Communications in Information and Systems
Vol. 9, No. 3, pp. 235-252, 2009

ASK NOT WHAT STRINGOLOGY CAN DO FOR YOU: ADVANCES IN PATTERN MATCHING DRIVEN BY COMPUTATIONAL BIOLOGY[*]

ALBERTO APOSTOLICO[†]

Abstract. Molecular biology has posed a number of fascinating and sometimes daunting computational problems, which came naturally expressed in its native language of character strings. Through the years, some such problems have found elegant and even useful solutions in response to the needs that originally motivated them. What is perhaps even more remarkable, several of the ideas inspired by computational molecular biology have found application in remote and diverse domains, so that it may be argued that molecular biology did more for computing than the latter did for it. As a modest tribute, this paper reviews a small sample of these cases drawing from the personal exposure of the author.

Keywords: Computational biology, Design and analysis of algorithms; Pattern matching, Pattern discovery; Motif, Sequence alignment, Waka, Freakanomics, Data analysis; Data compression.

1. Introduction. With its basic lexicon of characters, sequences and trees, molecular biology has posed a number of interesting computational problems, some more challenging than others, of which a few have also found elegant and even useful solutions in the domain which originally inspired them. Among the many ideas and paradigms brought up in the course of the development of computational molecular biology [47], and the numerous implements developed in response to its needs, several have found use in remote applications, so that it may be argued that by way of inspiration and challenge molecular biology ultimately did more for computing than vice versa.

That computational problems arising, e.g., in molecular sequence analysis might have an impact in distant areas is not new. Nuances of the phenomenon resonate across the milestone “time warps” volume edited by D. Sankoff and J. Kruskal in the early Eighties [41], which listed applications of string editing ranging from tectonics to the study of songs (see also [36]) and bird signals. Therefore, the point of the present paper is not to show that such a fascinating interplay is possible, but only to display a few instances, taken somewhat arbitrarily from the direct exposure of this author.

It is only fair to say that some of the notions that permeate both computing and biosequence analysis go back to quite ancient times. Palindromic structures,

[*] Dedicated to Michael Waterman on the occasion of his 67th birthday.

[†] Università di Padova (Italy) & Georgia Institute of Technology (USA). Dipartimento di Ingegneria dell’ Informazione, Università di Padova, Padova, Italy *and* College of Computing, Georgia Institute of Technology, 801 Atlantic Drive, Atlanta, GA 30318, USA. Work Supported in part by the Italian Ministry of University and Research under the Bi-National Project FIRB RBIN04BYZ7, and by the Research Program of Georgia Tech. E-mail: axa@dei.unipd.it

for instance, come up in certain mysterious constructs such as the “Sator Square”, represented by the arrangement:

```
S A T O R
A R E P O
T E N E T
O P E R A
R O T A S
```

that reads the same top-to-bottom, bottom-to-top, left-to-right, and right-to-left. Since the earliest known appearance of the square was found in the ruins of Herculaneum, it is assumed that it originated in the Christian era, although the exact meaning and use of it, if any ever existed, are still fuzzy.

It is well known that some patterns in DNA and RNA encode for higher order structures, such as e.g. the secondary structure of RNA. And there are reasonably good methods to obtain the secondary structure from the sequence, driven by dynamic programming approaches to energy minimization. An alternative is to undertake a stem and loop analysis, which may be considered a variation of the theme of homology search. These approaches resemble the recognition of *palindromes* in a string, which are words that read the same forward and backward. But the palindromes of interest in biosequences are different in at least two respects: the copies are approximate, and they are copies up to base complementation. For instance, ATCGCGAT would be an exact palindrome under this convention. In computer science, palindromic forms are especially relevant to formal languages in so far as they epitomize balanced forms akin to parenthesis systems and the likes that are paradigmatic of context-free languages. Whereas the latter are naturally associated to the *stack* fixture of a *pushdown automaton*, the study of more conventional algorithms to find all palindromes of a string or to recognize special classes of languages called $PALSTAR$ [28] has many fascinating aspects and some real jewels of algorithmics, such as Manacher's linear time detection of all maximal palindromes [34].

Among remote cases of premonition we find the “semantic” Hamming distance proposed in the following excerpt from Vanity Fair of March 29, 1879, that I believe to have been brought to my attention by Mike Waterman.

> “A new puzzle... The rules of the puzzle are simple enough. Two words are proposed, of the same length; and the Puzzle consists in linking these together by interposing other words, each of which shall differ from the next word in one letter only"

Amidst the initial “edit scripts” that Lewis Carroll used to trigger the game and those contributed by the readership one could find small pearls such as:

$$HEAD \rightarrow Heal \rightarrow Teal \rightarrow Tell \rightarrow Tall \rightarrow TAIL$$
$$TEARS \rightarrow Sears \rightarrow Stars \rightarrow Stare \rightarrow Stale \rightarrow Stile \rightarrow SMILE$$
$$TREE \rightarrow Free \rightarrow Flee \rightarrow Fled \rightarrow Feed \rightarrow Weed \rightarrow Weld \rightarrow Wold \rightarrow WOOD$$

$$GRASS \rightarrow Crass \rightarrow Cress \rightarrow Tress \rightarrow Trees \rightarrow Frees \rightarrow Freed \rightarrow Greed \rightarrow GREEN$$
$$ONE \rightarrow Owe \rightarrow Ewe \rightarrow Eye \rightarrow Dye \rightarrow Doe \rightarrow Toe \rightarrow Too \rightarrow TWO$$
$$APE \rightarrow Are \rightarrow Ere \rightarrow Err \rightarrow Ear \rightarrow Mar \rightarrow MAN$$

This problem becomes clearly vacuous under standard Hamming distance [25], in which words are not required to have a meaning. But distances defined in terms of edit script allowing for insertion and deletion of characters in addition to the substitution (sometimes called Levenstein distances, in an allusion to their lineage from error correcting codes [31]) would later dominate the scene in computational molecular biology. I am not sure that Roberto Benigni, the Oscar winning director of "Life is Beautiful" ever knew of Lewis Caroll's puzzle, but during a weekend gathering of friends at Umberto Eco's he introduced a similar variation on semantic Hamming consisting of a riddle that found the answer into a movie title, once just one letter was changed. For instance, the solution to the riddle: "lack of servants" (*penuria di domestici*, in Italian) was Almodovar's "Women on the brink of a nervous breakdown", since *donne sull'orlo di una crisi di* n*ervi* could be changed into *donne sull'orlo di una crisi di* s*ervi.* Also in this case, both originators and subsequent external contributors amassed a lot of additional examples, enough material for Stefano Bartezaghi to compile an entire book, one line at a time [15].

