Computational Statistics
DOI 10.1007/s00180-007-0021-8

ORIGINAL PAPER

Using a VOM model for reconstructing potential coding regions in EST sequences

Armin Shmilovici · Irad Ben-Gal

Abstract This paper presents a method for annotating coding and noncoding DNA regions by using variable order Markov (VOM) models. A main advantage in using VOM models is that their order may vary for different sequences, depending on the sequences' statistics. As a result, VOM models are more flexible with respect to model parameterization and can be trained on relatively short sequences and on low-quality datasets, such as expressed sequence tags (ESTs). The paper presents a modified VOM model for detecting and correcting insertion and deletion sequencing errors that are commonly found in ESTs. In a series of experiments the proposed method is found to be robust to random errors in these sequences.

Keywords Variable order Markov model · Coding and noncoding DNA · Context tree · Gene annotation · Sequencing error detection and correction

1 Motivation and introduction

In order to reveal complex biological processes, many research projects nowadays focus on whole-genome analyses. A main analysis task is to find the *coding* sequences that provide templates for a protein production. Interest in protein-coding DNA has dramatically grown with the exposure of microarray

A. Shmilovici (✉)
Department of Information Systems Engineering, Ben-Gurion University, P.O. Box 653, Beer-Sheva, Israel
e-mail: armin@bgumail.bgu.ac.il

I. Ben-Gal
Department of Industrial Engineering, Tel-Aviv University, Tel-Aviv 69978, Israel

gene-expression data. Microarray analysis heralds an important advance in the determination of genes functionality under various conditions and in the identification of co-expressed genes (Delcher et al. 1999; Hanisch et al. 2002). For these purposes, automated sequencing techniques have become a vital scientific tool for producing a large amount of relatively reliable DNA data. Current gene-banks contain more than 3 billions bases of human DNA sequences and complete DNA sequences for dozens of other species. New genomes are being published in public databases and an increased number of gene-finding programs are now available for predicting protein-coding genes (Majoros et al. 2004; Brejova et al. 2005).

Given the genome data, many problems in computational biology can be reduced to optimal genome annotation, which is the process of attaching biological information to sequences (Cawley and Pachter 2003). Gene-finder programs (Fickett 1996; Larsen and Krogh 2003) are often used for annotating DNA sequences into segments of various suspected functionalities. The actual functionality of a segment may be validated by biologists using in-vivo (wet-laboratories) experiments.

The identification of protein-coding regions by computational techniques is challenging from several reasons. First, it is estimated that less than 3% of the human genome contains protein-coding sequences. Such a ratio introduces a combinatorial complexity to the search. Second, more than 60% of all the database entries today consist of expressed sequence tags (ESTs)[1] (Hatzigorgiou et al. 2001; Iseli et al. 1999; Lottaz et al. 2003). ESTs are sub-sequence of a transcribed protein-coding or non-protein coding DNA sequences. They represent a snapshot of genes that are expressed in certain tissues at a certain developmental stages. An EST is produced by a single-pass sequencing of a cloned mRNA taken from a cDNA library. The resulting sequence, whose length is limited to several hundreds nucleotides, has a relatively low quality and often contains sequencing errors such as insertion, substitution and deletion. Most EST extraction projects result in a large numbers of sequences. These are usually submitted to public databases, such as *GenBank* and *dbEST*, as batches of dozens to thousands of entries with a great deal of redundancy and errors. Hence, there is a vital need for efficient tools that can search within EST databases for protein-coding sequences which merit further investigation.

Since the computational identification of protein-coding DNA regions in the genome is a difficult problem, there is not a single standard solution for it, but rather a wealth of different approaches. Gene finding approaches can be roughly divided into several categories which are sometimes combined into a unified computational procedure (Fickett 1996; Brejova et al. 2005):

- *Extrinsic approaches* use a reverse translation of the genetic code to search the target genome for sequences that are similar to known mRNA sequences or proteins products. The advantage of this method is that once such a similarity is found, it yields good clues regarding the functionality of the

[1] dbEST release 042905 contains 26,773,945 public entries. 6,057,770 sequences are of human origin (http://www.ncbi.nlm.nih.gov/dbEST/dbEST_summary.html).

potentially new gene. The disadvantage is that when no homologue is found to the genomic sequence, one cannot conclude that the sequence is noncoding. The continuous growth of protein databases increases both the prediction ability and the required computational efforts of these approaches.

- *Comparative Genomics approaches* take advantage of the fact that genes and other functional elements undergo mutation at a slower rate then the rest of the genome. Genes can thus be detected by comparing genomes of related species. It is assumed that comparative genomics approaches will gain more attention as new genomes will be sequenced.
- *Ab Inito approaches* search systematically within the target genome for signals or contents of protein-coding genes, such as Pribnow box and transcription factor binding sites (Kel et al. 20003). *Pattern-based statistical analysis* is often used to reveal the different statistical properties of coding and noncoding regions. This approach is problematic as many patterns are not yet recognized and their identification often requires complex probabilistic models (e.g., Bernaola-Galvan et al. 2000; Nicorici et al. 2003; Ben-Gal et al. 2005). The hexanucleotide bias, which was formalized as an inhomogenous three-periodic fifth-order Markov chain [hereafter denoted as Markov(5)], is an example of a known model that was incorporated in genefinders such as GENSCAN (Burge and Karlin 1998). Hidden Markov models (HMM) are another popular model used to combine different statistical models (Cawley and Pachter 2003).

The implementation of variable order Markov (VOM) model to identify DNA sequences (e.g., Herzel and Grosse 1995; Ohler et al. 1999; Orlov et al. 2002; Shmilovici and Ben-Gal 2004) belongs to the last category of approaches. The VOM model can be seen as a generalization of the conventional (fixed-order) Markov model. In VOM models, unlike the Markov models, the dependence order is not fixed but rather depends on the sequence of observed nucleotides (called context). The flexibility in the selection of the model order per each context guarantees an efficient model parameterization, and in general, is found to balance well the bias-variance effects (Ben-Gal et al. 2005). Accordingly, the model can be better adapted to low quality data, such as EST, or to small datasets, such as an initially sequenced genome.

The contribution of this work is twofold. The first contribution is the use of the VOM model for supervised identification of coding and noncoding sequences, as proposed in Shmilovici and Ben-Gal (2004). The underlying idea is to construct two VOM models based on two given training sets: one VOM model representing the coding sequences, and the other VOM model representing noncoding sequences. Then, at the classification stage, the sequence type is determined by the VOM model which obtains a higher likelihood score or equivalently a higher compression rate. In Sect. 3, we experimentally demonstrate that such a VOM-based classifier is fairly robust to random substitution errors in the DNA sequence.

The second and main contribution of this work is the implementation of the constructed VOM models for detecting and 'correcting' insertion and deletion

sequencing errors that are often found in EST sequences. The goal here is to propose an algorithm that can automatically 'correct' low quality sequence data. Our preliminary sequence annotation program shows the usefulness of the suggested algorithm when used for this purpose.

The rest of this paper is organized as follows. Section 2, describes the VOM model and its use for compression. Section 3 presents sequence-annotation experiments that focuse primarily on error-correction ability of the proposed VOM model. Section 4 concludes the paper.

