A TYPICAL BEHAVIOR OF SOME DATA COMPRESSION SCHEMES

Extended Abstract

Wojciech Szpankowski*
Department of Computer Science
Purdue University
W. Lafayette, IN 47907
U.S.A.

Abstract

Recently, Wyner and Ziv have proved that the typical length of a repeated subword found within the first $n$ positions of a stationary sequence is $(1/h) \log n$ in probability where $h$ is the entropy of the alphabet. Wyner and Ziv used this finding to obtain several insights into certain universal data compression schemes, most notably Lempel-Ziv data compression algorithm. In addition, these authors have also conjectured that their result can be extended to a stronger almost sure convergence. In this paper, we settle this conjecture in the negative. We prove, under some additional assumption regarding mixing conditions, that the length of a repeated subword oscillates with probability one between $(1/h_1) \log n$ and $(1/h_2) \log n$ where $0 < h_2 < h \leq h_1 < \infty$.

1. INTRODUCTION

Periodicities, repeated patterns, and related phenomena in words (sequences, strings) are known to play a central role in many facets of telecommunications and theoretical computer science, notably in coding theory and data compression, in the theory of formal languages, and in the design and analysis of algorithms. Several efficient algorithms have been designed to detect and to exploit the presence of repeated substrings and other kinds of avoidable or unavoidable regularities in words. Many of these algorithms, most notably in data compressions and algorithms on words, crucially depend on a solution to the following problem: given a sequence $X$, and two arbitrary suffixes $S_1$ and $S_2$ of $X$, what is the longest common prefix of $S_1$ and $S_2$?

This issue falls into a general class of problems investigating properties of repeated subwords of a word. In data compression such a repeated subsequence can be used to compress the original sequence and send over a channel only the encoded (compressed) version of the sequence (cf. universal data compression proposed by Lempel and Ziv [24], [14], [25]). In exact string matching algorithms (cf. Knuth-Morris-Pratt [13] and Boyer-Moore [14]), the largest suffix that matches a substring of the pattern string is used for "fast" shift of the pattern over a text string.

The above problem is studied here in a probabilistic framework. We assume that a stationary and ergodic information source generates an infinite sequence over a finite alphabet.
alphabet \( A \). The usefulness of our probabilistic model is illustrated on the following examples taken from data compression and algorithms on words. These examples are also used to motivate our further study.

EXAMPLE 1.1 Data Compression: Algorithm 1

Consider the following universal compression scheme discussed in Wyner and Ziv [23]. Let \( \{X_i\}_{i=-\infty}^\infty \) be a stationary ergodic sequence built from symbols of a finite alphabet \( A \). For every integer \( \ell \) define the random variable \( N_\ell \) as the smallest \( N > 0 \) such that \( (X_0, X_1, \ldots, X_{\ell+1}) = (X_{-N}, X_{-N+1}, \ldots, X_{-N+\ell-1}) \), that is, a word of length \( \ell \) has appeared before \( N \) positions backward. Then, consider the following universal compression scheme discussed in [23]. The encoder sends the first \( n \) source symbols, say \( X_1^{n-1} \) with no compression, but the next \( \ell \) symbols \( X_{n-1}^n \) are encoded as follows: if \( X_{n-1}^n \) is a substring of \( X_1^n \) (i.e., \( N_\ell \leq n \)) then \( X_{n-1}^n \) is compressed by specifying only \( N_\ell \); otherwise \( X_{n-1}^n \) is not compressed. As noted by Wyner and Ziv [23] in this scheme the average number of symbols required to encode \( X_{n-1}^n \) is

\[
Pr\{N_\ell \leq n\} \log_2 |A| n + Pr\{N_\ell > n\}\ell + 1,
\]

where \( \log_2 |A| n \) is the number of symbols required to transmit \( N_\ell \). The efficiency of such a scheme can be measured by the average number of transmitted symbols per source symbol, and in our case it depends on the ratio \( \log_2 |A| n / \ell \).

