

A General Probabilistic Framework for Mining Labeled Ordered Trees*

Nobuhisa Ueda[†]

Kiyoko F. Aoki[†]

Hiroshi Mamitsuka[†]

Abstract

We propose a new probabilistic model for mining labeled ordered trees. A noteworthy feature of the proposed model is to consider ordered siblings by modeling the dependencies of a node in a tree on the elder sibling as well as the parent. This model is reasonably extended from a variety of existing probabilistic models for strings and trees. We further propose a new learning/mining method to estimate the parameters of this model, based on an EM algorithm. This is also an extension of those for various simpler probabilistic models, such as hidden Markov models and hidden tree Markov models. We evaluated the effectiveness of our proposed method using both synthetic and real-world data sets, comparing the results with those of several simpler probabilistic models. Experimental results have shown that our proposed method outperforms the other methods compared, being statistically significant in all cases tested. This result tells us that the proposed methodology is highly effective for mining labeled ordered trees, which have recently emerged as one of the typical data structures in numerous data mining domains, including the web, text mining and bioinformatics. **Keywords:** Labeled ordered trees, probabilistic models, multiply connected Bayesian network, bioinformatics, XML mining, stochastic models

1 Introduction

Mining frequent patterns is one of the most important issues in the field of data mining. Patterns to be found in conventional data mining are limited to relatively simple patterns, such as associations and sequences. In these recent years, structured patterns such as trees and graphs have appeared in several major data mining fields, such as text mining, web mining and bioinformatics. For example, in the web, a tree-structured text format called XML has become a popular method for storing documents and has been extensively used recently [2]. With larger XML documents stored on the web, mining XML documents has become an important data mining domain. XML documents are data sets of semi-structured data, or more specifically, labeled ordered trees, and thus a number of approaches, including mining frequent patterns [10, 25, 24] and mining data-streams [3], have been developed for data sets of labeled ordered trees. Kernels for labeled ordered trees

*Supported in part by Grant-in-Aid for Scientific Research on Priority Areas (C) “Genome Information Science” from the Ministry of Education, Culture, Sports, Science and Technology of Japan.

[†]Bioinformatics Center, Institute for Chemical Research, Kyoto University, Gokasho, Uji, 6110011, JAPAN

Table 1: Extension of HMM to our model.

Models	Dependency relation
HMM	Two adjacent labels in a sequence
PTMM	Parent-child (PC) in a tree
Our model	PC and two adjacent siblings in a tree

have also been developed in these last few years [19].

We propose a new probabilistic model for mining frequent patterns in labeled ordered trees, and provide a general time-efficient learning algorithm for estimating parameters of our proposed model. A hidden Markov model [4] (HMM) has been popularly used for learning sequences in a number of applications, including speech recognition [22] and biological sequence analysis [14]. A straightforward extension of HMM and its learning algorithm has been developed for mining labeled trees [16, 13]. This extended model, called a probabilistic tree Markov model (PTMM) (or probabilistic Markov tree model), captures a statistical dependency in which each vertex in a tree depends on its parent. This dependency is equivalent to that of HMMs, where there exists a dependency between two adjacent letters (labels) in a sequence¹. On the other hand, siblings in a labeled ordered tree have an order, and so there must be a dependency between these siblings. For a labeled ordered tree, we need a model to capture not only parent-child dependencies but also those between siblings. We therefore extend PTMM to a new model in which a vertex depends on the immediately elder sibling as well as on the parent. Table 1 summarizes these two extensions from HMM to our proposed model. In all of these three models, a state transition is a hidden variable. One general extension of HMM is a probabilistic context free grammar (PCFG) [20] and further a probabilistic tree grammar (PTG) [1]. These models have been proposed to capture long-range dependencies between letters in a sequence, while our model attempts to capture dependencies between labels in a tree. We note that our extension goes into a different direction from this.

¹More precisely, a dependency in HMM exists between two states emitting two adjacent letters, and a dependency in PTMM is between two states emitting two letters, one attached to a node and one to its parent.

A model for labeled ordered trees and an algorithm for estimating its parameters have never been proposed in the fields² of Bayesian belief networks or graphical modeling (i.e. in [11, 5] and corresponding older proceedings), to the best of our knowledge. Our model can be regarded as a Bayesian belief network. However, our model is categorized into a so-called ‘multiply connected Bayesian network³,’ and the parameters of the multiply connected Bayesian network cannot be estimated efficiently by existing learning algorithms for Bayesian networks, such as belief-propagation [21], etc. We emphasize that our algorithm for estimating the probability parameters of our model is based on an EM algorithm and is extremely time-efficient. In the field of grammatical inference, a regular grammar was once attempted to have developed for modeling labeled ordered trees [9], but no probabilistic models for labeled ordered trees have been developed.

Semi-structured patterns also appear in biological domains, such as glycans in the field of glycobiology. Such glycans, or sugar chains, are rooted tree structures, whose nodes are labeled by monosaccharides, which are the molecular base units for glycans. Because of the difficulty in determining the structure of glycans in detail due to technical challenges inherent in the biology of glycans, there is much to learn about these glycans essential to the development and function of complex multicellular organisms [6]. Since an understanding of the structure of glycans is important in understanding their function, and since glycans are known to be used as identifiers for enzymes as well as for recognition by microbes and pathogens [7], there seems to be some underlying complex pattern inherent in the structure of glycans. In glycans, the siblings labeled by monosaccharides are ordered, and we may consider glycans as labeled ordered trees. The siblings and parent-child dependencies within glycan structures, are essential in understanding their function in organisms, thus providing us a vital field of research to which our method for mining labeled ordered trees may be extremely useful.

We evaluate the effectiveness of our proposed method in our experiments using synthetic data sets as well as real data sets of glycans. In our experiments, we compare the performance of our method with those of several simpler probabilistic models, all of which are

²Hidden Markov decision trees [17] and hierarchical hidden Markov models [23] can be the closest work of ours. We easily see that their inputs are not labeled trees, and thus our model and its learning algorithm for mining labeled ordered trees are significantly different from them.

³A multiply connected Bayesian network is a network with a cycle, when each of all directed edges of the network is replaced with an undirected edge. See Figure 3 to confirm that our model is a multiply connected Bayesian network.

unsupervised learning methods. We evaluate the performance of each of the methods based on discriminative ability in a supervised manner, i.e. discriminating positive examples from randomly generated negative examples, as our objective is to evaluate the performance of each of the unsupervised methods by three measures, area under ROC curve [15], prediction accuracy and precision. Experimental results show that our proposed method outperforms all of the other methods compared, being statistically significant in all cases.

The key contributions of our work are the following: 1) **A general probabilistic model for mining labeled ordered trees.** Our probabilistic model is a reasonable extension from a currently-known probabilistic tree Markov model for labeled trees, to a model for mining labeled ordered trees; i.e. a probabilistic model for an emerging domain in the data mining field. Our extension considers the dependencies between ordered siblings as well as parent-child relationships in a tree, giving this model representational power, that is effective enough for capturing various types of patterns in labeled ordered trees. 2) **Learning algorithm for a probabilistic model for labeled ordered trees.** Our method provides a learning algorithm to estimate the probability parameters of the proposed model for labeled ordered trees, based on an EM algorithm. The computation time for each iteration of the algorithm is roughly $O(|S|^3 \cdot |V|)$, where $|S|$ and $|V|$ are the number of states in this model and input labels, respectively, while the corresponding computation time for HMM and PTMM is roughly $O(|S|^2 \cdot |V|)$. We claim that our model provides a far richer expressive power, which cannot be compared to the increase in computation time. In addition, recall the computational complexity of PCFG that is roughly $O(|S|^3 \cdot |V|)$, and has been used in many application domains related to natural language processing. However, a richer grammar than PCFG has almost never been used in these domains. This indicates that the complexity of PCFG is a practical upper limit, and as such, our model is also within the practical limit set by PCFG.

