
EDY: an Algorithm for Discovering Complex Events in Symbolic Sequences

Ugo Galassi

GALASSI@MFN.UNIPMN.IT

Dipartimento di Informatica, Università Amedeo Avogadro, Via Bellini 25G, Alessandria, Italy

Attilio Giordana

ATTILIO@MFN.UNIPMN.IT

Dipartimento di Informatica, Università medeo Avogadro, Via Bellini 25G, Alessandria, Italy

Lorenza Saitta

SAITTA@MFN.UNIPMN.IT

Dipartimento di Informatica, Università medeo Avogadro, Via Bellini 25G, Alessandria, Italy

Abstract

This paper presents an algorithm for inferring a Hierarchical Profile Hidden Markov Model (HPHMM) from a set of sequences. The HPHMMs are a sub-class of the Hierarchical Hidden Markov Models that nicely extend the profile HMMs, developed in bioinformatics. They are well suited to problems of process/user profiling, in network monitoring or in adaptive user interfaces, as well as to problems of molecular biology. The learning algorithm is unsupervised, and follows a mixed bottom-up/top-down strategy, in which elementary facts in the sequences (motifs) are progressively grouped, thus building up the abstraction hierarchy of a HPHMM, layer after layer. An extensive evaluation on a suite of artificial datasets is reported, where the challenge for the learning algorithm is to reconstruct the model that generated the data.

1. Introduction

Many tasks in today applications require that an accurate *profile* of an agent (be it a process or a human user) be built up. Typical examples are Intrusion Detection Systems in computer networks (Stolfo et al., 2000), Fraud Detection Systems in telephony (Fawcett & Provost, 1997), or customer tracing in e-commerce (MacDonald, 2001). An agent's profile is an abstract characterization of its activity relevant to the task at

hand. Current methods for building profiles are still quite primitive and, in many cases, reduce to the measure of the frequency of selected classes of actions executed by an agent in a temporal window.

This paper presents a new methodology, where more complex profiles, accounting for typical sequences of actions and for typical state transitions, are learned by induction from *logs* of the agent's behavior. The methodology exploits a Hierarchical Hidden Markov Model (HHMM) (Fine et al., 1998), and is an extension of previous work (Botta et al., 2004). The basic assumption underlying the approach is that the behavior of an agent repeatedly shows (short) sequences of actions, typical of the task it executes, interleaved with phases that do not contain recognizable patterns. By analogy with DNA analysis in molecular biology, we will call *motifs* such characteristic sequences of actions, and we make heavy use of techniques developed in bioinformatics (Durbin et al., 1998). The intervals between motifs are called *gaps*.

The choice of the Hidden Markov Model (HMM) framework is justified by the success reported in several application domains, ranging from speech recognition (Rabiner, 1989; Rabiner & Juang, 1993) to DNA analysis (Durbin et al., 1998), where the problems are similar to the ones considered here.

With respect to previous work (Botta et al., 2004), the present paper has three major novelties: first of all, an optimization of the system allowed to scale up, of at least one order of magnitude, the length of the analyzed sequences. Then, the refinement procedure of the initial model has been greatly improved, with a consequent improvement of the quality of the results. Finally, a systematic test methodology, allowing a quantitative analysis of the quality of the results has

Appearing in *Proceedings of the ICML Workshop on Learning in Structured Output Spaces*, Pittsburgh, PA, 2006. Copyright 2006 by the author(s)/owner(s).

been introduced.

2. Hierarchical Profile HMM

As previously mentioned, the behavior of an agent can be described as a partially ordered graph of characteristic motifs, interleaved with gaps. This kind of structure can be modeled very well using a Hierarchical Hidden Markov Model (HHMM) (see (Fine et al., 1998), for details). A HHMM is a multilevel extension of the basic HMM (Rabiner, 1989) where emission at higher level states are sequences computed by sub-models at a lower level in the hierarchy. However, the full power of the original HHMM is not necessary in the present case. Then, a restricted HHMM is proposed in order to reduce the computational complexity of the algorithm. The restricted model is called Hierarchical Profile HMM (HPHMM). A HPHMM is restricted to have only two levels: a level-0, which consists of a collection of HMMs describing single motifs and gaps; a level-1 automaton, which describes a partial order on motifs and gaps. In principle, an HPHMM could have an arbitrarily large number of levels in the hierarchy. Nevertheless, up to now no need emerged of going beyond two levels. Then, for the sake of simplicity, the description in the following will be restricted to this simpler case.

