Convolution Kernels with Feature Selection for Natural Language Processing Tasks

Jun Suzuki, Hideki Isozaki and Eisaku Maeda
NTT Communication Science Laboratories, NTT Corp.
2-4 Hikaridai, Seika-cho, Soraku-gun, Kyoto,619-0237 Japan
{jun, isozaki, maeda}@cslab.kecl.ntt.co.jp

Abstract

Convolution kernels, such as sequence and tree kernels, are advantageous for both the concept and accuracy of many *natural language processing* (NLP) tasks. Experiments have, however, shown that the over-fitting problem often arises when these kernels are used in NLP tasks. This paper discusses this issue of convolution kernels, and then proposes a new approach based on statistical feature selection that avoids this issue. To enable the proposed method to be executed efficiently, it is embedded into an original kernel calculation process by using *sub-structure mining* algorithms. Experiments are undertaken on real NLP tasks to confirm the problem with a conventional method and to compare its performance with that of the proposed method.

1 Introduction

Over the past few years, many machine learning methods have been successfully applied to tasks in *natural language processing* (NLP). Especially, state-of-the-art performance can be achieved with kernel methods, such as *Support Vector Machine* (Cortes and Vapnik, 1995). Examples include text categorization (Joachims, 1998), chunking (Kudo and Matsumoto, 2002) and parsing (Collins and Duffy, 2001).

Another feature of this kernel methodology is that it not only provides high accuracy but also allows us to design a *kernel function* suited to modeling the task at hand. Since natural language data take the form of sequences of words, and are generally analyzed using discrete structures, such as trees (parsed trees) and graphs (relational graphs), *discrete kernels*, such as sequence kernels (Lodhi et al., 2002), tree kernels (Collins and Duffy, 2001), and graph kernels (Suzuki et al., 2003a), have been shown to offer excellent results.

These discrete kernels are related to *convolution kernels* (Haussler, 1999), which provides the concept of kernels over discrete structures. Convolution kernels allow us to treat structural features without explicitly representing the feature vectors from the input object. That is, convolution kernels are well suited to NLP tasks in terms of both accuracy and concept.

Unfortunately, experiments have shown that in some cases there is a critical issue with convolution kernels, especially in NLP tasks (Collins and Duffy, 2001; Cancedda et al., 2003; Suzuki et al., 2003b). That is, the *over-fitting problem* arises if large "sub-structures" are used in the kernel calculations. As a result, the machine learning approach can never be trained efficiently.

To solve this issue, we generally eliminate large sub-structures from the set of features used. However, the main reason for using convolution kernels is that we aim to use structural features easily and efficiently. If use is limited to only very small structures, it negates the advantages of using convolution kernels.

This paper discusses this issue of convolution kernels, and proposes a new method based on statistical feature selection. The proposed method deals only with those features that are statistically significant for kernel calculation, large significant sub-structures can be used without over-fitting. Moreover, the proposed method can be executed efficiently by embedding it in an original kernel calculation process by using sub-structure mining algorithms.

In the next section, we provide a brief overview of convolution kernels. Section 3 discusses one issue of convolution kernels, the main topic of this paper, and introduces some conventional methods for solving this issue. In Section 4, we propose a new approach based on statistical feature selection to offset the issue of convolution kernels using an example consisting of sequence kernels. In Section 5, we briefly discuss the application of the proposed method to other convolution kernels. In Section 6, we compare the performance of conventional methods with that of the proposed method by using real NLP tasks: question classification and sentence modality identification. The experimental

results described in Section 7 clarify the advantages of the proposed method.

2 Convolution Kernels

Convolution kernels have been proposed as a concept of kernels for discrete structures, such as sequences, trees and graphs. This framework defines the kernel function between input objects as the convolution of "sub-kernels", i.e. the kernels for the decompositions (parts) of the objects.

Let X and Y be discrete objects. Conceptually, convolution kernels $K(X, Y)$ enumerate all substructures occurring in X and Y and then calculate their inner product, which is simply written as:

$$K(X, Y) = \langle \phi(X), \phi(Y) \rangle = \sum_i \phi_i(X) \cdot \phi_i(Y). \quad (1)$$

ϕ represents the feature mapping from the discrete object to the feature space; that is, $\phi(X) = (\phi_1(X), \ldots, \phi_i(X), \ldots)$. With sequence kernels (Lodhi et al., 2002), input objects X and Y are sequences, and $\phi_i(X)$ is a sub-sequence. With tree kernels (Collins and Duffy, 2001), X and Y are trees, and $\phi_i(X)$ is a sub-tree.