2 Methods

2.1 Preliminaries and Notation A *tree* is an acyclic connected graph. In this paper, we refer to a vertex of a tree as a *node* of the tree. A *rooted tree* is a tree having a special node called the root (or endpoint). Any node on a unique path from the root to a node is called an *ancestor* of the node, and if node x is an ancestor of node y , y is called a *descendant* of x . Each of the closest descendants (i.e. nodes which are only one edge away from a node) is called a *child*, and if node x is a child of node y , y is called a *parent* of x . We call

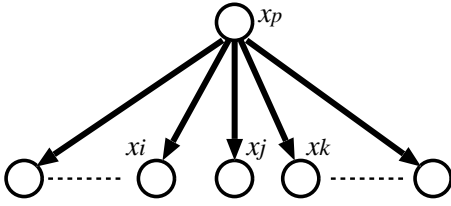


Figure 1: For node x_j in a labeled ordered tree, the immediately elder and younger nodes are nodes x_i and x_k , and the parent node is node x_p .

nodes x and y *siblings* if x and y have the same parent. We call a node having no children a *leaf*. A *subtree* of tree T is a tree whose nodes and edges are subsets of those of T . An *ordered tree* is the rooted tree in which the children of each node are ordered. A *labeled tree* is a tree in which a label is attached to each node. We note that all trees considered in this paper are ordered, labeled and rooted trees.

We use the following notations in this paper. Let $\mathbf{T} = \{T_1, \dots, T_{|\mathbf{T}|}\}$ be a set of labeled ordered trees, where $T_u = (V_u, E_u)$ and $V_u = \{x_1^u, \dots, x_{|V_u|}^u\}$ and E_u are a set of nodes and a set of edges, respectively. Let x_1^u be the root of tree T_u , and $|V| = \max_u |V_u|$. Let $t_u(i)$ be a subtree of T_u , having x_i^u as the root of $t_u(i)$. Let $L_u \subseteq \{1, \dots, |L_u|\}$ be a set of indices of leaves in t_u , and $C_u(p) \subseteq \{1, \dots, |C_u(p)|\}$ be a set of indices of children of x_p^u in T_u . Let $|C| = \max_{u,p} |C_u(p)|$. Let $x_{\leftarrow}^u(p)$ and $x_{\rightarrow}^u(p)$ be the eldest and youngest child of node p , and $Y_u(p) = C_u(p) - x_{\leftarrow}^u(p)$. Each node x_j^u has label $o_j^u \in \Sigma$, where $\Sigma = \{\sigma_1, \dots, \sigma_{|\Sigma|}\}$ is a set of labels on nodes. For the sake of simplicity, we will often use node j instead of x_j^u , if understood from the context. For a node j , we will often indicate the immediately elder and younger siblings as i and k , respectively, while the parent will be annotated as p . Figure 1 illustrates these four nodes in a subtree. For a probabilistic model, let θ be a set of parameters.

2.2 Proposed Method: Probabilistic Sibling-dependent Tree Markov Model (PSTMM)

In this section, we first describe a preliminary version of our proposed model before presenting our proposed probabilistic model. Both models have a ‘state,’ which corresponds to a node, and a label, which is attached to a node in a tree, is generated depending on the state corresponding to the node. Let $S = \{s_1, \dots, s_{|S|}\}$ be a set of states, and let $z_j^u \in S$ be a state for node j in a tree, and state z_j^u generates label o_j^u .

2.2.1 Probabilistic Tree Markov Model (PTMM)

We will now describe a preliminary version of our proposed model. In this preliminary model, the state of a node depends only on the state of

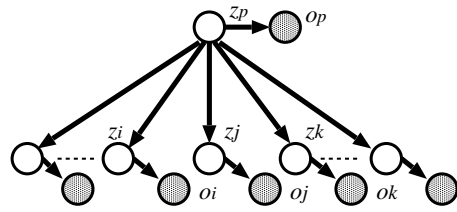


Figure 2: Dependencies in PTMM for Figure 1. A white node is a state, and a shaded node indicates a label.

its parent, and in this sense, this model is a straightforward extension of a hidden Markov model for a time-series sequence to a model for a labeled tree. This preliminary model is generally called a probabilistic tree Markov model (PTMM) [16, 13]. Figure 2 illustrates the dependencies in PTMM embedded in the tree fragment shown in Figure 1. PTMM has three types of probability parameters, π , a and b . The initial state probability $\pi[s_l]$ ($= P(z_1^u = s_l; \theta)$) is the probability that the state (z_1^u) of root node x_1^u is s_l . The state transition probability $a[s_l, s_m]$ ($= P(z_j^u = s_m | z_p^u = s_l; \theta)$) is the conditional probability that the state of a node is s_m given that the state of its parent is s_l . The label output probability $b[s_l, \sigma_h]$ ($= P(o_j^u = \sigma_h | z_j^u = s_l; \theta)$) is the conditional probability that the output label of a node is σ_h given that the state of this node is s_l . For simplicity, we hereafter use $\pi[l]$, $a[l, m]$ and $b[l, \sigma_h]$, instead of $\pi[s_l]$, $a[s_l, s_m]$ and $b[s_l, \sigma_h]$. Note that $\sum_l \pi[l] = 1$, $\sum_m a[l, m] = 1$, and $\sum_h b[l, \sigma_h] = 1$.

To describe the probabilistic structure of this model, we further define an upward probability $\alpha_u(s_l, v_j^u)$ ($= P(t_u(j) | z_j^u = s_l; \theta)$), which is the probability that the labels of subtree $t_u(j)$ are all generated and that the state of node x_j^u is s_l . We hereafter use $\alpha_u(l, j)$, instead of $\alpha_u(s_l, v_j^u)$, for simplicity. Note that the upward probabilities at node j can be calculated from the upward probability at each of the children of j and the probability parameters of PTMM. This can be formulated as follows:

$$\alpha_u(l, p) = \begin{cases} \text{If } p \in L_u \text{ then } b[l, o_p^u], & \text{otherwise} \\ b[l, o_p^u] \prod_{j \in C_u(p)} \sum_m a[l, m] \alpha_u(m, j) \end{cases}$$

We can repeat this computation of the upward probability at each node from the leaves to the root for a given tree. Using this bottom-up dynamic programming procedure, we can obtain the upward probabilities for all nodes of the tree.

Finally, using the upward probability at the root $\alpha_u(l, 1)$, the likelihood for T_u is given by PTMM as follows: $L(T_u; \theta) = \sum_l \pi[l] \alpha_u(l, 1)$. Using the likelihood for a tree, we can compute the likelihood for a given set of trees as a product of the likelihood for each tree, as

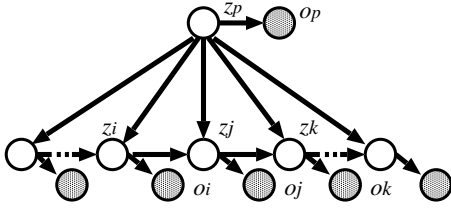


Figure 3: Dependencies in PSTMM for Figure 1. A white node is a state, and a shaded node is a label.

follows:

$$L(\mathbf{T}; \theta) = \prod_u \sum_l \pi[l] \alpha_u(l, 1).$$

One criterion for estimating the probability parameters of PTMM is the maximum likelihood (ML), in which parameters are estimated to maximize the above likelihood of a set of given trees. We can use a general scheme called EM (Expectation-Maximization) algorithm [12] to obtain the ML estimators of PTMM. In the EM algorithm for PTMM, we define a downward probability $\beta_u(s_l, v_i^u)$, which is the probability that all labels of tree T_u except for those of subtree $t_u(i)$ are generated and that the state of node i is s_l . By using the downward probability as well as the upward probability, we can implement the EM algorithm for PTMM in a similar manner to the forward-backward and Baum-Welch algorithms in HMM.