A further restriction is that the global model be left-to-right. This means that no back-loops are allowed, and self-loops are allowed only in the HMM at level-0. A last restriction, which is not fundamental but allows several algorithms developed in BioInformatics to be immediately exploited, is that motif models are *Profile Hidden Markov Model (PHMM)* (Durbin et al., 1998).

2.1. Modeling Motifs with Profile HMM

A PHMM assumes that a motif has a canonical instantiation form, which can be affected by insertion and deletion errors. As described in Figure 1, a PHMM is basically a forward graph with three categories of states:

- *match* states, where the emission corresponds to the unique symbol expected in the canonical instantiation;
- *insertion* states, where a number of symbols can be inserted, due to random noise;
- *deletion* states, where no emission occurs where it was supposed to.

In addition, other two non emitting states are required; the *start* state, and the *end* state. Recursion is con-

sidered only in the form of self-loops associated to *insertion* states. It has been experimentally shown that PHMM is more accurate than string matching to detect motifs. Moreover, the algorithmic complexity inherent to PHMM is linear for both Viterbi and Forward-Backward algorithms.

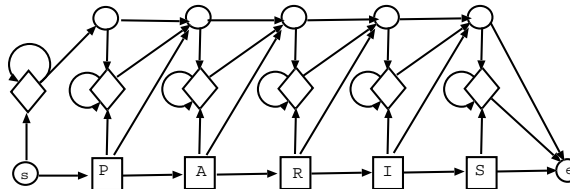


Figure 1. Example of Profile Hidden Markov Model. Circles denote states with no observable emission: *start* state, *end* state and *deletion* states; squares denote *match* states; diamonds denote *insertion* states.

2.2. Modeling Gaps

The presence of possibly long gaps between motifs is a problem that needs to be explicitly addressed. A simple way of dealing with short gaps, in the framework of PHMM, is to add an *insertion* state with a self-loop in between two motifs. However, this simple approach does not work when gaps are long. In fact, a long gap may hide a complex process that needs to be completed before entering the phase generating the next motif. Then, the gap lengths can follow a distribution different from the exponential one, which is the unique distribution associated to a self-loop, and may be not at all realistic. An example of a more plausible distribution is reported in Figure 2-(a).

It is easy to verify that any length distribution can be encoded using a forward model like the one described in Figure 2-(b), given a sufficient number of states.

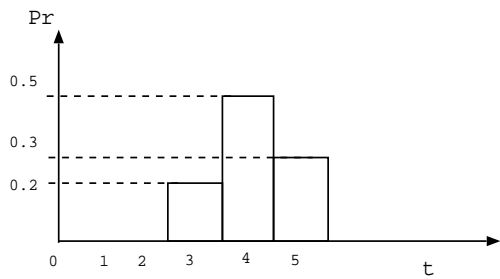
2.3. The High Level Architecture

As mentioned previously, level 1 in a HPHMM describes how motifs and gaps may be interleaved. In order to tame complexity, this level is restricted to be a forward graph with no self-loops. Two kinds of states are defined:

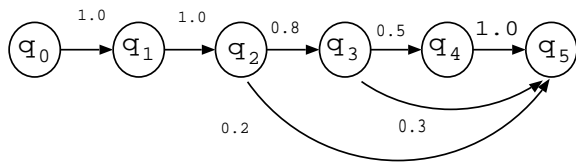
- *emitting* states, whose emission is defined by one PHMM at level 0, and
- *delay* states, associated to a gap model at level 0.

An example of HPHMM is given in Figure 3.

Under the given restrictions, two important properties (not proven here) hold for HPHMMs, whereas they



(a)



(b)

Figure 2. Hidden Markov Model for iteration/gaps. (a) Example of a probability distribution over the durations of a gap due to a physical process; (b) Left-to-right unfolded model that correctly generates distribution (a).

do not hold for unrestricted HHMMs.

Property 1: Any HPHMM can be compiled into a single level Left-to-Right HMM (LRHMM) without losing the structure enforced by the hierarchy.

In other words, flattening a HPHMM into a single level LRHMM, denoted as $LRHMM'$, can be simply done by replacing the states defined at level 1 by the level-0 PHMMs.

