Managing and Analyzing Carbohydrate Data *

Kiyoko F. Aoki[†], Nobuhisa Ueda[†], Atsuko Yamaguchi[†],
Tatsuya Akutsu[†], Minoru Kanehisa[†], and Hiroshi Mamitsuka[†‡]

Abstract

One of the most vital molecules in multicellular organisms is the carbohydrate, as it is structurally important in the construction of such organisms. In fact, all cells in nature carry carbohydrate sugar chains, or glycans, that help modulate various cell-cell events for the development of the organism. Unfortunately, informatics research on glycans has been slow in comparison to DNA and proteins, largely due to difficulties in the biological analysis of glycan structures. Our work consists of data engineering approaches in order to glean some understanding of the current glycan data that is publicly available. In particular, by modeling glycans as labeled unordered trees, we have implemented a tree-matching algorithm for measuring tree similarity. Our algorithm utilizes proven efficient methodologies in computer science that has been extended and developed for glycan data. Moreover, since glycans are recognized by various agents in multicellular organisms, in order to capture the patterns that might be recognized, we needed to somehow capture the dependencies that seem to range beyond the directly connected nodes in a tree. Therefore, by defining glycans as labeled ordered trees, we were able to develop a new probabilistic tree model such that sibling patterns across a tree could be mined. We provide promising results from our methodologies that could prove useful for the future of glycome informatics.

1 Introduction

Carbohydrate sugar chains, or glycans, have only recently drawn some attention from those in informatics/computer science. Glycans are involved in the field of glycobiology, where the study of the structure, biosynthesis and biology of glycans take place, the majority of which are located on the outer surface of cellular macromolecules. They assist in crucial activities for the development and function of complex, multicellular organisms, including serving as regulatory switches [6, 24]. The understanding of glycans, however, is far from complete. This is mainly due to their complexity in that they cannot be as easily studied as genes and proteins. For example, they cannot be amplified as nucleic acids, they are non-linear structures, and their behaviors vary relative to their environment (i.e., in vivo vs. in vitro) [18].

From an informatics perspective, a closer look at glycans reveals that their structures can easily be modeled as labeled trees. As such, we have developed some data engineering approaches to the analysis of the tree structures of glycans. In this paper, we introduce a tree matching algorithm for performing structural queries on glycans, called KEGG Carbohydrate Matcher (KCaM) [1], and a probabilistic sibling tree Markov model (PSTMM) for the mining and prediction of glycan tree structures [22, 2].

The release of a new database for glycan structures prompted the development of our tree matching algorithm for glycan structures. We looked into the various algorithms available for labeled unordered tree matching, and we were able to develop our algorithm based on methodologies known to be efficient and accurate [11].

Another important issue was the mining of frequent patterns in glycan tree structures. Considering that these structures in general are in actuality labeled ordered tree structures, we could consider various mining approaches [3, 16, 25]. However, one of the drawbacks of using any of these approaches for glycans came from the fact that unlike XML, glycan data is noisy. That is, due to the newness of glycan informatics research in general, not all of the data has been completely verified, and we can expect many errors in our data. Thus, a probabilistic approach would be more suited to the data at hand, and so we developed a probabilistic model for mining frequent patterns in labeled ordered trees as well as a time-efficient learning algorithm for estimating the parameters of our model.

We note here that our two applications presented in this paper consider the glycan tree structures differently. The matching algorithm assumes that its input trees are labeled unordered trees, and the data mining model explicitly considers labeled ordered trees. Although the matching algorithm can easily be applied to labeled ordered trees, because of (1) the lack of data where many structures in the database may not have order information and (2) the level of importance of this ordering is not yet certain, we have implemented KCaM for the more difficult case of using labeled unordered trees. In contrast, in the case of mining for frequent patterns in glycans, we have successfully tackled the more difficult case of using labeled ordered trees. We show thus in this paper that glycans currently cannot be categorized as being ordered or unordered; they have both characteristics, and we have applied our methodologies accordingly.

