

LINEAR GLOBAL DETECTORS OF REDUNDANT AND RARE SUBSTRINGS

(Extended Abstract)

Alberto Apostolico*

Purdue Univ. & Univ. of Padova

Mary Ellen Bock†

Purdue University

Stefano Lonardi‡

Purdue Univ. & Univ. of Padova

1 Introduction

The identification of strings that are, by some measure, redundant or rare in the context of larger sequences is an implicit goal of any data compression method. In many other applications, such strings are routinely sought in order to unveil structure, to infer minimal or compact descriptions, and for purposes of feature extraction and classification. In bio-sequence analysis, for instance, such unusual strings have been variously implicated in biological functions and mechanisms (refer, e.g., to [van-98], [LMS-96], [S-97] and references therein). In this and other domains, tables for storing the number of occurrences in a string of substrings of (or up to) a given length need to be computed and stored. In Data Compression, tables of this nature find use in, and in fact subtend, the development of context trees (cf., e.g., [Jes-98], [Yo-98] and references therein). In the straightforward approach to searching for unusual substrings, the words (up to a certain length) are enumerated more or less exhaustively and individually checked in terms of observed and expected frequencies, variances, and scores of discrepancy and significance thereof. As is well known, clever methods are available to compute and organize the counts of occurrences of *all* substrings of a

*Department of Computer Sciences, Purdue University, Computer Sciences Building, West Lafayette, IN 47907, USA and Dipartimento di Elettronica e Informatica, Università di Padova, Padova, Italy. axa@cs.purdue.edu. Work supported in part by NSF Grant CCR-9700276, by NATO Grant CRG 900293, by British Engineering and Physical Sciences Research Council Grant GR/L19362, and by the Italian Ministry of Research.

†Department of Statistics, Purdue University, Math. Sciences Building, West Lafayette, IN 47907, USA. mbock@stat.purdue.edu.

‡Department of Computer Sciences, Purdue University, Computer Sciences Building, West Lafayette, IN 47907, USA. stel@cs.purdue.edu.

given string. The corresponding tables take up the tree-like structure of a special kind of digital search index or *trie* (see, e.g., [Mc-76], [Ap-85], [AP-96]). These trees have found use in numerous applications [Ap-85], and represent a familiar tool in Data Compression and Computational Molecular Biology (see, e.g., [Wa-95]). We assume familiarity of the reader with these trees, their basic properties and uses.

Once the index itself is built, its entries can be annotated with the expected values and variances that may be associated with them under one or more probabilistic models. One such process of annotation is addressed in [ABX-97], where we show that assuming a random source emitting symbols from a known alphabet independently and according to a given distribution, mean, variance and some of the adopted measures of significance can be assigned to all substrings of a string of n symbols in optimal $O(n^2)$ time.

We show here that under several accepted measures of deviation from expected frequency, the candidates over- or underrepresented words are restricted to the $O(n)$ words that end at internal nodes of a compact suffix tree, as opposed to the $\Theta(n^2)$ possible substrings. This surprising fact is a consequence of properties in the form that if a word that ends in the middle of an arc is, say, overrepresented, then its extension to the nearest node of the tree is even more so. Based on this, we design global linear detectors of favored and unfavored words for our probabilistic framework, and display the results of some preliminary that apply our constructions to the analysis of genomic sequences.

2 Preliminaries

Given an alphabet Σ , we use Σ^+ to denote the free semigroup generated by Σ , and set $\Sigma^* = \Sigma^+ \cup \{\lambda\}$, where λ is the empty word. An element of Σ^+ is called a *string* or *sequence* or *word*, and is denoted by one of the letters s, u, v, w, x, y and z . The same letters, upper case, are used to denote *random* strings. We write $x = x_1x_2\dots x_n$ when giving the symbols of x explicitly. The number of symbols that form a string w is called the *length* of w and denoted by $|w|$. If $x = vw y$, then w is a *substring* of x and the integer $1 + |v|$ is its (*starting*) *position* in x . Let $X = X_1X_2\dots X_n$ be a *textstring* produced randomly by a *source* that emits symbols from alphabet Σ independently and according to a given probability distribution. We use x to denote an observation of X . Let $y = y_1y_2\dots y_m$ ($m < (n + 1)/2$) be an arbitrary but fixed *pattern* string on Σ . For $i \in \{1, 2, \dots, n - m + 1\}$, define $Z_i|y$ to be 1 if y occurs in X starting at position i and 0 otherwise. We are interested in the the expected value and variance of $Z|y$, the total number of occurrences of y in X :

$$Z|y = \sum_{i=1}^{n-m+1} Z_i|y.$$

It is immediate that $E[Z|y] = (n - m + 1)\hat{p}$, where, with p_i denoting the probability for any given k that $X_k = y_i$, $\hat{p} = \prod_{i=1}^m p_i$.