Property 2: The complexity of the Forward-

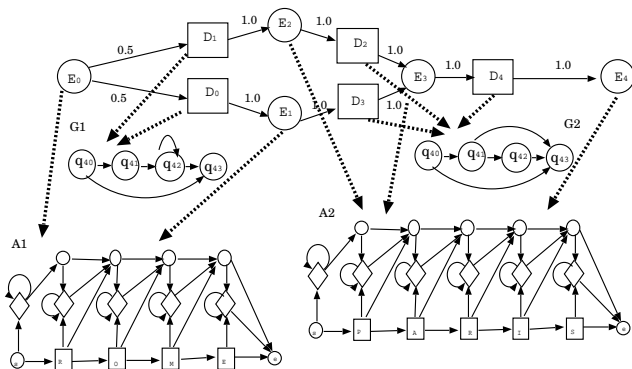


Figure 3. Example of Hierarchical Hidden Markov Model. Circles denotes states with observable emission, whereas rectangles denote states associated to *gaps*.

Backward and Viterbi algorithms for HPHMMs is $O(B, |S|, n)$, where n is the length of the observed sequence, B is the average branching factor, and $|S|$ is the number of states of the corresponding $LRHMM'$.

As the EM algorithm is based on the Forward-Backward algorithm, every EM cycle has the same complexity. This property is very important for dealing with long sequences and complex HPHMM.

3. Learning Algorithm

The HPHMM λ of a complex event is constructed incrementally going through a learning cycle in which a model is progressively extended and refined, by repeatedly incorporating new motifs and gaps. The cycle may initiate with a *empty model* or with a model supplied by an expert of the domain, and terminates when there is no more evidence of new motifs to incorporate. The rationale behind this architecture is that regularities due to the presence of motifs may be difficult (or impossible) to distinguish from randomness when considered in isolation, but may become evident in the context established by a partial model. Therefore, the learning strategy tends to discover first the motifs, which are detectable even in absence of the model or in presence of a small subset of it.

Before describing in details the EDY system, we need to describe *sequence tagging* and *sequence abstraction*, which are the basic procedures of the learning strategy.

Sequence tagging. Let λ_t denote the current HPHMM, learned by EDY after t iterations on a learning set \mathcal{LS} . Sequence tagging is accomplished by using the Viterbi algorithm to find, in each sequence $s \in \mathcal{LS}$, the most likely instantiations of λ_t . From these instantiations it is easy to determine the regions (and hence the gaps in between) where most likely the motifs and gaps described by λ_t occur. Such regions are tagged with the *id* of the corresponding motif and gap models. In the following $\mathcal{LS}(\lambda_t)$ will denote the set of learning sequences tagged using λ_t .

Sequence abstraction. After sequence tagging has been done, an abstract description $s'(\lambda_t)$ can be generated, for each sequence $s \in \mathcal{LS}$, by replacing the tagged regions with the corresponding motif or tag *id*. In the following, $\mathcal{LS}'(\lambda_t)$ will denote the set of all sequences abstracted using λ_t .

The main algorithm of EDY iteratively performs a cycle in which two operators can be applied: the *Extend* and the *Refine* operators. Let EXTEND denote the control variable that activates the operator *Extend*, whereas HALT denote the one that controls

the overall cycle execution. The abstract scheme of the learning algorithm is the following one:

```

EDY( $\lambda$ )
STABLE = False, HALT = False
while  $\neg$  HALT do
  while  $\neg$  STABLE do
     $\lambda_{new} = Refine(\lambda)$ 
    if  $\mathcal{LS}(\lambda_{new}) \simeq \mathcal{LS}(\lambda)$ 
      then STABLE = True
    endif
     $\lambda = \lambda_{new}$ 
  endwhile
  Apply Extend( $\lambda$ )
  if Extend( $\lambda$ ) fails
    then HALT = True
    else  $\lambda_{new} = Extend(\lambda)$ 
     $\lambda = \lambda_{new}$ 
  endif
endwhile
    
```

In the next subsection the functioning of the algorithm will be briefly illustrated.