To a computer or information scientist it seems odd that genetic information needs anything more than a binary alphabet at the core. Why wasn't it expedient for genomes to evolve on two rather than four characters? Perhaps one strong argument against the binary alphabet is posed by repetitive DNA. It is estimated that more than 10% of the human genome is made of *tandem repeats*, which are two or more contiguous occurrences of exact or approximate copies of a string. They come from relatively little understood mutational transformations called *tandem duplication.* The latter are originally exact duplications but they degenerate into approximate ones by subsequent mutations.

It is possible to divide tandem repeats into three families, depending on location and span. Repeats in centromeric or telomeric regions, called *satellites*, have spans up to one million bases and replicas from 5 to few hundreds base pairs. The other families are called *mini* and *micro* satellites, respectively. The first ones have replicas of about 15 bases spanning in the hundreds to few thousands, the second have short replicas of 2-5 bases spanning in few hundreds.

Tandem repeats are believed to influence gene regulation and act as protein binding sites, thereby affecting DNA function and expression. They are believed to be implicated into inherited human diseases called *trinucleotide repeat* diseases, which include Huntington's disease, Friedreich's ataxia, myotonic dystrophy, spinal and bulbar muscular atrophy, among others (see [16] and references therein). Although the functional role of satellites is still not understood, they tend to be highly polymorphic, which makes them useful as genetic markers. For all these reasons, it is of interest to develop efficient methods for the detection of exact and approximate repeats in sequences.

At the beginning of last century Axel Thue posed the problem of whether or not it would be possible to write indefinitely long sequences over some alphabet without ever repeating the same substring twice in a row [45, 46]. It is easy to see that this becomes quickly impossible with a binary alphabet, say, $\{a, b\}$: starting w.l.o.g. with an a, forces to follow up with a b; now we must have an a again, lest the pair bb be created - at this point, however, it is impossible to preserve the property, as we must either create $abab$ or $abaa$, both no longer square-free. One would conjecture that adding one character would make the longest possible square-free string only a few units longer. Thue proved instead that with three characters or more it is possible to build indefinitely long strings without any square. He established his result by providing a square-free morphism, i.e., a rewriting rule by which square-freeness is preserved. Thue's morphism started on abc and it consisted of the rewriting rules: $a \leftarrow acab, b \leftarrow acabcb, c \leftarrow acbcacb$, but many years later Sorin Istrail [27] gave the shortest morphism on a three character alphabet, namely: $a \leftarrow abc, b \leftarrow bc, c \leftarrow b$. In view of the above, it may be refreshing to know that squares and DNA tandem repeats are not *unavoidable regularities* (or a "fact of life", as one would be tempted to say) and hence their appearance serves some biological significance and perhaps purpose. Although with some exceptions (e.g., the $TATA$ box) squares and repetitions in genomic sequences occur mostly in approximate forms. At any rate, once it becomes clear that exact squares are avoidable it is natural to ask how many of them a string could pack and then how long would it take to find them.

Since there are $\Theta(n^2)$ ways to pick a substring (hence, a candidate root of a square) in a sequence, one might expect as many roots and then squares. In 1981, Max Crochemore [21] proved that there can be at most $O(n \log n)$ squares in a string, and it would take a particularly pathological one such as a *Fibonacci word* to accommodate this number. A Fibonacci word follows the standard recurrence except concatenation replaces addition, so as to produce the words:

$$F_0 = b; F_1 = a; F_2 = F_1F_0 = ab; F_3 = F_2F_1 = aba; F_4 = abaab; F_5 = abaababa$$

and so on. Algorithms have been produced ever since to find all occurrences of squares [11, 33, 21, 24], bound and detect all distinct roots of squares, as well as all maximal repetitions, that is, strings of maximum length in the form $u^k u'$ with u' a prefix of u [29], together with the companion problems of computing a table counting, for any substring of a string, the maximum number of its non-overlapping occurrences [12, 13, 20].

So much for mutual reverberations between biology and stringology. In the remainder of this paper, I will survey a few circumstances in which problems of molecular biology helped in solving problems in the outside world.

2. Freakanomics and Multiple Sequence Alignment. A recent bestseller [32] reports the intriguing story of a Chicago School that had been desperately falling in the ranking. The teachers of that School, whose compensation depended on the overall School performance, resorted to tampering with student examination records in an attempt at improving performance in one of the tests uniformly administered across the area district. Each record consisted of a string of a fixed number of alphanumeric characters, each marking the answer to one of the consecutive questions on the test. The scheme of the cheaters was to artificially change some of the wrong answers so as to make them right, but they only had limited time to do that –not enough to create personalized amended answer sheets.

In Multiple Sequence Alignment (refer to, e.g., [8]), we are given k sequences placed each on one of k consecutive rows and asked to align them by inserting a minimum number of spaces between consecutive characters in each sequence in such a way that the cost resulting from aligning spaces of non identical characters is minimum. There are several ways to choose the cost structure but we do not need belabor this point here. One of the ways to get around the inherent intractability of the problem is to set up heuristic methods capable of converging within reasonable time (e.g., some low polynomial in the size of the input) even though of course no such method can guarantee achieving optimality. Suppose that a sufficiently long good alignment existed among relatively well centered substrings of the sequences. We could use this "anchor" pattern to divide the problem in two halves of approximately the same size, and recursively solve the latter. One might expect $\log k$ stages to suffice in the average case, and the overall cost will depend on the cost of the anchor-seeking routine (incidentally, it has been argued occasionally that anchoring based on the user's expertise will actually improve the solution [37]). If we make the further simplification that the anchor must be a definite solid shared substring, then the problem reduces to the folowing:

Input: a family of k sequences $s_1, s_2, ..., s_k$ of total length n;
Output: for any integer $c \leq k$ a longest substring shared by c of the k strings.