When implemented, these kernels can be efficiently calculated in quadratic time by using dynamic programming (DP).

Finally, since the size of the input objects is not constant, the kernel value is normalized using the following equation.

$$\hat{K}(X, Y) = \frac{K(X, Y)}{\sqrt{K(X, X) \cdot K(Y, Y)}} \quad (2)$$

The value of $\hat{K}(X, Y)$ is from 0 to 1, $\hat{K}(X, Y) = 1$ if and only if $X = Y$.

2.1 Sequence Kernels

To simplify the discussion, we restrict ourselves hereafter to sequence kernels. Other convolution kernels are briefly addressed in Section 5.

Many kinds of sequence kernels have been proposed for a variety of different tasks. This paper basically follows the framework of *word sequence kernels* (Cancedda et al., 2003), and so processes *gapped word sequences* to yield the kernel value.

Let Σ be a set of finite symbols, and Σ^n be a set of possible (symbol) sequences whose sizes are n or less that are constructed by symbols in Σ. The meaning of "size" in this paper is the number of symbols in the sub-structure. Namely, in the case of sequence, size n means length n. S and T can represent any sequence. s_i and t_j represent the ith and jth symbols in S and T, respectively. Therefore, a

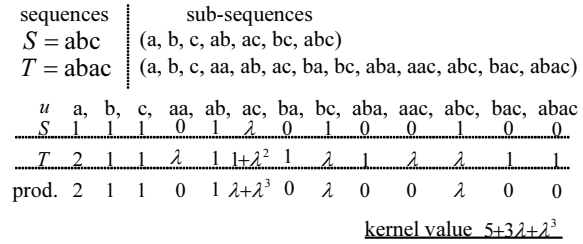

Figure 1: Example of sequence kernel output

sequence S can be written as $S = s_1 \ldots s_i \ldots s_{|S|}$, where $|S|$ represents the length of S. If sequence u is contained in sub-sequence $S[i : j] \stackrel{\text{def}}{=} s_i \ldots s_j$ of S (allowing the existence of gaps), the position of u in S is written as $\mathbf{i} = (i_1 : i_{|u|})$. The length of $S[\mathbf{i}]$ is $l(\mathbf{i}) = i_{|u|} - i_1 + 1$. For example, if $u = ab$ and $S = cacbd$, then $\mathbf{i} = (2 : 4)$ and $l(\mathbf{i}) = 4 - 2 + 1 = 3$.

By using the above notations, sequence kernels can be defined as:

$$K^{\text{SK}}(S, T) = \sum_{u \in \Sigma^n} \sum_{\mathbf{i}|u=S[\mathbf{i}]} \lambda^{\gamma(\mathbf{i})} \sum_{\mathbf{j}|u=T[\mathbf{j}]} \lambda^{\gamma(\mathbf{j})}, \quad (3)$$

where λ is the decay factor that handles the gap present in a common sub-sequence u, and $\gamma(\mathbf{i}) = l(\mathbf{i}) - |u|$. In this paper, $|$ means "such that". Figure 1 shows a simple example of the output of this kernel.

However, in general, the number of features $|\Sigma^n|$, which is the dimension of the feature space, becomes very high, and it is computationally infeasible to calculate Equation (3) explicitly. The efficient recursive calculation has been introduced in (Cancedda et al., 2003). To clarify the discussion, we redefine the sequence kernels with our notation.

The sequence kernel can be written as follows:

$$K^{\text{SK}}(S, T) = \sum_{m=1}^{n} \sum_{1 \le i \le |S|} \sum_{1 \le j \le |T|} J_m(S_i, T_j). \quad (4)$$

where S_i and T_j represent the sub-sequences $S_i = s_1, s_2, \ldots, s_i$ and $T_j = t_1, t_2, \ldots, t_j$, respectively.

Let $J_m(S_i, T_j)$ be a function that returns the value of common sub-sequences if $s_i = t_j$.

$$J_m(S_i, T_j) = J'_{m-1}(S_i, T_j) \cdot I(s_i, t_j) \quad (5)$$

$I(s_i, t_j)$ is a function that returns a matching value between s_i and t_j. This paper defines $I(s_i, t_j)$ as an indicator function that returns 1 if $s_i = t_j$, otherwise 0.

