac{\log p_t}{l} - R(D)$. Thus the Codebook is defined in terms of δ to consist of all the codewords such that

$$C_l(D, \delta) = \left\{\bar{v} \middle| \Pr(\bar{v}) \geq \exp\left(-l(R(D) + \delta)\right)\right\}$$

The value $\delta = \delta(l, r)$ is determined by r (the rate of the Code) such that $\left|C_l(D, \delta)\right| = e^{lr}$. Equivalently, we may say that the error event is the event that a D-Ball around a sourceword $\bar{u}$ does not contain any word from the selected codebook. In other words,

$$Er(\delta, D) = \left\{\bar{u} \middle| \max_{\bar{v}:\rho(\bar{u},\bar{v})\leq D} \Pr(\bar{v}) < p_t\right\}$$

Using the definition of p_t, we obtain

$$Er(\delta, D) = \left\{\bar{u} \middle| \min_{\bar{v}:\rho(\bar{u},\bar{v})\leq D} -\frac{1}{l}\log \Pr(\bar{v}) - R(D) > \delta\right\}$$

It is known (Ornstein and Shields, 1990; Sadeh, 1993b, 1995) that the error probability, denoted by $P_e(l, \delta, D) = \Pr\left\{Er\right\}$, decays to zero with l for all acceptable partitions that attain asymptotically the entropy $R(D)$ and for the rate that exceeds $R(D)$.

Now, we present compression schemes that consist of a bootstrapping stage of blockcoding and approximate string matching.

Data Compression Scheme A:

1. Verify readiness of the decoder.
2. Take a string $\bar{u} = u_0^{l-1}$ of length l.
3. If u_0^{l-1} can be approximately matched up to the tolerance D by a substring of v_{-n}^{-1}, encode it by specifying $DN_l(\bar{u}, v_{-n}^{-1})$ to the decoder. Add a bit as a header flag to indicate that there is a match. Append the string $v_{-DN_l}^{l-1-DN_l}$ to the database in the decoder and encoder at the position 0.
4. If not, indicate that there is no match, transmit to the decoder and append the string v_0^{l-1} to the database in the encoder and decoder. This is the associated D-Ball centre, obtained by blockcoding on the current string u_0^{l-1} and based on the accumulated empirical distribution in the past of u. The codeword is transmitted as it is, without compression.
5. Shift the indices by l to the appropriate values. Update n to $n+l$. Repeat the process from Step 1, with a new string of length l and a database v_{-n}^{-1}.

Decompression Scheme A:

1. Handle the coming string according to the "flag". If the flag indicates "Match", copy a substring of v_{-n}^{-1}, specified by the pointer $DN_l(\bar{u}, v_{-n}^{-1})$, to the decoder database. Append the string $v_{-DN_l}^{l-1-DN_l}$ to the database in the decoder at the position 0.
2. If the flag indicates that there is no match, append the string received in the input buffer at the beginning of the decoder database.
3. Shift the indices by l to the appropriate values. Update n to $n+l$. Repeat the process from Step 1, with a new input.

The scheme has two modes of operation. If there is an approximate matching, the database is augmented by a string and concatenated. Otherwise, there is a mode of "bootstrap" in which the machine appends an optimal blockcoded codeword based on the empirical distribution of u. Optimal blockcoding algorithms are discussed in (Ornstein and Shields, 1990; Sadeh, 1993b). One of them can be adopted. However, there is no compression during that mode of operation. The point is that the generated database entropy will not asymptotically exceed $R(D)$. After a sufficiently long period of time the probability of no approximate matching (No-Match event) decays to zero. When the database is sufficiently large, the encoder is Turing's machine that works almost surely in the mode of approximate string matching.

Scheme A is universal in the sense that the pre-knowledge of the source distribution is not essential. However, we assume that the machine has already worked for infinite time. Thus the empirical distribution in the source is the exact distribution. The decoder does not have to know anything about the source. The encoder performs an adaptive *on-line* learning stage when the accumulated empirical source distribution is learned, but it still works as an *on-line* machine. The encoder uses the empirical distribution of the source during the "bootstrap" mode. However, as time elapses, the generated database is sufficient. The decoder is very simple and it

does not require any prior knowledge, not even a Codebook. The common database process v is sufficient. The main advantage of the scheme is the simplicity of the decoding procedure: copy a string either from the input or from the database. The complexity lies only in the encoding procedure and, as time elapses, it tends also to be a Turing machine which performs the approximate string matching and pointing. We conjecture that it is possible to use a probabilistic method as described in (Alon and Spencer, 1992) to "guess" the best partition in a faster procedure. Such an algorithm would replace the deterministic partition algorithm such as (Ornstein and Shields, 1990) in Step 4. It would accelerate the "bootstrap" stage and would bring the method much closer to real systems.

Scheme A has several advantages over blockcoding algorithms (Ornstein and Shields, 1990; Sadeh, 1994). The first and most important one is that the preknowledge of the source distribution is not essential at the decoder side. The encoder can learn the source statistics as an *off-line* task, otherwise the convergence to $R(D)$ is slower. The encoder uses *on-line* the source empirical distribution only during the "bootstrap" mode and only for a period of time whose length depends on the nature of the source (Markovity or periodicity helps). The machine is asymptotically Turing's one. As time elapses, the created database is sufficient and the encoder tends to be a sequential machine that attains asymptotically the bound $R(D)$. The decoding procedure is a simple Turing machine: copy a string from either the source or the database. We conjecture that practically, after a short operating time, it is not necessary to compute in a complex way and to store a large Codebook in the encoder memory as in a blockcoding procedure. The approximated algorithm, presented later on and denoted by "Scheme $\widehat{A}$", can then work successfully.