PTMM can be used for mining labeled trees when a dependency exists between a node and its parent in a tree. However, in the ordered siblings in a labeled ordered tree, there must exist dependencies, which cannot be captured by PTMM.

2.2.2 Probabilistic Sibling-dependent Tree Markov Model (PSTMM) We propose a new model, which is a reasonable extension of PTMM for mining labeled ordered trees, and we call this model PSTMM, for *probabilistic sibling-dependent tree Markov model*. In PTMM, the state of a node depends on the state of its parent only, whereas in our proposed model, the state of a node depends on the state of the immediately elder sibling as well as on the state of its parent. Figure 3 illustrates the dependencies in PSTMM embedded in the tree fragment of Figure 1. We emphasize that incorporating this dependency on the immediately elder sibling drastically improves the performance of PTMM for finding patterns in given labeled ordered trees. Recall that HMMs can capture a distant (long-range) dependency in a sequence indirectly if a state transition can be set to capture such a dependency. Similarly, PTMM can capture a long-range dependency in a tree indirectly, but this

indirect dependency is limited to the one between a node and its descendant. However, PSTMM can capture a variety of dependencies in a tree as well as the descendant dependency. For example, a dependency between distant siblings may be found by PSTMM indirectly. Furthermore, an indirect dependency between a node and its distant sibling's descendant may also be captured by PSTMM.

PSTMM has three probability parameters, π , a and b , where π and b are the same as those defined in PTMM. a differs between PTMM and PSTMM. The state transition probability $a[\{s_q, s_l\}, s_m]$ ($= P(z_j^u = s_m | z_p^u = s_q, z_i^u = s_l; \theta)$) is newly defined as the conditional probability that the state of a node is s_m given that the states of its parent and the immediately elder sibling are s_q and s_l , respectively. As we defined for PTMM, for simplicity, we hereafter use $\pi[l]$, $a[\{q, l\}, m]$ and $b[l, \sigma_h]$, instead of $\pi[s_l]$, $a[\{s_q, s_l\}, s_m]$ and $b[s_l, \sigma_h]$, respectively. Note that $\sum_m a[\{s_q, s_l\}, s_m] = 1$.

To describe the probabilistic structure of PSTMM, we define new forward and backward probabilities in addition to the upward probability that was already introduced in describing PTMM. The forward probability $F_u(s_q, s_l, x_j^u)$ is the probability that for node j , all labels of a subtree for each of the elder siblings are generated and the state of node j is s_l and the state of parent p is s_q . The backward probability $B_u(s_q, s_m, x_j^u)$ is the probability that for node j , all labels of a subtree for each of the younger siblings and node j are generated, the state of node j is s_m , and the state of parent p is s_q . Again, the upward probability $U_u(s_q, x_p^u)$ is the probability that all labels of subtree $t_u(p)$ are generated and the state of node p is s_q . For simplicity, we hereafter use $F_u(q, l, j)$ and $B_u(q, m, j)$ and $U_u(q, p)$, instead of $F_u(s_q, s_l, x_j^u)$, $B_u(s_q, s_m, x_j^u)$ and $U_u(s_q, x_p^u)$, respectively.

We can compute these three probabilities, from leaves to the root, using a bottom-up dynamic programming procedure similar to that used in PTMM. This computation can be formulated as follows:

$$F_u(q, l, j) = \begin{cases} \text{If } x_j^u = x_{\leftarrow}^u(p) \text{ then } a[\{q, -\}, l], \\ \text{otherwise} \\ \sum_m F_u(q, m, i) U_u(m, i) a[\{q, m\}, l]. \end{cases}$$

$$B_u(q, m, j) = \begin{cases} \text{If } x_j^u = x_{\rightarrow}^u(p) \text{ then } U_u(m, j), \\ \text{otherwise} \\ U_u(m, j) \sum_l a[\{q, m\}, l] B_u(q, l, k). \end{cases}$$

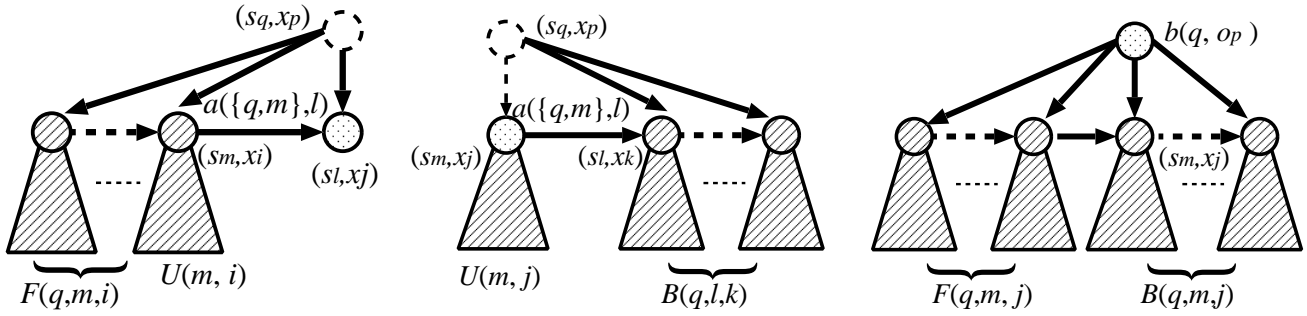


Figure 4: Updating (a) $F_u(q, l, j)$, (b) $B_u(q, m, j)$ and (c) $U_u(q, p)$. A sparse shaded node is j (or p for $U_u(q, p)$), and dense shaded areas are used for updating.

$$U_u(q, p) = \begin{cases} \text{If } C_u(p) = \emptyset \text{ then } b[q, o_p^u], \\ \text{otherwise} \\ b[q, o_p^u] \sum_m F_u(q, m, j) B_u(q, m, j). \\ \text{(s.t. } j \in C_u(p)) \end{cases}$$

Figure 4 is a schematic of this procedure.

The likelihood for a given tree is obtained by using $U_u(l, 1)$, U at the root of the tree, as was done for PTMM. Here we note that to obtain $U_u(l, 1)$, we can omit updating F , although F appears in calculating U at each node. This is because $U_u(q, p)$ can be calculated using any child of p , j , and if we always use the eldest sibling as j , we can always use $a[\{q, -\}, m]$ instead of $F_u(q, m, j)$ in calculating $U_u(q, p)$. This indicates that the likelihood for a given tree can be computed using U and B only. The likelihood for a given tree is given in the following equation, using U at the root of the tree.

$$L(T_u; \theta) = \sum_l \pi[l] U_u(l, 1).$$

The likelihood for a given set of trees is computed as a product of the likelihood for each tree in the set, as shown in the previous section.

In PTMM, U at a node is iteratively computed from the U s at its children, whereas in PSTMM, U at a node is computed using the F s and B s at an arbitrary child, without using any U s. In PSTMM, F at a node is iteratively computed using the U at the immediately elder sibling. Similarly, B at a node is iteratively computed using the U at that node. Thus, interestingly, in PSTMM, U s are used for computing F and B , and F s and B s are used for computing U .

