Journal of Computational Methods in Sciences and Engineering 7 (2007) 45–54
IOS Press

Gene-finding with the VOM model

K.O. Shohat-Zaidenraise[b], A. Shmilovici[a,*] and I. Ben-Gal[b]
[a]*Department of Information Systems Engineering, Ben-Gurion University, P.O. Box 653, Beer-Sheva 84105, Israel*
[b]*Department of Industrial Engineering, Tel-Aviv University, Ramat-Aviv, Tel-Aviv 69978, Israel*

Abstract. We present the architecture of an elementary gene-finding algorithm that is based on a Variable Order Markov model (VOM). The VOM model is a generalization of the traditional Markov model that can cope with varying memory dependencies. The VOM model is more efficient in terms of its parameterization and therefore can be trained on relatively short sequences. Experiments with the proposed gene-finder (GF) on three prokaryotic genomes indicate its potential advantage on the detection of short genes.

Keywords: Gene finding, Variable Order Markov models, sequence analysis

Mathematics Subject Classification: 92D20, 60J20, 91B82, 60-08

1. Introduction

Due to automatic sequencing techniques, the number of sequenced bacterial genomes that are publicly available is growing rapidly.[1] It has been estimated that only 60%–80% of the genes in newly sequenced organisms have known homologues in other species [1]. Thus, comparative genomics by itself is not sufficient for automating the gene-finding procedures. Computational gene finding exploits the statistical difference in codon (base triplets coding for amino acids) usage between coding and non-coding regions of DNA [2]. A *gene prediction* (or *gene-finder*) algorithm takes as input a DNA sequence and produces as output a feature table describing the location of all candidate genes – protein-coding regions – in the sequence, including those that can not be found with a homology match to known proteins. The reliability of the gene prediction must be questioned since only a relatively small number of genes have been verified in laboratories [3].

A computational gene-finder (GF) usually consists of two stages:

a) Recognizing Open Reading Frames (ORF). ORFs are sections of DNA that contains a series of codons located between the start codon (a known initiation codon) and a stop codon (a known terminating codon). An ORF represents a potential location of a gene that encodes a protein.
b) Gene parsing due to recognition of motifs and start/stop codons. Gene parsing attempts to resolve overlapping ORFs.

*Corresponding author. E-mail: armin@bgumail.bgu.ac.il.
[1]Over 100 microbial genomes were published by the beginning of 2003.

Although many GF programs exist, the identification of genes is far from being trivial even for relatively simple bacterial genomes. Some examples for the complex issues involved in the identification process are:

– *Random ORF.* Any random sequence, as well as non-coding regions in real sequences, might contain a large number of ORFs. It is difficult to discriminate between random ORFs and short genes, therefore, many genomes are over-annotated. Skovgaad et al. [3] estimate that the verified number of protein-coding genes in organisms that were analyzed experimentally is smaller by 10%–30% in comparison to the number of genes annotated by a GF.
– *Organism-specific vs. organism-generic gene finding*. Annotating recently-sequenced genomes is often attempted by utilizing GF programs that were trained on datasets of sequences taken from *different* organisms. Long ORFs that are found in this manner are assumed reliable, and their specific codon usage is often used to bootstrap a GF retraining process [4]. Yet the obtained accuracy of such a GF procedure strongly depends on the characteristics of the training dataset and its similarity to the target genome.
– *Resolving overlapping ORFs*. Some genomes contain 'real' gene overlaps that might be mistakenly identified as a byproduct of an under-trained or an over-trained GF.

Statistical GF algorithms initially construct different statistical models for the coding and the non-coding regions of DNA and use these models in a later stage to annotate the ORFs. An over parameterized model needs unpractical large and highly-accurate training dataset, and yet may over-fit this training set. An under-fitted model, on the other hand, can not capture some of the available information in the training sequences and might result with short ORFs of a limited reliability. This paper focuses only on improving the basic ORF detection process. Other algorithms (such as [4,5]) can be used for resolving overlaps between adjacent ORFs.