Solving this problem gives a handle towards exposing the fraudulent scheme of the teachers. For, suppose that some c looks large in comparison to the length of the shared substring. Then one can argue that such a deep uniformity among test records could not have occurred by chance.

To solve the above problem by brute force we would have to extract substrings one by one and check for each one how many of the sequences contain it. There are many linear time algorithms for string searching (see, e.g., [7]), but applying any of them to the potentially

$$\sum_{j=1}^{k} |s_j|^2$$

candidate substrings yields a prohibitive $O(n^3)$ time bound.

Remarkably, the problem has a solution that takes time proportional to

$$\sum_{j=1}^{k} |s_j| = n$$

overall, i.e., including searches. The problem is known as the *set color problem* and the original solution is due to Hui [26]. It is based on the following non-intuitive property. Imagine a rooted tree with leaves variously colored (in our case, the colors would be precisely k, each corresponding to one of the sequences). We want to find, for each node, how many different colors are found in the subtree rooted at that node. Of course this is different from counting how many leaves are found in each subtree. It is noteworthy that D. Knuth had conjectured around 1972 that finding the longest string common to two textstrings could not be solved in time better than $O(n \log n)$ [28]. The conjecture was unusually short-lived, as a linear algorithm emerged a few months afterwards as a byproduct of Weiner's "repetition finder" [48].

In order to find surprising strings among the shared ones one needs to weight the color counts against their expected number. This is done by computing scores such as

$$z_1(w) = c(w) - E_c(w)$$
$$z_2(w) = \frac{c(w)}{E_c(w)}$$
$$z_3(w) = \frac{(c(w) - E_c(w))^2}{E_c(w)}$$

where $E_c(w)$ and $c(w)$ are, respectively, the expected and observed number of sequences that contain at least one occurrence of w. Given k sequences of respective sizes n_i for $i \in [1, k]$, a typical estimate for $E_c(w)$ is

$$E_c(w) = \sum_{i=1}^{k} \left(1 - e^{-\mathcal{E}_i(w)}\right)$$

where $\mathcal{E}_i(w)$ is the expected number of occurrences of w in the i-th sequence. An estimator of the true expectation is calculated after Stuckle *et al.* [43] by assuming a first order stationary Markov chain

$$\mathcal{E}_i(w) = \frac{f_i(w_{[1,2]}) f_i(w_{[2,3]}) \ldots f_i(w_{[m-1,m]})}{f_i(w_{[2]}) f_i(w_{[3]}) \ldots f_i(w_{[m-1]})}$$

where $f_i(w)$ is the observed number of occurences of w in the i-th sequence.

3. Motifs, Compression and the Wheel of Fortune. In Computational Molecular Biology and genomic studies a prominent role is played by patterns, sometimes also called *motifs* which include varieties of *gapped* strings such as those generated by some consensus or alignment. These patterns may be visualized as sequences of characters intermixed with gaps. They may be *rigid*, in which case a controlled number of gaps are admitted at predefined fixed positions, or *extensible*, when a sequence of gaps can now be stretched within prescribed bounds. Rigid motifs are thus strings over $\Sigma \cup \{\circ\}$, where Σ is the input alphabet and "$\circ$" is a wild card or *don't care* character matching any character from Σ. Allowing for variable length spacers in a motif makes it extensible. Spacers may be indicated by annotating the dot characters with the number of times it may be repeated. More specifically, an annotated "$\circ$" character is written as $\circ^{\alpha}$ where α is a set of positive integers $\{\alpha_1, \alpha_2, \ldots, \alpha_k\}$ or an interval $\alpha = [\alpha_l, \alpha_u]$, representing all integers between α_l and α_u including α_l and α_u. We may also define d, the maximum number of consecutive dots allowed in a string, and then the dash symbol "-" may be used in place of the annotated dot character, $\circ^{[1,d]}$ in the string, so that a string of the form $a \circ^{[1,d]} b$ will be simply written as a-b. In conclusion, a motif m is *extensible* if it contains at least one annotated dot. Unlike a rigid string an extensible string m may have multiple occurrences starting at a position of a sequence x. This complicates the probabilistic analysis of extensible motifs even for basic probabilistic models.

When looking for recurrent motifs, it seems natural to exclude from consideration those that could be further specified by some extension or by changing don't cares into additional solid characters without sacrifice of the corresponding occurrence count. Motifs that meet such condition have been also called *maximal* or *saturated*. An algebraic-flavored notion stronger than maximality, called *irredundancy*, was introduced for these motifs by L. Parida (see, e.g., [39]) and subsequently studied also by others. The idea is that, from the roster of all saturated rigid motifs, it is possible to extract a *base* of *irredundant* motifs with the property, that any other motif can be inferred both in terms of its pattern structure and list of occurrence by a suitable subset of irredundant motifs on the base. Unfortunately, the size of the base of motifs in a sequence can be exponential in the the size of the input [38]. However, when the minimum acceptable number of occurrences for a motif is just 2, then it is seen that the irredundant motifs come from the consensus patterns generated by the autocorrelations of the input. Furthermore, the size of the base is itself linear in the input [10] and for binary alphabets it can be actually built for an input string of n characters in $O(n^2 \log n)$ time incrementally for the entire set of suffixes of that string [14].