It is reasonable to assume that the process $v_{-\infty}^{-1}$ is stationary and ergodic, as explained below. In the terminology of (Gray, 1990), the machine described in Scheme A is a deterministic channel, or a sequence coder, as defined in Gray (1990). When Scheme A starts, at the beginning the coder puts out l symbols from the alphabet V every time it takes in l symbols from the input alphabet U. Thus the output is cyclostationary with period l. However, as time elapses, roughly after $e^{lR(D)}$ time units, the machine scans and, with increasing frequency, finds matching in the created database. The matching occurs for almost all the strings with probability that increases to 1. The index of matching with respect to the block length l, i.e. on the interval $0 \ldots l-1$ is distributed uniformly, for symmetry reasons. Thus the blocking property disappears gradually and the output process becomes stationary after infinite time has elapsed.

Consider now the situation after Scheme A has been implemented for infinite time. Due to ergodicity of the source, all strings with positive measure have already occurred infinitely often. It follows that the probability of "No Match" is zero and almost surely all strings of length l in the output v are generated by an assignment from the process itself. The quantities v have well defined probabilities. Furthermore, the probabilities do not depend on the location of the strings in the output. Thus the string matching procedure is actually randomizing the index from where the assignment is being taken.

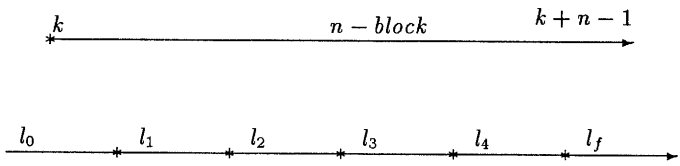

Fig. 1. Fixed length assignment.

To understand better the disappearance of cyclostationarity and the onset of stationarity, we try to explain the property of stationarity in a non-formal exposition. Consider a block of length n from the position k. We assume that processing v (the output of Scheme A) started at time $t_0 = -\infty$. It is clear that the number of assignments included in the block depends only on n and l, but not on k. Also, the knowledge of k provides no information for the determination of the matching points on the prefix of database. Once k and n are given, the matching indices relative to l (mod l) are random variables defined at the points of the interval $[0 \ldots l-1]$ and are uniformly distributed with pmf $q(prefix) = q(suffix) = 1/l$. Thus the expression $\Pr\{v_k^{k+n-1} = \bar{v}\}$ is independent of the initial position k for all n and for all $\bar{v} \in V^n$.

A final property to quantify the behaviour of the coding scheme is that of ergodicity. It is known that a stationary deterministic channel is ergodic (Gray, 1990). Thus the scheme is ergodic.

Theorem 1. (Limit Theorem A) *Given a D-semifaithful database* $v_{-\infty}^{-1}$ *generated by Scheme A from a stationary ergodic process* u *suppose that* $v_{-\infty}^{-1}$ *is a stationary and ergodic process. Then, for all* $\beta > 0$,

$$\lim_{l\to\infty} \Pr\left\{\left|\frac{\log DN_l(\bar{u}, v_{-\infty}^{-1})}{l} - R(D)\right| > \beta\right\} = 0 \tag{7}$$

and the average compression ratio attains the bound $R(D)$.

Proof. The proof consists of five steps. The first step discusses known results on blockcoding, which is in our algorithm a "bootstrap" procedure. Recall that blockcoding can obtain asymptotically the bound $R(D)$. The second step is an application of the AEP Theorem (Breiman, 1957) to the lossy case as presented in (Sadeh, 1995). Only at the third step we study the behaviour of the algorithm after a long period of time. In the third and fourth step, we show that the average compression ratio does not exceed the entropy rate of the generated database in probability. In the last step, we prove that the entropy rate of the database $v_{-\infty}^{-1}$ is $R(D)$. These results imply (7).

Step 1.

We have already applied a deterministic partition of the sourcewords space. Such a partition induces probabilities and the l-th order entropy on the process v such that the Codebook set creates the best blockcoding. We define spheres around all the possible codewords $\bar{v}$,

$$\Upsilon(\bar{v}) = \left\{\bar{u} \middle| \rho(\bar{u}, \bar{v}) \leq D\right\} \tag{8}$$

After the algorithm has been executed, all the codewords are sorted such that to each codeword we assign the set of sourcewords mapped to it. The probability of each codeword is defined to be the probability of the set of sourcewords assigned to it. (Actually, the Codebook is constructed in accordance with the empirical source distribution.) In other words,

$$\Pr(\bar{v}^m) = \Pr\left(\boldsymbol{A}(\bar{v}^m)\right) \tag{9}$$

Then the subsets $\boldsymbol{A}(\bar{v}^m)$ form a full partition of the set of all the sourcewords. The "Codebook" set $C_l(D,\delta)$ is defined as the set of all the codewords whose probabilities exceed a threshold p_t,

$$p_t = e^{-l(R(D)+\delta)} \tag{10}$$

The Codebook is

$$C_l = \left\{\bar{v}\middle|\, \Pr(\bar{v}) \geq p_t\right\} \tag{11}$$

and the cardinality of the set is $\mid C_l \mid = e^{lr}$. We define a function $\delta(r,l)$ such that $\delta = -\frac{\log p_t}{l} - R(D)$. Thus the Codebook is defined in terms of δ to consist of all the codewords such that

$$C_l(D,\delta) = \left\{\bar{v}\middle|\, \Pr(\bar{v}) \geq \exp\left(-l(R(D)+\delta)\right)\right\} \tag{12}$$

The value $\delta = \delta(l,r)$ is determined by r (the rate of the Code) such that $\left|C_l(D,\delta)\right| = e^{lr}$. Equivalently, we may say that the error event is the event that a D-Ball around the sourceword $\bar{u}$ does not contain any word from the selected codebook, i.e.

$$Er(\delta,D) = \left\{\bar{u}\middle|\, \max_{\bar{v}:\rho(\bar{u},\bar{v})\leq D} \Pr(\bar{v}) < p_t\right\} \tag{13}$$

Using the definition of p_t, we obtain

$$Er(\delta,D) = \left\{\bar{u}\middle|\, \min_{\bar{v}:\rho(\bar{u},\bar{v})\leq D} -\frac{1}{l}\log\Pr(\bar{v}) - R(D) > \delta\right\} \tag{14}$$

It is known (Ornstein and Shields, 1990; Sadeh, 1993b, 1994) that the error probability, denoted by $P_e(l,\delta,D) = \Pr\left\{Er\right\}$, decays to zero with l for all acceptable partitions that attain asymptotically the entropy $R(D)$ and for the rate that exceeds $R(D)$.