EXAMPLE 1.2 Data Compression: Algorithm 2

Another compression algorithm can be constructed as follows (cf. Lampel and Ziv [14], and Wyner and Ziv [23]). For every \( n \) define \( L_n \) as the smallest integer \( L > 0 \) such that \( X_{n-1}^n \neq X_{-m+L-1}^m \) for all \( 1 \leq m \leq n \), that is, the largest repeated pattern in the first \( n \) positions is of length \( L \). Assume, as in the first scheme, that \( X_{n-1}^n \) is known to the decoder. The decoder observes \( X_{n-1}^n \), and for some \( m_0 \) we have \( X_{m_0}^n = X_{-m_0+L-1}^n \). Therefore, \( X_{m_0}^n \) is a repeated pattern of length \( L \). To encode \( m_0 \) we need \( \log_2 |A| \) symbols, and it is known that \( L_n \) can be represented by \( \log_2 L_n \) bits (cf. [18]). As noted by Wyner and Ziv [23] the number of encoded symbols per source symbol is asymptotically

\[
\frac{\log_2 |A| n}{L_n} + \frac{\log_2 L_n}{L_n} + \frac{\log_2 |A| / L_n}{L_n},
\]

Again, the ratio \( \log_2 |A| n / L_n \) determines asymptotically the efficiency of the compression scheme.

EXAMPLE 1.3 String Matching Algorithms

Repeated substrings also arise in many algorithms on strings, notably string matching algorithms (cf. [2], [6], [12], [18], [22]). A string matching algorithm searches for all (exact) occurrences of the pattern string \( P \) in the text string \( T \). Consider either Knuth-Morris Pratt (KMP) algorithm [13] or Boyer-Moore (BM) algorithm [6]. Both algorithms rely on an observation that in the case of a mismatch between \( T \) and \( P \), say at position \( n + 1 \) of \( P \), the next attempt to match depends on the internal structure (i.e., repeated substrings) of the first \( n \) symbols of the pattern \( P \) (since they
agree with the last $n$ symbols of the unknown text string $T$). It turns out that this problem can be efficiently solved by means of a suffix tree (cf. [1], [2], [3], [9], [7], [11], [15], [22]). We shall show that suffix tree can be used to analyze $N_l$ and $L_n$ defined in the data compression examples. 

From the above one may conclude that suffix tree can be used to unify analyses of repeated subpatterns. Therefore, a short description of suffix tree follows. An interested reader may find more on suffix tree and its applications in Aho et al. [2] and Apostolico [1]. A suffix tree is a digital tree built from suffixes of a string $X$. In general, a digital tree - that is also called a trie - stores a set of words (strings, sequences, keys) $W = \{X_1, \ldots, X_n\}$ composed of symbols from a finite alphabet $A$. A trie consists of branching nodes, called also internal nodes, and external nodes that store the keys. In addition, we assume that every external node is able to store only one key. The branching policy at any level, say $k$, is based on the $k$-th symbol of a string (key, word). For example, for a binary alphabet $A = \{0, 1\}$, if the $k$-th symbol in a key is "0", then we branch-out left in the trie, otherwise we go to the right. This process terminates when for the first time we encounter a different symbol between a key that is currently inserted into the trie and all other keys already in the trie. Then, this new key is stored in a newly generated external node. In other words, the access path from the root to an external node (a leaf of a trie) is the minimal prefix of the information contained in this external node; it is minimal in the sense that this prefix is not a prefix of any other keys. Having this in mind, we repeat that a trie built over $W = \{S_1, S_2, \ldots, S_n\}$ where $S_i$ is the $i$th suffix of one-sided sequence $\{X\}_{j=1}^{\infty}$, is called suffix tree.

**EXAMPLE 1.4 Suffix tree**

Let $X = 0101101110\ldots$. Then the first five suffixes are $S_1 = 0101101110\ldots$, $S_2 = 1011011110\ldots$, $S_3 = 01101110\ldots$, $S_4 = 1101110\ldots$ and $S_5 = 101110\ldots$. The suffix tree built from these first five suffixes of $X$ is presented in Figure 1.

