Stochastic Context-Free Grammars in Computational Biology: Applications to Modeling RNA

Yasubumi Sakakibara[†,*], Michael Brown[‡], Rebecca C. Underwood[‡], I. Saira Mian[§], David Haussler[‡]

† Institute for Social Information Science
Fujitsu Laboratories Ltd.
140, Miyamoto, Numazu, Shizuoka 410-03, Japan

‡ Computer and Information Sciences
§ Sinsheimer Laboratories
University of California, Santa Cruz, CA 95064, USA.

email: yasu@iias.flab.fujitsu.co.jp, haussler@cse.ucsc.edu

Keywords: Stochastic Context-Free Grammar, RNA, Transfer RNA, Multiple Sequence Alignments, Database Searching.

Abstract

Stochastic context-free grammars (SCFGs) are applied to the problems of folding, aligning and modeling families of homologous RNA sequences. These models capture the common primary and secondary structure of the sequences with a context-free grammar, much like those used to define the syntax of programming languages. SCFGs generalize the hidden Markov models used in related work on protein and DNA sequences. The novel aspect of this work is that the SCFGs developed here are learned automatically from initially unaligned and unfolded training sequences. To do this, a new generalization of the forward-backward algorithm, commonly used to train hidden Markov models, is introduced. This algorithm is based on tree grammars, and is more efficient than the inside-outside algorithm, which was previously proposed to train SCFGs. This method is tested on the family of transfer RNA (tRNA) sequences. The results show that the model is able to reliably discriminate tRNA sequences from other RNA sequences of similar length, that it can reliably determine the secondary structure of new tRNA sequences, and that it can produce accurate multiple alignments of large collections of tRNA sequences. The model is also extended to handle introns present in tRNA genes.

*This work was done while this author was visiting UC Santa Cruz.
榊原康文, 富士通研究所 情報社会科学研究所, 〒410-03 沼津市宮本140

1 Introduction

Attempts to understand the folding, structure, function and evolution of molecules has resulted in the confluence of many diverse disciplines ranging from structural biology and chemistry, through computer science and computational linguistics. Rapid generation of sequence data in recent years thus provides abundant opportunities for developing of new approaches to problems in computational biology such as Hidden Markov Models (HMMs) [Rab89, HKMS93, BCHM93]. In this paper, we apply *stochastic context-free grammars* (SCFGs) to the problems of statistical modeling, database searching, multiple alignment, and prediction of the secondary structure of RNA families. This approach is highly related to our previous work on modeling protein families with HMMs [HKMS93].

RNA is mostly involved in the biological machinery that expresses the genetic information from DNA to protein. Information is encoded in RNA by the linear arrangement of the four different constituent nucleotides (the primary structure). The individual nucleotides, adenine (A), cytosine (C), guanine (G) and uracil (U), interact in specific ways to form characteristic secondary structure motifs such as helices, loops and bulges. Further folding and hydrogen-bonding interactions between remote regions orient these secondary structure elements with respect to each other to form the functional system. Higher order interactions with other proteins and/or nucleic acids may also occur. In general, however, the folding of an RNA chain into a functional molecule is largely governed by the formation of intramolecular A-U and G-C Watson-Crick pairs as well as G-U base pairs.

Since base pairing interactions, most notably A-U, G-C and G-U, play such a dominant role in determining RNA structure and function, any statistical method that does not consider this will eventually encounter insurmountable problems. The problem is that if two positions are base paired in the typical RNA, then the bases occurring at these two positions will be highly correlated. Such base pairs constitute so-called *biological palindromes* in the genome. Thus although in principle HMMs which have been successfully applied to modeling protein families could be used for RNA, we strongly suspect that the more general statistical models described below will be required to obtain useful results.