3.1. Model Extension

In the following, symbol λ will denote a HPHMM, whereas symbol μ will denote sub-models of motifs and gaps. Given the current model $\lambda = \lambda_t$, the algorithm applies the *Refine* operator until (approximately) no more difference exists, in the abstracted sequences in \mathcal{LS} , between two consecutive cycles. When this happens, EDY tries to extend the current model, by transforming some gaps into motifs. However, a candidate motif is not substituted to the gap, but both are left in parallel (see Figure 4) for the *Refine* operator to decide. Notice that at most one candidate motif is output in an extension step, with the only exception in the first cycle, when no model exists. In this case, the algorithm may construct a more complex initial model containing two or more motifs. In order to find motifs, EDY searches for regularities, from which it builds up a PHMM. In doing this, existing techniques, developed in Molecular Biology for inducing PHMMs from symbolic sequences, are used. In building motifs, two basic notions are exploited: *edit distance* and *alignment* between strings. Informally, the edit distance is measured as the number of corrections necessary to make two strings identical. The alignment of two strings s_1, s_2 is a pair of strings s'_1, s'_2 obtained by s_1, s_2 by inserting a proper number of spaces such that identical symbols are put into correspondence to a maximum extent. An alignment is said *local* when it is restricted to a subset of s_1, s_2 , and is said *multiple* when it involves a number of strings larger than 2.

Therefore, motifs are discovered and modeled according to the following steps:

1. For every pair of sequences (s_1, s_2) in \mathcal{LS} , EDY

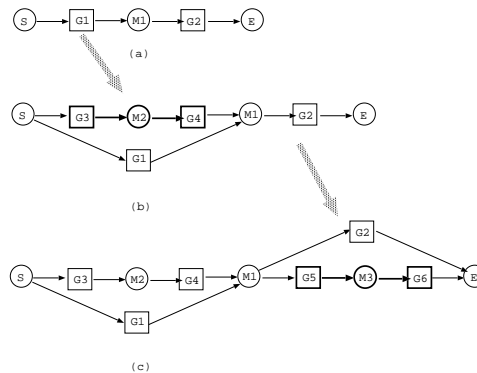


Figure 4. Example of transformations of the level 1 automaton due to the *extension operator*. (a) Previous model. (b) Gap G1 is analyzed discovering the presence of a motif M2. As M2 is not present in all instances of Gap G1, G1 is still kept in the model. (c) An analogous transformation occurs when analyzing Gap G2.

finds all local, statistically relevant *alignments* between them, and collects the aligned substrings into a set \mathbf{A} . These will be the candidate motifs. Motif detection is accomplished by classical string matching algorithms based on Levenstein's edit distance (Levenstein, 1966) and string similarity (Smith & Waterman, 1981; Waterman, 1983; Gussfield, 1997).

2. Apply a *clustering* algorithm to the subsequences in \mathbf{A} , using an incremental version of the classical k-Means algorithm, with Levenstein's distance (Levenstein, 1966) as distance measure. Discard clusters with too few elements.
3. For every retained cluster C_i construct a model μ_i of the subsequences contained in it, using the algorithm described in (Durbin et al., 1998). The algorithm searches first for multiple alignments among all subsequences in C_i . Then, it converts the aligned strings into an PHMM μ_i . Finally, μ_i 's parameters are estimated from the sequences in C_i .

3.2. Model Refinement

Model refinement is the core of the learning process and concerns several aspects of model construction, all motivated by the desire of better exploiting context information, embedded in the upper level of the hierarchy, which was partially ignored during learning the lower level. By summarizing, in every refinement step six different actions are tried, in the following order: Boundary refinement, Model diversification, Model unification, Parameter refinement, Gap model

refinement, and Hierarchy revision. The starting point

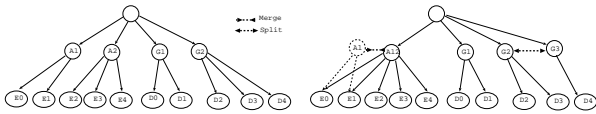


Figure 5. Example of cluster hierarchy obtained for the HPHMM exemplified in Figure 3.

is the tagged dataset $\mathcal{LS}(\lambda_t)$ constructed before calling the refinement operator. All sequence segments corresponding to motif and gap instances, which have been detected by the Viterbi algorithm are collected into, a two levels hierarchical clustering. Figure 5-a provides an example of the clusters for the HPHMM in Figure 3. The clusters in the leaves of the tree corresponds to emitting and delay states in the level 1 automaton. Each leaf contains all the subsequences which have been emitted in the corresponding state (motifs or gaps instances). However, emissions in different states can be generated by a same level 0 model. Then, the clusters at the second level group together the leaves whose elements are generated by a same sub-model μ . The root node of the tree is a dummy node, representing the whole set of segments. During the refinement process second level nodes can be split or merged (see Figure 5-b) in analogy to (Fisher, 1987). Given a distance measure between instances (the edit distance in the specific case), two clusters of motif/gap instances are merged if the distance between their centers is not greater than their average intra-cluster distance. Alternatively, a cluster, whose children have an intra-cluster distance much smaller than the inter-cluster distance, may be split. More details are provided in the following.

