An Efficient Unordered Tree Kernel and Its Application to Glycan Classification*

Tetsuji Kuboyama[1], Kouichi Hirata[2], and Kiyoko F. Aoki-Kinoshita[3]

[1] Center for Collaborative Research, University of Tokyo
4-6-1 Komaba, Meguro, Tokyo 153-8505, Japan
kuboyama@ccr.u-tokyo.ac.jp
[2] Department of Artificial Intelligence, Kyushu Institute of Technology
680-4 Kawazu, Iizuka 820-8502, Japan
hirata@ai.kyutech.ac.jp
[3] Faculty of Engineering, Soka University
1-236 Tangi-cho, Hachioji, Tokyo 192-8577, Japan
kkiyoko@t.soka.ac.jp

Abstract. The problem of computing *unordered tree kernels* based on exhaustive counts of subtrees has known to be #P-complete. In this paper, we develop an efficient and general unordered tree kernel based on *bifoliate q-grams* that are unordered trees with at most two leaves and just q nodes. First, we introduce a *bifoliate q-gram profile* as a sequence of the frequencies of all bifoliate q-grams embedded into a given tree. Then, we formulate a *bifoliate tree kernel* as an inner product of bifoliate q-gram profiles of two trees. Next, we design an efficient algorithm for computing the bifoliate tree kernel. Finally, we apply the bifoliate tree kernel to classifying glycan structures.

1 Introduction

A *rooted labeled tree* is a fairly general data structure that models a wide variety of hierarchical data including parse trees for natural language texts, semi-structured data such as HTML/XML, and biological data such as RNA secondary structures and glycans.

In this paper, we concentrate on a binary classification problem based on kernel methods with support vector machines (SVMs). Let X be the input space (e.g. a set of rooted labeled unordered trees in this paper), and $Y = \{+1, -1\}$ be the output domain. A training set is a finite set of training data, denoted by $D = \{(x_1, y_1), \ldots, (x_m, y_m)\} \subseteq X \times Y$. The purpose of the learning procedure in SVMs is to give a decision function $f_d(\cdot)$ from a training set D. The learning procedure outputs a decision function $f_d : X \to Y$ so that $y_i = f_d(x_i)$ approximates the probabilistic relation between inputs and outputs.

A number of tree structure classification problems have been successfully addressed by kernel methods with SVMs in the past decade. In order to apply kernel

* This work is partly supported by Grant-in-Aid for Scientific Research No. 17700138 from the Ministry of Education, Culture, Sports, Science and Technology, Japan.

T. Washio et al. (Eds.): PAKDD 2008, LNAI 5012, pp. 184–195, 2008.

methods to a specific domain, the most important task is to design similarity functions, so called *kernel functions*, between two objects. One of the earliest work on tree kernels was by Collins and Duffy [4], who presented a *parse tree kernel* as a counting function of common subtrees between two parse trees. Inspired by the parse tree kernel, Kashima and Koyanagi [9] extended it to general rooted labeled ordered trees and proposed a quadratic-time algorithm. These kernels employ the convolution kernel [5] as their design framework by counting all the common subtrees between two trees.

On the other hand, in our previous work [12,13,14,15,16], we introduced an *ordered tree q-gram* as a rooted ordered labeled tree isomorphic to a path. We further proposed a *spectrum tree kernel* [14] and a *gram distribution kernel* [13] based on the frequencies of all common q-grams embedded in a given tree, which are more efficient and representative than the tree kernels by Kashima and Koyanagi [9].

In contrast to ordered trees, Kashima, Sakamoto, and Koyanagi [10] recently showed that their approach to design kernel functions inherently for *unordered trees*, in which the order of sibling nodes is arbitrary, leads to #P-completeness. It is also known that the problem of computing the similarity of trees based on *tree edit distance* [20] and *alignment of trees* [7] is intractable. On the other hand, Vishwanathan first presented a fast kernel for unordered trees [17] based on a string kernel using suffix trees. Kailing *et al.* also proposed a tractable algorithm for computing the structural dissimilarity between unordered trees [8]. The effectiveness of these methods, however, has yet to be proven.

In this paper, we aim at developing an expressive and efficient kernel for *rooted labeled unordered trees* by circumventing the issues in the previous work. In fact, our kernel counts all the common subtrees with q nodes and at most two leaves, as an extension of all the common paths with q nodes (q-grams) [12,13,14,15,16] and restricting to all the common subtrees between two trees [9]. We call such a subtree a *bifoliate q-gram*.

Our contributions are as follows: (1) we introduce a *bifoliate q-gram profile* as a sequence of the frequencies of all bifoliate q-grams embedded in a given tree; (2) we design an efficient algorithm for computing a *bifoliate tree kernel* as an inner product of the bifoliate q-gram profiles of two trees; (3) we apply the bifoliate tree kernel to classifying glycan structures in bioinformatics and compare the performance of the bifoliate tree kernel with the kernel based on the structural similarity of unordered trees proposed by Kailing *et al.* [8].

