Nucleic Acids Research, v. 22 (1994) p. 4768-4778

A Hidden Markov Model that finds genes in *E. coli* DNA

Anders Krogh
Nordita
Blegdamsvej 17, DK-2100 Copenhagen, Denmark
email: krogh@nordita.dk

I. Saira Mian
Sinsheimer Laboratories
University of California, Santa Cruz, CA 95064, USA
email: saira@fangio.ucsc.edu

David Haussler
Computer and Information Sciences
University of California, Santa Cruz, CA 95064, USA
email: haussler@cse.ucsc.edu

February 23, 1995

Keywords: Hidden Markov Models, DNA, *E. coli*, Database Annotation, EM algorithm.

Abstract

A hidden Markov model (HMM) has been developed to find protein coding genes in *E. coli* DNA using *E. coli* genome DNA sequence from the EcoSeq6 database maintained by Kenn Rudd. This HMM includes states that model the codons and their frequencies in *E. coli* genes, as well as the patterns found in the intergenic region, including repetitive extragenic palindromic sequences and the Shine-Delgarno motif. To account for potential sequencing errors and or frameshifts in raw genomic DNA sequence, it allows for the (very unlikely) possiblity of insertions and deletions of individual nucleotides within a codon. The parameters of the HMM are estimated using approximately one million nucleotides of annotated DNA in EcoSeq6 and the model tested on a disjoint set of contigs containing about 325,000 nucleotides. The HMM finds the exact locations of about 80% of the known *E. coli* genes, and approximate locations for about 10%. It also finds several potentially new genes, and locates several places were insertion or deletion errors and or frameshifts may be present in the contigs.

1 Introduction

Sequencing of the genomes of organisms and organelles has and will continue to produce large quantities of complex map and DNA sequence data. The development of algorithms, techniques, software and databases is crucial in accumulating and interpreting these data in a robust and "automated" manner. Sequencing of the *E. coli* genome is now about 50% complete [1, 2] and as such, it serves as an important testbed for both laboratory and computer analysis techniques. Here we describe a new computer method for locating the protein coding genes in unannotated *E. coli* contigs and translating them into protein sequences.

There are two principal methods for finding genes, both of which have been incorporated into systems that analyse eucaryotic DNA [3]. The first locates signals in DNA like promoter sequences and splice junctions using techniques such as neural networks [4, 5, 6] or statistical methods [7, 8, 9]. The second approach scores a certain window of DNA in various ways in order to decide whether the window belongs to a coding or a non-coding region (reviewed in [10]). Staden and McLachlan [11, 3] proposed deviation from average codon usage as a way of determining the probability that the window is coding or not. Later, Gribskov *et al.* [12] used a similar measure as a part of their "codon preference plot", but their measure did not require the knowledge of an average codon usage from other sources. Most other scoring methods are related to codon usage in some way [13, 3]. Recently, neural networks [4, 14, 15, 16] and Markov chains [17, 18, 19] have been used to analyze coding (and non-coding) regions. In particular, the program GeneMark [20] finds genes in *E. coli* DNA using a Markov model for the coding region related to the one discussed here, and a very simple Markov model for the non-coding regions. Whether looking for signals in the DNA or using window scoring, there remains the problem of combining all the scores and/or signals detected in a given contig to produce a coherent "parse" into genes separated by intergenic regions. The output of this final parsing step could be a list of genes, each represented by its begin and end position within the contig. Snyder and Stormo have recently proposed an elegant dynamic programming method to accomplish this final step [21]. Other more linguistically motivated approaches to this kind of sequence parsing problem are described in [22, 23, 24, 25].

One aim of this paper is to combine all the aforementioned methods for locating protein coding regions (the search for initiation signals, the scoring of possible coding regions, and the final dynamic programming to get the best parse) in a single simple framework of Hidden Markov Models (HMMs). HMMs have been used to analyse DNA [18], to model certain protein-binding sites in DNA [8, 9] and in protein analysis [26, 27, 28, 29, 30, 31, 32]. The HMM we use to find genes in *E. coli* is much larger and more complex than those used in the early HMM work. Since only one strand is modelled, the HMM is applied twice, once to the direct strand and then to the complementary strand. The basic HMM architecture is identical to our earlier work [29], but here it is organised into a series of looping structures (Figure 3) containing explicit submodels for each of the 64 codons and for gene overlaps. It allows for the possiblity of insertions and deletions of individual nucleotides within a codon because such errors may result in completely or partially incorrect translated protein sequences (see [33, 34, 35]). These sequence "errors" are distinct from real frameshifts and other programmed recoding events *i.e.* alternative reading of the genetic code (see [36, 37]). In the HMM, if for example, a base is omitted such that one of the "codons" is only two bases long, the model compensates by skipping one of the bases in the codon model (similarly for insertions). To avoid modelling any DNA sequence as a gene with many errors or frameshifts, the probability of this behavior is small. Models for certain intergenic features such as repetitive extragenic palindromic sequences (REPs) [38, 39], emerged from what were initially more generic models during the HMM training procedure *i.e.* estimation of the parameters of the HMM.

The HMM was trained on approximately one million nucleotides from the EcoSeq6 database of labelled genes (Kenn Rudd, personal communication; [40]) and tested on the remainder (about 325,000 nucleotides). Since EcoSeq6 is not fully annotated yet (K. Rudd, personal communication), our results should assist in identifying the locations of new genes and highlighting errors and or inconsistencies in the data. For each contig in this test set we used the Viterbi algorithm [41, 29], a standard dynamic programming procedure for HMMs, to find its most likely path through the hidden states of the HMM. Based on the stochastic model represented by our HMM, this path was then used to define a parse of the contig into genes separated by intergenic regions. Of about 240 labelled genes in the test set, we found about 80%[1] of the sequences labeled as protein-coding genes in EcoSeq6 exactly, *i.e.* with precisely the same start and stop codons. Approximately 5% were found within 10 codons of the start codon, 5% overlap by at least 60 bases or 50% and about 5% were missed completely. For each of genes predicted by the parser but not labelled in EcoSeq6, we performed a database search using the program BLASTP [42] and the predicted protein sequence. The results indicate that many of these appear to encode known proteins. In addition, there are several instances where the HMM suggests insertion or deletion errors in the labelling of the contigs.

The most distinctive aspects of our work are the complexity of the intergenic model and the simplicity of the overall HMM framework for combining coding measures and specific sensors to produce useful parses. The Viterbi algorithm replaces the Snyder-Stormo style dynamic programming approach in this combination of coding measures and specific sensors. To demonstrate the advantages of explicitly modeling the structures in the intergenic region, we also trained and tested a much simpler HMM that did not include a sophisticated intergenic model, but instead relied only on the statistics of the codon models (Figure 1). While this model performed quite well also (about 70% exactly correct), our more complex HMM performed significantly better.

[1] The actual percentage of exactly correct predictions on the test set is about (85%), but since performance on the training set (about 1000 genes) was only 78% exactly correct, we believe that 80% is a more realistic performance estimate.

2 Methods

A Parser with a Simple Intergenic Model

An HMM for DNA patterns generates sequences of A, C, T and Gs according to a random process. The simplest HMM used in this research is illustrated in Figure 1 and consists of a collection of rings, all connected to a central state. Each ring possesses one or more HMMs whose structure is essentially the same as that used in our work on modelling protein families [29]. There is one codon HMM for each of the 61 DNA triplets that code for amino acids as well as a ring which generates the intergenic region and its flanking stop and start codons.

