On the Average Sequence Complexity*

Svante Janson
Dept. of Mathematics
Uppsala University
Uppsala, Sweden
svante.janson@math.uu.se

Stefano Lonardi
Dept. of Computer Science
University of California
Riverside, CA 92521
stelo@cs.ucr.edu

Wojciech Szpankowski
Dept. of Computer Sciences
Purdue University
West Lafayette, IN 47907
spa@cs.purdue.edu

November 17, 2003

Abstract

In this paper we study the average behavior of the number of distinct substrings in a text of size n over an alphabet of cardinality k. This quantity is called the *complexity index* and it captures the "richness of the language" used in a sequence. For example, sequences with low complexity index contain a large number of repeated substrings and they eventually become periodic (e.g., tandem repeats in a DNA sequence). In order to identify unusually low- or high-complexity strings one needs to determine how far are the complexities of the strings under study from the average or maximum string complexity. While the maximum string complexity was studied quite extensively in the past, to the best of our knowledge there are no results concerning the average complexity. We first prove that for a sequence generated by a mixing model (which includes Markov sources) the average complexity is asymptotically equal to $n^2/2$ which coincides with the maximum string complexity. However, for memoryless source we establish a more precise result, namely the average string complexity is $n^2/2 - n\log_k n + \big(1 + (1-\gamma)/\ln k + \phi_k(\log_k n) + o(1)\big)n$ where $\gamma \approx 0.577$ and $\phi_k(x)$ is a periodic function with a small amplitude for small alphabet size.

Key Words: String complexity, suffix trie, suffix trees, statistical analysis of sequences, probabilistic analysis, Stein–Chen method, Mellin transform.

[1]This work was supported in part by NSF Grants CCR-0208709, DBI-0321756 and NIH grant R01 GM068959-01.

1 Introduction

In the last decades, several attempts have been made to capture mathematically the concept of "complexity" of a sequence. The notion is connected with quite deep mathematical properties, including the rather elusive concept of randomness of a string (see, e.g., [13, 18, 21]).

In this paper, we are interested in studying a measure of complexity of a sequence called the *complexity index*. The complexity index captures the "richness of the language" used in a sequence. Formally, the *complexity index* $c(x)$ of a string x is equal to the number of distinct substrings in x (see e.g. [19]). The measure is simple but quite intuitive. Sequences with low complexity index contain a large number of repeated substrings and they eventually become periodic. However, in order to classify low complexity sequences one needs to determine average and maximum string complexity. In this paper we concentrate on the average string complexity.

We assume that sequences are generated by some probabilistic source (e.g., Bernoulli, Markov, etc.). As a consequence, the number $c(x)$ of distinct substrings can be modeled by a random variable over a discrete domain. Given a source emitting strings of size n over an alphabet of cardinality k, we call this random variable $C_{n,k}$. The main objective of this study is to give a detailed characterization of the expectation of the random variable $C_{n,k}$.

A related notion is that of the *l-subword complexity* or *l-spectrum* $c^l(x)$ of a string x, which is the number of distinct l-mers in x, for $1 \leq l \leq |x|$. We define $C^l_{n,k}$ to be the random variable associated with the number of distinct words of size l in a random string of size n over an alphabet of cardinality k. Clearly, $C_{n,k} = \sum_{l=1}^{n} C^l_{n,k}$.

The idea of using the complexity index or the l-spectrum to characterize sequence statistically has a long history of applications in several fields, such as data compression, computational biology, data mining, computational linguistics, among others.

In dictionary-based data compression, the average length of the pointer is connected with the expected size of the dictionary which in turns depends on the number of distinct subwords (see, e.g., [6]). Low-complexity strings contain more repeated substrings and therefore one can expect them to be more compressible that strings with high complexity index. For example, in [13] bounds between subword complexity and Lempel-Ziv complexity are established.

In the analysis of biosequences, the problem of characterizing the "linguistic complexity" of DNA or proteins is quite old. In the early days of computational biology, it was almost routine to compute the number and/or the frequency of occurrences of distinct l-mers and draw conclusions about the string under study based on those counts (see [22, 7, 14, 12, 15], just to mention a few).

In these and several other application domains, the typical problem associated with the complexity index is to determining whether a particular sequence x has a *statistically significant* complexity index. An example of a significance score proposed in a recent paper by Troyanskaya et al. [30] in the context of the analysis of prokaryotic genomes, is the following

$$s(x) = c(x) - \max\{C_{n,k}\} = c(x) - \sum_{i=1}^{n} \min(k^i, n - i + 1).$$

Here, the authors compare the observed complexity $c(x)$ with the maximum possible complexity for a string of size n over an alphabet of cardinality k. Note however, that the score disregards both the distribution of $C_{n,k}$, and the probabilistic characteristics of the source.

A more statistically-sound approach would entail the following steps. First, select an appropriate model for the source that emitted x (Bernoulli, Markov, etc.). Then, measure the statistical significance as a function of the discrepancy between the observed complexity $c(x)$ and the model-based expectation.

This approach of standardizing the value of an observation with respect to the expectation and the standard deviation of the associated random variable is common practice in Statistics. The underlying assumption is that the random variable is normally distributed. The standardized z-score for the complexity index would be

$$z(x) = \frac{c(x) - \mathbf{E}(C_{n,k})}{\sqrt{\mathbf{Var}(C_{n,k})}}$$

for a given string x. Although we do not know under which conditions $C_{n,k}$ is distributed normally, such a score is nonetheless more sound that other *ad hoc* scores.

A similar situation takes place when describing the significance of other events in texts, like the number of occurrences, periodicities, etc. Although the normality of the corresponding random variables can be proved under specific conditions, there is a consensus that standardized z-scores should be always preferred over simpler scores (see, e.g., [24] and references therein).

Given an observation x of the source, we would like to compute the statistical significance $z(x)$ of its complexity index. As far as we know, however, the moments $\mathbf{E}(C_{n,k})$ and $\mathbf{Var}(C_{n,k})$ have never been characterized before. The goal of this paper is to study $\mathbf{E}(C_{n,k})$. The asymptotically analysis of the variance remains an open problem.

