Locally Aligned Terminal Distinguishable Languages and Identification of Proteins

K.V. Seshu Kumar, G. Nagaraja and P.V. Balaji

Artificial Intelligence : Theory and Practice
Proceedings of the International Conference
KBCS-2002
Mumbai, Dec. 19 - 21, 2002
pp 311 - 321

M. Sasikumar et.al. (Eds.)
Vikas Publishing House Pvt. Ltd., New Delhi

Locally Aligned Terminal Distinguishable Languages and Identification of Proteins

K. V. Seshu Kumar
HP ISO Pvt. Ltd.
29C, Cunningham Road
Bangalore, India
Email: *seshuk@india.hp.com*

G. Nagaraja
Dept. of CSE
IIT Bombay, Powai
Mumbai 400076, India
Email: *gn@cse.iitb.ac.in*

P. V. Balaji
Biotechnology Centre,
IIT Bombay, Powai
Mumbai 400076, India
Email: *balaji@btc.iitb.ac.in*

Abstract

Grammatical Inference is the problem of learning a formal grammar that characterizes a set of strings over an alphabet. In particular, the problem of inferring regular grammars from positive data is considered in this paper. In the framework of identification in the limit, subfamilies of the family of regular languages have been defined in the literature. These include Strictly Locally Testable Languages(SLT) and Terminal Distinguishable Regular Languages (TDRL). TDRL's have been shown to possess many desirable properties for inference. SLT languages have been applied to protein sequence analysis. When dealing with biological sequences, local alignment issues become important. In this paper, a framework has been proposed for introducing local alignment considerations into grammar inference models and a learning algorithm has been given for a family defined as *Locally Aligned TDRL*. Results of application of the algorithm to protein sequence data and comparison with SLT are presented. Experimentally, it is inferred that the generalization of LA-TDRL is not affected by noise that is present in the data and the amount of generalization can be controlled by an input parameter.

1 Introduction

Grammatical inference is concerned with the problem of inferring a grammar of the Chomsky hierarchy associated with a given set of strings belonging to a formal language over a given alphabet of symbols. E.M. Gold considered this problem in its generality and defined the learning model called "Identification in the limit". Due to space constraints, the details are skipped here and the reader can refer to [Gold, 1967]. Identification can be from (i) both positive and negative data or (ii) from positive data only. In many applications, negative data (strings not belonging to the language to be identified) may not be available. It has been shown by Gold [1978], [1967] that the family of Regular Languages is not identifiable in the limit from positive data only. Angluin [1980] has proved that subfamilies of the family of regular languages are learnable from positive data.

In the framework of identification in the limit, subfamilies of the family of regular languages have been defined in the literature. These include Strictly Locally Testable Languages and Terminal Distinguishable Regular languages (TDRL). TDRL's have been shown to possess many desirable properties for inference. *SLT* languages have been applied to protein sequence analysis. When dealing with protein sequences, local alignment issues become important. In this paper, a framework has been proposed for introducing local alignment considerations into grammar inference models and a learning algorithm has been given for a family defined as *Locally Aligned TDRL*(LA-TDRL). Results of application of the algorithm to protein sequence data and comparison with *SLT* are presented. The experimentally observed advantage of LA-TDRL is that generalization of the sample strings will not be affected by noise that is present in the data and the amount of generalization can be controlled by an input parameter.

Section 2 contains notation used throughout this paper. Section 3 describes the proposed method along with an algorithm for the inference of a regular grammar from a finite sample set. Section 4 reports the experiments we performed to identify alpha chain region of amino acid sequences of hemoglobin. The last section ends with conclusions and problems identified.

2 Notations and Definitions

2.1 Notations

We follow the notation in [Hopcroft and Ullman, 1985], [Fernau, 2000]. A Grammar G is denoted by a quadruple (N, Σ, P, S), where N is the set of non-terminal symbols, Σ is the set of terminal symbols disjoint from N, P is the rule set and $S \in N$ is the start symbol. A regular language L may be referred by $L(G)$ where G is a regular grammar. Σ^* is the set of words over Σ. $\Sigma^k(\Sigma^{<k})$ collects the words whose length is equal(less than) k. S^+ represents positive sample set and every string x in S^+ belongs to the language $L(G)$. S^- represents negative sample set and no string x in S^- belongs to the language $L(G)$. $|S^+|$ represents the size of the positive sample set and similarly $|S^-|$ represents the size of negative sample set. Let the sample set be $S^+ = \{w_1, w_2, ..w_M\}$.Let each w_i be represented as $a_{i1}, a_{i2}, \ldots a_{in_i}$, where $a_{ij} \in \Sigma,\ \ 1 \leq i \leq M,\ \ 1 \leq j \leq n_i$. We consider negative data for validation purpose only. Inference algorithm uses only positive data.