2 Introduction to VOM models

The VOM model was first suggested by Rissanen (1983) that called it the *context tree* and used it for data compression purpose. Later variants of the model were used in various applications, including genetic text modeling (Orlov et al. 2002), classification of protein families (Bejerano 2001), and classification of transcription factor binding sites (Bilu et al. 2002). Ziv (2001) proves that in contrast to other models the convergence of the context tree model to the 'true distribution' model is fast and does not require an infinite sequence length. The VOM algorithm we used here can be considered as a variant of the prediction by partial match (PPM) tree (with a different smoothing procedure), which was found in Begleiter et al. (2004) to outperform other VOM models variants. Yet, note that the used VOM model is different in its parameterization, growth, and pruning stages from the previous versions of the model. These differences become vital when comparing the algorithms on small datasets (Ziv 2001; Ben-Gal et al. 2005; Begleiter et al. 2004).

Next, we shortly present the VOM model that we used in our experiments. We follow the explanations and style in Begleiter et al. (2004) and Ben-Gal et al. (2005) that contain further details on the model and its construction.

Let $\sum$ be a finite alphabet of size $|\sum|$. In case of the DNA sequences $\sum = a, c, g, t$ and $|\sum| = 4$. Consider a sequence $x_1^n = x_1 x_2 \cdots x_n$ where $x_i \in \sum$ is the symbol at position i, with $1 \leq i \leq n$ in the sequence and $x_i x_{i+1}$ is the concatenation of x_i and x_{i+1}. Based on a training set x_1^n, the construction algorithm learns a model $\hat{P}$ that provides a probability assignment for any future symbol given its past (previously observed symbols). Specifically, the learner generates a conditional probability distribution $\hat{P}(x|s)$ for a symbol $x \in \Sigma$ given a context $s \in \Sigma^*$, where the * sign represents a context of any length, including the empty context. VOM models attempt to estimate conditional distributions of the form $\hat{P}(x|s)$ where the context length $|s| \leq D$ varies depending on the available statistics. In contrast, conventional Markov models attempt to estimate these conditional distributions by assuming a fixed contexts' length $|s| = D$ and, hence, can be considered as special cases of the VOM models. Effectively, for a given training sequence, the VOM models are found to obtain better model parameterization than the fixed-order Markov models (Ben-Gal et al. 2005).

As indicated in Begleiter et al. (2004), most learning algorithms include three phases: counting, smoothing, and context selection. In the counting phase, the

algorithm constructs an initial context tree T of maximal depth D, which defines an upper bound on the dependence order[2] (i.e., the contexts' length). The tree has a root node, from which the branches are developed. A branch from the root to a node represents a context that appears in the training set in a *reversed* order. Thus, an extension of a branch by adding a node represents an extension of a context by an *earlier* observed symbol. Each node has at most $|\sum|$ children. The tree is not necessarily balanced (i.e., not all the branches need to be of the same length) nor complete (i.e., not all the nodes need to have $|\sum|$ children). The algorithm constructs the tree as follows. It incrementally parses the sequence, one symbol at a time. Each parsed symbol x_i and its D-sized context, x_{i-D}^{i-1}, define a potential path in T, which is constructed if it does not yet exist. Note that after parsing the first D symbols, each newly constructed path is of length D. Each node contains $|\sum|$ counters of symbols given the context. The algorithm updates the contexts by the following rule: traverse the tree along the path defined by the context x_{i-D}^{i-1} and increment the count of the symbol x_i in all the nodes until the deepest node is reached. The count $N_x(s)$ denotes the number of occurrences where symbol x follows context s in the training sequence. These counts are used to calculate the probability estimates of the predictive model.

We illustrate the VOM learning algorithm with the following toy example: consider $\Sigma \equiv \{a, c, g, t\}$, and a training sequence x_1^{180}, which is composed of 30 consecutive repetitions of the pattern *"aaacgt"*. Figure 1 presents the resulting context tree for $D = 3$. Only nodes that were traversed at leased once (having at least one non-zero counts) are presented. Each node is labeled by the context which leads to it. The first line of numbers below the node label presents the four counts $N_x(s)$. For example, in the node labeled as "*aa*", the counts are $N_a(aa) = 30, N_c(aa) = 30, N_g(aa) = 0$, and $N_t(aa) = 0$. That is, from a total of 60 training sequence that contained the subsequence "*aa*", 30 sequences preceded by the symbol '*a*' and 30 sequences preceded by the symbol '*b*'.

The purpose of the second phase of the construction algorithm is to use the counts as a basis for generating the predictor $\hat{P}(x|s)$. An important question is how to handle unobserved events with zero value counts. Most variants of the VOM algorithm use some pseudo-counts to smooth the probability of zero frequency events. We used the following estimation rule for the conditional probability:

$$\hat{P}(x|s) = \frac{\frac{1}{2} + N_x(s)}{\frac{|\Sigma|}{2} + \sum_{x' \in \Sigma} N_{x'}(s)}$$

The last line in each extended node in Fig. 1 presents the respective four estimators $\hat{P}(x|s)$. For example, in the same node

[2] We use $D \leq \log(n+1)/\log(|\Sigma|)$, where n denotes the lengths of the training sequences. In our experiments $D = 8$.

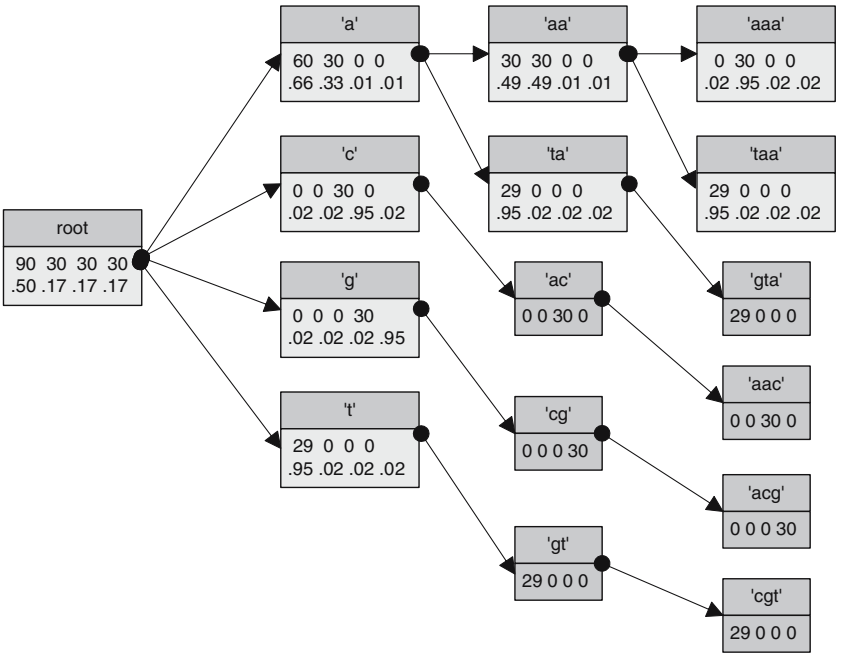

Fig. 1 The VOM model generated from 30 consecutive repetitions of the sequence "*aaacgt*". The *first line* of numbers below each node label present the four counts ordered with respect to nucleotides a, c, g, t. The *second line* of numbers below each node label presents the four conditional probabilities estimates. Truncated nodes are *dark shaded*

$$\hat{P}(a|aa) = \frac{\frac{1}{2}+30}{\frac{4}{2}+60} \cong 0.49.$$

The purpose of the third phase of the algorithm is to select only the significant contexts, thus reducing the number of parameters. Such reduction avoids over-fitting to the training data, and reduces both the memory usage and the computation time. We prune any leaf (external node) that does not contribute additional information relative to its "parent" node in predicting the symbol x. In particular, we compute the Kullback–Leibler divergence (KL) of the conditional probabilities of symbols between all leaves at depth k and their parent node at depth $k-1$:

