Hidden Markov Models for Protein Sequence Alignment

Naila Mimouni [1]*, Gerton Lunter [2], Charlotte Deane [1]

[1]Peter Medawar Building, University of Oxford, 2 South Parks Road, Oxford OX1 3TG, [2]Department of Statistics, Oxford Centre for Gene Function, University of Oxford, 1 South Parks Road, Oxford OX1 3TG

ABSTRACT

Motivation: Protein homology detection and sequence alignment are at the basis of protein structure prediction, function prediction and evolutionary analysis. This work investigates the use of pair HMMS in pairwise protein sequence alignment. It uses a newly-written local software called HMMOC to perform the task. The resulting alignments are evaluated against the HOMSTRAD database of structural alignments.

Results: The basic sequence to sequence alignment HMM gives 41.20% using only sequence information, with a slight improvement when structural information is added. This is low considering Clustalw performs at 76.88% and EMBOSS Stretcher (Needleman-Wunch based) performs at 80.16%. The low performance is indicative of the presence of a bug in the system. The methodology behind the investigative steps for bug detection is explained, along with the results obtained at each step. Due to the complexity and time constraints of the project, a final answer has not been reached.

Contact: naila.mimouni@bnc.ox.ac.uk,

1 INTRODUCTION

Evolution has resulted in families of homologous proteins, where members of a family share similar amino acid sequences and protein secondary structures. The degree of similarity in sequence can vary from family to family, and for diverse families pairwise similarity can be very low. Structure, on the other hand, is more conserved. There are far more known protein sequences than protein structures. We can infer the structure of a protein of known sequence and unknown structure using sequence-structure homology recognition. Sequence-structure homology recognition uses a database of known folds to detect the fold that is most likely to be similar to the fold of the unknown structure.

*to whom correspondence should be addressed

For accurate structure prediction, it is imperative to have as "accurate" an alignment as possible between the target and a sequence of known structure. The work described in this report tackles the alignment problem. It uses a pair HMM to align two protein sequences, only using sequence information first and then adding structural information. Evaluating the resulting alignments is carried out against the HOMSTRAD structural alignments.

Section 1.1 is an introduction to protein structure. Section 1.2 discusses fold recognition approaches. Hidden Markov models, which we use, are described in section 1.3, with a short introduction to pair HMMs, HMMs used for pairwise alignment.

1.1 Protein Structure

Proteins are organic macro-molecules which are essential to the cell's structure and function. They are a component of cell membrane, and can perform a range of functions as enzymes, antibodies, hormones and transport molecules. An understanding of protein function is facilitated by the study of protein structure. Protein structures are far more complex than simple organic chemicals, because the size of the molecules allow for many possible 3-D arrangements. Protein structure is described in terms of a hierarchy, in this work, we will mainly deal with the first two.

1. Proteins are built from an alphabet of twenty smaller molecules, known as *amino acids*. A list of amino acids and their corresponding symbols is included in appendix 1. The primary structure is the sequence of amino acids in the polypeptide chain. Val-Leu-Ser-Glu-Gly-Glu-Trp-Gln-Leu-Val- represents the first ten amino acids of myoglobin. Amino acids contain an amino group (NH_2), a carboxyl group ($COOH$), and a hydrogen atom attached to a central α carbon. In addition, each amino acid also has a distinct side chain (or R group) attached to its α carbon.

   Amino acids form bonds with each other through a reaction of their respective carboxyl and amino groups. The resulting bond is called the *peptide bond*. A protein

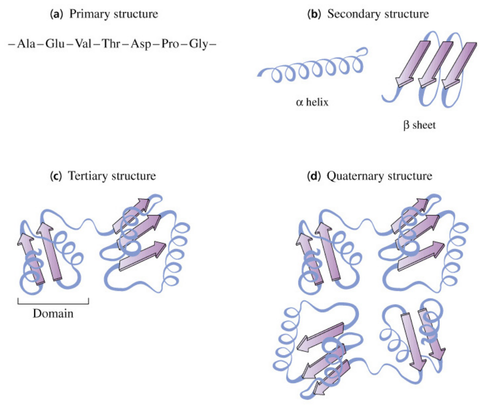

Fig. 1. The hierarchy of protein structure. Primary structure is the sequence, secondary structure includes helices and sheets. Tertiary structure is the overall folding, and quaternary structure is the grouping of several proteins.

is synthesised by the formation of a linear succession of peptide bonds between amino acids, and can thus be referred to as a *polypeptide*

2. The secondary structure is the organisation of the polypeptide into regular repetitive patterns over short segments of amino acids. The limitations imposed on the primary structure by the peptide bond and hydrogen bonds allow for only a certain number of conformations. The polypeptide chain has rotational freedom only about the bonds formed by the α carbons (ϕ angle between C_α and N, and ψ angle between C_α and the carbon of the carboxyl group). α helices and β sheets are the two types of secondary structure seen in protein. These regular structures are interspersed with regions of irregular structure that are referred to as loop or coil.
3. Tertiary structure is the overall folding of the whole polypeptide.
4. Quaternary structure is the grouping of several protein molecules into a single larger entity; the subunits may act cooperatively with each other to give the grouping special properties not possessed by the single subunit (Bourne and Weissig, 2003).

Figure 1, adapted from http://cwx.prenhall.com/horton/medialib/media_portfolio/, gives a graphical representation of the hierarchy of protein structure.

1.2 Protein Fold Recognition

Using currently-available methods, it is still not possible to make a sufficiently accurate *ab initio* protein structure prediction. Instead, databases of folds are created, and the problem becomes identifying which of the folds in the database is similar to the unknown fold of a new protein of known sequence only. This technique is based on the observation that unrelated proteins adopt the same fold, and has led to the development of programs that detect structure similarities in the absence of sequence similarities (Bourne and Weissig, 2003).

There exists a number of protein structure classifications: SCOP (Murzin *et al.*, 1995), CATH (Orengo *et al.*, 1997), and FSSP (Holm and Sander, 1994). Several methods for sequence structure similarity detection have been developed in parallel. One class fits a query sequence onto each structure in a database optimising an energy function derived from statistical or structural considerations (Threading) (Xu and Xu, 2000). Another class includes profile methods that use both sequence and structure such as 3D-PSSM (Kelley *et al.*, 2000), and FUGUE (Shi *et al.*, 2001).

Another class of methods for fold recognition is based on advanced techniques for remote homology detection based on sequence information alone (explained in the next section).

Profile HMMs

Profile HMMs[1] (Haussler *et al.*, 1993; Eddy, 1995), implemented by programs such as SAM (Karplus *et al.*, 1998), and HMMER (Eddy, 2001) use the sequence family to build a profile which includes the position-specific probabilities of variation in amino acids, as well as insertions and deletions. This indicates conserved positions (important to the family), and non-conserved positions which are variable among family members. The sequence, whose structure is not known, is then aligned to the profile, indicating the degree of homology. The membership of a sequence to a family is either given by the most probable path through the model (the Viterbi algorithm is explained in the next section), or by its posterior probability summed over all possible paths. Figure 2, adapted from Soding (2004), represents an alignment of a sequence to a profile HMM.

Profile HMMs perform well in homology detection and sequence alignment because they contain more information about the sequence family than a single sequence (Krogh *et al.*, 1994; Eddy, 1998; Karplus *et al.*, 2001). A number of structure prediction servers rely on profile-profile comparison (Rychlewski *et al.*, 2000; Ginalski *et al.*, 2003; Tang *et al.*, 2003; Tomii and Akiyama, 2004), and perform well for fold recognition in automated structure prediction contests CAFASP, LIVEBENCH, and EVA (Fischer *et al.*, 2003; Rychlewski *et al.*, 2003; Koh *et al.*, 2003). Newer methods for homology detection include HMM-HMM comparisons (Soding, 2004).

Motif Detection

Motifs are conserved sequence patterns that describe

[1] HMMs are explained in the next section

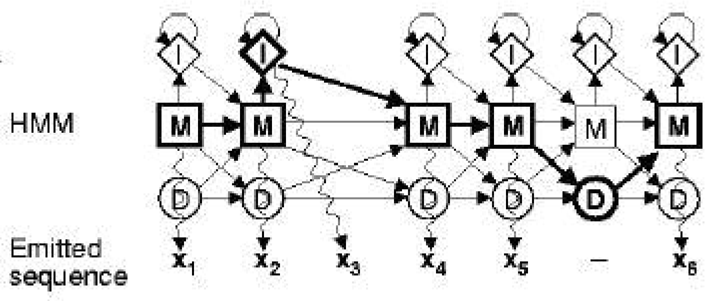

Fig. 2. The alignment of a sequence to a profile HMM. The squares indicate a match state. The diamonds an insert state, and the circles a delete state. The path through the HMM is shown in bold arrows.