Then, $J'_m(S_i, T_j)$ and $J''_m(S_i, T_j)$ are introduced to calculate the common *gapped* sub-sequences between S_i and T_j.

$$J'_m(S_i, T_j) = \begin{cases} 1 \quad \text{if} \quad m = 0, \\ 0 \quad \text{if} \quad j = 0 \text{ and } m > 0, \\ \lambda J'_m(S_i, T_{j-1}) + J''_m(S_i, T_{j-1}) \\ \qquad \text{otherwise} \end{cases} \quad (6)$$

$$J''_m(S_i, T_j) = \begin{cases} 0 \quad \text{if} \quad i = 0, \\ \lambda J''_m(S_{i-1}, T_j) + J_m(S_{i-1}, T_j) \\ \qquad \text{otherwise} \end{cases} \quad (7)$$

If we calculate Equations (5) to (7) recursively, Equation (4) provides exactly the same value as Equation (3).

3 Problem of Applying Convolution Kernels to NLP tasks

This section discusses an issue that arises when applying convolution kernels to NLP tasks.

According to the original definition of convolution kernels, all the sub-structures are enumerated and calculated for the kernels. The number of sub-structures in the input object usually becomes exponential against input object size. As a result, all kernel values $\hat{K}(X, Y)$ are nearly 0 except the kernel value of the object itself, $\hat{K}(X, X)$, which is 1. In this situation, the machine learning process becomes almost the same as *memory-based learning*. This means that we obtain a result that is very precise but with very low recall.

To avoid this, most conventional methods use an approach that involves smoothing the kernel values or eliminating features based on the sub-structure size.

For sequence kernels, (Cancedda et al., 2003) use a feature elimination method based on the size of sub-sequence n. This means that the kernel calculation deals only with those sub-sequences whose size is n or less. For tree kernels, (Collins and Duffy, 2001) proposed a method that restricts the features based on sub-trees depth. These methods seem to work well on the surface, however, good results are achieved only when n is very small, i.e. $n = 2$.

The main reason for using convolution kernels is that they allow us to employ structural features simply and efficiently. When only small sized sub-structures are used (i.e. $n = 2$), the full benefits of convolution kernels are missed.

Moreover, these results do not mean that larger sized sub-structures are not useful. In some cases we already know that larger sub-structures are significant features as regards solving the target problem. That is, these significant larger sub-structures, which the conventional methods cannot deal with efficiently, should have a possibility of improving the performance furthermore.

The aim of the work described in this paper is to be able to use any significant sub-structure efficiently, regardless of its size, to solve NLP tasks.

Table 1: Contingency table and notation for the chi-squared value

	c	$\bar{c}$	$\sum$ row
u	$O_{uc} = y$	$O_{u\bar{c}}$	$O_u = x$
$\bar{u}$	$O_{\bar{u}c}$	$O_{\bar{u}\bar{c}}$	$O_{\bar{u}}$
$\sum$ column	$O_c = M$	$O_{\bar{c}}$	N

4 Proposed Feature Selection Method

Our approach is based on statistical feature selection in contrast to the conventional methods, which use sub-structure size.

For a better understanding, consider the two-class (positive and negative) supervised classification problem. In our approach we test the statistical deviation of all the sub-structures in the training samples between the appearance of positive samples and negative samples. This allows us to select only the statistically significant sub-structures when calculating the kernel value.

Our approach, which uses a statistical metric to select features, is quite natural. We note, however, that kernels are calculated using the DP algorithm. Therefore, it is not clear how to calculate kernels efficiently with a statistical feature selection method. First, we briefly explain a statistical metric, the chi-squared (χ^2) value, and provide an idea of how to select significant features. We then describe a method for embedding statistical feature selection into kernel calculation.

4.1 Statistical Metric: Chi-squared Value

There are many kinds of statistical metrics, such as chi-squared value, correlation coefficient and mutual information. (Rogati and Yang, 2002) reported that chi-squared feature selection is the most effective method for text classification. Following this information, we use χ^2 values as statistical feature selection criteria. Although we selected χ^2 values, any other statistical metric can be used as long as it is based on the contingency table shown in Table 1.

We briefly explain how to calculate the χ^2 value by referring to Table 1. In the table, c and $\bar{c}$ represent the names of classes, c for the positive class