$\mathrm{KL}(\mathrm{leaf}(s)) = \sum_{x\in\Sigma} \hat{P}(x|s) \log_2 \left(\frac{\hat{P}(x|s)}{\hat{P}(x|s_-)}\right)$, where $s = x_k x_{k-1} x_{k-2} \cdots$ denotes the context represented by the leaf, and $s^- = x_{k-1} x_{k-2} \cdots$ denotes the sub-context represented by parent node.

A leaf is pruned if its symbols' distribution is insufficiently different from the symbols' distribution in its parent node, thus, if it complies with $\mathrm{KL}(\mathrm{leaf}(s)) \leq 2(|\Sigma|+1)\log_2(n+1)$. The pruning process continues recursively to deeper nodes in the tree. In Fig. 1, the truncated nodes are smaller and dark shaded. For example, the leaf labeled "*aac*" has exactly the same counts distribution

as its parent node "*ac*", i.e., effectively, $\mathrm{KL}(\mathrm{leaf}(aac)) = 0$. Thus, the longer context "*aac*" does not change the predictor $\hat{P}(x|aac)$ over $\hat{P}(x|ac)$ and, therefore, is truncated from the VOM tree. The truncation process is repeated recursively, and the node labeled "*ac*" is truncated too at a later iteration, as seen in Fig. 1.

Once the VOM tree is constructed it can be used to derive the likelihood scores of test sequences $\hat{P}(x_1^T) = \prod_{i=1}^{T} \hat{P}(x_i|x_1 \cdots x_{i-1})$. Consider for example the VOM model in Fig. 1 and a test sequence $x_1^5 = gtaac$. The likelihood of this sequence is computed as follows: $P(gtaac) = P(g) \times P(t|g) \times P(a|gt) \times P(a|gta) \times P(c|gtaa) \cong P(g) \times P(t|g) \times P(a|t) \times P(a|ta) \times P(c|taa) = 0.17 \times 0.95 \times 0.95 \times 0.95 \times 0.02 = 0.002915$ (respectively, by nodes "*root*", "*g*", "*t*", "*ta*", and "*taa*" in Fig. 1).

Sequences with similar statistical properties to sequences from the training set (i.e., sequences that belong to the same class of the training dataset) are expected to obtain a higher likelihood score. The log of the inverse likelihood, $-\log \hat{P}(x_i|x_1 \cdots x_{i-1})$, is called the log-loss and known to be the ideal compression or "code length" of x_i, in bits per symbol, with respect to the conditional distribution $\hat{P}(X|x_1 \cdots x_{i-1})$ (Begleiter et al. 2004). That is, a good compression model that minimizes the log-loss can be used as a good prediction model that maximizes the likelihood and vice-versa (Feder and Merhav 1994). For example, the number of bits required to represent the previous x_1^5 is approximately $-\log_2(0.002915) \cong 8.42$ bits. Simple binary coding of 2 bits per symbol would require 10 bits to code this sequence of length five. Thus, the VOM succeeded to compress x_1^5. Moreover, the longer the sequence, the higher is the probability to obtain a lower compression rate. These known properties are used in the next section when identifying the test sequences by the model that provides the best compression ratio.

3 Annotating coding and noncoding DNA

Sequence annotation, such as the identification of protein-coding sequences, is a multi-step process (Fickett 1996; Brejova et al. 2005). In coding sequences, the nucleotides operate in triplets, called *codons*, where each codon encodes one amino-acid. The coding sequences are scanned in three alternative codon reading frames (sliding windows), since even for an error-free sequence, the first complete codon can start from the first, second or the third nucleotide in the sequence. The three reading frames are monitored simultaneously to indicate which of them represents the 'correct' encoding frame. After detecting a stop codon, the program scans the sequence backward, searching for a start codon. Sequence boundaries are also scanned for similarities with other known motifs, such as transcription factor binding sites and splice signals (Ohler and Niemann 2001). Here, however, we focus only on certain modules of the genefinder programs, without relying on additional biological information. Namely, by using the sequences' compression scores, which are based on the likelihood obtained by the VOM models, we classify sequences to "coding" or "noncoding" regions,

and for the latter we detect the correct reading frames. The classification scheme is applied to given sequences of fixed length, without using the ability of the VOM models to detect known motifs as shown in Ben-Gal et al. (2005).

3.1 Datasets and VOM-based classifiers

For completeness reason we now outline the construction process of the VOM classifier that was used in our experiments. For further details on the VOM classifier see Shmilovici and Ben-Gal (2004).

As our *training dataset* we rely on the GENIE dataset (GENIE 1998) that contains 462 coding sequences (called *exons*) and 2,381 noncoding sequences (called *introns*, when present within a gene). These sequences are representative segments of the human genome with less than 80% homology between sequences. We extracted a special dataset in which the sequences were chopped into segments of size 54 base pairs (bp), with the first nucleotide taking always the first position in the codon triplet. The training set that was used for training the VOM classifier contains 4,079 coding sequences and 25,333 noncoding sequences. An equally sized non-overlapping *testing dataset* was used to measure the accuracy of the VOM classifier.

To imitate the effect of short EST sequences,[3] we extracted a set of sequences of length 486 nucleotides each: 300 sequences (having a total of 145,800 nucleotides) were extracted from coding sequences, and 1,000 sequences (having a total of 486,000 nucleotides) were extracted from noncoding sequences. Once again, all the coding test sequences start from frame position 1. Recall that the number of sequences has a potential effect on the size of the truncated context tree as a result of the pruning algorithm. The used dataset is available from the authors.