An important parameter of a suffix tree, that plays crucial role in the analysis and design of algorithms on strings and data compression, is the depth of a suffix. Let $S_n$ be a suffix tree constructed from the first $n$ suffixes of a sequence $X$. Then, the depth of the $i$th suffix $L_n(i)$ in $S_n$ is the length of path from the root to this suffix. We shall write $L_n = \max_{i=1}^{n} L_n(i)$ (see Section 2 for more details regarding this definition). This parameter can be called the depth of insertion since it gives the length of the path for the $n+1$st suffix after its insertion into suffix tree $S_n$. We can show that the depth of insertion $L_n$ in a suffix tree $S_n$ built over $\{X_k\}$, is the same as $L_n$ discussed above in the data compression examples (for details see Szpankowski [21]).

From the previous discussion it should be clear that behavior of $L_n$ is of considerable importance to combinatorial problems on words, in particular to data compression and string algorithms. The probabilistic behavior of $L_n$ for stationary and ergodic sequences was studied by Wyner and Ziv [23]. They established the following asymptotic result.
Figure 1: Suffix tree built from the first five suffixes of $X = 0101101110...$. 

$S_1 = 0101101110$
$S_2 = 101101110$
$S_3 = 01101110$
$S_4 = 1101110$
$S_5 = 101110$
Theorem 0. (Wyner and Ziv [23]) Let \( \{X_n\}_{n=-\infty}^{\infty} \) be a stationary and ergodic sequence built over a finite alphabet \( \mathcal{A} \). Then as \( n \to \infty \)

\[
\frac{L_n}{\log n} \to \frac{1}{h} \quad \text{in probability (pr.)} \tag{1.3a}
\]

and

\[
\frac{\log N_t}{\ell} \to h \quad \text{in probability (pr.)} \tag{1.3b}
\]

where \( h \) is the entropy of \( X \). ■

This result concerns the convergence in probability (pr.) of \( L_n \), and Wyner and Ziv [23] asked whether it can be extended to a stronger almost sure (a.s.) convergence. We shall settle this question in the negative, and show that \( L_n \) oscillates with probability one between \((1/h_1) \log n \) and \((1/h_2) \log n \) where \( 0 < h_2 < h \leq h_1 < \infty \). Moreover, during the course of the proof we establish another result - not discussed in this extended abstract - regarding a typical (probabilistic) behavior of a suffix tree \( S_n \).

Let height \( H_n \) and shortest path (called also shallowness) \( s_n \) be the longest and the shortest paths in a suffix tree \( S_n \). Then, under some additional assumption involving mixing conditions, we can show that \( H_n \sim (1/h_1) \log n \) (a.s.) and \( s_n \sim (1/h_1) \log n \) (a.s.), that is, \( L_n \) infinitely often is equal to \( H_n \) and \( s_n \).

Asymptotic analyses of suffix trees and (universal) data compressions are very scanty in the literature. To our best knowledge, asymptotic analysis of universal data compression was pursued by Ziv and Lampel (cf. [24], [14], [25]), and Wyner and Ziv [23]. The average case analysis of suffix trees was initialized by Apostolico and Szpankowski [9]. For the Bernoulli model (independent sequence of letters from a finite alphabet) the asymptotic behavior of the height was recently obtained by Devroye, Szpankowski and Rais [7], and the limiting distribution of the depth in a suffix tree is reported in Jacquet and Szpankowski [11]. Some related topics were discussed by Guibas and Odlyzko in [8] and [9]. However, our findings were inspired by a seminal paper of Pittel [16] who considered a typical behavior of a trie constructed from independent words.

In this conference version we only concentrate on data compression problems, and delayed all proofs to the final version of the paper (cf. [21]).