2.2 Definitions

Given a set of strings S^+, let N_{ij} denote the j-th node in the derivation tree of i-th string. The skeletal Grammar associated with the sample set S^+ is defined as follows.
$G_S = (N_S \cup \{S\}, \Sigma, P, S)$ where $N_S = \{N_{ij} \mid 1 \leq i \leq M, 1 \leq j \leq n_i\}$ and $P_S = \{S \rightarrow N_{i1} \mid 1 \leq i \leq M\}$ $\cup\{N_{ij} \rightarrow a_{ij}N_{ij+1} \mid\ \ 1 \leq i \leq M, 1 \leq j \leq n_i\}$ $\cup\{N_{in_i} \rightarrow a_{in_i}\ 1 \leq i \leq M\}$

Definition 1: Frontier string of N_{ij} , denoted by $FS(N_{ij})$, is defined as $a_{ij} \ldots a_{in_i}$.

Definition 2: Head string of N_{ij} is denoted by $HS(N_{ij})$, defined as $a_{i1} \ldots a_{ij-1}$.

It follows that $HS(N_{ij})FS(N_{ij}) = w_i$.

Definition 3: Given $x \in \Sigma^*$, let

$$\begin{array}{lll} Ter_k(x) & = & (\pi_k(x), \mu_k(x), \sigma_k(x)) \text{ where} \\ \mu_k(x) & = & \{a \in \Sigma^{k+1} | uav = x \ \ and\ \ u, v \in \Sigma^*\} \end{array}$$

and $\pi_k(x)[\sigma_k(x)]$ is the prefix[suffix] of length k of x, if $x \notin \Sigma^{<k}$, and $\pi_k(x) = \sigma_k(x) = x$ if $x \in \Sigma^{<k}$.

Definition 4: Truncated frontier string of N_{ij}, denoted by $FSL(N_{ij})$, is equal to the first l or $< l$ characters of $FS(N_{ij})$ depending upon whether $|FS(N_{ij})| \geq l$ or $|FS(N_{ij})| < l$.

Definition 5: Truncated head string of N_{ij}, denoted by $HSL(N_{ij})$, is equal to the last l or $< l$ characters of $HS(N_{ij})$ depending upon $|HS(N_{ij})|$. If $|HS(N_{ij})| \geq l$ then last l characters of $|HS(N_{ij})|$ else $< l$ characters of $|HS(N_{ij}|)$.

We now describe local and global alignment of two sequences s_1 and s_2. Let j be some fixed position with j^{th} symbol of s_i denoted by s_{ij}. If $|s_1| = |s_2|$ and $s_{1j} = s_{2j}$, then we say that the two strings are aligned globally at position j. But if the two strings are of different lengths then some gaps will be introduced in s_1 and s_2 such that they are aligned globally at maximum number of places and the gap penalty [Lerman, 2000] is least. For example, if $s_1 = ababa$ and $s_2 = abbba$, then $|s_1 = s_2$ and these are globally aligned at 4 out of 5 positions.

For local alignment considerations, we consider a window of length k and match substrings of s_1 and s_2 while sliding the window. Let $Window(s_1, s_2, i, j, k)$ represent the window containing substring $s_{1i} \ldots s_{1i+k}$ and $s_{2j} \ldots s_{2j+k}$ where i and j represent the starting positions of the window in s_1 and s_2 respectively and k is the length of the window such that $s_{1l} = s_{2m}$ where $i < l < i + k, j < m < j + k$. The maximum possible value of window size k locally is considered and disjoint windows are used over the sequences s_1 and s_2. Thus, if the strings are $s_1 = ababa$ and $s_2 = abbba$, then there are two windows of size 2 each where s_1 and s_2 are same (initial substring ab and final substring ba).