Markov models are well-applied to the analysis of microbial sequences. Roughly speaking, a Markov chain is a sequence of random variables X_i, for which the conditional probability distributions depends only on a *fixed* number of preceding k variables $X_{i-1}, \ldots, X_{i-k}$. In DNA sequence analysis, the Markov chain contains the conditional probability of any base $b \in \{A, C, G, T\}$ in the sequence given its k prior bases. These preceding k bases are commonly referred to as the '*context*' of base b in the sequence. For example, a fifth-order Markov model, which is the underlying model in *GeneMark* [6], uses the five previous bases in the sequence to predict the next base. The number of parameters in the Markov model grows exponentially with its order. In general, a kth-order Markov model of a DNA sequence requires the estimation of 4^{k+1} probability parameters from the training data (e.g., 4096 probability parameters for a fifth-order model). Ideally, larger values for k are preferable. However, for small training datasets such parameterization often results in model over-fitting to the training set and poor variance-bias tradeoff [7]. To estimate the model's probability parameters, many occurrences of almost all the possible k-mers must be present in the training dataset. Yet, in a typical microbial genome the frequencies of different sequences of variable length might change considerably. For example, some 4-mers will not occur too frequently in the genome to obtain a reliable estimate of the probability of the next base, while some 8-mers may occur frequently enough to obtain very reliable estimates. Thus, constructing such fixed-order models accurately is a difficult task when there is insufficient training data. In fact, the parameterization for the coding regions is even more difficult – the estimation of probabilities is inhomogeneous, thus, depend on the base position in the codon. Accordingly, gene-finders construct three separate models, one for each codon position. This is known as a 3-periodic Markov model [6].

The Variable-Order Markov (VOM) Model overcomes the frequency changes by enabling a variable memory order for each base in the sequence, depending on its context. The VOM model is a true generalization of the traditional fixed-order Markov model [7] and is more efficient in terms of its parameterization. Therefore, the VOM model can be trained on relatively small datasets in comparison to the fixed-order Markov models. Given a context of length k, the VOM model estimates the conditional probability distribution for the (k+1)th base, by using as many of the preceding bases in the training dataset as required. Different algorithms were proposed for training VOM-like models from datasets. Begleiter et al. [8] describes six different VOM-like algorithms. For example, the Interpolated Markov Model (IMM), which is implemented in the *Glimmer* GF [4,9], uses a combination of 3-periodic nonhomogenous Markov models, weighing each model according to its predictive power.

In this paper we present a new approach for gene finding, based on a VOM model that was trained via the *context tree algorithm* of Rissanen [10]. Unlike the heuristic justification for the use of the IMM, the use of the VOM is justified by the superior convergence properties of the context tree algorithm as studied in Ziv [11]. Furthermore, a related scheme of the proposed model – Prediction by Partial Match (PPM) – was found to be one of the two outperforming algorithms in sequence prediction tasks [8]. The contribution of this paper is twofold. First, it demonstrates the potential use of VOM models to annotate DNA sequences. Second, it showcases the ability to predict short genes that may be undetectable by other GF programs. We expect that the proposed VOM GF, when integrated with other gene parsing algorithms and applied to newly sequenced bacterial genomes, can contribute to conventional gene-finders that are based on the fifth-order Markov models.

The rest of this paper is organized as follows: Section 2 introduces the VOM training algorithm and the VOM-based GF; Section 3 presents some comparative genome experiments; and Section 4 concludes with a short discussion.

2. Computational methods

This section sketches the implementation of the VOM GF. It outlines the context tree algorithm, its implementation within a simple GF, and describes some preliminary experiments for the optimization of the algorithm's parameters.