2 Background

2.1 Carbohydrate sugar chains The difficulty in working with glycans in terms of informatics is that their structures are not simple like sequences; glycans are branched tree structures with various types of linkages. The basic component of glycans is the monosaccharide unit, or sugar, of which a handful are most common in higher animal oligosaccharides (see Table 1). Each sugar is linked to different hydroxyl groups on one or more other sugars by either an α or β type linkage (i.e., anomer). Classes of glycans

*This work is 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, Kyoto, 611-0011, Japan

[‡]Contact author: mami@kuicr.kyoto-u.ac.jp

are defined according to core sub-structures which consistently exist in glycans within the same class. Two major glycan classes are N- and O-glycans. The boxed glycan in Figure 2 is an example of an O-glycan found in both human gastric mucin as well as swine trachea mucin. In this example, the sugar from which this structure starts is the GalNAc at the far right, and the rest of the structure extends from this sugar; one Gal and one GlcNAc is each beta-linked to the third and sixth hydroxyl group, respectively, of this GalNAc. Successive linkages extend from these two sugars.

Table 1: Common monosaccharide names, abbreviations and symbols.

Monosaccharide name	Abbr.	Sym.
Glucose	Glc	▲
Galactose	Gal	●
Mannose	Man	○
N-acetyl neuraminic / sialic acid	NeuNAc	◆
N-acetylglucosamine	GlcNAc	■
N-acetylgalactosamine	GalNAc	□
Fucose	Fuc	△
Xylose	Xyl	▽
Glucuronic acid	GlcA	◈
Iduronic acid	IdA	◈

2.2 KEGG GLYCAN The KEGG GLYCAN database is the newest addition to the KEGG (Kyoto Encyclopedia of Genes and Genomes) bioinformatics resource. Not only are glycan structures maintained, but corresponding annotation information are linked to KEGG's various other genomic data such as pathway and enzymatic reaction information as well as literary databases such as PubMed [15].

3 Methodologies

Our first project involving KEGG GLYCAN was the development of a fast and efficient method for querying this data. Considering that it would be inappropriate to require the user to specify every hydroxyl group and linkage type of a glycan structure as a query, not to mention the fact that biologically speaking, many of these details are currently unknown for many structures, we characterize glycans as labeled unordered tree structures. This led us to the development of a *labeled unordered tree matching algorithm* that is known to be efficient based on proven results from theoretical computer science [11]. We call this algorithm KCaM, for KEGG Carbohydrate Matcher.

Another project developed based on KEGG GLYCAN data was a data mining approach for finding frequent patterns in glycans. Biologically, it has been shown in the literature that lectins recognize glycans via certain monosaccharide configurations (patterns) on the outer-most portion of their tree structures; it seems that recognition can be affected by specific structural variations and modifications of certain monosaccharides, their linkage to the underlying sugar chain, and the structure of these chains [23]. Not only would an understanding of structural patterns in glycans be used to further support studies in sugar recognition, but such work may be helpful in unraveling their biological functions [5, 9]. As such, we could also characterize glycans as labeled ordered trees, thus prompting the development of a *sibling dependent* probabilistic model for labeled ordered trees. This new model, called PSTMM for probabilistic sibling dependent tree Markov model, has resulted in some interesting and promising findings in glycobiology.

3.1 Terminology and Notation We first provide some preliminary definitions in this section before proceeding to describe our methodologies.

A *tree* is defined as an acyclic connected graph, whose vertices we refer to as *nodes*. A *rooted tree* is a tree having a specific node called the root, from which the rest of the tree extends. Any node x on a unique path from the root to a node y is called an *ancestor* of y, in which case y is a *descendant* of x. Nodes that extend from a node x by one edge are called the *children* of x, and conversely, x would be called the *parent* of these children. Nodes that have the same parent are *siblings*, and a node with no children is a *leaf*. A *subtree* of tree T is a tree whose nodes and edges are subsets of those of T, an *ordered tree* is a rooted tree in which the children of each node are ordered, and a *labeled tree* is a tree in which a label is attached to each node. Also a *common subtree* of a set of trees is a subtree that can be found in every tree in the set. A *forest* is a set of trees. Note that all trees in this paper are labeled and rooted trees.