Another interesting point is that the parameter updating of PSTMM has roughly two common features to that of HMM. First, F and B of PSTMM are similar to those of HMM (although calculating these probabilities in PSTMM is far more complex than in HMM). In calculating these probabilities, a set of siblings in PSTMM correspond to a sequence given to HMM. The

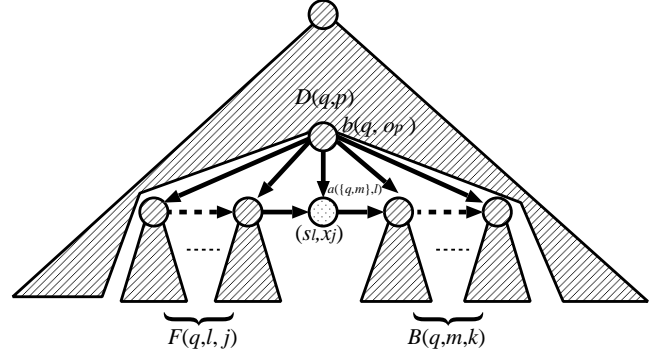


Figure 5: Updating $D_u(l, j)$. A sparse shaded circle is node j , and dense shaded areas are used for updating $D_u(l, j)$.

forward (backward) probability of HMM is the probability that the first (last) part of a sequence is generated, while F (B) of PSTMM is also the probability that the labels of the subtrees of elder siblings (younger siblings and itself) are all generated. In updating, F (B) of PSTMM is calculated from the F s (B s) of smaller subtrees, while those of PSTMM is calculated from those of shorter subsequences. Second, U in PSTMM at a node is similar to the likelihood for a sequence in HMM. In HMM, the likelihood for a sequence is computed by forward and backward probabilities (at any letter in the sequence), so as to generate all letters in the sequence. In PSTMM, U at a node x also can be computed using F and B (at any child), so as to generate all labels of the subtree whose root is x . In summary, calculating F , B and U of PSTMM corresponds to calculating the forward probability, the backward probability and the likelihood for a given sequence in HMM.

2.2.3 Estimating Parameters of PSTMM For estimating the probability parameters of PSTMM, we again use the ML estimators and an EM algorithm.

To describe the EM procedure for PSTMM, in addition to F , B and U , we define downward probability $D_u(s_l, x_j^u)$, which is the probability that all labels of

a tree except for those of subtree $t_u(j)$ are generated and that the state of node x_j^u is s_l . For simplicity, we hereafter use $D_u(l, j)$, instead of $D_u(s_l, x_j^u)$. The downward probability at a node can be computed using the downward probability at its parent and the forward and backward probabilities at its siblings, as follows:

$$D_u(l, j) = \begin{cases} \text{If } j \text{ is the root then } \pi[l], \\ \text{else if } j = x_{\rightarrow}^u(p) \text{ then} \\ \quad \sum_q D_u(q, p) b[q, o_p^u] F_u(q, l, j), \\ \text{otherwise} \\ \quad \sum_q D_u(q, p) b[q, o_p^u] F_u(q, l, j) \\ \quad \sum_m a[\{q, l\}, m] B_u(q, m, k) \end{cases}$$

Figure 5 illustrates this procedure.

We note that for any node i , the likelihood for a tree is calculated by the upward and downward probabilities at node i , as follows:

$$L(T_u; \theta) = \sum_l U_u(l, i) D_u(l, i).$$

Figure 6 provides the pseudocode for calculating the four types of probabilities, F , B , U and D , based on a bottom-up and top-down dynamic programming procedure. As shown in the figure, we first perform a so-called level-order numbering of nodes of a given tree, from the root to leaves⁴. We then calculate U , B and F in the reverse order, from leaves to the root, and finally calculate D in order from the root back to the leaves.

Using these four probability parameters, we compute the three expectation values, γ , δ and η . $\gamma_u(\{s_q, s_m\}, s_l)$ is the expectation value that the state of a node is s_l and that the states of its parent and immediately elder sibling are s_q and s_m , respectively. $\delta_u(s_m, \sigma_h)$ is the expectation value that the state and the label of a node is s_m and σ_h , respectively. $\eta_u(s_m)$ is the expectation value that the state of the root is s_m . We denote $\gamma_u(\{s_q, s_m\}, s_l)$, $\delta_u(s_m, \sigma_h)$ and $\eta_u(s_m)$ by $\gamma_u(\{q, m\}, l)$, $\delta_u(m, \sigma_h)$ and $\eta_u(m)$ for simplicity. These expectation values can be calculated using F , B , U and D . We then perform the iterations of the EM algorithm as follows:

In the E-step, the expectation values are calculated using current probability parameters:

$$\gamma_u(\{q, -\}, l) = \frac{\sum_{p: C_u(p) \neq \emptyset} D_u(q, p) b[q, o_p^u] a[\{q, -\}, l] B_u(q, l, j)}{L(T_u; \theta)},$$

where $j = v_{\leftarrow}^u(p)$,

⁴We here assume that the ordering used is set according to the ordering of siblings.

$$\gamma_u(\{q, m\}, l) = \frac{\sum_{p: C_u(p) \neq \emptyset} D_u(q, p) b[q, o_p^u] \sum_{j: x_j^u \in Y_u(p)} H_u(q, m, l, j)}{L(T_u)},$$

where

$$H_u(q, m, l, j) = F_u(q, m, i) U_u(m, i) a[\{q, m\}, l] B_u(q, l, j),$$

$$\delta_u(m, \sigma_h) = \frac{\sum_{i: o_i^u = \sigma_h} D_u(m, i) U_u(m, i)}{L(T_u)},$$

$$\eta_u(m) = \frac{\pi[m] U_u(m, 1)}{L(T_u)}.$$

In the M-step, probability parameters are updated using the expectation values calculated in the E-step:

$$\begin{aligned} \hat{a}[\{s_q, -\}, s_l] &= \frac{\sum_u \gamma_u(\{s_q, -\}, s_l)}{\sum_u \sum_{l'} \gamma_u(\{s_q, -\}, s_{l'})}, \\ \hat{a}[\{s_q, s_m\}, s_l] &= \frac{\sum_u \gamma_u(\{s_q, s_m\}, s_l)}{\sum_u \sum_{l'} \gamma_u(\{s_q, s_m\}, s_{l'})}, \\ \hat{b}[s_m, \sigma_h] &= \frac{\sum_u \delta_u(s_m, \sigma_h)}{\sum_u \sum_i \delta_u(s_m, \sigma_i)}, \\ \hat{\pi}[s_m] &= \frac{\sum_u \eta_u(s_m)}{\sum_u \sum_{k'} \eta_u(s_{k'})}. \end{aligned}$$

We repeat the E- and M-steps alternately until a certain convergence criterion is satisfied.

2.3 Computation Time for Estimating Parameters of PSTMM Out of all the equations for calculating the expectation values and updating the parameters in the E- and M-steps of our learning algorithm, the most time-consuming part of this algorithm is obviously calculating γ . The computation time for calculating γ with all possible parameter values reaches $O(|\mathbf{T}| \cdot |S|^3 \cdot |V| \cdot |C|)$ since we must compute $O(|V| \cdot |C|)$ for each γ and then repeat this $O(|S|^3)$ times for all possible combinations of states. Table 2 summarizes the time for computing (updating) each of the probability parameters and expectation values.

Table 2: Time for computing parameters.