3 Local Alignment and Terminal Distinguishable Languages

3.1 Motivation

The TDRL inference algorithm [Radhakrishnan and Nagaraja, 1987][Radhakrishnan, 1987] infers a good underlying grammar compared to many other inference methods for regular grammars as shown in [Gronfors and Juhola, 1992]. The inferred grammar performance, which will be explained later, will depend upon the characteristic sample set. The characteristic sample set is not known in many problem domains such as the protein classification problem. However TDRL fails to generalize when either the size of the sample set is very small or the prefix similarity $PS(s_1, s_2)$ in sample strings is very low which is obvious from the definition of $FS(N_{ij})$,$HS(N_{ij})$ and similarly for the case when sample strings have high local similarity. Let s_1, s_2 be two strings of length l and let s_2 be formed by simply changing one letter in s_1. Let the part of s_1, s_2 after the changed position be the same. Thus, we conclude s_1 and s_2 are quite the same but the TDRL inference algorithm is unable to exploit it. Generally real data contains some amount of noise. So this limits the TDRL performance.

There are two cases in which the performance of TDRL algorithm will be affected[Seshu Kumar, 2001]. Let s_1, s_2 be two strings.

- Let us say that s_2 is formed by replacing the last symbol in s_1 with a new symbol that is not present in s_1. This can be interpreted as the result of some noise due to which the last character of one string changed to a new character. The following example illustrates this case and the performance of TDRL in exploiting the common structural information that is present in the strings.

  Example 1: Let $s_1 = ababa, s_2 = ababc$. $Ter(N_{1k}) \neq Ter(N_{2k})$ and $FS(N_{1k}) \neq FS(N_{2k})$ where $1 \leq k \leq 5$. Because of this TDRL inference algorithm will not be able to exploit the structural similarity of s_1, s_2 as the non-terminal merging is not possible. So it will generate different production rules for s_1, s_2. Similarly if first letter is replaced by some other letter then also the TDRL algorithm will fail to merge the non-terminals.

- This case illustrates when one character near the beginning position and one character near the ending position are not the same.

  Example 2: Let $s_1 = abababababab, s_2 = aaababababaa$ clearly s_1 and s_2 have the same terminals except at positions $2, 12$. In this case also the TDRL inference algorithm will not recognize the similarity present in both the strings to merge the non terminals.

From above observations we can conclude that if two characters (one near to the beginning of the string and the other near to the end of the string) are replaced by the some other characters then the TDRL inference algorithm will fail to exploit the structural information that is present in the given sample strings. This limits the application of TDRL inference algorithm to many of the real data with some noise and in some cases the structure may be not be same for the entire length of the strings. Now we provide a method which overcomes this problem[Seshu Kumar, 2001]. It is presented in the following sections.

In order to overcome the above mentioned problem, we simply replace the definitions of $FS(N_{ij}), HS(N_{ij})$ in TDRL algorithm with $FSL(N_{ij}), HSL(N_{ij})$.

3.2 Informal description

We place a window of length l and we compare the FS, HS of each node within the window and if they satisfy the TDRL heuristics for this window length then the nodes will be merged. The window is moved to the next position in the string. From this we can also call LA-TDRL as $Sliding\ Window\ TDRL$. The advantage of sliding window is that it allows some amount of noise which can be tolerated in the inference and when the strings are having low positional dependency $PS(s_1, s_2)$. This is explained in the next section.

The LA-TDRL inference algorithm is outlined as follows. Call N_{ij} to be equivalent to N_{kl} if (1) $FSL(N_{ij}) = FSL(N_{kl})$ or (2) $HSL(N_{ij}) = HSL(N_{kl})$ and $Ter_k(FSL(N_{ij})) = Ter_k(FSL(N_{kl}))$. Now merge N_{ij} and N_{kl} if they are equivalent. The algorithm in detail is given below.