The departure of a pattern w from expectation is commonly measured by so-called z-scores, which take the form

$$z(w) = \frac{f(w) - E(w)}{N(w)}$$

where $f(w) > 0$ represents a frequency, $E(w) > 0$ an expectation and $N(w) > 0$ is the expected value of some function of w. For given z-score function, set of patterns $\mathcal{W}$, and real positive *threshold* T, patterns such that $z(w) > T$ or $z(w) < -T$ are considered to be *surprising*.

Under *i.i.d* assumptions, it is seen [3, 2] that for an input sequence of length n and a pattern w of length $m \leq (n+1)/2$ we have:

$$E(w) = (n - m + 1)p(w)$$

and

$$\begin{aligned} Var(w) &= E(w)(1 - p(w)) - p(w)^2(2n - 3m + 2)(m - 1) \\ &+ 2p(w) \sum_{l=1}^{s} (n - m + 1 - d_l) \prod_{j=m-d_l+1}^{m} p(w_{[j]}) \end{aligned}$$

where $p(w)$ is the probability of occurrence of w and $\{d_1, d_2, \ldots, d_s\}$ are the lengths of the periods of w. A string w has a *period* z if w is a prefix of z^k for some integer k. Alternatively, a string z is a period of a string w if $w = z^l v$ and v is a possibly empty prefix of z. Truncating $Var(w)$ after the first term yields $\hat{Var}(w) = E(w)(1 - p(w))$.

For scores of this kind, it is possible [3, 2] to confine the computation to a number of candidate surprising words which is linear in the length of the host sequence. Moreover, the set $\mathcal{W}$ of these candidates can be identified *a priori*, and their relationship to any other, e.g., over-represented word not in $\mathcal{W}$ is as follows (under-represented words obey a symmetric property). For any word w not in $\mathcal{W}$ such that $z(w) > t$, there is a word w' in $\mathcal{W}$ such that:

1. $w' = wv$ for some nonempty word v, i.e., the “neglected” word is embedded in a word of $\mathcal{W}$ as a prefix;
2. $z(w') > z(w)$, i.e., w' is at least as surprising as w.

Such a drastic limitation on the order of the number of candidates, as well as their identification, weighing and display are all inextricably interwoven reflections of a same combinatorial property, which has to do with the score being *monotone* within certain families of patterns. This property prescribes that if, say, w and an extension $w' = wv$ of w are nonempty substrings of the text x such that $f(w) = f(wv)$, then the score of w does not exceed that of w'. Under these conditions, w can be neglected as the surprise it conveys is subsumed by w'.

In order to convey the intuition behind these facts, let us define a *condensation* of a pattern u in some textstring x as any pattern v that may be obtained by inserting one or more extra solid characters in u while every starting position of an occurrence of u remains also a starting position of an occurrence of v. Then the following facts hold for extensible motifs, hence in particular for rigid motifs and solid words [5].

THEOREM 1. *Let v and u be extensible motifs under the i.i.d. model and let v be a condensation of u. Then, there is a probability $\hat{p} \leq 1$ such that $p_v = p_u \hat{p}$.*

THEOREM 2. *If $f(u) = f(v) > 0$, $N(v) < N(u)$, and $E(v)/N(v) \leq E(u)/N(u)$, then*

$$\frac{f(v) - E(v)}{N(v)} > \frac{f(u) - E(u)}{N(u)}$$

In the particular case of the *i.i.d* model this becomes

THEOREM 3. *Let u and v be motifs generated with respective probabilities p_u and $p_v = p_u \hat{p}$ according to an iid process. If $f(u) = f(v)$ and $p_u < 1/2$ then*

$$\frac{f(v) - E(v)}{\sqrt{E(v)(1 - p_v)}} > \frac{f(u) - E(u)}{\sqrt{E(u)(1 - p_u)}}$$

These facts identify the intervals of monotonicity within which z-score computation may be limited to precisely the set of saturated motifs. Then, a prudent combination of saturation conditions (expressed in terms of minimum number of don't cares compatible with a given list of of occurrences) and monotonicity of scores is seen to afford significant parsimony in the generation and testing of candidate over-represented rigid and extensible motifs.

The notion of saturated pattern finds a congenial *habitat* in data compression by textual substitution, where patterns recurring more or less frequently are replaced by pointers to a single common copy. In those contexts, it comes natural to impose that the pattern chosen for encoding satisfy conditions that prevent forfeiting information gratuitously. To begin with, once a motif is chosen it seems reasonable to exploit the set of its occurrences to the fullest, compatibly with self-overlaps. Likewise, it seems reasonable to exclude from consideration patterns that could be enriched in terms of solid characters without prejudice in the corresponding set of occurrences.

Data compression methods are partitioned traditionally into lossy and lossless. Typically, lossy compression is applied to images and more in general to signals susceptible to some degeneracy without lethal consequence. On the other hand, lossless compression is used in situations where fidelity is of the essence, which applies to high quality documents and perhaps most notably to textfiles. Lossy methods rest mostly on transform techniques whereby, for instance, cuts are applied in the frequency, rather than in the time domain of a signal. By contrast, lossless textual substitution methods are applied to the input in native form, and exploit its redundancy in terms of more or less repetitive segments or patterns.

When textual substitution is applied to digital documents such as fax, image or audio signal data, one could afford some loss of information in exchange for savings in time or space. In fact, even natural language can easily sustain some degrees of indeterminacy where it is left for the reader to fill in the gaps. The two versions below of the opening passage from the Book1 of the Calgary Corpus, for instance, are equally understandable by an average reader and yet when applied to the entire book the first variant requires 163,837 less bytes than the second one, out of 764,772.

