NAIST-IS-DD0361207

Doctoral Dissertation

Kernels for Structured Data in Natural Language Processing

Jun Suzuki

March 24, 2005

Department of Information Processing
Graduate School of Information Science
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Jun Suzuki

Kernels for Structured Data in Natural Language Processing[*]

Jun Suzuki

Abstract

The recent success of statistical *natural language processing (NLP)* has made feasible the development of challenging applications. For example, *text classification* traditionally classified texts into 'topics' but now researchers attempt to classify texts according to 'sentiments', such as 'intention' and 'polarity'. *Text summarization* traditionally only extracted important sentences from single documents but now strives to automatically generate an abstract from multiple-source documents.

These trends indicate that recent tasks have increasingly demanded a text to be interpreted semantically or contextually. In other words, solving recent tasks with higher performance has required methods that can handle richer types of linguistic information.

Conventionally, in the field of NLP, a set of words, called *bag-of-words*, is the most widely used model for representing features of texts. However, it is known that bag-of-words models lack many of the linguistic features found in texts. It is widely accepted that the lack of structures in bag-of-words models has led to inadequate performance in recent tasks. Therefore, methods are needed that are capable of handling richer linguistic information within texts.

For these reasons, this dissertation proposes a methodology that is capable of handling richer structural information derived from syntactic and semantic analysis. I formalize all of my proposed methods within the framework of *kernel*

[*] Doctoral Dissertation, Department of Information Processing, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DD0361207, March 24, 2005.

methods. More specifically, the proposed methods are defined as *kernel functions*, a very generalized mathematical framework developed in the *Machine Learning* field. Since this formalization embeds the proposed methods in a generalized framework, I can apply these methods to many other tasks, and even to other research fields.

This dissertation discusses the following several methodologies that might lead to a substantial improvement with respect to recent tasks in the application areas of NLP. They are:

1. Effectively handling different levels of word attribute, such as words, semantic information obtained from dictionaries and part-of-speech

2. Effectively handling richer structural information derived from integrated syntactic and semantic analysis

3. The effect of statistical feature selection for structural features.

Chapters, 3 to 5, respectively, address the above three topics.

Before moving on to these topics I first describe in Chapter 1 the present state of tasks in NLP application areas to provide the background to my work.

In Chapter 2, I explain the concepts necessary for understanding the dissertation: namely kernel methods and kernels for discrete structures, *discrete kernels*, in theory and mathematical formalism. This paves the way for embedding the proposed methods in the framework of kernel functions. I also explain some specific examples of kernel methods and discrete kernels, i.e., *Support Vector Machines*, as well as *sequence kernels* and *tree kernels*. Then, in the last part of this chapter, I describe *sentiment classification* tasks, which I use to evaluate the effect of the proposed method in the subsequent chapters.

Chapter 3 proposes a feature extraction method named *Word Attribute N-gram*. Unlike the bag-of-words representation, the proposed method can handle several levels of word information. Moreover, this method deals not only with a set of word attributes, but also with conjunctions, or N-grams, of word attributes which are expected to capture some important linguistic expressions. I assume that these linguistic expressions are more effective than single words or attributes. Moreover, I show the relationship between the word attribute N-gram

method and *sequence kernels*, because the word attribute N-gram method can be rewritten into the framework of kernels. In my experiments, I clarify the effects of the proposed method and verify the effects of each feature on given tasks.

Chapter 4 proposes *Hierarchically Structured Graph Kernels*, a method dealing with integrated structural information that reflects the results of syntactic and semantic analysis within text. I assume that this richer structural information gives clues of much higher quality for solving the target tasks. I define the proposed method as kernels on a certain class of graph, called a *hierarchically structured graph*, which is a graph with a recursive hierarchical structure constructed by using subgraphs and edges from vertices to subgraphs. Experiments demonstrate the performance of this method compared with other discrete kernels, which can only deal with restricted syntactic and semantic information within text.

Chapter 5 proposes a statistical feature mining method for discrete kernels, such as sequence and tree kernels. Unfortunately, previous experiments have shown that in some cases there is a critical issue with discrete kernels, especially in NLP tasks. That is, the *over-fitting problem* arises due to too many and sparse but redundant features that are processed implicitly in these kernels. As a result, the machine learning approach may not be trained efficiently. Conventionally, this issue is addressed by eliminating large substructures from the set of features used. However, the main reason for using discrete kernels is that we aim to use structural features easily and efficiently. Therefore, I propose a new approach based on statistical feature mining that avoids over-fitting without lacking any statistically important features. Moreover, I embed our proposed feature selection method into an original kernel calculation process by using *sub-structure mining* algorithms, thus allowing for efficient computation. Experiments are undertaken on certain sentiment classification tasks to confirm the problem with a conventional method and to evaluate the effect of the proposed method.

Finally, I summarize the dissertation and describe possible future directions in Chapter 6.

Keywords:

natural language processing, structured text, kernel methods, discrete kernels, hierarchically structured graph, feature mining

Contents

List of Figures

List of Tables

1. Introduction

Statistical *Natural Language Processing (NLP)* and machine learning based NLP have gained great popularity since the late 1980s. This trend is caused by the rapid growth of the Internet and the progress made in computer technology including the huge increase in CPU speed and the improved capacity-cost of storage devices. These developments have allowed us to obtain a large number of natural language corpora, and to apply complex but powerful machine learning methods to large scale documents.

This recent success in the development of statistical NLP has led to improvements in fundamental text analysis; such as part-of-speech (POS) tagging, phrase chunking, dependency analysis and parsing. Using these components as fundamental building blocks, many NLP researchers have become interested in analyzing text "semantically" or "contextually". For example, *named entity* tagging, *semantic role* tagging and *discourse parsing* are being investigated in the NLP fields.

This move towards taking contextual or semantic information into account has occurred in application areas of NLP such as text classification, text summarization, information retrieval and question answering. That is, the research interest in application areas of NLP[1] has moved on to more challenging tasks.