N_S	HSL	FSL	$Ter(FSL)$	$Ter_1(FSL)$
N_{11}	λ	aba	$\{a,b\}$	$(a,\{ab,ba\},a)$
N_{12}	a	bab	$\{a,b\}$	$(b,\{ba,ab\},b)$
N_{13}	ab	ab	$\{a,b\}$	$(a,\{ab,ba\},b)$
N_{14}	aba	b	$\{b\}$	$(b,\{\phi\},b)$
N_{21}	λ	aba	$\{a,b\}$	$(a,\{ab,ba\},a)$
N_{22}	a	bab	$\{a,b\}$	$(b,\{ba,ab\},b)$
N_{23}	ab	aba	$\{a,b\}$	$(a,\{ab,ba\},a)$
N_{24}	aba	bab	$\{a,b\}$	$(b,\{ba,ab\},b)$
N_{25}	bab	ab	$\{a,b\}$	$(a,\{ab\},b)$
N_{26}	aba	b	$\{b\}$	$(b,\{\theta\},b)$
N_{31}	λ	bab	$\{a,b\}$	$(b,\{ba,ab\},b)$
N_{32}	b	aba	$\{a,b\}$	$(a,\{ab,ba\},a)$
N_{33}	ba	ba	$\{a,b\}$	$(b,\{ba\},a)$
N_{34}	bab	a	$\{a\}$	$(a,\{\phi\},a)$
N_{41}	λ	bab	$\{a,b\}$	$(b,\{ba,ab\},b)$
N_{42}	b	aba	$\{a,b\}$	$(a,\{ab,ba\},a)$
N_{43}	ba	bab	$\{a,b\}$	$(b,\{ba,ab\},b)$
N_{44}	bab	aba	$\{a,b\}$	$(a,\{ab,ba\},a)$
N_{45}	aba	ba	$\{a,b\}$	$(b,\{ba\},a)$
N_{46}	bab	a	$\{a\}$	$(a,\{\phi\},a)$

Table 1: Table for example of section 3.4

3.3 Inference Algorithm

Here we present an algorithm similar to the inference algorithm presented in [Radhakrishnan and Nagaraja, 1987], [Fernau, 2000].
Algorithm 1:
Input : Sample set S^+, k, l where $l \neq 0, k \geq 0$
Output : Inferred Grammar G.
Process:

1. Construct skeletal grammar G_S
2. Compute for all possible values of $i, j,\ \ HSL(N_{ij}), FSL(N_{ij}), Ter_k(FSL(N_{ij}))$
3. Partition N_S into $N_S = N_S^1 \cup ... \cup N_S^r$ such that $\forall 1 \leq p \leq r, \forall N', N'' \in N_S^p : HSL(N') = HSL(N'') \wedge Ter_k(FSL(N')) = Ter_k(FSL(N''))$.
4. Compute FSL^p=$\{FSL(N') \mid N' \in N_S^p\}$
5. Identify N_S^p and N_S^q iff $FSL^p \cap FSL^q \neq \phi$. In this way we get a coarser union $N_S = N_1 \cup .... \cup N_s$ with $s \leq r$.
6. Output $G = \{N, \Sigma, P, S\}$, where $N = \{N_1 \cup .... \cup N_s\} \cup S$ and P contains $N_u \to aN_v$ iff there are nonterminals $N^1 \in N_u$ *and* $N^2 \in N_v$ and a rule $N^1 \to aN^2$ *in* P_S. Moreover, P contains rules $S \to N_u$ iff there is a non-terminal $N' \in N_u$ such that $FSL(N') \in S^+$.

3.4 An Example

We consider the example given in [Fernau, 2000]. Let the sample set be $S^+ = \{w_1 = abab, w_2 = ababab, w_3 = baba, w_4 = bababa\} \subset \{(ab)^n, (ba)^n \mid n > 1\} = L$. Table 1 shows the corresponding $Ter(FSL), FSL, HSL$ corresponding to different samples given in S^+ for $k = 1$ and $l = 3$;

Consider k=1. In step 3 we get as a partition :
$N_S = \{N_{11}, N_{21}\} \cup \{N_{12}, N_{22}\} \cup \{N_{31}, N_{41}\} \cup \{N_{32}, N_{42}\} \cup \{N_{14}, N_{26}\} \cup N_S'$ where N_S' collects the remaining non-terminals of G_S as singleton sets. After step 5 the following groups are obtained. $N_1 = \{N_{11}, N_{21}, N_{23}, N_{42}, N_{32}, N_{44}\}$, $N_3 = \{N_{13}, N_{25}\}$, $N_4 = \{N_{14}, N_{26}\}$, $N_2 = \{N_{12}, N_{22}, N_{41}, N_{31}, N_{43}, N_{24}\}$,