> *DESCRIPTION OF FARMER OAK – AN INCIDENT*
> *When Farmer Oak smile., the corners .f his mouth spread till the. were within an unimportant distance .f his ears, his eye. were reduced to chinks, and ...erging wrinkleŮred round them, extending upon ... countenance li.e the rays in a rudimentary sketch of the rising sun. His Christian name was Gabriel, and on working days he was a young man of sound judgment, easy motions, proper dress, and ...eral good character. On Sundays, he was a man of misty views rather given to postponing, and .ampered by his best clothes and umbrella : upon ... whole, one who felt himself to occupy morally that ... middle space of Laodicean neutrality which ... between the Communion people of the parish and the drunken section, – that ... he went to church, but yawned privately by the t.ime the cong.egation reached the Nicene creed,- and thought of what there would be for dinner when he meant to be listening to the sermon.*

> *DESCRIPTION OF FARMER OAK – AN INCIDENT*
> *When Farmer Oak smiled, the corners of his mouth spread till they were within an unimportant distance of his ears, his eyes were reduced to chinks, and diverging wrinkles appeared round them, extending upon his countenance like the rays in a rudimentary sketch of the rising sun. His Christian name was Gabriel, and on working days he was a young man of sound judgment, easy motions, proper dress, and general good character. On Sundays he was a man of misty views, rather given to postponing, and hampered by his best clothes and umbrella : upon the whole, one who felt himself to occupy morally that vast middle space of Laodicean neutrality which lay between the Communion people of the parish and the drunken section, – that is, he went to church, but yawned privately by the time the congregation reached the Nicene creed,- and thought of what there would be for dinner when he meant to be listening to the sermon.*

In practice, the development of optimal lossless textual substitution methods is made hard by the circumstance that the majority of the schemes are NP-hard [42]. Obviously, this situation cannot improve with lossy ones. As an approximation, heuristic off-line methods of textual substitution can be based on greedy iterative selection as follows (see e.g., [9, 12]). At each iteration, a substring w of the text x is identified such that encoding a maximal set of non-overlapping instances of w

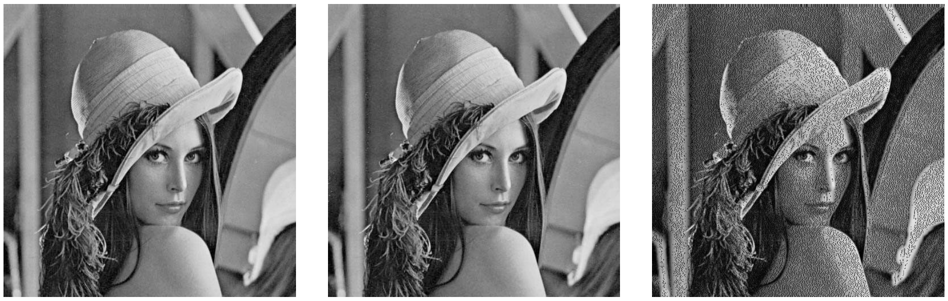

Fig. 1. *Lossy compression of “Lena” at 1/3 $'\circ'$/char density yields a gain of 25,33% over GZip. Interpolation (center figure) leaves 10,13% differences from original, unnoticeable to the naked eye.*

in x yields the highest possible contraction of x; this process is repeated on the contracted textstring, until substrings capable of producing contractions can no longer be found. This may be regarded as inferring a “straight line” grammar [22, 23, 30] by repeatedly finding the production or rule that, upon replacing each occurrence of the “definition” by the corresponding “nonterminal”, maximizes the reduction in size of the current textstring representation. Implementations of such greedy off-line strategies [9] compare favorably with other methods, particularly as applied to ensembles of otherwise hardly compressible inputs such as biosequences (sic!). They also appear to be the most predictable ones in terms of the achievable approximation to optimum descriptor sizes [30].

Off-line methods are advantageous in applications such as mass production of Cd-Roms, backup archiving, and any other scenario where extra time or parallel implementation may warrant the additional effort imposed by the encoding.

The idea of trading some amount of errors in reconstruction in exchange for increased compression is ingrained in Rate Distortion Theory [17, 18], and has been recently revived in a number of papers, mostly dealing with the design and analysis of lossy extensions of Lempel-Ziv on-line schemata.

For our lossy off-line schemata [6, 4, 5], we slightly expanded the notion of a *motif* by including additional parameters related to the density of solid characters, the maximum motif length and minimum allowed number of occurrences. Interpolation on images was carried out by averaging from the two solid pixels adjacent to each gap. The corresponding discrepancies from the original pixel values reach into 16% in terms of % *number* of inexact pixels, but was found to be only a few percentage points if the variation in *value* of those pixels was measured instead as a percentage of the affected pixels, and entirely negligible (a fraction of a percent) when averaged over all pixels, as displayed in the figure. These performances were impressive enough for IBM to file for more than one patent.

4. Markov Chains, Natural Languages and Protein Families. Probabilistic models of various classes of sources are developed in the context of coding and compression as well as in machine learning and classification. In the first domain, the repetitive structures of substrings are regarded as redundancies and sought to be removed. In the second, repeated subpatterns are unveiled as carriers of information and structure. Source modeling is made hard in practice by the fact that we do not know the source probabilities, the latter being actually rather fictitious entities or models. In fact, one pervasive problem is that of learning or estimating these probabilities from the observed strings. In summary, the problem is twofold. From an information theoretic standpoint, the question is how to define a notion of information relative to a class of sources. Once one such characterization is agreed upon, interesting algorithmic questions revolve around the computational cost inherent to the process of learning or estimating probabilities within that class.

Some popular probabilistic automata typically built in these contexts are subtended by uniform, fixed-memory Markov models. In practice, such automata tend to be unnecessarily bulky and computationally imposing both during their synthesis and use. One of the crucial parameters for a Markov model is its memory length L. The corresponding automaton must have one distinct state for each word of length L, so that the number of states grows exponentially with memory, and the automaton risks becoming rapidly bigger than the model sequence. For sequences in important families ranging from natural language, to speech, handwriting, and molecular sequence analysis, the "memory" exhibited decays exponentially fast with length. In other words, there is a maximum length L of the recent history of a sequence, above which the empirical probability distribution of next symbol given the last $L' > L$ symbols does not change appreciably. It is thus customary to model these sources by Markov chains of order L, this maximum useful memory length. Even so, the exponential growth in size by such automata makes them rapidly unpractical. In recet work by Ron *et al* [40], much more compact, tree-shaped variants of probabilistic automata (called *PST*s) are built which assume an underlying Markov process of variable memory length not exceeding some maximum L. These variants were subsequently adapted and applied successfully to learning and prediction of protein families. The probability distribution generated by these automata is equivalent to that of a Markov chain of order L, but the description of the automaton itself is much more succinct. The process of learning the automaton from a given training set S of sequences requires $\Theta(Ln^2)$ worst-case time, where n is the total length of the sequences in S and L is the length of a longest substring of S to be considered for a candidate state in the automaton. Once the automaton is built, predicting the likelihood of a query sequence of m characters may cost time $\Theta(m^2)$ in the worst case.