The essence of the idea can be expressed most clearly in terms of formal language theory. As in the work of Searls [Sea92], we can view the strings of characters representing pieces of DNA, RNA and protein as sentences derived from a formal grammar. The simplest kind of grammar is a *regular* grammar, in which strings are derived from productions (rewriting rules) of the form $S \rightarrow aS$ or $S \rightarrow a$, where S is a *nonterminal symbol* that does not appear in the final string, and a is a *terminal symbol*, which will appear as a letter in the final string. Searls has shown base pairing in RNA can be described by a *context-free grammar* (CFG), a more powerful class of formal grammars than the regular grammar (see Section 2.1 for an example). A CFG is similar to a regular grammar but permits a greater range of productions, such as those of the form $S \rightarrow SS$ and $S \rightarrow aSa$. As is beautifully described by Searls, it is precisely these additional types of production that are needed to describe the base pairing structure in RNA[1] [Sea92]. In particular, the productions of the forms $S \rightarrow \mathtt{A}\ S\ \mathtt{U}$, $S \rightarrow \mathtt{U}\ S\ \mathtt{A}$, $S \rightarrow \mathtt{G}\ S\ \mathtt{C}$, and $S \rightarrow \mathtt{C}\ S\ \mathtt{G}$ describe the structure in RNA due to Watson-Crick base pairing. Using productions of this type, a CFG can specify the language of biological palindromes.

If we specify a probability for each production in a grammar, we obtain a *stochastic gram-*

[1]Not all RNA structure can be described by CFGs but we believe they can account for enough to make useful models. In particular, CFGs cannot account for pseudoknots, structures generated when a single-stranded loop region forms standard Watson-Crick base pairs with a complementary sequence outside the loop.

mar. A stochastic grammar assigns a probability to each string it derives. Stochastic regular grammars are exactly equivalent to HMMs. This provides an alternate way of examining HMMs and suggests an interesting generalization from HMMs to *stochastic context-free grammars* (SCFGs) [Bak79].

In this paper, we pursue a stochastic model of the family of transfer RNAs (tRNAs) by using a SCFG that is similar to our previous protein HMMs [HKMS93] but which additionally incorporates base pairing information. A SCFG that forms a statistical model of tRNA sequences can be built in much the same way as our construction of an HMM representing a statistical model of the globin protein family. We use such a model to search a database for tRNA-like sequences and to obtain a multiple alignment in the same manner as for globins. We also use the model to fold unfolded tRNA sequences.

First, in order to see how well the SCFG can model families of RNA sequences, especially their common primary and secondary structure, we derive a SCFG directly from an existing alignment of tRNA sequences. We then repeat this experiment, but this time we attempt to "learn" the parameters entirely automatically from a set of *unaligned* primary sequences. To do this, we introduce a new generalization of the *forward-backward algorithm*, commonly used to train HMMs. Our algorithm is based on tree grammars, and is more efficient than the *inside-outside algorithm*, a computationally expensive generalization of the forward-backward algorithm to train SCFGs [Bak79]. Thus we derive two grammars: the *alignment grammar*, directly derived from an existing multiple alignment of tRNAs, and the *trained grammar*, deduced by our training algorithm from a training set of tRNA sequences. For our training set, we chose 500 sequences at random from 1477 tRNA sequences in EMBL Data Library's database. These training sequences are unfolded and unaligned. We withhold the remaining 977 sequences in order to test the trained grammar on data not used in the training process.

We compare the two grammars by evaluating their abilities to perform three tasks: to discriminate tRNA sequences from non-tRNA sequences, to produce multiple alignments, and to ascertain the secondary structure of new sequences. The results show that both grammars can perfectly discriminate tRNA sequences from other RNA sequences of similar length, can produce accurate multiple alignments of large collections of tRNA sequences, and can reliably determine the secondary structure of new tRNA sequences.

Surprisingly, the trained grammar can discriminate more reliably than the alignment grammar because the trained grammar exhibits a greater gap between Z-scores of tRNAs and non-tRNAs. This is unexpected because the trained grammar is obtained using only 500 tRNA training sequences, while the alignment grammar is obtained using all 1477 aligned tRNA sequences.