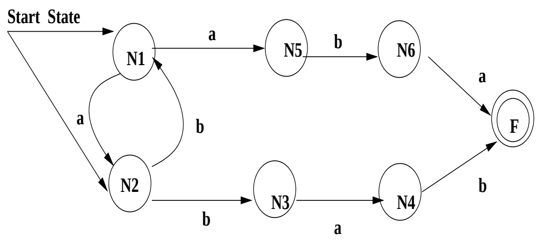

Figure 1: Inferred directed graph of finite automaton for l = 3

$N_5 = \{N_{33}, N_{45}\}, N_6 = \{N_{34}, N_{46}\}$. Step 6 yields the grammar with rules P= $\{S \rightarrow N_1, N_1 \rightarrow aN_2, N_2 \rightarrow bN_3, N_1 \rightarrow aN_5N_4 \rightarrow b, S \rightarrow N_2, N_2 \rightarrow bN_1, N_3 \rightarrow aN_4, N_5 \rightarrow bN_6, N_6 \rightarrow b\}$.

The obtained regular grammar is represented by the directed graph for a finite automaton shown in Figure 1

A brief worst case complexity analysis of the above algorithm is presented, as it may be difficult to give exact worst case complexity. Let the cardinality of the sample set S^+ be t. Let l be the sum of all the strings in S^+. Step 3 of the algorithm is the costliest step in partitioning the node sets. At most $O(l^2)$ pairwise comparisons are required. Let us assume that every head string is of length l. Then the time requirement for step 3 will be $O(l^3)$.

3.5 LA-TDRL and other subfamilies

Let $A_i, B_i \in \Sigma^*, 1 \leq i \leq 4$ and $V, U, W, Z \in \Sigma^l, X, Y \in \Sigma^{<l}$.

Definition 6: A language L is said to be locally aligned terminal distinguishable language iff whenever

1. $A_1VWB_1, A_2UWB_2 \in L$ then $A_3VZB_3 \in L$ iff $A_4UZB_4 \in L$.
2. $XWB_1, YWB_2 \in L$ then $XZB_3 \in L$ iff $YZB_4 \in L$.
3. $A_1VX, A_2UX \in L$ then $A_3VY \in L$ iff $A_4UY \in L$.

k-reversible regular languages(k-RL) have been defined in [Angluin, 1982].
Theorem 1: $\forall k \geq 0$, k-RL $\subset$ LA-TDRL.
Proof: A Language $L \subset \Sigma^*$ is k-RL iff, whenever $vw, uw \in L$, then for all $z \in \Sigma^*$ with $\pi_k(w) = \pi_k(z)$ we have $uz \in L$ iff $vz \in L$. Now we have to show that L is included in the LA-TDRL subclass of regular languages. Let $B_1, B_2, B_3, B_4 = \epsilon$, $l = min(|v|, |u|, |w|)V$, U=last l characters of v, u respectively, W, Z=first l characters of w, z. We have to show that whenever vw, uw are there in LA-TDRL then $vz \in$ LA-TDRL iff $uz \in$ LA-TDRL, which is true from the definition of LA-TDRL.

Example: The language $ca^*b \cup c(aa)^*b \in$LA-TDRL but does not belong to k-RL.

It is known from [Garcia *et al.*, 1990] that $\forall k \geq 0$, (k+1)-SLT $\subset$ k-RL.

From this and Theorem 1, it follows that: $\forall k \geq 0$, (k+1)-SLT $\subset$ k-RL $\subset$ LA-TDRL for some l

Theorem 2:(Hierarchy theorem) $\forall l > 0$, for some k, LA-TDRL(l,k) $\subset$ LA-TDRL(l+1,k).
Proof: Consider the characterization of LA-TDRL as in Definition 6. In case 1 of LA-TDRL(l,k) we have $|V|, |U|, |W|, |Z| = l$ and similarly in case of LA-TDRL(l+1,k) we have $|V_1|, |U_1|, |W_1|, |Z_1| = l + 1$ but $V_1 = aV$, $U_1 = bU$, $W_1 = Wc, Z_1 = Zd$ where $a, b, c, d \in \Sigma$ and also $Ter_k(W_1) = Ter_k(Z_1)$ implies $Ter_k(W) = Ter_k(Z)$. Similarly for the case 2, 3 one can proceed as above, so the result follows.

The above result will be used in later sections in order to perform the identification experiments for alpha chain region of hemoglobin using LA-TDRL.