We will also use the following notation. Let $\mathbf{T} = \{T_1, \ldots, T_{|\mathbf{T}|}\}$ be a set of labeled trees, where $T_u = (V_u, E_u)$ and V_u $(=\{x_1^u, \ldots, x_{|V_u|}^u\})$ and E_u are a set of nodes and a set of edges, respectively. Note that $\mathbf{T}$ can be a set of labeled unordered trees or a set of labeled ordered trees. 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 tree $t_* = (v_*, e_*)$ denote a common subtree of $\mathbf{T}$, and let $\mathbf{t}$ be a set of all possible t_* of $\mathbf{T}$. Let $C_u(p) \subseteq \{1, \ldots, |C_u(p)|\}$ be a set of children of x_p^u in a labeled unordered tree T_u, and let it denote a set of indices of children of x_p^u in a labeled ordered tree 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, respectively, of node p. Each node x_j^u has label $o_j^u \in \Sigma$, where $\Sigma = \{\sigma_1, \ldots, \sigma_{|\Sigma|}\}$ is a set of labels on nodes. For simplicity, we will often use node j instead of x_j^u, if understood from the context. For a node j in a labeled ordered tree, we will often indicate the parent, immediately elder and younger siblings as p, i and k, respectively.

3.2 Database Queries with KCaM The maintenance of a database usually includes the ability to query the data. With the development of KEGG GLYCAN, however, the tree structure of glycans disallowed the use of straightforward matching or align-

ment algorithms as are often used for linear, sequence databases. Therefore, we turned to computer science, where work on tree similarity began in the 70s [21] and was later applied to bioinformatics [17].

We also note the difference in scoring measures between the fields of bioinformatics and computer science, where the former often uses higher "similarity scores," and the latter lower "distance measures" to indicate better matches between two structures. As we are involving computer science techniques, it was necessary for us to take such differences in scoring techniques into consideration.

The problem of querying glycan tree structures can be likened to finding the maximum common subtree of two trees, which is defined as the following.

Input: $\mathbf{T}$ where $|\mathbf{T}| = 2$.

Output: t_* such that $\forall t_i \in \mathbf{t}, |v_*| \geq |v_i|$.

Since this problem has been solved in polynomial time in [11], we combined it with the dynamic programming approach used for sequence alignment [20]. With the objective of finding an optimal match score for two labeled unordered trees, we also incorporated an appropriate scoring mechanism, thus resulting in an accurate and efficient glycan tree matching algorithm called KCaM.

We developed two variations of KCaM; one is an approximate matching approach which compares nodes with nodes and allows gaps in the alignment, and the other is an exact matching approach where edges are compared with edges and gaps are not allowed. As both approaches are derived from sequence alignment procedures, each can be performed locally or globally.

The approximate matching algorithm can be used for example when sub-structures separate from one another are being queried. In this case, we can enter the query with a single tree containing two target substructures, and the gapped alignment will return all similar trees, some with unmatched portions in the intermediate section. However, it may be the case that the user is only looking for a single specific connected component with known anomer and linkage information (i.e., a match with no gaps), in which case the exact matching algorithm would be more appropriate.

Approximate Matching The global dynamic programming procedure for finding the maximum common subtree of two trees T_1 and T_2 calculates a similarity score $Q[x_p^1, x_q^2]$ for the subtrees rooted at every possible pair of nodes x_p^1 and x_q^2. For simplicity, we refer to x_p^1 and x_q^2 as p and q, respectively. Denoting $d(x_p^u)$ as the cost for deleting x_p^u (i.e. introducing a gap), $w(p,q)$ as the similarity[1] between p and q, and $Y(p,q)$ as the set of one-to-one mappings from $C_1(p)$ to $C_2(q)$ (assuming without loss of generality that $|C_1(p)| \leq |C_2(q)|$), we calculate the following: $Q[p,0] = \sum_{i \in t_1(p)} d(i)$ and $Q[0,q] = \sum_{k \in t_2(q)} d(k)$, the scores for matching node p (resp. q) in T_1 (resp. T_2) with no nodes in T_2 (resp. T_1), in addition to

$$Q[p,q] = \max \begin{cases} \max_{k \in C_2(q)} \left\{ \begin{matrix} Q[p,k] + d(q) + \\ \sum_{j \in C_2(q) \setminus \{k\}} Q[0,j] \end{matrix} \right\}, \\ \max_{i \in C_1(p)} \left\{ \begin{matrix} Q[i,q] + d(p) + \\ \sum_{j \in C_1(p) \setminus \{i\}} Q[j,0] \end{matrix} \right\}, \\ w(p,q) + \\ \quad \max_{y \in Y(p,q)} \\ \qquad \left\{ \begin{matrix} \sum_{i \in C_1(p)} Q[i, y(i)] + \\ \sum_{k \in C_2(q) \setminus y(C_1(p))} Q[0,k] \end{matrix} \right\}. \end{cases}$$