This paper is organized as follows: in Section 2, we introduce a *bifoliate q-gram* and a *bifoliate q-gram profile*. We also formulate the *bifoliate tree kernel* of two given trees as an inner product of their bifoliate q-gram profiles. In Section 3, we design an efficient algorithm *Bifoliate_Profile* to compute a bifoliate q-gram profile of a given tree, which runs correctly in $O(qd\min(q,d)ln)$ time, where n, d and l are the number of nodes, the depth, and the number of leaves, respectively. This implies that we can also compute bifoliate tree kernels efficiently. In Section 4, we apply the bifoliate tree kernel to classifying glycan structures. Our experimental results illustrate the effectiveness of our kernel. Section 5 concludes the paper by summarizing our contributions.

2 Bifoliate Tree Kernel

We first introduce the basic notions used in this paper. A *tree* is a connected graph without cycles. For a tree $T = (V, E)$, we sometimes denote $v \in T$ instead of $v \in V$, and $|T|$ instead of $|V|$. A *rooted tree* is a tree with one node r chosen as its *root*. For each node v and u in T, let $UP_v(u)$ be the unique path from v to u in T.

For a root r of T, we call the number of edges in $UP_r(v)$ the *depth* of v (in T) and denote it by $dep(v)$. In particular, since $UP_r(r)$ has no edges, we set $dep(r) = 0$. For a tree T, we call $\max\{dep(v) \mid v \in T\}$ the *depth* of T and denote it by $dep(T)$.

The *parent* of $v(\neq r)$ is the node adjacent to v on the path $UP_r(v)$. We say that u is a *child* of v if v is the parent of u. A *leaf* is a node having no children, and a *branch* is a node having just two children. We denote the number of all leaves in T by $lvs(T)$.

A rooted tree is *ordered* if a left-to-right order for the children of each node is given, and it is *unordered* otherwise. A rooted tree $T = (V, E)$ is *labeled* (by an alphabet Σ of labels) if there exists an onto function $l : V \to \Sigma$ such that $l(v) = a$ $(v \in V, a \in \Sigma)$. In the remainder of this paper, we simply call a rooted unordered labeled tree and a rooted ordered labeled tree a *tree* and an *ordered tree*, respectively.

Let T be an ordered tree with the root v and the children $v_1, \ldots, v_m$ of v. The *postorder traversal* (*postorder*, for short) of T is obtained by visiting v_i $(1 \leq i \leq m)$ in order, recursively, and then visiting v.

Let T be an ordered tree with n nodes and suppose that the sequence $v_1 \cdots v_n$ is the postorder of T. Also let $p(v_i)$ be the index j such that v_j is a parent of v_i for every $1 \leq i \leq n-1$. Then, we formulate the *depth sequence* $D(T)$, the *label sequence* $L(T)$ and the *parent sequence* $PS(T)$ of T as follows.

$$D(T) = dep(v_1) \cdots dep(v_n),\ L(T) = l(v_1) \cdots l(v_n),\ PS(T) = p(v_1) \cdots p(v_{n-1}).$$

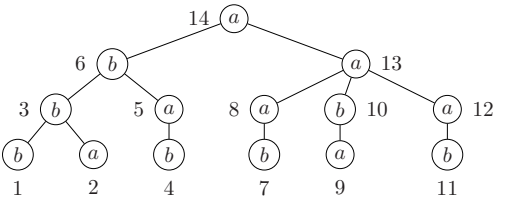
For the depth sequence D of T, we denote $\max\{d \mid d \in D\}$ by $\max D$. It is obvious that $dep(T) = \max D$.

Example 1. Consider the tree T in shown at the top of Figure 1. The depth sequence $D(T)$, the label sequence $L(T)$, and the parent sequence $PS(T)$ of T are given below the tree in the figure.

In this paper, as an extension of tree q-grams [12,13,14,15,16], we introduce the concept of *bifoliate q-grams*. Note that we are here only concerned with their structures. Thus, their labels are omitted.

Definition 1. A *bifoliate q-gram* is a tree with at most two leaves and exactly q nodes, denoted by $P^q_{k,b}$ for $\lfloor q/2 \rfloor \leq k \leq q-2$ and $0 \leq b \leq k-1$ and $P^q_{q-1,0}$, where k is the depth of a leaf located relatively far from the root (hereafter called a *deeper leaf*) and b is the depth of a branch.