It is well known that the distribution of the nucleotides depends on whether they belong to a coding sequence and if so, on their position in the codon (first, second, or third position). Accordingly, we constructed four VOM models from the training dataset. One VOM tree was constructed from noncoding segments and denoted by T_{NC}. Three inhomogeneous VOM trees—one for each position in the codon—were constructed from the phased coding segments and denoted respectively by $T_{C_i}, i = 1, 2, 3$. Thus, each inhomogeneous (position-dependent) VOM model was constructed based on a third of the coding sequence. All the VOM models were used simultaneously to score each test sequence: scoring the sequence by T_{NC}, then by a combination of VOM models $T_{C_1}, T_{C_2}, T_{C_3}$ depending on the nucleotide position in the codon. The likelihood score of each nucleotide in the testing set was obtained from the respective VOM tree, where the coding length of a nucleotide is given by $-\log_2$(likelihood_of_nucleotide). The coding length of a sequence is the sum of the coding lengths of the nucleotides

[3] EST sequences have about one error in 100 nucleotides and contain redundancies (Lottaz et al. 2003).

in that sequence. The following classification rule was applied to each test sequence:

If *length*(coding by $T_{C_i}, i = 1,2,3$) < *length*(coding by T_{NC}) then classify as "*coding DNA;*"

Otherwise, classify as "*noncoding DNA*"

The obtained accuracy of the classification rule was 86.1%. It was computed as the average of the correct classification ratios on the true coding and the true noncoding testing subsets (i.e., the average of the *true positive* and the *true negative* rates). In comparison, the accuracy for a Markov model of order 5 [Markov(5) model] was 86.3%. The Markov(5) model was considered as the best model in Fickett and Tung (1992), Iseli et al. (1999) and Lottaz et al. (2003) for both the non-phased and the phased sequences. The 95% confidence interval for the accuracy mean of the testing set was estimated to be approximately[4] $\pm 0.6\%$.

Although the obtained accuracy of the VOM model is equivalent to that of the Markov(5) model, its number of parameters (2,563) is much smaller than that of the Markov(5) model (15,149).[5] Further improvement in the accuracy of the VOM model (88.9%) was obtained via the method of boosting the training set[6] (Shmilovici and Ben-Gal 2004). The boosted VOM was used in the experiments in the following sub-sections.

3.2 EST annotation via VOM classifier

As noted by Iseli et al. (1999), tuning the parameters of a gene-finding program is sometimes a subjective process that is motivated by the desire to maximize the percentage of *true positive* discoveries subject to an acceptable level of *false negative* discoveries. Yet, the main purpose of the following set of experiments, which is described in Table 1, is to demonstrate the viability of the VOM classifier for gene-finding programs. Therefore, these experiments are not exhaustive, and no attempt was made to fully optimize the performance of the VOM-based classifier to the used datasets.

For a preliminary annotation of raw sequence, a sliding window of size 54 nucleotides (starting 26 nucleotides upstream the annotated nucleotide) was compressed by each one of four VOM models: the three phased coding VOM

[4] Based on the normal approximation to the Binomial distribution and on the worst (least accurate) obtained case with $p = 0.861$ we use: $1.96 \times 0.5\sqrt{\frac{p(1-p)}{n_1} + \frac{p(1-p)}{n_2}}$, with $n_1 = 4,079, n_2 = 25,333$.

[5] There are four parameters (all needed for the smoothing proceadure) in each node of the four tree models (one non-coding tree and three coding trees). Thus, the number of parameters in each of the four Markov(5) trees is $4^{5+1} = 4,096$, leading to a total of 16,384 parameters. The actual obtained number (15,149) was smaller than that, since some contexts with specific biological functions (such as contexts that include the "stop" codons) are excluded from the learning dataset.

[6] Boosting is a conventional machine-learning technique that attempts to improve the efficiency of a weak classifier via re-weighting of the training set. We followed the conventional boosting procedure as described in Freund and Schapira (1997).

Table 1 Description of key sequences used in our experiments

Exp. No.	Description of sequences (size 486)
1	Compression of sliding windows of length 54 by the four VOM models
2	Classification of the nucleotides of sequence#1, based on the minimum compression rate—raw annotation
3	A penalized re-annotation of series#2 eliminating short jitter in annotation
4	As#3, only that each sequence was contaminated with 3 *indel* errors.
5	As#4, only that the *indels* in the sequence were corrected before re-annotation
6	As#3, only with 1/40 random substitution error
7	As#6, only with 1/12 random substitution error
8	As#6, only with 1/16 random substitution error
9	As#5, only that the correction and annotation performed with ESTScan

trees (denoted respectively by '1', '2', '3') and the noncoding VOM tree (denoted by '4'). The annotation was defined by the model that obtained the highest compression rate, i.e., obtaining the minimum number of bits per a sequence window of size 54. The first line in Table 2 (series#1) presents the mean compression rate (in bits) as obtained by the above four VOM models for both coding and noncoding sequences (the standard deviation was approximately fixed in all cases to 11 bits). As expected, the 'phase 1' VOM model (alternatively, the 'noncoding' VOM model) obtained the highest compression for the 'coding' dataset (alternatively, for the 'noncoding' dataset). Figure 2a plots the four compression rates (in bits per windows in the ordinate) for an exemplified *coding* sequence aligned in the abscissa. The *bottom* graph in Fig. 2a is used to identify the best model for annotation. Figure 2b presents a raw annotation resulting from Fig. 2a. As depicted, the graph of the phase 1 coding rate is the lowest graph for most of the aligned nucleotides. Thus, most of the nucleotides are correctly classified as coding with a phase 1 window. Series#2 in Table 2 presents the statistics of the *annotated* symbols. The classification for the coding (noncoding) dataset is correct for 79.67% (71.85%) of the annotated nucleotides. We speculate that such a reduction in the accuracy might be partially related to edge effects (e.g., the sliding windows are shorter at the edges).

Figure 2b presents a jittery and an unrealistic annotation. It is well known that both coding DNA sequences, as well as noncoding sequences, have certain minimal lengths. One can conjecture that small scale jittery behavior is the result of a-typical local characteristic of the DNA sequence, rather than an indication of a transition in the annotated model (transitions resulting from sequencing errors are handled in the following subsection). The longer the sequence, the higher is the probability of the occurrence of an a-typical local characteristic. Accordingly, one can apply an algorithmic solution to reduce the local jittery behavior of the annotation, while retaining the large-scale minimal cost of the annotation.

One standard method to reduce the jittering occurrences is by introducing a *penalty* term associated with each transition in the annotation. The optimal annotation for each sequence is defined by an annotation path that minimizes

Table 2 Nucleotide classification statistics for experiments (described in Table 1)

Series/Experiment No.	Nucleotide classification statistics for 300 coding sequences The following VOM classifiers				Nucleotide classification statistics for 1,000 noncoding sequences The following VOM classifiers			
	Phase1	Phase2	Phase3	Non-cod	Phase1	Phase2	Phase3	Non-cod
1 (mean bits) (Std_dev.∼11)	**98.22**	106.12	107.02	104.93	106.72	106.63	106.63	**99.66**
2	**79.67%**	4.26%	4.77%	11.29%	9.32%	9.44%	9.39%	**71.85%**
3	**88.69%**	2.75%	2.59%	5.97%	8.27%	9.87%	9.62%	**72.24%**
4	**43.07%**	23.90%	23.33%	9.70%	8.90%	9.73%	9.11%	**72.27%**
5	**72.78%**	12.33%	6.00%	8.90%	N/A	N/A	N/A	N/A
6	**87.46%**	2.96%	3.07%	6.50%	N/A	N/A	N/A	N/A
7	**86.78%**	2.86%	3.28%	7.09%	N/A	N/A	N/A	N/A
8	**86.58%**	3.21%	3.22%	6.99%	N/A	N/A	N/A	N/A
9	**74.67%** combined			25.33%	N/A	N/A	N/A	N/A