In order to proceed, we need to introduce some standard concepts and notation about strings. The set Σ denotes a nonempty *alphabet* of *symbols* and a *string* over Σ is an ordered sequence of symbols from the alphabet. In the rest of the paper, we assume that $|\Sigma| = k$. Given a string x, the number of symbols in x defines the *length* $|x|$ of x. Henceforth, we assume $|x| = n$.

The *concatenation* (or *product*) of two strings x and y is denoted by xy, and corresponds to the operation of appending y to the last symbol of x. Let us decompose a text x in uvw, i.e., $x = uvw$ where u, v and w are strings over Σ. Strings u, v and w are called *substrings*, or *words*, of x.

We write $x_{[i]}$, $1 \leq i \leq |x|$ to indicate the i-th symbol in x. We use $x_{[i,j]}$ shorthand for the substring $x_{[i]}x_{[i+1]} \dots x_{[j]}$ where $1 \leq i \leq j \leq |x|$, with the convention that $x_{[i,i]} = x_{[i]}$. Substrings in the form $x_{[1,j]}$ corresponds to the *prefixes* of x, and substrings in the form $x_{[i,n]}$ to the *suffixes* of x.

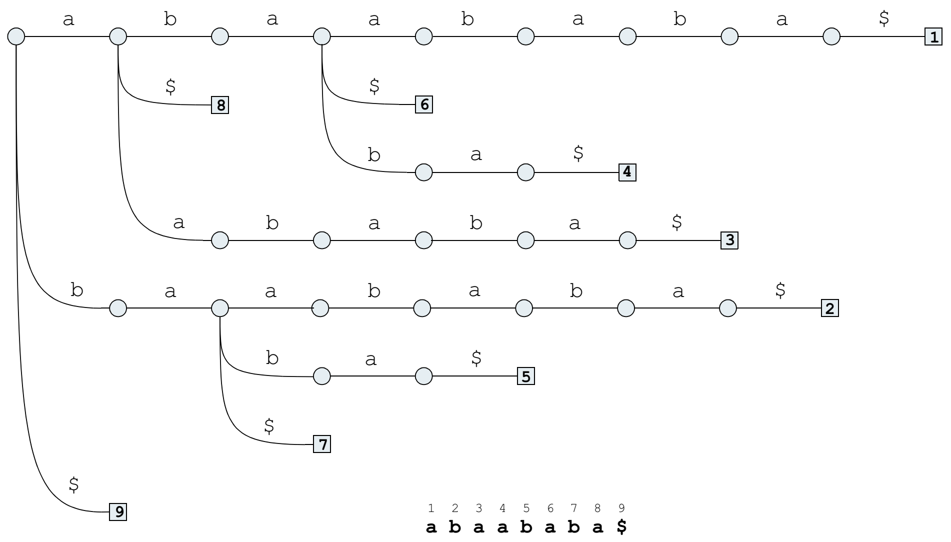

Figure 1: The non-compact suffix trie T_x for $x =$ `abaababa`. There are 24 internal non-root nodes, therefore $c(x) = 24$.

Finally, we recall that the *subword complexity* function $c^l(x)$ of a string x is defined as the number of distinct substrings of x of length l. The quantity $c(x) = \sum_{l=1}^{n} c^l(x)$ is the *complexity index* of x. Observe first that $c^l(x)$ is upper bounded by $\min(k^l, n-l+1)$ since there are precisely $n-l+1$ words of length l, of which at most k^l can be distinct. Therefore

$$c(x) \leq \sum_{l=1}^{n} \min(k^l, n-l+1).$$

Note that if we choose $n \leq k$, then $\min(k^l, n-l+1) = n-l+1$ for all $1 \leq l \leq n$. Therefore when $n \leq k$, the bound simplifies to $c(x) \leq n(n+1)/2$.

The value $c(x)$ is strongly connected with the structure of the *non-compact suffix trie* for x. A non-compact suffix trie of a string x is a digital search tree built from all the suffixes of x. The trie for a string of size n has $n+1$ leaves, numbered 1 to $n+1$, where leaf $n+1$ correspond to an extra unique symbol $\$ \notin \Sigma$, and each edge is labeled with a nonempty substring of $x\$$. No two edges outgoing from a node can have labels beginning with the same character. The trie has the property that for any leaf i, the concatenation of the labels on the path from the root the the leaf i spells out exactly the suffix of x that starts at position i, that is $x_{[i,n]}$. The substrings of x can be obtained by spelling out the words from the root to any internal node of the tree. In other words, each internal node (except the root) is in one-to-one correspondence with each distinct substring of x. As a consequence, *the complexity index $c(x)$ can be obtained by counting the non-root internal nodes in the non-compact suffix trie for x.* This would take, however, $O(n^2)$ time and space. The non-compact suffix trie for `abaababa` is illustrated in Figure 1.

A faster solution to compute $c(x)$ involves the use of the *suffix tree* $\bar{T}_x$ of x. The suffix tree can obtained by "path-compression" of the non-compact suffix trie, that is, by deleting