The random process used by the HMM to generate a sequence of nucleotides is a random walk starting in the middle of any of the HMMs. Assume we begin at the central state and enter any of the rings by traversing one of the arrows shown in Figure 1. Each such *state transition* has an associated probability and transitions out of the central state are chosen at random according to these probabilities (they sum to one). For example, a transition leading to the AAC codon model HMM generates the three nucleotides AAC with very high probability and then, with probability 1, makes the transition back to the central state. Subsequently, a new transition out of the central state is selected randomly and independently of the previous transition. Choosing one of the 61 codon models repeatedly results in a "random gene". The gene eventually terminates upon entry into one of the rings below the central state. The probability of such a transition is fairly small.[2] One stop codon HMM generates both TAA and TGA, each according to its frequency of occurrence in *E. coli*, and the other TAG. In the simple HMM, a sequence of nucleotides representing an intergenic region are produced independently and at random by looping in the state labelled "Intergene model". Next, the start codon HMM generates either ATG, GTG or TTG, each with the appropriate probability (TTG is very rare in *E. coli*). A transition is made back to the central state and the whole process repeated *i.e.* generation of several random codons followed by another intergenic region and so on. This entire procedure produces a sequence of nucleotides that is statistically similar to a contig of *E. coli* DNA consisting of a collection of genes interspersed with intergenic regions.

Each random walk has a well-defined probability determined by the probability parameters of the HMM. This probability is inverted and employed to locate the beginning and ends of genes. For a given contig of *E. coli* DNA, the most likely random walk through the HMM that generates this sequence is calculated with a dynamic programming method known as the Viterbi algorithm (described in [41]; see also [29]). The Viterbi algorithm generates a parse of the contig, *i.e.* labels genes in the DNA by identifying portions of the path that begin with the start codon at the end of the intergenic ring, pass through several amino acid codon HMMs, and return to one of the stop codons at the beginning of the intergenic ring. The model parses a gene in one direction only and thus finds all genes on the direct strand. To locate genes on the opposite strand, the reverse complement (A and T interchanged, G

[2] This probability is roughly determined by the number of intergenic regions divided by the number of codons in a typical contig of *E. coli* DNA.

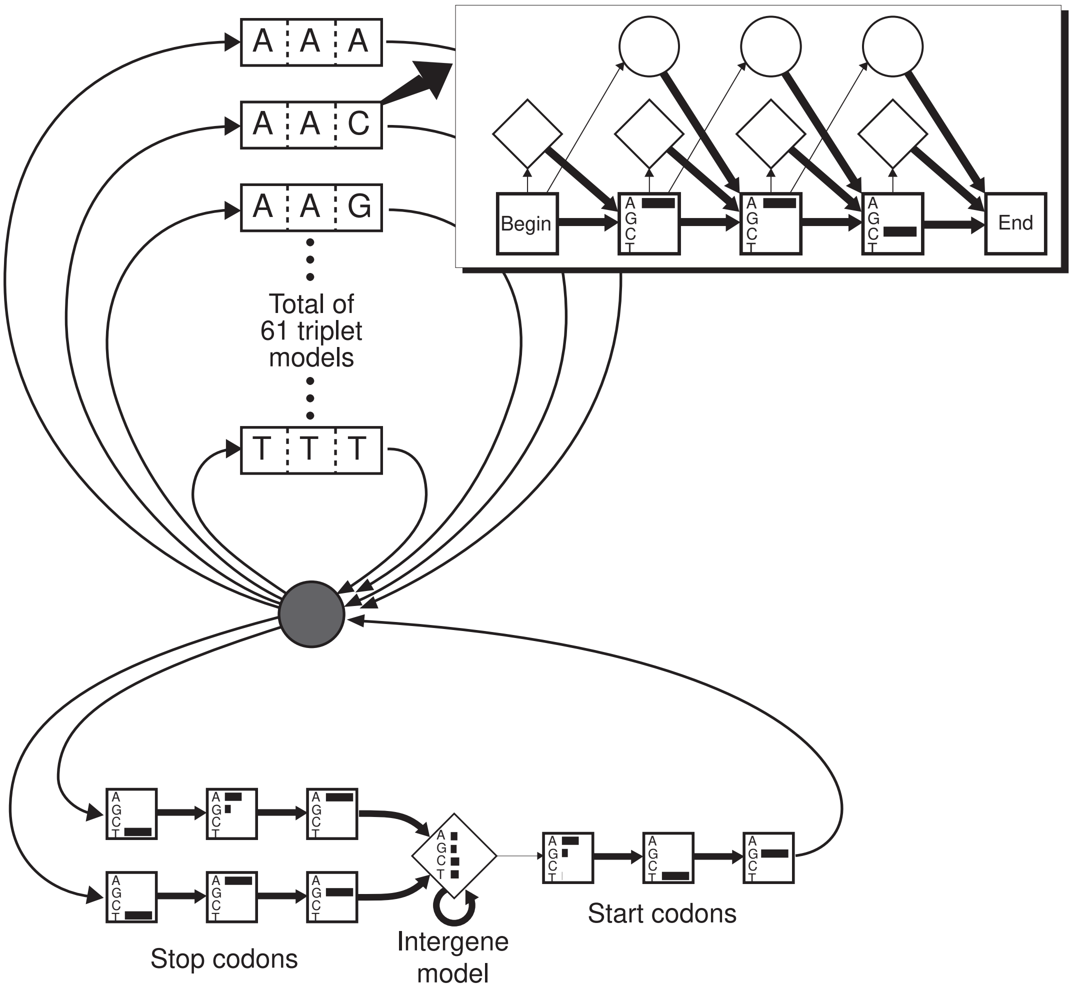

Figure 1: HMM architecture for a parser for *E. coli* DNA with a simple intergenic model. The *central state* (shaded circle), generates no nucleotides and is used to connect all the models. The 61 triplet or codon models above the central state all have identical structures, shown in detail for the codon AAC. Squares represent main states; diamonds denote a state where a nucleotide can be inserted between consecutive codon nucleotides whereas circles generate no nucleotide and can be used to delete one of the three nucleotides. The thickness of the arrows indicate the fraction of sequences making the given transistion. The insert state in the middle of the intergenic model (diamond) produces random sequences from a base distribution estimated from the actual distribution of bases in the intergenic regions of the training set. The four bases have almost the same frequency.

and C interchanged, and the sequence reversed) is parsed as just described.

The Gene Model

Codon	Aa	Usage	Random	Codon	Aa	Usage	Random	Codon	Aa	Usage	Random	Codon	Aa	Usage	Random
AAA	Lys	3.5	1.3	GAA	Glu	4.3	1.6	CAA	Gln	1.3	1.4	TAA	*	*	*
AAG	Lys	1.1	1.6	GAG	Glu	1.8	1.8	CAG	Gln	3.0	1.7	TAG	*	*	*
AAC	Asn	2.4	1.4	GAC	Asp	2.2	1.7	CAC	His	1.1	1.5	TAC	Tyr	1.4	1.4
AAT	Asn	1.4	1.3	GAT	Asp	3.2	1.5	CAT	His	1.2	1.4	TAT	Tyr	1.5	1.3
AGA	Arg	0.1	1.6	GGA	Gly	0.6	1.8	CGA	Arg	0.3	1.7	TGA	*	*	*
AGG	Arg	0.1	1.8	GGG	Gly	1.0	2.2	CGG	Arg	0.4	2.0	TGG	Trp	1.4	1.8
AGC	Ser	1.6	1.7	GGC	Gly	3.2	2.0	CGC	Arg	2.4	1.8	TGC	Cys	0.7	1.6
AGT	Ser	0.7	1.5	GGT	Gly	2.8	1.8	CGT	Arg	2.5	1.6	TGT	Cys	0.5	1.5
ACA	Thr	0.5	1.4	GCA	Ala	2.0	1.7	CCA	Pro	0.8	1.5	TCA	Ser	0.6	1.4
ACG	Thr	1.4	1.7	GCG	Ala	3.6	2.0	CCG	Pro	2.6	1.8	TCG	Ser	0.8	1.6
ACC	Thr	2.5	1.5	GCC	Ala	2.5	1.8	CCC	Pro	0.4	1.6	TCC	Ser	0.9	1.5
ACT	Thr	0.9	1.4	GCT	Ala	1.6	1.6	CCT	Pro	0.6	1.5	TCT	Ser	0.9	1.4
ATA	Ile	0.3	1.3	GTA	Val	1.1	1.5	CTA	Leu	0.3	1.4	TTA	Leu	1.1	1.3
ATG	Met	2.5	1.5	GTG	Val	2.7	1.8	CTG	Leu	5.7	1.6	TTG	Leu	1.2	1.5
ATC	Ile	2.7	1.4	GTC	Val	1.5	1.6	CTC	Leu	1.0	1.5	TTC	Phe	1.8	1.4
ATT	Ile	2.8	1.3	GTT	Val	1.9	1.5	CTT	Leu	0.9	1.4	TTT	Phe	1.9	1.2