Note that the range of the depth b of the branch varies depending on the depth k of the deeper leaf.



i	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$D(T)$	3	3	2	3	2	1	3	2	3	2	3	2	1	0
$L(T)$	b	a	b	b	a	b	b	a	a	b	b	a	a	a
$PS(T)$	3	3	6	5	6	14	8	13	10	13	12	13	14	–

Fig. 1. The tree T and its corresponding depth, label, and parent sequences

Proposition 1. The number of bifoliate q-grams is $\lfloor q/2 \rfloor (q - \lfloor q/2 \rfloor - 1) + 1$.

Proof. Let $p = \lfloor q/2 \rfloor$. Note that the depth k of a deeper leaf varies from p to $q-2$. If $k = q - i$ $(2 \leq i \leq q - p)$, then the number of bifoliate q-grams is $q - 2(i-1)$. Since $i = q - k$, the number of bifoliate q-grams for k $(p \leq k \leq q-2)$ is $2k+2-q$. Hence, the number of bifoliate q-grams is $\sum_{k=p}^{q-2}(2k + 2 - q) + 1$. □

We denote the number $\lfloor q/2 \rfloor (q - \lfloor q/2 \rfloor - 1) + 1$ in Proposition 1 by $\tilde{q}$.

Proposition 2. *Let $\succeq$ be a lexicographic order on depth sequences, where a deeper leaf is regarded as the left-most leaf in ordered trees. Then, the bifoliate q-gram $P^q_{k,b}$ is the $(k(q-k-1) - b + 1)$-th element under $\succeq$.*

Proof. It is obvious that the first element of a bifoliate q-gram under $\succeq$ is $P^q_{q-1,0}$. Let j be an integer such that $2 \leq j \leq \tilde{q}$. By Proposition 1, in the case that $k = q-i$, there exist $q-2(i-1)$ bifoliate q-grams. Since $P^q_{k,b}$ is the $(k-b-(i-2))$-th element from the first element $P^q_{k,k-i+1}$ under $\succeq$ for $k = q - i$, $P^q_{k,b}$ is the $1 + \left\{ \sum_{l=2}^{i-1} (q - 2(l-1)) + k - b - (i-2) \right\}$-th element from $P^q_{q-1,0}$ under $\succeq$. By replacing i with $q - k$, we obtain the statement in Proposition 2. □

Hence, we also denote the j-th bifoliate q-gram under $\succeq$ by Q^q_j $(1 \leq j \leq \tilde{q})$.

Example 2. All of the bifoliate 5-grams with their depth sequences are described in Figure 2. Here, the deeper leaf is set to the left.

For labeled trees, we denote a bifoliate q-gram by a pair $(Q^q_j, L(Q^q_j))$, where Q^q_j is an ordered tree and $L(Q^q_j)$ is its label sequence. It is obvious that $L(Q^q_j) \in \Sigma^q$.

Definition 2 (*cf.* Zhang & Shasha [19]). Let T and P be trees. Then, we say that P *matches* T *at a node* v if there exists a bijection f from the nodes of P into the nodes of T satisfying the following conditions.

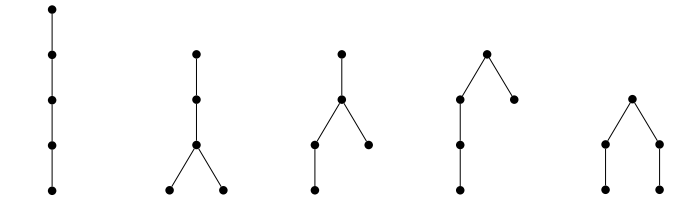


Fig. 2. All of the bifoliate 5-grams

1. f maps the root of P to v.
2. Suppose that f maps x to y and x has children $x_1, \ldots, x_l$. Then, y has children $y_1, \ldots, y_m$ such that $m \geq l$ and there exists an injection $g : \{1, \ldots, l\} \to \{1, \ldots, m\}$ such that $f(x_i) = y_{g(i)}$.
3. $l(x) = l(f(x))$ for each $x \in P$.

Definition 3. Let T be a tree and (Q_j^q, w) be a bifoliate q-gram for $1 \leq j \leq \tilde{q}$ and $w \in \Sigma^q$. Then, we say that (Q_j^q, w) *is embedded into* T if there exists a node v in T such that (Q_j^q, w) matches T at v. Furthermore, we denote the number of (Q_j^q, w) embedded into T by $L_q(T)[Q_j^q, w]$.

We order all of the strings in Σ^q by $w_1, \ldots, w_{|\Sigma|^q}$. For $1 \leq j \leq \tilde{q}$, we denote the sequence $(L_q(T)[Q_j^q, w_1], \ldots, L_q(T)[Q_j^q, w_{|\Sigma|^q}])$ by $L_q(T)[Q_j^q]$.