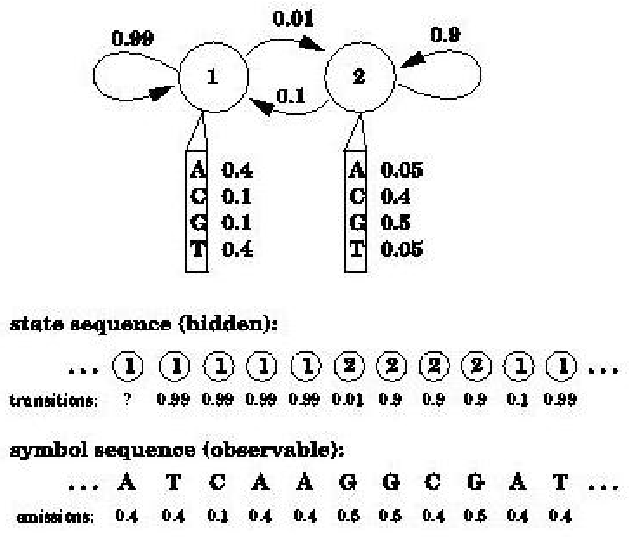
0.99
0.01
0.9
0.1
1
2
A 0.4
C 0.1
G 0.1
T 0.4
A 0.05
C 0.4
G 0.5
T 0.05
state sequence (hidden):
... 1 1 1 1 1 2 2 2 2 1 1 ...
transitions: ? 0.99 0.99 0.99 0.99 0.01 0.9 0.9 0.9 0.1 0.99
symbol sequence (observable):
... A T C A A G G C G A T ...
emissions: 0.4 0.4 0.1 0.4 0.4 0.5 0.5 0.4 0.5 0.4 0.4

Fig. 3. A two-state HMM modelling a DNA sequence, the first generating AT-rich sequences, and the second generating CG-rich sequences. State transitions and their associated probabilities are indicated by arrows, and symbol emission emission probabilities for A, C, G, T for each state are indicated below the states. This model generates a state sequence as a Markov chain (middle) and each sequence generates a symbol according to its own emission probability distribution (bottom). The probability of the sequence is the product of the state transitions and the symbol emissions. For a given observed DNA sequence, the hidden state sequence that generated it, i.e. whether this position is in a CG-rich or an AT-rich segment, is inferred.

specifically all members within a protein family. Often, motifs centre around sites that are functionally important to all members of the family. Motifs can be derived by an expert to capture biological knowledge PROSITE (Bucher and Bairoch, 1994), or using automated methods PRATT (Jonassen, 1997), MEME (Bailey and Elkan, 1994) and MAST (Bailey and Gribskov, 1998). Finally, HMMSTR is a HMM which captures sequence as well as structure features based on the initiation-sites library of sequence-structure motifs (Bystroff *et al*., 2000).

1.3 Description of Hidden Markov Models

Hidden Markov Models (HMMs) are statistical models which are generally applicable to time series or linear sequences. They have been widely used in speech recognition applications (Rabiner, 1989), and have been introduced to bioinformatics in the late 80's (Churchill, 1989). A HMM can be visualised as a *finite state machine*. Finite state machines move through a series of states and produce some kind of output, either when the machine has reached a particular state or when it is moving from state to state. The HMM generates a protein sequence by *emitting* amino acids as it progresses through a series of states. Each state has a table of amino acid *emission probabilities*, and *transition probabilities* for moving from state to state. Transition probabilities define a distribution over the possible next states.

Any sequence can be represented by a path through the model. This path follows the Markov assumption, that is, the choice of the next state is only dependent on the choice of the current state. However, the state sequence is not known; it is hidden. Finally, the alignment probability, given the model, is the product of the emission and transition probabilities along the path (Karchin, 1999). Figure 3, adapted from (Eddy, 1996), represents a two-state HMM modelling a DNA sequence, with heterogeneous base composition.

1.3.1 *Parameter Estimation*

HMM design issues include defining how many states to model, and how to estimate the transition and emission probabilities. The former is dealt with in section 2.2.

When the paths are known for the training sequences, a Maximum Likelihood approach to estimating the emission and transition parameters is used as follows:

$$a_{kl} = \frac{A_{kl}}{\sum_{l'} A_{kl'}} \tag{1}$$

$$e_k(b) = \frac{E_k(b)}{\sum_{b'} E_k(b')} \tag{2}$$

Where $E_k(b)$ is the number of instances in the training data where symbol b is emitted in state k and A_{kl} is the number of transitions from state k to l in the training data. Sometimes, pseudo-counts are added to account for transitions not seen in the training data.

On the other hand, when the paths are unknown, the Baum-Welch algorithm, an Expectation Maximisation algorithm, initialises parameters and iteratively re-estimates them using the forward backward values (refer to Durbin *et al*. (1998) for a thorough explanation).

1.3.2 The Viterbi Algorithm The most probable state sequence (or path) can be determined recursively using the Viterbi algorithm. If the probability $V_k(i)$ of the most probable path ending in state k with observation i is known for all the states k, then these probabilities can be calculated for observation x_{i+1} as:

$$V_l(i+1) = e_l(x_{i+1})max_k(V_k(i)a_{kl}) \tag{3}$$

All sequences start in the begin state, and by keeping pointers backward, the actual state sequence can be found by backtracking. The full algorithm is presented in Algorithm 1. A very good explanation of theory and implementation of HMMs along with examples can be found in (Durbin *et al*., 1998).

```
Initialisation (i=0):      $V_S(0) = 1, V_k(0) = 0$ for all k  0
Recursion (for i=1...L):   $V_l(i) = e_l(x_i)max_k(V_k(i-1)a_{kl}$
                           $ptr_i = argmax_k(V_k(i-1)a_{kl}$
Termination:               $P(x,\pi) = max_k V_k(L)a_{k0}$
                           $\pi_L = argmax_k(V_k(L)a_{k0})$
Traceback(for i=L...1):    $\pi_{i-1} = ptr_i(\pi_i)$
```

where:
k,l= hidden states
S= start state
E= end state
π = the most probable path
L= the length of the emission sequence
ptr= the backward pointer.
x_i= the ith symbol in the emission sequence
a_{kl}=the transition probability from state k to l
$e_l(b)$=the emission probability of symbol b from state l
$V_k(i)$=the most probable path ending in state k with observation i

Algorithm 1: The Viterbi algorithm for estimating the most probable path

Pair HMMs
Pair HMMs are HMMs used for alignment of two sequences. They proceed in the same way explained above, except that they do not emit symbols, but a match state, an insert state, or a delete state. The design of the pair HMM used for this project is explained in section 2.2.

1.4 Aims of the work

There exists more known protein sequences than protein structures, structure prediction relies heavily on sequence alignment of the protein of unknown structure to other proteins of known structure. According to recent CASP reports (Tramontano *et al*., 2001; Bradley *et al*., 2003), alignment inaccuracy is still a major problem. The quality of alignment does not correlate well with the level of sequence identity, and alignment algorithms do not perform as well for highly divergent sequences. Therefore, improving sequence alignments ultimately leads to better structure and function prediction methods.

Several techniques, including the motif detection and profile HMMs described above have been developed to tackle the task. This work investigates the use of pair HMMs for sequence alignment. It uses a locally-written hidden Markov model compiler (HMMoc). The parameters are estimated from the HOMSTRAD alignments. An evolutionary model is not incorporated. The resulting alignments are evaluated against the HOMSTRAD alignments. The performance of the system is low, 41.20% for basic sequence to sequence alignment, with only a slight improvement when structure information is added. The next sections explain the HMM design, the workings of the system, and the steps undertaken for bug detection. Unfortunately, due to the time constraints of the project, no final answer has been reached.

2 APPROACH

Sequence alignment is the first step towards protein structure, which in turn is a step towards function prediction. Because sequences can be very similar or highly divergent, it would be useful to have an idea of evolutionary distances. Ideally, the model should also take in consideration the varying substitution rates and patterns across sites, the effect of variation in selection across sites (purifying vs. positive selection), the interaction between sites as well as the structural context. A high level representation of the different components of a pair HMM alignment algorithm that uses an evolutionary model are shown in figure 4. Surely, incorporating phylogenetic information would improve alignment accuracy. However, the time constraints of this project made it unfeasible to incorporate such a model. Instead, the same approach presented in figure 4 was taken without including the evolutionary model. More specifically, figure 5 shows the different steps undertaken for this work. This includes data preprocessing, parameter estimation, the use of *HMMoc*, a hidden Markov model compiler written locally, and finally the evaluation.

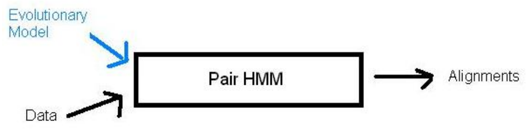

Fig. 4. The overall approach for this work does not include an evolutionary model (shown in blue).

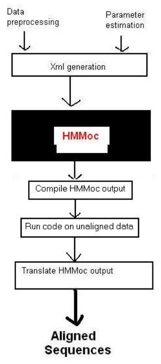

Fig. 5. The different steps undertaken for this work include data preprocessing, parameter estimation, alignment using HMMoc, translating the HMMoc alignments and evaluating them.

2.1 Dataset

The dataset used consisted of HOMSTRAD alignments of 1031 protein families downloaded in September 2004. The family size varies from 2 members for XPG_C to 41 for globin. For the untrimmed data (as it appears in the HOMSTRAD alignments), the total number of columns is 251,471, and the number of aligned columns (no gap) is 210,665.

As a first step, the whole set of protein families were used for parameter estimation and evaluation. Different subsets should be used for parameter estimation and evaluation. However, due to time constraints, moving to the next step was not done. Each protein family is represented by the alignments of its sequences, with additional information such as secondary structure, Ooi number and solvent accessibility. The two sequences selected to be aligned were the first and last sequences in the family alignment file, as they are assumed to be the most divergent. Figure 6 is an example of the XPG_C family (two members), including sequence, structure and Ooi number information.

2.2 HMM Design

The HMM design used for global alignment of two protein sequences is presented in figure 7. There are two silent states, the begin and end states which do not emit any symbols. The three non-silent states include the match state indicated by an M which has an emission probability distribution $p_{x_i y_i}$ for emitting the aligned pair x_i:y_i. Both states X and Y have a probability distribution of emitting x_i in sequence x and y_i in sequence y respectively against a gap. Transition probabilities between the states are also estimated from the data. They are shown on top of transition arrows, and follow the condition that the sum of all transition probabilities leaving a state sum to 1.