For any symbol a in Σ , computing the expected value $Z|ya$ from \hat{p} and the probability of a is trivially done in constant time. Thus, the expected values associated

with all prefixes of a string can be computed in linear time. Under the stated assumption¹ that $m \leq (n + 1)/2$, it is possible to express the variance in the following form [ABX-97]:

$$\text{Var}(Z|y) = (n-m+1)\hat{p}(1-\hat{p}) - \hat{p}^2(2n-3m+2)(m-1) + 2\hat{p} \sum_{l=1}^{s_m} (n-m+1-d_l) \prod_{j=m-d_l+1}^m p_j$$

where the d_l 's are the *periods* of y that satisfy $1 \leq d_1 < d_2 < \dots < d_{s_m} \leq \min(m-1, n-m)$. Recall that a string z has a period w if z is a prefix of w^k for some integer k . A string may have several periods. Sometimes the word “period” is also used to refer to the length of a period. The shortest period (or period length) of a string z is called *the* period of z . Clearly, a string is always a period of itself. This period is called the trivial period. We say that a non-empty string w is a *border* of a string z if z starts and ends with an occurrence of w . That is, $z = uw$ and $z = vw$ for some possibly empty strings u and v . Clearly, a string is always a border of itself. This border is called the trivial border. The notions of period and border are complementary.

Fact 2.1 *A string x of length k has period of length q , such that $q < k$, if and only if it has a non-trivial border of length $k - q$.*

Suppose that we wanted to compute the variance of $Z|y$ for all substrings y of x in accordance to the formula above. Applying the formula from scratch to each substring would require time $\Theta(|x|^3)$, since the number of possible distinct words appearing as substrings of x may be quadratic in $|x|$. In [ABX-97], it is proved that our variance can be computed for all prefixes of a string y in overall time $O(|y|)$, which brings the overall cost for string x down to $O(|x|^2)$. The key to this is a recurrence that speeds up computation of the term

$$B(m) = \sum_{l=1}^{s_m} (n-m+1-d_l) \prod_{j=m-d_l+1}^m p_j.$$

In this expression, $B(m)$ refers to the prefix $y_1 y_2 \dots y_m$ of some string y , $\mathcal{S}(m) = \{b_{l,m}\}_{l=1}^{s_m}$ is the set of borders $\text{bord}(m)$ is the longest border of $y_1 y_2 \dots y_m$. In other words, the computation of B depends on the structure of all periods d_l of $y_1 y_2 \dots y_m$ that are less than or equal to $\min(m-1, n-m)$. By adaptation of the “failure function” computations involved in classical string searching (see, e.g., [AG-97]), it is possible to derive $B(m)$ quickly from knowledge of $\text{bord}(m)$ and of the previously computed values $B(1), B(2), \dots, B(m-1)$. Specifically, letting the border associated with period d_l at position m to be

$$b_{l,m} = m - d_l,$$

the following expression of $B(m)$ holds [ABX-97]:

¹We concentrate on this assumption for practical reasons and brevity only; the treatment of the case $m > (n + 1)/2$ is quite similar.