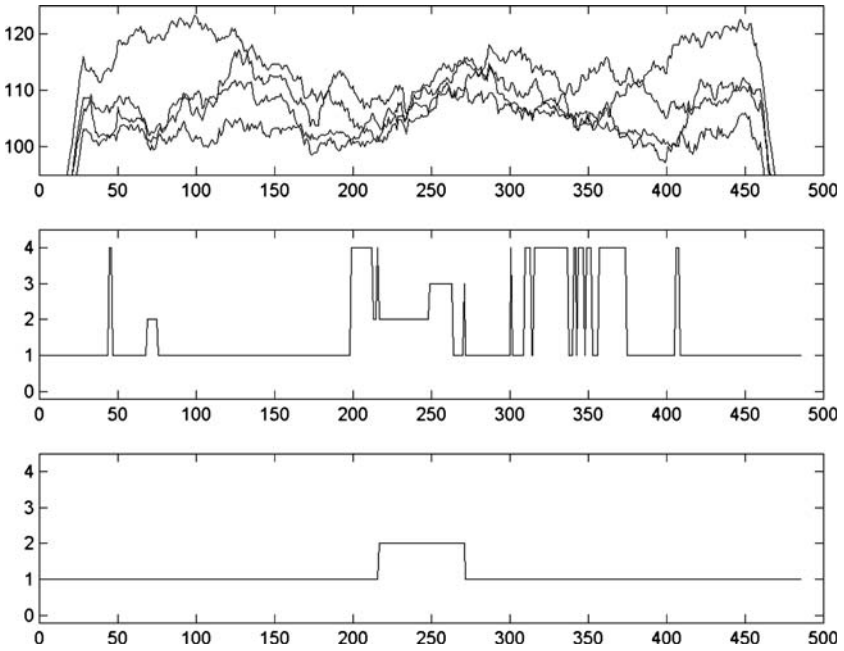

Fig. 2 **a** The four compression rates for the fifth coding sequence of experiment#1 (*top*); **b** raw annotation of top sequence from experiment#2 (*middle*); **c** experiment#3, penalized re-annotation of middle sequence (*bottom*). *1* denotes the correct coding phase; *2*, *3* denotes incorrect coding phase and *4* denotes the incorrect noncoding

the sum of the compression rate (the sum of the number of bits accumulated from all the models chosen by the annotation program) and the penalty term multiplied by the number of model transitions in this path. Finding the optimal algorithm (summing over all possible paths) is NP-hard for long sequences, yet, it is computationally tractable for short sequences with a Viterbi like

dynamic-programming method (corresponding to finding the most likely path), as implemented for annotation of DNA sequences by Xu et al. (1995).

Following the main concept of the algorithm in Hatzigorgiou et al. (2001), we assume that a simpler algorithm that has at most one backtrack of the solution path (unlike the dynamic-programming algorithm) could generate a sufficiently good annotation. The outline of the algorithm is presented as follows:

1. Scan the raw annotation along the sequence from left to right until the next transition in the annotation is detected (e.g., nucleotide#216 in Fig. 2c). Terminate if the end of the sequence was located.
2. Continue the scan until another transition in the raw annotation is detected (e.g., nucleotide#274 in Fig. 2c), or the end of the sequence is detected as the next transition.
3. For the current window between the two transition points (or the transition point and the sequence ending), compute *sum1* as the sum of the compression measures by the annotation *before* the transition window, and *sum2* as the sum of the compression measures by the annotation within the transition window (e.g., in Fig. 2c there is a transition window in the annotation from phase 1 to phase 2).
4. If *sum1*<(*sum2*+ *penalty term*) then eliminate the transition window by extending the annotation previous to the transition window. Otherwise, keep the annotation in the current transition window.
5. Goto 1

Note that such a simple algorithm can reliably eliminate short jitters, yet for wide transition windows it is incapable to optimize the exact position of the transition point. This deficiency of the algorithm has only a small impact when it is applied as a part of a full gene-finding program, since other algorithmic procedures and additional biological information can be used to re-annotate the edges of the transition window. When the algorithm is used for error-detection and error-correction in sequences, as exemplified in the next subsection, such deficiency of the algorithm may introduce a bias in the location of the exact position of the sequencing errors. A high penalty will result in the elimination (as opposed to frame correction) of out-of-frame regions. As noted by Iseli et al. (1999), the optimal penalty value often depends on the quality of the sequence.

The current algorithm, with *penalty* $= 100$ was used to remove the jitters from the previously annotated sequence. Figure 2c introduces the re-annotation of the sequence that was annotated in Fig. 2b. As can be seen, the jitter was removed, except for one (erroneous) transition window of width 56. This penalty value eliminates transition windows shorter than (roughly) 40 nucleotides, in comparison to a minimum window length of 60 in Hatzigorgiou et al. (2001) and a default window length of 33 in ESTScan (http://www.ch.embnet.org/software/ESTScan.html). Series no. 3 in Table 2 presents the statistics of the re-annotated sequence. As seen, the accuracy increased significantly for the re-annotation of the coding sequences (88.69%). Note, however, that there is only a modest increase in the annotation accuracy for the noncoding sequences (72.24%). This result could be explained by the random nature of noncoding

Table 3 Entire sequence-based statistics for series of Table 2

Series no.	Percentage of complete sequences entirely classified as				Total no. transitions	No. of sequences with the following no. of transitions							
	Phase1 (%)	Phase2 (%)	Phase3 (%)	Non-cod (%)		0	1	2	3	4	5	6	7
2 coding	**8.67**	0.0	0.0	0.0									
2 non-coding	0.5	0.4	0.6	**4.5**									
3 coding	**59.0**	0.33	0.0	0.67	199	179	64	41	11	5	0	0	0
3 non-coding	0.0	0.1	0.0	**41.6**	1197	430	208	185	109	49	18	1	0
4 coding	**0.0**	0.67	0	1.67	992	5	4	39	182	59	9	1	1
4 non-coding	0.1	0.2	0.6	**42.1**	1164	428	212	201	103	40	15	1	0
5 coding	**59.67**	7.67	2.0	0.0	156	207	42	41	8	2	0	0	0
6 coding	**55.33**	0.0	0.0	1.0	232	169	62	48	12	7	2	0	0
7 coding	**56.0**	0.0	0.0	1.0	245	171	53	46	20	10	0	0	0
8 coding	**50.67**	0.0	0.0	1.0	256	155	68	50	21	5	1	0	0
9 coding		18.67% combined		**22.67**	–	–	–	–	–	–	–	–	–

sequences, as indicated by the number of false transitions in experiment#3 in Table 3. The left side of Table 3 introduces the *entire* sequence-based statistics for series/experiment of Tables 1, 2 (rather than the nucleotide-based statistics in Table 2), such as the classification of an entire sequences to one class. The right side of Table 3 introduces the distribution of the number of transitions in the dataset in each experiment (for examples, there were a total of 199 transitions in the 300 coding sequences in experiment 3. Only 179 sequences—59.0%—were transition free, while five sequences contained four transitions each etc.) The average accuracy for the prediction of both coding and noncoding nucleotides is 80.47% (88.69/2 + 72.24/2 as seen in Table 2). In comparison, Hatzigorgiou et al. (2001) also used a sliding windows of length 54, to which they applied a neural network corrected by a dynamic programming algorithm as well as an identification of the start and end motifs of each gene. They obtained an accuracy of 84% on their test set (using only the neural network coding module) and an overall accuracy of 89.7% for the prediction of both coding and noncoding nucleotides in EST sequences. Since 86% of the sequences in their dataset contained coding sequences, the accuracy in our experiments is roughly equivalent to theirs. Note, that their program was subject to an optimization procedure, unlike the proposed one-pass algorithm that was implemented here.