For example, while information retrieval has been studied for a long time, question answering is now being very actively investigated in the NLP field. The purpose of these tasks is the same: to extract useful information from a large number of documents. However, in contrast to information retrieval, which finds relevant documents to a given query, the question-answering task requires the extraction of the exact answer to a given question. If a question answering system receives the question "When was Queen Victoria born?" it should answer "in 1832". Question Answering has been studied intensively throughout the world since the start of the *Question Answering Track* at *Text REtrieval Conference (TREC)* [Voorhees99] in U.S., and now the *Question Answering Challenge (QAC)* [Fukumoto03] has been held in Japan every year or year and a half.

[1] Hereafter I use the abbreviation "text processing tasks" for tasks in NLP application areas.

In text summarization, the first step is to extract important sentences from single documents, but now, the generation of an abstract from multi-sources and multi-documents is being discussed. In addition, there are summarization workshops, called *Document Understanding Conference (DUC)* [Harman04] in U.S., and *Text Summarization Challenge (TSC)* [Fukushima01] in Japan, which foster the development of summarization techniques.

Even with text classification tasks, one of the traditional NLP tasks, the target classes have recently been diversified from topics such as 'sports' and 'economics' to the contents of texts such as 'polarity' or 'subjectivity', called *sentiment classification or sentiment analysis* [Pang02, Pang04]. Sentiment classification is defined as a task classifying texts into the *sentiment matter* contained in the texts, such as like vs. dislike, recommend vs. not recommend, and subjective vs. objective.

These trends indicate that recent tasks increasingly demand a text to be interpreted semantically or contextually. In other words, solving recent tasks with higher performance requires methods that can handle richer types of linguistic information.

Conventionally, in the field of NLP, a set of words, called *bag-of-words* [Salton75], is the most widely used method for representing the features of texts. However, it is known that the bag-of-words model lacks many of the linguistic features found in texts. It is widely accepted that the lack of structures in bag-of-words models leads to inadequate performance. This indicates that the bag-of-words model does not scale well to more demanding tasks. Thus, methods are required that are capable of handling richer linguistic information within texts.

For these reasons, the goals of this dissertation are as follows:

- To **propose efficient methodologies that are capable of handling rich information** derived from syntactic and semantic analysis
  This syntactic and semantic information can provide important information for understanding natural language and for tackling real tasks in the application areas of NLP.

- Then, to **confirm the effectiveness of the rich information** provided by the proposed method

Intuitively, we can say it is a better approach for dealing with richer information. However, no previous work has shown that richer structural information improves the performance of text processing, and so we do not know if dealing with richer information really improves performance. Therefore, I undertook an experiment in this dissertation to verify the fact by using real sentiment classification tasks.

I formalize all of my proposed methods within the framework of *kernel methods* [Vapnik95, Cristianini00]. More specifically, the proposed methods are defined as *kernel functions*, a very generalized mathematical framework developed in the field of *machine learning.* Since this formalization embeds the proposed methods in a generalized framework, I can apply the proposed methods to many other tasks, and even to other research fields.

To achieve the outlined goals, this dissertation discusses the following methodologies that are shown to lead to substantial improvement with respect to the recent tasks described above. They are:

1. Effectively handling different levels of word attribute, such as words, semantic information obtained from dictionaries and part-of-speech (Chapter 3)

2. Effectively handling richer structural information derived from integrated syntactic and semantic analysis (Chapter 4)

3. The effect of statistical feature mining for structural features (Chapter 5).

In chapter 3, I discuss the effect of handling simple linguistic information that is constructed by the gapped N-gram with different levels of word attributes. As noted above, text classification and information retrieval have mainly employed the bag-of-words model to represent texts. However, considering the tasks in sentiment classification, I assume that more detailed information is required for higher performance. In order to test this assumption, I propose a feature extraction method called *word attribute N-gram*, which is capable of dealing with a combination of several levels of word attributes. More specifically, I take advantage of the information obtained from dictionaries. Thus, compared with the bag-of-words model, this method deals not only with a set of word attributes, but also with conjunctions, or N-grams, of word attributes, which are expected to capture some important linguistic expressions. In the experiments, I clarify the effect of the proposed method, which is equivalent to evaluating whether or not richer information can provide better performance. Additionally, I analyze the effect of each primitive feature to verify that the linguistic features are really effective.

Chapter 4 discusses the effect of richer structural information derived from an integrated syntactic and semantic analysis. Kernel methods [Vapnik95, Cristianini00] suitable for NLP have recently been devised. *Convolution Kernels* [Haussler99, Watkins99] demonstrate how to build kernels over discrete structures. Since natural language data take the form of sequences of words and parsed trees, *discrete kernels*, such as sequence kernels [Lodhi02] and tree kernels [Collins01] have been developed. Using the methodology of discrete kernels, I propose *hierarchically structured graph (HS-graph) kernels*, that is, kernels that can handle integrated analysis results of syntactic and semantic information within text. The accuracies of fundamental NLP tools, such as POS taggers, NP chunkers, named entities taggers and dependency analyzers, have been improved to the point where they can realize practical application tasks. It is natural to expect that dealing with all of this integrated richer syntactic and semantic information provides much better performance than conventional methods. I show that the results obtained using the proposed method are significantly better than those obtained with conventional methods while evaluating using sentiment classification tasks.

Finally, Chapter 5 discusses the topic of feature selection for discrete structures. Convolution kernels [Haussler99, Watkins99], such as sequence and tree kernels, are advantageous in terms of both the concept and accuracy of many NLP tasks. However, unfortunately experiments have shown that in some cases there is a critical issue with convolution kernels, especially in NLP tasks [Collins01, Cancedda03, Suzuki03b]. That is, the *over-fitting problem* arises due to too many and sparse but redundant features that are processed implicitly in these kernels. As a result, the machine learning approach may not be trained efficiently. To solve this issue, we generally eliminate the effect of large sub-structures from the feature set. However, the main reason for using discrete kernels is that we aim to use structural features easily and efficiently. If its use is limited to only very small structures, it does not take full advantage of using these kernels. Therefore, I propose a new approach based on statistical feature mining that avoids over-fitting without missing any statistically important features. Moreover, I embed our proposed feature selection method in an original kernel calculation process by using *sub-structure mining* algorithms, thus allowing efficient computation. Experiments are undertaken on some sentiment classification tasks to confirm the problem with a conventional method and to evaluate the effect of the proposed method.