$$B(m) = (n - 2m + 1 + \text{bord}(m)) \prod_{j=\text{bord}(m)+1}^m p_j + 2(\text{bord}(m) - m) \sum_{l=1}^{s_{\text{bord}(m)}} \prod_{j=b_l, \text{bord}(m)+1}^m p_j \\ + \left(\prod_{j=\text{bord}(m)+1}^m p_j \right) B(\text{bord}(m)),$$

where the fact that $B(m) = 0$ for $\text{bord}(m) \leq 0$ yields the initial conditions. Note that each product of probabilities can be extracted in constant time from a precomputed table containing the products of the probabilities of all consecutive prefixes of x . From knowledge of $n, m, \text{bord}(m)$ and these prefix probability products, the first term of $B(m)$ is computed in constant time. Except for $(\text{bord}(m) - m)$, the second term is essentially a sum of probability products taken over all distinct borders of $y_1 y_2 \dots y_m$. Thus, given such a sum and $B(\text{bord}(m))$ at this point enables one to compute $B(m)$ whence also the variance, in constant time. Maintaining knowledge of the value of such sums during the computation of longest borders is easy, since the value of the sum

$$T(m) = \sum_{l=1}^{s_{\text{bord}(m)}} \prod_{j=b_l, \text{bord}(m)+1}^m p_j$$

obeys the recurrence:

$$T(m) = T(\text{bord}(m)) \cdot \prod_{j=\text{bord}(m)+1}^m p_j + \prod_{j=\text{bord}(\text{bord}(m))+1}^m p_j,$$

with $T(m) = 0$ for $\text{bord}(\text{bord}(m)) \leq 0$. In conclusion, the following holds.

Theorem 2.2 *Under the independently distributed source model, the mean and variances of all prefixes of a string can be computed in time and space linear in the length of that string.*

Application of this treatment to every suffix of a string yields the mean and variance of all substrings in overall optimal quadratic time. The table of Figure 1 compares the costs of computing $B(m)$ with both methods for *Fibonacci words* of increasing lengths. Fibonacci words are defined by a recurrence in the form: $F_{i+1} = F_i F_{i-1}$ for $i \geq 1$, with $F_0 = b$ and $F_1 = a$, and exhibit a rich repetitive structure.

The last column of the table compares, for lengths in a realistic range, the values of the variance obtained with and without consideration of overlaps, that is, the absolute error incurred when overlaps are neglected and the computation of the variance is truncated after the term $\hat{V}ar(Z|y) = (n - m + 1)\hat{p}(1 - \hat{p})$. As it turns out, relative errors are found to increase with the length of y , while absolute errors attain their maxima for relatively short values of $|y|$.

3 Linear Global Detectors of Unusual Words

We have seen that mean, variance and some related scores of significance can be computed for each of the $O(n^2)$ distinct substrings of a string of n symbols in optimal

i	$ F_i $	Direct (secs)	[ABX] (secs)	$\max_y \{ \ Var(Z y) - \hat{Var}(Z y)\ \}$
8	55	0.02	0.02	9.3307557400
10	144	0.17	0.8	25.36759
12	377	1.62	0.51	67.3815
14	987	14.44	3.35	177.38
16	2584	132.5	23.43	465.6
18	6765	1150	163.2	1220

Figure 1: Number of seconds (averaged over 100 runs) for computing the table of $B(m)$ $m = 1, 2, \dots, |F_i|$) for some initial Fibonacci words; the last column displays errors resulting when \hat{Var} is used to approximate Var .

overall $O(n^2)$ time. In this section, we show that the values and scores stored only at the $O(n)$ leaves and branching internal nodes of the suffix tree T_x of x suffice in most cases. These are the only nodes in a compact suffix tree, and there are less than $2n = 2|x|$ such nodes. A string ending precisely at a leaf or branching node of T_x will be said to have a *locus* in T_x . One key element in this construction is offered by the following well known fact [Mc-76].

Fact 3.1 *If $w = av$, $a \in \Sigma$, has a locus in T_x , then so does v .*

To exploit this fact, *suffix links* are maintained in the tree that lead from the locus of each string av to the locus of its suffix v . Here we are interested in Fact 3.1 for a different reason, namely, because the incremental computation of our variance formulae along the suffix links of the tree rather than the original arcs will achieve the claimed overall linear-time weighting of the entire tree. We begin by observing that the frequency counter associated with the locus of a string in T_x reports its correct frequency even when the string terminates in the middle of an arc. This important “right-context” property is conveniently reformulated as follows.