2.1. The context tree model

Following the notation in Ziv [11], consider finite-alphabet sequences of symbols $X_{-N}^{m} \equiv X_{-N}, \ldots, X_0, \ldots, X_m$ (sequence bases in our case) emitted by a stationary source with unknown properties, where each symbol X_i belongs to an alphabet A with cardinality $|A|$. The *context tree* [10] can be viewed as a data structure which is used to store the probability of the different symbols conditioned on their prefix contexts. The tree assigns a context for each symbol in the sequence. It has a root node on top, from which the branches are developed downwards. Each node has at most $|A|$ children with differently labelled ingoing edges. 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 all the $|A|$ children). Each node contains $|A|$ conditional probability parameters of symbols given the context, which is represented by the *reversed* path from that node to the root (see also [7,12]).

The construction of the context tree contains two stages. In the first stage, the tree is grown from its root downwards based on the training sequence X_{-N}^{0}. During this stage, counts in each node are updated to represent the conditional frequency of the symbols given their context. The counts denote the number of instances where symbol X_i follows the context $X_{i-1-K_{\max}}^{i-1}$ in the training sequence X_{-N}^{0}. $K_{\max}$ is the initial tree depth prior to any pruning, and it is determined by practical memory capacity

Outline for the Construction of the Context-Tree (VOM algorithm)

***Tree growing stage*:**

***Step* 0.** Start with the root as the initial tree, with all symbol counts equal to zero.

***Step* 1**. *Counter update*: Recursively, having constructed the current tree from the current sequence, read the next symbol X_i in the sequence. Traverse the tree along the path defined by the context X_{-k}^{0} and increment the count of the symbol X_i in all nodes until the deepest node is reached.

***Step* 2**. *Tree growth*: If the last updated count is at least 1 and the depth of the node is $k < K_{\max}$ where $K_{\max} \le \log(N+1)/\log(|A|)$, create a new node of depth $k = k+1$ and initialize all symbol counts to zero except for the symbol X_i whose count is set to 1.

***Tree pruning stage*:**

***Step* 3.** Estimate the KL divergence of the distribution of symbols between a leaf of depth $k(leaf)$ and its parent node of depth $k(leaf)-1$:

$$KL(leaf) = \sum_{X_i \in A} P\left(X_i \middle| X_{i-k(leaf)}^{i-1}\right) \log_2 \left(\frac{P\left(X_i \middle| X_{i-k(leaf)}^{i-1}\right)}{P\left(X_i \middle| X_{i-k(leaf)-1}^{i-1}\right)} \right).$$

Repeat for all leaves.

***Step* 4.** Prune the leaf if $KL(leaf) \ge C(|A|+1)\log(N+1)$, where the logarithm is to the base 2, and the default value for the pruning coefficient C is C=2. Practically, this pruning step keeps the leaf only if its symbols' distribution. is sufficiently different from the symbols' distribution in its parent node.

***Step* 5.** If all leaves are left unpruned stop. Otherwise, go back to step 3 and repeat for all the pruned leaves.

Fig. 1. Outline of the context-tree algorithm.

constraints [7]. In the second stage the tree is pruned to obtain its variable-length branches. The pruning is based on the *Kullback-Leibler* (*KL*) divergence measure for the conditional probabilities of symbols between a child node and its parent node. If the *KL* divergence measure is smaller than a pre-selected pruning threshold, the child node is pruned. A small *KL* divergence implies that there is no significant change in the symbols' distribution when using the reduced order of the model, or in other words, that the larger model order, which is represented by the child node, does not add much information and can be pruned without practically affecting the prediction probability.

Once the context tree is pruned, a pseudo-count is added to all the tree counts to compensate for unobserved subsequences with zero counts in the tree [7,12]. Finally, the obtained counts in the pruned tree are used to estimate the conditional probability $P(X_i|X_0^{i-1})$ for prediction or compression purposes.

The outline of the context-tree$\backslash$VOM algorithm, which we use in this paper, is given in Fig. 1 below. The complete details of the algorithm that has a linear complexity in the sequence length can be found in Ben-Gal et al. [12].