2. Preliminary

As mentioned in Chapter 1, the main topic of this dissertation is an evaluation of the effect of handling richer information within text for text processing. I have formalized all of the proposed methods in this dissertation (Chapters 3, 4 and 5) in the framework of *kernels*.

Therefore, before explaining the proposed methods, this chapter provides the background knowledge needed to understand the dissertation: Kernel methods and discrete kernels in theory and mathematical formalism.

Section 2.1 explains the essence of the kernel methods, and then provides one specific example, that is, Support Vector Machine. Section 2.2 introduces the framework of discrete kernels, which is the key idea behind all the methods proposed in the dissertation. The last part of this section briefly describes some examples of specific discrete kernels, that is, sequence and tree kernels, which are referred to many times in this dissertation, especially for the comparison methods in the experimental part. Finally, Section 2.3 introduces the latest tasks in the text classification area called *sentiment classification*. The experiments in this dissertation are done on some sentiment classification tasks to evaluate the performance of the proposed methods, and to clarify the effect of the richer information derived by the proposed methods.

2.1 Kernel Methods

Kernel methods, such as Support Vector Machines [Vapnik95], constitute one of the most successful recent developments in machine learning research area [Herbrich02, Schölkof01]. They have been applied to a number of real world problems and are now considered to provide state-of-the-art performance in various domains, such as bioinformatics and natural language processing (NLP).

Kernel methods are characterized by the use of a *kernel function* $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$, where $\mathcal{X}$ can be any non-empty set. Intuitively, a kernel function $K(x, x')$ expresses a degree of similarity between two input objects $x, x' \in \mathcal{X}$. Generally, kernel functions are required to satisfy the following two conditions: (1) symmetric $K(x, x') = K(x', x)$ and (2) positive definite $\sum_{i=1}^{N} \sum_{j=1}^{N} c_i c_j K(x_i, x_j) \geq 0$, for any $N \geq 1$, $\{x_i | x_i \in X, i = \{1, \ldots, N\}\}$ and $\{c_i | c_i \in \mathbb{R}, i = \{1, \ldots, N\}\}$. Under these two conditions, it is known from Mercer's theorem [Vapnik98] that mapping, $\phi : \mathcal{X} \rightarrow \mathcal{F}$, from $\mathcal{X}$ to some (Hilbert) space $\mathcal{F}$, usually called the *feature space*, exists, such that $K(x, x') = \phi(x) \cdot \phi(x')$ for all $x, x' \in \mathcal{X}$. That is,

$$\begin{aligned} \sum_{i=1}^{N} \sum_{j=1}^{N} c_i c_j K(x_i, x_j) &= \sum_{i=1}^{N} \sum_{j=1}^{N} c_i c_j \phi(x_i) \cdot \phi(x_j) \\ &= \sum_{i=1}^{N} c_i \phi(x_i) \sum_{j=1}^{N} c_j \phi(x_j) \\ &= \sum_{i=1}^{N} c_i \phi(x_i) \sum_{i=1}^{N} c_i \phi(x_i) \\ &\geq 0 \end{aligned} \tag{1}$$

Therefore, calculating $K(x, x')$ is equivalent to mapping two input objects x and x' into feature space, then calculating the inner products between x and x' in the feature space. This means that kernel functions can be used even if the nature of the corresponding mapping ϕ is not known. The computational attractiveness of kernel methods originates from the fact that they can be applied without suffering the high cost of explicitly computing the mapped data, $\phi(x)$.

In other words, kernel methods allow us to handle complex and high (even infinite) dimensional feature space without increasing the computational complexity. This approach usually leads to better performance than the conventional non-kernel-based method.

Support Vector Machine

I continue to introduce *Support Vector Machines (SVM)* [Vapnik98, Cortes95] which are now very well known kernel-based machine learning methods. An SVM offer the following advantages over conventional statistical learning algorithms (i.e., decision tree learning, maximum entropy method):

1. high generalization performance even with feature vectors of high dimensions (See sections 2.1.1 and 2.1.2), and

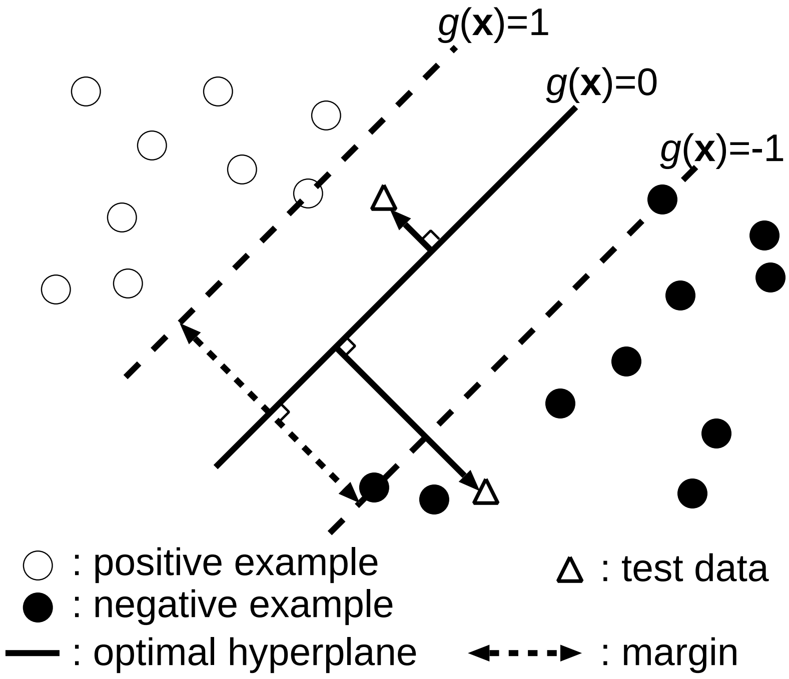


Figure 1. Support Vector Machines

2. the ability to manage *kernel functions* (See section 2.2)

SVM starts with a set of l training data $(x_1, y_1), \cdots, (x_l, y_l)$ where $x_i(\in \mathcal{X})$ is a non-empty set and $y_i(\in \{+1, -1\})$ is the class label of i-th data. Formally, we can define the classification problem as a learning and building process of the decision function $f : \mathcal{X} \to \{\pm 1\}$.