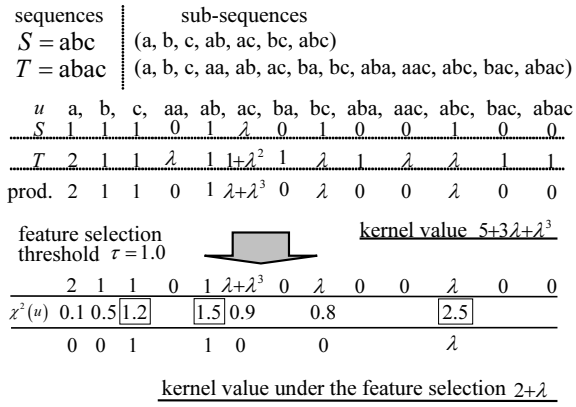

Figure 2: Example of statistical feature selection

and $\bar{c}$ for the negative class. O_{uc}, $O_{u\bar{c}}$, $O_{\bar{u}c}$ and $O_{\bar{u}\bar{c}}$ represent the number of u that appeared in the positive sample c, the number of u that appeared in the negative sample $\bar{c}$, the number of u that did not appear in c, and the number of u that did not appear in $\bar{c}$, respectively. Let y be the number of samples of positive class c that contain sub-sequence u, and x be the number of samples that contain u. Let N be the total number of (training) samples, and M be the number of positive samples.

Since N and M are constant for (fixed) data, χ^2 can be written as a function of x and y,

$$\chi^2(x,y) = \frac{N(O_{uc}\cdot O_{\bar{u}\bar{c}} - O_{\bar{u}c}\cdot O_{u\bar{c}})^2}{O_u \cdot O_{\bar{u}} \cdot O_c \cdot O_{\bar{c}}}. \quad (8)$$

χ^2 expresses the normalized deviation of the observation from the expectation.

We simply represent $\chi^2(x,y)$ as $\chi^2(u)$.

4.2 Feature Selection Criterion

The basic idea of feature selection is quite natural. First, we decide the threshold τ of the χ^2 value. If $\chi^2(u) < \tau$ holds, that is, u is not statistically significant, then u is eliminated from the features and the value of u is presumed to be 0 for the kernel value.

The sequence kernel with feature selection (FSSK) can be defined as follows:

$$K^{\text{FSSK}}(S,T) = \sum_{\tau \le \chi^2(u) | u \in \Sigma^n} \sum_{\mathbf{i}|u=S[\mathbf{i}]} \lambda^{\gamma(\mathbf{i})} \sum_{\mathbf{j}|u=T[\mathbf{j}]} \lambda^{\gamma(\mathbf{j})}. \quad (9)$$

The difference between Equations (3) and (9) is simply the condition of the first summation. FSSK selects significant sub-sequence u by using the condition of the statistical metric $\tau \le \chi^2(u)$.

Figure 2 shows a simple example of what FSSK calculates for the kernel value.

4.3 Efficient $\chi^2(u)$ Calculation Method

It is computationally infeasible to calculate $\chi^2(u)$ for all possible u with a naive exhaustive method. In our approach, we use a *sub-structure mining* algorithm to calculate $\chi^2(u)$. The basic idea comes from a sequential pattern mining technique, PrefixSpan (Pei et al., 2001), and a statistical metric pruning (SMP) method, Apriori SMP (Morishita and Sese, 2000). By using these techniques, all the significant sub-sequences u that satisfy $\tau \le \chi^2(u)$ can be found efficiently by depth-first search and pruning. Below, we briefly explain the concept involved in finding the significant features.

First, we denote uv, which is the concatenation of sequences u and v. Then, u is a specific sequence and uv is any sequence that is constructed by u with any suffix v. The upper bound of the χ^2 value of uv can be defined by the value of u (Morishita and Sese, 2000).

$$\begin{aligned}\chi^2(uv) &\le \max\left(\chi^2(y_u, y_u), \chi^2(x_u - y_u, 0)\right)\\ &= \widehat{\chi}^2(u)\end{aligned}$$

where x_u and y_u represent the value of x and y of u. This inequation indicates that if $\widehat{\chi}^2(u)$ is less than a certain threshold τ, all sub-sequences uv can be eliminated from the features, because no sub-sequence uv can be a feature.

The PrefixSpan algorithm enumerates all the significant sub-sequences by using a depth-first search and constructing a TRIE structure to store the significant sequences of internal results efficiently. Specifically, PrefixSpan algorithm evaluates uw, where uw represents a concatenation of a sequence u and a symbol w, using the following three conditions.

1. $\tau \le \chi^2(uw)$
2. $\tau > \chi^2(uw), \quad \tau > \widehat{\chi}^2(uw)$
3. $\tau > \chi^2(uw), \quad \tau \le \widehat{\chi}^2(uw)$

With 1, sub-sequence uw is selected as a significant feature. With 2, sub-sequence uw and arbitrary sub-sequences uwv, are less than the threshold τ. Then w is pruned from the TRIE, that is, all uwv where v represents any suffix pruned from the search space. With 3, uw is not selected as a significant feature because the χ^2 value of uw is less than τ, however, uwv can be a significant feature because the upper-bound χ^2 value of uwv is greater than τ, thus the search is continued to uwv.