Step 2.

Now, we denote by R the entropy rate of the database process v as $l \to \infty$. It is clear that $R \geq R(D)$. We construct a set $\Gamma_l(D,\delta)$ for every l and a fixed $\delta > 0$.

We define the code that consists of strings of length l from the database by using δ as an independent variable:

$$\Gamma_l(D,\delta) = \left\{ \bar{v} \middle| \Pr(\bar{v}) \geq \exp\left(-l(R+\delta)\right) \right\} \tag{15}$$

The set $\Gamma_l(D,\delta)$ consists of all codewords of length l whose probabilities exceed a given threshold $e^{-l(R+\delta)}$ based on the database entropy rate. The new error set is

$$Error(l,\delta,D) = \left\{ \bar{u} \middle| \min_{\bar{v}:\rho(\bar{u},\bar{v})\leq D} -\frac{1}{l}\log \Pr(\bar{v}) - R > \delta \right\} \tag{16}$$

The set $Error$ consists of all sourcewords $\bar{u}$ for which all the elements in their D-Ball have probabilities below the threshold. By definition, the probability of the set $Error$ is the same as the measure of the set Γ^c. But it is known by the Shannon-McMillan-Breiman Theorem and their successors that the measure of that set tends to zero as the blocklength l tends to infinity. It turns out (c.f. the AEP Theorem) (Breiman, 1957) that

$$\lim_{l\to\infty} \Pr\left\{ Error \right\} = 0 \tag{17}$$

Clearly, we define a set $Error^c$ as the "good" sourceword set,

$$Error^c(\delta,l,D) = \left\{ \bar{u} \middle| \min_{\bar{v}:\rho(\bar{u},\bar{v})\leq D} -\frac{1}{l}\log \Pr(\bar{v}) - R \leq \delta \right\} \tag{18}$$

Step 3.

At this step, we start by considering the algorithm in Scheme A. For a fixed l, we have

$$\frac{\log DN_l(\bar{u}, v_{-\infty}^{-1})}{l} = \frac{\log \min_{\bar{v}:\rho(\bar{u},\bar{v})\leq D} N_l(\bar{v})}{l} = \min_{\bar{v}:\rho(\bar{u},\bar{v})\leq D} \frac{\log N_l(\bar{v})}{l} \tag{19}$$

For convenience we abbreviate the notation and write $N_l(\bar{v})$ instead of $N_l(\bar{v}, v_{-\infty}^{-1})$.

According to the stationarity assumption and its reasoning, we can consider the process v as stationary ergodic and we can apply the Extended Kac Lemma (1)–(4). We identify the set $B = D\text{-Ball}(\bar{u})$. We fix the blocklength as a specific value l. The Extended Kac Lemma yields, for all $\bar{u} \in U^l$,

$$E\left\{ \min_{\bar{v}:\rho(\bar{u},\bar{v})\leq D} N_l(\bar{v}) \right\} \Pr\left\{ D\text{-Ball}(\bar{u}) \right\} = 1 \tag{20}$$

Following the preceding discussion, the entropy of the database process generated by Scheme A attains asymptotically a bound $R \geq R(D)$. Hence each triplet (D,δ,l) defines a partition into two sets on the codeword space as in (15), and into two sets on the sourceword space as in (16). The stationarity of v guarantees the possibility of such partitions for every l. Thus, for all $\bar{u} \in Error^c(D,\delta,l)$, at least one codeword

in the selected code, $\bar{v} \in \Gamma$, is included in D-Ball($\bar{u}$). Thus, using the definition of the code in (15), for all $\bar{u} \in Error^c(D, \delta, l)$,

$$\Pr\Big\{D\text{-Ball}(\bar{u})\Big\} \geq \exp\Big(-l(R+\delta)\Big)$$

Since the database is an asymptotically stationary ergodic process which is observed after a long period of time, we obtain by (20), for all l and for all $\bar{u} \in Error^c(D, \delta, l)$,

$$E\Big\{\min_{\bar{v}:\rho(\bar{u},\bar{v})\leq D} N_l(\bar{v})\Big\} \leq \exp\Big(l(R+\delta)\Big) \tag{21}$$

The expected value given the set $\bar{u} \in Error^c(D, \delta, l)$ is bounded by the same expression, i.e.

$$E\Big\{\min_{\bar{v}:\rho(\bar{u},\bar{v})\leq D} N_l(\bar{v})\Big| \quad \bar{u} \in Error^c(D, \delta, l)\Big\} \leq \exp\Big(l(R+\delta)\Big) \tag{22}$$