4 Analysis of Protein Sequences

Proteins are sequences over the 20 letter alphabet of amino acids. A string of letters constituting a protein is a substring of a longer string of letters over this alphabet. This is the primary structure of a protein. The secondary structure of proteins are described as α - helix, β - sheet or random coil. Since each amino acid consists of a 3 letter code over the 4 letter alphabet A,C,G,T of nucleic acids, protein sequences can also be considered as sequences over this 4 letter alphabet which is the alphabet of DNA or RNA language. The role of splicing in recombinant DNA sequencing is well known. Splicing refers to the process by which coding regions (exons) of a nuclear RNA sequence are separated from non-coding regions (introns) to create a new sequence called messenger RNA (mRNA). This process happens by the interaction of restriction enzymes and nuclear RNA sequences resulting in the recombination of DNA molecules.

The splicing phenomenon has been modeled at the syntactic level by the notion of a splicing system. It has been shown that persistent splicing systems are related to k-SLT languages. The notation of splicing system was introduced by Head [1998a] as a mathematical model for restriction enzyme digestion and subsequent relegation in the recombination of DNA molecules. He proved an interesting relationship between k-SLT languages and persistent splicing systems [Yokomori *et al.*, 1994]. Yokomori et.al [1994] performed experiments using the above relationship for identification of alpha chain region of amino acid sequence of hemoglobin by assuming that it can be produced by persistent splicing systems. We use the relationship between LA-TDRL and k-SLT. The procedure will be explained in the following sub sections.

4.1 Molecular Considerations

Each protein can be represented by a string from an alphabet containing 20 symbols. Each symbol represents an amino acid. The amino acids $A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y$ are terminal symbols in the formal language representation. Each alpha chain region of amino acid sequence of hemoglobin generally contains 141 to 143 amino acids.

4.2 Protein α chain Identification

Working Assumption: some types of amino acid sequences are generated by persistent splicing systems. Hence they are characterized by locally testable languages. But using the formal language relationship between LA-TDRL and k-SLT which was proved in theorem 1, we performed the experiments for identification of alpha chain region of hemoglobin.

1. **Goal:** is to find a Grammar G that identifies the alpha chain region of amino acid sequences in hemoglobin.
2. **Process:**The positive data of alpha chain region of amino acid sequences of hemoglobin is drawn from PRF database.
   (a) Select an encoding scheme.
   (b) Apply the LA-TDRL inferencing algorithm.
   (c) Using some rating functions calculate the performance. The rating functions are taken from [9].

In our experiments, we have used $|S^+ + Test| = 300$, $|S^-| = 5000$ and $|S^+| = \{25, 38, 57, 72, 99\}$. Each positive sample string length is between 141 and 144, negative samples between 134 and 144. Negative samples are for validation purpose only.

4.3 Coding Scheme Used

There are two coding schemes generally employed. One is Dayhoff's coding and the other is binary coding. Table Dayhoff's coding is from 20 symbol to 6 symbols. Table 2 shows Dayhoff's encoding.

Amino Acids	Properties	New Symbol
C	sulfur polymerizations	a
S, T, P, A, G	small	b
N, D, E, Q	acid & amide	c
H, R, K M, I, L, V	basic hydrophobic	d e
F, Y, W	aromaticity	f

Table 2: Dayhoff's coding

l	l=2	l=4	l=8	l=15	l=23
R_{pos}	74	37	27	25	24.5
R_{neg}	96	99	99	99	99.7

Table 3: Experimental Results for Dayhoff's Coding, Rating function $R_1, k = 1, |S^+| = 25, |S^-| = 500$

4.4 Experimental Procedure

The experiment were conducted as follows.

1. Select some samples and shuffle the order of samples.
2. Then using this sample set S^+, apply the learning algorithm different values of k, l.
3. Calculate the performance which is evaluated using remaining samples from negative data as well as test data $Test$, where $Test$ is disjoint from sample set S^+. We used two rating functions. Let G be the inferred grammar.
   (a) First Rating function R_1 is as follows.

$$R_1 = \langle\, R_{pos}, R_{neg} \,\rangle \text{ where}$$

$$R_{pos} = \frac{|Test \cap L(G)|}{|Test|}$$

$$R_{neg} = \frac{|S^- \cap \neg L(G)|}{|S^-|}$$

   (b) Second Rating function R_2 which was introduced in [Brazma *et al.*, 1998].