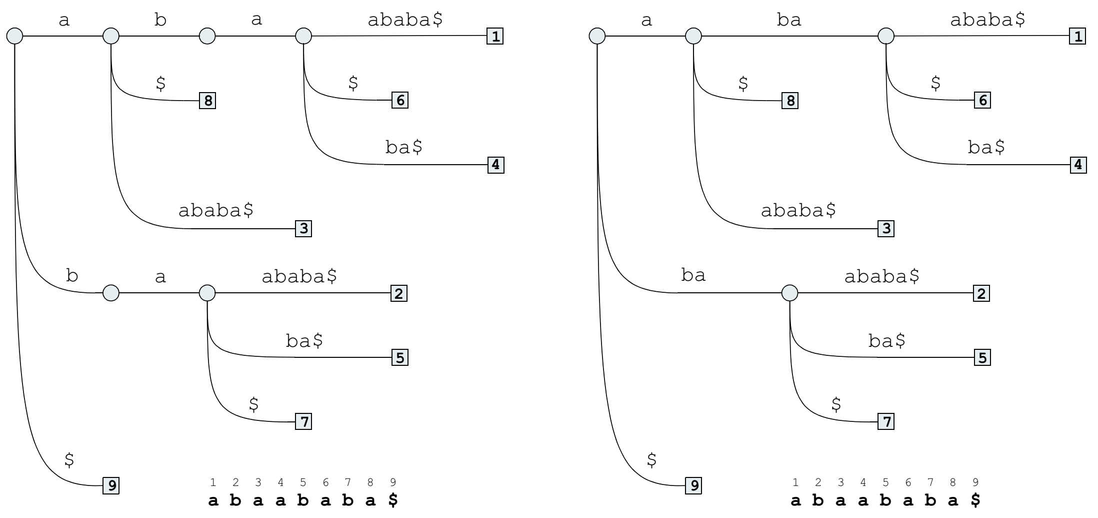

Figure 2: The compact suffix trie (LEFT) and the suffix tree (RIGHT) for $x =$ abaababa.

internal nodes with only one child and coalescing consecutive edges in a single edge labeled by the concatenation of the symbols. If one deletes unary nodes only at the *bottom* of the non-compact suffix tries, the resulting tree is called *compact suffix trie*. The compact suffix trie and the suffix tree for abaababa are shown in Figure 2.

In practice, suffix trees can be build without the need of building the suffix trie first. For example, the "brute force" construction algorithm works by inserting the suffixes of x one at a time. In the worst case still requires quadratic time, while the average time complexity is $O(n \log n)$ [4, 8].

There are however several $O(n \log |\Sigma|)$ constructions. The algorithm by McCreight [20] and the one by Chen and Seiferas [9] are variation of the Weiner's algorithm [33]. Note that these algorithms take only linear time for finite alphabets. All these constructions are *off-line* because they process the text right to left. An on-line algorithm by Ukkonen [31] achieves also linear time. Recently, Farach [11] proposed an optimal construction for large alphabets.

The unary nodes which have been removed in the compaction process are called *implicit nodes*. An edge labeled by a string of length $m + 1$ has m implicit nodes. The complexity index $c(x)$ of a text-string x can be computed on the suffix tree by counting the number of implicit and explicit (non-root) nodes in the tree. As a consequence, the $c(x)$ can be computed in $O(n)$ time and space. The relation between non-compact suffix tries and suffix trees will be used later in paper to obtain the leading term of the complexity for a general probabilistic model.

Finally, we briefly describe some recent results regarding the maximum of $c^l(x)$. It is known that $c^l(x)$ is also strongly connected with the structure of the suffix tree $\bar{T}_x$. de Luca [10] proved that

$$\max\{c^l(x) : 1 \leq l \leq n\} = n - \max\{R, K\} + 1 = n - \max\{L, H\} + 1$$

where K is the length of the shortest suffix of x that occurs only once, H is the length of the shortest prefix of x that occurs only once, $R-1$ is the height of the deepest branching node in the suffix tree $\bar{T}_x$ and $L-1$ is the height of the deepest branching node in the suffix tree $\bar{T}_{x^R}$. de Luca also gave a closed form for $c(x)$

$$c(x) = 1 + \frac{(n+K)(n-K+1)}{2} - \sum_{j=2}^{|\Sigma|}\sum_{i=0}^{R} ig(j,i)$$

where $g(j,i)$ is the count of the words of length i which are branching nodes of the suffix tree with at least j children [10].

Shallit [25] derived a simpler bound for $c(x)$ for binary alphabets ($k=2$)

$$c(x) \le \frac{(n-d+1)(n-d)}{2} + 2^{d+1} - 1 \sim \frac{n^2}{2}$$

where d is the unique integer such that $2^d + d - 1 \le n < 2^{d+1} + d$. More importantly, he proved that the upper bound is attained for all n by using a property of the *de Bruijn* graphs.

Kása [16] studied the probability distribution of random variable associated with the complexity index for the family of words of length equal to the size of the alphabet ($n = k$). He proved several facts about the random variable $C_{k,k}$, and he also conjectured a property of the smallest value of the complexity after which all the frequencies are non-zero. The conjecture, proved later by Leve and Seebold [17], states that if one chooses $k = \frac{l(l+1)}{2} + 2 + i$ where $l \ge 2$ and $0 \le i \le l$ then

$$\mathbf{P}\left(C_{k,k} = t\right) > 0 \quad \text{for all } t \text{ such that } \frac{l(l^2-1)}{2} + 3l + 2 + i(l+1) \le t \le \frac{k(k+1)}{2}$$

In this paper we mostly deal with the average string complexity. First, for a general probabilistic model (e.g., Markov source) we prove that the average complexity is asymptotically is equal to $n^2/2$ which coincides with the maximum string complexity. We shall strengthen this result for unbiased memoryless sources. In particular, we prove that

$$\mathbf{E}(C_{n,k}) = \binom{n+1}{2} - n\log_k n + \Big(\frac{1}{2} + \frac{1-\gamma}{\ln k} + \phi_k(\log_k n)\Big)n + O(\sqrt{n\log n})$$

where $\gamma \approx 0.577$ and $\phi_k(x)$ is a continuous function with period 1 and small amplitude for small alphabet size (e.g., $|\phi_2(x)| < 2 \cdot 10^{-7}$). To prove this result we use the Stein–Chen method together with the Mellin transform approach.