In recent work with G. Bejerano, we have introduced automata equivalent to PSTs but having the desirable properties that their construction and use takes linear time.

That is to say, in particular, that the size of the learned classifier does not exceed that of the observation upon which it is based. The crux in the improvement resides in speeding-up a test that asks, virtually on all substrings s of the source string, whether there is a symbol $\sigma \in \Sigma$ such that:

$$\frac{\tilde{P}(\sigma|s)}{\tilde{P}(\sigma|suffix(s))} \geq r \quad or \quad \frac{\tilde{P}(\sigma|s)}{\tilde{P}(\sigma|suffix(s))} \leq 1/r,$$

where $\tilde{P}$ denotes empirical probabilities or frequencies and $r \geq 1$ is a fixed parameter value. Essentially, it is possible to set up an algorithm to answer the collection of all those tests for all substrings of a textstring x in overall linear time and space. What seems counterintuitive is the fact that a string of n characters may contain $\Theta(n^2)$ distinct substrings, roughly corresponding to the number of ways in which the text can be cleaved at two different positions. As it turns out, however, substrings may be partitioned into $O(n)$ equivalence classes in such a way that for subwords s and $s' = s\sigma$ in a same class the empirical conditional probabilities in the form $\tilde{P}(\sigma|s)$ must be 1 and thus there is no need to compute it explicitly. This surprising property may be described as follows. Given two words x and y, let the *start-set* of y in x be the set of *occurrences* of y in x, *i.e.*, $pos_x(y) = \{i : y = x_i...x_j\}$ for some i and j, $1 \leq i \leq j \leq n$. Two strings y and z are equivalent on x if $pos_x(y) = pos_x(z)$. The equivalence relation instituted in this way is denoted by $\equiv_x$ and partitions the set of all strings over Σ into equivalence classes. In the string *abaababaabaababaababa*, for instance, $\{ab, aba\}$ forms one such class and so does $\{abaa, abaab, abaaba\}$. It is then possible to prove that the number of equivalence classes is bounded by $2n$. This is seen immediately by considering the suffix tree of the text string: all words ending in the middle of an arc are in the same class as the word ending at the closest downward branching node. Actually, branching nodes and leaves are in one-on-one correspondence with the classes in our equivalence relation. Since the tree has n leaves it also has $n - 1$ branching nodes, whence the bound. This is one of very few cases, and surely the most elegant one I know, of a combinatorial property on words explicated by a data structure. See [19] for a combinatorial argument as applied to the symmetric case of the equivalence relation defined in terms of the ending positions.

5. Waka and the String Resemblance System. Since I started my brief *excursus* with the discussion of a case of cheating it shall not be inappropriate to close it with the discussion of a case of suspected plagiarism.

The stage for this is offered by Waka, a form of traditional Japanese poetry with a 1300-year history. The structure of a poem consists of 5 lines and 31 syllables arranged in the form: 5-7-5-7-7. The fundamental device of Waka poetry is *honkadori* or poetic allusion. Honkadori consists of introducing few subtle changes in some older poem in order to produce a new one [35]. Here below is the phonetic transliteration of one example.

Alluded to: Kokin-shu #315	Allusive variation shugyoku-shu #3528
(Minamoto-no-Muneyuki)	(Jien)
ya-ma-sa-to-ha	ya-to-sa-hi-te
fu-yu-so-sa-hi-shi-sa	hi-to-me-mo-ku-sa-mo
ma-sa-ri-ke-ru	ka-re-nu-re-ha
hi-to-me-mo-ku-sa-mo	so-te-ni-so-no-ko-ru
ka-re-nu-to-o-mo-he-ha	a-ki-no-shi-ra-tsu-yu

which is rendered in English as follows.

Alluded to: Kokin-shu #315	Allusive variation shugyoku-shu #3528
A hamlet in mountain is	My home has been deserted
the drearier in winter.	
I feel that there is no one to see	Now in autumn, there is no one to see
and no green around	And no green around
	There is a pearl dew left in my sleeve

Although almost invisible to the naked eye, the transition from original to allusion presents almost a compendium of the genomic evolution. The first two verses feature a deletion between “ya” and ”to” (it must be by accident that those two syllables almost sound like ”hiatus”). A similar fate affects the last verse, which is advanced to the third position losing some syllables on account of metric constraints. There is a transposition of the entire fourth verse of the original, moved to the second position. And so on.

By the genesis and structure of these poems, it clearly required a lot of erudition for sophisticated contemporaries to appreciate every new composition. But how much of that could be made accessible to modern scholars? Perhaps computing can help. Since honkadori holds key to the subtlest nuances of a waka poem, it is tempting to arrange the available corpus of almost 500,000 poems in order of decreasing similarity, based on the mutual allusions. Some unifying framework for this is represented by the so called *string resemblance system* assembled by M. Takeda and co-workers [44], who formulated several measures of similarity based on shared substrings and for the most part originally developed in applications of protein analysis and classification. The system is built on the premise that the similarity of two objects depends on the degree of common structure and may be thus measured by the maximum score achieved by shared patterns. For instance, if the discriminating patterns are words with don’t care characters, then similarity is measured by the length of longest common subsequence.