Boundary refinement - This operator is meant to correct possible segmentation errors performed during the initial learning phase. Before trying to refine a motif model, the algorithm for searching local alignments is run on the new set of instances, but allowing the alignments to possibly extend into the adjoining gap regions for one or two positions. Instances of the motif can thus be extended (or reduced) if the original segmentation is found inaccurate.

Model diversification - If μ is a PHMM associated to two different emission states E_j, E_k , and the two associated instance clusters C_j and C_k significantly differ, then μ is split into μ_j and μ_k , which are trained independently on C_j and C_k , respectively.

Model unification - When two sub-models μ_j and μ_k have children that cannot be distinguished among themselves according to the distance criterion, the models can be merge into a single one, μ , whose parameters can be estimated from the cluster obtained

as union of μ_j and μ_k 's children. The procedure for merging gap models is analogous, but based on a different criterion. More precisely, considering two clusters C_j and C_k of gap instances, the histograms h_j and h_k of the corresponding gap lengths are constructed. Histograms are compared among each other, and "similar" ones are merged. This operator is only activated optionally, as it may slow down convergence to a stable hierarchy.

Parameter refinement - As the instances of a model may be currently different from those used to initially learn it, the model's parameters are re-estimated from the new set of instances.

Gap model refinement - This operator works in a way similar to the preceding one, except that the parameters to be estimated are those appearing in the distribution of the gap lengths, in order to make it closer to the observed one.

Hierarchy Revision. The algorithm for the construction/reconstruction of the high layer (level 1) of the hierarchical HPHMM is very similar to the one that constructs the PHMMs at level 0. The difference is that it works on the abstracted sequences belonging to \mathcal{LS}' , and that the level 1 automaton is no longer a PHMM, but a left-to-right HMM without cycles. The abstract sequences in \mathcal{LS}' are aligned constructing a multiple alignment, which is used to generate a left-to-right automaton.

As the above algorithm is computationally inexpensive, it is always repeated at every refinement step, in order to propagate at the level 1, changes in the structure at level 0.

4. Evaluation on Artificial Traces

The purpose of the evaluation presented here is to assess the ability of EDY algorithm to discover a model hidden in a set of sequences (called the *Target model* in the following), assuming that the model could be reasonably approximated by means of a HPHMM. This is done using two suites of artificial data generated using handcrafted models. The first suite is simpler and emulates a problem of text analysis. An ordered sequence of selected words from natural language has been interleaved with irrelevant text randomly generated. Furthermore, noise has been added by randomly changing some letters. The task is to discover the original sequence, filtering the noise and eliminating the irrelevant text. The second suite is more complex. Several Hierarchical Hidden Markov Models, of different complexity, have been constructed and then used to generate a set of learning sequences. Every model con-

tains both motif and gap models, including an initial and a final gap model, which bracket the relevant part of the sequence with an initial and final string of random length. The complexity of the models varies from one hundred to four hundred states.

The performance on a sequence s of a HPHMM λ learned by the algorithm is measured by computing the edit distance between the tagging on \mathcal{LS} , obtained from the target model λ_T , and the tagging generated from the model λ_D discovered by EDY. Let $\delta_E(\mathcal{LS}(\lambda_T), \mathcal{LS}(\lambda_D))$ be the edit distance between the two taggings on \mathcal{LS} . Let moreover, $L(\mathcal{LS}(\lambda_T))$ be the total length of the motif instances tagged in $\mathcal{LS}(\lambda_T)$. The performance of λ_D will be evaluated as

$$e(\lambda_D) = \delta_E(\mathcal{LS}(\lambda_T), \mathcal{LS}(\lambda_D)) / L(\mathcal{LS}(\lambda_T)) \quad (1)$$

4.1. Task 1: Detecting a sequence of words

A sequence of problems has been generated varying the number of words ($5 \leq w \leq 8$), the word length ($5 \leq L \leq 8$) and the noise level ($N \in \{0\%, 5\%, 10\%, 15\%\}$). For every triple $\langle w, L, N \rangle$, 10 different datasets have been generated for a total of 640 learning problems.