P1;1a77
sequence
```
 LTREKLIELAILVGTDYNPGGIKGIGLKKALEIVRHSKDPLAKFQKQSDVDL
 YAIKEFFLNPPVTDNYNLVWRDPDEEGILKFLCDEHDFSEERVKNGLERLKK
 AIKSGKQSTLESWFKR
```
P1;1b43a
sequence
```
 ISLDDLIDIAIFMGTDYNPGGVKGIGFKRAYELVRS-GVAKDVLKKEV-EY
 YDEIKRIFKEPKVTDNYSLSLKLPDKEGIIKFLVDENDFNYDRVKKHVDKL
 YNLIANKT--------
```
P1;1a77
secondary structure and phi angle
```
 CCHHHHHHHHHHHPCCCCCPPCCPCCHHHHHHHHHCCCCHHHHCHHHCCCCH
 HHHHHHHHCCCCCCCCCCCCCCCCHHHHHHHHCCCCPCCHHHHHHHHHHHHH
 HHHHHHHHHCCCCCCC
```
P1;1b43a
secondary structure and phi angle
```
 CCHHHHHHHHHHHPCCCCCPPCCPCCHHHHHHHHHC-PCHHHHHHHHC-CP
 HHHHHHHHHCCCCCCCCCCCCCCCCHHHHHHHHCCCCPCCHHHHHHHHHHH
 HHHHHHHC--------
```
P1;1b43a
Ooi number
```
 2223334554553432232244223222233233222234433321222334
 3333222211110000011222222223222221111222222222322222
 2221111111110000
```
P1;1a77
Ooi number
```
 22233345445433323322342222234333322-22432343233-22
 332332222111100000111222232232222211112122222223222
 22221111--------
```

Fig. 6. The XPG_C family alignment as it appears in HOMSTRAD, including sequence and structure. The sequence information includes the amino acids. The secondary structure and phi angle includes C for coil, H for helix, E for sheet, and P for positive phi angle. The Ooi number, a count of the surrounding C alpha atoms in a 14 Angstrom radius, is presented in numbers.

The HMM does not model context dependence, it assumes independence of columns. It does not have a transition from state begin to end (no empty alignments), and disallows transitions from state X to Y (no insertion followed by a deletion, and no deletion followed by an insertion).

2.2.1 Parameter Estimation Parameter estimation was performed on the HOMSTRAD alignments using the maximum likelihood method explained in equations (1) and (2). The alignments were first trimmed to the subsequences delimited by the first and last aligned amino acid residues. The resulting emission and transition probabilities are presented in Appendix 2.

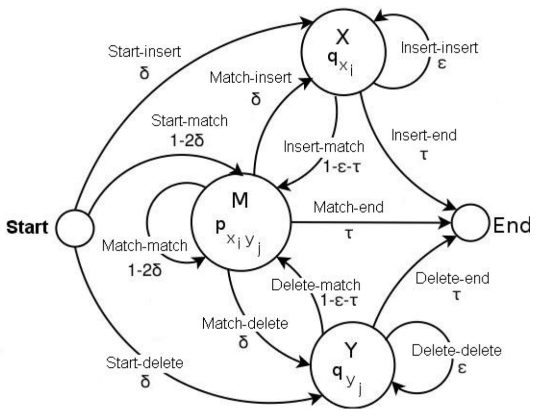

Fig. 7. The HMM design for the work presented in this report. The are two silent states, the begin and end states which do not emit anything. The three non-silent states are the Match state indicated by M, the insert state indicated by X, the delete state indicated by Y. The transitions between states are indicated by arrows.

2.3 HMMoc

HMMOC is a HMM compiler written by Gerton Lunter. It is meant as a high-level abstraction of hidden Markov models. The low-level details of the HMM computation is hidden from the user including the posterior probability calculations and path decoding. HMMOC requires the input of a HMM topology along with the emission and transition probabilities in an xml format (presented in appendix 3). It outputs the most probable path, and the forward backward probabilities as needed. It allows for multiple output tapes (pair HMMS) and higher order states. This tool is very useful, as it simplifies the programming of the technicalities of HMMS, while requiring the user to think about the HMM design and topology. However, as it is a recently developed package, and was not tested especially for pair HMMS, some time was spent debugging it.

2.4 Adding Structural Information

A study by Deane *et al*. (2004) investigated the most informative features to use for improvement of alignment accuracy. Their results illustrate that indels occur mostly in coil regions about twice as high compared to α helices. Also, indels are twice as likely to occur in α helices than in β sheets. The most significant result relates to the Ooi number; a count of C-α atoms within a radius of 14 $\mathring{A}$ of the residue's own C-α. The logarithm of the indel propensity is very nearly linearly related to the Ooi number. This suggests that incorporating secondary structure information, as α helices and β strands could improve alignment. Also, incorporating Ooi number information would improve it further.

The HOMSTRAD alignments contain secondary structure and ϕ angle information as well as Ooi number information. The experiment following the basic sequence information experiment included structure information as classes of α helices, β strands, positive ϕ angles and coils. Parameter estimation was conducted as before.

The Ooi numbers in the HOMSTRAD alignments range from 0-8, with very few occurrences of 9. These were grouped into 4 classes, because of syntax limitations imposed by HMMOC. HMMOC only takes single characters in the alphabet, and there is not a sufficient number of ASCII characters to assign a unique character to each mapping of sequence and Ooi number. This also holds for combining sequence structure and Ooi number. Parameter estimation was carried out as before. The results are presented below.

2.5 Evaluation

A few programs had to be built to handle generating input data in the format required for HMMOC and also translating its output. Figure 8 includes a typical example of the HMMOC output. This needs to be translated into an alignment, and then an evaluation program produces the final result.

The result represents the percentage of correctly-aligned columns, excluding gapped ones, obtained by the program divided by the aligned columns, excluding gapped ones, given by HOMSTRAD. This value is averaged over all sequences giving the final result.

Evaluation was conducted using the HOMSTRAD database. HOMSTRAD contains structural alignments, and we therefore use it as an "independent" set of gold standard alignments. Because the HMM is for global alignment, the sequences are trimmed to the subsequences between the first and last amino acid.

3 RESULTS AND ANALYSIS

The results obtained with the transition and emission probabilities estimated from the data for basic sequence information were 41.20%, and 42.05% for sequence augmented with structure information[2]. These results were surprisingly low, and adding structure information did not improve the results dramatically. This pointed to the presence of a bug in the code. A thorough re-check of all estimation and transition probabilities was carried out at this stage.

[2] Training on the first and second sequences expectedly decreased the performance.

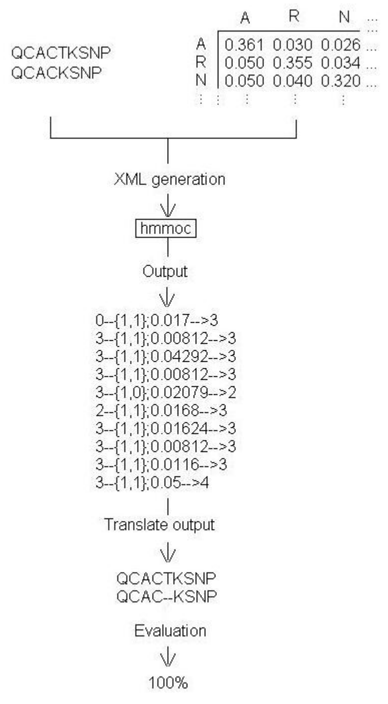

Fig. 8. A typical example (except for the score) of alignment performed by the pair HMM

3.1 Comparison with Other Alignment Algorithms

The first step undertaken was to compare the results with other alignment algorithms: Clustalw and EMBOSS Stretcher (a faster version of the Needleman-Wunch (Myers and Miller, 1998)) were selected. HMMer was considered, but it is a more sophisticated method than the pair HMM; it requires the whole protein family to build a profile and then aligns two members of that same family. This would definitely give it an advantage, producing much better results. Clustalw performs at 76.88%, and EMBOSS Stretcher, performs at 80.16%. A closer look at the software was taken.

As the model does not allow for transitions from delete to insert and insert to delete states, a measure of how often they occur in the HOMSTRAD data is useful. Insert to delete and delete to insert occur in 12% of the alignments. These will of course be missed by the pair HMM.

3.2 Rewriting the Code

Clustalw and Stretcher were fed the same data as the pair HMM, they both use the evaluation code used for the pair HMM. The difference lies in the parameters used, the code around HMMoc, and the use of HMMoc itself.

To eliminate the possibility that the code includes a programming bug (as opposed to a conceptual error), all the code written for the project (in Perl initially), including the evaluation code was rewritten in Java. The results were exactly the same.

3.3 Changing the Parameters

Comparing the gold standard alignments with the alignments produced by the pair HMM suggested a difference in gap opening. The HOMSTRAD alignments contain several small gaps along the sequence. The pair HMM alignment would align the correct residues at the beginning, but would not open a gap when the gold standard does. Changing the emission and transition parameters was the next step.

3.3.1 Emission Probabilities I worked out the probability matrix from the BLOSUM62 matrix used by Clustalw. Using the resulting probability matrix with the original transition parameters gave a slight improvement of 41.41%. However, this does not account for the 35% difference with Clustalw.

Similarly, setting all the insertion and deletion emission parameters to the same value gave 39.45%.

3.3.2 Transition Parameters In the first model I tried, I set the transitions match to match, match to insert, and match to delete to be the same. Similarly, for delete to delete and delete to match, insert to insert and insert to match. This should give bad results, as it is equally likely to stay in a match or open a gap. The result was 34%.

Table 1 summarises the results in percentages for different combinations of transition probabilities explained in tables 2 and 3. The results vary from 36.53% to 41.71%. The highest result is obtained with a high match to match transition probability (80%), and delete/insert to match of 60% and delete to delete of 38%.