2. MAIN RESULTS

Let \( \{X_k\}_{k=-m}^{\infty} \) be a stationary ergodic sequence of symbols generated from a finite alphabet \( \mathcal{A} \). Define a partial sequence \( X_m^k \) as \( X_m^k = (X_m, ..., X_k) \) and \( m < n \). Furthermore, for every \( n \geq 1 \) let the nth-order probability distribution for \( \{X_k\} \) be

\[
P(X_k^n) = \Pr \{X_k = x_k, 1 \leq k \leq n, x_k \in \mathcal{A} \} \tag{2.1}
\]

Finally, the entropy of \( \{X_k\} \) is

\[
h = \lim_{n \to \infty} \frac{E \log P^{-1}(X_k^n)}{n} \tag{2.2}
\]
The existence of the above limit is guaranteed by Kolmogorov-Sinai Theorem (cf. [4]). It is also known that $h \leq \log |A|$. Hereafter, all logarithms – unless stated explicitly otherwise – are natural logarithms.

It is well known that the entropy of a stationary ergodic information source is intimately related to coding and certain data compression schemes, most notably universal compression scheme of Lampel and Ziv [14], [24]. To illustrate it Wyner and Ziv [23] introduced two quantities that capture some properties of repeated subsequences, and we define them in sequel. These quantities have been already discussed in Examples 1.1 and 1.2, but below we give more formal definitions. Let for every positive integer $\ell$ the quantity $N_\ell$ be the smallest integer $N > 0$ such that

$$X_{\ell}^{\ell-1} = X_N^{N+\ell-1}. \tag{2.3}$$

In words, for every sequence $\{X_k\}$ and given $\ell \geq 1$ the first occurrence of the pattern $X_{\ell}^{\ell-1}$ appears at position $-N_\ell$. Application of $N_\ell$ for data compression are discussed in Example 1.1.

Definition (2.3) suggests also another quantity of interest. If one fix $N = n$ in (2.3) then we call look for such an $L_n = \ell$ that (2.3) holds. More precisely, following Wyner and Ziv, we define $L_n$ as the smallest integer $L > 0$ such that

$$X_{0}^{L-1} \neq X_{-m+L-1}^{n+\ell-1} \quad \text{for all} \quad 1 \leq m \leq n. \tag{2.4}$$

It turns out, however, that another definition of $L_n$ is more convenient for our purpose. For every $n$ in the range $1 \leq m \leq n$ define self-alignment $C_{m,n+1}$ as the the length of the longest common prefix of the $m$th suffix $X_{m}^{\infty}$ and the $n + 1$st suffix $X_{n}^{\infty}$ of the sequence $X_m^{\infty}$. In other words, $C_{m,n+1} = k$ if $X_{m}^{k}$ and $X_n^{k}$ agree on the first $k$ symbols and disagree on the $k + 1$st symbol. Then, as easy to see, we can alternatively define $L_n$ as follows

$$L_n = \max_{1 \leq m \leq n} \{C_{m,n+1}\} + 1. \tag{2.5}$$

We shall use this definition to relate $L_n$ to some parameters of a suffix tree built from the first $n + 1$ suffixes of $X_m^{\infty}$. This equivalence is crucial for our results and proofs (for details see Szpankowski [21]).

EXAMPLE 2.1 Illustration of definitions

Let a sequence of $\{X_k\}$ be given as in Figure 2. This sequence is identical to the one discussed by Wyner and Ziv [23], and our Example 1.4. Note that according to definition (2.4) one finds (for the data shown in Figure 2) that $L_4 = 5$, and by (2.3) we have $N_3 = N_4 = 3$ but $N_5 > 4$. On the other hand, using the self-alignments we obtain $C_{4,5} = C_{7,5} = 0$, $C_{3,3} = 4$ and $C_{1,3} = 1$, hence according to
(2.5) $L_4 = \max\{1, 0, 4, 0\} = 4$ as needed for definition (2.4). We also note that $L_4$ is equal to the depth of insertion of the fifth suffix $S_5$ into the suffix tree $S_4$ (cf. Figure 1 where $S_5$ is shown).

There is a relation between $N_t$ and $L_n$ which we discuss next. From definitions (2.3) and (2.5) Wyner and Ziv [23] noted that

$$\{N_t > n\} = \{X_0^{t-1} = X_{-m+\ell-1}, 1 \leq m \leq n\} = \{L_n \leq \ell\}.$$  \hspace{1cm} (2.6)

Later, we shall discuss one more relationship between $N_t$ and $L_n$ that occurs in a suffix tree built from $\{X_k\}_{k=-n}^{\infty}$.