2 Main Results

As a warm-up exercise, we studied the closed forms for $\mathbf{E}\left(C_{n,k}\right)$ and $\mathbf{Var}\left(C_{n,k}\right)$ for short strings (e.g., $n \le 5$) for a symmetric memoryless source. Some facts about $\mathbf{P}(C_{n,k} = t)$ are

$n \to$	2	3	4	5
$\mathbf{P}(C_{n,k}=2)$	$1/k$	0	0	0
$\mathbf{P}(C_{n,k}=3)$	$1-1/k$	$1/k^2$	0	0
$\mathbf{P}(C_{n,k}=4)$	0	0	$1/k^3$	0
$\mathbf{P}(C_{n,k}=5)$	0	$3(k-1)/k^2$	0	$1/k^4$
$\mathbf{P}(C_{n,k}=6)$	0	$(k-1)(k-2)/k^2$	0	0
$\mathbf{P}(C_{n,k}=7)$	0	0	$3(k-1)/k^3$	0
$\mathbf{P}(C_{n,k}=8)$	0	0	$4(k-1)/k^3$	0
$\mathbf{P}(C_{n,k}=9)$	0	0	$6(k-1)(k-2)/k^3$	$3(k-1)/k^4$
$\mathbf{P}(C_{n,k}=10)$	0	0	$(k-1)(k-2)(k-3)/k^3$	0
$\mathbf{P}(C_{n,k}=11)$	0	0	0	$10(k-1)/k^4$
$\mathbf{P}(C_{n,k}=12)$	0	0	0	$2(k-1)(3k-5)/k^4$
$\mathbf{P}(C_{n,k}=13)$	0	0	0	$19(k-1)(k-2)/k^4$
$\mathbf{P}(C_{n,k}=14)$	0	0	0	$10(k-1)(k-2)(k-3)/k^4$
$\mathbf{P}(C_{n,k}=15)$	0	0	0	$(k-1)(k-2)(k-3)(k-4)/k^4$
$\mathbf{P}(C_{n,k}\geq 16)$	0	0	0	0

Table 1: The probability distribution of $C_{n,k}$ for $2 \leq n \leq 5$ over any alphabet of size k, under a symmetric Bernoulli source

immediate. For example

$$\begin{aligned}
\mathbf{P}(C_{n,k} < n) &= 0 \\
\mathbf{P}(C_{n,k} = n) &= k^{1-n} \\
\mathbf{P}\left(C_{n,k} > \frac{n(n+1)}{2}\right) &= 0 \text{ when } n \leq k \\
\mathbf{P}\left(C_{n,k} > \sum_{i=1}^{n} \min(k^i, n-i+1)\right) &= 0 \text{ when } n > k
\end{aligned}$$

Following the lines by Kása [16], we were able to obtain closed form for the cases shown in Table 1 assuming a symmetric Bernoulli model for the source. Given the discrete distribution of $C_{n,k}$ one can easily compute the expectation and the variance of $C_{n,k}$.

Corollary 1 *The expectation and the variance of the random variable $C_{n,k}$ for $2 \leq n \leq 5$ over any alphabet of size k, under a symmetric Bernoulli source is*

$$\begin{aligned}
\mathbf{E}(C_{2,k}) &= 3-(1/k) \\
\mathbf{E}(C_{3,k}) &= 6-(3/k) \\
\mathbf{E}(C_{4,k}) &= 10-(6/k)+(1/k^2)-(1/k^3) \\
\mathbf{E}(C_{5,k}) &= 15-(10/k)+(4/k^2)-(6/k^3)+(2/k^4)
\end{aligned}$$

and

$$\begin{aligned}
\mathbf{Var}\,(C_{2,k}) &= (k-1)/k^2 \\
\mathbf{Var}\,(C_{3,k}) &= 3(k-1)/k^2 \\
\mathbf{Var}\,(C_{4,k}) &= (k-1)(6k^4-5k^3+12k^2-k+1)/k^6 \\
\mathbf{Var}\,(C_{5,k}) &= 2(k-1)(5k^6-10k^5+33k^4-28k^3+16k^2-10k+2)/k^8
\end{aligned}$$

As it turns out, obtaining a closed form for $\mathbf{E}\,(C_{n,k})$ and $\mathbf{Var}\,(C_{n,k})$ for any n, k is a very challenging problem. In practical applications, moreover, having the closed form only for small values of n would be of limited interest. It is certainly more valuable to study the behavior of the moments of $C_{n,k}$ asymptotically, that is, when n is very large.

The main result of our study is a characterization of $\mathbf{E}\,(C_{n,k})$ for large n. In our first result we show that for quite general sources the average complexity asymptotically coincides with the maximal string complexity. We consider mixing sources in which the probability of two events A and B defined on two substrings separated by g symbols is bounded as follows: $(1-\psi(g))\mathbf{P}(A)\mathbf{P}(B) \le \mathbf{P}(AB) \le (1+\psi(g))\mathbf{P}(A)\mathbf{P}(B)$ where the mixing coefficient $\psi(g) \to 0$ as $g \to \infty$ (cf. [27] for a detailed definition).

Theorem 1 *Let $C_{n,k}$ be the complexity index of a string of length n generated by a strongly mixing stationary source. Then, for large n,*

$$\mathbf{E}C_{n,k} = \binom{n+1}{2} - O(n\log n).$$

Hence $C_{n,k} = n^2/2 + O_p(n\log n)$, i.e. $(n^2/2 - C_{n,k})/n\log n$ is bounded in probability.

Proof We start with a simple observation. For a given sequence x of size n, build a non-compact suffix trie and a compact suffix trie. Recall that in a compact trie we collapse all unary links at the *bottom* of the suffix trie. In other words, in a compact suffix trie a path from the root to an external node (containing a suffix) is the minimal prefix of any two suffixes that distinguishes them. Also recall that the string complexity of x is the number of non-root internal nodes in the non-compact trie. We shall argue below that the most contribution to the string complexity comes from the nodes that are in the non-compact trie but not in the compact trie. The upper bound follows immediately from

$$C_{n,k} \le n(n+1)/2.$$

To find a matching lower bound, we consider the compact and non-compact suffix tries. We know that a typical depth and height in a compact suffix trie is $O(\log n)$. More precisely let H_n be the height of a compact suffix tree. It was shown in [26] that (at least for $\psi(g)$ satisfying $\sum_{g\ge 0}\psi^2(g) < \infty$) $H_n/\ln n \to 2/h_1$ a.s., where h_1 is the Renyi's entropy (cf. [27, p. 157]). More precisely, the proof shows (for any $\psi(g) \to 0$) that for any $\epsilon > 0$

$$\mathbf{P}\left(H_n \le \frac{2}{h_1}(1+\epsilon)\log n\right) = 1 - O(\log n/n^{\epsilon}). \qquad (1)$$