	Comp. time
F	$O(\mathbf{T} \cdot S ^3 \cdot V)$
B	$O(\mathbf{T} \cdot S ^3 \cdot V)$
U	$O(\mathbf{T} \cdot S ^2 \cdot V)$
D	$O(\mathbf{T} \cdot S ^3 \cdot V)$
γ	$O(\mathbf{T} \cdot S ^3 \cdot V \cdot C)$
δ	$O(\mathbf{T} \cdot S \cdot V)$
η	$O(\mathbf{T} \cdot S)$
a	$O(\mathbf{T} \cdot S ^3)$
b	$O(\mathbf{T} \cdot S \cdot V)$
π	$O(\mathbf{T} \cdot S)$

2.4 Other Methods Compared with PSTMM in our experiments We show two simple probabilistic models and their corresponding mixture versions, which were compared with our proposed model, PSTMM, in our experiments. The simpler one of the two models treats each node independently, and the other more complex model focuses on each parent-child pair. Note that both models do not have ‘states,’ which are used in PTMM and PSTMM, since a state in these simple models corresponds one-to-one with a label, making states unnecessary.

2.4.1 Label Model (LM) and its Mixture (MLM)

2.4.1.1 Label model

We first explain the simplest model, which we call LM, for *label model*. This model has only one probability parameter, $w[\sigma_h]$ ($\sum_h w[\sigma_h] = 1$), which is the label output probability, the probability that label σ_h is outputted at a node. The value of this parameter is simply calculated by counting the labels appearing in the given labeled ordered trees. Parameter estimation for this model is not an iterative procedure but runs just once.

I. Estimating parameters

1. Compute $n[\sigma_h]$, the number of times label σ_h appears in the input tree set.
2. Compute $w[\sigma_h]$ as follows: $w[\sigma_h] = \frac{n[\sigma_h]}{\sum_{h'} n[\sigma_{h'}]}$.

II. Computing likelihoods

The likelihood L for a given set of trees is given as follows: $L = \prod_u \prod_k w[o_k^u]$.

2.4.1.2 Mixture of Label Models

Here we consider a mixture of LMs, which we call MLM, for *mixture of LMs*. We hereafter call each LM of MLM a *component*. Let c be a component and Z be the number of components. For each component, different

probability parameter values are given, and we have two probability parameters, $w[c, \sigma_h]$ ($\sum_h w[c, \sigma_h] = 1$ for each c) and $v[c]$ ($\sum_c v[c] = 1$). The method for estimating parameters of MLM is done by an EM procedure, which is provided in the Appendix. The likelihood for given set of trees is computed as follows:

$$L = \prod_u \sum_c v[c] \prod_k w[c, o_k^u].$$

2.4.2 Label Pair Model (LPM) and its Mixture (MLPM)

2.4.2.1 Label Pair Model

This model, which we call LPM, standing for a *label pair model*, has two parameters, $w[\sigma_h, \sigma_{h'}]$ ($\sum_h w[\sigma_h, \sigma_{h'}] = 1$) and $\pi[h]$ ($\sum_h \pi[\sigma_h] = 1$). The $w[\sigma_h, \sigma_{h'}]$ ($= P(o_j^u = \sigma_h | o_p^u = \sigma_{h'})$) is the label pair probability, or the probability that label σ_h is outputted at a node given that label $\sigma_{h'}$ is outputted at its parent node, and $\pi[\sigma_h]$ is the probability that the root label is σ_h . As in LM, estimating the parameters of this model is not an iterative procedure but runs just once.

I. Estimating parameters

1. Compute $n[\sigma_h, \sigma_{h'}]$, the number of times labels σ_h and $\sigma_{h'}$ appear at both a node and its parent, respectively, and $r[\sigma_h]$, the number of times label σ_h appears at the root in the input tree set.
2. Compute $w[\sigma_h, \sigma_{h'}]$ and $\pi[\sigma_h]$ as follows: $w[\sigma_h, \sigma_{h'}] = \frac{n[\sigma_h, \sigma_{h'}]}{\sum_i n[\sigma_i, \sigma_{h'}]}$ and $\pi[\sigma_h] = \frac{r[\sigma_h]}{\sum_i r[\sigma_i]}$.

II. Computing likelihoods

The likelihood L for a given set of trees is given as follows: $L = \prod_u \pi[o_1^u] \prod_{j (\neq 1)} a[o_j^u, o_p^u]$.

2.4.2.2 Mixture of Label Pair Models

We extend LPM to a mixture model, which we call MLPM, for *mixture of label pair models*. The probability parameters in MLPM are $w[c, \sigma_h, \sigma_{h'}]$ ($\sum_h w[c, \sigma_h, \sigma_{h'}] = 1$ for each pair of c and h') and $\pi[c, \sigma_h]$ ($\sum_h \pi[c, \sigma_h] = 1$ for each c). $w[c, \sigma_h, \sigma_{h'}]$ ($= P(o_j^u = \sigma_h | o_p^u = \sigma_{h'}, c)$) is the conditional probability that label σ_h is outputted at a node given that $\sigma_{h'}$ is outputted at its parent node, for component c . $\pi[c, \sigma_h]$ ($= P(o_1^u = \sigma_h | c)$) is the probability that the root label is σ_h for component c . As in MLM, we describe the procedure to obtain the ML estimators of MLPM in the Appendix. The likelihood L for a given set of trees is given by MLPM as follows:

$$L = \prod_u \sum_c p(c) \pi[c, o_1^u] \prod_j w[c, o_j^u, o_p^u].$$

Here let us compare these models for trees with models for sequences. First, in HMM, which corre-

Table 3: Comparison between models for labeled trees and for sequences.

For labeled trees	For sequences
LPM	P1MM
MLPM	MP1MM
PTMM	HMM

sponds to PTMM for labeled trees, a letter emitting probability is attached to each state, and a string is not uniquely generated by a state transition, which then can be a hidden variable. If only one symbol is generated at each state in HMM, this model can be a simpler model having no hidden variables. This simple model is usually called a probabilistic first-order Markov model (P1MM), which corresponds to LPM for labeled trees. We can naturally consider a mixture of P1MM (MP1MM), which corresponds to MLPM for trees. Table 3 summarizes these relations between models for trees and models for sequences.

We further note that for capturing sequence patterns based on the first order Markov property, MP1MM has a representational power that is equal to that of HMM, and as such, MP1MM has been used frequently for obtaining multiple sequence patterns from given sequences such as in web access patterns [8]. Similarly, this equivalence can be applied to MLPM and PTMM for capturing multiple parent-child relationships in a given set of trees. For this reason, we suffice it to use MLPM to compare with the performance of PSTMM in our experiments.

3 Experimental Results

We evaluate the performance of our proposed method, comparing it with those of the other four simpler probabilistic models, LM, MLM, LPM, and MLPM. In this evaluation, we use synthetic data sets as well as real-world data sets of labeled ordered trees, data sets of glycans.

3.1 Evaluation Procedure We evaluated these five methods in a manner for supervised learning methods. More concretely, we first generated training and test examples, consisting of positive examples only, and trained each method with the generated training examples. We then generated negative test examples so that the distribution of parent-child pair labels was equal to that of the positive test examples. We evaluated each of the five methods by their ability to discriminate positives from negatives in each test data set. Note that since the distribution of parent-child pair labels was made the same for both positive and negative examples, it was a difficult task for the simpler probabilistic

```

procedure calculate()
  calculate(root);
  calculateU(root);
  calculateFB(root);
  calculateD(root);
procedure calculate(x)
  /* for all children c of x, from oldest to youngest */
  for each c ∈ C(x) do
    calculate(c)
  /* from oldest child to youngest child */
  calculateU(firstBorn);
  calculateFB(firstBorn);
procedure calculateU(x)
  for each sq ∈ S do
    find U(q, x);
  /* traverse to immediately younger sibling */
  if x has younger sibling do
    calculateU(youngerSibling);
procedure calculateFB(x)
  for each sq ∈ S do
    for each sl ∈ S do
      find F(sq, sl, x);
  if x has younger sibling do
    /* traverse to immediately younger sibling */
    calculateFB(youngerSibling);
  else
    /* traverse to immediately elder sibling */
    calculateBF(elderSibling);
procedure calculateBF(x)
  for each sq ∈ S do
    for each sm ∈ S do
      find B(sq, sm, x);
  /* traverse to immediately elder sibling */
  if x has elder sibling do
    calculateBF(elderSibling);
procedure calculateD(x)
  for each sq ∈ S do
    find D(q, x);
  /* for all children c of x */
  for each c ∈ C(x) do
    calculateD(c)

```

Figure 6: Pseudocode for F, B, U and D in PSTMM

models, LM, MLM, LPM and MLPM, to discriminate positives from negatives. We evaluated the performance of each of the methods by the following three measures: AUCs, prediction accuracies and precisions at recall of 30%.