Definition 4. For a tree T, the following sequence $\mathcal{L}_q(T)$ of the number of every embedded bifoliate q-gram into T is a *bifoliate q-gram profile* of T.

$$\mathcal{L}_q(T) = (L_q(T)[Q_1^q], \ldots, L_q(T)[Q_{\tilde{q}}^q]).$$

We are now ready to formulate the *bifoliate tree kernel* of two trees T_1 and T_2 as an inner product of their bifoliate q-gram profiles as follows.

Definition 5 (Bifoliate Tree Kernel). For rooted labeled unordered trees T_1 and T_2 and a fixed integer $q \geq 2$, the *bifoliate tree kernel* $\mathbf{K}_q(T_1, T_2)$ is defined as

$$\mathbf{K}_q(T_1, T_2) = \langle \mathcal{L}_q(T_1), \mathcal{L}_q(T_2) \rangle.$$

For $q = 1$, we assume that $\mathbf{K}_q(T_1, T_2)$ denotes the inner product of the label frequency vectors of T_1 and T_2.

3 Computing a Bifoliate q-Gram Profile

In this section, we design the algorithm to compute a bifoliate q-gram profile. First, we prepare subroutines as given in Algorithm 1, where D, L and PS denote the depth sequence, the label sequence and the parent sequence, respectively, of an ordered tree.

```
    procedure pseq(D)
        /* D: a depth sequence */
1       T[0] ← |D|;
2       for i = |D| − 1 downto 1 do
3           PS[i] ← T[D[i] − 1]; T[D[i]] ← i;
4       return PS;

    procedure labels(i, k, PS, L)
        /* PS: a parent sequence, L: a label sequence */
5       w ← ε; pt ← i;
6       for m = 1 to k do
7           w ← w · L[pt]; pt ← PS[pt];
8           return (w, pt);

    procedure shift_table(q, D)
        /* shift is assumed to be an empty array */
9       for d = max D − 1 downto 1 do
10          for k = 1 to q − 1 do
11              if d + k ≤ max D then
12                  shift[d] ← shift[d] ∪ {(d + k, k)};
13      return shift;
```

Algorithm 1. Subroutines for computing a bifoliate q-gram profile

The algorithm $pseq(D)$ constructs the parent sequence from a given depth sequence D. The algorithm $labels(i, k, PS, L)$ concatenates the labels from the node indexed by i with length k by selecting nodes and labels according to a parent sequence PS and a label sequence L. "$\cdot$" and ε denote the concatenation of two strings and an empty string, respectively. The algorithm *shift_table* constructs the table *shift* (*cf.*, [12,13,14,15,16]).

Using these subroutines, we can design the algorithm *Bifoliate_Profile* to compute a bifoliate q-gram profile of a given tree described as in Algorithm 2. Here, we use an *ordered q-gram* [12,13,14,15,16], which is an ordered tree with q nodes isomorphic to a path whose depth of the left leaf is k, and we denote it by P_k^q. We also denote $\prec$ as a lexicographic order on Σ^q. Furthermore, the algorithm adopts the table $id[j][k]$ in order to store the indices of the left leaf of P_k^p for some $p < q$. We will show below that $p = D[i] + 2k + 1 - j$ for a current depth $D[i]$.

Example 3. Consider the tree T in Example 1 (Figure 1) and let q be 5. Note first that the result applying the algorithm *shift_table* to the depth sequence $D(T)$ is given in Figure 3.

Figure 4 describes the transition of the table id in the algorithm *Bifoliate_Profile*. Here, the first and second lines are the depth sequence $D(T)$ of T and index i, respectively. The numbers in bold in the i-th column satisfies the condition of line 7 at the $(i + 1)$-th iteration of the main loop.

```
    procedure Bifoliate_Profile(q, D, L)
        /* D: a depth sequence, L: a label sequence */
        /* initialize: Every P[k][b][w] is assumed to be zero. */
        /* initialize: Every id[k][j] is assumed to be empty. */
 1      PS ← pseq(D); shift ← shift_table(q, D);
 2      for i = 1 to |D| do
 3          for j = max D downto 1 do
 4              for k = 1 to min(j, q − 1) do
 5                  p ← D[i] + 2k + 1 − j;
 6                  s ← j − k; /* s : the depth of the root of P_k^p */
 7                  if 2 ≤ p ≤ q and q − p ≤ s then
 8                      foreach c ∈ id[j][k] do
 9                          (w1, _) ← labels(c, k, PS, L);
10                          (w2, pt) ← labels(i, p − k − 1, PS, L);
11                          if (|w1| < |w2|) or (|w1| = |w2| and w1 ≺ w2) then
12                              (w1, w2) ← (w2, w1);
13                          w_r ← L[pt]; pt ← PS[pt]; w ← w1 · w2 · w_r;
                            /* pt : the index of the root of P_k^p */
14                          if j ≠ D[i] + k then
                                /* not P_{p−1}^p */
15                              (w3, _) ← labels(pt, q − p, PS, L); w ← w · w3;
16                              P[|w1| + q − p][q − p][w]++;
17                          else if p = q then P[q − 1][0][w]++;
18          if D[i] < max D then
19              foreach (j, k) ∈ shift[D[i]] do
20                  id[j][k + 1] ← id[j][k + 1] ∪ id[j][k];
21                  id[j][k] ← ∅;
22          id[D[i]][1] ← id[D[i]][1] ∪ {i};
23      return P;
```

Algorithm 2. *Bifoliate_Profile*

Consider the indices 1, 2 and 3 in the third column for index 4. They denote the left leaves of ordered 5-grams whose index of the right leaf is 4.