Figure 3 shows a simple example of PrefixSpan with SMP that searches for the significant features

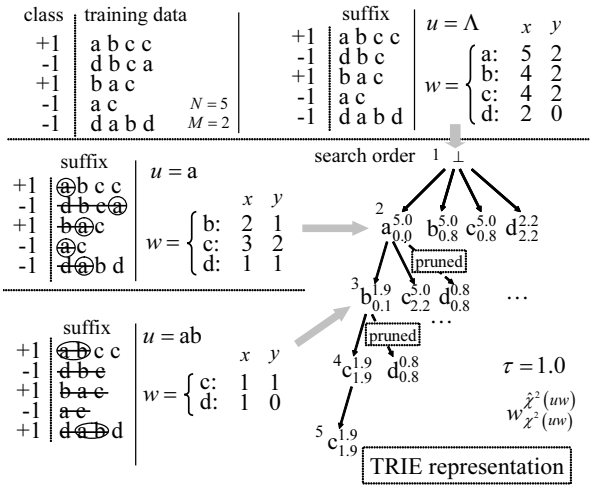

Figure 3: Efficient search for statistically significant sub-sequences using the PrefixSpan algorithm with SMP

by using a depth-first search with a TRIE representation of the significant sequences. The values of each symbol represent $\chi^2(u)$ and $\widehat{\chi}^2(u)$ that can be calculated from the number of x_u and y_u. The TRIE structure in the figure represents the statistically significant sub-sequences that can be shown in a path from $\perp$ to the symbol.

We exploit this TRIE structure and PrefixSpan pruning method in our kernel calculation.

4.4 Embedding Feature Selection in Kernel Calculation

This section shows how to integrate statistical feature selection in the kernel calculation. Our proposed method is defined in the following equations.

$$K^{\text{FSSK}}(S,T) = \sum_{m=1}^{n} \sum_{1\le i\le |S|} \sum_{1\le j\le |T|} \mathcal{K}_m(S_i,T_j) \quad (10)$$

Let $\mathcal{K}_m(S_i,T_j)$ be a function that returns the sum value of all statistically significant common subsequences u if $s_i = t_j$.

$$\mathcal{K}_m(S_i,T_j) = \sum_{u\in\Gamma_m(S_i,T_j)} \mathcal{J}_u(S_i,T_j), \quad (11)$$

where $\Gamma_m(S_i,T_j)$ represents a set of sub-sequences whose size $|u|$ is m and that satisfy the above condition 1. The $\Gamma_m(S_i,T_j)$ is defined in detail in Equation (15).

Then, let $\mathcal{J}_u(S_i,T_j)$, $\mathcal{J}'_u(S_i,T_j)$ and $\mathcal{J}''_u(S_i,T_j)$ be functions that calculate the value of the common sub-sequences between S_i and T_j recursively, as well as equations (5) to (7) for sequence kernels. We introduce a special symbol Λ to represent an "empty sequence", and define $\Lambda w = w$ and $|\Lambda w| = 1$.

$$\mathcal{J}_{uw}(S_i,T_j) = \begin{cases} \mathcal{J}'_u(S_i,T_j)\cdot\mathcal{I}(w) \\ \quad \text{if } uw \in \widehat{\Gamma}_{|uw|}(S_i,T_j), \\ 0 \quad \text{otherwise} \end{cases} \quad (12)$$

where $\mathcal{I}(w)$ is a function that returns a matching value of w. In this paper, we define $\mathcal{I}(w)$ is 1.

$\widehat{\Gamma}_m(S_i,T_j)$ has realized conditions 2 and 3; the details are defined in Equation (16).

$$\mathcal{J}'_u(S_i,T_j) = \begin{cases} 1 \quad \text{if} \quad u=\Lambda, \\ 0 \quad \text{if} \quad j=0 \quad \text{and} \quad u\neq\Lambda, \\ \lambda\mathcal{J}'_u(S_i,T_{j-1}) + \mathcal{J}''_u(S_i,T_{j-1}) \\ \qquad \text{otherwise} \end{cases} \quad (13)$$

$$\mathcal{J}''_u(S_i,T_j) = \begin{cases} 0 \quad \text{if} \quad i=0, \\ \lambda\mathcal{J}''_u(S_{i-1},T_j) + \mathcal{J}_u(S_{i-1},T_j) \\ \qquad \text{otherwise} \end{cases} \quad (14)$$