Genes for tRNA often possess introns, regions that are excised out during formation of the mature tRNA molecule, i.e., the DNA sequence coding for a particular tRNA contains additional nucleotides that are not present in the RNA that folds to form the final structure. This means that when we search databank files that represent genomic sequences (such as those in GenBank), the grammar needs to be extended to handle this situation in order to correctly identify tRNAs. A useful advantage of SCFGs is that an intron grammar can be deduced separately from the plain tRNA grammar and these two separate grammars can then be combined into a single grammar. In a preliminary experiment, we use 55 sequences of introns for training a (sub)grammar to model introns, and combine two trained grammars for introns and intron-free tRNAs into a single grammar modeling tRNAs with introns. We test the grammar on the same 55 tRNA sequences with introns, and the grammar correctly identifies the positions of introns and the introns themselves in 80% of these sequences. Further work, using separate training and testing sets of larger size, is underway.

$$
\begin{array}{lllll}
\text{Productions } P = \{ & S_0 \to S_1, & S_4 \to \mathtt{U}\ S_5\ \mathtt{A}, & S_8 \to \mathtt{G}, & S_{10} \to \mathtt{G}\ S_{11}\ \mathtt{C}, \\
& S_1 \to \mathtt{C}\ S_2\ \mathtt{G}, & S_5 \to \mathtt{C}\ S_6\ \mathtt{G}, & S_8 \to \mathtt{U}, & S_{11} \to \mathtt{A}\ S_{12}\ \mathtt{U}, \\
& S_1 \to \mathtt{A}\ S_2\ \mathtt{U}, & S_6 \to \mathtt{A}\ S_7, & S_9 \to \mathtt{A}\ S_{10}\ \mathtt{U}, & S_{12} \to \mathtt{U}\ S_{13}, \\
& S_2 \to \mathtt{A}\ S_3\ \mathtt{U}, & S_7 \to \mathtt{U}\ S_7, & S_{10} \to \mathtt{C}\ S_{10}\ \mathtt{G}, & S_{13} \to \mathtt{C} \\
& S_3 \to S_4\ S_9, & S_7 \to \mathtt{G}\ S_8, & & \}
\end{array}
$$

Figure 1: The symbols $S_0, S_1, \ldots, S_{13}$ are nonterminals and A, U, G, C are terminals representing the four nucleotides.

2 Methods

2.1 Context-free grammars as models of RNA

The context-free grammar (CFG) is a more powerful class of formal grammars than the regular grammar and is often used to define the syntax of programming languages. An example CFG that generates a particular set of RNA sequences is shown in Figure 1.

A formal grammar is a set of productions (rewriting rules) that are used to generate a set of strings, that is, a *language*. The productions are applied iteratively to generate a string, a process called *derivation*. For example, the grammar in Figure 1 generates the RNA sequence CAUCAGGGAAGAUCUCUUG by the following derivation:

$$
\begin{aligned}
S_0 &\Rightarrow S_1 \Rightarrow \mathtt{C}S_2\mathtt{G} \Rightarrow \mathtt{CA}S_3\mathtt{UG} \Rightarrow \mathtt{CA}S_4S_9\mathtt{UG} \Rightarrow \mathtt{CAU}S_5\mathtt{A}S_9\mathtt{UG} \Rightarrow \mathtt{CAUC}S_6\mathtt{GA}S_9\mathtt{UG} \\
&\Rightarrow \mathtt{CAUCA}S_7\mathtt{GA}S_9\mathtt{UG} \Rightarrow \mathtt{CAUCAG}S_8\mathtt{GA}S_9\mathtt{UG} \Rightarrow \mathtt{CAUCAGGGA}S_9\mathtt{UG} \Rightarrow \mathtt{CAUCAGGGAA}S_{10}\mathtt{UUG} \\
&\Rightarrow \mathtt{CAUCAGGGAAG}S_{11}\mathtt{CUUG} \Rightarrow \mathtt{CAUCAGGGAAGA}S_{12}\mathtt{UCUUG} \Rightarrow \mathtt{CAUCAGGGAAGAU}S_{13}\mathtt{UCUUG} \\
&\Rightarrow \mathtt{CAUCAGGGAAGAUCUCUUG}.
\end{aligned}
$$

Formally, a *context-free grammar* consists of a set of nonterminal symbols N, a terminal alphabet Σ, a set P of productions (rewriting rules), and the start symbol S_0. For a nonempty set X of symbols, let X^* denote the set of all finite strings of symbols in X. Every CFG production has the form $S \to \alpha$ where $S \in N$ and $\alpha \in (N \cup \Sigma)^*$. The production $S \to \alpha$ means that the nonterminal S can be replaced by the string α.