Basically, SVM tries to separate positive and negative examples by using linear hyper-planes,

$$\mathbf{w} \cdot x + b = 0 \qquad \mathbf{w} \in \mathbf{R}^n, b \in \mathbf{R}. \tag{2}$$

Figure 1 shows training examples linearly separated into two classes. In this figure,

$$y_i[(\mathbf{w} \cdot x) + b] \geq 1 \tag{3}$$

are called separating hyper-planes. The distance between separating hyper-planes is generally called *margin*. The basic idea of SVM is to maximize the margin between the positive and negative examples.

The distance from the separating hyper-plane to point x_i can be written as:

$$d(\mathbf{w}, b; x_i) = \frac{|\mathbf{w} \cdot x_i + b|}{||\mathbf{w}||}. \tag{4}$$

Thus, the margin between two separating hyper-planes can be written as:

$$\begin{aligned} & \min_{x_i : y_i = 1} d(\mathbf{w}, b; x_i) + \min_{x_i : y_i = -1} d(\mathbf{w}, b; x_i) \\ = & \min_{x_i : y_i = 1} \frac{|\mathbf{w} \cdot x_i + b|}{||\mathbf{w}||} + \min_{x_i : y_i = -1} \frac{|\mathbf{w} \cdot x_i + b|}{||\mathbf{w}||} \\ = & \frac{2}{||\mathbf{w}||} \end{aligned} \tag{5}$$

To maximize this margin, $||\mathbf{w}||$ should be minimized. In other words, this problem becomes the equivalent of solving the following optimization problem:

$$\begin{aligned} \min_{\mathbf{w}, b} & : \frac{1}{2}||\mathbf{w}||^2 \\ \text{s.t.} & : y_i[(\mathbf{w} \cdot x) + b] \geq 1 \quad i = 1, \ldots, l. \end{aligned} \tag{6}$$

In general, it is not necessary to separate training examples into individual classes. Slack Variables $\xi_i (\geq 0)$ are introduced for misclassification errors. Taking slack variables into account, optimization problem (5) can be reformulated as follows:

$$\begin{aligned} \min_{\mathbf{w}, b, \xi} & : \frac{1}{2}||\mathbf{w}||^2 + C \sum_{i=1}^{l} \xi_i \quad \xi_i \geq 0. \\ \text{s.t.} & : y_i[(\mathbf{w} \cdot x) + b] \geq 1 \quad i = 1, \ldots, l. \end{aligned} \tag{7}$$

The first term in equation (7) specifies the size of the margin and the second term represents the cost of the misclassification. C is the parameter that defines the balance of two quantities. With increasing C, a greater number of classification errors are neglected.

The optimization problem (7) can be rewritten into a dual form problem using Lagrangian multipliers $\alpha_i \geq 0$:

$$\begin{aligned} \max_{\alpha} &: \sum_{i=1}^{l} \alpha_i + \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j (x_i, x_j) \\ \text{s.t.} &: \sum_{i=1}^{l} \alpha_i y_i = 0 \quad (0 \leq \alpha_i \leq C). \end{aligned} \tag{8}$$

In this dual form problem, x_i with non-zero α_i is called a *support vector (SV)*. For the SVM, $\mathbf{w}$ and b can thus be expressed as follows

$$\begin{aligned} \mathbf{w} &= \sum_{i:x_i \in SVs} \alpha_i y_i x_i \\ b &= \mathbf{w} \cdot x_i - y_i. \end{aligned} \tag{9}$$

The support vectors lie on the separating hyper-planes. Finally, the decision function $f : \mathcal{X} \rightarrow \{\pm 1\}$.

The decision function $f(x)$ can be written as:

$$f(x) = \text{sgn}(g(x)) \tag{10}$$

$$g(x) = \sum_{i}^{l} \alpha_i y_i (x_i \cdot x) + b \tag{11}$$

$$= \mathbf{w} \cdot x + b. \tag{12}$$

Using a kernel function, we can rewrite Equation (11) as:

$$g(\mathbf{x}) = \sum_{i}^{l} \alpha_i y_i K(x_i, x) + b. \tag{13}$$

Moreover, in the SVM training part, we can rewrite Equation (8) as:

$$\max_{\alpha} : \sum_{i=1}^{l} \alpha_i + \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \cdot K(x_i, x_j). \tag{14}$$

2.1.1 Generalization Ability

Generalization analysis of classifiers is concerned with determining the factors that affect the accuracy of a classifier. One of the most popular way is to estimate the bounds on the generalization error, that is the probability of misclassifying a randomly chosen example. According to [Vapnik98], it is possible to estimate an upper bound of test error $R[f]$, called *risk*, with respect to the training error $\hat{R}[f]$, called *empirical risk* hold following inequality:

$$R[f] \leq \hat{R}[f] + \sqrt{\frac{h(\ln\frac{2l}{h}+1)-\ln\frac{\eta}{4}}{l}}.$$

This inequality hold with a certain probability $1-\eta$. In this inequality, h is a non-negative integer and a measure of the complexity of the given decision function f, called the *VC dimension*. That is, in order to minimize the *risk*, we have to minimize the empirical risk as well as the VC dimension.

It is known that the following theorem holds for VC dimension and margin.

Theorem 1 *Suppose that X is the ball of radius R in $\mathbb{R}^n$, where $X = \{x \in \mathbb{R}^n : ||x|| \leq R\}$. Suppose n as the dimension of given training samples and M as the margin. Then, VC dimension h are bounded by*

$$h = \min\left(\frac{R^2}{M^2}, n\right) + 1$$

VC dimension suggests that maximizing the margin can lead to a better generalization performance. Moreover, this result shows that the VC dimension of large margin hyperplanes need not depend on the dimension of the feature space, when the dimensions of feature space is high enough. This result indicates that SVMs have a high generalization performance even with feature vectors of high dimensions.