The same combination of transition parameters as specified in the tables is used with the BLOSUM matrix, only a small increase is obtained.

I also tried the Baum-Welch method for parameter estimation using HMMoc. The program gave some compilation errors, and there was no time to look at it any further.

At this stage, the 35% difference with Clustalw is still not explained. Several options were investigated. The bug is either due to a conceptual error, or to HMMoc itself. Due to the time constraints, it was not possible to pursue the bug tracking any further.

	Indel 1	Indel 2	Indel 3
Match 1	40.15	37.55	36.53
Match 2	41.61	41.52	37.37
Match 3	41.32	41.71	40.93

Table 1. Summary of the results for the different settings explained in tables 2 and 3

	Match-end	Match-insert	Match-delete	Match-match
Match 1	0.05	0.25	0.25	0.45
Match 2	0.05	0.15	0.15	0.65
Match 3	0.05	0.075	0.075	0.80

Table 2. A table describing the labels match 1, match 2, match 3

	Delete/Insert-end	Delete/Insert-match	Delete/Insert-delete
Indel 1	0.02	0.70	0.28
Indel 2	0.02	0.60	0.38
Indel 3	0.02	0.50	0.48

Table 3. A table describing the labels Indel 1, indel 2, indel 3

4 CONCLUSION AND FUTURE WORK

Protein structure prediction methods rely heavily on sequence alignment. This project examined the use of pair HMMs for global alignment of two protein sequences. A newly developed HMM compiler was used. HMMoc requires the HMM topology and emission probabilities in an xml format, and outputs the most probable path, abstracting away all the calculations. The parameter estimation and the evaluation were conducted using the HOMSTRAD alignments. The performance of the pair HMM for basic sequence information was 41.20%, with a slight improvement when structure information is included. Relative to other alignment methods, this was at least 35% inferior.

The possibility of the presence of a bug lead to examining different paths. First, to eliminate the possibility of a programming bug, the whole code for the project, initially written in Perl, was re-written in Java. This gave the same results. Next, the BLOSUM62 matrix, the one used by Clustalw, was converted into a probability matrix and used. The results changed only slightly. After that, a combination of different emission probabilities (match, insert and delete) and transition probabilities were tried. Still, the results did not improve. It is clear that there is either a conceptual error, or HMMoc itself is buggy. Given more time, it would be interesting to elucidate which of these two hypotheses is factually correct.

4.1 Including More information

It would have been be useful to measure the impact of adding more information to the basic sequence alignment. Adding secondary structure information and Ooi number information, as well as a combination of both, was undertaken. None gave a considerable improvement. Using a system that functions correctly, these are expected to improve alignment accuracy. Other options include hydrophobicity, and cis-peptide information, whose absence is not interesting, but presence potentially useful.

4.2 Evolutionary Model

Current alignment tools, including SAM (Karplus *et al.*, 1998) and HMMER (Eddy, 2001) use sequence weighting to correct for phylogenetic bias during training. Incorporating a statistical model of protein sequence evolution allows for estimating selection pressures acting on the sequence family. The standard evolutionary model is the TKF91 model (Thorne *et al.*, 1991), which deals with finite sequences and allows only single residue indel events. An improvement allowing indels of arbitrary length was developed by Miklos *et al*. (2004)

Evolutionary models give a joint distribution of all sequences in a family, conditioned on their phylogenetic tree (Holm and Bruno , 2001). Correlation between sequences are therefore built into the model, which improves alignment accuracy.

4.3 Higher Order Markov Chain

So far, the Markov property states that the probability of a state depends only on the probability of the previous state. A higher order Markov model would allow for building more HMM memory into the states, to capture residue dependencies. Inhomogeneous Markov chain models would allow for different probability distributions at different positions or regions. This includes more information into the model, and would produce better results.

REFERENCES

Bailey, T. and Elkan, C. (1994) Fitting a mixture model by expectation maximization to discover motifs in biopolymers, *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology*, 28-36, AAAI Press, Menlo Park, California.

Bailey, T. and Gribskov, M. (1998) Combining evidence using p-values: application to sequence homology searches, *Bioinformatics*, **14**, 48-54.

Bourne, P., Weissig, H. (2003) Structural Bioinformatics, *Wiley-Liss*.

Bradley, P. *et al* (2003) Rosetta predictions in CASP5: Successes, failures and prospects for complete automation. *Proteins: Structure, Function, and Genetics.*, **53**, 457-468.

Bucher, P. and Bairoch, A. (1994) A generalized profile syntax for biomolecular sequence motifs and its function in automatic sequence interpretation. *ISMB*, 53-61.

Bystroff, C., Thorsson, V. and Baker, D. (2000) HMMSTR: A hidden Markov model for local sequence-structure correlations in proteins, *J. Mol.Biol.*, **301**, 173-190.

Churchill, G. A. (1998) Stochastic models for heterogeneous DNA sequences. *Bull Math Biol*, **51**, 79-94.

Deane, C., Perdersen, J. and Lunter, G. (2004) Insertions and deletions in protein alignment, *Proc. 8th Intl. Conf. Mol. Biol. (Recomb04)*, available:*http://recomb04.sdsc.edu/posters/deaneATstats.ox.ac.uk_207.pdf*.

Durbin, R., Eddy, S., Krogh, A., Mitchison, G. (1998) Biological Sequence Analysis: Probabilistic models of proteins and nucleic acids, *Cambridge University Press*.

Eddy, S. (1995) Multiple alignment using hidden Markov models. *Proc. Third Intl. Conf. Intelligent Systems for Molecular Biology*, 114-120.

Eddy, S. (1996) Hidden Markov Models. *Current Opinion in Structural Biology*, **6**, 361-365.

Eddy, S. (1998) Profile Hidden Markov Models. *Bioinformatics*, **14**, 755-763.

Eddy, S. (2001) HMMER: Profile hidden Markov models for biological sequence analysis. Available: *http://hmmer.wustl.edu/*

Fischer, D. *et al* (2003) CAFASP3: The third critical assessment of fully automated structure prediction methods. *Proteins*, **53**, 503-516.

Ginalski, K. *et al* (2003) ORFeus: Detection of distant homology using sequence profiles and predicted secondary structure. *Nucleic Acids Research*, **31**, 4804-3807.

Haussler, D. *et al* (1993) Protein Modeling using hidden Markov models: Analysis of Globin. *Proceedings of the Hawaii International Conference on System Sciences*, **1**, IEEE Computer Society Press, 792-802

Hegyi, H. & Gerstein, M. (1999) The relationship between protein structure and function: a comprehensive survey with application to the yeast genome. *J. Mol. Biol*, **288**, 147-164.

Holm, L. & Sander, C. (1994) The FSSP database of structurally aligned protein fold families. *Nucleic Acids Res.*, **22(17)**, 3600-9.

Holmes, I. & Bruno, C. (2001) Evolutionary HMMs: A Bayesian approach to multiple alignment. *Bioinformatics*, **17(9)**, 803-20.

Jonassen, I. (1997) Efficient discovery of conserved patterns using a pattern paragraph. *CABIOS*, **13**, 509-522.

Karchin, R. (1999) Hidden Markov Models and Protein Sequence Analysis. *ISMB 1999*, Available on: *https://www.cse.ucsc.edu/research/compbio/ismb99.handouts/KK185FP.html*

Karplus, K., Barrett, C. and Hughey, R. (1998) Hidden Markov Models for detecting Remote Protein Homologies, *Bioinformatics*, **14(10)**, 846-856.

Karplus, K. *et al* (2001) What is the value added by human intervention in protein structure prediction. *Proteins.*, **Suppl.5**, 86-91.

Koh, I. *et al* (2003) EVA: Evaluation of protein structure prediction servers. *Nucleic Acids Research*, **31**, 3311-3315.

Kelley, L. A., MacCallum, R. M., Sternberg, M. J. E. (2000) Enhanced genome annotation using structural profiles in the program 3D-PSSM. *J. Mol.Biol.*, **299**, 499-520.

Krogh, A. *et al* (1994) Hidden markov models in computational biology: applications to protein modelling. *J. Mol.Biol.*, **235**, 1501-1531.

Miklos I., Lunter G.A. & Holmes I. (2004) A ”Long Indel” Model For Evolutionary Sequence Alignment. *Mol. Biol. Evol.*, **21(3)**, 529-540.

Mizuguchi, K. *et al* (1998) HOMSTRAD: A database of protein structure alignments for homologous families, *Protein Sci.*, **7(11)**, 2469-71.

Murzin, A. G., Brenner S. E., Hubbard T., Chothia C. (1995) SCOP: A structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol.*, **247**, 536-540.

Myers, E. W. & Miller, W. (1998) Optimal alignments in linear space. *CABIOS*, **4(1)**, 11-17.

Orengo, C.A., Michie, A.D., Jones, S., Jones, D.T., Swindells, M.B., and Thornton, J.M. (1997) CATH: A hierarchic classification of protein domain structures, *Structure*, **5(8)**, 1093-1108.

Rabiner, L. R. (1989) A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE*, **77 (2)**, 257-285.

Russell, R. B., Saqi, M. A. S., Bates, P. A., Sayle, R. A.& Sternberg, M. J. E. (1998). Recognition of analogous and homologous folds. Assessment of prediction success and associated alignment accuracy using empirical matrices. *Protein Eng.*, **11**, 1-9.

Rychlewski, L., Fisher, D., Elofsson, A. (2003) LiveBench-6: Large-scale automated evaluation of protein structure prediction servers. *Proteins*, **53**, 542-547.

Rychlewski, L. *et al* Comparison of sequence-profiles: strategies for structural prediction using sequence information. *Protein Science*, **12**, 2262-2272.