Our work in modeling RNA uses only productions of the following forms: $S \to SS$, $S \to aSa$, $S \to aS$, $S \to S$, or $S \to a$, where S is a nonterminal and a is a terminal. Productions may have one of the following forms: $S \to aSa$, used to describe the base-pairing in RNA; $S \to aS$ and $S \to a$, used to describe a loop of unpaired bases; $S \to SS$, used to describe the branched secondary structure; and $S \to S$, (called *skip productions*), used in the context of multiple alignments, as described below.

A derivation can be arranged in a tree structure, called a *parse tree*. A parse tree represents the syntactic structure of an RNA sequence given by the grammar, and hence reflects the actual physical secondary structure. Figure 2 shows the derivation arranged in a parse tree reflecting the physical secondary structure.

2.2 Stochastic context-free grammars

In a *stochastic* context-free grammar (SCFG), every production for a nonterminal S has an associated probability value, such that a probability distribution exists over the set of productions for S. We denote the associated probability for a production $S \to \alpha$ by $\mathcal{P}(S \to \alpha)$.

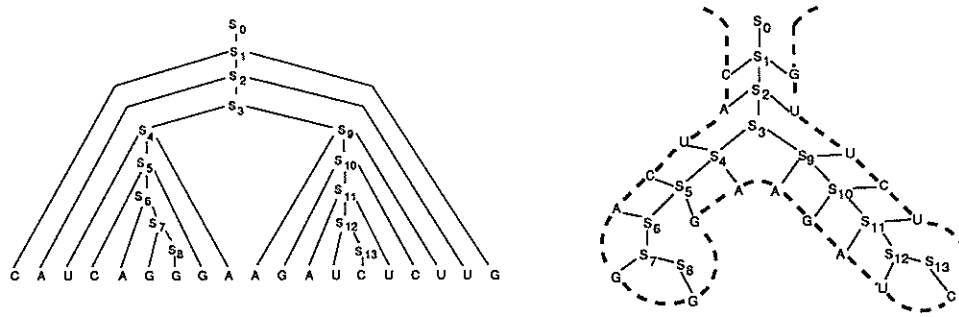


Figure 2: For the RNA sequence CAUCAGGGAAGAUCUCUUG, the grammar depicted in Figure 1 gives a parse tree (left) that reflects corresponding secondary structure (right).

An SCFG, G, generates sequences and assigns a probability to each generated sequence, and hence defines a probability distribution on the set of sequences. The probability of a parse tree can be calculated as the product of the probabilities of the productions used to generate the sequence. The probability of a sequence s is the sum of probabilities over all possible parse trees or derivations that could generate s, written as follows:

$$\begin{aligned} \text{Prob}(s \mid G) &= \sum_{\text{all derivations (or parse trees) } d} \text{Prob}(S_0 \stackrel{d}{\Rightarrow} s \mid G) \\ &= \sum_{\alpha_1, \cdots, \alpha_n} \text{Prob}(S_0 \Rightarrow \alpha_1 \mid G) \cdot \text{Prob}(\alpha_1 \Rightarrow \alpha_2 \mid G) \cdot \cdots \cdot \text{Prob}(\alpha_n \Rightarrow s \mid G) \end{aligned}$$

Efficiently computing this quantity, $\text{Prob}(s \mid G)$, presents a problem because the number of possible parse trees for s is exponential in the length of the sequence. However, a dynamic programming technique analogous to the Cocke-Kasami-Young or Early methods [AU72] for non-stochastic CFGs can accomplish this task efficiently (in time proportional to the cube of the length of s). We define the negative logarithm of the probability of a sequence given by the grammar, i.e., $-\log(\text{Prob}(s \mid G))$, as the *negative log likelihood (NLL)-score* of the sequence. This quantifies how well the sequence s fits the grammar.

Since CFGs generally have an ambiguity in that the grammar gives more than one parse tree for a sequence, and alternative parse trees reflect alternative secondary structures (foldings), a grammar often gives several possible secondary structures for one RNA sequence. An advantage of a SCFG is that it can provide the most likely parse tree from this set of possibilities. If the grammar and the probabilities are carefully designed, the correct secondary structure will appear as the most likely parse tree among the alternatives. As discussed in Section 3.3, the most likely parse tree given by the trained grammar we produce for tRNAs gives exactly the correct secondary structures for the tRNA sequences we test.