Fact 3.2 *Let the substrings of x be partitioned into equivalence classes C_1, C_2, \dots, C_k , so that the substrings in C_i ($i = 1, 2, \dots, k$) occur precisely at the same positions in x . Then $k < 2n$.*

In the string *abaababaabaababaababa*, for instance, $\{ab, aba\}$ forms one such C-class and so does $\{abaa, abaab, abaaba\}$. Fact 3.2 suggests that we might only need to look among $O(n)$ substrings of x in order to find its most unusual words. The following considerations show that under our probabilistic assumptions this statement can be made even more precise. First, we have to choose among a number of measures set up to assess the departure of observed from expected behavior and its statistical significance. (Refer, e.g., to [LMS-96, S-97] for a recent discussion and references.) Some such measures are computationally easy, others quite imposing. Perhaps the naivest possible measure is the difference: $\delta_w = f_w - (n - |w| + 1)\hat{p}$, where \hat{p}

is the product of symbol probabilities for w and $Z|w$ takes the value f_w . Let us say that an over-represented (respectively, under-represented) word w in some class C is δ -significant if no extension (respectively, prefix) of w in C achieves at least the same value of $|\delta|$.

Theorem 3.3 *The only over-represented δ -significant words in x are the $O(n)$ ones that have a locus in T_x . The only under-represented δ -significant words are the ones that represent one unit-symbol extensions of words that have a locus in T_x .*

Proof: We prove first that no over-represented δ -significant word of x may end in the middle of an arc of T_x . Specifically, any over-represented δ -significant word in x has a proper locus in T_x . Assume for a contradiction that w is a δ -significant over-represented word of x ending in the middle of an arc of T_x . Let $z = wv$ be the shortest extension of w with a defined locus in T_x , and let \hat{q} be the probability associated with v . Then, $\delta_z = f_z - (n - |z| + 1)\hat{p}\hat{q} = f_z - (n - |w| - |v| + 1)\hat{p}\hat{q}$. But we have, by construction, that $f_z = f_w$. Moreover, $\hat{p}\hat{q} < \hat{p}$, and $(n - |w| - |v| + 1) < (n - |w| + 1)$. Thus, $\delta_z > \delta_w$. For this specification of δ , it is easy to prove symmetrically that the only candidates for δ -significant under-represented words are the words ending precisely one symbol past a node of T_x . \square

We now consider more sophisticated "measures of surprise" by giving a new definition of δ of the more general form: $\delta_w = (f_w - E_w)/N_w$, where: (a) f_w is the frequency or count of the number of times that the word w appears in the text; (b) E_w is the typical or average nonnegative value for f_w (and E is often chosen to be the expected value of the count); (c) N_w is a nonnegative normalizing factor for the difference. (The N is often chosen to be the standard deviation for the count.)

Once again it is assumed that δ lies on a scale where positive and negative values which are large in absolute value correspond to highly over- and under-represented words, respectively, and thus are "surprising". We give three assumptions that insure that the theorem above is also true for this new definition of δ . Let $w^+ = wv$ be a word which is an extension of the word w , where v is another nonempty symbol or string. First we assume that the "typical" value E always satisfies: $E_{w^+} \leq E_w$. This says that the typical or average count for the number of occurrences of an extended word is not greater than that of the original word. (This is automatically true if E is an expectation.)

The next assumption concerns under-represented words. Of course, if a word w or its extension w^+ never appears, then $f_w = f_{w^+} = 0$. We would want the corresponding measure of surprise δ to be stronger for a short word not appearing than for a longer word not appearing, i.e. we would want the two negative δ values to satisfy: $\delta_w \leq \delta_{w^+}$. Thus δ_w is larger in absolute value (and more surprising) than δ_{w^+} when neither word appears. This is the rationale for the following assumption: $E_{w^+}/N_{w^+} \leq E_w/N_w$, in the case that both N 's are positive.

The third assumption insures that for over-represented words (i.e. δ positive), it is more surprising to see a longer word over-represented than a shorter word. We assume: $N_{w^+} \leq N_w$.

The import of all these assumptions is that whenever we have $f_w = f_{w+}$, then $\delta_w \leq \delta_{w+}$. This implies that we can confine attention to the nodes of a tree when we search for extremely over-represented words since the largest positive values of δ will occur there rather than in the middle of an arc. Likewise it implies that the most under-represented words occur at unit symbol extensions of the nodes.