Next, we pick the current string $\bar{u}$ of length l from the input process u. We abbreviate the notation $N_l(\bar{v}, v_{-\infty}^{-1})$ as $N_l(\bar{v})$. Recall Markov's inequality that states: for every positive random variable Y, $\Pr\{Y \geq a\} \leq E(Y)/a$. Using Markov's inequality, (19) and (22) yield, for $\beta > 0$,

$$\begin{aligned}
&\Pr\Big\{\min_{\bar{v}:\rho(\bar{u},\bar{v})\leq D} \frac{\log N_l(\bar{v})}{l} \geq R+\beta\Big| \quad \bar{u} \in Error^c(\delta, l, D)\Big\} \\
&\quad = \Pr\Big\{\min_{\bar{v}:\rho(\bar{u},\bar{v})\leq D} N_l(\bar{v}) \geq \exp\big(l(R+\beta)\big)\Big| \quad \bar{u} \in Error^c(D, \delta, l)\Big\} \\
&\quad \leq \frac{E\Big\{\min_{\bar{v}:\rho(\bar{u},\bar{v})\leq D} N_l(\bar{v})\Big| \quad \bar{u} \in Error^c(D, \delta, l)\Big\}}{\exp\big(l(R+\beta)\big)} \leq e^{-l(\beta-\delta)}
\end{aligned} \tag{23}$$

We now write, for a fixed blocklength l and $\delta > 0$,

$$\begin{aligned}
&\Pr\Big\{\min_{\bar{v}:\rho(\bar{u},\bar{v})\leq D} \frac{\log N_l(\bar{v})}{l} \geq R+\beta\Big\} \\
&\quad = \Pr\Big\{\bar{u} \in Error^c(D, \delta, l)\Big\} \Pr\Big\{\min_{\bar{v}:\rho(\bar{u},\bar{v})\leq D} \frac{\log N_l(\bar{v})}{l} \geq R+\beta\Big|\bar{u} \in Error^c\Big\} \\
&\quad +\Pr\Big\{\bar{u} \in Error(\delta, l)\Big\} \Pr\Big\{\min_{\bar{v}:\rho(\bar{u},\bar{v})\leq D} \frac{\log N_l(\bar{v})}{l} \geq R+\beta\Big| \quad \bar{u} \in Error\Big\} \\
&\quad \leq \Pr\Big\{\min_{\bar{v}:\rho(\bar{u},\bar{v})\leq D} \frac{\log N_l(\bar{v})}{l} \geq R+\beta\Big|\bar{u} \in Error^c(\delta, D, l)\Big\} + \Pr\Big\{Error(\delta, l)\Big\}
\end{aligned}$$

Using (23), we get

$$\Pr\Big\{\min_{\bar{v}:\rho(\bar{u},\bar{v})\leq D} \frac{\log N_l(\bar{v})}{l} \geq R+\beta\Big\} \leq e^{-l(\beta-\delta)} + P_e(l, \delta, D) \tag{24}$$

We know that the error probability is equal to the probability of the set

$$\Gamma_l{}^c(l, D, \delta) = \left\{\bar{v} \middle| \Pr(\bar{v}) < \exp(-l(R+\delta))\right\}$$

Following the asymptotic equipartition property (AEP) of (Breiman, 1957) and (Algoet and Cover, 1988), as well as the fact that the database entropy rate is R and $n > e^{-l(R+\delta)}$, we can deduce that this set has the measure that tends to zero as $l \to \infty$. Thus setting $\delta = \beta/2$ yields, for all $\beta > 0$,

$$\lim_{l\to\infty} \Pr\left\{ \min_{\bar{v}:\rho(\bar{u},\bar{v})\le D} \frac{\log N_l(\bar{v})}{l} \ge R+\beta \right\} = 0 \tag{25}$$

which guarantees, as shown in the next step, that the average compression ratio does not exceed in probability the entropy rate of the generated database.

Step 4.

The number of bits required to encode the l–length string $\bar{u}$ by specifying the pointer $DN_l(\bar{u}, v_{-\infty}^{-1})$ is $\log(DN_l(\bar{u}, v_{-\infty}^{-1})) + O(\log\log(DN_l(\bar{u})))$ as proved by Elias (1975) for the pointer. Following (25), we obtain

$$\lim_{l\to\infty} \Pr\left\{\text{Compression Ratio} \ge R+\beta\right\} = 0$$

On the other hand, if $\lim_{l\to\infty} \Pr\left\{\text{Compression Ratio} \le R(D)-\beta\right\} > 0$ for positive β, then it is possible to compress the input more than the bound $R(D)$ in contradiction to the definition of $R(D)$ as a minimum. Thus Scheme A attains a compression ratio in probability in the range R–$R(D)$. Still we have to show that $R = R(D)$.

Step 5.

Now, we justify that the entropy rate of the stationary database process $v_{-\infty}^{-1}$ is $R(D)$ as $l \to \infty$. Suppose that after n input symbols the l-length block-coder attains the l-th order entropy $R_{n,l}(D, u_{-n}^{-1})$ associated to the codewords. We denote by R_n the n-th order entropy of the database. The entropy rate of the database as $l \to \infty$ is denoted by R.

We assume that $R > R(D)$. Thus we choose $\delta > 0$ and $\epsilon > 0$ satisfying $R(D) < R(D) + \delta = R - \epsilon < R$. Define a pair (l, n) such that $n = e^{l(R(D)+\delta)} = e^{l(R-\epsilon)}$. We denote by $p_{n,l}$ the probability of the event *Approximate Match* of string of length l in a stationary database of length n. Since we assume stationarity, the probability of finding the first matching is less than n steps in a long database. This quantity $p_{n,l}$ is approximately the fraction of strings in the stationary part of the database of length n generated by string matching mode in Scheme A. Following (3), (4) we can describe roughly the asymptotic probability of the matching event as follows

$$\begin{aligned}
\lim_{l\to\infty} p_{n,l} &= \lim_{l\to\infty} \Pr\left\{\bar{u} | E_v \min_{\bar{v}:\rho(\bar{u},\bar{v})\le D} N_l(\bar{v}, v_{-\infty}^{-1}) \le n\right\} \\
&= \lim_{l\to\infty} \Pr\left\{\bar{u} | \max_{\bar{v}:\rho(\bar{u},\bar{v})\le D} \Pr(\bar{v}) \ge \frac{1}{n}\right\} = \lim_{l\to\infty} \Pr\left\{\bar{v} | \Pr(\bar{v}) \ge \frac{1}{n}\right\} \\
&= \lim_{l\to\infty} \Pr\left\{\bar{v} | \Pr(\bar{v}) \ge e^{-l(R-\epsilon)}\right\} \le \epsilon
\end{aligned}$$

These equalities are based on the definition of the Match event and AEP (Breiman, 1957). This implies that, as $l \to \infty$, the database is constructed by optimal blockcoding with probability of at least $1 - \epsilon$.