We first drew an ROC curve by plotting ‘sensitivity (true positive rate)’ against ‘false positive rate.’ The sensitivity is the proportion of the number of correctly predicted examples to the total number of positive examples, and the false positive rate is the proportion of the number of false positives to the total number of negative examples. As is well known, for a certain false positive rate, the higher the sensitivity, the better the performance. We then measured the area surrounded

by (1) the computed ROC curve, (2) the line obtained by the false positive rate of 1 and (3) the line obtained by the sensitivity of 0, which is called area under ROC curve (AUC) [15]. The AUC takes a value between 1 and 0, and we can easily see that the larger the AUC, the better the performance of the method.

We then computed the prediction accuracy for each of the five methods by first sorting the examples according to their computed likelihoods (or scores) and then finding a threshold that discriminates the positives from the negatives to maximize the discrimination accuracy. The prediction accuracy⁵ also takes a value of between 1 and 0, with a larger value indicating more accuracy.

We finally used ‘precision’ and ‘recall.’ The recall is the same as the sensitivity, and the precision is the proportion of the number of correctly predicted examples to the number of those examples predicted to be positives. If we actually check the true label of a predicted example whose label is unknown, we will choose a relatively small subset, for which a reasonably high precision can be achieved. For our experiments, we chose 0.3 as a reasonable recall value.

We further used the pairwise mean significance statistical test of ‘*t*’ values to statistically compare the performance of PSTMM with each of the other methods. The *t* values are calculated using the following formula: $t = \frac{|ave(A)|}{\sqrt{\frac{var(A)}{n}}}$, where *A* denote the difference between the performance measures of the two methods for each data set in our five trials, *ave(W)* is the average of *W*, *var(W)* is the variance of *W*, and *n* is the number of data sets (five in our case). For *n* = 5, if *t* is greater than 4.604 then it is more than 99% statistically significant that PSTMM has a higher performance than the other.

All of our experiments were performed on a linux machine with dual Intel Xeon 3.0GHz processors and 8GByte of main memory.

3.2 Synthetic Data In our experimental setting for synthetic data, each positive example was embedded with tree fragment patterns, and each negative example was simply randomly generated from the parent-child distribution of the positive examples. To embed such tree fragment patterns to each example, we first randomly generated five of the patterns, whose labels differed. We then randomly generated tree structures and labels, the number of nodes of each of which was all fixed at twenty, and then we randomly selected five

nodes from each generated tree and attempted to embed each of the generated patterns into each of the selected nodes. We used seven types of tree fragment patterns in our experiments. For all of the tree examples tested, we fixed the parameters of the models in our experiments to the following⁶: $|T| = 200$ for training (400 for test), $|S| = 10$, $|V_u| = 20$, $|\Sigma| = 10$, $|C| = 5$ and $Z = 10$. We allowed any transition between arbitrary two states in PSTMM. We note that in this setting, both *a* in PSTMM and *w* in MPLM have the same parameter size, 1,000 (= 10 × 10 × 10), and we can say that these methods were tested on an almost equal condition.

3.2.1 Seven Patterns and Performance Results

The most important feature of our proposed model is that the dependency between siblings is considered. As such, six of our seven patterns (abbreviated Q1 through Q7) contain two or more siblings, as illustrated in Figure 7. In Figure 7, each label in a pattern is attached to a solid circle. Q1 is the most standard pattern, which contains a parent and two siblings. Q2 is a simpler pattern (subfragment of Q1), which contains only two siblings. We can thus check the performance of our method, when the dependency of a child on its parent is removed. Q3, an extension of Q2, has three consecutive siblings, and we can check whether or not this consecutive pattern will be captured by PSTMM. Q4 is a combination of Q1 and Q2, where node *j* of Q2 is node *p* of Q1. Q5 is a pattern between a node and its immediately younger sibling’s child. Q6 is a combination of Q5 and Q2, where node *i* of Q2 is node *i*₁ of Q5. Finally, Q7 is a modification of Q3, by setting node *j* of Q3 randomly. Each of Q5 through Q7 contains distant nodes, so as to check whether or not such distant relationships in a tree can be captured by PSTMM.

Tables 4, 5 and 6 show the AUCs, prediction accuracies and precisions (at recall of 0.3), respectively, for the five methods tested on these seven patterns. In each of these tables, *t*-values are added in parentheses. As shown in these tables, PSTMM outperformed all of other methods in terms of all measures tested, being statistically significant. Specifically, the AUCs of PSTMM reached 70 to 96% for all cases, while those of the other methods were at most 60% except for one (MLPM for Q1). The results show that Q3 is the easiest pattern for which PSTMM can discriminate between positives and negatives, Q2 and Q1 are the second and the third easiest, and Q5 is the most difficult. In our experimen-

⁵More precisely, all the AUCs, the prediction accuracies and precisions take a value between 1 and around 0.5, since in our experimental setting (i.e. even size of positives and negatives), even a random guessing can achieve the AUC (prediction accuracy, precision) of 0.5.

⁶Experiments using different parameter settings were also performed, particularly in changing the number of states used by PSTMM. We note that in all cases we tested, the performance advantage of PSTMM over other methods was consistent with what is presented in this paper. This is also true of real data.

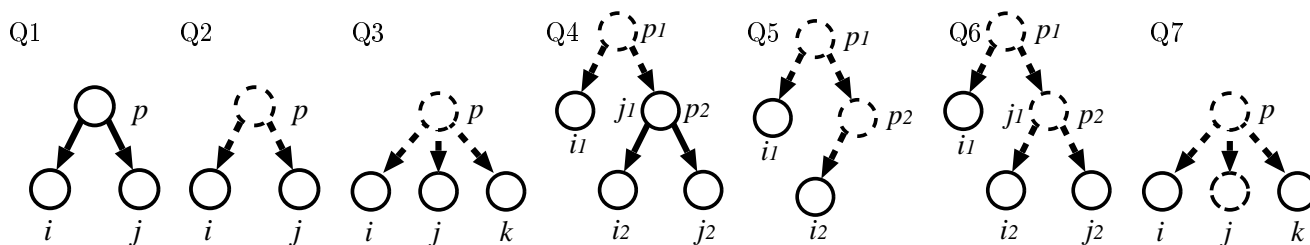


Figure 7: Seven patterns of tree fragments used in our experiments.

Table 4: AUCs and t -values (in parentheses) for synthetic data

Methods	Q1	Q2	Q3	Q4	Q5	Q6	Q7
PSTMM	87.6	90.4	96.1	80.0	70.3	77.6	84.5
MLPM	71.8 (12.5)	48.5 (32.8)	51.2 (75.2)	58.9 (6.24)	51.8 (16.1)	51.5 (21.4)	49.1 (21.5)
LPM	50.7 (40.0)	58.7 (27.6)	58.4 (36.8)	60.3 (6.48)	50.0 (18.3)	50.1 (56.3)	51.5 (39.1)
MLM	49.9 (64.0)	52.6 (39.0)	55.6 (37.8)	49.9 (9.65)	49.9 (21.7)	49.9 (36.0)	49.9 (29.6)
LM	57.6 (40.9)	53.9 (33.9)	56.3 (31.2)	54.2 (8.90)	47.1 (16.2)	47.5 (26.6)	49.8 (27.4)

Table 7: Data Summary.