Table 1: The relative frequencies of the 64 codons (in percent) in the *E. coli* DNA training data used in this study ("Usage"). "Random" gives the corresponding values if codon usage was simply a result of the relative frequencies of the four nucleotides (A, 23.66, G, 27.89, C, 25.30, and T, 23.15). "Aa" and "*" denote amino acid and stop codon respectively.

The role of the codon HMMs in Figures 1 and 3 is similar to the role played by codon usage statistics in many other gene finding methods [3]. Codon usage statistics are far from what would be expected if they were based on randomly chosen nucleotides (see Table 1). In our model, the codons in a gene are considered random and independent. Therefore, the probability that a region is coding is simply the product of the probabilities of the individual codons. The probability of an open reading frame (ORF) consisting of codons $c_1, c_2, \ldots c_k$ and excluding start and stop codons is

$$\mathrm{Prob}(c_1, \ldots c_k) = \prod_{i=1}^{k} p(c_i), \tag{1}$$

where $p(c_i)$ is the probability of codon c_i given in Table 1 for *E. coli*. We define the gene index of an ORF to be the negative logarithm of this divided by the length of the contig,

$$I(c_1, \ldots c_k) = -\frac{1}{k} \sum_{i=1}^{k} \log_{64} p(c_i). \tag{2}$$

The average value for a typical *E. coli* gene is equal to the entropy of the *E. coli* codon probability distribution.[3] Using an estimate of this distribution obtained from our training set (Table 1) yields

$$\mathrm{average}(I) = 0.935. \tag{3}$$

For genes in the training set, relatively few have a large gene index: roughly 16% have an index greater than 0.96, 7% greater than 0.98, and only about 2.5% have a gene index

[3]Since logarithm base 64 is used, the entropy of any codon distribution will be at most 1. Therefore, typical genes will have an index less than 1.

larger than 1.0, see Figure 2. This gene index will be used to rank predictions and resolve ambiguities of the predictions by the HMM.

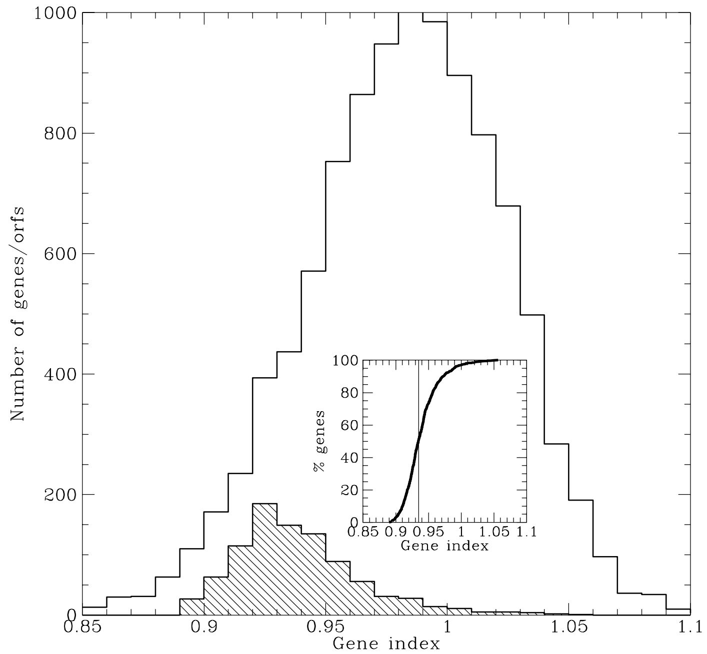

Figure 2: Distribution of gene index for 920 genes in the training set (lower dark histogram). Any genes with a length not divisable by 3 or with unusual start codons (not ATG, GTG and TTG) or stop codons (not TAA, TAG, and TGA) are not counted. The inset shows the cumulative distribution, *i.e.* the fraction of genes with a gene index below a certain value; the vertical line denotes the average gene index. For comparison the larger histogram shows the gene index for orfs (open reading frames) in the training data. The following criteria were used for selecting orfs: 1) they do not have the same stop codon as a labeled gene, 2) the length is more than 100 base pairs, 3) if several orfs had the same stop codon, only the one with the lowest gene index was included.

The gene model uses the codon probability as the probability of making a transition into the corresponding codon model. Assume that a particular path through the HMM starts in the intergenic model and goes through the start codon model before looping in the gene model k times (producing k codons), and then enters one of the stop codon models before ending in the intergenic model. This corresponds to an ORF of length k (not counting start and stop codons) flanked by intergenic regions. The probability of that path will contain the probability for the ORF as given in Equation 1. Thus, using the Viterbi algorithm with such a model gives an overall parser similar to Staden and McLachlan's codon-usage method of locating genes [11], or the related method of Gribskov *et al.* [12], and then following this by a simple dynamic programming method like that of [21].

The 61 codon models are designed to generate one nucleotide triplet each. In the main states (squares), the probability of generating the letters of the codons is set to one and the others to zero. To allow for the possiblity of frameshifts and sequencing errors, insertions or deletions are modelled in the same manner that insertions and deletions are modelled in our HMMs built for protein families [29] (see Figure 1). For each of the three nucleotides in the codon independently, there is a very small probability, P_{indel}, that that nucleotide is deleted (*i.e.* missing in the sequence). Similarly, independently between each pair of consecutive nucleotides, before the first nucleotide, and after the last nucleotide, a randomly chosen nucleotide is inserted with probability P_{indel}. Experiments (data not shown) indicated that "zeroth order" codon statistics were almost as good as higher order models, for example, those incorporating statistics on which codons are likely to follow other codons. Thus, we focus on constructing good models of the intergenic regions while keeping the gene model simple. This contrasts with the work of others such as Borodovsky and McIninch [17, 20].

A Parser with a Complex Intergenic Model

The more complex HMM (Figure 3), intergenic model consists of several parts in addition to the start and stop codon models described earlier. After generating the stop codon, the model chooses either the transition to the long intergenic HMM or the short intergenic HMM, with appropriate probabilities. The short intergenic HMM tends to generate intergenic regions of lengths from 1 to 14 or so, with statistics determined from examples of such short intergenic regions in actual *E. coli* contigs. Similarly, the parameters of the long intergenic model are adjusted to capture the statistics of longer intergenic regions. The parameters of the two intergenic models were estimated from a set of known intergenic regions by a learning procedure known as the *forward-backward* algorithm. As a result of the training process, the long intergenic region develops patterns, without having to explicitly encode them. For example, it discovers a structure about 5 to 10 nucleotides before the start codon that corresponds to the well known Shine-Delgarno sequence [43] (positions marked 36–40 in Figure 4). The strong nucleotide preferences imediately following the stop codon (positions 5–18) resemble a repetitive extragenic palindrome or REP sequence [38, 39]. All of these features are considered by the Viterbi method when matching a segment of the sequence to one of the intergenic models and thus provide statistical information not used in other gene-finding methods.