$$\begin{aligned} R_2(L, Test, S^-) &= \tfrac{|Test \cap L(G)|}{|S^- \cap L(G)|} \quad \text{if } |Test \cap L(G)| \geq t \text{ and } |S^- \cap L(G)| > 0 \\ &= \infty, \quad \text{if } |Test \cap L(G)| \geq t \text{ and } |S^- \cap L(G)| = 0 \\ &= 0, \quad \text{otherwise.} \end{aligned}$$

4.5 LA-TDRL Inference - Experimental Results

We obtained the following results for LA-TDRL inference method using Dayhoff's coding. The input parameters are $l = 2$ and for different values of k the results are tabulated in the table 3 using the rating function R_1. The performance significantly declines with increasing l while it improves on negative data. Figure 2 shows the graph plotted for different values of S^+ vs success rate. Table 4 shows the best scores obtained. $^!$ indicates the over all best success rate for positive data and for negative data obtained. $^+$ indicates the over all best success rate for positive data obtained. * indicates the over all best success rate for negative samples obtained. As the number of samples increases , the generalization ability improves.

4.6 Comparison of k-SLT and LA-TDRL

We have performed experiments with k-SLT inferencing method for the same data that was used for LA-TDRL algorithm. Table 5 shows the best scores obtained: $^+$ indicates the over all best success rate for positive data obtained, * indicates the over all best success rate for negative samples obtained.

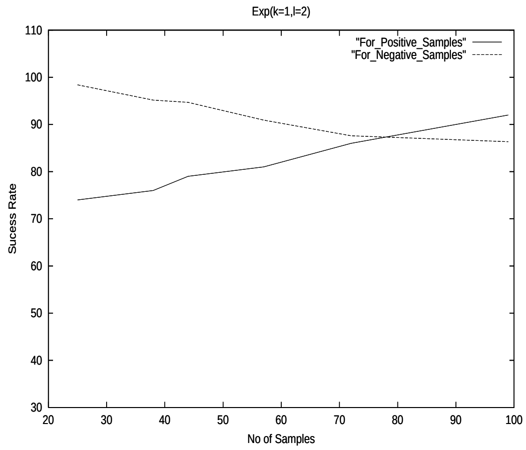

Figure 2: Experimental Results for Dayhoff's coding.

No of Samples	R_{pos}	R_{neg}
80 $^{!}$	90	87
99	92 $^{+}$	86.34
25	74	98.5 *

Table 4: Overall best performance for Dayhoff's Coding, Rating function $R_1 k = 2$.

No of Samples	R_{pos}	R_{neg}
75	98.67 $^{+}$	68.28
25	90	81.8 *

Table 5: Overall best performance for Dayhoff's Coding, Rating function $R_1,\ k = 2$.

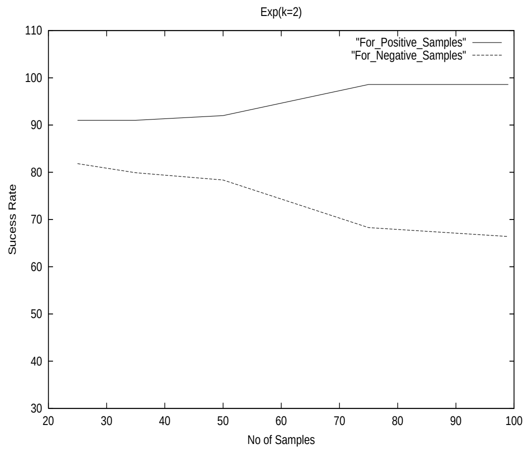

Figure 3: Experimental Results for k-SLT for Dayhoff's coding

$\lvert S^+ \rvert$	25	38	57	72	99
LA-TDRL	3.066	1.12	0.887	0.435	0.4172
k-SLT	0.31	0.295	0.256	0.187	0.177

Table 6: Comparison between $k{-}SLT(k=2), LA{-}TDRL(k=1, l=2)$ for various values of $|S^+|, |S^-| = 5000$

Figure 3 shows the graph plotted for different values of S^+ vs success rate.

We compared the two inference algorithms using second rating function R_2 and the results are tabulated in Table 6. LA-TDRL has significantly better scores than k-SLT in all cases.

5 Conclusions

In this paper, we have considered incorporating local alignment issues into regular inference algorithms with potential improvements in the performance of inference methods for TDRL. Protein identification experiments have been carried out. The advantage of TDRL over SLT has been ascertained. The following observations are made as conclusions.