2.1.2 Dimensions of feature space

In general, the primal problem P is:

$$\min_f : \{\mathcal{L}(f, \{x_i, y_i\}) + \Omega(||f||)\}$$

where $i = 1...l$. Note that the SVM can also be written in this form according to Equation (6):

$$\min_f : \sum_{i=1}^{l} \max(0, 1 - y_i f(x_i)) + \frac{\lambda}{2}||f||^2.$$

If the problem satisfies the following two conditions: (1) the loss function $\mathcal{L}$ is pointwise, that is, $\mathcal{L}(f, \{x_i, y_i\}) = \mathcal{L}(\{x_i, y_i, f(x_i)\})$ which only depends on $\{f(x_i)\}$, the values of f at the data points, and (2) $\Omega(\cdot)$ is monotonically increasing. Then the *Representer Theorem* [Kimeldorf71] states that every minimizer of P admits a representation of the form

$$f(\cdot) = \sum_{i=1}^{l} \alpha_i K(\cdot, x_i).$$

therefore, the optimal f^* is a linear combination of a finite set of functions given by the data. This is a powerful result. It shows that although we search for the optimal solution in an infinite-dimensional feature space, adding the regularization term reduces the problem to finite-dimensional.

2.2 Discrete Kernels

One characteristic of the kernel approach is that it not only provides high accuracy and the concept of faster calculation for non-linear classifiers realized by the kernels but also allows us to *design a kernel function* suited to modeling the task at hand. In fact, the designing of appropriate kernel functions for specific tasks is now the goal of many researchers in various domains, i.e. natural language processing and bio-informatics. In particular, as most real world data is represented not as numerical vectors, but as discrete structures such as graphs (including sequences and trees), research into designing kernel functions has begun to focus on kernels on discrete structures.

Convolution Kernels [Haussler99, Watkins99] is the first method to provide a general framework for building kernels over discrete structures, such as sequences, trees and graphs. This framework defines the kernel function between input objects (no longer restricted to vectors) 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, a convolution kernel $K(X, Y)$ enumerate all sub-structures occurring in X and Y and then determines their inner product,

$$K(X, Y) = \langle \phi(X), \phi(Y) \rangle = \sum_i \phi_i(X) \cdot \phi_i(Y). \tag{15}$$

ϕ represents the feature mapping from the discrete object to the feature space; that is, $\phi(X) = (\phi_1(X), \ldots, \phi_i(X), \ldots)$.

Subsequently, a number of kernels for discrete structures, such as sequences, trees and graphs, have been successively proposed in the last five years. For example, since natural language data take the form of sequences of words or parsed trees, sequence kernels [Lodhi02, Cancedda03] and tree kernels [Collins01, Kashima02], have been developed in the *natural language processing (NLP)* field and shown to offer excellent results.

As an example, with sequence kernels [Lodhi02], input objects X and Y are sequences, and $\phi_i(X)$ corresponds to a sub-sequence. In the case of tree kernels [Collins01], X and Y are trees, and $\phi_i(X)$ represents a sub-tree. When implemented, these kernels can be efficiently calculated in quadratic time by using the dynamic programming (DP) technique.

Other examples in the area of bio-informatics, sequence kernels [Jaakkola00, Leslie04] and graph kernels [Kashima03] have been proposed mainly for classifying protein or chemical compounds. Other graph kernels [Gärtner03] have been introduced in the machine learning field. The developments generated by these researches have realized ways of dealing with discrete data in original and natural representation.

Since the size of the input objects is usually not a constant, the kernel value over discrete structures is sometimes normalized using the following equation:

$$\hat{K}(X, Y) = \frac{K(X, Y)}{\sqrt{K(X, X) \cdot K(Y, Y)}}. \tag{16}$$

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

2.2.1 Sequence Kernels

Many kinds of sequence kernels have been proposed for a variety of different tasks. This section basically follows the framework of *word sequence kernels* [Cancedda03], and processes *gapped word sequences* to yield the kernel value.

Let Σ be a set of finite labels[2] , and Σ^n be a set of possible (label) sequences whose "sizes" are n that are constructed by labels in Σ. Then, let Σ_1^n be a set of sequences whose "sizes" are n or less, that is,

$$\Sigma_1^n = \bigcup_{i=1}^{n} \Sigma^i. \tag{17}$$

Moreover, let Σ_1^* be a set of sequences of any size,

$$\Sigma_1^* = \bigcup_{i=1}^{\infty} \Sigma^i. \tag{18}$$

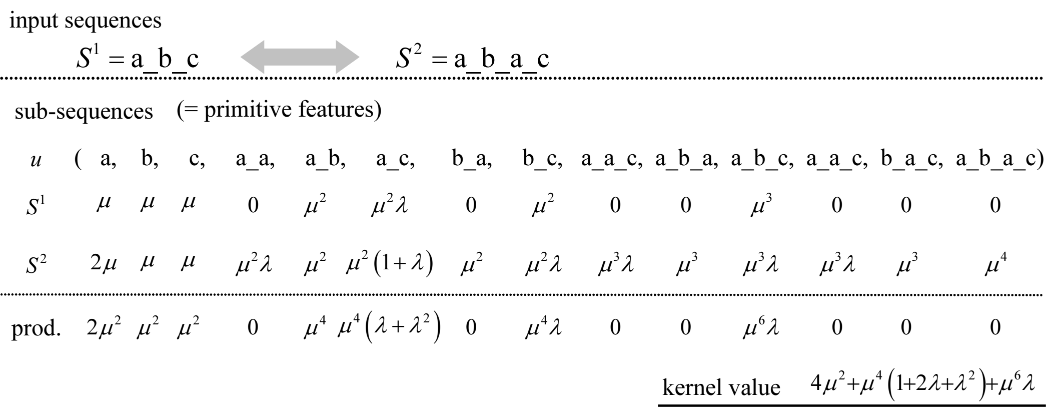
In this dissertation, "size of sub-structure" indicates the number of labels in the sub-structure. Namely, for a sequence, size n means the length of the sequence is n. S can represent any sequence. s_i represents the ith label in S. Therefore, a sequence S can be written as $S = s_1 \dots s_i \dots s_{|S|}$, where $|S|$ represents the length of S. I introduce u to represent a sub-structure, that is, a sub-sequence in the sequence, where $u = u_1 \dots u_{|u|}$. If u is contained in sub-sequence $S[i:j] \overset{\text{def}}{=} s_i \dots s_j$ of S (allowing the existence of skipping some labels), the position of u in S is written as $\mathbf{i} = (i_1 : i_{|u|})$. This means $s_{i_j} = u_j$ holds for any u_j. The length of $S[\mathbf{i}]$ is $l(\mathbf{i}) = i_{|u|} - i_1 + 1$. For example, if $u =$ 'a''b' and $S =$ 'c''a''c''b''d', where 'a', 'b' 'c' and 'd' represent a label, then $\mathbf{i} = (2:4)$ and $l(\mathbf{i}) = 4 - 2 + 1 = 3$.