Thus the score for the global optimal matching $Q[|T_1|, |T_2|]$, and the resulting maximum common subtree correlated with this score can be retrieved by backtracking to find the matching nodes that contributed to this score.

As in the global approximate matching procedure, the local procedure is similarly derived from the local sequence alignment procedure. The interested reader may refer to [1] for details.

Exact Matching Local exact matching adapts the dynamic programming procedure for local approximate matching by letting p and q represent edges, and letting $w(p,q)$ equal one (1) for a match and zero (0) otherwise. As the exact matching algorithm does not allow gaps, the gap penalty for a mismatch is $-\infty$, thus returning the single largest maximum common connected subtree for T_1 and T_2.

The global exact matching algorithm was implemented such that all possible connected matches between T_1 and T_2 could be found; it recursively calls the local exact matching procedure for all subtrees of the two input trees that have not been matched. In other words, after the first largest matching subtree has been found, the resulting, unmatched portions of both trees are then used as the input trees for another round of the local exact matching algorithm. The recursion ends when either no matches are found, or one of the input trees has been matched in its entirety to the other. Accordingly, the score returned is the summation of the scores for each recursive call to the local exact matching procedure, and the set of matched subtrees returned make up the forest of matching subtrees.

3.3 Mining with Probabilistic Sibling Tree Markov Model (PSTMM) The problem of mining for frequent patterns in glycan data is best solved by a probabilistic model considering the noisiness of the data set. A hidden Markov model [4] (HMM) is often used for learning sequences of data, such as in speech recognition [19] and biological sequence analysis [10]. HMM has been extended for mining labeled trees [8, 14] by capturing statistical dependencies between each vertex and its parent; this model is known as PTMM, or probabilistic Markov tree model. A hierarchical HMM has also been developed for sequences [13] which appear to be similar to our model, but is slightly different as it is used for sequence analysis. Thus, to take sibling relationships directly into account in labeled ordered trees based on the inher-

[1] Due to space limitations, details on the calculation of the similarity function $w(p,q)$ can be found in [1].

ent sibling-dependent structure of glycans, we were prompted to develop our probabilistic sibling tree Markov model, or PSTMM.

Our model consists of *states*, which correspond to nodes in a tree, and *labels*, each of which are attached to each node and are generated depending on the state of that node. We thus define $S = \{s_1, \ldots, s_{|S|}\}$ as a set of states, and $z_j^u \in S$ as a state for node j in a tree. State z_j^u then generates label o_j^u, and θ is our set of parameters.

The dependencies of PSTMM are modeled by setting dependencies between the state of a node and that of both its parent and its immediately elder sibling. This results in a drastic improvement in the performance for finding patterns in given labeled ordered trees. Just as HMMs can capture distant or long-range dependencies in a sequence indirectly when such state transitions are defined, PSTMM extends the range of the dependencies that PTMM can capture through the sibling-sibling relationships that consequently extend indirectly down and across the tree. Figure 1 illustrates the dependencies embedded within a tree in PSTMM.

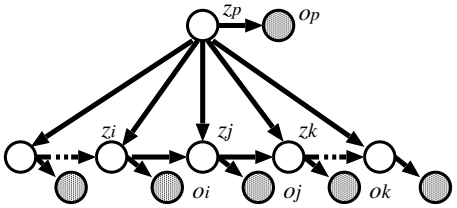

Figure 1: Dependencies in PSTMM. The white nodes are states, and shaded nodes are labels.

PSTMM has three 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 $a[\{s_q, s_l\}, s_m]$ $(= P(z_j^u = s_m | z_p^u = s_q, z_i^u = s_l; \theta))$ is 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. 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 s_l is the state of this node[2].