The following five equations are introduced to select a set of significant sub-sequences. $\Gamma_m(S_i,T_j)$ and $\widehat{\Gamma}_m(S_i,T_j)$ are sets of sub-sequences (features) that satisfy condition 1 and 3, respectively, when calculating the value between S_i and T_j in Equations (11) and (12).

$$\Gamma_m(S_i,T_j) = \{u \mid u\in\widehat{\Gamma}_m(S_i,T_j), \tau \le \chi^2(u)\} \quad (15)$$

$$\widehat{\Gamma}_m(S_i,T_j) = \begin{cases} \Psi(\widehat{\Gamma}'_{m-1}(S_i,T_j), s_i) \\ \quad \text{if} \quad s_i = t_j \\ \emptyset \quad \text{otherwise} \end{cases} \quad (16)$$

$$\Psi(F,w) = \{uw \mid u\in F, \tau \le \widehat{\chi}^2(uw)\}, \quad (17)$$

where F represents a set of sub-sequences. Notice that $\Gamma_m(S_i,T_j)$ and $\widehat{\Gamma}_m(S_i,T_j)$ have only subsequences u that satisfy $\tau \le \chi^2(uw)$ or $\tau \le \widehat{\chi}^2(uw)$, respectively, if $s_i = t_j (= w)$; otherwise they become empty sets.

The following two equations are introduced for recursive set operations to calculate $\Gamma_m(S_i,T_j)$ and $\widehat{\Gamma}_m(S_i,T_j)$.

$$\widehat{\Gamma}'_m(S_i,T_j) = \begin{cases} \{\Lambda\} \quad \text{if } m=0, \\ \emptyset \quad \text{if } j=0 \quad \text{and} \quad m>0, \\ \widehat{\Gamma}'_m(S_i,T_{j-1}) \cup \widehat{\Gamma}''_m(S_i,T_{j-1}) \\ \qquad \text{otherwise} \end{cases} \quad (18)$$

$$\widehat{\Gamma}''_m(S_i,T_j) = \begin{cases} \emptyset \quad \text{if} \quad i=0 \ , \\ \widehat{\Gamma}''_m(S_{i-1},T_j) \cup \widehat{\Gamma}_m(S_{i-1},T_j) \\ \qquad \text{otherwise} \end{cases} \quad (19)$$

In the implementation, Equations (11) to (14) can be performed in the same way as those used to calculate the original sequence kernels, if the feature selection condition of Equations (15) to (19) has been removed. Then, Equations (15) to (19), which select significant features, are performed by the PrefixSpan algorithm described above and the TRIE representation of statistically significant features.

The recursive calculation of Equations (12) to (14) and Equations (16) to (19) can be executed in the same way and at the same time in parallel. As a result, statistical feature selection can be embedded in oroginal sequence kernel calculation based on a dynamic programming technique.

4.5 Properties

The proposed method has several important advantages over the conventional methods.

First, the feature selection criterion is based on a statistical measure, so statistically significant features are automatically selected.

Second, according to Equations (10) to (18), the proposed method can be embedded in an original kernel calculation process, which allows us to use the same calculation procedure as the conventional methods. The only difference between the original sequence kernels and the proposed method is that the latter calculates a statistical metric $\chi^2(u)$ by using a sub-structure mining algorithm in the kernel calculation.

Third, although the kernel calculation, which unifies our proposed method, requires a longer training time because of the feature selection, the selected sub-sequences have a TRIE data structure. This means a fast calculation technique proposed in (Kudo and Matsumoto, 2003) can be simply applied to our method, which yields classification very quickly. In the classification part, the features (subsequences) selected in the learning part must be known. Therefore, we store the TRIE of selected sub-sequences and use them during classification.

5 Proposed Method Applied to Other Convolution Kernels

We have insufficient space to discuss this subject in detail in relation to other convolution kernels. However, our proposals can be easily applied to tree kernels (Collins and Duffy, 2001) by using string encoding for trees. We enumerate nodes (labels) of tree in postorder traversal. After that, we can employ a sequential pattern mining technique to select statistically significant sub-trees. This is because we can convert to the original sub-tree form from the string encoding representation.

Table 2: Parameter values of proposed kernels and Support Vector Machines

parameter	value
soft margin for SVM (C)	1000
decay factor of gap (λ)	0.5
threshold of χ^2 (τ)	2.7055 3.8415

As a result, we can calculate tree kernels with statistical feature selection by using the original tree kernel calculation with the sequential pattern mining technique introduced in this paper. Moreover, we can expand our proposals to hierarchically structured graph kernels (Suzuki et al., 2003a) by using a simple extension to cover hierarchical structures.