Most of the widely used scores meet the above assumptions. For instance, consider as a possible specification of δ the following ζ -score (cf. [LMS-96], [Wa-95]), in which we are computing the variance neglecting all terms due to overlaps:

$$\zeta_w = \frac{f_w - (n - |w| + 1)\hat{p}}{\sqrt{(n - |w| + 1)\hat{p}(1 - \hat{p})}}$$

The inferred choices of E and N automatically satisfy the first two assumptions above. The concave product $\hat{p}(1 - \hat{p})$ which appears in the denominator term N_w of the above fraction is maximum for $\hat{p} = 1/2$, so that, under the realistic assumption that $\hat{p} \leq 1/2$, the third assumption is satisfied. Thus ζ increases with decreasing \hat{p} along an arc of T_x .

In conclusion, once one is restricted to the branching nodes of T_x or their one-symbol extensions, all typical count values E (usually expectation) and their normalizing factors N (usually standard deviation) and other measures discussed earlier in this paper can be computed and assigned to these nodes and their one-symbol extensions in overall linear time and space. The key to this is simply to perform our incremental computations of, say, standard deviations along the suffix links of the tree, instead of spelling out one by one the individual symbols found along each original arc. The details are easy and are left to the reader. The following Theorem summarizes our discussion.

Theorem 3.4 *The set of all δ -significant subwords of a string x over a finite alphabet can be detected in linear time and space.*

4 Software and Experiments

The algorithms and the data structures described above have been coded in C++ using the Standard Template Library (STL) collection of containers and generic functions [MS-94]. Overall, the implementation consists of circa 2,500 lines of code. Besides outputting information in more or less customary tabular forms, our programs generate source files capable of driving some graph drawing programs such as DOT [GKNV-93] or DAVINCI [FW-95] while allowing the user to dynamically set and change visual parameters such as font size and color. The overall facility was dubbed VERBUMCULUS in an allusion to its visualization features. The program is, however, a rather extensive analysis tool that collects the statistics of a given text file in one or more suffix trees, annotates the nodes of the tree with the expectations and variances, etc.

Example visual outputs of VERBUMCULUS are displayed in the colored figures which are found at the end of the paper. The whole word terminating at each node is

printed in correspondence with that node and with a font size that is proportional to its score; under-represented words that appear in the string are printed in red italics, black is reserved for over-represented words. To save space, words that never occur in the string are not displayed at all, and the tree is pruned at the bottom. The first figure shows an application to the first 512 bps of the mitochondrial DNA of the yeast under score ζ .

The last figure is related to computations presented in [LMS-96] in the context of a comparative analysis of various statistical measures of over- and under-representation of words. It should be clarified that here we are only interested in the issue of effective detection of words that are unusual according to some pre-determined score or measure the intrinsic merits of which are not part of our concern. The histogram of Figure 3 represents our own reproduction of computations and tables originally presented in [LMS-96], and related to occurrence counts of few extremal 4- and 5-mers in some genomes. Such occurrences are counted in a sliding window of length approximately 0.5% of the genomes themselves. We use for a concrete example the counts of to the occurrences of *GAGGA* in HSV1 (circa 150k bps). Peaks are clearly visible in the table, thereby denouncing local departures from average in the behavior of that word. One such peaks are found, e.g., in correspondence with some initial window position in the chart. Note that in order to find, say, which 5-mers occur with unusual frequencies (by whatever measure), one would have to first generate and count individually each 5-mer in this way. Next, to obtain the same kind of information on 6-mers, 7-mers or 8-mers, the entire process would have to be repeated. Thus, every word is processed separately, and the count of a specific word in a given window is not directly comparable to the counts of other words, both of the same as well as different length, appearing in that window.

Figure 4 displays the tree (suitably pruned at the bottom) produced at the same peak-window in the corresponding histograms. Not surprisingly, VERBUMCULUS isolates the word found in the corresponding table. However, in this and all other cases tested, the program exposes as unusual a family of words related to the single one used to generate the histogram, and sometimes other words as well. The words in the family are typically longer than the one of the histogram, and each actually represents an equally, if not more surprising, context string. Finally, VERBUMCULUS finds that the specific words of the histograms are, with their detected extensions, over-represented with respect to the entire population of words of every length within a same window, and not only with respect to their individual expected frequency.