Table 1. Performances obtained with *town names* dataset. The sequence length ranges from 60 to 140 characters.

		Previous System				EDY			
		Noise Level				Noise Level			
w	L	0%	5%	10%	15%	0%	5%	10%	15%
5	5	0.00	0.02	0.02	0.02	0.00	0.00	0.00	0.00
5	6	0.06	0.12	0.12	0.09	0.00	0.00	0.00	0.00
5	7	0.00	0.02	0.03	0.05	0.00	0.00	0.00	0.00
5	8	0.00	0.03	0.02	0.04	0.00	0.00	0.00	0.00
6	5	0.06	0.01	0.04	0.04	0.00	0.00	0.03	0.02
6	6	0.02	0.10	0.06	0.19	0.00	0.00	0.00	0.02
6	7	0.00	0.03	0.02	0.05	0.00	0.00	0.00	0.00
6	8	0.00	0.04	0.05	0.05	0.00	0.00	0.00	0.00
7	5	0.02	0.05	0.11	0.07	0.00	0.00	0.01	0.01
7	6	0.01	0.10	0.05	0.14	0.00	0.01	0.00	0.02
7	7	0.00	0.06	0.02	0.05	0.00	0.00	0.00	0.00
7	8	0.01	0.06	0.09	0.09	0.00	0.00	0.00	0.00
8	5	0.00	0.00	0.01	0.10	0.00	0.00	0.01	0.00
8	6	0.03	0.08	0.10	0.14	0.00	0.01	0.03	0.00
8	7	0.00	0.01	0.01	0.08	0.00	0.00	0.00	0.00
8	8	0.01	0.03	0.08	0.09	0.00	0.00	0.00	0.00

The most important results are summarized in Table 1. The table entries report the performance measure (1). The left part of the table reports the performance obtained by a previous algorithm described in (Botta et al., 2004). The right part reports the performance obtained by EDY.

It appears that in most cases the final model extracted from the data is equivalent to the target. Moreover, the method seems to be little sensitive to the sequence length, while the error rate roughly increases proportionally to the noise in the original model. In fact, almost always, the maximum likelihood sequence generated by the learned model corresponds to nominal sequence without noise. In other words, the algorithm

succeeded in cleaning the data from the noise.

Table 2. Structural complexity of the models learned in Task 1. $|Q|_F$ denotes the number of states in the flattened LRHMM. $|Q|_T$ denotes the number of states in the automata at level 1, and $|Q|_M$ the number of states globally included in motif models.

w	L	$ Q _F$	$ Q _T$	$ Q _M$
5	5	165.88	13.68	100.80
5	6	228.90	13.92	119.03
5	7	252.02	14.07	137.18
5	8	227.50	14.22	150.90
6	5	247.53	15.98	119.62
6	6	205.90	16.18	135.23
6	7	298.40	16.40	166.43
6	8	247.95	15.80	171.15
7	5	190.05	17.43	131.33
7	6	234.50	17.80	164.18
7	7	273.53	17.60	181.27
7	8	345.05	18.32	214.80
8	5	172.60	18.18	137.55
8	6	324.15	19.85	183.45
8	7	359.87	20.70	217.20
8	8	285.68	19.35	227.10

4.2. Task 2: Discovering a target HMM

Two groups of target HPHMMs have been constructed and used to generate quite huge sequence datasets. The HPHMMs in the first group contains 7 states at the level 1, where three states are emitting states associated to motif models, and the other four are delay states associated to gap models (one initial gap and one final gap plus two intermediate gaps). The HPHMMs in the second group have a similar, but more complex, structure. They encode a sequence of six motifs separated by 7 gaps.

The procedure for generating the set of models in each group is the following. Firstly, for every group three model templates have been handcrafted, containing motifs of 5, 8, and 11 match states, respectively. Templates are parametric with respect to the alphabets and the probability distributions. Then, starting from the templates, the set of models has been generated by varying the size of the observation alphabet, the probability distribution on the emission, the probability of insertion and deletion error, and the probability distribution on the gap length. More specifically, 5 classes of alphabets have been considered, of cardinality: 7, 10, 14, 19, and 25, and four classes of probability distributions: N0, N1, N2, and N3. Each distribution is a normal distribution, and the four classes are charac-

terized by an increasing amplitude of the standard deviation. The effect of the standard deviation increase on the probability distribution on the sequences, (generated by a HPHMM) is not obvious to evaluate. We will measure it by considering the average edit distance between the set of motifs contained in the maximum likelihood sequence generated by a model λ and the motifs obtained from a sample of hundred sequences generated from λ . According to this criterion, the following values (referred the length of the maximum likelihood motif sequence), are obtained:

Class:	N0	N1	N2	N3
δ_E :	0	0.11	0.19	0.28

In all cases, gaps contain random subsequences.