2.3 Estimating SCFGs from sequences

2.3.1 SCFGs from multiple alignments

All parameters in the SCFG (i.e., the production probabilities) could in principle be chosen from an existing alignment of RNA sequences. The method that we use to derive a SCFG

from a multiple alignment estimates a distribution of four nucleotides for each column in the alignment corresponding to a nucleotide that is not base paired, and a distribution of 16 pairs of nucleotides for each pair of columns corresponding to nucleotides that are base paired in the secondary structure.

2.3.2 EM training algorithm

In order to estimate the parameters of a SCFG from unaligned training RNA sequences, we introduce a new method for training SCFGs that is a generalization of the forward-backward algorithm, commonly used to train HMMs. This algorithm is more efficient than the inside-outside algorithm [Bak79], which was previously proposed to train SCFGs.

We have developed a method to obtain a SCFG for an RNA family like tRNA that takes only time n^2 and hence may be practical on larger RNA sequences, while the inside-outside algorithm requires time proportional to n^3, where n is the length of the model (and the typical training sequence). Our new algorithm demands folded RNA as training examples, rather than unfolded ones. Thus the base pairs in each training sequence have to be identified before the algorithm can begin iteratively reestimating the grammar parameters. If such base pair information is not available, we can use a fancier version of the algorithm, as described in Section 2.4.

A labeled tree t representing a folded RNA sequence has the shape of a parse tree, so to parse the folded RNA, the grammar G needs only to assign nonterminals to each internal node according to the productions. We assume all internal nodes in t are numbered from 1 to T (the number of internal nodes) in some order, and for an internal node n ($1 \leq n \leq T$), let t/n denote the subtree of t with root n and let $t \backslash n$ denote the tree obtained by removing a subtree t/n from t. Let the quantity $in_n(S)$ define the probability of the subtree t/n given that the nonterminal S is assigned to node n and given G, for all nonterminals S and all nodes n such that $1 \leq n \leq T$. We can calculate $in_n(S)$ inductively as follows:

1. Initialization: $in_n(\mathtt{X}) = 1$, for all leaf nodes n and all terminals X (each nucleotide). This extension of $in_n(S)$ is for the convenience of the inductive calculation of $in_n(S)$.
2. Induction:

$$in_m(S) = \sum_{Y_1,\ldots,Y_k \in (N \cup \Sigma)} in_{n_1}(Y_1) \cdots in_{n_k}(Y_k) \cdot \mathcal{P}(S \rightarrow Y_1 \cdots Y_k),$$

   for all nonterminals S, all internal nodes m, and all m's children nodes $n_1, \ldots, n_k$.
3. Termination: for the root node n and the start symbol S_0,

$$\text{Prob}(t \mid G) = in_n(S_0). \tag{1}$$

This effective calculation enables us to estimate the new parameters of a SCFG in time proportional to the square of the number of nonterminals in the grammar multiplied by the total size of all the folded training sequences. We need one more quantity, $out_n(S)$, which defines the probability of $t \backslash n$ given that the nonterminal S is assigned to node n and given G. This quantity $out_n(S)$ can be calculated in a similar manner.

Given a set of folded training sequences $t(1), \ldots, t(n)$, we can see how well a grammar fits them by calculating the probability that it generates them. This probability is simply a product of terms of the form given by (1). The goal is to obtain a high value for this quantity, called the *likelihood* of the grammar. Here we give a version of the EM method to estimate the parameters of a SCFG from folded training RNA sequences. It proceeds as follows:

1. An initial grammar is created by assigning values to the production probability $\mathcal{P}(S \rightarrow Y_1 \cdots Y_k)$ for all S and all $Y_1, \ldots, Y_k$, where S is a nonterminal and Y_i $(1 \leq i \leq k)$ is a nonterminal or terminal.
2. Using the current grammar, the values $in_n(S)$ and $out_n(S)$ for each nonterminal S and each node n for each folded training sequence are calculated in order to get a new estimate of the production probability, $\hat{\mathcal{P}}(S \rightarrow Y_1 \cdots Y_k)$=