Shi, J., Blundell, T. L., Mizuguchi, K. (2001) FUGUE: Sequence-structure Homology Recognition Using Environment-specific substitution table and structure-dependent gap penalties. *J. Mol.Biol.*, **310**, 243-257.

Soding, J. (2004) Protein homology detection by HMM-HMM comparison. *Bioinformatics Advance Access*, Nov 5th.

Tang, C. *et al* (2003) On the role of structural information in remote homology detection and sequence alignment: new methods using hybrid sequence profiles. *J. Mol. Biol.*, **334**, 1034-1062.

Thorne, J. L., Kishino, H. and Felsenstein, J. (1991) An evolutionary model for maximum likelihood alignment of DNA sequences. *J. Mol. Evol.*, **33**, 114-124.

Tommi, K. and Akiyama, Y. (2004) FORTE: A profile-profile comparison tool for protein fold recognition. *Bioinformatics*, **20**, 594-595.

Tramontano, A., Leplae, R. and Morea, V. (2001) Analysis and assessment of comparative modeling predictions in CASP4. *Proteins*, **45 Suppl.5**, 22-38.

Xu, Y. and Xu, D. (2000) Protein threading using PROSPECT: Design and evaluation *Proteins: Structure, Function, and Genetics.*, **40**, 343-354.

1 AMINO ACIDS

Name	Abbr.	Linear structure formula
Alanine	ala A	CH3-CH(NH2)-COOH
Arginine	arg R	HN=C(NH2)-NH-(CH2)3-CH(NH2)-COOH
Asparagine	asn N	H2N-CO-CH2-CH(NH2)-COOH
Aspartic acid	asp D	HOOC-CH2-CH(NH2)-COOH
Cysteine	cys C	HS-CH2-CH(NH2)-COOH
Glutamine	gln Q	H2N-CO-(CH2)2-CH(NH2)-COOH
Glutamic acid	glu E	HOOC-(CH2)2-CH(NH2)-COOH
Glycine	gly G	NH2-CH2-COOH
Histidine	his H	NH-CH=N-CH=C-CH2-CH(NH2)-COOH
Isoleucine	ile I	CH3-CH2-CH(CH3)-CH(NH2)-COOH
Leucine	leu L	(CH3)2-CH-CH2-CH(NH2)-COOH
Lysine	lys K	H2N-(CH2)4-CH(NH2)-COOH
Methionine	met M	CH3-S-(CH2)2-CH(NH2)-COOH
Phenylalanine	phe F	Ph-CH2-CH(NH2)-COOH
Proline	pro P	NH-(CH2)3-CH-COOH
Serine	ser S	HO-CH2-CH(NH2)-COOH
Threonine	thr T	CH3-CH(OH)-CH(NH2)-COOH
Tryptophan	trp W	Ph-NH-CH=C-CH2-CH(NH2)-COOH
Tyrosine	tyr Y	HO-p-Ph-CH2-CH(NH2)-COOH
Valine	val V	(CH3)2-CH-CH(NH2)-COOH

2 EMISSION AND TRANSITION PROBABILITIES

2.1 Emission Parameters for the Insertion and Deletion states

	Insertion	Deletion
A	0.074	0.057
R	0.045	0.058
N	0.056	0.084
D	0.080	0.091
C	0.008	0.035
Q	0.041	0.072
E	0.076	0.077
G	0.099	0.086
H	0.022	0.063
I	0.036	0.039
L	0.063	0.045
K	0.073	0.082
M	0.017	0.053
F	0.031	0.048
P	0.065	0.092
S	0.068	0.079
T	0.057	0.066
W	0.012	0.054
Y	0.030	0.054
V	0.047	0.042

2.2 Emission Parameters for the Match State

2.3 Transition Parameters

	A	R	N	D	C	Q	E	G	H	I
A	0.361	0.030	0.026	0.031	0.014	0.029	0.048	0.062	0.011	0.032
R	0.050	0.357	0.034	0.035	0.005	0.045	0.055	0.032	0.022	0.022
N	0.050	0.040	0.321	0.087	0.007	0.037	0.055	0.063	0.026	0.018
D	0.046	0.031	0.066	0.396	0.004	0.034	0.100	0.046	0.018	0.012
C	0.083	0.017	0.021	0.014	0.463	0.015	0.017	0.028	0.009	0.037
Q	0.066	0.062	0.043	0.052	0.006	0.258	0.099	0.035	0.029	0.022
E	0.063	0.043	0.037	0.089	0.004	0.057	0.354	0.036	0.017	0.019
G	0.070	0.022	0.036	0.035	0.006	0.017	0.031	0.574	0.010	0.011
H	0.041	0.048	0.050	0.044	0.006	0.047	0.049	0.031	0.350	0.022
I	0.045	0.019	0.013	0.012	0.009	0.014	0.021	0.014	0.008	0.334
L	0.048	0.024	0.013	0.014	0.009	0.017	0.021	0.016	0.011	0.100
K	0.055	0.101	0.044	0.047	0.003	0.050	0.075	0.037	0.021	0.022
M	0.058	0.027	0.021	0.020	0.010	0.023	0.026	0.022	0.013	0.086
F	0.038	0.016	0.014	0.014	0.010	0.014	0.017	0.018	0.015	0.055
P	0.063	0.028	0.024	0.037	0.005	0.022	0.043	0.037	0.012	0.022
S	0.096	0.036	0.049	0.055	0.011	0.030	0.050	0.059	0.016	0.019
T	0.074	0.033	0.037	0.040	0.010	0.030	0.048	0.037	0.014	0.036
W	0.039	0.024	0.016	0.015	0.004	0.014	0.022	0.022	0.014	0.032
Y	0.036	0.028	0.025	0.021	0.007	0.018	0.026	0.020	0.028	0.034
V	0.074	0.021	0.014	0.016	0.014	0.017	0.024	0.018	0.009	0.136

	L	K	M	F	P	S	T	W	Y	V
A	0.052	0.038	0.015	0.019	0.034	0.064	0.049	0.006	0.015	0.064
R	0.043	0.116	0.011	0.014	0.025	0.041	0.037	0.007	0.020	0.030
N	0.027	0.059	0.010	0.014	0.025	0.064	0.048	0.005	0.021	0.023
D	0.022	0.048	0.007	0.010	0.030	0.054	0.039	0.004	0.013	0.020
C	0.054	0.013	0.014	0.028	0.016	0.043	0.038	0.004	0.018	0.068
Q	0.043	0.079	0.013	0.016	0.028	0.047	0.045	0.005	0.018	0.034
E	0.030	0.067	0.009	0.011	0.030	0.044	0.042	0.005	0.015	0.028
G	0.019	0.029	0.006	0.010	0.022	0.044	0.027	0.004	0.010	0.017
H	0.043	0.053	0.012	0.027	0.023	0.039	0.033	0.009	0.045	0.028
I	0.153	0.022	0.031	0.039	0.017	0.019	0.034	0.008	0.021	0.167
L	0.423	0.024	0.041	0.053	0.016	0.021	0.029	0.009	0.024	0.087
K	0.037	0.317	0.011	0.011	0.032	0.042	0.044	0.004	0.017	0.030
M	0.175	0.030	0.244	0.047	0.016	0.028	0.041	0.011	0.026	0.076
F	0.116	0.016	0.024	0.401	0.013	0.022	0.023	0.022	0.095	0.057
P	0.032	0.040	0.007	0.012	0.485	0.044	0.036	0.005	0.014	0.032
S	0.034	0.043	0.010	0.016	0.036	0.294	0.088	0.005	0.017	0.036
T	0.047	0.045	0.016	0.017	0.029	0.089	0.315	0.005	0.016	0.062
W	0.061	0.016	0.016	0.066	0.015	0.020	0.022	0.477	0.071	0.034
Y	0.061	0.027	0.015	0.108	0.017	0.026	0.025	0.027	0.408	0.043
V	0.109	0.024	0.022	0.033	0.020	0.028	0.048	0.007	0.022	0.344

Start-match	1033/1034
Start-insert	(1/2)/1034
Start-delete	(1/2)/1034
Match-end	1033/212016
Match-insert	(6817/2)/212016
Match-delete	(6817/2)/212016
Match-match	204166/212016
Delete-end	1/27413
Delete-match	6817/27413
Delete-delete	20595/27413
Insert-end	1/27413
Insert-match	6817/27413
Insert-insert	20595/27413