The pruned context tree model is equivalent to a set of conditional probabilities of symbols, along with their conditioning contexts. The size of the final context tree model is determined by the value of

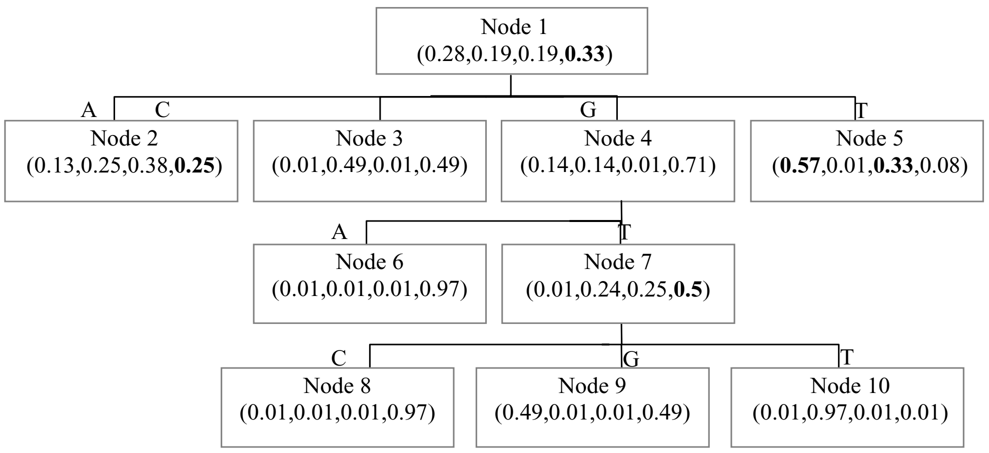

Fig. 2. The VOM tree generated from a DNA sequence. The empirical probabilities of the nucleotides in each node are ordered alphabetically as (*A, C, G, T*).

the pruning coefficient C. Rissanen [10] recommended a pruning coefficient value of $C = 2$ for best compression (this value was also found to be a good choice in our experiments [13]). The truncation process is intended to avoid over-fitting of the model to the training sequence. If the truncation process is too aggressive (selecting a high value for C), it removes contexts that represent the occurrences of rare events with small count values, smoothes out their predictions, and reduces the overall prediction quality of the model.

In our experiments, the VOM algorithm was implemented by a MATLAB script with $K_{\max} = 9$. Figure 2 represents an example of a VOM tree of a certain training set of sequences.

For illustration purpose, let us compute the likelihood of the sequence TATGT based on the VOM tree in Fig. 2: $P(TATGT) = P(T) \cdot P(A|T) \cdot P(T|TA) \cdot P(G|TAT) \cdot P(T|TATG) \cong P(T) \cdot P(A|T) \cdot P(T|A) \cdot P(G|T) \cdot P(T|TG) = 0.33 \cdot 0.57 \cdot 0.25 \cdot 0.33 \cdot 0.5 =$ 0.078 (respectively, by the bolded parameters in nodes 1, 5, 2, 5, & 7). In comparison, the likelihood of the sequence based on the unconditioned distribution of the symbols (a Zeroth-order Markov model) is much smaller: $0.33 \cdot 0.28 \cdot 0.33 \cdot 0.19 \cdot 0.33 = 0.0019$. Note that given a certain context (leaves) there are nucleotides that are very likely to occur. This probability is strengthened as the tree deepens and may be used for correcting sequencing errors as explained in [14].

2.2. *The VOM based gene-finder*

The following section outlines the required steps to annotate the ORFs using the VOM model. First, the sequence is annotated into non-coding regions and coding regions of different reading frames. Then, the coding regions are matched with start and stop codons. Finally, some conflicts between the ORFs are resolved. The main steps are discussed below (further details on these steps can be found in [14]):

– *Initialization*: Prepare both the coding sequences and the non-coding sequences for the training set. Construct a VOM model for the non-coding regions and a 3-periodic inhomogeneous VOM model for the coding regions.
– *Initial annotation* (adapted from [14]): compute a likelihood measure for each sequence, by a sliding window of size $W = 54$ nucleotides – starting 26 nucleotides upstream and ending 27 nucleotides downstream of the current nucleotide. The likelihood of the window is evaluated by

using four alternative VOM models: the non-coding VOM model, and three phased realization of the 3-periodic inhomogeneous VOM model (since the codons can start at any reading frame). The maximum likelihood model defines whether the sequence is considered as coding or non-coding.