6 Experiments

We evaluated the performance of the proposed method in actual NLP tasks, namely *English question classification* (EQC), *Japanese question classification* (JQC) and *sentence modality identification* (MI) tasks.

We compared the proposed method (FSSK) with a conventional method (SK), as discussed in Section 3, and with *bag-of-words* (BOW) Kernel (BOW-K)(Joachims, 1998) as baseline methods.

Support Vector Machine (SVM) was selected as the kernel-based classifier for training and classification. Table 2 shows some of the parameter values that we used in the comparison. We set thresholds of $\tau = 2.7055$ (FSSK1) and $\tau = 3.8415$ (FSSK2) for the proposed methods; these values represent the 10% and 5% level of significance in the χ^2 distribution with one degree of freedom, which used the χ^2 significant test.

6.1 Question Classification

Question classification is defined as a task similar to text categorization; it maps a given question into a question type.

We evaluated the performance by using data provided by (Li and Roth, 2002) for English and (Suzuki et al., 2003b) for Japanese question classification and followed the experimental setting used in these papers; namely we use four typical question types, LOCATION, NUMEX, ORGANIZATION, and TIME_TOP for JQA, and "coarse" and "fine" classes for EQC. We used the *one-vs-rest* classifier of SVM as the multi-class classification method for EQC.

Figure 4 shows examples of the question classification data used here.

question types	input object : word sequences ([ ]: information of chunk and ⟨ ⟩: named entity)
ABBREVIATION	what,[B-NP] be,[B-VP] the,[B-NP] abbreviation,[I-NP] for,[B-PP] Texas,[B-NP],⟨B-GPE⟩ ?,[O]
DESCRIPTION	what,[B-NP] be,[B-VP] Aborigines,[B-NP] ?,[O]
HUMAN	who,[B-NP] discover,[B-VP] America,[B-NP],⟨B-GPE⟩ ?,[O]

Figure 4: Examples of English question classification data

Table 3: Results of the Japanese question classification (F-measure)

	(a) TIME_TOP					(b) LOCATION					(c) ORGANIZATION					(d) NUMEX				
n	1	2	3	4	∞	1	2	3	4	∞	1	2	3	4	∞	1	2	3	4	∞
FSSK1	-	**.961**	.958	.957	.956	-	.795	.793	.798	.792	-	.709	.720	.720	**.723**	-	.912	.915	.908	.908
FSSK2	-	**.961**	.956	.957	.956	-	.788	.799	**.804**	.800	-	.703	.710	.716	.720	-	.913	**.916**	.911	.913
SK	-	.946	.910	.866	.223	-	.791	.775	.732	.169	-	.705	.668	.594	.035	-	.912	.885	.817	.036
BOW-K	.902	.909	.886	.855	-	.744	.768	.756	.747	-	.641	690	.636	.572	-	.842	.852	.807	.726	-

6.2 Sentence Modality Identification

For example, sentence modality identification techniques are used in automatic text analysis systems that identify the modality of a sentence, such as "opinion" or "description".

The data set was created from Mainichi news articles and one of three modality tags, "opinion", "decision" and "description" was applied to each sentence. The data size was 1135 sentences consisting of 123 sentences of "opinion", 326 of "decision" and 686 of "description". We evaluated the results by using 5-fold cross validation.

7 Results and Discussion

Tables 3 and 4 show the results of Japanese and English question classification, respectively. Table 5 shows the results of sentence modality identification. n in each table indicates the threshold of the sub-sequence size. $n = \infty$ means all possible subsequences are used.

First, SK was consistently superior to BOW-K. This indicates that the structural features were quite efficient in performing these tasks. In general we can say that the use of structural features can improve the performance of NLP tasks that require the details of the contents to perform the task.

Most of the results showed that SK achieves its maximum performance when $n = 2$. The performance deteriorates considerably once n exceeds 4. This implies that SK with larger sub-structures degrade classification performance. These results show the same tendency as the previous studies discussed in Section 3. Table 6 shows the precision and recall of SK when $n = \infty$. As shown in Table 6, the classifier offered high precision but low recall. This is evidence of over-fitting in learning.

As shown by the above experiments, FSSK provided consistently better performance than the conventional methods. Moreover, the experiments confirmed one important fact. That is, in some cases maximum performance was achieved with $n = \infty$. This indicates that sub-sequences created using very large structures can be extremely effective. Of course, a larger feature space also includes the smaller feature spaces, $\Sigma^n \subset \Sigma^{n+1}$. If the performance is improved by using a larger n, this means that significant features do exist. Thus, we can improve the performance of some classification problems by dealing with larger substructures. Even if optimum performance was not achieved with $n = \infty$, difference between the performance of smaller n are quite small compared to that of SK. This indicates that our method is very robust as regards substructure size; It therefore becomes unnecessary for us to decide sub-structure size carefully. This indicates our approach, using large sub-structures, is better than the conventional approach of eliminating sub-sequences based on size.