3 XML INPUT FOR HMMOC

```
<?xml version="1.0"?>

<hml debug="true">

<author>Naila Mimouni</author>

<alphabet id="AminoAcids"> ARNDCQEGHILKMFPSTWYV </alphabet>
```

```
<output id="sequence1"> <alphabet idref="AminoAcids"/> <identifier
type="length" value="iLen1"/> <identifier type="sequence"
value="iSeq1"/> <code type="parameter" value="int iLen1"/> <code
type="parameter" value="char* iSeq1"/> </output>

<output id="sequence2"> <alphabet idref="AminoAcids"/> <identifier
type="length" value="iLen2"/> <identifier type="sequence"
value="iSeq2"/> <code type="parameter" value="int iLen2"/> <code
type="parameter" value="char* iSeq2"/> </output>

<hmm id="Aligner">

<description> trimmed_probs </description>

<outputs>
 <output idref="sequence1"/>
 <output idref="sequence2"/>
</outputs>

<block id="block1">
 <state id="start"/>
</block>

<block id="block2">
 <state id="Match" emission="emitMatch"></state>
 <state id="Insertion" emission="emitInsertion"></state>
 <state id="Deletion" emission="emitDeletion"></state>
</block>

<block id="block3">
 <state id="end" emission="empty"/>
</block>

<graph>
 <block idref="block1"/>
 <block idref="block2"/>
 <block idref="block3"/>
</graph>

<emission id="empty">
  <probability><code type="expression"> 1.0 </code></probability>
 </emission>

<emission id="emitMatch">
  <output idref="sequence1"/>
  <output idref="sequence2"/>
  <probability>
   <code type="statement">
    <identifier output="sequence1" value="iSymb1"/>
    <identifier output="sequence2" value="iSymb2"/>
    <identifier type="result" value="iResult"/>
    <![CDATA[
      switch (iSymb1) {
```

```
case ’A’:
switch (iSymb2) {
 case ’A’: iResult =0.361 ; break;
 case ’R’: iResult =0.030 ; break;
 case ’N’: iResult =0.026 ; break;
 case ’D’: iResult =0.031 ; break;
 case ’C’: iResult =0.014 ; break;
 case ’Q’: iResult =0.029 ; break;
 case ’E’: iResult =0.048 ; break;
 case ’G’: iResult =0.062 ; break;
 case ’H’: iResult =0.011 ; break;
 case ’I’: iResult =0.032 ; break;
 case ’L’: iResult =0.052 ; break;
 case ’K’: iResult =0.038 ; break;
 case ’M’: iResult =0.015 ; break;
 case ’F’: iResult =0.019 ; break;
 case ’P’: iResult =0.034 ; break;
 case ’S’: iResult =0.064 ; break;
 case ’T’: iResult =0.049 ; break;
 case ’W’: iResult =0.006 ; break;
 case ’Y’: iResult =0.015 ; break;
 case ’V’: iResult =0.064 ; break;
}
case ’R’:
switch (iSymb2) {
 case ’A’: iResult =0.050 ; break;
 case ’R’: iResult =0.355 ; break;
 case ’N’: iResult =0.034 ; break;
 case ’D’: iResult =0.035 ; break;
 case ’C’: iResult =0.005 ; break;
 case ’Q’: iResult =0.045 ; break;
 case ’E’: iResult =0.055 ; break;
 case ’G’: iResult =0.032 ; break;
 case ’H’: iResult =0.022 ; break;
 case ’I’: iResult =0.022 ; break;
 case ’L’: iResult =0.043 ; break;
 case ’K’: iResult =0.116 ; break;
 case ’M’: iResult =0.011 ; break;
 case ’F’: iResult =0.014 ; break;
 case ’P’: iResult =0.025 ; break;
 case ’S’: iResult =0.041 ; break;
 case ’T’: iResult =0.037 ; break;
 case ’W’: iResult =0.007 ; break;
 case ’Y’: iResult =0.020 ; break;
 case ’V’: iResult =0.030 ; break;
}
case ’N’:
switch (iSymb2) {
 case ’A’: iResult =0.050 ; break;
 case ’R’: iResult =0.040 ; break;
 case ’N’: iResult =0.320 ; break;
 case ’D’: iResult =0.087 ; break;
 case ’C’: iResult =0.007 ; break;
 case ’Q’: iResult =0.037 ; break;
 case ’E’: iResult =0.055 ; break;
 case ’G’: iResult =0.063 ; break;
 case ’H’: iResult =0.026 ; break;
 case ’I’: iResult =0.018 ; break;
 case ’L’: iResult =0.027 ; break;
 case ’K’: iResult =0.059 ; break;
 case ’M’: iResult =0.010 ; break;
 case ’F’: iResult =0.014 ; break;
 case ’P’: iResult =0.025 ; break;
 case ’S’: iResult =0.064 ; break;
 case ’T’: iResult =0.048 ; break;
 case ’W’: iResult =0.005 ; break;
 case ’Y’: iResult =0.021 ; break;
 case ’V’: iResult =0.023 ; break;
}
case ’D’:
switch (iSymb2) {
 case ’A’: iResult =0.046 ; break;
 case ’R’: iResult =0.031 ; break;
 case ’N’: iResult =0.066 ; break;
 case ’D’: iResult =0.397 ; break;
 case ’C’: iResult =0.004 ; break;
 case ’Q’: iResult =0.034 ; break;
 case ’E’: iResult =0.100 ; break;
 case ’G’: iResult =0.046 ; break;
 case ’H’: iResult =0.018 ; break;
 case ’I’: iResult =0.012 ; break;
 case ’L’: iResult =0.022 ; break;
 case ’K’: iResult =0.048 ; break;
 case ’M’: iResult =0.007 ; break;
 case ’F’: iResult =0.010 ; break;
 case ’P’: iResult =0.030 ; break;
 case ’S’: iResult =0.054 ; break;
 case ’T’: iResult =0.039 ; break;
 case ’W’: iResult =0.004 ; break;
 case ’Y’: iResult =0.013 ; break;
 case ’V’: iResult =0.020 ; break;
}
case ’C’:
switch (iSymb2) {
 case ’A’: iResult =0.083 ; break;
 case ’R’: iResult =0.017 ; break;
 case ’N’: iResult =0.021 ; break;
 case ’D’: iResult =0.014 ; break;
 case ’C’: iResult =0.462 ; break;
 case ’Q’: iResult =0.015 ; break;
 case ’E’: iResult =0.017 ; break;
 case ’G’: iResult =0.028 ; break;
 case ’H’: iResult =0.009 ; break;
 case ’I’: iResult =0.037 ; break;
 case ’L’: iResult =0.054 ; break;
 case ’K’: iResult =0.013 ; break;
 case ’M’: iResult =0.014 ; break;
 case ’F’: iResult =0.028 ; break;
 case ’P’: iResult =0.016 ; break;
 case ’S’: iResult =0.043 ; break;
 case ’T’: iResult =0.038 ; break;
 case ’W’: iResult =0.004 ; break;
 case ’Y’: iResult =0.018 ; break;
 case ’V’: iResult =0.068 ; break;
}
case ’Q’:
switch (iSymb2) {
 case ’A’: iResult =0.066 ; break;
 case ’R’: iResult =0.062 ; break;
 case ’N’: iResult =0.043 ; break;
 case ’D’: iResult =0.052 ; break;
 case ’C’: iResult =0.006 ; break;
 case ’Q’: iResult =0.258 ; break;
 case ’E’: iResult =0.099 ; break;
 case ’G’: iResult =0.035 ; break;
 case ’H’: iResult =0.029 ; break;
 case ’I’: iResult =0.022 ; break;
 case ’L’: iResult =0.043 ; break;
 case ’K’: iResult =0.079 ; break;
 case ’M’: iResult =0.013 ; break;
 case ’F’: iResult =0.016 ; break;
 case ’P’: iResult =0.028 ; break;
 case ’S’: iResult =0.047 ; break;
 case ’T’: iResult =0.045 ; break;
 case ’W’: iResult =0.005 ; break;
 case ’Y’: iResult =0.018 ; break;
 case ’V’: iResult =0.034 ; break;
}
case ’E’:
switch (iSymb2) {
 case ’A’: iResult =0.063 ; break;
 case ’R’: iResult =0.043 ; break;
 case ’N’: iResult =0.037 ; break;
 case ’D’: iResult =0.089 ; break;
 case ’C’: iResult =0.004 ; break;
 case ’Q’: iResult =0.057 ; break;
 case ’E’: iResult =0.356 ; break;
 case ’G’: iResult =0.036 ; break;
 case ’H’: iResult =0.017 ; break;
 case ’I’: iResult =0.019 ; break;
 case ’L’: iResult =0.030 ; break;
 case ’K’: iResult =0.067 ; break;
 case ’M’: iResult =0.009 ; break;
 case ’F’: iResult =0.011 ; break;
 case ’P’: iResult =0.030 ; break;
 case ’S’: iResult =0.044 ; break;
 case ’T’: iResult =0.042 ; break;
 case ’W’: iResult =0.005 ; break;
 case ’Y’: iResult =0.015 ; break;
 case ’V’: iResult =0.028 ; break;
}
case ’G’:
```

```
switch (iSymb2) {
 case ’A’: iResult =0.070 ; break;
 case ’R’: iResult =0.022 ; break;
 case ’N’: iResult =0.036 ; break;
 case ’D’: iResult =0.035 ; break;
 case ’C’: iResult =0.006 ; break;
 case ’Q’: iResult =0.017 ; break;
 case ’E’: iResult =0.031 ; break;
 case ’G’: iResult =0.574 ; break;
 case ’H’: iResult =0.010 ; break;
 case ’I’: iResult =0.011 ; break;
 case ’L’: iResult =0.019 ; break;
 case ’K’: iResult =0.029 ; break;
 case ’M’: iResult =0.006 ; break;
 case ’F’: iResult =0.010 ; break;
 case ’P’: iResult =0.022 ; break;
 case ’S’: iResult =0.044 ; break;
 case ’T’: iResult =0.027 ; break;
 case ’W’: iResult =0.004 ; break;
 case ’Y’: iResult =0.010 ; break;
 case ’V’: iResult =0.017 ; break;
}
case ’H’:
switch (iSymb2) {
 case ’A’: iResult =0.041 ; break;
 case ’R’: iResult =0.048 ; break;
 case ’N’: iResult =0.050 ; break;
 case ’D’: iResult =0.044 ; break;
 case ’C’: iResult =0.006 ; break;
 case ’Q’: iResult =0.047 ; break;
 case ’E’: iResult =0.049 ; break;
 case ’G’: iResult =0.031 ; break;
 case ’H’: iResult =0.351 ; break;
 case ’I’: iResult =0.022 ; break;
 case ’L’: iResult =0.043 ; break;
 case ’K’: iResult =0.053 ; break;
 case ’M’: iResult =0.012 ; break;
 case ’F’: iResult =0.027 ; break;