Next, we consider the limit as $l \to \infty$ of the block-coder output $\lim_{l\to\infty} R_{n,l}(D, u_{-n}^{-1})$. The induced probability distribution on the codeword space, based on the knowledge of the empirical distribution obtained by the prefix of length n of the input process u, is denoted by M_n. Suppose that M is the stationary distribution of the process v that asymptotically dominates M_n. (For example, M is the stationary mean of M_n.)

We denote by $R_l(D)$ the l-th order entropy induced on the codewords based on the exact knowledge of the source distribution, e.g. an infinitely long prefix of u. This function is dominated by the distribution M. Consider the probability of deviation of $R_{n,l}(D, u_{-n}^{-1})$ from $R_l(D)$. In other words, by using Markov's inequality,

$$M_n\Big\{ -\frac{1}{l}\log M_n(v_0^{l-1}) \geq -\frac{1}{l}\log M(v_0^{l-1}) + \delta \Big\}$$

$$= M_n\Big\{ \frac{M(v_0^{l-1})}{M_n(v_0^{l-1})} \geq e^{l\delta} \Big\} \leq e^{-l\delta} E_{M_n}\Big\{ \frac{M(v_0^{l-1})}{M_n(v_0^{l-1})} \Big\} = e^{-l\delta}$$

Using the assumption on $n = e^{l(R(D)+\delta)}$, we obtain that

$$\lim_{l\to\infty} M_n\Big\{ R_{n,l}(D, u_{-n}^{-1}) > R(D) \Big\} = 0$$

The entropy of the long database is composed mainly by strings of length l obtained by blockcoding. Thus we can bound the entropy rate. The main result of Step 5 is that

$$R = \lim_{l\to\infty} R_n \leq \lim_{l\to\infty} \lim_{\epsilon\to 0} \Big\{ \frac{l}{n} \sum_{k=1}^{n/l} R_{kl,l}(D, u_{-kl}^{-1}) + \epsilon \max_{N(n,l)} \min_{\bar{v}:\rho(\bar{u},\bar{v})\leq D} \frac{\log N_l(\bar{v}, v_{-N(n,l)}^{-1})}{l} \Big\}$$

where $N(n,l) < n$ is some index on the database. Following (Gray, 1990) and known results from the calculus, we conclude that

$$R = \lim_{l\to\infty} R_n = R(D) \tag{26}$$

After infinite time, we have

$$\lim_{l\to\infty} \Big\{ \text{Output Entropy} \Big\} = R(D)$$

After a sufficiently long period of time the "bootstrap" stage has been completed and the database process tends to be an ergodic and stationary one with entropy $R(D)$. All rare strings that have some positive probability have already occurred and the probability of No-Match is zero. Then, the compression ratio is determined only by

the approximate string matching mode according to the database entropy. Thus (25) and (26) yield in probability

$$\lim_{l\to\infty}\left\{\text{Compression Ratio}\right\} = R(D)$$

■

The schemes are deterministic assignment processes that convert the stationary and ergodic source U into an output process V. On the assumption that V is a stationary process, after the scheme has run for infinite time, it is shown that the optimal compression ratio $R(D)$ is achieved. This reduces the problem of the universal lossy coder to the proof of stationarity of the output process V in the proposed algorithm.

The missing link in the above theory is the proof of the assumption about the stationarity of the output process of Scheme A. Such a proof will solve, in effect, the universal lossy coding problem by means of compression of Scheme A. This is still an open problem, but it is now reduced to a technical (though by no means easy) proof of stationarity.

Since blockcoding is not practical in reality, we propose the following suboptimal scheme. It is not guaranteed that the bound will be attained, but it is more practical. We believe that in practical cases, after a short operating time, this scheme may successfully replace Scheme A.

Data Compression Scheme $\widehat{A}$:

1. Verify the readiness of the encoder.
2. Take a string $\bar{u} = u_0^{l-1}$ of length l.
3. If u_0^{l-1} can be approximately matched up to tolerance D by a substring of v_{-n}^{-1}, then encode and transmit it by specifying $DN_l(\bar{u}, v_{-n}^{-1})$. Add a bit as a header flag to indicate that there is a match. Append the string $v_{-DN_l}^{l-1-DN_l}$ to the database at the position 0.
4. If not, indicate that there is no matching, transmit and append the string v_0^{l-1}, which satisfies $\rho(u_0^{l-1}, v_0^{l-1}) = 0$, to the database in the encoder and the decoder.
5. Update $n = n + l$. Shift the indices by l to the appropriate values. Repeat the process from Step 1 with a new string of length l and the database v_{-n}^{-1}.

The scheme has two modes of operation. If there is an approximate matching, the database is augmented by a string and concatenated. Otherwise, the mode of "bootstrap" appends a zero distorted image of the input string. After a sufficiently long period of time, the probability of no approximate matching (No-Match event) decays to zero. When the database is sufficiently large, we have a Turing machine with infinite memory, that performs an approximate string matching only.

3.4. Quasi Parsing Data Compression

We present an extension of the Lempel-Ziv compression scheme. The machine is based on a quasi parsing procedure which creates a new phrase as soon as a prefix of

the still unparsed part of the input sequence u differs from all the substrings in the database v by more than D per letter. Encoding each input phrase consists of the pointer N to the last approximately matched string, the string length DL_n and the last reproducing letter with zero distance from the last input symbol.