To describe the probabilistic structure of PSTMM, we define forward, backward and upward probabilities. The forward probability $F_u(s_q, s_l, x_j^u)$ is the probability that for node j, all labels of the subtrees for each of the elder siblings are generated, the state of j is s_l, and the state of p is s_q. The backward probability $B_u(s_q, s_m, x_j^u)$ is the probability that for j, all labels of the subtrees of each of the younger siblings and j are generated, the state of j is s_m, and the state of p is s_q. 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. These three probabilities[3] are formulated as follows:

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

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

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

We can compute U, B and F from the leaves to the root using a bottom-up dynamic programming procedure. The likelihood for a given tree, $L(T_u; \theta) = \sum_l \pi[l]\ U_u(l,1)$, is obtained by using $U_u(l,1)$, U at the root of the tree, and the likelihood for a given set of trees, $L(\mathbf{T}; \theta) = \prod_u \sum_l \pi[l]\, U_u(l,1)$, is computed as a product of $L(T_u; \theta)$ for all $T_u \in \mathbf{T}$.

Estimating Parameters of PSTMM For estimating the probability parameters of PSTMM, we use the maximum likelihood (ML) estimators, in which parameters are estimated to maximize the likelihood above for a given set of trees, and an EM (Expectation-Maximization) algorithm [7] to obtain the ML estimators.

In the EM procedure, in addition to F, B and U, we define a downward probability $D_u(s_l, x_j^u)$, the probability that all labels of a tree except for those of subtree $t_u(j)$ are generated and that for node x_j^u, s_l is the state[4]. 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[5]:

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

In calculating the values for all four probabilities using dynamic programming, U, B and F are calculated as before, and D is calculated top-down from the root back to the leaves. Now using these four

[2] Hereafter, we use $\pi[l]$, $a[\{q,l\},m]$ and $b[l,\sigma_h]$ for $\pi[s_l]$, $a[\{s_q,s_l\},s_m]$ and $b[s_l,\sigma_h]$, respectively. Also note that $\sum_m a[\{s_q,s_l\},s_m] = 1$.

[3] We 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.

[4] For simplicity, we hereafter use $D_u(l,j)$ for $D_u(s_l,x_j^u)$.

[5] We note that the likelihood for a tree can be calculated using the upward and downward probabilities at any node i: $L(T_u;\theta) = \sum_l\ U_u(l,i)\ D_u(l,i)$.

probability parameters, we compute $\gamma_u(\{s_q, s_m\}, s_l)$, 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. Denoting $\gamma_u(\{s_q, s_m\}, s_l)$ by $\gamma_u(\{q, m\}, l)$ for simplicity, we can use $\gamma_u(\{q, m\}, l)$ to recursively update the state transition probability

$$a[\{q, m\}, l] = \frac{\sum_u \gamma_u(\{q, m\}, l)}{\sum_u \sum_{l'} \gamma_u(\{q, m\}, l')}.$$

We can similarly recursively update the values for $b[s_l, \sigma_h]$ and $\pi[s_l]$ using their respective expectation values based on an EM procedure. Please refer to [22, 2] for details.

3.4 Computation Time Because the degree of each vertex is bounded (i.e., the maximum number of hydroxyl groups for most sugars is constant), the running times for all algorithms except the global exact matching algorithm are on the order of $O(|V_1| \cdot |V_2|)$. The global exact matching algorithm, runs in $O(|V_1| \cdot |V_2| \cdot min(|V_1|, |V_2|))$ due to the recursion of the local exact matching algorithm.

Out of all the equations for calculating the expectation values and updating the parameters in our learning algorithm, the most time-consuming is calculating γ, whose computation time 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.

4 Results

Using the boxed glycan structure in Figure 2 as our query, we ran the global approximate matching algorithm, resulting in a listing of 91 entries, with the maximum score being 1500 (matching itself). Figure 2 also lists a sampling of three glycans returned from this query. The second most similar structure (a) scored 1300 as it contains one less sugar. An O-Glycan (b) and a glycan in the Sphingolipid class (c) were returned with scores of 675 and 200, respectively. Although the Sphingolipid may look quite similar to the query at first glance, the lower score seems accurate as it belongs to a different class. Investigating this glycan further, we find that it is actually involved with an enzyme that is known to degrade certain intestinal mucin glycosphingolipids [12]. Interestingly, recalling that our query glycan is also found in mucin, we can justify that the difference in class should not exclude it from the result. For other examples, the reader is encouraged to access the KEGG GLYCAN web site, and documentation on the access and usage (i.e., search, query process) of KCaM is currently in preparation for publication.