 case ’P’: iResult =0.023 ; break;
 case ’S’: iResult =0.039 ; break;
 case ’T’: iResult =0.033 ; break;
 case ’W’: iResult =0.009 ; break;
 case ’Y’: iResult =0.045 ; break;
 case ’V’: iResult =0.028 ; break;
}
case ’I’:
switch (iSymb2) {
 case ’A’: iResult =0.045 ; break;
 case ’R’: iResult =0.019 ; break;
 case ’N’: iResult =0.013 ; break;
 case ’D’: iResult =0.012 ; break;
 case ’C’: iResult =0.009 ; break;
 case ’Q’: iResult =0.014 ; break;
 case ’E’: iResult =0.021 ; break;
 case ’G’: iResult =0.014 ; break;
 case ’H’: iResult =0.008 ; break;
 case ’I’: iResult =0.336 ; break;
 case ’L’: iResult =0.153 ; break;
 case ’K’: iResult =0.022 ; break;
 case ’M’: iResult =0.031 ; break;
 case ’F’: iResult =0.039 ; break;
 case ’P’: iResult =0.017 ; break;
 case ’S’: iResult =0.019 ; break;
 case ’T’: iResult =0.034 ; break;
 case ’W’: iResult =0.008 ; break;
 case ’Y’: iResult =0.021 ; break;
 case ’V’: iResult =0.167 ; break;
}
case ’L’:
switch (iSymb2) {
 case ’A’: iResult =0.048 ; break;
 case ’R’: iResult =0.024 ; break;
 case ’N’: iResult =0.013 ; break;
 case ’D’: iResult =0.014 ; break;
 case ’C’: iResult =0.009 ; break;
 case ’Q’: iResult =0.017 ; break;
 case ’E’: iResult =0.021 ; break;
 case ’G’: iResult =0.016 ; break;
 case ’H’: iResult =0.011 ; break;
 case ’I’: iResult =0.100 ; break;
 case ’L’: iResult =0.423 ; break;
 case ’K’: iResult =0.024 ; break;
 case ’M’: iResult =0.041 ; break;
 case ’F’: iResult =0.053 ; break;
 case ’P’: iResult =0.016 ; break;
 case ’S’: iResult =0.021 ; break;
 case ’T’: iResult =0.029 ; break;
 case ’W’: iResult =0.009 ; break;
 case ’Y’: iResult =0.024 ; break;
 case ’V’: iResult =0.087 ; break;
}
case ’K’:
switch (iSymb2) {
 case ’A’: iResult =0.055 ; break;
 case ’R’: iResult =0.101 ; break;
 case ’N’: iResult =0.044 ; break;
 case ’D’: iResult =0.047 ; break;
 case ’C’: iResult =0.003 ; break;
 case ’Q’: iResult =0.050 ; break;
 case ’E’: iResult =0.075 ; break;
 case ’G’: iResult =0.037 ; break;
 case ’H’: iResult =0.021 ; break;
 case ’I’: iResult =0.022 ; break;
 case ’L’: iResult =0.037 ; break;
 case ’K’: iResult =0.318 ; break;
 case ’M’: iResult =0.011 ; break;
 case ’F’: iResult =0.011 ; break;
 case ’P’: iResult =0.032 ; break;
 case ’S’: iResult =0.042 ; break;
 case ’T’: iResult =0.044 ; break;
 case ’W’: iResult =0.004 ; break;
 case ’Y’: iResult =0.017 ; break;
 case ’V’: iResult =0.030 ; break;
}
case ’M’:
switch (iSymb2) {
 case ’A’: iResult =0.058 ; break;
 case ’R’: iResult =0.027 ; break;
 case ’N’: iResult =0.021 ; break;
 case ’D’: iResult =0.020 ; break;
 case ’C’: iResult =0.010 ; break;
 case ’Q’: iResult =0.023 ; break;
 case ’E’: iResult =0.026 ; break;
 case ’G’: iResult =0.022 ; break;
 case ’H’: iResult =0.013 ; break;
 case ’I’: iResult =0.086 ; break;
 case ’L’: iResult =0.175 ; break;
 case ’K’: iResult =0.030 ; break;
 case ’M’: iResult =0.245 ; break;
 case ’F’: iResult =0.047 ; break;
 case ’P’: iResult =0.016 ; break;
 case ’S’: iResult =0.028 ; break;
 case ’T’: iResult =0.041 ; break;
 case ’W’: iResult =0.011 ; break;
 case ’Y’: iResult =0.026 ; break;
 case ’V’: iResult =0.076 ; break;
}
case ’F’:
switch (iSymb2) {
 case ’A’: iResult =0.038 ; break;
 case ’R’: iResult =0.016 ; break;
 case ’N’: iResult =0.014 ; break;
 case ’D’: iResult =0.014 ; break;
 case ’C’: iResult =0.010 ; break;
 case ’Q’: iResult =0.014 ; break;
 case ’E’: iResult =0.017 ; break;
 case ’G’: iResult =0.018 ; break;
 case ’H’: iResult =0.015 ; break;
 case ’I’: iResult =0.055 ; break;
 case ’L’: iResult =0.116 ; break;
 case ’K’: iResult =0.016 ; break;
 case ’M’: iResult =0.024 ; break;
 case ’F’: iResult =0.399 ; break;
 case ’P’: iResult =0.013 ; break;
 case ’S’: iResult =0.022 ; break;
 case ’T’: iResult =0.023 ; break;
 case ’W’: iResult =0.022 ; break;
 case ’Y’: iResult =0.095 ; break;
 case ’V’: iResult =0.057 ; break;
}
case ’P’:
switch (iSymb2) {
 case ’A’: iResult =0.063 ; break;
 case ’R’: iResult =0.028 ; break;
 case ’N’: iResult =0.024 ; break;
 case ’D’: iResult =0.037 ; break;
 case ’C’: iResult =0.005 ; break;
 case ’Q’: iResult =0.022 ; break;
 case ’E’: iResult =0.043 ; break;
 case ’G’: iResult =0.037 ; break;
 case ’H’: iResult =0.012 ; break;
 case ’I’: iResult =0.022 ; break;
 case ’L’: iResult =0.032 ; break;
 case ’K’: iResult =0.040 ; break;
 case ’M’: iResult =0.007 ; break;
 case ’F’: iResult =0.012 ; break;
 case ’P’: iResult =0.485 ; break;
 case ’S’: iResult =0.044 ; break;
 case ’T’: iResult =0.036 ; break;
 case ’W’: iResult =0.005 ; break;
 case ’Y’: iResult =0.014 ; break;
 case ’V’: iResult =0.032 ; break;
}
case ’S’:
switch (iSymb2) {
 case ’A’: iResult =0.096 ; break;
 case ’R’: iResult =0.036 ; break;
 case ’N’: iResult =0.049 ; break;
 case ’D’: iResult =0.055 ; break;
 case ’C’: iResult =0.011 ; break;
 case ’Q’: iResult =0.030 ; break;
 case ’E’: iResult =0.050 ; break;
 case ’G’: iResult =0.059 ; break;
 case ’H’: iResult =0.016 ; break;
 case ’I’: iResult =0.019 ; break;
 case ’L’: iResult =0.034 ; break;
 case ’K’: iResult =0.043 ; break;
 case ’M’: iResult =0.010 ; break;
 case ’F’: iResult =0.016 ; break;
 case ’P’: iResult =0.036 ; break;
 case ’S’: iResult =0.294 ; break;
 case ’T’: iResult =0.088 ; break;
 case ’W’: iResult =0.005 ; break;
 case ’Y’: iResult =0.017 ; break;
 case ’V’: iResult =0.036 ; break;
}
case ’T’:
switch (iSymb2) {
 case ’A’: iResult =0.074 ; break;
 case ’R’: iResult =0.033 ; break;
 case ’N’: iResult =0.037 ; break;
 case ’D’: iResult =0.040 ; break;
 case ’C’: iResult =0.010 ; break;
 case ’Q’: iResult =0.030 ; break;
 case ’E’: iResult =0.048 ; break;
 case ’G’: iResult =0.037 ; break;
 case ’H’: iResult =0.014 ; break;
 case ’I’: iResult =0.036 ; break;
 case ’L’: iResult =0.047 ; break;
 case ’K’: iResult =0.045 ; break;
 case ’M’: iResult =0.016 ; break;
 case ’F’: iResult =0.017 ; break;
 case ’P’: iResult =0.029 ; break;
 case ’S’: iResult =0.089 ; break;
 case ’T’: iResult =0.315 ; break;
 case ’W’: iResult =0.005 ; break;
 case ’Y’: iResult =0.016 ; break;
 case ’V’: iResult =0.062 ; break;
}
case ’W’:
switch (iSymb2) {
 case ’A’: iResult =0.039 ; break;
 case ’R’: iResult =0.024 ; break;
 case ’N’: iResult =0.016 ; break;
 case ’D’: iResult =0.015 ; break;
 case ’C’: iResult =0.004 ; break;
 case ’Q’: iResult =0.014 ; break;
 case ’E’: iResult =0.022 ; break;
 case ’G’: iResult =0.022 ; break;
 case ’H’: iResult =0.014 ; break;
 case ’I’: iResult =0.032 ; break;
 case ’L’: iResult =0.061 ; break;
 case ’K’: iResult =0.016 ; break;
 case ’M’: iResult =0.016 ; break;
 case ’F’: iResult =0.066 ; break;
 case ’P’: iResult =0.015 ; break;
 case ’S’: iResult =0.020 ; break;
 case ’T’: iResult =0.022 ; break;
 case ’W’: iResult =0.478 ; break;
 case ’Y’: iResult =0.071 ; break;
 case ’V’: iResult =0.034 ; break;
}
case ’Y’:
switch (iSymb2) {
 case ’A’: iResult =0.036 ; break;
 case ’R’: iResult =0.028 ; break;
 case ’N’: iResult =0.025 ; break;
 case ’D’: iResult =0.021 ; break;
 case ’C’: iResult =0.007 ; break;
 case ’Q’: iResult =0.018 ; break;
 case ’E’: iResult =0.026 ; break;
 case ’G’: iResult =0.020 ; break;
 case ’H’: iResult =0.028 ; break;
 case ’I’: iResult =0.034 ; break;
 case ’L’: iResult =0.061 ; break;
 case ’K’: iResult =0.027 ; break;
 case ’M’: iResult =0.015 ; break;
 case ’F’: iResult =0.108 ; break;
 case ’P’: iResult =0.017 ; break;
 case ’S’: iResult =0.026 ; break;
```

```
                case ’T’: iResult =0.025 ; break;
                case ’W’: iResult =0.027 ; break;
                case ’Y’: iResult =0.408 ; break;
                case ’V’: iResult =0.043 ; break;
               }
               case ’V’:
               switch (iSymb2) {
                case ’A’: iResult =0.074 ; break;
                case ’R’: iResult =0.021 ; break;
                case ’N’: iResult =0.014 ; break;
                case ’D’: iResult =0.016 ; break;
                case ’C’: iResult =0.014 ; break;
                case ’Q’: iResult =0.017 ; break;
                case ’E’: iResult =0.024 ; break;