Scheme B:

1. Set $l = 1$.
2. Take the string u_0^{l-1}.
3. If u_0^{l-1} can be approximately matched up to tolerance D by a substring of v_{-n}^{-1}, then store the pointer N to that substring and increment l. Go to Step 2.
4. If not, append to the data base track the string v_{-N}^{l-2-N} at the position zero and further, and append the letter v_{l-1}, i.e. the reproducing letter which satisfies $d(u_{l-1}, v_{l-1}) = 0$. The encoding is done by the pointer to the string v_{-N}^{l-2-N}, the length $DL_n(u)$ and the last reproducing letter associated to the last source letter.
5. Repeat the process from Step 1, where the database is appended with the chosen string denoted by $v_0^{DL_n}$. Shift the indices to adapt to the algorithm. The database contains now $n + DL_n(u)$ reproducing symbols.

This scheme has only one mode of operation and it does not require a block-coder which is hard to implement. Thus it is more attractive than Scheme A. The entire processing is repeated each time with the database augmented with the new generated block $v_0^{DL_n}$ which is also known to the decoder. Note that the scheme is universal in that the encoder does not have to know or to compute the source statistics to perform its task. However, it depends on the initial database. There is still an open question concerning the convergence limit of compression ratio in the general case of an arbitrary initial database. In practice, the case of no initial database is the most important one.

Theorem 2. (Limit Theorem B) *Let u be a stationary ergodic process defined on an alphabet U. Assume that there is a suffix v_{-n}^{-1} taken from the database $v_{-\infty}^{-1}$ generated by an encoder-decoder pair as described in Scheme A. At time zero we switch to Scheme B. If we assume that the scheme preserves stationarity, then for the new sample sequence $\bar{u}$ encoded from the input u by Scheme B, we have in probability*

$$\lim_{n\to\infty} \left\{ \frac{\log n}{DL_n(\bar{u}, v_{-n}^{-1})} \right\} = R(D) \tag{27}$$

Proof. We have

$$\lim_{n\to\infty} \Pr\left\{ \frac{\log n}{DL_n(\bar{u}, v_{-n}^{-1})} \geq R(D) + \delta \right\} = \lim_{n\to\infty} \Pr\left\{ \frac{\log n}{R(D) + \delta} \geq DL_n(\bar{u}, v_{-n}^{-1}) \right\}$$

We define an integer l such that the pair l, n satisfies

$$l = \frac{\log n}{R(D) + \delta}$$

and apply Lemma 2 and Theorem 1, obtaining

$$\lim_{n\to\infty} \Pr\left\{\frac{\log n}{R(D)+\delta} \geq DL_n(\bar{u}, v_{-n}^{-1})\right\} = \lim_{n\to\infty} \Pr\left\{DN_l(\bar{u}, v_{-\infty}^{-1}) > n\right\}$$

$$= \lim_{n\to\infty} \Pr\left\{\frac{\log DN_l(\bar{u}, v_{-\infty}^{-1})}{l} > \frac{\log n}{l}\right\}$$

$$= \lim_{l\to\infty} \Pr\left\{\frac{\log DN_l(\bar{u}, v_{-\infty}^{-1})}{l} > R(D)+\delta\right\} = 0$$

■

Theorem 3. *Let us consider a maximum average distortion $D \geq 0$ and the database $v_{-\infty}^{-1}$ generated by Scheme A. Assuming switching to Scheme B at time zero, the quasi-parsing data compression scheme attains the bound R(D) in probability.*

Proof. The construction procedure of the process v is done in a recursive way. We consider the first iteration. A new block is appended to the output database track. The block is exactly defined by the pointer to the interval $\Big(-1, -(n - DL_n)\Big)$, the length DL_n and the last reproducing letter. The pointer requires $\log n$ bits, the length DL_n can be encoded (Rodeh *et al.*, 1981) by $\log DL_n + O(\log\log DL_n)$ bits (when the length is large enough) and the last symbol requires at most $\log(|V|-1)$ bits (since one letter can be excluded). Thus the average number of bits required to encode the string of length DL_n in the scheme is

$$\log(n - DL_n) + \log DL_n + \log(|V|-1) + O(\log\log DL_n)$$

Therefore the number of bits per source symbol with compression is

$$\frac{\log(n - DL_n) + \log DL_n + \log(|V|-1) + O(\log\log DL_n)}{DL_n}$$
$$\approx \frac{\log(n) + O(\log DL_n(\bar{u}))}{DL_n(\bar{u})} \tag{28}$$

As the database size tends to infinity, we obtain the result of possible compressibility from (27). Thus, for large n, in probability

$$\lim_{n\to\infty} \frac{\log(n) + O(\log DL_n(\bar{u}, v_{-n}^{-1}))}{DL_n(\bar{u}, v_{-n}^{-1})} = R(D)$$

The algorithm proceeds with a database with entropy $R(D)$. Thus the compression ratio attains the bound in probability. ■

For practical data compression the dimension of the database must be determined since the memory is limited. After a while, the database size is held fixed. This constraint imposes that we are close to $R(D)$ but cannot achieve it. We are interested in finding the convergence rate of the accepted rate to $R(D)$ as a function of the database length n.