– *Smoothed annotation*: Neighboring transitions in the reading frames are eliminated with a Viterbi-like algorithm. The annotation of a sequence is the path that maximizes the likelihood of that path minus a cost penalty P, which is used to eliminate frequent frame shifts. The penalty effectively eliminates regions shorter than $\sim$12 codons. The coordinates of each coding sequence and its phase are recorded.
– *ORF construction*: Record the location of potential start and stop codons in the three different phases. Match each putative coding region with its proximal start and stop codons. Record its coordinates. No information containing potential promoter sites or other motifs is used at this stage.
– *Complementary sequence processing*: the complementary sequence is constructed in the translating direction (by converting every ‘A’ to ‘T’ and vice versa, and every ‘C’ to ‘G’ and vice versa) and the ORF detection process is repeated for the complementary sequence and their coordinates are recorded.

2.3. Optimization experiments for the VOM GF

The VOM GF contains several algorithms that can be further optimized. Now, we briefly describe preliminary optimization directions that have been tested. Further details are given in [13].

The purpose of the first set of experiments was to improve the accuracy of distinguishing coding and non-coding DNA segments. These experiments were based on the benchmark dataset of Fickett and Tung [15], which contains representative segments of the human genome. First, the effect of the pruning constant C in the VOM algorithm was investigated. It was concluded that a pruning constant of $C = 2$ provides a better accuracy with respect to the *Hexamer* (fifth-order Markov) model, while using only about half the number of parameters. Attempting to further improve the accuracy with architecture of boosted multiple VOM classifiers (see [16]) resulted only in a marginal improvement. Accordingly, this architecture was not implemented.

The purpose of the second set of experiments was to optimize the gene annotation algorithm. The experiments were conducted by using combinations of four window sizes, W, four penalty costs, P, and four different heuristics for matching the start/stop codons with the coding segments. The experiments were applied to the genome of *Synechocystis PCC6803*. Different versions of the VOM GF were constructed to annotated the genome. The *Sensitivity* (*Sn*) and *Specificity* (*Sp*) of each annotation was computed by comparing it to the annotation published in *GenBank* that was considered as an accurate benchmark source. It was concluded that the most accurate annotation is produced with a window size of $W = 54$, a penalty value of $P = 100$, and the start/stop codons that are the closest ones outside the coding segment boundary (and, thus, biased towards longer ORFs).

3. Comparative genome experiments

Table 1 presents five bacteria genomes that were annotated by the proposed VOM-based GF – with the best parameters found above. The annotation was compared with the ‘true’ annotation in *GenBank*, which is considered here as an accurate source for validation purpose.

In the first set of experiments, the proposed VOM GF was compared with two other gene-finders that are listed in Table 1: The dicodon GF of Kim [17] and *GeneMark.fbf* [6,18]. The VOM model

Table 1
Genomes used for comparison experiments

Species	Accession Number	Length
Synechocystis PCC6803 (SP)	NC_000911	3,573,470
Pyrococcus horikoshii (PH)	NC_000961	1,738,505
Bacillus subtilis (BS)	AL009126	4,214,814
Mycobacterium tuberculosis (MT)	AL123456	4,411,529
Helicobacter pylori (HP)	AE000511	1,667,867

Table 2
Comparing to Kim [17] and to *GeneMark.fbf* [6]