                case ’G’: iResult =0.018 ; break;
                case ’H’: iResult =0.009 ; break;
                case ’I’: iResult =0.136 ; break;
                case ’L’: iResult =0.109 ; break;
                case ’K’: iResult =0.024 ; break;
                case ’M’: iResult =0.022 ; break;
                case ’F’: iResult =0.033 ; break;
                case ’P’: iResult =0.020 ; break;
                case ’S’: iResult =0.028 ; break;
                case ’T’: iResult =0.048 ; break;
                case ’W’: iResult =0.007 ; break;
                case ’Y’: iResult =0.022 ; break;
                case ’V’: iResult =0.346 ; break;
               }
      }
    ]]>
   </code>
  </probability>
 </emission>

<emission id="emitInsertion">
  <output idref="sequence1"/>
  <probability>
   <code type="statement">
    <identifier output="sequence1" value="iSymb1"/>
    <identifier type="result" value="iResult"/>
    <![CDATA[
      switch (iSymb1) {
                case ’A’: iResult =0.074 ; break;
                case ’R’: iResult =0.045 ; break;
                case ’N’: iResult =0.056 ; break;
                case ’D’: iResult =0.080 ; break;
                case ’C’: iResult =0.008 ; break;
                case ’Q’: iResult =0.041 ; break;
                case ’E’: iResult =0.076 ; break;
                case ’G’: iResult =0.099 ; break;
                case ’H’: iResult =0.022 ; break;
                case ’I’: iResult =0.036 ; break;
                case ’L’: iResult =0.063 ; break;
                case ’K’: iResult =0.073 ; break;
                case ’M’: iResult =0.017 ; break;
                case ’F’: iResult =0.031 ; break;
                case ’P’: iResult =0.065 ; break;
                case ’S’: iResult =0.068 ; break;
                case ’T’: iResult =0.057 ; break;
                case ’W’: iResult =0.012 ; break;
                case ’Y’: iResult =0.030 ; break;
                case ’V’: iResult =0.047 ; break;
               }
    ]]>
   </code>
  </probability>
 </emission>

<emission id="emitDeletion">
  <output idref="sequence2"/>
  <probability>
   <code type="statement">
    <identifier output="sequence2" value="iSymb2"/>
    <identifier type="result" value="iResult"/>
    <![CDATA[
      switch (iSymb2) {
                case ’A’: iResult =0.057 ; break;
                case ’R’: iResult =0.058 ; break;
                case ’N’: iResult =0.084 ; break;
                case ’D’: iResult =0.091 ; break;
                case ’C’: iResult =0.035 ; break;
                case ’Q’: iResult =0.072 ; break;
                case ’E’: iResult =0.077 ; break;
                case ’G’: iResult =0.086 ; break;
                case ’H’: iResult =0.063 ; break;
                case ’I’: iResult =0.039 ; break;
                case ’L’: iResult =0.045 ; break;
                case ’K’: iResult =0.082 ; break;
                case ’M’: iResult =0.053 ; break;
                case ’F’: iResult =0.048 ; break;
                case ’P’: iResult =0.092 ; break;
                case ’S’: iResult =0.079 ; break;
                case ’T’: iResult =0.066 ; break;
                case ’W’: iResult =0.054 ; break;
                case ’Y’: iResult =0.054 ; break;
                case ’V’: iResult =0.042 ; break;
               }
    ]]>
   </code>
  </probability>
 </emission>

<transitions>
 <transition from="start" to="Match" probability="start_match" />
 <transition from="start" to="Insertion" probability="start_insert" />
 <transition from="start" to="Deletion" probability="start_delete" />

 <transition from="Deletion" to="end" probability="delete_end" />
 <transition from="Deletion" to="Match" probability="delete_match" />
 <transition from="Deletion" to="Deletion" probability="delete_delete" />

 <transition from="Match" to="end" probability="match_end"/>
 <transition from="Match" to="Insertion" probability="match_insert"/>
 <transition from="Match" to="Deletion" probability="match_delete"/>
 <transition from="Match" to="Match" probability="match_match"/>

 <transition from="Insertion" to="end" probability="insert_end"/>
 <transition from="Insertion" to="Match" probability="insert_match"/>
 <transition from="Insertion" to="Insertion" probability="insert_insert"/>

</transitions>

<probability id="start_match"><code type="expression"> 1033.0/1034
</code></probability> <probability id="start_insert"><code
type="expression"> (1.0/2)/1034 </code></probability> <probability
id="start_delete"><code type="expression"> (1.0/2)/1034
</code></probability>

<probability id="match_end"><code type="expression"> 1033.0/212016
</code></probability> <probability id="match_insert"><code
type="expression"> (6817.0/2)/212016 </code></probability>
<probability id="match_delete"><code type="expression">
(6817.0/2)/212016 </code></probability> <probability
id="match_match"><code type="expression"> 204166.0/212016
</code></probability>

<probability id="delete_end"><code type="expression"> 1.0/27413
</code></probability> <probability id="delete_match"><code
type="expression"> 6817.0/27413 </code></probability> <probability
id="delete_delete"><code type="expression"> 20595.0/27413
</code></probability>

<probability id="insert_end"><code type="expression"> 1.0/27413
</code></probability> <probability id="insert_match"><code
type="expression"> 6817.0/27413 </code></probability> <probability
id="insert_insert"><code type="expression"> 20595.0/27413
</code></probability>

</hmm>

<!-- Code generation -->

<forward realtype="bfloat" outputTable="yes" name="Forward"
id="fw">

 <!-- Specify HMM to make code for -->
 <hmm idref="Aligner"/>

</forward>

<backward realtype="bfloat" outputTable="yes" baumWelch="yes"
name="Backward" id="bw">
```

```
 <!-- Specify HMM to make code for -->
 <hmm idref="Aligner"/>

</backward>

<sample name="Sample" id="smp">

 <hmm idref="Aligner"/>

</sample>

<viterbi realtype="bfloat" name="Viterbi" id="vit">

 <hmm idref="Aligner"/>

</viterbi>

<codeGeneration file="aligner.cc" header="aligner.h"
language="C++">

 <forward idref="fw"/>
 <backward idref="bw"/>
<!--  <sample idref="smp"/>  -->
 <viterbi idref="vit"/>

</codeGeneration>

<codeGeneration file="main.cc" language="C++">

<!--  This is just to package this example in one file. It is
actually rather inconvenient to edit main.cc in here.. -->

<code> <![CDATA[

#include <iostream> #include <fstream> #include <string> #include
<vector>

using namespace std;

#include "aligner.cc"

int main() {
//  char seq1[]="LTREKLIELAILVGTDYNP";
// char seq2[]="LTREKLIELAILV";
 int count =0;
 double** rates;
 DPTableAligner* pT1, *pT2, *pT3;

 BaumWelchAligner *pBW;  int id;
 while (!cin.eof())
 {
   std::string sseq1, sseq2, dummy;
   cin >> sseq1;
   cin >> sseq2;

   if (cin.eof()) break;

   /* Find length of sequences */
   int iLen1 = sseq1.size();
   int iLen2 = sseq2.size();
   count = count +1;
   char * seq1 = const_cast<char*>(sseq1.c_str());
   char * seq2 = const_cast<char*>(sseq2.c_str());

   /* Print sequence, and forward and backward likelihoods. */
   /* The backward algorithm also computes Baum-Welch posterior counts */

   cout << "count " << count <<endl;
       cout << "Sequence1: " << seq1 << endl;
   cout << "Sequence2: " << seq2 << endl;
   cout << "Forward:" << endl <<
    Forward(&pT1,seq1,seq2,iLen1,iLen2) << endl;
   cout << "Backward:" << endl <<
    Backward(&pBW,pT1,&pT2,seq1,seq2,iLen1,iLen2) << endl;

   /* Print posterior counts for all transitions */
   for (int i=0; i<pBW->transNum00; i++) {
   id = pBW->transId00[i];
   //cout << pT1->transitionId[id] << " " <<
   // pT1->transitionFrom[id] << "->" << pT1->transitionTo[id]
   // << ":" << pBW->transCount00[i] << endl;
   }

   /* Print Viterbi likelihood, and Viterbi path */

   cout << "Viterbi:" << endl;
   cout << Viterbi_recurse(&pT3,seq1,seq2,iLen1,iLen2) << endl;
   cout << Viterbi_trace(pT3,seq1,seq2,iLen1,iLen2) << endl;

     }

return 0; }

]]>

</code>

</codeGeneration>

</hml>
```