Models for Overlapping Genes

The possibility of overlapping genes are dealt with by two overlap HMMs. In Figure 3, the box labelled "Overlap models" represents separate HMMs for handling overlaps of 1 or 4 nucleotides, each forming its own ring with the central state. The HMM for overlaps of 1 generates the sequences TAATG or TGATG with high probability and other sequences with very small probability. Each time this overlap model is encountered in a parse, TAA or TGA is taken to be the stop codon for one gene and ATG is the start codon for another gene (the middle nucleotide A is shared). With high probability, the HMM for overlaps of length 4 produces sequences that match the regular expression NN[AG]TGANN, where N stands for any of the four nucleotides, and [AG] means either A or G. TGA is assumed to be the stop codon of a gene extending to the left, and the triplet ATG (or GTG) the start codon of a gene extending to the right. The two Ns on either side are needed to keep the overall HMM in the correct reading frame both before and after the overlap. In the *E. coli* training data, about 75% of the overlaps were of lengths 1 or 4. Instead of modeling the remaining overlaps (greater than 4 bases) explicitly, we find them in a special post-processing step before the final parse of the contig is produced (described below).

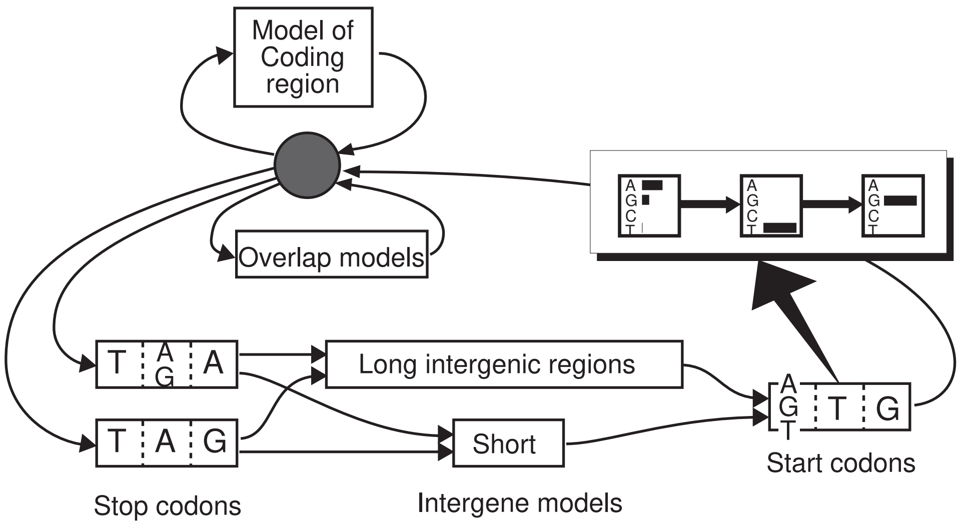

Figure 3: HMM architecture for a parser for *E. coli* DNA with a complex intergenic model. The gene model above the central state that contains the 61 triplet models is identical to the gene model of the simple parser shown in Figure 1. The detailed structure of the long intergenic model is shown in Figure 4.

Training Data

We used the EcoSeq6 database [40, 44] maintained and provided to us by Kenn Rudd (personal communication). It contains about 460 contigs of *E. coli* DNA but is not fully annotated yet because a significant amount of gene discovery remains to be done (K. Rudd, personal communication). All contigs containing genes not coding for proteins were omitted leaving 429 contigs which were then split at random into a training set of 300 contigs and a test set of 129 contigs. Because of extenstive stretches of bases of unknown identity *i.e.* those labelled "N", 5 of the contigs in the training set were subsequently modified as follows. Runs of Ns were excised leaving one contig shortened at one end (adhEeco), three split into 2 fragments (bolAeco, entDeco, fimBeco) and one split into 3 pieces (pyrGeco). Statistics for the two sets are shown in Table 2.

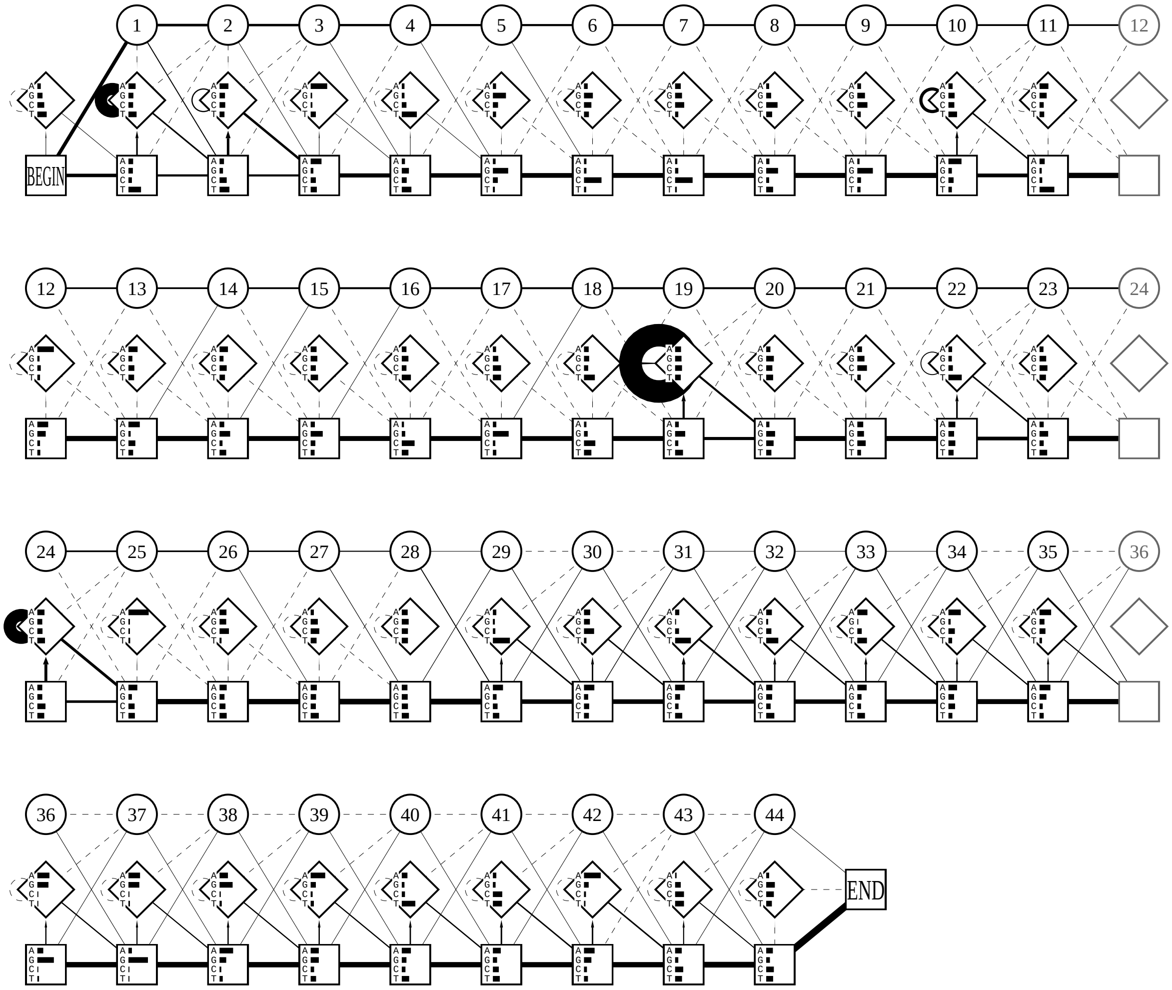

Figure 4: The model for long intergenic regions shown in Figure 3. This model was trained by the forward-backward algorithm on 424 intergenic regions of lengths larger than 10.