Table 6: Precision and recall of SK: $n = \infty$

	Precision	Recall	F
MI:Opinion	.917	.209	.339
JQA:LOCATION	.896	.093	.168

8 Conclusion

This paper proposed a statistical feature selection method for convolution kernels. Our approach can select significant features automatically based on a statistical significance test. Our proposed method can be embedded in the DP based kernel calculation process for convolution kernels by using substructure mining algorithms.

Table 4: Results of English question classification (Accuracy)

	(a) coarse					(b) fine				
n	1	2	3	4	∞	1	2	3	4	∞
FSSK1	-	.908	.914	**.916**	.912	-	.852	.854	.852	.850
FSSK2	-	.902	.896	.902	.906	-	**.858**	.856	.854	.854
SK	-	.912	.914	.912	.892	-	.850	.840	.830	.796
BOW-K	.728	.836	.864	.858	-	.754	.792	.790	.778	-

Table 5: Results of sentence modality identification (F-measure)

	(a) opinion					(b) decision					(c) description				
n	1	2	3	4	∞	1	2	3	4	∞	1	2	3	4	∞
FSSK1	-	.734	.743	.746	**.751**	-	.828	.858	.854	.857	-	.896	.906	.910	.910
FSSK2	-	.740	.748	.750	.750	-	.824	.855	.859	**.860**	-	.894	.903	.909	.909
SK	-	.706	.672	.577	.058	-	.816	.834	.830	.339	-	.902	**.913**	.910	.808
BOW-K	.507	.531	.438	.368	-	.652	.708	.686	.665	-	.819	.839	.826	.793	-

Experiments show that our method is superior to conventional methods. Moreover, the results indicate that complex features exist and can be effective. Our method can employ them without over-fitting problems, which yields benefits in terms of concept and performance.

References

N. Cancedda, E. Gaussier, C. Goutte, and J.-M. Renders. 2003. Word-Sequence Kernels. *Journal of Machine Learning Research*, 3:1059–1082.

M. Collins and N. Duffy. 2001. Convolution Kernels for Natural Language. In *Proc. of Neural Information Processing Systems (NIPS'2001)*.

C. Cortes and V. N. Vapnik. 1995. Support Vector Networks. *Machine Learning*, 20:273–297.

D. Haussler. 1999. Convolution Kernels on Discrete Structures. In *Technical Report UCS-CRL-99-10*. UC Santa Cruz.

T. Joachims. 1998. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In *Proc. of European Conference on Machine Learning (ECML '98)*, pages 137–142.

T. Kudo and Y. Matsumoto. 2002. Japanese Dependency Analysis Using Cascaded Chunking. In *Proc. of the 6th Conference on Natural Language Learning (CoNLL 2002)*, pages 63–69.

T. Kudo and Y. Matsumoto. 2003. Fast Methods for Kernel-based Text Analysis. In *Proc. of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-2003)*, pages 24–31.

X. Li and D. Roth. 2002. Learning Question Classifiers. In *Proc. of the 19th International Conference on Computational Linguistics (COLING 2002)*, pages 556–562.

H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. 2002. Text Classification Using String Kernel. *Journal of Machine Learning Research*, 2:419–444.

S. Morishita and J. Sese. 2000. Traversing Itemset Lattices with Statistical Metric Pruning. In *Proc. of ACM SIGACT-SIGMOD-SIGART Symp. on Database Systems (PODS'00)*, pages 226–236.

J. Pei, J. Han, B. Mortazavi-Asl, and H. Pinto. 2001. PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth. In *Proc. of the 17th International Conference on Data Engineering (ICDE 2001)*, pages 215–224.

M. Rogati and Y. Yang. 2002. High-performing Feature Selection for Text Classification. In *Proc. of the 2002 ACM CIKM International Conference on Information and Knowledge Management*, pages 659–661.

J. Suzuki, T. Hirao, Y. Sasaki, and E. Maeda. 2003a. Hierarchical Directed Acyclic Graph Kernel: Methods for Natural Language Data. In *Proc. of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-2003)*, pages 32–39.

J. Suzuki, Y. Sasaki, and E. Maeda. 2003b. Kernels for Structured Natural Language Data. In *Proc. of the 17th Annual Conference on Neural Information Processing Systems (NIPS2003)*.