$$\frac{\sum_{\text{sequences } t} \left(\sum_{\text{nodes } n} out_m(S) \cdot \mathcal{P}(S \rightarrow Y_1 \cdots Y_k) \cdot in_{n_1}(Y_1) \cdots in_{n_k}(Y_k) \;/\; \text{Prob}(t \mid G) \right)}{\text{norm}},$$

   where G is the old grammar and "norm" is the appropriate normalizing constant so that $\sum_{Y_1,\ldots,Y_k} \hat{\mathcal{P}}(S \rightarrow Y_1 \cdots Y_k) = 1$.
3. A new current grammar is created by simply replacing $\mathcal{P}(S \rightarrow Y_1 \cdots Y_k)$ with the re-estimated probability $\hat{\mathcal{P}}(S \rightarrow Y_1 \cdots Y_k)$.
4. Steps 2 and 3 are repeated until the parameters of the current grammar change only insignificantly.

2.4 Iterative usage of the training algorithm

If only unfolded training sequences are available, then we iteratively estimate the folding of the training sequences as well using the following method:

1. First, we design a rough initial grammar which might represent only a portion of the base pairing interactions. This is used to parse the initial unfolded RNA training sequences to obtain a set of partially folded RNA sequences.
2. Next, we estimate a SCFG using the partially folded sequences and our training algorithm to obtain a new estimated grammar. Further productions might be added to the grammar at this stage, although we have not experimented with this possibility yet.
3. Then we use the trained grammar to obtain more accurately folded training sequences and estimate a SCFG using these.
4. We repeat this process until the trained grammar gives no changes to the folding.

2.5 Dealing with introns

Introns are sometimes present in tRNA genes. This means that when we search databank files of genomic sequences, the sequence of the tRNA may be interrupted by non-tRNA coding nucleotides. The grammar needs to be extended to handle this situation.

An extremely useful advantage of SCFGs is their modularity. We see this clearly in this case: an intron grammar can be deduced separately from the grammar for plain tRNA, then these two separate grammars can be combined into a single grammar simply by uniting the two sets of independent productions and maintaining their different probability distributions.

3 Results

3.1 Data

The experiments used data from three sources:

1. From EMBL Data Library's database, we obtained 1477 aligned and folded tRNA sequences. Of these 1477 tRNA sequence descriptions, we selected randomly 500 as training examples for deriving a grammar and used the rest as test data.
2. The Ribosomal Database Project's (RDP) aligned, folded large subunit ribosomal RNA data file provided primary source of non-tRNA sequences.
3. From NCBI's NewGenBank and GenBank databases, we used 55 unaligned and unfolded tRNA sequences with introns of rather short lengths (from 4 to 25). The GenBank databases also include descriptions of other RNA besides tRNA.

3.2 Discriminations of tRNAs from non-tRNAs: Database search

As described in Section 2.2, we calculate a NLL-score for each test sequence and use it to measure how well the sequence fits the grammar. This raw NLL-score depends too much on the length of test sequence to be used directly to decide whether a sequence belongs to the family modeled by the grammar. However, this problem can be overcome by normalizing the NLL-score appropriately. Details are described in [HKMS93]. Essentially, we calculate the difference between the NLL-score of a sequence and the average NLL-score of a typical non-tRNA sequence of the same length, measured in standard deviations. This number is called the *Z-score* for the sequence. We then choose a Z-score cutoff, and sequences with Z-scores above the cutoff are classified as positive examples.

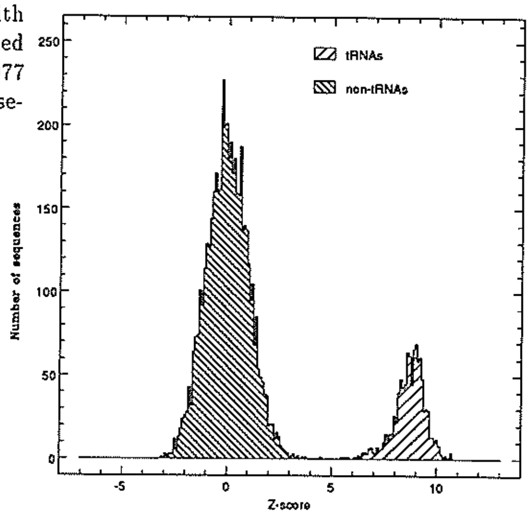
For the alignment grammar and the trained grammar, NLL-scores and Z-scores were computed for 977 test tRNA sequences and 4885 non-tRNA sequences of length 71 to 90. For each tRNA sequence, there are five non-tRNA sequences of the same length. The grammar distinguishes perfectly between tRNAs and non-tRNAs: the lowest Z-score of tRNAs is 4.984 and the highest Z-score of non-tRNAs is 4.589. Thus, choosing a Z-score cutoff between them, we can discriminate tRNA sequences from non-tRNA sequences perfectly.