For index $1 \in id[3][2]$, the algorithm *Bifoliate_Profile* constructs the label sequences $w_1 = l(v_1)l(v_3) = bb$ and $w_2 = l(v_4)l(v_5) = ba$. Since $|w_1| = |w_2|$, $w_2 \prec w_1$ and $w_r = l(v_4) = b$, w is set to $bbbab$. Furthermore, it holds that $3 \neq D[4]+2 = 5$. Since $p = 3+2\cdot 2+1-3 = 5 = q$, the algorithm *Bifoliate_Profile* constructs $w_3 = \varepsilon$ and increments the frequency of bifoliate 5-gram $(P^5_{2,0}, bbbab)$, where $|w_1| = 2$.

Moreover, for index $2 \in id[3][2]$, the algorithm *Bifoliate_Profile* constructs the label sequences $w_1 = l(v_2)l(v_3) = ab$ and $w_2 = ba$. Since $|w_1| = |w_2|$, $w_1 \prec w_2$ and $w_r = l(v_4) = b$, w is set to $baabb$ by replacing w_1 with w_2. Similarly to the case for index 1, the algorithm *Bifoliate_Profile* increments the frequency of bifoliate 5-gram $(P^5_{2,0}, baabb)$.

d	(j,k)
2	$(3,1)$
1	$(2,1),(3,2)$

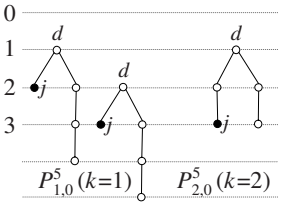


Fig. 3. The table *shift* for $q = 5$ and $\max D = 3$

id		3	3	2	3	2	1	3	2	3	2	3	2	1	0
j	k	1	2	3	4	5	6	7	8	9	10	11	12	13	14
3	1	**1**	1, 2		4			7		9		11			
3	2			**1, 2**	**1, 2**	1, 2, 4			**7**	**7**	**7, 9**	**7, 9**	7, 9, 11		
3	3						1, 2, 4	1, 2, 4	1, 2, 4	1, 2, 4	1, 2, 4	1, 2, 4	**1, 2, 4**	1, 2, 4, 7, 9, 11	
2	1			**3**	3	3, 5			**8**	8	**8, 10**	8, 10	8, 10, 12		
2	2						3, 5	**3, 5**	3, 5	**3, 5**	3, 5	**3, 5**	3, 5	3, 5, 8, 10, 12	
1	1						**6**	6	**6**	6	**6**	6	6	6, 13	

Fig. 4. The transition of the table id in the algorithm *Bifoliate_Profile*

On the other hand, for index $3 \in id[2][1]$, the algorithm *Bifoliate_Profile* constructs the label sequences $w_1 = l(v_3) = b$ and $w_2 = ba$. Since $|w_1| < |w_2|$ and $w_r = l(v_4) = b$, w is set to $babb$ by replacing w_1 with w_2. Furthermore, it holds that $3 \neq D[4] + 2 = 5$. Since $p = 3 + 2 \cdot 1 + 2 - 3 = 4$ and $q - p = 1$, the algorithm *Bifoliate_Profile* constructs $w_3 = l(v_6) = a$ and increments the frequency of bifoliate 5-gram $(P^5_{3,1}, bbbab)$, where $|w_1| + q - p = 2 + 5 - 4 = 3$.

As a result, we obtain the frequencies of bifoliate 5-grams in T that are non-negative for every $P^q_{k,b}$ as in Figure 5.