Genome	Gene-finder	Sn	Sp	Sn+Sp
SP	VOM GF	0.9627	0.8774	1.8401
	Kim GF	0.9640	0.9850	1.9490
	GeneMark	0.9696	0.9970	1.9666
PH	VOM GF	0.7356	0.8769	1.6125
	Kim GF	0.9730	0.9390	1.9120
	GeneMark	0.9777	0.9027	1.8808
BS	VOM GF	0.7971	0.9630	1.7601
	Kim GF	0.9750	0.9770	1.9520
	GeneMark	0.9554	0.9917	1.9471

was trained on the GENIE [20] human genome and not on a bacteria database. The dicodon GF in Kim [17] is based on an algorithm similar to the proposed GF, except that instead of a VOM model the author uses a dicodon fifth-order Markov (Hexamer) model for the coding segments. Kim's GF also includes a self-learning mechanism. The *GeneMark.fbf* is a state-of-the-art GF that combines various algorithms, including motif identification [19]. The results of the first group of experiments based on the first three organisms are presented in Table 2. There is no doubt that the proposed VOM GF is inferior in comparison to the other two GFs. Possible explanation for this inferiority can be attributed to the lack of a learning mechanism that the others GFs have, and to the fact that the current version of the VOM GF has no mechanism for resolving overlapping genes. However, an interesting observation is that despite the average inferior performance of the proposed VOM GF, surprisingly it outperforms the other GF when searching short genes in these bacterial Genomes as described next.

Table 3 compares the annotated genes by both the VOM GF and the *GeneMark* GF to the ‘true’ genes annotation in GeneBank. The Table reveals an advantage of the VOM GF for the identification of short genes, which are defined as coding regions with less than 120 base pairs. As seen in Table 3, considerably more short genes were discovered by the proposed VOM GF than by *GeneMark.fbf*. Table 4 compares the percent (%) of nucleotide-wise false positive (FP) and the false negative (FN) rates of the VOM-based GF and *GeneMark.fbf*. The FP rate of the VOM GF is higher or equal to that of the *GeneMark.fbf*, yet the FN rate is much higher. These values partially explain the differences in the sensitivity and specificity of the compared GFs as indicated in Table 2. Nevertheless, we claim that combining the VOM GF algorithm with other identification features (such as motif detection) might improve the final accuracy of the annotation.

In an initial analysis of a newly sequenced genome, the appropriate statistical models of the coding and non-coding regions are unknown. Yet, one has to train the model based on a given training set which is assumed “close” enough to the newly sequenced genome. Evidently, using an inappropriate model reduces the accuracy of the GF. The purpose of the second set of experiments is to analyze the performance of differently trained VOM models when implemented to annotate a new prokaryotic genome. The following cross-validation experiment was performed for each genome listed in Table 1:

Table 3
Short Genes Summary

Genome	Identification	The VOM gene-finder	Gene-Mark
SP	Full identification	14	3
	Partial identification	1	0
	Un-identified genes	0	12
PH	Full identification	24	0
	Partial identification	17	0
	Un-identified genes	7	48
BS	Full identification	65	5
	Partial identification	28	1
	Un-identified genes	12	99

Table 4
Comparing the FP Rates (%)

	SP		PH		BS	
	FP	FN	FP	FN	FP	FN
GeneMark	0.26	2.65	8.87	1.87	0.72	3.99
VOM GF	11.7	3.24	8.68	22.25	2.74	18.13

Table 5
The total length of the coding/non-coding sequences used for training the foreground/background models and the resulting number of parameters

#	Tested species	Coding	Non-coding	#Par. Coding	#Par. Non-coding	#Par. Total	Sn	Sp	Sn+Sp
1	SP	9,672,346	6,696,444	3421	2533	5954	0.5051	0.8835	1.3886
2	PH	10,502,504	7,700,097	3465	2548	6013	0.3366	0.8525	1.1891
3	BS	9,449,721	6,278,708	3545	2585	6130	0.8556	0.9081	1.7635
4	MT	5,037,648	6,171,476	1620	1441	3061	0.9999	0.9040	1.9039
5	HP	10,598,605	7,677,179	3397	2444	5841	0.3951	0.9415	1.3366
6	Only MT	6,277,558	2,459,500	2776	1372	4148	0.9990	0.9039	1.9039
7	Only BS	2,352,268	1,865,485	632	609	1241	0.9964	0.8936	1.8900
8	GENIE dataset	1,367,982	220,266	241	2341	2582	N/A	N/A	N/A