For every model a set of 5 different learning sets, each one containing 100 sequences, has been generated. The performances obtained by EDY are reported in Figures 6 and 7. It appear that $e(\lambda_D)$ (1) increases when the alphabet cardinality and the motif length decrease, as well as when the standard deviation of the target model increase, as it is reasonable to expect. In fact, when the alphabet is small it is more difficult to distinguish real motifs from apparent regularities due to randomness. For the same reason, short motifs are more difficult to detect. Then, the performance degradation is due, in general, to the failure of the algorithm, which searches for new motifs without finding the correct ones. The decrease in the similarity between the target model and the discovered model, when the probability distributions have long tails, is also in agreement with what one expects.

5. Conclusions

A method for automatically synthesizing complex profiles from traces has been proposed. The method builds on HHMM and Profile HMM. The main contribution of the paper consists in organizing and generalizing into a unique framework different methods developed in the past. The outcome is the new architecture of the HPHMM, which is powerful enough to model many real world problems, and has a very low computational complexity.

In several tests on artificial datasets, the method succeeded in reconstructing non trivial two-level HPHMMs.

References

Botta, M., Galassi, U., & A.Giordana (2004). Learning complex and sparse events in long sequences. *Proceedings of the European Conference on Artificial Intelligence, ECAI-04*. Valencia, Spain.

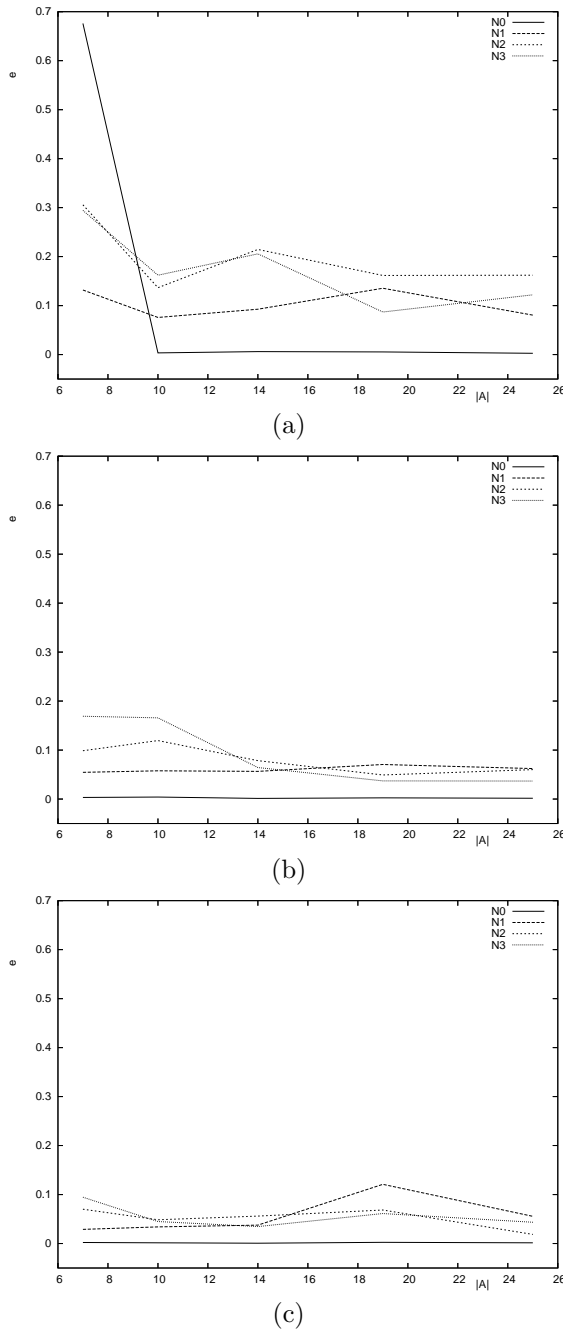
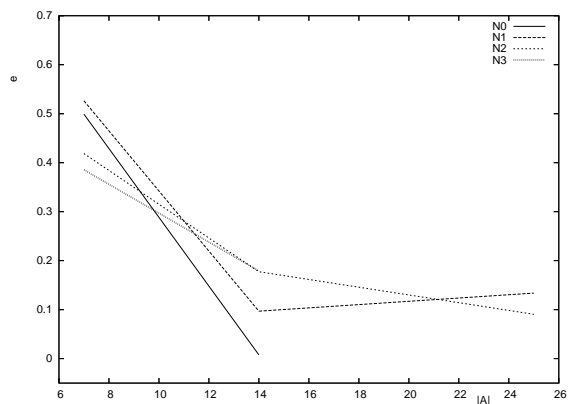
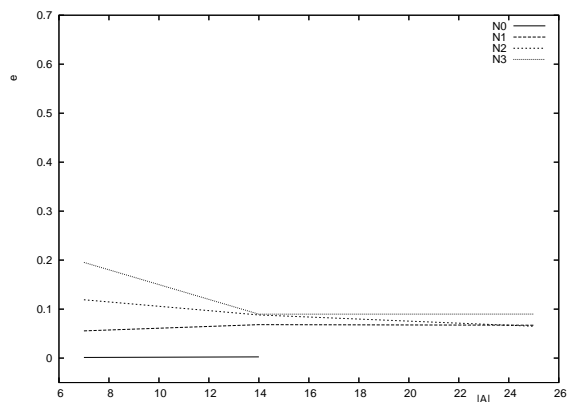