	Training set	Test set
Total number of contigs	300	129
Total number of characters	1,271,528	324,684
Number of genes	1007	251
Average length (internal genes)	1008	1015
Overlaping genes, length 1	50	7
Overlaping genes, length 4	40	12
Overlaping genes, length > 4	34	1

Table 2: Statistics on the 429 contigs of *E. coli* DNA used in our experiments.

To train models for the intergenic regions, all regions between two genes in the direct strand (including stop and start codons) were excised from the training sequences. Intergenic regions at the beginning or end of a contig and those with non-standard start or stop codons were removed, leaving a total of 424 for training. By standard start codons we mean ATG, GTG, and TTG, and by standard stop codons TAA, TGA, and TAG. For each contig, the complementary sequence was generated and intergenic regions between these genes generated in the same way. Note that intergenic regions often contain genes in the opposite direction.

Codon usage statistics were then calculated for the genes in the training set (Table 1). Only genes that did not begin or end a contig and had a length divisible by 3 were used. All codons that did not contain the letter "N", representing an unknown nucleotide, were counted. The relative frequencies of the 61 codons that are not stop codons were then used to set the transition probabilities in the codon models. Statistics were collected for the usage of start and stop codons in the same manner.

Parameter Estimation

Although the model contains many parameters (probabilities), all but one (P_{indel}) are determined automatically from the training contigs. Because of the problem of overfitting with such a large number of parameters, the test contigs provide independent cross validation of the results. The parameters of the short and long intergenic models were established by a learning procedure known as the *forward-backward* algorithm, a special case of the more general EM method [45]. A detailed description of the forward-backward algorithm can be found elsewhere [41]. In our implementation [29], we use the algorithm to find a maximum *a posteriori* setting of the parameters given the training sequences. The prior probabilities are exactly like those used in [29], but rather than estimating this prior from other sources, we use a uniform prior on the four possible nucleotides in each HMM state that generates a nucleotide. The only significant difference is that the distributions on the four nucleotides in what are called "insert states" in [29] are estimated from the training sequences here, rather than being "hardwired" to the uniform distribution.

The long intergenic model (Figure 4) was trained on regions with 10 or more bases between the stop and start codons, roughly the minimum length of an intergenic region with a Shine-Delgarno pattern. Because of the importance of this pattern, it was trained in two steps. First, a model was trained on the 20 (or fewer) nucleotides just before the start codon from intergenic regions longer than 10. This model, of length 15, was incorporated into a longer model and fixed while training the rest of the long intergenic model. The final intergenic model had a length of 44. This length was determined automatically during training by the "model surgery" method described in [29]. The short model was trained on sequences of length 1 to 14. (Note that some sequences were used to train both.) This model had a length of 9.

Since there is an insufficient number of examples of frameshifts and indel errors to estimate P_{indel}, the probability that a nucleotide is inserted in a codon, this manually-tunable parameter was fixed at 10^{-8} after a few experiments (this avoided modelling any DNA as a gene with many errors or frameshifts). The remaining parameters are all associated with transitions from the central state to one of the HMMs or transitions between sub HMMs. The probability of entering each codon model is set proportional to the codon usage shown in Table 1. The constant of proportionality, *i.e.* the overall probability of making a transition from the central state to one of the codon models, is called $\mathrm{P_{gene}}$. Using the data, one can estimate $\mathrm{P_{gene}}$ by $\mathrm{P_{gene}} = 1 - 1/(N_{codon} - 1)$, where N_{codon} is the average number of codons in a gene. The other parameters are estimated empirically in a similar fashion.

Post Processing

The parser does make some mistakes. For instance, it sometimes predicts a frame shift very near to a region of two overlapping genes, instead of actually predicting overlap between two genes (particularly long overlaps often lead to a "frameshift"). Another common mistake is to predict short genes entirely overlapping with a long gene in the opposite direction. Predicted genes often compete with a "gene" on the opposite strand that is in the complementary reading frame. These so-called "shadow genes" [20] arise because coding regions have an excess of self-complementary RNY (R:purine, Y: pyrimidine) type codons [46]. The codons that correspond to stop codons on the other strand (TTA, TCA, and CTA) are uncommon codons which enhance the probability of long ORFs opposite from real genes. If the possibility of stop codons is ignored, the average gene index of the complementary region is $-\sum_{i=1}^{58} p(\tilde{c}_i)\log_{64}(p(c_i))$, where $\tilde{c}_i$ is the codon complementary to c_i. The sum is only over 58 codons that do not have a stop codon as complementary codon. The result is:

$$\text{Average gene index for complementary region} = 0.964 \tag{4}$$

which is less than one and similar to the average index of a real gene (0.936).

We have devised three simple rules to minimise these errors. The parameters in these rules are rather ad hoc, but post-processing appears fairly robust to small changes (the last rule is the most sensitive). After genes have been predicted in both directions of a contig of DNA the predictions are post-processed as follows:

1. Each predicted frameshift is checked to see if there is a possible stop/start pair near by. The first stop codon up to 200 bases downstream from the frameshift that is in the reading frame used prior to the frameshift is located. If such a stop codon is found, then the nearest start codon is located (if any) up to 40 bases before or after the stop codon in the reading frame used after the frameshift. If both a stop and start codon are found the predicted gene is split into two.

2. Genes predicted at either end of a contig that are less than 100 bases long and those in the middle which are less than 20 are disregarded.

3. If two predicted genes in opposite directions overlap by more than 15 bases, one of them is suppressed. If they are both long (more than 400 bases), or if they have comparable

lengths (ratio of short to long > 0.5), the prediction with the lower gene index (as given in Equation (2)) is retained. Otherwise, the shorter of the two is suppressed (unless the longer one has already been suppressed by an even longer one).[4] Merely comparing the gene indices of the two opposite predictions is ineffective because a very short spurious prediction often has a very low gene index. One simple rule that works almost as well as is simply to always suppress the shorter of the two.

3 Results

The performances of the simple parser (Figure 1) and parser with the more complex intergenic region model (Figure 3) were evaluated by counting the number of whole genes correctly predicted before and after post-processing in both the training and test sets (Table 3). Parser mistakes on gene fragments at the ends of contigs that were less than 100 bases long were not counted, because such short end fragments generally contain too little information for reliable recognition. The table does not include a number of cases we discarded during testing. These are 19 genes which had either a stop or start codon different from the standard ones, a stop codon in the reading frame of the gene or genes with many unknown bases. Also 17 predictions subsequently identified as tRNA genes were disregarded. In order to make a fair comparison the simple parser was augmented with the two overlap models. Thus, the only difference between the simple and the more complex parsers is the model of the intergenic region.

The importance of modelling the intergenic region can be seen by comparing the results from the complex and simple parsers both with and without post-processing. In all cases, the rate of false negatives ("Not found") is approximately 5-6%, *i.e.*, the two parsers discover roughly the same number of genes. However, the complex parser has a better accuracy; more of the discovered genes are perfect or almost perfect. Thus, better modeling of sequence elements prior to the start of a gene ensures selection of the correct start of the gene in situations with many possible start codons.

The surprisingly good performance of the simple parser in terms of identifying labelled genes is accomplished at the cost of a much greater number of (possible) false positives (about 50% more than the actual number of genes, which is around 1000 for the training set and 250 for the test set). However, post-processing reduces this number to less than half without degrading the number of correctly predicted genes significantly. It seems like the post-processing is doing most of the work, choosing between ORFs in opposite directions. This provides good evidence that the post-processing rules work.

For the more complex parser, post-processing moves about 4% of the predictions from "partly" found to "perfect" (for the training set), because it resolves overlapping genes. The raw parser often predicts two overlapping genes as one long gene with a "frame shift" close to the region of the overlap (before the stop codon of the first gene). Provided the predicted

[4] In principle, this can lead to odd situations where genes suppress each other in a cascade, but this is very unlikely in practice.