Examples 1.1-1.3 of Introduction clearly show that asymptotic properties of $L_n$ and $N_t$ are of considerable interest to many algorithms on words, including data compression. To formulate our main result we need to strengthen a little bit our assumption regarding the sequence $\{X_k\}_{k=-n}^{\infty}$. Namely, we assume mixing condition (cf. [4]). More precisely, denote $\mathcal{F}_m$ a $\sigma$-field generated by $\{X_k\}_{k=-m}^{n}$ for $m \leq n$. It is said that $\{X_k\}$ satisfies mixing condition if there exist two constants $c_1 \leq c_2$ and integer $d$ such that for all $-\infty < m \leq m + d \leq n$ the following holds

$$c_1 \Pr\{B\} \Pr\{C\} \leq \Pr\{BC\} \leq c_2 \Pr\{B\} \Pr\{C\}$$  \hspace{1cm} (2.7a)

where $B \in \mathcal{F}_m$ and $C \in \mathcal{F}_{m+d}$. In some statements of our results we need stronger form of the above mixing condition. Namely, we assume strong $\alpha$-mixing condition which replaces (2.7a) by

$$(1 - \alpha(d)) \Pr\{B\} \Pr\{C\} \leq \Pr\{BC\} \leq (1 - \alpha(d)) \Pr\{B\} \Pr\{C\}$$  \hspace{1cm} (2.7b)

where $B \in \mathcal{F}_m$ and $C \in \mathcal{F}_{m+d}$ and $\alpha(d) \to 0$ as $d \to \infty$.

To formulate our main result we follow Pittel [16] and define two new parameters of $\{X_k\}$, namely

$$h_1 = \lim_{n \to \infty} \max_{n} \frac{\log \sum_{i \geq 1} P^{-1}(X_i^t), P(X_i^t) > 0}{n} = \lim_{n \to \infty} \frac{\log(1/ \min\{P(X_i^t), P(X_i^t) > 0\})}{n}$$ \hspace{1cm} (2.8a)

$$h_2 = \lim_{n \to \infty} \frac{\log \sum_{i \geq 1} (P(X_i^t))^{-1}}{2n} = \frac{\log(\sum_{i \geq 1} P^2(X_i^t))^{-1}}{2n}.$$ \hspace{1cm} (2.8b)

The existence of $h_1$ and $h_2$ under assumption (2.7a) was established by Pittel [16]. Then, our main result is given in the following theorem.

**Theorem 1.** Let strongly $\alpha$-mixing condition (2.7b) holds together with $h_2 > 0$ and $h_1 < \infty$. Then,

$$\lim_{n \to \infty} \inf_{n} \frac{L_n}{\log n} = \frac{1}{h_1} \quad (a.s) \quad \lim_{n \to \infty} \sup_{n} \frac{L_n}{\log n} = \frac{1}{h_2}$$ \hspace{1cm} (2.9a)

and

$$\lim_{\ell \to \infty} \inf_{\ell} \frac{N_\ell}{\ell} = h_2 \quad (a.s) \quad \lim_{\ell \to \infty} \sup_{\ell} \frac{N_\ell}{\ell} = h_1$$ \hspace{1cm} (2.9b)
for all stationary ergodic sequences \( \{X_k\}_{k=-\infty}^{\infty} \) provided
\[
\alpha(n) = O(n^\beta)
\]
for some constants \( 0 < \rho < 1 \) and \( \beta \). □