$P^5_{k,b}$	$(w, \text{frequency})$
$P^5_{2,0}$	$(babaa, 1)$ $(abaaa, 2)$ $(bbbaa, 1)$ $(bbbab, 1)$ $(baaba, 3)$ $(bbaaa, 2)$ $(baabb, 1)$
$P^5_{3,0}$	$(babaa, 1)$ $(bbbaa, 1)$ $(ababa, 1)$ $(baaba, 2)$ $(abbaa, 1)$
$P^5_{3,1}$	$(bbaba, 1)$ $(babaa, 2)$ $(abaaa, 2)$ $(babba, 1)$ $(ababa, 1)$ $(baaaa, 2)$
$P^5_{3,2}$	$(babba, 1)$

Fig. 5. The frequencies of bifoliate 5-grams in T that are non-negative

Theorem 1. *For a tree T, let $D = D(T)$, $L = L(T)$, $n = |T|$, $d = dep(T)$ and $l = lvs(T)$. Then, the algorithm Bifoliate_Profile(q, D, L) described in Algorithm 2 is correct and runs in $O(qd \min(q, d)ln)$ time.*

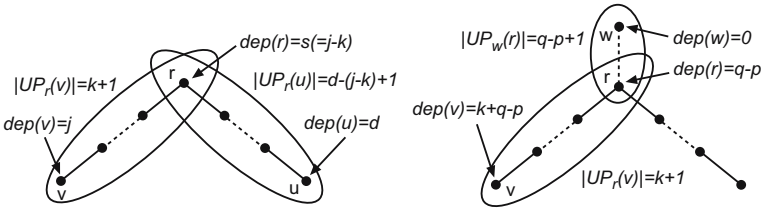
Proof. First, we discuss the correctness of the algorithm *Bifoliate_Profile*.

Consider an ordered p-gram P^p_k with left leaf v and right leaf u, where $j = dep(v)$ and $d = dep(u)$. Also let s be $j - k$. Then, it holds that $p = d + 2k + 1 - j$,

and s is the depth of root r of P_k^p (Figure 6(a)). This corresponds to lines 5–6 in the algorithm *Bifoliate_Profile*.

Let k' be the depth of a deeper leaf of P_k^p. If $q - p > s$, then there exists no bifoliate q-gram $P^q_{k'+q-p,q-p}$. Otherwise, if $q - p \leq s$ (line 7), then the algorithm *Bifoliate_Profile* finds the label sequences w_1 on the path from v to the child of r on $UP_r(v)$ and w_2 on the path from u to the child of r on $UP_r(u)$ (lines 9–10) using the subroutine *labels*. By comparing the length of w_1 with that of w_2, the algorithm *Bifoliate_Profile* determines which of v and u is a deeper leaf, and it then constructs the label sequence $w_{12r} = w_1 \cdot w_2 \cdot w_r$ (where $w_r = l(r)$) of a bifoliate q-gram $P^p_{|w_1|,0}$ by setting v (corresponding to w_1) to a deeper leaf (lines 11–12).

Furthermore, if $j \neq d + k$, then it holds that $p \neq q$, so the algorithm *Bifoliate_Profile* finds a path from r to r' in Figure 6(b) that is the root of a given tree with length $q-p$ and its label sequence w_3 (line 15). Hence, $w_{12r3} = w_1 \cdot w_2 \cdot w_r \cdot w_3$ is the label sequence of a bifoliate q-gram $P^q_{|w_1|+q-p,q-p}$, and the algorithm *Bifoliate_Profile* increments the bifoliate q-gram $(P^q_{|w_1|+q-p,q-p}, w_{12r3})$ in the table P (line 16). Otherwise, if $j = d + k$ and $p = q$, that is, u is the root of an ordered q-gram P^q_{q-1}, then the algorithm *Bifoliate_Profile* increments the bifoliate q-gram $(P^q_{q-1,0}, w_{12r})$ in table P (line 17).



(a) (b)

Fig. 6. The relationship of the parameters in P_k^p (left) and $P^q_{k,b}$ (right)

The algorithm *Bifoliate_Profile* maintains the indices already searched in a table $id[j][k]$. Note that $l \in id[j][k]$ means that v_l is the left leaf of P_k^p. For an index i, the algorithm *Bifoliate_Profile* first stores it in $id[D[i]][1]$ (line 22). Next, for every $(j, k) \in shift[D[i]]$, the algorithm *Bifoliate_Profile* shifts the indices in $id[j][k]$ to $id[j][k+1]$ (lines 19–21), because $D[i]$ and j are the depths of the root and the left leaf of P_k^q, respectively. In this case, the algorithm *Bifoliate_Profile* finishes searching for P_k^q and begins searching for P^q_{k+1}.