The ultimate test for the quality of the annotation program is often selected as the number of coding sequences that were correctly identified as "pure" phase 1 coding. The jitter reduction algorithm increased this number from 8.67 to 59.67% for the coding sequences, and from 4.5 to 41.6% for the noncoding segments (Table 3, series no. 2 and 3). Thus, the jitter reduction algorithm is sufficiently effective for gene annotation purposes, although the values of the *penalty* term and the sliding window length could be the subject of further optimization. In addition, a coding region was detected in 99.3% of the coding sequences. In comparison, Hatzigorgiou et al. (2001) detected correctly 92.7% of their coding sequences as having coding regions, yet only 38.9% of their noncoding sequences were predicted as "pure" noncoding. Iseli et al. (1999) managed to extract about 95% of true coding sequences, while obtaining a 10% false positive rate.

3.3 Annotation of sequences with errors

As indicated above, EST sequences are relatively short, containing only several hundred nucleotides, and are prone to sequencing errors such as insertion, substitution and deletion (Hatzigorgiou et al. 2001; Iseli et al. 1999; Lottaz et al. 2003). Sequencing errors generates false signals and sharp deterioration in the performance of most gene-finding algorithms (Brown et al. 1998). The purpose of the following set of experiments is to demonstrate the robustness of VOM-based annotation to sequencing error, and to observe its error correction performance.

Two categories of errors are considered: (1) substitution errors; and (2) insertion and deletion errors (*indels*). Substitution errors may cause a local reduction

in the coding potential, which might results in a relatively gradual deterioration of the accuracy due to the window weighting. Indels, on the other hand, are potentially more severe errors as they change the frame of the coding sequence. Due to the implemented window weighting procedure, a frame change might be detected only several nucleotides away from its actual location.

Error correction methods via VOM-like methods have been applied before in the context of text correction (Vert 2001). In particular, the VOM model can be used in a predictive mode as follows: given a sequence s, one aims to predict the next nucleotide $\hat{x}$ in the series. The predicted value would be the nucleotide that maximize the likelihood $\hat{x} = \arg\max_{x'}\{\hat{P}_D(x'|s)\}$, where $\hat{P}_D$ denotes the probabilities estimated from a VOM model constructed with a depth limit D. For illustration, consider the VOM model in Fig. 1 and the prediction of the next nucleotide in the sequence "*aaca*". Given this VOM tree, the longest context from this sequence is '*a*' (node '*a*'), with the nucleotide '*a*' obtaining the maximal likelihood, $P(a|a) = 0.66$. Therefore, the nucleotide '*a*' is selected as the most probable prediction. Note that for prediction within the coding regions, one has to use the VOM model of the appropriate frame.

The first step in the correction of indel errors is the detection of the position of the frame shift (as detected by the previous annotation algorithm). A frame shift of one position implies that there was a deletion error that can be corrected by the insertion of one nucleotide. The inserted nucleotide can be either an "unspecified" one (marked by N, or X), or the most probable nucleotide according to the suffix context to the error. A frame shift of two phases implies that there was an insertion error (or a deletion of two nucleotides). Removing one nucleotide (or adding two) will correct the frame-shift error. Note that due to the annotation "lag", the insertion (deletion) position can be several nucleotides away from the true position of the indel transition. The current correction point is determined by the identification of the transition point. An improved correction algorithm might search the neighborhood of the transition point for the best position to introduce the error correction. Any application of a correction algorithm should avoid the generation of a stop codon by accident. As noted by Iseli et al. (1999), error correction may be unrealistic, but this aspect is often less relevant from a practical point, since it is used as a mathematical artifact to annotate low quality ESTs. If a sequence is considered sufficiently important, then it will be re-sequenced again to remove the errors.

The previous dataset of 300 coding sequences of length 486 bp was contaminated with sequencing errors to simulate the effect of sequencing errors on the annotation of EST sequences. The sequencing errors were introduced as follows (the errors were generated sufficiently far apart—twice the window width—to reduce interaction effects): nucleotides 121 and 242 were removed (deletion error) to introduce frame shifts in positions 121 and 241, respectively. Nucleotides 362,363 were duplicated (an insertion error) to introduce a two-phase frame shift in position 363. Hereafter, the term "annotation" is used as a reference to the penalized annotation algorithm presented in the previous subsection. Figure 3a presents the annotation of the error-contaminated

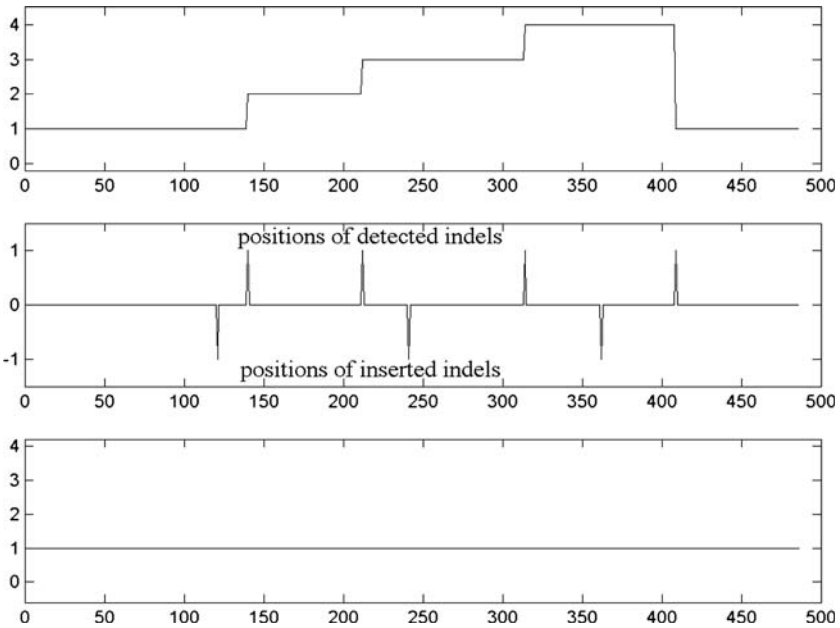

Fig. 3 **a** The fifth coding sequence of experiment#4, sequence contaminated with three indel errors (*top*); **b** the positions of the four detected indels compared to the positions of the true indels (*middle*); **c** experiment#5, penalized re-annotation of the indel corrected sequence (*bottom*). *1* denotes the correct coding phase; *2*, *3* denotes incorrect coding phase and *4* denotes noncoding

sequence of Fig. 2. Line no. 4 in Table 2 presents the statistics of the resulting annotation. Comparing it with line no. 3, note that the generation of the above three errors was sufficient to reduce the correct frame coding annotation from 88.69 to 43.07%.