We claim that the main contribution to $C_{n,k}$ comes from strings that are in the non-compact trie but not in the compact trie. In fact, the i-th suffix string has $n-i$ internal nodes of which at least $n-i-H_n$ are not in the compact trie. These nodes all correspond to unique substrings of x, and thus

$$C_{n,k} \geq \sum_{i=1}^{n} (n-i-H_n) = \tfrac{1}{2}n(n+1) - nH_n.$$

By (1), for a suitable constant B and large n, $\mathbf{P}\left(H_n > B\log n\right) < n^{-1}$ and thus

$$\mathbf{E}\Big(\tfrac{1}{2}n(n+1) - C_{n,k}\Big) \leq n\mathbf{E}H_n \leq n\big(B\log n + n\mathbf{P}\left(H_n > B\log n\right)\big) = O(n\log n),$$

which completes the proof. □

However, from theoretical point of view the most interesting case is when the string is generated by an unbiased source (such a source should have the largest complexity). In this case, we are able to characterize very precisely the average complexity.

Theorem 2 *Let $C_{n,k}$ be the complexity index of a string generated by an unbiased memoryless source. Then the average l-subword complexity is*

$$\mathbf{E}(C^l_{n,k}) = k^l(1-e^{-nk^{-l}}) + O(l) + O(nlk^{-l}). \tag{2}$$

Furthermore, for large n the average complexity index becomes

$$\mathbf{E}(C_{n,k}) = \binom{n+1}{2} - n\log_k n + \Big(\frac{1}{2} + \frac{1-\gamma}{\ln k} + \phi_k(\log_k n)\Big)n + O(\sqrt{n\log n})$$

where $\gamma \approx 0.577$ is Euler's constant and

$$\phi_k(x) = -\frac{1}{\ln k}\sum_{j\neq 0}\Gamma\Big(-1-\frac{2\pi ij}{\ln k}\Big)e^{2\pi ijx}$$

is a continuous function with period 1. $|\phi_k(x)|$ is very small for small k: $|\phi_2(x)| < 2\cdot 10^{-7}$, $|\phi_3(x)| < 5\cdot 10^{-5}$, $|\phi_4(x)| < 3\cdot 10^{-4}$.

Interestingly enough, the term $O(n)$ of the average complexity contains a fluctuating term $\phi_k(x)$. (Note that $\phi_2(\log_2 n)$ equals $-P_0(n)$ in [27, p. 359].) The formula

$$|\Gamma(-1-iy)| = |\Gamma(-iy)/(-1-iy)| = \big(\big(y(1+y^2)\sinh(\pi y)/\pi\big)\big)^{-1/2} \tag{3}$$

[27, p. 42] shows that the coefficients in ϕ_k are small and decrease geometrically. Numerical evaluations give the bounds for $k=2,3,4$ stated in the theorem and also, for example, $|\phi_k(x)| < 0.01$ for $k \leq 12$ and $|\phi_k(x)| < 0.1$ for $k \leq 200$. Even for very large k, this term is not very large; we have, still using (3) (we omit the details), $|\phi_k(x)| < 0.5$ for $k \leq 10^9$, and $|\phi_k(x)| < \ln\ln(k)/\pi$ for all k. The fluctuating phenomenon is quite common in asymptotics of discrete problems.

3 Proof of Theorem 2

In this section we prove Theorem 2, however, to simplify our notation we restrict ourselves to the binary case $k = 2$. Extension to $k > 2$ is straightforward.

Recall that $C^l_{n,2}$ is the number of distinct substrings of length l in our random binary string of size n, and let

$$A_l = \mathbf{E}(C^l_{n,2}).$$

Thus $C_{n,2} = \sum_{l=1}^n C^l_{n,2}$ and $\mathbf{E}(C_{n,2}) = \sum_{l=1}^n A_l$. Define

$$\begin{aligned} \delta_l &= A_l + l - 1 - 2^l\bigl(1 - e^{-n2^{-l}}\bigr) && (4)\\ &= A_l - (n+1-l) + 2^l\bigl(e^{-n2^{-l}} - 1 + n2^{-l}\bigr). && (5) \end{aligned}$$

Then

$$\begin{aligned} \mathbf{E}(C_{n,2}) &= \sum_{l=1}^n A_l = \sum_{l=1}^n \Bigl((n+1-l) + \delta_l - 2^l\bigl(e^{-n2^{-l}} - 1 + n2^{-l}\bigr)\Bigr)\\ &= \binom{n+1}{2} - \sum_{l=1}^n 2^l\bigl(e^{-n2^{-l}} - 1 + n2^{-l}\bigr) + \sum_{l=1}^n \delta_l. && (6) \end{aligned}$$

Below we will use several times the estimates (for $x > 0$) $0 < 1 - e^{-x} < \min(1, x)$ and $0 < e^{-x} - 1 + x < \min(x, x^2)$. In particular,

$$\begin{aligned} &0 < 2^l\bigl(1 - e^{-n2^{-l}}\bigr) < \min(2^l, n), && (7)\\ &0 < 2^l\bigl(e^{-n2^{-l}} - 1 + n2^{-l}\bigr) < n^2 2^{-l}. && (8) \end{aligned}$$

We begin by estimating δ_l. First (for short strings, i.e., for small l), we use $1 \le C^l_{n,2} \le 2^l$, and thus $0 \le A_l \le 2^l$ and, using (4) and (7),

$$\delta_l = O(2^l). \tag{9}$$

For long substrings, i.e., for large l's, there is another simple estimate. Clearly, $0 \le C^l_{n,2} \le n - l + 1$. Observe that

$$\mathbf{E}(n-l+1-C^l_{n,2}) \le \mathbf{E}\left(\bigl|\{(i,j) : i < j \text{ and } x_{[i,i+l-1]} = x_{[j,j+l-1]}\}\bigr|\right) = \binom{n-l+1}{2} 2^{-l} \le n^2 2^{-l}$$

i.e.

$$0 \le n - l + 1 - A_l \le n^2 2^{-l}.$$

Hence, using (5) and (8),

$$\delta_l = O(n^2 2^{-l}). \tag{10}$$

Note that (9) and (10) easily yield $\sum_{l=1}^n \delta_l = O(n)$, which by (6) yields $\mathbf{E}(C_{n,2})$ up to $O(n)$. In order to obtain our more precise result, we need a better estimate of δ_l when $2^l \approx n$.