Next, we consider the running time of the algorithm *Bifoliate_Profile*. Since $|id[j][k]| \leq l$ and $labels(i, k, PS, D)$ runs in $O(k)$ time, the running time of the routine from lines 8 to 17 is $O(ql)$. Here, in lines 16 and 17, we use the hash function to increment the element of $P[\cdot][\cdot][w]$ (by encoding a string w as a numeral), so the running time is assumed to be constant. Furthermore, the algorithms $pseq(D)$ and $shift\_table(q, D)$ (line 1) run in $O(n)$ and $O(qd)$ time, respectively. Since $|shift[D[i]]| \leq q$ for every i, the algorithm *Bifoliate_Profile* runs in $O(n + (d \times \min(q, d) \times ql + ql)n) = O(qd \min(q, d)ln)$ time. □

Table 1. Summary of the glycan data used in experiments

data set	# of data	avg.# of nodes	avg.height
leukemic cells	192	16.1	6.0
other blood components	294	10.4	5.4
colon cancer	93	7.8	4.2
other colon-related	46	9.7	4.5

4 Experimental Results

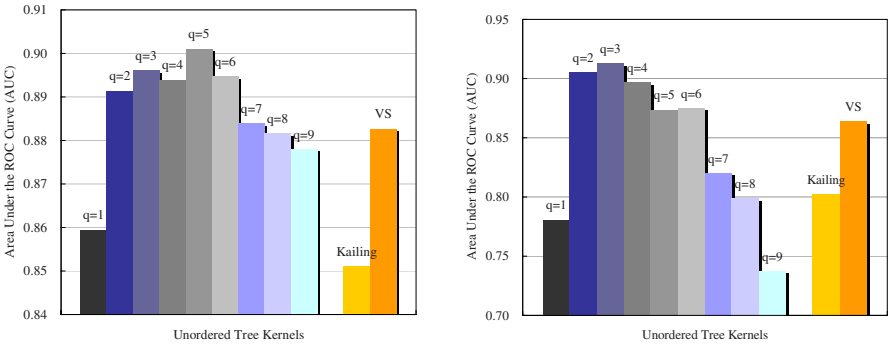
In this section, we evaluate the effectiveness of our kernel by empirically comparing its predictive performance in glycan structure classification problems with two other kernels for unordered trees. Glycans are defined as the third major class of biomolecules next to DNA and proteins and play important roles in various fundamental biological processes such as cell-cell interactions. Glycan structures are modeled as either ordered or unordered trees according to its context since the level of appropriate abstractions in modeling the structures depend on the problem to be addressed (cf. [1]). In this paper, we focus on unordered tree modeling of glycans.

We consider the following two competitors to the bifoliate tree kernel. One is the tree kernel by Vishwanathan [17] based on a string kernel, and the other, denoted by $K_H(T_1, T_2)$, is defined based on three simple vectors used in the dissimilarity measure proposed by Kailing *et al.* [8], which are the vectors of the degree histogram $V_d(T)$, the height histogram $V_h(T)$, and the label histogram $V_l(T)$ for an unordered tree T. We define the kernel $K_H(T_1, T_2)$ for two trees T_1 and T_2 as the sum of the inner products of each pair of vectors.

$$K_H(T_1, T_2) = \langle V_d(T_1), V_d(T_2) \rangle + \langle V_h(T_1), V_h(T_2) \rangle + \langle V_l(T_1), V_l(T_2) \rangle.$$

These kernels were implemented in Ruby and executed on a Windows XP machine with a Pentium M processor running at 1.50 GHz and 750 MB of memory. We used LIBSVM [3] as the SVM implementation, and we computed the area under the ROC curve (AUC) for measuring performance. AUC is the prevailing performance measure for a decision function with a kernel that separates positive examples from negative ones. The AUC values range from 0.5 to 1.0, where the value 0.5 indicates a random separation and the value 1.0 indicates a perfect separation.

The glycan data that we used in the first experiment basically follows Hizukuri *et al.* [6]; we retrieved the glycan structures from the KEGG/GLYCAN database [11] and used the annotations from the CarbBank/CCSD database [2]. Based on these annotations, we extracted those structures annotated with blood components, labeled as *leukemic cells*, and other non-leukemic blood components (*erythrocyte*, *serum*, and *plasma*). *Leukemia* is a cancer of the blood induced by an abnormal proliferation of blood components (usually white blood cells). In



Leukemic vs the other organs in blood Cancer vs the other organs in colon

Fig. 7. The areas under the ROC curves for two experiments

the second experiment, we employ two data sets from colon, i.e. glycans related to colon cancer, and others not related to cancer but related to the colon. We retrieved 29 distinct node labels. We have summarized the data used in our experiments in Table 1.

Figure 7 shows the comparison of the results by the proposed method while varying the parameter q. The kernel by Vishwanathan [17] is indicated by "VS", and the kernel based on dissimilarity proposed by Kailing *et al.* [8] is indicated by "Kailing." All of the performance measures were calculated with 5-fold cross validation.

Our tree kernel achieves the best performances at $q = 5$ and $q = 3$ for the leukemia and colon data sets, respectively. The tree kernel due to Vishwanathan also achieves relatively good performance in spite of its restricted expressive power. Since the nodes near the leaves tend to determine the functionalities of glycans, this data set seems to be well-suited to this tree kernel.