The error correction was performed as follows: the annotated coding sequences were scanned, right to left, to detect frame shifts. Any single-phase frame transition (deletion error) was corrected by inserting the predicted nucleotide (left to right) from the appropriate VOM-phased model into the position of the transition. Any two-phase frame transition (insertion error) was corrected by deleting the nucleotide at the phase transition. The resulting sequence was adjusted to length 486 (by truncation of excess nucleotides or by repeated duplication of the last nucleotide). Figure 3b presents the four indels (identified in positions 140, 212, 314, 409 compared to the actual (true) indels of positions 121, 241 and 362. Figure 3c presents the annotation of the indel-corrected sequence. As can be seen, all the phase errors, including the false one detected in Fig. 2c are indicated and removed. The average position of the indels with respect to all the sequences in the coding dataset is found to be 124.04, 247.84, 367.88, which introduce, respectively, an average error detection "lag" of 3.04, 6.84, 5.88 nucleotides (measured only for those indels which were corrected, while matching is performed left to right), a median "lag" of 2, 3, 2.5

nucleotides, respectively, and standard deviations of 29.8, 29.07, 27.95 nucleotides, respectively.

Table 2 presents the statistics of the annotated nucleotides in the corrected sequences: as seen from series no. 5, correct frame coding annotation increased from 43.07% before the error correction to 72.78% after the error correction. Note from Table 3 that 59.67% of the sequences were detected as "pure" phase 1 coding sequences after the correction, which is equivalent to the annotation quality before the error contamination process.[7] The error detection and correction statistics is also presented in Table 3: the annotation before the indel contamination (series no. 3) contained 199 (apparently false) annotated region transitions concentrated in 40.33% of the sequences, of which 125 are phase transitions, such as the one in Fig. 2c. The error contaminated coding sequences (series no. 4) had 922 region transitions in 98.33% of the sequences. During the error-correction process, 720 deletion errors were detected and corrected (compared to the actual 600 deletion errors) and 82 insertion errors were corrected (compared to the actual 300 insertion errors). The re-annotation of the error-corrected sequences detected 156 region transitions concentrated in 31.0% of the sequences. Comparing coding series no. 3 to coding series no. 5 we see that the error correction restored the annotation accuracy its quality level before the contamination (59.0% vs. 59.67%).

Computing the accuracy of the error correction algorithm is a difficult task in general, since it seems to correct also some of the spurious "annotation noise". The error correction rate can be estimated either by the proportion of corrections of the total artificially inserted errors, $(720+82)/900 = 89\%$, or by the proportion of transitions before and after the correction, $(992 - 156)/992 = 84\%$. This result indicates that VOM based correction outperforms[8] the dynamic programming error-correction algorithm presented in Xu et al. (1995). The algorithm there detected and corrected 76% of the indels, with an average distance of 9.4 nucleotides between the position of an indel and the predicted position.

Note that the performance of an error-correcting algorithm depends on the error rate in the data. Therefore, in general, different correction algorithms should be compared with the same dataset. ESTScan of Iseli et al. (1999) use a Markov(5) model to detect sequencing errors and suggest a correction. Lottaz et al. (2003) improve ESTScan with the inclusion of models for the un-translated regions as well as the start and stop sites). In experiment no. 9 the same 300 error contaminated coding sequences used in experiment no. 5 were processed with the ESTScan program (the 300 sequences were transferred manually one by one, using the program's defaults). Comparing experiments 5 and 9 in Table 3, the ESTScan erroneously detected 22.67% of the error corrected sequences as noncoding (compared with 0% for the VOM program). The percentage of nucleotides detected as noncoding (Table 2) is even higher and equals 25.33%

[7] The 95% confidence interval is approximately equal to: $1.96 \cdot \sqrt{\frac{0.59(1-0.59)}{300}} \cong 4.8\%$.

[8] The 95% confidence interval is approximately equal to: $1.96 \cdot \sqrt{\frac{0.84(1-0.84)}{992}} \cong 2.3\%$.

Table 4 The EST datasets

Starting GenBank[a] sequence no.	No. of sequences	Average length seq.	*N* nucleot. (%)	Nucleot. identified as coding (%)	No. Cod. longer than 50	No. Cod. longer than 100	No. Corrected indels
62639482	150	556	0	79.44	144	136	311
62727861	345	726	1.07	61.06	324	300	827

[a] http://www.ncbi.nlm.nih.gov/blast/db/FASTA/est_human.gz

(compared with 8.90% for the VOM based program). Similar to the discussion in Lottaz et al. (2003), the VOM model cannot correct indel errors with proximity shorter than the context length. However, these situations lead to very few misinterpreted nucleotides and just two or three wrong amino acids.

The purpose of the last set of experiments was to test the sensitivity of the VOM-based classifier to random substitution error. Nucleotides in the dataset were substituted with a random probability of 1/40, 1/24 and 1/16, respectively. Comparing experiments no. 6, 7 and 8 to the noiseless experiment no. 3, we see a degradation in the performance rate. Yet, as seen from Table 3 the VOM classifier is found robust to substitution errors even for these lower quality sequences.

3.4 Annotation of real EST sequences

The dataset used in the previous section is in a sense "too ideal" as EST sequences. In practice, an EST sequence might contain both coding and non-coding sequences. The purpose of this set of experiments was to demonstrate the VOM-based annotation of real EST sequences in the presence of sequencing error, and to test it for error correction. Table 4 presents the annotation statistics of two different datasets of human ESTs. The first dataset contained relatively shorter sequences compared to the second dataset and of higher quality (0% nucleotides of unspecified type N). Based on the human VOM model as considered in the previous sub-section, 79.44% (respectively 61.06%) of the nucleotides in the first (second) dataset were detected as coding. Note that 144 of the sequences (respectively 324) contain coding regions longer than 50 nucleotides, thus, are of interest for further analysis. Considering the fact that both EST datasets originated from mRNA (thus, most of their sequences contain a coding region, possibly corrupted by sequencing errors), the VOM model correctly detected over 93% of the ESTs as sequences that contain coding regions.

The last column in Table 4 presents the number of indel errors that were corrected by the VOM based program. Note that the indel correction did not change the number of ESTs that contained significant coding regions. Yet, here, unlike in the previous section, the correct phase of the coding regions in unknown.

4 Discussion

Recognition of coding DNA regions is an important phase of any gene-finder procedure. In general, current gene-recognition approaches are exceedingly multifaceted, implementing a variety of well-established algorithms. Yet, we believe that there exist niche datasets with specific characteristics that are not entirely addressed by conventionally used algorithms. Two examples to where such niche sets can be found are:

(a) Datasets from newly sequenced genomes that share little homology with known datasets. Often, in such cases it is difficult to tune properly the gene-finders, e.g., due to over-parameterization which is not well understood.
(b) Short and relatively low-quality sequences (such as ESTs) that contain sequencing errors. In this case, the performance of homology-based methods such as in Brown et al. (1998), or HMM such as in Iseli et al. (1999), could result in a relatively poor performance, as seen in experiment 9 in Table 3.