Let x be a sequence generated by an i.i.d. source (i.e., $p(0) = p(1) = 1/2$) and let us define

$$\mathbf{P}(n,l) \;=\; \mathbf{P}(x_{[1,l]} \neq x_{[j,j+l-1]} \text{ for } j = 2,\ldots,n-l+1)$$

By counting each repeated substring in x only the last time it occurs, it is easily seen (by shift invariance) that

$$A_l = \sum_{m=l}^{n} \mathbf{P}(m,l) \tag{11}$$

Now fix l and m, and let us define $I_j = \mathbf{1}[x_{[1,l]} = x_{[j,j+l-1]}]$ for $j = 2,\ldots,m-l+1$. Then

$$\mathbf{E}(I_j) = \mathbf{P}(I_j = 1) = \mathbf{P}[x_{[1,l]} = x_{[j,j+l-1]}] = 2^{-l} \text{ for every } j \geq 2.$$

In the next lemma we establish that I_i and I_j are uncorrelated when $i, j > l$.

Lemma 1 *If $i, j \geq l+1$ and $i \neq j$ then $\mathbf{E}(I_i I_j) = \mathbf{P}(x_{[1,l]} = x_{[i,i+l-1]} = x_{[j,j+l-1]}) = 2^{-2l}$.*

Proof Assume $i < j$. Scan the three substring left to right. Each bit in $x_{[i,i+l-1]}$ is either a fresh random bit or coincides with an earlier bit in $x_{[j,j+l-1]}$. In any case, it fixes one new bit each in $x_{[1,l]}$ and $x_{[j,j+l-1]}$, so the probability of success in this step is 2^{-2}. □

Observe that, if $j \geq l+1$, to condition on I_j is the same as to change every bit in $x_{[j,j+l-1]}$ to the corresponding bit in $x_{[1,l]}$, leaving all other bits untouched. Let $x^{(j)}$ be the resulting string, and let $J_{ij} = \mathbf{1}[x^{(j)}_{[1,l]} = x^{(j)}_{[i,i+l-1]}]$. Clearly, $J_{ij} = I_i$ if $|i-j| \geq l$. Note also that Lemma 1 yields, when $i \neq j$ and $i, j > l$,

$$\mathbf{E}(J_{ij}) = \mathbf{E}(I_i \mid I_j = 1) = \frac{\mathbf{E}(I_i I_j)}{\mathbf{E}(I_j)} = 2^{-l}.$$

Let us now set, for fixed integers l and $m \geq 2l$,

$$W = \sum_{j=l+1}^{m-l+1} I_j.$$

Let $d_{TV}(X,Y)$ be the total variation distance between random variables X and Y. Then, with $\mathrm{Po}(\lambda)$ being the Poisson distribution with mean λ, by the Stein–Chen method (cf. [5, page 24]) we find, with $\lambda = \mathbf{E}W = (m-2l+1)2^{-l}$,

$$\begin{aligned}
d_{TV}\big(W, \mathrm{Po}((m-2l+1)2^{-l})\big) &\leq \frac{\min(1,\lambda)}{\lambda} \sum_j \mathbf{E}(I_j)\mathbf{E}\left| I_j + \sum_{i\neq j}(I_i - J_{ij}) \right| \\
&\leq \frac{1}{\lambda} \sum_j \mathbf{E}(I_j)\Big(\mathbf{E}(I_j) + \sum_{0<|i-j|<l} \big(\mathbf{E}(I_i) + \mathbf{E}(J_{ij})\big)\Big) \\
&\leq \frac{1}{\lambda} \sum_{j=l+1}^{m-l-1} 2l \cdot 2 \cdot 2^{-2l} \\
&= 4l2^{-l}.
\end{aligned}$$

In particular,

$$\left|\mathbf{P}(W=0)-e^{-(m-2l+1)2^{-l}}\right| \le d_{TV}\big(W, \mathrm{Po}((m-2l+1)2^{-l})\big) = O(l2^{-l}). \tag{12}$$

Moreover, by the first moment method

$$\mathbf{P}\left(\sum_{j=2}^{l} I_j \neq 0\right) \le (l-1)\mathbf{E}(I_j) = (l-1)2^{-l}. \tag{13}$$

Observe that

$$\mathbf{P}(m,l) = \mathbf{P}\left(\sum_{j=2}^{l} I_j + W = 0\right).$$

Then by (13) and (12)

$$\mathbf{P}(m,l) = \mathbf{P}(W=0) + O(l2^{-l}) = e^{-(m-2l+1)2^{-l}} + O(l2^{-l}) = e^{-(m-l)2^{-l}} + O(l2^{-l}). \tag{14}$$

We have assumed $m \ge 2l$. However, by the first moment method directly, the same estimate holds for $l \le m < 2l$ too.