Theorem 4. (Convergence Rate Theorem) *For a sufficiently large database generated by Scheme A with size $n \gg 1$, the number of encoded bits per source symbol in Scheme B is approximately*

$$R(D) + O\left(\frac{\log\log n}{\log n}\right) \tag{29}$$

Proof. Due to reasoning similar to (28), the number of bits per source symbol is, in probability,

$$\begin{aligned} &\frac{\log(n - DL_n) + \log DL_n + \log(|V| - 1) + O(\log\log DL_n)}{DL_n} \\ &\approx \frac{\log(n) + O(\log DL_n(\bar{u}))}{DL_n(\bar{u})} \end{aligned} \tag{30}$$

As the database size increases, then, following (27),

$$\frac{\log(n) + O(\log DL_n(\bar{u}, Q))}{DL_n(\bar{u}, Q)} \approx R(D) + O\left(\frac{\log\log n}{\log n}\right)$$

■

The practical meaning of Theorem 4 is that a large initial database is required to implement the compression ratio close to $R(D)$. To handle practical design problems it is not sufficient to know that the performance of the algorithm converges to the optimum; one should also know something about the finite database performance of such a machine. Based on (29), we obtain that in order to attain in probability an accuracy of 0.01 from the limit, one needs a database of approximately 2^{1000} symbols. To guarantee in probability a compression ratio within the range of 0.1 from the limit, one needs approximately 2^{64} symbols in the database. Therefore any possibility of accelerating the convergence is important for practical applications. However, the practical experience with the Lempel-Ziv algorithm shows a better convergence rate. Thus we may hope for better convergence rates than the theoretical results. Special cases where there is prior information and/or there is a Markov property and/or some kind of periodicity will certainly reduce the size of the initial database and accelerate the convergence to $R(D)$.

The experience with asymptotic $R(D)$ algorithms is that, with finite data like images, they perform rather far from this bound. Pearlman (1994) reports that his theoretical work with stationary sources shows that, with source strings of finite length, the $R(D)$ bound can be approached more closely if the images are first decomposed into subbands (Padmanabha Rao and Pearlman, 1991). Then the approximate string matching procedures will be more efficient, given a proper bit allocation. This also brings the hot topic of wavelets into the mix.

4. Conclusions

In this paper, we obtain theorems concerning approximate string matching algorithms and their relation to the rate distortion function $R(D)$ of a stationary ergodic source. We use these results to gain insight into our new data-compression schemes. Moreover, these results give a new and conceptually simple way of estimating the rate distortion function of an ergodic stationary source with arbitrarily chosen D. The basis for our theory is the extension of the Kac Lemma to an approximate string matching of a stationary source. We have also proved an extension to a non-ergodic case. We use the results for the stationary case for proving the convergence of the algorithms.

We propose two practical source coding schemes based on approximate string matching. One is an approximate fixed-length string matching data compression, and the other is a LZ-type quasi parsing method by approximate string matching. It is shown in this paper that in the former algorithm the compression rate after infinite time of operation converges to the theoretical bound of $R(D)$ for ergodic and stationary processes as the average string length tends to infinity. The main advantages of this algorithm are the asymptotic complexity of the encoder and that it does not necessitate knowing the exact source distribution at the decoder side. Moreover, the decoder scheme is a very simple machine. Thus it provides a simple solution for cheap receivers, whereas the transmitters can afford the costly computation required for the coding. However, after a period of time when the encoder works as a blockcoder based on the *empirical distribution* it tends to be a sequential Turing machine linear with database size (which is exponential with typical blocklength). However, it is an *on-line* implementable machine. The Wiener-Ziv algorithm (1989) is obtained as a special case for no-distortion. We propose also a sub-optimal practical scheme without using blockcoding and with better complexity characteristics, polynomial with database from the beginning.

A similar result holds for the latter algorithm in the limit to infinity of an appropriate initial database size taken as a suffix of the database generated by the former algorithm after infinite time. We evaluate the performance of the algorithm in practical cases when the database size is finite. In addition to that, we prove some asymptotic properties concerning the performance of an approximate string matching algorithms for ergodic stationary sources.

Our algorithms are thought to be the first universal asymptotically sequential algorithms that attain the bound $R(D)$. But we emphasize that all the results are obtained after infinite time of operation and based on the accumulated data. It is still better than blockcoding because we send only the necessary messages and not all the pre-calculated codebook. Weak points are the slow convergence to $R(D)$, the initial complexity of the encoding procedure and the requirement for a good and long initial database. However, we believe the rate of convergence is faster in practical cases. In real life there exist properties such as Markovity, periodicity and/or some *a priori* information about the signal. In such cases the algorithm is paving the way for practical solutions. The most important issues are the asymptotical complexity of the encoder and that the decoder in both algorithms is very simple. It is based on copying strings from either a specified location in the database or the input data. In

practice, the low complexity of the decoder is an important advantage. It reduces the overall price of communications systems. The transmitter can be implemented in a costly way, but the receivers must be cheap.

Other important properties of the algorithms are as follows:

1. Optimality is obtained for a general stationary ergodic source.
2. Optimality is obtained for all memoryless distortion measures.
3. Easy adaptation to MultiMedia applications.
4. The Lempel Ziv algorithm (CCITT Standard) is recovered as $D = 0$.
5. Realization with relatively low complexity and implementation with a dynamic dictionary.
6. An appropriate definition of the distortion measure enables us to reduce the information content of a video/voice record while keeping a minimal visual/audio distortion.
7. Sub-optimal tree-structure algorithms are proposed.
8. Noise reduction.

The most common methods for lossy compression are based on predictive coding (DPCM), transforms (e.g. DFT, DCT, DST, WHT, wavelet, Haar), and vector quantization, with possible hybrid combinations. Most of them lack any general proof of optimality. Most of them use some kind of MSE criteria or data compression is achieved by coding only "high energy" coefficients in their transform. Moreover, almost all of these techniques are not real-time algorithms. All transform coding methods are performed after the signal (image/speech record) is received and only then transformed and processesed. Thus it is not real-time computation. Our algorithms achieve almost optimal compression performance with tolerable resources. Another important property is that our algorithms decompress much faster than they compress.

Acknowledgement

The author wishes to express his gratitude to Prof. Zeev Schuss, Prof. William Pearlman and Prof. Benjamin Weiss for many helpful suggestions during the preparation of the paper.