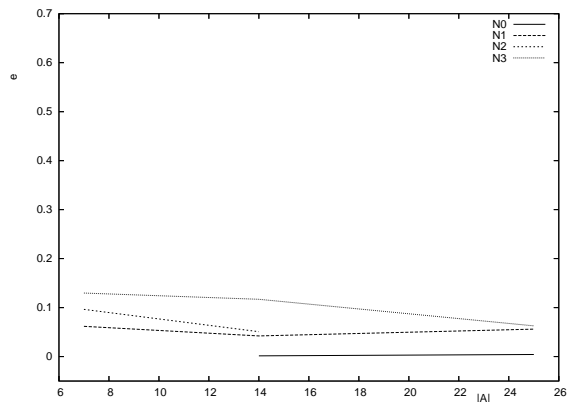
Figure 6. EDY's evaluation models in Group 1 (3 motifs). In the diagrams, x-axis represents the alphabet cardinality $|A|$ and the y-axis reports the edit distance between the tagging obtained from the model discovered by EDY and the one obtained from the model that has generated the learning sequences. Diagram (a), (b), and (c) correspond to models with motif of 5, 8, and 11 match states, respectively.



(a)



(b)



(c)

Figure 7. EDY's evaluation on models in Group 2 (6 motifs). The diagrams (a), (b), and (c) are as in Figure 6.

Durbin, R., Eddy, S., Krogh, A., & Mitchison, G. (1998). *Biological sequence analysis*. Cambridge University Press.

Fawcett, T., & Provost, F. (1997). Adaptive fraud detection. *Data Mining and Knowledge Discovery Journal*, 1, 291–316.

Fine, S., Singer, Y., & Tishby, N. (1998). The hierarchical hidden markov model: Analysis and applications. *Machine Learning*, 32, 41–62.

Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2, 139–172.

Gussfield, D. (1997). *Algorithms on strings, trees, and sequences*. Cambridge University Press.

Levenstein, V. (1966). Binary codes capable of correcting insertions and reversals. *Soviet. Phys. Dokl.*, 10, 707–717.

MacDonald, R. D. (2001). *Web-based user profiling using artificial neural networks*. Honours Thesis, Acadia University.

Rabiner, L. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of IEEE*, 77(2), 257–286.

Rabiner, L., & Juang, B. (1993). *Fundamentals of speech recognition*. Englewood Cliffs, NY: Prentice Hall.

Smith, T., & Waterman, M. (1981). Identification of common molecular subsequences. *Journal of Molecular Biology*, 147, 195–292.

Stolfo, S., Fan, W., Lee, W., Prodromidis, A., & Chan, P. (2000). Cost-based modeling for fraud and intrusion detection: Results from the jam project. *Proceedings of the 2000 DARPA Information Survivability Conference and Exposition (DISCEX '00)*.

Waterman, M. (1983). Sequence alignments in the neighborhood of the optimum with general application to dynamic programming. *Proceedings of National Academy of Science*, 80, 3123–3147.