We thus have, by (11) and (14) and summing a geometric series,

$$A_l = \sum_{m=l}^{n} \mathbf{P}(m,l) = \frac{1-e^{-(n+1-l)2^{-l}}}{1-e^{-2^{-l}}} + O(nl2^{-l}).$$

Since

$$\begin{aligned} \frac{1-e^{-(n+1-l)2^{-l}}}{1-e^{-2^{-l}}} &= \frac{1-e^{-(n+1-l)2^{-l}}}{2^{-l}(1+O(2^{-l}))} \\ &= (2^l+O(1))(1-e^{-(n+1-l)2^{-l}}) \\ &= 2^l(1-e^{-(n+1-l)2^{-l}}) + O(n2^{-l}) \\ &= 2^l(1-e^{-n2^{-l}}) + O(l) + O(n2^{-l}), \end{aligned}$$

we find

$$A_l = 2^l(1-e^{-n2^{-l}}) + O(l) + O(nl2^{-l})$$

which proves (2). Thus by (4)

$$\delta_l = O(l) + O(nl2^{-l}). \tag{15}$$

Using (9) for $1 \le l \le \log_2 \sqrt{n \ln n}$, (15) for $\log_2 \sqrt{n \ln n} < l \le 2\log_2 n$, and (10) for $2\log_2 n < l \le n$, we obtain

$$\sum_{l=1}^{n} \delta_l = O(n \log n)^{1/2}. \tag{16}$$

We turn to the first sum in (6). Let, for $x > 0$,

$$f(x) = \frac{e^{-x} - 1 + x\mathbf{1}[x < 1]}{x}.$$

Then $|f(x)| < x$ for $0 < x < 1$ and $|f(x)| < 1/x$ for $x \geq 1$. Hence,

$$\begin{aligned}\sum_{l=1}^{n} 2^l\bigl(e^{-n2^{-l}} - 1 + n2^{-l}\bigr) &= n\sum_{l=1}^{n}\bigl(f(n2^{-l}) + \mathbf{1}[2^l \leq n]\bigr)\\ &= n\sum_{l=1}^{n} f(n2^{-l}) + n\lfloor\log_2 n\rfloor\\ &= n\sum_{l=-\infty}^{\infty} f(n2^{-l}) + O(1) + n\lfloor\log_2 n\rfloor\\ &= n\psi(n) + n\log_2 n + O(1),\end{aligned} \tag{17}$$

where, with $\langle x\rangle = x - \lfloor x\rfloor$, the fractional part of x,

$$\psi(x) = \sum_{l=-\infty}^{\infty} f(x2^{-l}) - \langle\log_2 x\rangle, \qquad x > 0.$$

(The series converges by the estimates above.) It is easily verified that ψ is bounded, continuous (also at powers of 2), and periodic in $\log_2 x$, i.e., $\psi(2x) = \psi(x)$. Hence $\psi(2^y)$ has period 1 and may be expanded in a Fourier series

$$\psi(2^y) = \sum_{\nu=-\infty}^{\infty} c_\nu e^{2\pi i\nu y} \tag{18}$$

where

$$\begin{aligned}c_\nu &= \int_0^1 e^{-2\pi i\nu y}\psi(2^y)\,dy\\ &= \int_0^1 e^{-2\pi i\nu y}\Bigl(\sum_{l=-\infty}^{\infty} f\bigl(2^{y-l}\bigr) - y\Bigr)\,dy\\ &= \sum_{l=-\infty}^{\infty}\int_0^1 e^{-2\pi i\nu y} f\bigl(2^{y-l}\bigr)\,dy - \int_0^1 e^{-2\pi i\nu y} y\,dy\\ &= \int_{-\infty}^{\infty} e^{-2\pi i\nu y} f\bigl(2^y\bigr)\,dy - \int_0^1 e^{-2\pi i\nu y} y\,dy.\end{aligned} \tag{19}$$

Further, changing variables back to $(0, \infty)$,

$$\int_{-\infty}^{\infty} e^{-2\pi i\nu y} f\bigl(2^y\bigr)\,dy = \frac{1}{\ln 2}\int_0^{\infty} x^{-2\pi i\nu/\ln 2} f(x)\frac{dx}{x} = \frac{1}{\ln 2}\mathcal{M}[f(x); -2\pi i\nu/\ln 2], \tag{20}$$

where $\mathcal{M}[f(x); s] = \int_0^\infty x^{s-1} f(x)\,dx$ is the Mellin transform of f at the point s (in our case $s = -2\pi i\nu/\ln 2$. (See [27, Chapter 9] for definition and basic properties.) Since $|f(x)| < \min(x, x^{-1})$, the Mellin transform $\mathcal{M}[f(x); z]$ is analytic in the strip $-1 < \Re z < 1$. In the smaller strip $0 < \Re z < 1$ we have, by [27, Table 9.1],

$$\begin{aligned}\mathcal{M}[f(x); z] &= \mathcal{M}\Big(\frac{e^{-x}-1}{x}; z\Big) + \mathcal{M}\big(\mathbf{1}[x<1]; z\big)\\ &= \mathcal{M}\big(e^{-x}-1; z-1\big) + \mathcal{M}\big(\mathbf{1}[x<1]; z\big)\\ &= \Gamma(z-1) + \frac{1}{z}.\end{aligned} \tag{21}$$

By analyticity, this extends to the strip $-1 < \Re z < 1$. In particular, taking the limit as $z \to 0$,

$$\mathcal{M}[f(x); 0] = \lim_{z\to 0}\Big(\frac{\Gamma(1+z)}{z(z-1)} + \frac{1}{z}\Big) = \lim_{z\to 0}\frac{\Gamma(1+z)-1+z}{z(z-1)} = -\big(\Gamma'(1)+1\big) = \gamma - 1. \tag{22}$$

Moreover, elementary integration yields

$$\int_0^1 e^{-2\pi i\nu y} y\,dy = \begin{cases} -1/2\pi i\nu, & \nu \neq 0,\\ 1/2, & \nu = 0.\end{cases} \tag{23}$$

By (19)–(23),

$$\begin{aligned} c_\nu &= \frac{1}{\ln 2}\Gamma\Big(-1 - \frac{2\pi i\nu}{\ln 2}\Big), && \nu \neq 0,\\ c_0 &= \frac{\gamma-1}{\ln 2} - 1/2. \end{aligned}$$

The theorem now follows, with $\phi(x) = -(\psi(2^x) - c_0)$, from (6), (16), (17) and (18).

The numerical bounds for $k \le 4$ are obtained from (3); for small k, $\sum_1^\infty |c_\nu|$ is dominated by $|c_1|$ which is very small.