first, a VOM GF was trained on a combined training set consisting of *all the other four* genomes, then, the obtained VOM GF was used to annotate the *fifth* genome. The VOM GF annotation was compared with the considered 'true' annotation in *GeneBank*. We assumed that the motifs on the main strand and on the complementary strand of the same organism have similar characteristics; therefore, we trained the model based on data taken only from the main strand. Similarly, the non-coding sequences were extracted from the 'real' non-coding sequence and were used for training the background model. The first five rows in Table 5 present the annotation based on the combined training set. The lengths of both coding and non-coding sequences that were extracted from the other four organisms are presented in columns 3 and 4, and the numbers of parameters of the respective VOM models are presented in columns 5 and 6. The sensitivity and specificity of the VOM GF annotation with respect to *GeneBank*'s annotation are given in the last three columns. Since the foreground model was constructed from the indicated genes, the overlapping sequences of the overlapping genes were taken into account twice. For example, note that Mycobacterium tuberculosis (MT) has a coding sequence that is almost 1.5 times its entire length. This inconsistency resulted from the large number of overlapping genes in this organism.

Row #8 in Table 5 presents the size of the datasets used (in Table 2) for training the VOM models on the GENIE human datasets [20]. Surprisingly, comparing the results (columns 8, 9, 10) to those of

Table 2, reveals that the VOM models that were trained on the human dataset produced a *higher* accuracy than the VOM models that were trained on the prokaryotic datasets. We do not have a full explanation for this paradoxical result. Yet, note that the GENIE dataset – unlike the combined bacterium datasets in the first five rows – is a smaller and a “cleaner” dataset. We suspect that the fact that repeating and similar sequences were not filtered out from the bacterium training dataset, probably resulted in VOM models that were over-fitted to the training set. To verify this assumption, further experiments (#6, #7) were conducted, in which we cross-tested the VOM models for two different genomes – *Mycobacterium tuberculosis* (MT) and *Bacillus subtilis* (BS) – belonging to the same family of bacterium (gram-positive bacterium). A significant improvement was detected for the *Bacillus subtilis*, while no improvement was observed for the *Mycobacterium tuberculosis*. Rows #6 and #7 in Table 5 present the lengths of both the coding and the non-coding sequences taken from the MT and the BS genomes, respectively. Once again, note that the sequences used for training the different VOM models in these rows do not have the same sizes. The training dataset for the BS was considerably smaller than the training set of the MT – a fact which may affect the accuracy of the fitted VOM GF. Moreover, note that most of the BS genes found in *GenBank* are computationally predicted with no experimental evidence. This fact also may influence the achieved accuracy when using this genome for training the VOM model since it contains noisier and longer sequences.

4. Conclusions

We introduce a potential approach for gene finding based on the variable-order Markov (VOM) Model. Although the VOM-based GF is relatively simple and self contained – e.g., it does not consider motifs other than the start and the stop codons – it can be applied for gene finding along with the commonly used Markov based GFs.

The VOM GF seems to detect some short genes even when the available training set is relatively small. In presence of a large training dataset, however, the performance of the VOM GF seems to be less competitive. Therefore, it is proposed to investigate a direction when the VOM GF is used as a complementary tool to conventional GFs that excel in detecting longer genes. The current version of the GF was implemented as semi-manual prototype software. Integrating it with other learning algorithms (such as self learning, advanced gene parsing, motif detection, homology, detection, etc.) is expected to improve its performance [21].

References

[1] D. Frishman, A. Mironov, H.W. Mewes and M. Gelfand, Combining diverse evidence for gene recognition in completely sequenced bacterial genomes, *Nucleic Acids Research* **26**(12) (1998), 2941–2947.