References

Algoet P.H. and Cover T.M. (1988): *A sandwitch proof of the Shannon McMillan Breiman theorem.* — The Annals of Probability, v.16, No.2, pp.899–909.

Alon N. and Spencer J. (1992): *The Probabilistic Method.* — New York: John Wiley.

Arratia R., Gordon L. and Waterman M. (1990): *The Erdos Renyi law in distribution for coin tossing and sequence matching.* — The Annals of Statistics, v.18, pp.539–570.

Arratia R. and Waterman M. (1989): *The Erdos Renyi strong law for pattern matching with given proportion of mismatches.* — The Annals of Probability, v.17, pp.1152–1169.

Berger T. (1971): *Rate Distortion Theory: A Mathematical Basis for Data Compression.* — New York: Prentice-Hall.

Blahut R.E. (1987): *Principles and Practice of Information Theory.* — New York: Addison-Wesley Publishing Co.

Breiman L. (1957): *The individual ergodic theorem of information theory.* — Annals on Math. Stat., v.28, pp.809–811.

Elias P. (1975): *Universal codeword sets and representations of the integers.* — IEEE Trans. Inform. Th., v.IT–21, pp.194–203.

Galil Z. and Giancarlo R. (1988): *Data structures and algorithms for approximate string matching.* — J. Complexity, v.4, pp.33–72.

Gilbert E. and Kadota T. (1992): *The Lempel Ziv algorithm and the message complexity.* — IEEE Trans. Inform. Th., v.IT–38, pp.1839–1842.

Gray R.M. (1975): *Sliding block source coding.* — IEEE Trans. Inform. Th., v.IT–21, pp.357–368.

Gray R.M. (1990): *Entropy and Information Theory.* — New York: Springer Verlag.

Jacquet P. and Szpankowski W. (1995): *Asymptotic behaviour of the Lempel Ziv parsing scheme and digital search trees.* — Theoretical Computer Science.

Kac M. (1947): *On the notion of recurrence in discrete stochastic processes .* — Bull. American Math. Society, v.53, pp.1002–1010.

Knuth D.E. (1981): *The Art of Computer Science. Seminumerical Algorithms.* — New York: Addison-Wesley, v.2.

Landau G.M. and Vishkin U. (1986): *Introducing efficient parallelism into approximate string matching and a new serial algorithm.* — Proc. 18 Annual ACM Symp. *Theory of Computing*, Berkeley.

Le Gall D. (1991): *MPEG a video compression standard for multimedia applications.* — Communications of the ACM, v.34, pp.46–58.

Lempel A. and Ziv J. (1977): *A universal algorithm for sequential data compression.* — IEEE Trans. Inform. Th., v.IT–23, pp.337–343.

Lempel A. and Ziv J. (1978): *Compression of individual sequences via variable-rate coding.* — IEEE Trans. Inform. Th., v.IT–24, pp.530–536.

Luczak T. and Szpankowski W. (1995): *A lossy data compression based on an approximate pattern matching.* — Submitted for publication.

Manber U. and Myers E.W. (1993): *Suffix trees: a new method for on-line string searchers.* — SIAM J. Computing, v.22, No.5, pp.935–948.

McCreight E.M. (1976): *A space economical suffix tree construction algorithm.* — J. ACM, v.32, No.2, pp.262–272.

Ornstein D.S. and Shields P.C. (1990): *Universal almost sure data compression.* — The Annals of Probability, v.18, pp.441–452.

Ornstein D.S. and Weiss B. (1993): *Entropy and Data Compression Schemes.* — IEEE Trans. Inform. Th., v.39, No.1.

Pittel B. (1985): *Asymptotic growth of a class of random trees.* — The Annals of Probability, v.13, pp.414–427.

Plotnik E., Weinberger M. and Ziv J. (1992): *Upper bounds on the probability of sequences emitted by finite state sources and on the redundancy of the Lempel Ziv algorithm.* — IEEE Trans. Inform. Th., v.38, pp.66–72.

Rodeh M., Pratt V. and Even S. (1981): *Linear algorithm for data compression via string matching.* — J. ACM, v.28, pp.16–24.

Padmanabha Rao R. and Pearlman W.A. (1991): *On entropy of pyramid structures.* — IEEE Trans. Inform. Th., v.37, No.2.

Sadeh I. (1993a): *On approximate string matching.* — Proc. *Data Compression Conference DCC'93*, Snowbird, Utah, USA.

Sadeh I. (1993b): *Data Compression in Computer Networks.* — Ph.D. Dissertation, Tel-Aviv University, Israel.

Sadeh I. (1994): *Data compression as a partition and covering problem.* — Submitted for publication.

Sadeh I. (1995): *Operational rate distortion theory.* — Appl. Math. and Comp. Science, v.5, No.1, pp.139–169.

Szpankowski W. (1993): *A generalized suffix tree and its unexpected asymptotic behaviours.* — SIAM J. Computing, v.22, pp.1176–1198.

Steinberg Y. and Gutman M. (1993): *An algorithm for source coding subject to a fidelity criterion, based on string matching.* — IEEE Trans. Inform. Th., v.39, pp.877–887.

Storer J.A. (1990): *Lossy On Line Dynamic Data Compression.* — Berlin: Springer-Verlag.

Welch T.A. (1984): *A technique for high performance data compression.* — IEEE Trans. Computers, v.17, No.6, pp.8–19.

Wyner A.D. and Ziv J. (1989): *Some asymptotic properties of the entropy of a stationary ergodic data source with applications to data compression.* — IEEE Trans. Inform. Th., v.35, pp.1250–1258.

Ziv J. (1978): *Coding theorem for individual sequences.* — IEEE Trans. Inform. Th., v.IT–24, pp.405–412.

Received: January 9, 1995