By using the above notations, the feature mapping ϕ for a sequence S is given by defining the u coordinate $\phi_u(S)$ for each $u \in \Sigma_1^*$, that is,

$$\phi_u(S) = \sum_{\mathbf{i}:u=S[\mathbf{i}]} \mu^{|u|} \lambda^{\zeta(\mathbf{i})}, \tag{19}$$

where λ and μ can be any constants. μ is usually defined as a decay factor of sub-sequence size, $0 \le \mu \le 1$, and λ is introduced as a decay factor of the gap in the sub-sequence, $0 \le \lambda \le 1$, while $\zeta(\mathbf{i}) = l(\mathbf{i}) - |u|$.

[2] In this dissertation, I use the term as a primitive symbol and assume Σ to be the set of labels, not *symbols* in the standard way.



input sequences

S^1 = a_b_c ⟷ S^2 = a_b_a_c

sub-sequences (= primitive features)

u	(a,	b,	c,	a_a,	a_b,	a_c,	b_a,	b_c,	a_a_c,	a_b_a,	a_b_c,	a_a_c,	b_a_c,	a_b_a_c)
S^1	μ	μ	μ	0	μ^2	$\mu^2\lambda$	0	μ^2	0	0	μ^3	0	0	0
S^2	2μ	μ	μ	$\mu^2\lambda$	μ^2	$\mu^2(1+\lambda)$	μ^2	$\mu^2\lambda$	$\mu^3\lambda$	μ^3	$\mu^3\lambda$	$\mu^3\lambda$	μ^3	μ^4
prod.	$2\mu^2$	μ^2	μ^2	0	μ^4	$\mu^4(\lambda+\lambda^2)$	0	$\mu^4\lambda$	0	0	$\mu^6\lambda$	0	0	0

kernel value $4\mu^2+\mu^4(1+2\lambda+\lambda^2)+\mu^6\lambda$

Figure 2. Example of sequence kernel output

These features measure the number of occurrences of sub-sequences in S, weighting them according to their sizes. Hence, the inner product of the feature vector for two sequence S^1 and S^2 gives a sum over all common sub-sequences weighted according to their frequency of occurrence and size. Thus, sequence kernels can be defined as:

$$
\begin{aligned}
K^{\text{SK}}(S^1, S^2) &= \sum_{u\in\Sigma_1^*} \phi_u(S^1)\cdot\phi_u(S^2) \\
&= \sum_{u\in\Sigma_1^*} \sum_{\mathbf{i}:u=S^1[\mathbf{i}]} \sum_{\mathbf{j}:u=S^2[\mathbf{j}]} \mu^{|u|}\mu^{|u|}\lambda^{\zeta(\mathbf{i})}\lambda^{\zeta(\mathbf{j})}. \qquad (20)
\end{aligned}
$$

Figure 2 shows a simple example of the output of this kernel.

In general, the number of features $|\Sigma_1^*|$, which is the dimension of the feature space, becomes very high. A naive computation of these features would involve $O(|\sum^n|)$ time and space. It is therefore computationally infeasible to calculate Equation (20) directly.

Therefore, to solve this problem, an efficient recursive calculation algorithm has been introduced in [Lodhi02, Cancedda03]. In order to clarify the discussion in this dissertation, I redefine the sequence kernels with the original notation.

The sequence kernel can be written as follows:

$$K^{\text{SK}}(S^1, S^2) = \sum_{i=1}^{|S^1|} \sum_{j=1}^{|S^2|} J(S_i^1, S_j^2). \tag{21}$$

where S_i^1 and S_j^2 represent the sub-sequences $S_i^1 = s_1^1, s_2^1, \ldots, s_i^1$ and $S_j^2 = s_1^2, s_2^2, \ldots, s_j^2$, respectively.

Let $J(S_i^1, S_j^2)$ be a function that returns the value of common sub-sequences if $s_i^1 = s_j^2$:

$$J(S_i^1, S_j^2) = J'(S_i^1, S_j^2) \cdot I(s_i^1, s_j^2) + I(s_i^1, s_j^2). \tag{22}$$

$I(s_i^1, s_j^2)$ is a function that returns a matching value between s_i^1 and s_j^2:

$$I(s_i^1, s_j^2) = \mu \cdot \delta(s_i^1, s_j^2). \tag{23}$$

Usually $\delta(s_i^1, s_j^2)$ is defined as an indicator function that returns 1 if $s_i^1 = s_j^2$, and 0 otherwise.

Then, $J'(S_i^1, S_j^2)$ and $J''(S_i^1, S_j^2)$ are introduced to calculate the common *gapped* sub-sequences between S_i^1 and S_j^2.

$$J'(S_i^1, S_j^2) = \begin{cases} 0 \quad \text{if} \quad j = 0, \\ \lambda J'(S_i^1, S_{j-1}^2) + J''(S_i^1, S_{j-1}^2) \quad \text{otherwise} \end{cases} \tag{24}$$

$$J''(S_i^1, S_j^2) = \begin{cases} 0 \quad \text{if} \quad i = 0, \\ \lambda J''(S_{i-1}^1, S_j^2) + J(S_{i-1}^1, S_j^2) \quad \text{otherwise} \end{cases} \tag{25}$$

Thus, considering only non-gapped sub-sequences, namely $\lambda = 0$ is set, then $J(S_i^1, S_j^2)$ can be simply written as:

$$J(S_i^1, S_j^2) = J(S_{i-1}^1, S_{j-1}^2) \cdot I(s_i^1, s_j^2). \tag{26}$$

If we calculate Equations (22) to (25) recursively, Equation (21) provides exactly the same value as Equation (20). The complexity of the efficient calculation comes down to $O(|S^1||S^2|)$.