Type of intergenic model	Post-processing	Data set	EcoSeq6 genes found by parser								Possible false positive
			Perfect		Almost perfect		Partly		Not found		
Complex	None	Training	731	(74.7)	57	(5.8)	141	(14.4)	50	(5.1)	665
		Test	203	(86.0)	12	(5.1)	11	(4.7)	10	(4.2)	191
	After	Training	767	(78.7)	62	(6.4)	88	(9.0)	57	(5.9)	310
		Test	201	(85.2)	13	(5.5)	8	(3.4)	14	(5.9)	82
Simple	None	Training	692	(70.8)	81	(8.3)	163	(16.7)	42	(4.3)	1524
		Test	179	(75.8)	23	(9.7)	25	(10.6)	9	(3.8)	412
	After	Training	694	(71.3)	81	(8.3)	143	(14.7)	55	(5.7)	331
		Test	174	(72.5)	22	(9.3)	23	(9.7)	17	(7.2)	98

Table 3: Performance of the parsers with simple and complex intergenic models in terms of prediction of whole genes. "Perfect" indicates cases where the starts and ends of the predicted genes are the same as those given in EcoSeq6; "Almost perfect", the start codon of the prediction is within 10 codons of that specified in EcoSeq6 (and in the same reading frame); "Partly", the prediction overlaps the labelled gene by at least 60 bases or 50%; "Not found", EcoSeq6 genes that are not predicted by the parser (false negatives); and "Possible false positive", genes that are predicted but not labeled as such in EcoSeq6. Numbers in parenthesis are in percent.

frame shift is within 200 bases of the downstream stop codon, post-processing will resolve this situation. Particularly long overlaps that are not modelled explicitly are found this way. Note that the start of the second gene is just chosen as the start codon closest to the stop codon of the other gene (40 bases upstream or downstream from the stop codon), which might not be the optimal one. As with the simple parser, the post-processing also reduces the number of possible false positives quite considerably.

The parser performs better on the test set than on the training set, which is the opposite of what one would expect if overfitting the training data was of concern. We believe that this is simply fortuitous. For instance, the test set contains only one instance of a gene overlap of more than 4 bases, whereas the training set contains 34 such instances. Note that each such instance influences the prediction of *two* genes, meaning that about 7% of the training genes are influenced and less than 1% of the test genes.

Partly discovered genes and false negatives

Table 4 gives more details on genes in Table 3 that were either "partly" found or "not found" (genes 1-101). Since EcoSeq6 is not fully annotated (K. Rudd, personal communication), some of the errors made by the parser may be incorrect labelings in the database or genuine errors in the sequences. We suspect the errors for genes 102–107 and 109–118 fall into this category because, for example, the lengths of genes 113 and 115 as given in EcoSeq6 are each not divisible by 3. The parser often makes predictions that start a few codons before or after the actual start codons. Those less than 10 codons off, "Almost perfect", were not investigated any further. In the training set, 28 predictions (2.8%) have a start codon between 10 and 20 codons from the correct one and 49 (4.9%) have a larger deviation (genes 1–84 in Table 4a that are not marked with "~"). Most of the predictions that differ by more than 20 codons occur in genes with a large gene index (those denoted with "#").

There are 13 cases of genes with inframe stop codons or stop/start codons that differ from those given in EcoSeq6 (103-107, 109-113, 115-118). The two genes ygiB (number 6 in Table 4a) and ygiA (listed as undiscovered in Table 4b) have a very large overlap of 146 bases and the parser has concatenated them into one. In four cases a gene was predicted as being two genes (108-111). There are 10 instances (86-95) of the parser predicting a "frameshift" or error, 8 of these occur in genes with a high gene index.

In Table 4b, the 13 false negatives (EcoSeq6 labelled genes that are not identified by the parser) consist of correctly predicted genes that are suppressed by shadow genes (96-101, 112-118), genes with different start/stop codons or inframe stop codons and very short genes. It is unclear why genes 98 and 99 are not found. The majority of false negatives are listed separately in Table 4c, because they have unusual codon statistics giving them an abnormally large gene index. Of the 53 false negatives in the training set, 32 had a gene index of more than 1.0, 17 had a gene index between 0.98 and 1.0, and 4 had a gene index between 0.96 and 0.98. These numbers are all fairly high compared to the average of 0.935.

Possible new genes or "false positives"

Some of the predictions considered as possible false positives may be real genes which have not been labelled yet whilst others might be spurious. We examined genes predicted by the complex parser (after post-processing) in more detail by translating each into the protein sequence and performing a database search using BLAST [42] and a non-redundant database composed of Swiss-Prot 27.0, PIR 38.0 and translated GenBank 79.0. Of 286 predicted genes, 95 matched a known protein. Some of these are known *E. coli* genes which have been not labeled in EcoSeq6 but will be in EcoSeq7 (Kenn Rudd, personal communications). Of the rest, 63 had a significant similarity to a known protein (Poisson Probability $P < 0.05$) and the rest (128) did not have any significant similarity. At the same time we became aware of similar work by Mark Borodovsky, Eugine Koonin and Kenn Rudd (personal comunications) carried out with a different method, but with strongly correlated results. Details of their results are given in their forthcoming paper. Based on further examination and analysis of the results from the parser, we found a new putative S-adenosyl-L-methionine methyltransferase domain that appears to be present in proteins from a variety of phylogenetically diverse organisms and organelles. These results are described in [47].

a)

Labelled EcoSeq6 Gene	Len.	Start of	Labelled EcoSeq6 Gene	Len.	Start of	Labelled EcoSeq6 Gene	Len.	Start of	Labelled EcoSeq6 Gene	Len.	Start of
1 glpG	831	33	22 cysM	912	54	43 yacA #	444	69	64 fhuE	2190	129
2 rnpA #	360	33	23 pcnB	1407	54	44 yjeB #	426	72	65 leuS	2583	147

3	hemB	1008	33	24	phnJ	846	54	45	mvrA ~	807	73	66	lipA	846	152
4	secD	1848	33	25	trxA	384	54	46	glgP	2430	75	67	xylE	1476	180
5	yfhC	537	33	26	deoD	720	57	47	xseA	1371	78	68	ycaE	675	183
6	ygiB	705	33	27	gcpE	1119	57	48	phnA	336	78	69	sohA #	336	195
7	fruF' ^	313	34	28	ycaG #	294	57	49	araJ #	1185	84	70	yicD #	825	198
8	rpoS ~	1089	36	29	yebD	453	57	50	aroK ~#	435	84	71	menD #	1389	204
9	rbsD	420	36	30	cyoA	948	60	51	dmsA	2358	87	72	trg ~	1608	228
10	bioD #	660	39	31	araE	1419	60	52	yjjB #	387	90	73	yggC #	474	240
11	srlQ	672	39	32	speC	2196	60	53	tdk #	618	90	74	nirC	555	252
12	ygdB #	366	42	33	recO	729	60	54	cirA	1992	96	75	bax' ^#	478	252
13	galE	1053	42	34	rho	1260	60	55	lacA #	612	96	76	celB	1254	279
14	hypE	969	42	35	fes #	1125	63	56	ychE #	549	99	77	cynT #	348	309
15	yjeC' ^	258	42	36	yfhB	573	63	57	carA	1149	105	78	rfe	774	330
16	ygjC	441	42	37	yggD	402	66	58	cdsA #	750	108	79	malS	2031	477
17	fepB	957	48	38	pcm ~	627	66	59	ybeB	210	108	80	mcrB #	1398	501
18	dnaE	3483	48	39	prs	948	66	60	fabA ~	516	108	81	ydbD' ^~#	1050	534
19	sdaA	1347	48	40	bisC	2181	66	61	glpR #	900	111	82	cadC #	1539	558
20	yhbD' ^	397	49	41	rfaQ #	969	66	62	cysB #	975	114	83	hsdS #	1395	1038
21	hisF	777	51	42	dnaA	1404	69	63	yhdG #	966	120	84	hsdR	3273	2097

b)