Surprisingly, the trained grammar was able to discriminate more reliably than the alignment grammar in that the trained grammar created a greater gap between Z-scores of tRNAs and non-tRNAs. Z-scores made by the trained grammar are shown in Figures 3. The lowest Z-score of tRNAs is 5.464 and the highest Z-score of non-tRNAs is 4.517. Thus the trained grammar distinguishes perfectly between tRNAs and non-tRNAs and more reliably than the alignment grammar. This is unexpected because the trained grammar is obtained using only 500 training sequences, so 977 test tRNA sequences are completely new for the trained grammar, while the alignment grammar is obtained using all 1477 aligned tRNA sequences.

3.3 Multiple sequence alignments

From a grammar it is possible to obtain a multiple alignment of all the sequences. The grammar can produce the most likely parse tree for the sequences to be aligned. This gives an alignment of all the nucleotides that align to the match nonterminals on the main line in the grammar. Between the match nonterminals there might be insertions of varying lengths, but by inserting enough spaces in all the sequences to accommodate the longest insertion, an alignment is obtained. Figure 4 shows the alignment produced by the trained grammar for 15 tRNA sequences in EMBL Data Library. The boundaries of the helices and loops are the same as those in the original alignment in EMBL Data Library. The major difference between the two alignments is the extra arm, which is itself highly variable in terms of its length and sequence. Both alignment

Figure 3: The number of sequences with a certain Z-score scored by the trained grammar. Shown is the test set of 977 tRNA sequences and 4885 non-tRNA sequences.




```
Base pairings                               Anticodon                                              Base pairings
(((((((  ((((                       )))) (((((  ===  )))))                                          (((((        ))))))))))))
GCCCGGGUGGUGAAAU.C.GG.Ua.GACACGCAGGACUUAAAAUCCUGU...........Ggcauaaaagcc..A-.............-UGUCGGUUCAAGUCCGACCCCGGGCA
GGAUGGAUGUCUGAGC.-.GGuUg.AAAGAGUCGGUCUUGAAAACCGAA...........GuauuucuaggaauA-.............CCGGGGGUUCGAAUCCCUCUCCAUCCG
GCGGAUGUGGCGGAAU.U.GG.Ca.GACGCGCUAGAAUCAGGCUCUAGU...........Gucuuuacagac..G-.............-UGGGGGUUCAAGUCCCUUCAUCCGCA
GCCCGGCUAGCUCAGU.C.GG.U..AGAGCAUGAGACUCUUAAUCUCAG...........G.............G-.............UCGUGGGUUCGAGCCCCACGUUGGGCG
GGGUGUAUAGCUCAGU.U.GG.U..AGAGCAUUGGGCUUUUAACCUAAU...........G.............G-.............UCGCAGGUUCAAGUCCUGCUAUACCCA
GACAUCGUAGCAAAGU.-.GG.UcuAAUGCGUCUGACUAGAAAUCAGAUcccuucggg..G.............G-.............-CGCAGGUUCGAACCCUGCCGAUGUCG
GCCGAGGUGGUGGAAU.U.GG.Ua.GACACGCUACCUUGAGGUGGUAGUgcccaauag..G.............GC.............UUACGGGUUCAAGUCCCGUCCUCGGUA
GCGAAGGUGGCGGAAU.U.GG.Ua.GACGCGCUAGCUUCAGGUGUUAGU...........Guccuuacggac..G-.............-UGGGGGUUCAAGUCCCCCCCUCGCA
GGAGAGAUGCCGGAGC.-.GGcUg.AACGGACCGGUCUCGAAAACCGGA...........GuaggggcaacucuA-.............CCGGGGGUUCAAAUCCCCCUCUCUCCG
GGGUCGUUAGCUCAGA.C.GG.U..AGAGCAGCGGACUUUUAAUCCGUU...........G.............G-.............UCGAAGGUUCGAAUCCUUCACGACCCA
GCGGGGGUGGCUGAGCcA.GG.CcaAAAGCGGCGGACUUAAGAUCCGCUcccguaggg..G.............U-.............UCGCGAGUUCGAAUCUCGUCCCCCGCA
GCGCGGGUAGCCAAGU.-.GG.CcaAAGGCGCAGCGCUUAGGACGCUGU...........G.............GUguagaccu.....UCGCAGGUUCGAACCCUGUCCCGCGCA
GGGCCCGUAGCUUAGUcU.GG.U..AGAGCGCCUGGCUUUUAAUCAGGC...........G.............G-.............UCGAGGGUUCGAAUCCCUUCGGGCCCG
GGAAGAUUACCCAAGU.CcGGcUg.AAGGGAUCGGUCUUGAAAACCGAG...........A.............GUcggggaaaccgag-CGGGGGUUCGAAUCCCUCAUCUUCCG
GCCUUUGUAGCGGAAU.-.GG.U..AACGCGGCAGACUCAAAAUCUGCUuugguaacccaG.............G-.............-UGGUAGUUCGACUCUCCCCAAAGGCA
```