Additionally, if we are only interested in sub-structures whose sizes are n or less, then Equations (21) to (25),respectively, can be rewritten as follows:

$$K^{\mathrm{SK}}(S^1, S^2) = \sum_{m=1}^{n} \sum_{i=1}^{|S^1|} \sum_{j=1}^{|S^2|} J_m(S^1_i, S^2_j). \tag{27}$$

$$J_m(S^1_i, S^2_j) = J'_{m-1}(S^1_i, S^2_j) \cdot I(s^1_i, s^2_j). \tag{28}$$

$$J'_m(S^1_i, S^2_j) = \begin{cases} 1 \quad \text{if} \quad m = 0, \\ 0 \quad \text{if} \quad j = 0 \text{ and } m > 0, \\ \lambda J'_m(S^1_i, S^2_{j-1}) + J''_m(S^1_i, S^2_{j-1}) \quad \text{otherwise} \end{cases} \tag{29}$$

$$J''_m(S^1_i, S^2_j) = \begin{cases} 0 \quad \text{if} \quad i = 0, \\ \lambda J''_m(S^1_{i-1}, S^2_j) + J_m(S^1_{i-1}, S^2_j) \quad \text{otherwise} \end{cases} \tag{30}$$

In this case, the complexity becomes $O(n|S^1||S^2|)$.

2.2.2 Tree Kernels

As well as sequence kernels, many kinds of tree kernels have been proposed for a variety of different tasks. This dissertation basically follows the framework of [Kashima02]. Namely a *labeled ordered tree kernel* that is introduced as the most general tree kernel framework.

In the case of a tree, "size" means the number of vertices (nodes) in the sub-tree. Let T be a tree and $u \in \Sigma_1^*$ be any sub-tree. Therefore, in the case of tree kernels, Σ_1^* represents a set of all possible (sub-)trees and $|u|$ denotes the number of vertices in sub-tree u.

By using the same notations of sequence kernels, the feature mapping ϕ for a tree T is given by defining the u coordinate $\phi_u(T)$ for each $u \in \Sigma_1^*$, that is,

$$\phi_u(T) = \sum_{\mathbf{i}:u=T[\mathbf{i}]} \mu^{|u|} \lambda^{\zeta(\mathbf{i})}, \tag{31}$$

where $\mathbf{i}$ represents a set of vertex indexes that construct u and $\zeta(\mathbf{i}) = l(\mathbf{i}) - |u|$ represents the same value as with sequence kernels, that is, the number of gapped

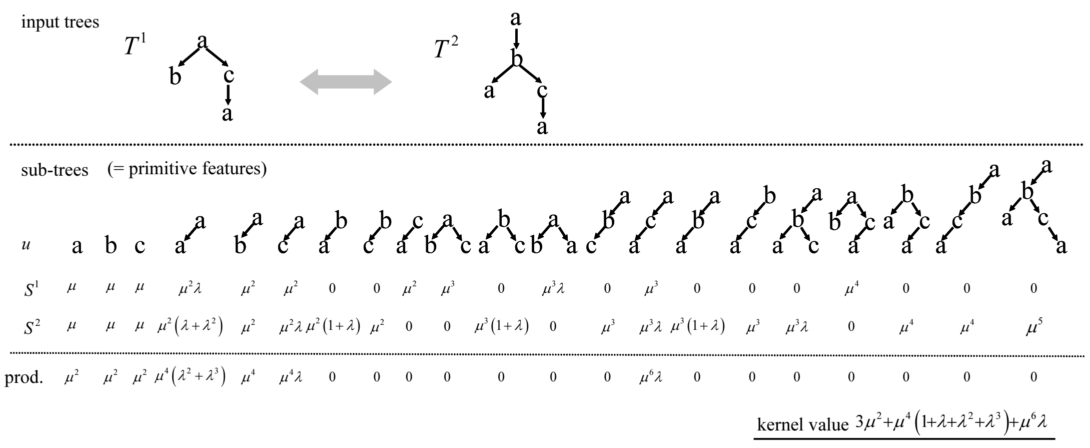

input trees
T^1
T^2
sub-trees (= primitive features)
u
S^1: μ μ μ $\mu^2\lambda$ μ^2 μ^2 0 0 μ^2 μ^3 0 $\mu^3\lambda$ 0 μ^3 0 0 0 μ^4 0 0 0
S^2: μ μ μ $\mu^2(\lambda+\lambda^2)$ μ^2 $\mu^2\lambda$ $\mu^2(1+\lambda)$ μ^2 0 0 $\mu^3(1+\lambda)$ 0 μ^3 $\mu^3\lambda$ $\mu^3(1+\lambda)$ μ^3 $\mu^3\lambda$ 0 μ^4 μ^4 μ^5
prod.: μ^2 μ^2 μ^2 $\mu^4(\lambda^2+\lambda^3)$ μ^4 $\mu^4\lambda$ 0 0 0 0 0 0 0 $\mu^6\lambda$ 0 0 0 0 0 0 0
kernel value $3\mu^2+\mu^4(1+\lambda+\lambda^2+\lambda^3)+\mu^6\lambda$

Figure 3. Example of tree kernel output

vertices in the sub-tree. In the case of tree kernels, the existence of a “gap” allows the structures of the subtrees to be elastic.

Therefore, by using the above notation, tree kernels (TK) can be defined as being exactly the same as sequence kernels, namely:

$$K^{\text{TK}}(T^1, T^2) = \sum_{u \in \Sigma_1^*} \phi_u(T^1) \cdot \phi_u(T^2)$$
$$= \sum_{u \in \Sigma_1^*} \sum_{\mathbf{i}: u = T^1[\mathbf{i}]} \sum_{\mathbf{j}: u = T^2[\mathbf{j}]} \mu^{|u|} \mu^{|u|} \lambda^{\zeta(\mathbf{i})} \lambda^{\zeta(\mathbf{j})}. \quad (32)$$

Figure 3 shows a simple example of the output of a tree kernel.

However, the definition of efficient computation is different. As well as sequence kernels, the number of features $|\Sigma_1^*|$ becomes very large, and a naive computation of Equation (32) also takes $O(|\sum^n|)$ time and space. An efficient recursive calculation algorithm has been introduced in [Kashima02].