Data set	# total trees	Average #nodes
N-Glycan	1865	11.08
O-Glycan	515	6.65

tal settings, the distribution of parent-child pairs is set the same for both positives and negatives, and thus the results indicate that patterns consisting of siblings can be more easily captured by PSTMM. For patterns Q5 and Q7, each of which does not contain any direct dependencies (parent-child or siblings), PSTMM achieved a high performance, and so we can claim that PSTMM can be applied to find such patterns consisting of indirect, long-range dependencies. The t -values for Q4 are the smallest among all the patterns, since Q4 (and Q1) contains parent-child relationships, making it the easiest pattern for MLM and MLPM, both of which attempt to capture parent-child pairs of labels.

3.3 Real Data: Glycans We further used a data set of glycans, which were all obtained from KEGG-Glycan database [18]. Glycans are classified into roughly ten to fifteen types, based on their biological properties. Out of these types, we selected two major classes, called ‘N-Glycan’ and ‘O-Glycan’ and for each class, we further selected those structures that contained at least one sibling pair, as data sets for our experiment. Table 7 gives a summary of these two data sets. We performed a five-fold cross-validation for each of the selected two data sets. That is, we divided the data set into five blocks of roughly equal size, and in each of five trials, a different block was selected as the test set while the remaining four were used for training. We finally repeated this process five times. The results were then averaged over the 25 ($= 5 \times 5$) runs. The

parameters used in this experiment were as follows: $|T| = 200$ for training⁷, $|S| = 10$, $|\Sigma| = 19$, $|C| = 5$ and $Z = 10$. As was done in synthetic data, we used a fully-connected state transition network for PSTMM. Since we had the varied sizes of the trees in each data set, the likelihood for a given tree drastically changed depending on the number of nodes in the tree. So, to correct for this discrepancy, each probability parameter value was multiplied by its size. That is, for example, we multiply $a[\{s_q, s_m\}, s_i]$ by $|S|$. We used these corrected parameter values to calculate the likelihood of each tree, and thus used the corrected likelihood (score) for each tree to evaluate each method.

3.3.1 Performance Results Table 8 shows the AUCs, prediction accuracies and precisions (at recall of 0.3), respectively, for the five methods tested on our two data set. t -values are added in parentheses in this table. The results show that PSTMM clearly outperformed the other four methods tested in all cases, being statistically significant. The performance results obtained by PSTMM for N-Glycan are higher than those for O-Glycan most likely due to the larger tree sizes and increased sibling pairs. Figure 8 shows the ROC curves obtained by the five methods tested for N-Glycan, and also indicates that PSTMM clearly outperformed the other four methods compared. Overall, these results are surprisingly excellent and almost equal to those obtained for synthetic data sets, implying that there must be some complex pattern that are not limited to parent-child (more generally, ancestor-descendant) relationships, in the real data sets of glycans.

⁷200 examples were randomly chosen from the training blocks, and the test size depended on the data set.

Table 5: Prediction accuracies and t -values (in parentheses) for synthetic data

Methods	Q1	Q2	Q3	Q4	Q5	Q6	Q7
PSTMM	80.9	83.8	90.7	73.9	66.1	72.3	78.2
MLPM	66.8 (10.7)	51.2 (47.5)	53.2 (43.4)	58.1 (5.97)	53.3 (10.4)	52.9 (23.9)	51.4 (18.7)
LPM	52.9 (30.4)	58.0 (32.7)	58.7 (37.4)	59.5 (4.86)	52.0 (15.1)	52.2 (29.6)	52.9 (30.2)
MLM	50.0 (35.0)	58.6 (26.9)	60.7 (35.0)	50.0 (8.73)	50.0 (18.5)	50.0 (32.8)	50.0 (25.1)
LM	58.3 (21.8)	55.9 (31.6)	56.5 (25.9)	56.6 (6.27)	52.2 (12.9)	52.5 (23.8)	53.3 (20.1)

Table 6: Precisions at recall of 0.3 and t -values (in parentheses) for synthetic data

Methods	Q1	Q2	Q3	Q4	Q5	Q6	Q7
PSTMM	99.2	95.3	99.0	87.3	75.7	85.2	88.2
MLPM	76.8 (14.0)	48.8 (61.5)	53.8 (39.1)	59.1 (4.64)	51.6 (14.8)	50.8 (38.9)	49.2 (19.0)
LPM	49.7 (28.3)	59.2 (35.4)	64.0 (13.7)	57.8 (6.44)	48.9 (13.2)	51.3 (20.2)	52.5 (44.2)
MLM	50.0 (123.5)	46.9 (75.0)	48.8 (55.7)	50.0 (6.99)	50.0 (13.6)	50.0 (29.7)	50.0 (29.2)
LM	54.3 (28.8)	50.0 (38.4)	55.8 (36.4)	51.2 (7.03)	45.4 (13.9)	45.6 (40.2)	47.4 (28.4)

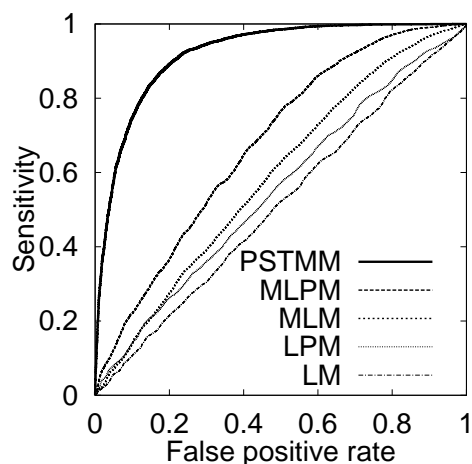


Figure 8: The ROC curves for N-Glycan.

4 Concluding Remarks

Labeled ordered trees, which have appeared as a new data structure in a number of data mining applications, such as on the web and in bioinformatics. Correspondingly, we have proposed a new probabilistic methodology to handle such trees. The effectiveness of the proposed method was experimentally confirmed by using both synthetic and real-world data.

One item of note is that our methodology is an unsupervised learning/mining approach and fits well for real-world applications, although we evaluated the performance of our method by a supervised learning manner in our experiments. That is, the most important work to do for real world applications, such as mining XML documents or glycans, is to find frequent patterns hidden in the given examples. This is work usually done by an unsupervised learning approach, and the following procedure can be done in this case: We first estimate the probability parameters of our model using

our proposed learning algorithm, and we then compute the maximum likelihood path for a highly possible state transition, as is done by the Viterbi algorithm for HMM for an arbitrary given sequence. As was shown in our experimental results, our method can capture a pattern that is more complex than parent-child (ancestor-descendant) relationships. Therefore, our probabilistic framework for labeled ordered trees is effective for mining such complex patterns, which have never been captured by any other existing probabilistic models, which can be trained by a time-efficient learning algorithm such as an EM algorithm. We believe that our methodology will greatly contribute to domains, in which mining labeled ordered trees is essential.