The performance of PSTMM was evaluated by comparing it with those of four other simpler probabilistic models: label model (LM), mixture label model (MLM), label pair model (LPM), and mixture label pair model (MLPM). LM counts the labels that appear in the given set of labeled ordered trees, and parameter estimation consists of one calculation of the probability that some label is output at some node from among the entire input tree set. MLM is a mixture of LMs, where each LM is called a *component*. Each component is applied a different probability parameter value which is then estimated and updated recursively by an EM procedure. LPM is similar to LM except that it calculates the probability that a label σ_h is outputted at a node given that label $\sigma_{h'}$ is outputted at its parent node. It also has an additional parameter for the probability that the root outputs a certain label. As in LM, parameter estimation consists of one calculation of these two parameters. Finally, MLPM is a mixture of LPMs where each LPM is a component with a different probability parameter which is estimated and updated recursively by EM.

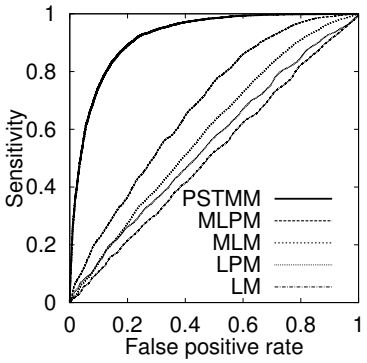

Figure 3: The ROC curves for N-Glycan.

We evaluated our model using both synthetic and real biological data but for space concerns we only present our results on glycan data here. We created data sets of glycans from KEGG GLYCAN based on two major classes, N- and O-Glycans, selecting from each those structures that contained at least one sibling pair. We performed a five-fold cross-validation for each data set by dividing them into five blocks of roughly equal size, and in each of five trials, selecting a different block as the test set while training with the rest. We repeated this process five times. The results were then averaged over the 25 ($= 5 \times 5$) runs. Figure 3 shows the ROC curves of the five methods tested for N-Glycan. We plotted *sensitivity* vs. *false positive rate*. Sensitivity is the ratio of the number of correctly predicted examples to the total number of positive examples, and false positive rate is the ratio of the number of false positives to the total number of negative examples. For a certain false positive rate, the higher the sensitivity, the better the performance. Our plot clearly indicates that PSTMM outperformed the other four methods by a large margin. Our results imply that some complex pattern exists in glycans that is not limited to parent-child relationships.

5 Discussion

We have attempted to tackle glycans from an informatics perspective by characterizing their structures as labeled trees. Further characterization requires a definition of the objective of the analysis. In our case,

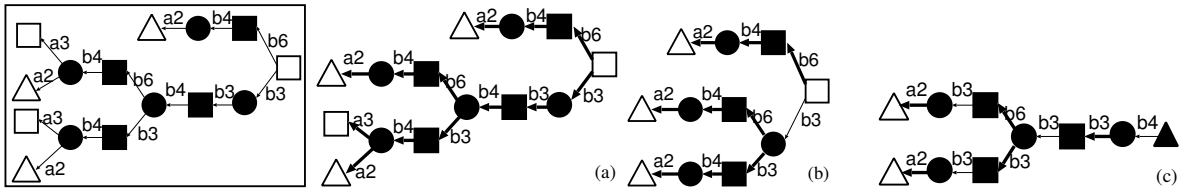

Figure 2: Examples of glycans resulting from running KCaM using the boxed glycan. The matching substructures are emphasized in bold linkages. Score for (a) 1300, (b) 675, and (c) 200.

we defined glycans differently in our algorithm for matching glycans and in our probabilistic model for mining glycans. In order to accomodate some leeway in matching glycan tree structures, we defined our input as labeled unordered trees. On the other hand, the recognition of glycans by lectins and other biological agents led us to model sibling relationships in PSTMM, which thus takes labeled ordered trees as input. As a result, our work has shown promise in two very different and yet meaningful areas of research in the analysis of glycans. In our results, we illustrated how the similarity scores corresponded to actual structures queried by KCaM. We also statistically demonstrated the effectiveness of PSTMM. It is exciting to see these promising results as they represent preliminary proof of concept of our models.