**Remarks.**

(i) **Probabilistic Models Satisfying (2.10).** Possibly condition (2.10) is more restrictive than necessary. In fact, we need it only for establishing the lower bound in the liminf case. Nevertheless, even with (2.10) we can cover many interesting cases including Bernoulli model and Markovian model. In the Bernoulli model symbols of the alphabet \( A \) are generated independently, that is, \( P(X_1^2) = P^n(X_1^1) \). In particular, we assume that the \( i \) th symbol from the alphabet \( A \) is generated according to the probability \( p_i \), where \( 1 \leq i \leq |A| \) and \( \sum_{i=1}^{|A|} p_i = 1 \). Naturally, (2.10) holds since in this case \( \alpha(n) = 0 \). In the Markovian model \( \{X_k\} \) is a stationary Markov chain, that is, the \( k+1 \)st symbol in \( \{X_k\} \) depends on the previously selected symbol. We define a transition probability as \( p_{ij} = P(X_{k+1} = j \mid X_k = i \in A) \). The transition matrix is denoted by \( P = (p_{ij})_{i,j=1}^{|A|} \). It is known (cf. [4]) that for a finite state Markov chain the coefficient \( \alpha(n) \) decays exponentially, that is, for some \( c > 0 \) and \( \rho < 1 \) we have \( \alpha(n) = cp^n \), as needed for (2.10).

(ii) **Bernoulli Model.** Obviously, in this case, the strong \( \alpha \)-mixing condition is satisfied. It is also easy to notice that \( h = \sum_{i=1}^{|A|} p_i \log p_i^{-1} (\{i\}) \), and from definition (2.8) we find that \( h_1 = \log(1/p_{\min}) \) and \( h_2 = 2\log(1/P) \) where
\[
p_{\min} = \min_{1 \leq i \leq |A|} \{p_i\}, \quad P = \sum_{i=1}^{|A|} p_i^2.
\]
The probability \( P \) can be interpreted as the probability of a match between any two symbols (cf. [20]).

(iii) **Markovian Model.** It is well known that the entropy \( h \) can be computed as \( h = -\sum_{i=1}^{|A|} \pi_i \log p_{ij} \), where \( \pi_i \) is the stationary distribution of the Markov chain. The other quantities, that is \( h_1 \) and \( h_2 \) are a little more harder to evaluate. In [19] Szpankowski evaluated the height of a regular tries with Markovian dependency, and show that the parameter \( h_2 \) is a function of the largest eigenvalue \( \theta \) of the matrix \( P_{ij} = P \circ P \) which represents the Schur product of \( P \) (i.e., elementwise product). More precisely, \( h_2 = -(1/2) \log \theta \) (cf. Pittel [16]). With respect to \( h_1 \) we need to refer to Pittel [16] who cited a nonprobabilistic result of Romanovski who proved that \( \min C(\ell(C)/|C|) \) where the minimum is taken over all simple cycles \( C = \{\omega_1, \omega_2, \ldots, \omega_v, \omega_1\} \) for some \( v \leq |A| \) such that \( \omega_v \in A \), and \( \ell(C) = -\sum_{i=1}^{|A|} \log(p_{ij+1} \mod |A|) \).

(iv) **Optimal Compression Ratio.** For universal data compression one may define compression ratio \( C \) as the ratio of the overhead information necessary to implement a data compression scheme and the length of repeated (compressed) subpatterns \( L_n \). Therefore, we can define the optimal compression ratio as
\[
C_{opt} = \frac{\text{length of the minimal overhead information}}{\text{length of repeated subword}}.
\]
If the length of data base (sequence already sent and available at the decoder) is \( n \), then the length of the minimum overhead information is \( \log_V n \) where \( V \) is the size of the alphabet used in decoding the compressed information. This estimate is a simple consequence of the fact that any overhead information must at least contains information regarding a position of occurrence of the repeated pattern in the sequence of length \( n \). Therefore, we have

\[
C_{opt} = \frac{\log_V n}{L_n}. \tag{2.19}\]

A probabilistic behavior of \( C_{opt} \) depends on the kind of convergence we want to investigate. Wyner and Ziv [23] proved that \( C_{opt} \sim h/\log V \) in probability. Our Theorem 1 shows that almost surely the optimal compression ratio "swings" between \( h/\log V \) and \( h/\log V \).

References


[7] L. Devroye, W. Szpankowski and B. Rais, A note of the height of suffix trees, Purdue University, CSD TR-905 (1989); submitted also to a journal.