Labelled EcoSeq6 Gene					Predicted EcoSeq6 Gene	
	Name	Len.	Gene Index	begin-end in contig	begin-end	Frameshift or error at base N
85	thdF	1320	1.030 #	11894-10575	11317-11141	
86	ygjA	876	0.989 #	334-1209	331-1240	984
87	^ybjB'	369	0.986 #	1-369	1-323	160
88	^rhsE'	2047	0.975 #	1-2047	47-1532	1525
89	rhsD	4281	0.972 #	460-4740	460-4168	4160
90	^rhsB	4236	0.962 #	101-4336	101-3879	3858
91	rhsA	4134	0.962 #	759-4892	759-4590	4510
92	^rhsC	4194	0.961 #	101-4294	101-4109	3868
93	yjdA	2229	0.958	13444-15672	13444-15916	15664,15915
94	^ydiB'	520	0.957	1-520	5-672	318
95	~mukB	4605	0.922	459-5063	459-4909	4621
96	^nadR'	366	0.958	3403-3768	* +	
97	~pheM	45	0.957	7131-7087	*	
98	rpmJ	117	0.957	3198-3082	*	
99	ybdD	198	0.948	12848-13045	*	
100	trpL	45	0.938	12670-12626	*	
101	~uxaB'	137	0.932	150-286	* +	
102	^fepE	267	!	10230-10496	10359-x	
103	dacB	1434	!	? 993-2426 ? (25)	992-2425	
104	^~ydbA'	1129	!	1-1129 ? (15)	1-1170	
105	yadB'	600	0.958	? 2295-1696 ?	2043-1693	
106	holC	443	0.939	3595-3153 ?	3595-3137	
107	~infC	543	0.936	? 8498-7956	8390-x	
108	^ycaF'	293	1.054 #	1-293	8-115 & 112-318	
109	fdnG	3051	!	451-3501 (1)	451-1038 & 1087-x	
110	barA	2757	!	114-2870 ? (10)	114-821 & 815-2869	
111	~fdhF	2148	!	2223-76 (1)	2223-1804 & 1755-x	
112	TerC	22	1.124 #	? 270-249 ?	*	
113	~ydbB'	3497 &	0.951	? 3739-7235	* +	
114	rpsG	537	!	3984-3448	* +	
115	prfB	1099 &	!	2712-1614 (24)	* +	
116	^holA'	230	!	230-1 (3)	*	
117	ssrA	362	!	? 147-508 ? (5)	*	
118	micF	174	!	? 1477-1650 ? (1)	*	

----

c)

Gene Index	Undiscovered labelled EcoSeq6 Gene					
> 1.0	ydcA	pgpA	avtA	yebB	rfaK	~priB
	^ydcB'	^div'	mcrC	selC	rfaZ	^~fucT'
	^yzzA	dsdC	sulA	yjjC	rfaS	~rmf
	mcrA	fimB	phnQ	fruL	radC	~ycdA
	relF	yidD	^sufI'	leuL		~trkG
	yibA	fimB	yjfA	rfaL		~appY
	xylU	hisL	pyrL	yibB		~lit
> 0.98	ygiA	ycfA	fimE	pinO	rfaB	^~yahA'
	bicB	rem	tnaL	ivbL		~ompT
	cysX	yiaB	^glnD5'	rfaJ		~pheL
	^glnD3'	ygdA	^yeiA'	rfaI		~rcsA
> 0.96	glgS	hycA	yhhA	rfaY		~fecE

Table 4: Details on the "partly" and "not found" genes of Table 3 and labelled EcoSeq6 genes with possible errors (incorrect predictions of 102-118 were not counted). *(a)* Genes predicted with the start codon more than 10 codons from the correct location. Last column shows how far the predicted start is from the correct start. *(b)* Other mistakes made by the parser and possible errors in the database labeling. "begin-end" gives the nucleotide positions for the beginnings and ends of the labelled genes as given in EcoSeq6 and for the genes predicted by the parser; "x" signifies that the stop codon is in correct location. *(c)* Genes undiscovered by the parser. All have high gene indices. The symbols are as follows: "^": gene located at the beginning or end of contig; tilde: genes from the test set; "#": gene has a large gene index (> 0.96); "&": genes whose length is not divisible by 3 (note genes 107 and 111 are very short); "!": gene index not calculated because of in frame stop codon(s) or many codons with unknown bases (for example, genes 85 and 113 have 71% and 33% dirty codons respectively); "$*$": labelled gene that was not predicted; "+": predicted gene is suppressed by a shadow gene; "?": potentially mislabelled start (left hand side) and stop (right hand side) codons. The number in parenthesis is the number of stop codons in the reading frame of the gene.

4 Discussion

Here we have described a completely automated HMM based method that makes predictions about the locations of genes in *E. coli* DNA. The predictive power of the method was tested in terms of finding whole genes in EcoSeq6, a database of labelled *E. coli* DNA contigs. The HMM parser predicts about 80% of the genes correctly *i.e.* same stop/start codons as that given in EcoSeq6 and another 4.5–6% almost correctly (about 6% better than a model with a very simple treatment of the intergenic region). About 5% the genes are missed completely, almost entirely due to those genes having unusual codon statistics. Of the remaining roughly 10% of the genes, the parser makes fairly good predictions in about half of these instances. This gives a total rate of useful predictions of about 90%. The results from our parser should aid in the process of identifying the location of new genes and highlighting errors and inconsistencies in the data. Indeed, we find that many of the genes predicted by the parser but not identified in EcoSeq6 *do* correspond to existing sequences in the protein databases. Examination of the results from performing database searches on these false positives suggests the possible function of some of these and revealed a novel putative methyltransferase domain present in a phylogenetically diverse group of organisms [47].

With the current approach the parser is not very likely to perform better than 90%. Firstly, there is no reason to believe that the 5% of the genes that the parser missed because of unusual codon usage can be found with the kind of model we use for the coding region since it only looks at codon usage. Similarly, many of the roughly 5% serious errors the parser makes occur in genes with unusual codon usage. To locate these genes correctly would require a more sophisticated gene model. One significant improvement in the model of codon usage would be to take into account the non-stationary character of the G+C *vs* A+T content. It has been shown that there is a significant drift in the average G+C content in the *E. coli* genome over periods of several kilobases that cannot be accounted for solely by the change from coding to noncoding regions [48]. A new class of "Walking Markov" models has been proposed to model this phenomenon. The results of some preliminary calculations to see if extreme variations in G+C content could account for some of our erroneous predictions show that this indeed may account for some of the problems, but that it does not account for all of it. At this point, it is still unclear as to the best means to combine the walking Markov idea with the kind of hidden Markov model that we use. However, we suspect that other nonstationary aspects of the time series represented by the *E. coli* genome will also have to be taken into account.

The modularity of HMM design, exploited in modeling proteins [49], is a great advantage in building complex models to capture the structure of biological sequences. In future work, we plan to incorporate more explicit models of intergenic regulatory regions and of structural RNA coding regions. We also intend to try to integrate our protein models with HMMs at the level of DNA by having a subHMM for each of the widely occurring protein motifs and domains, so that a DNA parser could pick out proteins in a particular family at the DNA level as well. There is a dual advantage in this, because the more precise the model (e.g., modeling all the motifs instead of just the triplets in a gene, and explicitly modeling regulatory regions), the more accurate the parse. This arises because consideration of higher-level patterns constrains the parse much better than low level statistical information alone.

Post Script

An electronic mail server has been set up with the program described in this paper. It is possible to mail an *E. coli* DNA sequence to the server, and it will reply with a parse. Send a mail message to ECOPARSE@cse.ucsc.edu containing the single word "help" to obtain information on how to use the parser.