References

- [1] N. ABE AND H. MAMITSUKA, *Predicting protein secondary structure using stochastic tree grammars*, Machine Learning, 29 (1997), pp. 275–301.
- [2] S. ABITEBOUL, P. BUNEMAN, AND D. SUCIU, *Data on the Web: from relations to semistructured data and XML*, Morgan Kaufmann, 2000.
- [3] T. ASAI ET AL., *Online algorithms for mining semi-structured data stream*, in Proceedings of the IEEE ICDM, 2002, pp. 27–34.
- [4] E. BAUM AND T. PETRIE, *Statistical inference for probabilistic functions of infinite state Markov chains*, Ann. Math. Stat., 37 (1966), pp. 1554–1563.
- [5] S. BECKER, S. THRUN, AND K. OBERMAYER, eds., *Proceedings of NIPS 15*, 2003.
- [6] C. BERTOZZI AND L. KIESSLING, *Chemical glycobiology*, Science, 291 (2001), pp. 2357–2364.
- [7] S. BROOKS, M. DWEK, AND U. SCHUMACHER, *Functional and Molecular Glycobiology*, BIOS Scientific, 2002.
- [8] I. CADEZ ET AL., *Model-based clustering and visualization of navigation patterns*, Data Mining and Knowledge Discovery, 7 (2003), pp. 399–404.

Table 8: Performance results for real data (t -values are in parentheses.).

Methods	N-Glycan			O-Glycan		
	ROC	Acc.	Prec.	ROC	Acc.	Prec.
PSTMM	92.0	85.5	95.6	80.1	75.3	84.1
MLPM	67.8 (28.5)	64.5 (22.5)	66.8 (29.2)	64.9 (11.4)	63.8 (10.5)	62.7 (13.5)
LPM	55.1 (45.5)	55.4 (33.6)	55.7 (39.8)	54.9 (24.4)	57.1 (19.2)	55.0 (20.1)
MLM	58.9 (38.2)	58.1 (31.6)	58.2 (46.3)	51.4 (29.4)	54.7 (24.5)	51.3 (23.2)
LM	51.2 (57.8)	52.9 (39.2)	50.8 (62.2)	49.4 (31.8)	53.4 (24.6)	49.5 (25.6)

- [9] R. CARRASCO, J. ONCINA, AND J. CALERA, *Stochastic inference of regular tree languages*, in Proceedings of the Fourth ICGI (LNCS, vol.1433), 1998, pp. 187–198.
- [10] G. CONG ET AL., *Discovering frequent substructures from hierarchical semi-structured data*, in Proceedings of the SIAM DM, 2002.
- [11] A. DARWICHE AND N. FRIEDMAN, eds., *Proceedings of the 18th Conference on UAI*, 2002.
- [12] A. DEMPSTER, N. LAIRD, AND D. RUBIN, *Maximum likelihood from incomplete data via the EM algorithm*, *J. R. Statist. Soc. B*, 39 (1977), pp. 1–38.
- [13] M. DILIGENTI, P. FRASCONI, AND M. GORI, *Hidden tree Markov models for document image classification*, *IEEE Trans. on PAMI*, (2003), pp. 519–523.
- [14] R. DURBIN ET AL., *Biological Sequence Analysis*, Cambridge University Press, 1998.
- [15] D. J. HAND AND R. J. TILL, *A simple generalisation of the area under the roc curve for multiple class classification problems*, *Machine Learning*, 45 (2001), pp. 171–186.
- [16] C. HYEOKHO AND R. BARANIUK, *Multiscale image segmentation using wavelet-domain hidden Markov models*, *IEEE Trans. on Image Proc.*, 46 (2001), pp. 886–902.
- [17] M. JORDAN, Z. GHARAMANI, AND L. SAUL, *Hidden Markov decision trees*, in Proceedings of NIPS 8, 1996, pp. 501–507.
- [18] M. KANEHISA ET AL., *The KEGG databases at GenomeNet*, *Nucl. Acids Res.*, 30 (2002), pp. 42–46.
- [19] H. KASHIMA AND T. KOYANAGI, *Kernels for semi-structured data*, in Proceedings of the Nineteenth ICML, 2002, pp. 291–298.
- [20] C. MANNIG AND H. SCHUTZE, *Foundations of Statistical Natural Language Processing*, MIT Press, 1999.
- [21] J. PEARL, *Probabilistic Inference in Intelligent Systems*, Morgan Kaufmann, 1988.
- [22] L. RABINER AND S. JUANG, *Fundamentals of Speech Recognition*, Prentice Hall, NJ, USA, 1993.
- [23] Y. S. S. FINE AND N. TISHBY, *The hierarchical hidden markov model: Analysis and applications*, *Machine Learning*, 32 (1998), pp. 41–62.
- [24] A. TERMIER, M. C. ROUSSET, AND M. SEBAG, *Treefinder: a first step towards XML data mining*, in Proceedings of the IEEE ICDM, 2002, pp. 450–457.
- [25] M. ZAKI AND C. AGGARWAL, *Xrules: An effective structural classifier for XML data*, in Proceedings of the Ninth ACM KDD, 2003, pp. 316–325.

Appendix

Estimating Parameters of MLM

Initialization stage: Compute $n[u, \sigma_h]$, the number of appearances of label σ_h in tree u .

Iterative stage: Initialize $v[i]$ and $w[i, c]$ with random values, and repeat the following two steps alternately until a certain convergence criterion is satisfied.

E-step: Compute $\theta_{c|u}$ as follows:

$$\theta_{c|u} = \frac{\theta_{u|c} v[c]}{\sum_{c'} \theta_{u|c'} v[c']},$$

where $\theta_{u|c}$ is calculated as follows: $\theta_{u|c} = \prod_k w[c, o_k^u]$.

M-step: Update $w[c, i]$ and $v[c]$ as follows:

$$v[c] = \frac{\sum_u \theta_{c|u}^{old}}{\sum_{c'} \sum_{u'} \theta_{c'|u'}^{old}},$$

$$w[c, \sigma_h] = \frac{\sum_u n[u, \sigma_h] \cdot \theta_{c|u}^{old}}{\sum_{h'} \sum_{u'} n[u', \sigma_{h'}] \cdot \theta_{c|u'}^{old}}.$$

Estimating Parameters of MLPM

Initialization stage: Compute $n[u, \sigma_h, \sigma_{h'}]$, the number of times label σ_h and $\sigma_{h'}$ appear at a node and its parent, respectively, in tree u , and $r[u, \sigma_h]$, the number of appearances of label σ_h at the root of tree u .

Iterative stage: Start with random initial probability values for $w[c, \sigma_h, \sigma_{h'}]$, $\pi[c, \sigma_h]$ and $v[c]$, and repeat the following two steps alternately until a certain convergence criterion is satisfied.

E-step: Compute $\theta_{c|u}$ as follows:

$$\theta_{c|u} = \frac{\theta_{u|c} v[c]}{\sum_{c'} \theta_{u|c'} v[c']},$$

where $\theta_{u|c}$ is calculated as follows: $\theta_{u|c} = \pi[c, o_1^u] \prod_{j (\neq 1)} w[c, o_j^u, o_p^u]$.

M-step: Compute $w[c, \sigma_h, \sigma_{h'}]$, $\pi[c, \sigma_h]$ and $v[c]$ as follows:

$$v[c] = \frac{\sum_u \theta_{c|u}^{old}}{\sum_{c'} \sum_{u'} \theta_{c'|u'}^{old}},$$

$$\pi[c, \sigma_h] = \frac{\sum_u r[u, \sigma_h] \cdot \theta_{c|u}^{old}}{\sum_{h'} \sum_{u'} r[u', \sigma_{h'}] \cdot \theta_{c|u'}^{old}},$$

$$w[c, \sigma_h, \sigma_{h'}] = \frac{\sum_u n[u, \sigma_h, \sigma_{h'}] \cdot \theta_{c|u}^{old}}{\sum_h \sum_{u'} n[u', \sigma_h, \sigma_{h'}] \cdot \theta_{c|u'}^{old}}.$$