Also, it is interesting to see that the value of q achieving the highest predictive performance varies between the two experiments, which indicates that the q size of the most characteristic features varies according to the data set. This corresponds with previous knowledge that structure of glycan biomarkers are varied depending on the cell population being studied.

5 Conclusion

We have presented a novel kernel function for rooted labeled unordered trees. Given two trees, our tree kernel counts the number of common *bifoliate q-grams* between them, which are trees with at most two leaves and a fixed number of nodes q. We conducted comparative experiments to illustrate the efficiency of our kernel by applying it to the classification problem of glycan structures. Our kernel outperformed the existing kernels for unordered trees in its predictive performance. The experiments also suggested that the performance depends on the fixed number q, and the optimal value q to give the best performance depends on the data set.

In the future, we plan to design a new tree kernel based on the bifoliate tree kernel so that we can select an appropriate parameter q to achieve better average performance regardless of the data set.

References

1. Aoki, K.F., Ueda, N., Yamaguchi, A., Akutsu, T., Kanehisa, M., Mamitsuka, H.: Managing and analyzing carbohydrate data. SIGMOD Rec. 33(2), 33–38 (2004)
2. Doubet, S., Albersheim, P.: CarbBank. Glycobiology 2(6), 505 (1992)
3. Chang, C.-C., Lin, C.-J.: LIBSVM: a library for support vector machines (2001), `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`
4. Collins, M., Duffy, N.: Convolution Kernels for Natural Language. In: Proc. NIPS 2001, pp. 625–632 (2001)
5. Haussler, D.: Convolution Kernels on Discrete Structures, Technical Report UCSC-CRL 99-10 (1999)
6. Hizukuri, Y., Yamanishi, Y., Nakamura, O., Yagi, F., Goto, S., Kanehisa, M.: Extraction of leukemia specific glycan motifs in humans by computational glycomics. Carbohydrate Research 340, 2270–2278 (2005)
7. Jiang, T., Wang, L., Zhang, K.: Alignment of trees - an alternative to tree edit. Theoret. Comput. Sci. 143, 137–148 (1995)
8. Kailing, K., Kriegel, H.-P., Schönauer, S., Seidl, T.: Efficient similarity search for hierarchical data in large databases. In: Bertino, E., Christodoulakis, S., Plexousakis, D., Christophides, V., Koubarakis, M., Böhm, K., Ferrari, E. (eds.) EDBT 2004. LNCS, vol. 2992, pp. 676–693. Springer, Heidelberg (2004)
9. Kashima, H., Koyanagi, T.: Kernels for Semi-Structured Data. In: Proc. ICML 2002, pp. 291–298 (2002)
10. Kashima, H., Sakamoto, H., Koyanagi, T.: Tree Kernels (in Japanese). J. JSAI 21(1), 113–121 (2006)
11. Hashimoto, K., Goto, S., Kawano, S., Aoki-Kinoshita, K.F., Ueda, N.: KEGG as a glycome informatics resource. Glycobiology 16, 63R–70R (2006)
12. Kuboyama, T., Hirata, K., Ohkura, N., Harao, M.: A q-gram based distance measure for ordered labeled trees. In: Proc. LLLL 2006, pp. 77–83 (2006)
13. Kuboyama, T., Hirata, K., Aoki-Kinoshita, K.F., Kashima, H., Yasuda, H.: A gram distribution kernel applied to glycan classification and motif extraction. In: Proc. GIW 2006, pp. 25–34 (2006)
14. Kuboyama, T., Hirata, K., Aoki-Kinoshita, K.F., Kashima, H., Yasuda, H.: A spectrum tree kernel. J. JSAI 22(2), 140–147 (2007)
15. Ohkura, N., Hirata, K., Kuboyama, T., Harao, M.: The q-gram distance for ordered unlabeled trees. In: Hoffmann, A., Motoda, H., Scheffer, T. (eds.) DS 2005. LNCS (LNAI), vol. 3735, pp. 189–202. Springer, Heidelberg (2005)
16. Ohkura, N., Hirata, K., Kuboyama, T., Nakano, S., Harao, M.: The gram distribution for rooted ordered trees. In: Proc. LLLL 2006, pp. 69–76 (2006)
17. Vishwanathan, S.V.N.: Kernel Methods: Fast Algorithms and Real Life Applications, PhD thesis, Indian Institute of Science, Bangalore (2002)
18. Yang, R., Kalnis, P., Tung, A.K.H.: Similarity evaluation on tree-structed data. In: Proc. SIGMOD 2005, pp. 754–765 (2005)
19. Zhang, K., Shasha, D.: Tree pattern matching. In: Apostolico, A., Galil, Z. (eds.) Pattern matching algorithms, pp. 341–371 (1997)
20. Zhang, K., Statman, R., Shasha, D.: On the editing distance between unordered labeled trees. Inform. Proc. Let. 42, 133–139 (1992)