Along these lines, one has

Definition 1. *A string resemblance system is composed by an alphabet* Σ, *a set of patterns* Π, *a map associating a pattern to a string in* Σ^*, *and a score* Φ *assigning a real number to a pattern.*

Here Σ^* denotes as usual the free monoid generated by the finite set Σ endowed with the operator of concatenation. Based on this, the similarity between x and y is $\sup\{\Phi(\pi) | \pi \in \Pi \text{ and } x, y \in L(\Pi)\}$.

The pattern sets comprised in this definition are restricted to the form $\Pi = (\Sigma \cup \{\dot{}\})^*$, but this repertoire is easily expanded in order to accommodate transpositions.

Definition 2. *An* order-free *pattern is a multiset* $\{u1, ..., uk\}$ *such that* $k > 0$ *and* $u_1, ..., u_k \in \Sigma^+$. *The language of this pattern is the union of the languages* $\Sigma^* u_{\sigma(1)} \Sigma^* u_{\sigma(2)} ... \Sigma^* u_{\sigma(k)} \Sigma^*$ *over all permutations of* $\sigma\{1, ..., k\}$.

For example, the language of the pattern $\{abc, de\}$ is $\Sigma^* abc \Sigma^* de \cup \Sigma^* de \Sigma^* abc \Sigma^*$. The membership problem for order-free patterns is NP-complete, and therefore the similarity computation is generally impractical. However the problem is polynomial-time solvable when k is fixed.

In [44], the authors test the similarity among poems based on three complementary measures. The first measure tries to capture the poetic allusion between individual line pairs. Pairwise line similarity is based on the longest common subsequence, but then this is slightly modified by allowing some alteration in the order of individual lines.

The other two measures assess the degree of alteration of the order of words appearing within the same line. Of these, one measure quantifies this alteration on a purely syntactic basis, whereas the other one incorporates the relative rarity of each shared pattern within the corpus.

Using these measures, the authors could identify instances of poetic allusion between the Kokin-Shu and Shin-Kokin-Shu anthologies. Among several consistent findings, they also found with great surprise an instance of poetic allusion that had never before been pointed out in the long history of Waka research. This led to the discovery that one of the most important poems by the renowned poet Fujiwara-no-Kanesuke was in fact heavily inspired by a poem found in Kokin-Shu! And this is how some tools of the trade of computational molecular biology ended up making waves at wide circles in literary communities and the general public alike.

REFERENCES

[1] A. Apostolico and G. Bejerano. *Optimal Amnesic Probabilistic Automata or How to Learn and Classify Proteins in Linear Time and Space.* Journal of Computational Biology, 7:3/4(2000), pp. 381–393.

[2] A. Apostolico, M. E. Bock, and S. Lonardi. *Monotony of Surprise and Large Scale Quest for Unusual Words.* Journal of Computational Biology, 10:3/4(2003), pp. 283–311.

[3] A. Apostolico, M.E. Bock, S. Lonardi, and X. Xu. *Efficient Detection of Unusual Words.* Journal of Computational Biology, 7:1/2(2000), pp. 71–94.

[4] A. Apostolico, M. Comin, and L. Parida. *Motifs in Ziv-Lempel-Welch Clef.* Proceedings of IEEE DCC Data Compression Conference, pp. 72–81, 2004.

[5] A. Apostolico, M. Comin, and L. Parida. *Conservative Extraction of Overrepresented Extensible Motifs.* Proceedings of ISMB 05, Intelligent Systems for Molecular Biology, Detroit, Mi., pp. 9–18, 2005.

[6] A. Apostolico, M. Comin, and L. Parida. *Bridging Lossy and Lossless Data Compression by Motif Pattern Discovery.* General Theory of Information Transfer and Combinatorics, Vol. II of Research Report ZIF (Center of interdisciplinary studies) Project, Bielefeld Oct. 1, 2002 – August 31, 2003, edited by R. Ahlswede with the assistance of L. Bäumer and N. Cai. Springer-Verlag LNCS 4123, pp. 787–799, 2006.

[7] A. Apostolico and Z. Galil, Eds. *Pattern Matching Algorithms.* Oxford University Press, 1997.

[8] A. Apostolico and R. Giancarlo. *Sequence Alignment in Molecular Biology.* Journal of Computational Biology, 5:2(1998), pp. 173–196.

[9] A. Apostolico and S. Lonardi. *Off-line Compression by Greedy Textual Substitution.* Proceedings of the IEEE , 88:11(2000), pp. 1733–1744.

[10] A. Apostolico and L. Parida. *Incremental Paradigms for Motif Discovery.* Journal of Computational Biology, 11:1(2004), pp. 15–25.

[11] A. Apostolico and F. P. Preparata. *Optimal Off-line Detection of Repetitions in a String.* Theoret. Comput. Sci., 22(1983), pp. 297–315.

[12] A. Apostolico and F.P. Preparata. *Structural Properties of the String Statistics Problem.* J. Comput. Syst. Sci., 31:3(1985), pp. 394–411.

[13] A. Apostolico and F.P. Preparata. *Data Structures and Algorithms for the String Statistics Problem.* Algorithmica, 15(1996), pp. 481–494.

[14] A. Apostolico and C. Tagliacollo. *Incremental Discovery of Irredundant Motif Bases of All Suffixes of a String in Time $O(|\Sigma|n^2 \log n)$.* Theoretical Computer Science, 408(2008), pp. 106–115.

[15] S. Bartezzaghi. *Sfiga All' OK Corrall*, Einaudi (1998).

[16] G. Benson. *Tandem Repeat Finder: a Program to Analyze DNA Sequences.* Nucleic Acids Research, 27:2(1999), pp. 573–580.

[17] T. Berger. *Rate Distortion Theory: A Mathematical Basis for Data Compression*, Prentice Hall, Englewood Cliffs, N.J., 1971.

[18] T. Berger, J. D. Gibson. *Lossy Source Coding.* IEEE Transactions on Information Theory, 44:6(1998), pp. 2693–2723.