Figure 4: Alignment produced by the trained grammar for 15 tRNA sequences from EMBL Data Library. The parentheses above the alignments indicate which columns (positions) form base pairs and "===" indicates the anticodon domain. Capital letters correspond to nucleotides aligned to the main line of the grammar, "-" to deletions by skip productions in the grammar, and lower-case letters to nucleotides treated as insertions by the grammar. The "." is used as a fill character to accommodate insertions.

give the same base-pairing. Once a grammar has been constructed, a similar multiple alignment can be produced for the entire set of 1477 tRNA sequences (or any subset).

3.4 Predictions of secondary structures

As discussed in the last section, the trained grammar produces the same alignment as the original alignment for the base pairing parts. This implies that the most likely parse trees produced by the trained grammar give the correct secondary structure for all 1477 tRNA sequences, 500 training plus 977 unseen test sequences.

3.5 Introns

Our experiments with introns are currently only very preliminary. We have not yet obtained a large enough data set to do truly meaningful experiments. In one experiment, we used 55 sequences of introns and a simple regular grammar with five nonterminals for training a (sub)grammar to model introns. The training process reestimated the distributions of the four nucleotides at the first and last two consective positions of the introns. This grammar trained for introns and the grammar previously trained for intron-free tRNAs were then combined into a single grammar modeling tRNAs with introns.

We tested the grammar on the same 55 tRNA sequences with introns. Of these, the grammar correctly identified the positions of introns and introns themselves for 44 sequences. Introns with the same lengths as the correct ones but incorrect positions for two or three bases (shifted) were found for 7 sequences. The grammar completely failed to identify introns for 4 sequences. Thus, this approach shows some promise in identifying introns and finding the correct secondary structure for tRNA sequences with introns.

References

[AU72] Alfred V. Aho and Jeffrey D. Ullman. *The Theory of Parsing, Translation and Compiling, Vol. I: Parsing.* Prentice Hall, Englewood Cliffs, N.J., 1972.

[Bak79] J. K. Baker. Trainable grammars for speech recognition. *Speech Communication Papers for the 97th Meeting of the Acoustical Society of America*, pages 547–550, 1979.

[BCHM93] P. Baldi, Y. Chauvin, T. Hunkapiller, and M. A. McClure. Hidden Markov models in molecular biology: new algorithms and applications. In Hanson, Cowan, and Giles, editors, *Advances in Neural Information Processing Systems 5*, pages 747–754, San Mateo, CA, 1993. Morgan Kauffmann Publishers.

[HKMS93] D. Haussler, A. Krogh, I. S. Mian, and K. Sjölander. Protein modeling using hidden Markov models: Analysis of globins. In *Proceedings of the Hawaii International Conference on System Sciences*, Los Alamitos, CA, 1993. IEEE Computer Society Press.

[Rab89] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc IEEE*, 77(2):257–286, 1989.

[Sea92] David B. Searls. The linguistics of DNA. *American Scientist*, 80:579–591, November–December 1992.