In the above inferencing algorithms, the secondary structure of alpha chain region of amino acid sequence of hemoglobin is not exploited by both LA-TDRL and k-SLT inference algorithms. The primary structure of alpha chain region of amino acid sequence of hemoglobin and beta chain region of amino acid sequence of hemoglobin are similar. There are many positions that are occupied by the same amino acids. So in order to discriminate alpha chain region of hemoglobin we have to make use of its secondary structure.. In order to achieve a good success rate for different rating functions we should modify the above algorithms to incorporate the above features. Another problem that we have to address is to find the applicability of the above algorithms to sequences of other proteins.

Acknowledgments

The authors thank the referees for constructive comments. This work was carried out by K.V. Seshu Kumar in the Department of Computer Science and Engineering, IIT Bombay, while he was a postgraduate student.

References

[Angluin, 1980] D. Angluin, *Inductive inference of formal languages from positive data.* Information and Control 45, pages 117–135, 1980.

[Angluin, 1982] D. Angluin, *Inference of Reversible Languages*, Journal of ACM Vol. 29. No. 3, July 1982, pp. 741-765

[Brazma *et al.*, 1998] A. Brazma, I. Jonassen, J. Vilo and E. Ukkonen. *Pattern Discovery in Biosequences*, Proc. ICGI 98., LNAI Vol 1433, Springer, 1998, pp 257-270.

[Fernau, 2000] H. Fernau, *k-gram Extensions of Terminal Distinguishable Languages.* International Conference on Pattern Recognition ICPR 2000, vol. 2, pages 125-128. IEEE Press, 2000. http://citeseer.nj.nec.com/318865.html

[Garcia *et al.*, 1990] P.Garcia, E.Vidal, and J.Oncina, *Learning locally testable languages in the strict sense* In the First International Workshop on Algorithmic Learning Theory ALT'90, pages 325-328, 1990

[Gold, 1967] E. M. Gold. *Language Identification in the limit.*, Information and Control, 10, 447-474, 1967.

[Gold, 1978] E.M.Gold, *Complexity of automation Identification from given data*, Information and control, Vol. 37, pp. 302-320, 1978.

[Gronfors and Juhola, 1992] T. Gronfors and M. Juhola, *Experiments and comparison of inference methods of regular grammars* IEEE Transactions on systems, man, and cybernetics, Vol. 22, No. 4, July/August 1992. pp 821-826

[Head, 1998a] T. Head, *Splicing Languages Generated with One Sided Context* , Theory and Experiments, pages 269-282. Springer-Verlag, Singapore, 1998. http://citeseer.nj.nec.com/head97splicing.html

[Head, 1998b] T. Head, *Splicing Representations of Strictly Locally Testable Languages*, Discrete Applied Mathematics, 87:139-147, 1998, http://citeseer.nj.nec.com/head98splicing.html

[Hopcroft and Ullman, 1985] John E. Hopcroft, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation*, Narosa Publishing House, New Delhi, 1988.

[Lerman, 2000] Lerman I.C et al., *Classification of aligned biological sequences*, http://www.irisa.fr/ lerman.

[Radhakrishnan, 1987] V. Radhakrishnan, *Grammatical inference from positive data: an effective integrated approach*, PhD thesis, Department of Computer Science and Engineering, Indian Institute of Technology Bombay(India), 1987.

[Radhakrishnan and Nagaraja, 1987] V.Radhakrishnan and G.Nagaraja, *Inference of Regular Grammars via Skeletons.* IEEE Transactions on Systems, Man and Cybernetics, Vol, SMC-17, No.6, November/December 1987. Pages 982-992.

[Seshu Kumar, 2001] K.V. Seshu Kumar and G.Nagaraja, *Machine Learning: Formal Languages and DNA Splicing*, M.Tech Dissertation, Department of Computer Science and Engineering, Indian Institute of Technology Bombay, December 2001.

[Yokomori *et al.*, 1994] T. Yokomori, N. Ishida, S. Kobayashi, *Learning Local Languages and Its Application to Protein alpha-Chain Identification*, Proc. of 27th Hawaii International Conference on System Sciences, Hawaii, pp.113-122, 1994, January, http://citeseer.nj.nec.com/76189.html