[19] A. Blumer, J. Blumer, A. Ehrenfeucht, D. Haussler, M.T. Chen, and J. Seiferas. *The Smallest Automaton Recognizing the Subwords of a Text.* Theoretical Computer Science, 40(1985), pp. 31–55.

[20] G. S. Brodal, R.B. Lyngs Anna, and C.N.S. Pedersen. *Solving the String Statistics Problem in Time $O(n \log n)$.* Proc. 29th International Colloquium on Automata, Languages, and Programming, Springer LNCS, pp. 728–739, 2002.

[21] M. Crochemore. *An Optimal Algorithm for Computing the Repetitions in a Word.* Inform. Process. Lett., 12:5(1981), pp. 244–250.

[22] K. S. Fu and T. L. Booth. *Grammatical Inference: Introduction and Survey – Part I.* IEEE Transactions on Systems, Man and Cybernetics, 5(1975), pp. 95–111.

[23] K. S. Fu and T. L. Booth. *Grammatical Inference: Introduction and Survey — Part II.* IEEE Transactions on Systems, Man and Cybernetics, 5(1975), pp. 112–127.

[24] D. Gusfield and J. Stoye. *Simple and Flexible Detection of Contiguous Repeats Using a Suffix Tree.* 9th CPM 98, Springer LNCS 1448, pp. 140–152, 1998.

[25] R. W. Hamming. *Error Detecting and Error Correcting Codes.* Bell System Tech. J., 29(1950), pp.147–160.

[26] L.C.K. Hui. *Color Set Size Problem with Application to String Matching.* CPM '92: Proceedings of the Third Annual Symposium on Combinatorial Pattern Matching. Springer-Verlag LNCS, pp. 230–243, 1992.

[27] S. Istrail. *Tag Systems Generating Thue Irreducible Sequences.* Inf. Process. Lett., 7:3(1978), pp. 129–131.

[28] D.E. Knuth, J.H. Morris, and V.R. Pratt. *Fast Pattern Matching in Strings.* SIAM J. Comput., 6(1977), pp. 322–350.

[29] R. Kolpakov and G. Kucherov. *Finding Maximal Repetitions in a Word in Linear Time.* FOCS '99: Proceedings of the 40th Annual Symposium on Foundations of Computer Science, IEEE Computer Society, pp. 596–605, 1999.

[30] E. Lehman and A. Shelat, *Approximation Algorithms for Grammar Based Compression.* Proceedings of the eleventh ACM-SIAM Symposium on Discrete Algorithms (SODA 2002), pp. 205–212, 2002.

[31] V.I. Levenshtein. *Binary Codes Capable of Correcting Deletions, Insertions and Reversals.* Soviet Phys. Dokl., 6(1966), pp. 707–710.

[32] S.D. Levitt, S.J. Dubner William. *Freakonomics: A Rogue Economist Explores the Hidden Side of Everything*, Morrow, 2005.

[33] M.G. Main and R.J. Lorentz. *An $O(n \log n)$ Algorithm for Finding all Repetitions in a String.* J. of Algorithms, pp. 422–432, 1984.

[34] G. Manacher. *A new Linear-Time On-Line Algorithm for Finding the Smallest Initial Palindrome of a String.* J. Assoc. Comput. Mach., 22(1975), pp. 346–351.

[35] C.C. McCullough. *Brocade by Night 'Kokin Wakashu' and the Court Style in Japanese Classical Poetry.* Stanford University Press, 1985.

[36] M. Mongeau and D. Sankoff. *Comparison of Musical Sequences.* Computers and the Humanities, 24:3(1990), pp. 161–175.

[37] B. Morgenstern, S. J. Prohaska, D. PŽhler, and P.F. Stadler. *Multiple Sequence Alignment with User-defined Anchor Points.* BMC Algorithms for Molecular Biology, pp. 1–6, 2006.

[38] N. Pisanti, M. Crochemore, R. Grossi, and M.-F. Sagot. *Bases of Motifs for Generating Repeated Patterns with Wild Cards.* IEEE/ACM Trans. Comput. Biol. Bioinformatics, 2:1(2005), pp. 40–50.

[39] I. Rigoutsos, A. Floratos, L. Parida, Y. Gao, and D. Platt. *The Emergence of Pattern Discovery Techniques in Computational Biology.* Journal of Metabolic Engineering, 2:3(2000), pp. 159–177.

[40] D. Ron, Y. Singer, and N. Tishby. *The Power of Amnesia: Learning Probabilistic Automata with Variable Memory Length.* Machine Learning, 25(1996), pp. 117–150.

[41] D. Sankoff and J. Kruskal. *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison.* Addison-Wesley, Reading, MA, 1983.

[42] J.A. Storer. *Data Compression: Methods and Theory.* Computer Science Press, 1988.

[43] E.E. Stuckle, C. Emmrich, U. Grob, and P. J. Nielsen. *Statistical Analysis of Nucleotide Sequences. Nucleic Acids Res.* 18(1990), pp. 6641–6647.

[44] M. Takeda, T. Fukuda, I. Nanri, M. Yamasaki, and K. Tamari. *Discovering Instances of Poetic Allusion from Anthologies of Classical Japanese Poems.* Theoretical Computer Science, 292:2(2003), pp. 497–524.

[45] A. Thue. *Über Unendliche Zeichenreihen.* Norske Vid. Selsk. Skr. Mat. Nat. Kl. (Cristiania), (7), pp. 1–22, 1906.

[46] A. Thue. *Über die Gegenseitige Lage Gleicher Teile Gewisser zeichenreihen.* Norske Vid. Selsk. Skr. Mat. Nat. Kl. (Cristiania), (1), pp. 1–67, 1912.

[47] M. Waterman. *Introduction to Computational Biology: Maps Sequences and Genomes*, Chapman and Hall, 1995.

[48] P. Weiner. *Linear Pattern Matching Algorithms.* In: Proc. 14th Symposium on Switching and Automata Theory, pp. 1–11, 1973.