In conclusion, our work represents groundwork for many other possible future projects. Currently, statistical analysis of KCaM, enabled by our similarity scores, is underway. From this, it will be possible to statistically calculate various aspects of glycan structures. For example, score matrices can be developed, relationships between various sugars can be identified, and new classes may be discovered. PSTMM can also be used not only for mining, but for prediction as well. Such work shall surely benefit glycobiologists and further encourage the progress of this new field called glycome informatics.

References

[1] K. F. Aoki et al., *Efficient tree-matching methods for accurate carbohydrate database queries*, Genome Informatics, 14 (2003), pp. 134–143.

[2] ——, *Application of a new probabilistic model for recognizing complex patterns in glycans*, in ISMB, 2004.

[3] T. Asai et al., *Online algorithms for mining semi-structured data stream*, in 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] C. R. Bertozzi and L. L. Kiessling, *Carbohydrates and glycobiology review: Chemical glycobiology*, Science, 291 (2001), pp. 2357–2364.

[6] S. A. Brooks et al., *Functional and Molecular Glycobiology*, BIOS Scientific Publishers Ltd., 2002.

[7] 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.

[8] M. Diligenti et al., *Hidden tree Markov models for document image classification*, IEEE Trans. PAMI, 25 (2003), pp. 519–523.

[9] K. Drickamer, *Two distinct classes of carbohydrate-recognition domains in animal lectins*, J. Biol. Chem., 263 (1988), pp. 9557–9560.

[10] R. Durbin et al., *Biological sequence analysis*, Cambridge University Press, Cambridge, 1998.

[11] J. Edmonds and D. Matula, *An algorithm for subtree identification*, SIAM Rev., 10 (1968), pp. 273–274.

[12] P. Falk, L. C. Hoskins, and G. Larson, *Bacteria of the human intestinal microbiota produce glycosidases specific for lacto-series glycosphingolipids*, J. Biochem, 108 (1990), pp. 466–474.

[13] S. Fine, Y. Singer, and N. Tishby, *The hierarchical hidden Markov model: Analysis and applications*, Machine Learning, 32 (1998), pp. 41–62.

[14] C. Hyeokho and R. Baraniuk, *Multiscale image segmentation using wavelet-domain hidden Markov models*, IEEE Trans. Image Proc., 46 (2001), pp. 886–902.

[15] M. Kanehisa et al., *The KEGG resource for deciphering the genome*, NAR, 32 (2004), pp. D277–D280.

[16] H. Kashima and T. Koyanagi, *Kernels for semi-structured data*, in ICML, 2002, pp. 291–298.

[17] G. Lin et al., *Edit distance between two RNA structures*, in RECOMB, 2001, pp. 211–220.

[18] I. Marchal, G. Golfier, O. Dugas, and M. Maied., *Bioinformatics in glycobiology*, Biochimie, 85 (2003), pp. 75–81.

[19] L. Rabiner and S. Juang, *Fundamentals of Speech Recognition*, Prentice Hall, NJ, USA, 1993.

[20] T. F. Smith and M. S. Waterman, *Identification of common molecular subsequences*, J. Mol. Biol., 147 (1981), pp. 195–197.

[21] K. Tai, *The tree-to-tree correction problem*, Journal of the ACM, 26 (1979), pp. 422–433.

[22] N. Ueda, K. F. Aoki, and H. Mamitsuka, *A general probabilistic framework for mining labeled ordered trees*, in SIAM DM, 2004.

[23] A. Varki, *Sialic acids as ligands in recognition phenomena*, FASEB J., 11 (1997), pp. 248–255.

[24] A. Varki et al., eds., *Essentials of Glycobiology*, Cold Spring Harbor Lab. Press, New York, 1999.

[25] M. Zaki and C. Aggarwal, *Xrules: An effective structural classifier for XML data*, in KDD, 2003, pp. 316–325.