References

[1] Apostolico, A. The myriad virtues of suffix trees. In *Combinatorial Algorithms on Words*, A. Apostolico and Z. Galil, Eds., vol. 12 of *NATO Advanced Science Institutes, Series F*. Springer-Verlag, Berlin, 1985, pp. 85–96.

[2] Apostolico, A., Bock, M. E., and Lonardi, S. Monotony of surprise and large-scale quest for unusual words. *J. Comput. Bio. 10*, 3-4 (2003), pp. 283–311.

[3] Apostolico, A., Bock, M. E., Lonardi, S., and Xu, X. Efficient detection of unusual words. *J. Comput. Bio. 7*, 1/2 (2000), pp. 71–94.

[4] Apostolico, A., and Szpankowski, W. Self-alignment in words and their applications. *J. Algorithms 13*, 3 (1992), pp. 446–467.

[5] Barbour, A., Holst, K., Janson, S. *Poisson Approximation.* Oxford Press, Oxford, 1992.

[6] Bell, T. C., Cleary, J. G., and Witten, I. H. *Text Compression.* Prentice Hall, 1990.

[7] Burge, C., Campbell, A., and Karlin, S. Over- and under-representation of short oligonucleotides in DNA sequences. *Proc. Natl. Acad. Sci. U.S.A. 89* (1992), pp. 1358–1362.

[8] Chang, W. I., and Lawler, E. L. Sublinear approximate string matching and biological applications. *Algorithmica 12*, 4/5 (1994), pp. 327–344.

[9] Chen, M. T., and Seiferas, J. Efficient and elegant subword tree construction. In *Combinatorial Algorithms on Words*, A. Apostolico and Z. Galil, Eds., vol. 12 of *NATO Advanced Science Institutes, Series F.* Springer-Verlag, Berlin, 1985, pp. 97–107.

[10] de Luca, A. On the combinatorics of finite words. *Theor. Comput. Sci. 218*, 1 (1999), pp. 13–39.

[11] Farach, M. Optimal suffix tree construction with large alphabets. In *Proc. 38th Annual Symposium on Foundations of Computer Science* (1997), pp. 137–143.

[12] Fickett, J. W., Torney, D. C., and Wolf, D. R. Base compositional structure of genomes. *Genomics 13* (1992), pp. 1056–1064.

[13] Ilie, L., Yu, S., and Zhang, K. Repetition Complexity of Words In *Proc. COCOON* (2002), pp. 320–329.

[14] Karlin, S., Burge, C., and Campbell, A. M. Statistical analyses of counts and distributions of restriction sites in DNA sequences. *Nucleic Acids Res. 20* (1992), pp. 1363–1370.

[15] Karlin, S., Mrazek, J., and Campbell, A. M. Frequent oligonucleotides and peptides of the *Haemophilus influenzae* genome. *Nucleic Acids Res. 24*, 21 (1996), pp. 4273–4281.

[16] Kása, Z. On the d-complexity of strings. *Pure Math. Appl. 9*, 1-2 (1998), pp. 119–128.

[17] Leve, F., and Seebold, P. Proof of a conjecture on word complexity. *Bulletin of the Belgian Mathematical Society 8*, 2 (2001).

[18] Li, M., and Vitanyi, P. *Introduction to Kolmogorov Complexity and its Applications.* Springer-Verlag, Berlin, Aug. 1993.

[19] Lothaire, M., Ed. *Algebraic Combinatorics on Words.* Cambridge University Press, 2002.

[20] McCreight, E. M. A space-economical suffix tree construction algorithm. *J. Assoc. Comput. Mach. 23*, 2 (Apr. 1976), pp. 262–272.

[21] Niederreiter, H., Some computable complexity measures for binary sequences, In *Sequences and Their Applications*, Eds. C. Ding, T. Hellseth and H. Niederreiter Springer Verlag, pp. 67-78, 1999.

[22] Pevzner, P. A., Borodovsky, M. Y., and Mironov, A. A. Linguistics of nucleotides sequences II: Stationary words in genetic texts and the zonal structure of DNA. *J. Biomol. Struct. Dynamics 6* (1989), pp. 1027–1038.

[23] Rahmann, S., and Rivals, E. On the distribution of the number of missing words in random texts. *Combinatorics, Probability and Computing* (2003), 12:73-87.

[24] Reinert, G., Schbath, S., and Waterman, M. S. Probabilistic and statistical properties of words: An overview. *J. Comput. Bio. 7* (2000), pp. 1–46.

[25] Shallit, J. On the maximum number of distinct factors in a binary string. *Graphs and Combinatorics 9* (1993), pp. 197–200.

[26] Szpankowski, W. A generalized suffix tree and its (un)expected asymptotic behaviors, *SIAM J. Computing*, 22, (1993), pp. 1176–1198.

[27] Szpankowski, W. *Average Case Analysis of Algorithms on Sequences.* John Wiley & Sons, New York, 2001.

[28] Tomescu, I. On the number of occurrences of all short factors in almost all words. *Theoretical Computer Science* 290:3 (2003), pp. 2031-2035.

[29] Tomescu, I. On Words Containing all Short Subwords. *Theoretical Computer Science* 197:1-2 (1998), pp. 235-240.

[30] Troyanskaya, O. G., Arbell, O., Koren, Y., Landau, G. M., and Bolshoy, A. Sequence complexity profiles of prokaryotic genomic sequences: A fast algorithm for calculating linguistic complexity. *Bioinformatics 18*, 5 (2002), pp. 679–688.

[31] Ukkonen, E. On-line construction of suffix trees. *Algorithmica 14*, 3 (1995), pp. 249–260.

[32] Wang, M.-W. and Shallit, J. On minimal words with given subword complexity. *Electronic Journal of Combinatorics*, 5(1) (1998), #R35.

[33] Weiner, P. Linear pattern matching algorithm. In *Proceedings of the 14th Annual IEEE Symposium on Switching and Automata Theory* (Washington, DC, 1973), pp. 1–11.