[2] C.B. Burge and S. Karlin, Finding the genes in genomic DNA, *Current Opinion in Structural Biology* **8**(3) (1998), 346–354.

[3] M. Skovgaad, L.J. Jensen, S. Brunak, D. Ussery and A. Krogh, On the total number of genes and their length distribution in complete microbial genomes, *Trends in Genetics* **17**(8) (2001), 425–428.

[4] A.L. Delcher, D. Harmon, S. Kasif, O. White and S. Salzberg, Improved microbial gene identification with Glimmer, *Nucleic Acids Research* **27**(23) (1999), 4636–4641.

[5] T.S. Larsen and A. Krough, EasyGene – a prokaryotic gene finder that ranks ORFs by statistical significance, *BMC Bioinformatics* (2003), 4–21.

[6] M. Borodovsky and J. McIninch, Genmark: Parallel Gene Recognition for both DNA Strands, *Computers & Chemistry* **17**(2) (1993), 123–1333, *GeneMark*, Available Online: http://opal.biology.gatech.edu/GeneMark/fbf.cgi.

[7] I. Ben-Gal, A. Shani, A. Gohr, J. Grau, S. Arbiv, A. Shmilovici, S. Posch and I. Grosse, Identification of Transcription Factor Binding Sites with Variable-order Bayesian Networks, *Bioinformatics* **21**(11) (2005), 2657–2666.

[8] R. Begleiter, R. El-Yaniv and G. Yona, On Prediction Using Variable Order Markov Models, *Journal of Artificial Intelligence* **22** (2004), 85–421.

[9] *Glimmer* gene-finder, Available Online: www.tigr.org/software/glimmer.

[10] J. Rissanen, A universal data compression system, *IEEE Transactions on Information Theory* **29**(5) (1983), 656–664.

[11] J. Ziv, A universal prediction lemma and applications to universal data compression and prediction, *IEEE Transactions on Information Theory* **47**(4) (2001), 1528–1532.

[12] I. Ben-Gal, G. Morag and A. Shmilovici, CSPC: A Monitoring Procedure for State Dependent Processes, *Technometrics* **45**(4) (2003), 293–311.

[13] K.O. Shohat-Zaidenraise, Gene finding via context learning models, Thesis submitted to Tel-Aviv University, Israel, February 2004, available upon request.

[14] A. Shmilovici and I. Ben-Gal, Using a Compressibility Measure to Distinguish Coding and Noncoding DNA, *Far East Journal of Theoretical Statistics* **13**(2) (2004), 215–234.

[15] J.W. Fickett and C.S. Tung, Assessment of Protein Coding Measures, *Nucleic Acids Research* **20**(24) (1992), 6441–6450.

[16] O.R. Duda, P.E. Hart and D.G. Stork, *Pattern Classification*, Chapter 9.5, John Wiley & Sons, 2001.

[17] J. Kim, *A Study on Dicodon-oriented Gene Finding using Self-Identification Learning*, A thesis submitted to School of Knowledge Science, Japan Advanced Institute of Science and Technology, February 2000.

[18] A.M. Shmatkov, A.A. Melikyan, F.L. Chernousko and M. Borodovsky, Finding prokaryotic genes by the 'frame-by-frame' algorithm: targeting gene starts and overlapping genes, *Bioinformatics* **15**(11) (1999), 874–886.

[19] J. Besemer, A. Lomsadze and M. Borodovsky, GeneMarkS: a self-training method for prediction of gene starts in microbial genomes. Implications for finding sequence motifs in regulatory regions, *Nucleic Acids Research* **29**(12) (2001), 2607–2618.

[20] GENIE data-sets, from Genbank version 105, 1998. Available: www.fruitfly.org/seq_tools/datasets/Human/intron_v105/.

[21] A. Shmilovici and I. Ben-Gal, Using a VOM model for reconstructing potential coding regions in EST sequences, *Journal of Computational Statistics* **22**(1) (2007), 49–69.