Acknowledgements

We would like to thank Ken Rudd, Mark Borodovsky, Flemming Hansen, Jacob Engelbrecht, Søren Brunak, Richard Durbin, and Harry Noller, for valuable comments on this work. Special thanks to Kenn Rudd for supplying the labeled *E. coli* sequences used in these experiments. This work was supported by NSF grants CDA-9115268 and IRI-9123692, ONR grant N00014-91-J-1162, NIH grant GM17129, and a grant from the Novo Nordisk Foundation.

References

[1] Kroger, M., Wahl, R., and Rice, P. (1993) *Nucleic Acids Res.* **21**, 2973–3000.

[2] Rudd, K. (1993) *ASM News* **59**, 335–341.

[3] Staden, R. (1990) *Meth. Enzymol.* **183**, 163–180.

[4] Lapedes, A., Barnes, C., Burks, C., Farber, R., and Sirotkin, K. (1989) In G. Bell and T. Marr, (ed.), Computers and DNA, SFI Studies in the Sciences of Complexity, volume **VII**, pp. 157–182 Addison-Wesley.

[5] Brunak, S., Engelbrecht, J., and Knudsen, S. (1991) *J. Mol. Biol.* **220**, 49–65.

[6] O'Neill, M. (1992) *Nucleic Acids Res.* **20**, 3471–3477.

[7] Stormo, G. and Hartzell, G. W. (1989) *Proc. Natl. Acad. Sci. U.S.A.* **86**, 1183–7.

[8] Lawrence, C. E. and Reilly, A. A. (1990) *Proteins* **7**, 41–51.

[9] Cardon, L. R. and Stormo, G. D. (1992) *J. Mol. Biol.* **223**, 159–170.

[10] Fickett, J. and Tung, C. (1992) *Nucleic Acids Res.* **20**, 6441–6450.

[11] Staden, R. and McLachlan, A. D. (1982) *Nucleic Acids Res.* **10**, 141–156.

[12] Gribskov, M., Devereux, J., and Burgess, R. (1984) *Nucleic Acids Res.* **12**, 539–549.

[13] Fickett, J. (1982) *Nucleic Acids Res.* **17**, 5303–5318.

[14] Uberbacher, E. and Mural, R. (1991) *Proc. Natl. Acad. Sci. U.S.A.* **88**, 11261–11265.

[15] Farber, R., Lapedes, A., and Sirotkin, K. (1992) *J. Mol. Biol.* **226**, 471–479.

[16] Craven, M. and Shavlik, J. (1993) In Proc. of the Hawaii Int. Conf. on System Sciences, Los Alamitos, CA: IEEE Computer Society Press. pp. 773–782.

[17] Borodovsky, M., Sprizhitsky, Y., Golovanov, E., and Alexandrov, A. (1986) *Molecular Biology* **20**, 1145–1150.

[18] Churchill, G. A. (1989) *Bull. Math. Biol.* **51**, 79–94.

[19] Tavare, S. and Song, B. (1989) *Bull. Math. Biol.* **51**, 95–115.

[20] Borodovsky, M. and McIninch, J. (1993) *Computers and Chemistry* **17**, 123–133.

[21] Snyder, E. and Stormo, G. (1992) *Nucleic Acids Res.* **21**, 607–613.

[22] Searls, D. B. (1992) *American Scientist* **80**, 579–591.

[23] Searls, D. B. (1994) GenLang Manual, Dept. of Genetics, U. Penn. PA 19104, .

[24] Searls, D. B. and Dong, S. (1993) In H. A. Lim, J. Fickett, C. R. Cantor, and R. J. Robbins, (ed.), Proc. 2nd Int. Conf. on Bioinformatics, Supercomputing, and Complex Genome Analysis, World Scientific : pp. 89–101.

[25] Collado-Vides, J. (1992) *Proc. Natl. Acad. Sci. U.S.A.* **89**, 9405–9409.

[26] Asai, K., Hayamizu, S., and Onizuka, K. (1993) In Proc. of the Hawaii Int. Conf. on System Sciences, Los Alamitos, CA: IEEE Computer Society Press. pp. 783–791.

[27] Stultz, C. M., White, J. V., and Smith, T. F. (1993) *Protein Science* **2**, 305–315.

[28] Baldi, P., Chauvin, Y., Hunkapillar, T., and McClure, M. (1994) *Proc. Natl. Acad. Sci. U.S.A.* **91**, 1059–1063.

[29] Krogh, A., Brown, M., Mian, I. S., Sjölander, K., and Haussler, D. (1994) *J. Mol. Biol.* **235**, 1501–1531.

[30] Brown, M. P., Hughey, R., Krogh, A., Mian, I. S., Sjölander, K., and Haussler, D. (1993) In L. Hunter, D. Searls, and J. Shavlik, (ed.), Proc. of First Int. Conf. on Intelligent Systems for Molecular Biology, Menlo Park, CA: AAAI/MIT Press. pp. 47–55.

[31] Hughey, R. (1993) Technical Report UCSC-CRL-93-14 University of California Santa Cruz, CA.

[32] Tanaka, H., Ishikawa, M., Asai, K., and Konagaya, A. (1993) In First Int. Conf. on Intelligent Systems for Molecular Biology, Menlo Park: AAAI Press. pp. 395–401.

[33] States, D. J. and Botstein, D. (1991) *Proc. Natl. Acad. Sci. U.S.A.* **88**, 5518–5522.

[34] Koop, B. F., Rowan, L., Chen, W. Q., Deshpande, P., Lee, H., and Hood, L. (1993) *Biotechniques* **14**, 442–447.

[35] Churchill, G. A. and Waterman, M. S. (1992) *Genomics* **14**, 89–98.

[36] Gesteland, R. F., Weiss, R. B., and Atkins, J. F. (1992) *Science* **257**, 1640–1641.

[37] Farabaugh, P. J. (1993) *Cell* **74**, 591–596.

[38] F. C. Neidhardt, J. L. Ingraham, K. Brooks Low, B. Magasanik, M. Schaechter, and H. E. Umbarger, (ed.) (1987) *Escherichia coli* and *Salmonella typhimurium.* Cellular and molecular biology, volume **1**, American Society for Microbiology, Washington, D.C.

[39] Stern, M. J., Ames, G. F., Smith, N. H., Robinson, E. C., and Higgins, C. F. (1984) *Cell* **37**, 1015–1026.

[40] Rudd, K., Miller, W., Werner, C., Ostell, J., Tolstoshev, C., and Satterfield, S. (1991) *Nucleic Acids Res.* **19**, 637–647.

[41] Rabiner, L. R. (1989) *Proc. IEEE* **77**, 257–286.

[42] Altschul, S., Gish, W., Miller, W., Myers, E., and Lipman, D. (1990) *J. Mol. Biol.* **215**, 403–410.

[43] Shine, J. and Delgarno, L. (1974) *Proc. Natl. Acad. Sci. U.S.A.* **71**, 1342–1346.

[44] Rudd, K. and Miller, W. (1992) In Jeffrey Miller, (ed.), A Short Course in Bacterial Genetics: A Laboratory Manual and Handbook for *Escherichia coli* and Related Bacteria., pp. 2.3–2.43 Cold Spring Harbor Laboratory Press Cold Spring Harbor, New York.

[45] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977) *J. Roy. Statist. Soc. B* **39**, 1–38.

[46] Shepard, J. C. W. (1981) *Proc. Natl. Acad. Sci. U.S.A.* **78**, 1596–1600.

[47] Krogh, A., Mian, I. S., and Haussler, D. (1993) Technical Report UCSC-CRL-93-33 University of California at Santa Cruz Computer and Information Sciences Dept., Santa Cruz, CA 95064.

[48] Fickett, J., Torney, D., and Wolf, D. R. (1992) *Genomics* **13**, 1056–1064.

[49] White, J., Stultz, C., and Smith, T. (1994) *Math. Biosciences* **119**, 35–75.