Figure 2: VERBUMCULUS + DOT on the first 512 bps of the mitochondrial DNA of the yeast *S. cerevisiae*, under score $\zeta_w = (f_w - (n - |w| + 1)\hat{p}) / \sqrt{(n - |w| + 1)\hat{p}(1 - \hat{p})}$

Figure 3: GAGGA count in a sliding window of size 800 bps of the whole HSV1 genome

Figure 4: VERBUMCULUS + DOT on window 0 (first 800 bps) of HSV1, under score $\zeta_w = (f_w - (n - |w| + 1)\hat{p}) / \sqrt{(n - |w| + 1)\hat{p}(1 - \hat{p})}$ (frequencies of individual symbols are computed over the whole genome)

References

- Ap-85 Apostolico, A., The Myriad Virtues of Suffix Trees, *Combinatorial Algorithms on Words*, (A. Apostolico and Z. Galil, eds.), Springer-Verlag Nato ASI Series F, Vol. 12, 85–96 (1985).
- ABX-97 Apostolico, A., M.E. Bock and X. Xuyan, Annotated Statistical Indices for Sequence Analysis, (invited paper) Proceedings of *Compression and Complexity of SEQUENCES 97* (B. Carpentieri, A. De Santis, U. Vaccaro and J. Storer, eds.) IEEE Computer Society Press, pp. 215–229 (1998).
- AG-97 Apostolico, A. and Z. Galil (eds.), *Pattern Matching Algorithms*, Oxford University Press (1997).
- AP-96 Apostolico, A. and F.P. Preparata, Data Structures and Algorithms for the String Statistics Problem, *Algorithmica*, **15**, 481–494 (1996).
- AS-92 Apostolico, A. and W. Szpankowski, Self-Alignments in Words and Their Applications, *Journal of Algorithms*, **13**, 446–467 (1992).
- BBT-86 Brendel, V., J.S. Beckman and E.N. Trifonov, Linguistics of Nucleotide Sequences: Morphology and Comparison of Vocabularies, *Journal of Biomolecular Structure and Dynamics*, **4**, 1, 11–21 (1986).
- FW-95 Frohlich, M., and M. Werner, Demonstration of the Interactive Graph Visualization System Davinci, In *Proceedings of DIMACS Workshop on Graph Drawing '94, Princeton (USA) 1994, LNCS No. 894* (1995), R. Tamassia and I. Tollis, Eds., Springer Verlag.
- GKNV-93 Gansner, E. R., Koutsofios, E., North, S., and Vo, K.-P., A Technique for Drawing Directed Graphs. *IEEE Trans. Software Eng.* **19**, **3**, 214–230 (1993).
- van-98 van Helden, J., Andr e, B., and Collado-Vides, J., Extracting Regulatory Sites from the Upstream Region of Yeast Genes by Computational Analysis of Oligonucleotide Frequencies. *J. Mol.Biol.*, **281**, 827–842 (1998).
- Jes-98 Jesper Larson, N., Context Trees of Block Sorting Compression, *Proceedings of DCC Data Compression Conference*, pages 202–211. IEEE Computer Society Press (1998).
- LMS-96 Leung, M.Y., G.M. Marsh and T.P. Speed, Over and Underrepresentation of Short DNA Words in Herpesvirus Genomes, *Journal of Computational Biology* **3**, 3, 345 – 360 (1996).
- Mc-76 McCreight, E.M., A Space Economical Suffix Tree Construction Algorithm, *Jour. of the ACM*, **25**, 262–272 (1976).
- MS-94 Musser, D. R., and A. A. Stepanov, Algorithm-oriented Generic Libraries, *Software-Practice and Experience* **24**, 7, 623–642 (1984).
- S-97 Schbath, S., An Efficient Statistic to Detect Over- and Under-represented Words, *J. Comp. Biol.* **4**, 2, 189–192 (1997).
- Wa-95 Waterman, M.S., *Introduction to Computational Biology*, Chapman & Hall (1995).
- Yo-98 Yokoo, H., Context Tables: A Tool for Describing Text Compression Algorithms, *Proceedings of DCC Data Compression Conference*, pages 299–308. IEEE Computer Society Press (1998).