In this paper, we introduce the VOM-based method to sequence annotation and error-correction. The VOM model was originally introduced in the field of information theory (Rissanen 1983) and since then has been implemented successfully in various research areas, such as statistical process control (Ben-Gal et al. 2003), analysis of financial series (Shmilovici et al. 2003) and Bioinformatics (Bejerano 2001; Ben-Gal et al. 2005).

In our experiments the prototype sequence-annotation program was demonstrated to be of either equivalent or superior accuracy compared to other methods from the literature (such as dynamic programming, neural networks and HMM) that were used for classifying EST sequences, while potentially being computationally more simple (Begleiter et al. 2004). The proposed model turned out to be robust to sequencing errors, and effective for error prediction and error-correction.

The initial encouraging results make it tempting to conjecture that elements of the proposed method could be integrated into other gene-finding procedures. Preliminary experiments (Zaidenraise et al. 2004) indicate that the VOM model is more reliable in detecting short genes in comparison to Markov(5) based gene-finders. In a current research, we are integrating the VOM algorithm into the *Glimmer* open source gene-finder (Delcher et al. 1999). This integration might enhance the Glimmer performance on short coding fragments taken from relatively low-quality sequenced data. Finally, note that human datasets, as those considered here, are of a huge size and are relatively error-free. This is a-typical in datasets of other organisms that are of interest. The VOM model, due to its efficient parameterization, is expected to be effective in these analyses when data is scarce, or when the quality of the sequence annotation is poor.

References

Begleiter R, El-Yaniv R, Yona G (2004) On prediction using variable order markov models. J Artif Intell 22:385–421

Bejerano G (2001) Variations on probabilistic suffix trees: statistical modeling and prediction of protein families. Bioinformatics 17(1):23–43

Ben-Gal I, Shmilovici A, Morag G (2003) CSPC: a monitoring procedure for state dependent processes. Technometrics 45(4):293–311

Ben-Gal I, Shani A et al. (2005) Identification of transcription factor binding sites with variable-order Bayesian networks. Bioinformatics 21(11):2657–2666

Bernaola-Galvan P, Grosse I et al. (2000) Finding borders between coding and noncoding DNA regions by an entropic segmentation method. Phys Rev Lett 85(6):1342–1345

Bilu Y, Linial M, Slonim N. Tishby N (2002) Locating transcription factors binding sites using a Variable Memory Markov Model, Leibintz Center TR 2002–57. Available online at http://www.cs.huji.ac.il/∼johnblue/papers/

Brejova B, Brown D.G, Li M, Vinai T (2005) ExonHunter: a comprehensive approach to gene finding. Bioinformatics 21(Suppl 1):i57–i65

Brown NP, Sander C et al. (1998) Frame: detection of genomic sequencing errors. Bioinformatics 14(4):367–371

Burge C, Karlin S (1998) Finding the genes in genomic DNA. Curr Opin Struct Biol 8(3):346–354

Cawley SL, Pachter L (2003) HMM sampling and applications to gene finding and alternative splicing. Bioinformatics 19(Suppl 2):ii36–ii41

Delcher AL, Harmon D, Kasif S, White O, Salzberg SL (1999) Improved microbial gene identification with GLIMMER. Nucl Acids Res 27(23):4636–4641

Feder M, Merhav N (1994) Relations between entropy and error probability. IEEE Trans Inf Theory 40(1):259–266

Fickett JW (1996) Finding genes by computer: the state of the art. Trends Genet 12(8):316–320

Fickett JW, Tung CS (1992) Assessment of protein coding measures. Nucl Acids Res 20(24): 6441–6450

Freund Y, Schapira RE (1997) A decision theoretic generalization of on-line learning and an application to boosting. J Comput Syst Sci 55(1):119–139

GENIE data-sets, from Genbank version 105 (1998) Available: http://www.fruitfly.org/seq_tools/datasets/Human/CDS_v105/ ; http://www.fruitfly.org/seq_tools/datasets/Human/intron_v105/

Hanisch D et al. (2002) Co-clustering of biological networks and gene expression data. Bioinformatics 1:1–10

Hatzigorgiou AG, Fiziev P, Reczko M (2001) DIANA-EST: a statistical analysis. Bioinformatics 17(10):913–919

Herzel H, Grosse I (1995) Measuring correlations in symbols sequences. Phys A 216:518–542

Iseli C, Jongeneel CV, Bucher P (1999) ESTScan: a program for detecting, evaluating, and reconstructing potential coding regions in EST sequences. In: Proceedings of intelligent systems for molecular biology. AAAI Press, Menlo Park

Kel AE, Gossling E et al. (2003) MATCH: a tool for searching transcription factor binding sites in DNA sequences. Nucl Acids Res 31(13):3576–3579

Larsen TS, Krogh A (2003) EasyGene—a prokaryotic gene finder that ranks ORFs by statistical significance. BMC Bioinf 4(21) Available Online www.biomedcentral.com/1471-2105/4/21

Lottaz C, Iseli C, Jongeneel CV, Bucher P (2003) Modeling sequencing errors by combining Hidden markov models. Bioinformatics 19(Suppl 2):ii103–ii112

Majoros WH, Pertea M, Salzberg SL (2004) TigrScan and GlimmerHMM: two open source ab initio eukaryotic gene-finders. Bioinformatic 20:2878–2879

Nicorici N, Berger JA, Astola J, Mitra SK (2003) Finding borders between coding and noncoding DNA regions using recursive segmentation and statistics of stop codons. Available Online: http://www.engineering.ucsb.edu/ ∼jaberger/pubs/FINSIG03_Nicorici.pdf

Ohler U, Niemann H (2001) Identification and analysis of eukaryotic promoters: recent computational approaches. Trends Genet 17:56–60

Ohler U, Harbeck S, Niemann H, Noth E, Reese M (1999) Interpolated Markov chains for eukaryotic promoter recognition. Bioinformatics 15(5):362–369

Orlov YL, Filippov VP, Potapov VN, Kolchanov NA (2002) Construction of stochastic context trees for genetic texts. In Silico Biol 2(3):233–247

Rissanen J (1983) A universal data compression system. IEEE Trans Inf Theory 29(5):656–664

Shmilovici A, Ben-Gal I (2004) Using a compressibility measure to distinguish coding and noncoding DNA. Far East J Theoret Stat 13(2):215–234

Shmilovici A, Alon-Brimer Y, Hauser S (2003) Using a stochastic complexity measure to check the efficient market hypothesis. Comput Econ 22(3):273–284

Vert JP (2001) Adaptive context trees and text clustering. IEEE Trans Inf Theory 47(5):1884–1901

Xu Y, Mural RJ, Uberbacher EC (1995) Correcting sequencing errors in DNA coding regions using a dynamic programming approach. Bioinformatics 11:117–124

Zaidenraise KOS, Shmilovici A, Ben-Gal I (2004) A VOM based gene-finder that specializes in short genes. In: Proceedings of the 23th convention of electrical and electronics engineers in Israel, September 6–7, Herzelia, Israel, pp. 189–192

Ziv J (2001) A universal prediction lemma and applications to universal data compression and prediction. IEEE Trans Inf Theory 47(4):1528–1532