An efficient computation of tree kernels can be written as follows:

$$K^{\text{TK}}(T^1, T^2) = \sum_{t_i^1 \in V(T^1)} \sum_{t_j^2 \in V(T^2)} C(t_i^1, t_j^2), \quad (33)$$

where $V(T)$ is a function that returns a set of vertices $\{t_i | 1 \le i \le |V(T)|\}$ in T, where $|V(T)|$ represents the number of vertices in T.

Let $H_{t_i^1,t_j^2}(k,l)$ be the sum of the products of occurrences each sub-tree appears at t_i^1 and t_j^2 when we consider only the vertices up to the i-th child of t_i^1 and the vertices up to the j-th child of t_j^2. Apparently,

$$C(t_i^1,t_j^2) = \begin{cases} I(t_i^1,t_j^2) & \text{if both } t_i^1 \text{ and } t_j^2 \text{ are leaves,} \\ C'_{t_i^1,t_j^2}(nc(t_i^1),nc(t_j^2)) & \text{otherwise} \end{cases} \tag{34}$$

where $nc(t_i^1)$ is the number of children of a vertex t_i^1. $I(t_i^1,t_j^2)$ is the same as for the sequence kernel, namely Equation (23).

Since all correspondences preserve the left-to-right ordering, $C'_{t_i^1,t_j^2}(k,l)$ can be recursively defined as

$$C'_{t_i^1,t_j^2}(k,l) = \begin{cases} 1 \quad \text{if} \quad k=0, l=0, \\ C'_{t_i^1,t_j^2}(k-1,l) + C'_{t_i^1,t_j^2}(k,l-1) + C'_{t_i^1,t_j^2}(k-1,l-1) \\ \quad +C'_{t_i^1,t_j^2}(k-1,l-1)\cdot C''(ch(t_i^1,k),ch(t_j^2,l)) \end{cases} \tag{35}$$

where $ch(t_i^1,k)$ represents the k-th child of vertices t_i^1.

Then, $C''(t_i^1,t_j^2)$ is introduced to calculate the common *gapped* sub-trees between t_i^1 and t_j^2.

$$\begin{aligned} C''(t_i^1,t_j^2) = {} & \lambda \sum_{l=1}^{nc(t_j^2)} C''(t_i^1,ch(t_j^2,l)) + \lambda \sum_{k=1}^{nc(t_i^1)} \cdot C''(ch(t_i^1,k)) \\ & -\lambda^2 \sum_{k=1}^{nc(t_i^1)} \sum_{l=1}^{nc(t_j^2)} C''(ch(t_i^1,k),ch(t_j^2,l)) + C(t_i^1,t_j^2). \end{aligned} \tag{36}$$

When considering only non-gapped sub-trees, namely $\lambda = 0$ is set, then $C'(T_i^1,T_j^2)$ can be simply written as:

$$C'_{t_i^1,t_j^2}(k,l) = \begin{cases} 1 \quad \text{if} \quad k=0, l=0, \\ C'_{t_i^1,t_j^2}(k-1,l) + C'_{t_i^1,t_j^2}(k,l-1) + C'_{t_i^1,t_j^2}(k-1,l-1) \\ \quad +C'_{t_i^1,t_j^2}(k-1,l-1)\cdot C(ch(t_i^1,k),ch(t_j^2,l)) \end{cases} . \tag{37}$$

Therefore, tree kernels can be efficiently computed $C(t_i^1,t_j^2)$ by th dynamic programming technique. Therefore, the computational complexity of tree kernels comes down to the quadratic time of the number of vertices in both trees, $O(|V(T^1)||V(T^2)|)$.

2.3 Sentiment Classification Tasks

Traditionally, the research field of text classification has dealt solely with the task settings involved in classifying texts into various subjects, that are generally called "topics". Most text classification research uses the bag-of-words features. This fact indicates that only information on the word itself is demanded from text classification tasks, and linguistic information of the text is unnecessary.

However, research interest has been moving to more challenging tasks, called *sentiment classification*. Sentiment classification is, of course, one of the subgroups of text classification tasks. Compared with the standard 'topic-based text classification', sentiment classification tasks require that text be classified according to the overall sentiment expressed in them, e.g,, like vs, dislike, recommend vs. not recommend. It is now known that these sentiment classification tasks are much more difficult than traditional topic-based text classification. This is because these tasks really demand linguistic information about the texts to realize higher performance.

To summarize the features of sentiment classification compared with topic-based text classification, I highlight out the following two points:

- Target texts are usually shorter than the documents that are dealt with in topic-based text classification: they are sometimes sentences or short passages.
- Target classes are strongly related to the meaning or context of the texts.

However, 'sentiment classification' is a term related to tasks that require text to be classified with respect to the *sentiment matter* contained in the texts. Therefore, there are several types of sentiment classification tasks. The following are examples that have already been discussed in the NLP field;

- *modality identification*
- *question classification*
- *subjectivity classification*

- *polarity classification.*

As regards modality identification, text must be classified in terms of modalities, such as “opinion”, “assertion”, and “description”. This setting assumes the use of an automatic text analysis system according to the context of the text.

Subjectivity classification requires the texts to be classified into “subjective” or “objective” and the polarity classification requires the texts to be classified into positive or negative statements. These two techniques are, for example, undertaken in commercial research that automatically gathers subjective opinions and classifies them into positive and negative market opinions with respect to certain products, services and brands that are dispersed throughout online texts such as product review articles, replies to questionnaires and messages in bulletin boards on the Internet. That is, we first collect large scale documents, and then employ ‘subjectivity classification’ to extract only subjective statements, and finally employ ‘polarity classification’ to verify whether the statements are positive or negative comments. Thus, these sentiment classification technologies are closely related to the demands of real use.

The question classification task requires questions to be classified into the classes that reflect the content of the question. More generally, this task is to visualize the situation of a dialogue system or an automatic question answering system, which must understand people’s intentions, such as requests, responses, agreement and disagreement.

As I have already explained, these sentiment classification tasks can be considered appropriate tasks designed to evaluate the effect of how well linguistic information is handled. This is why these tasks are used for evaluating the effects of the proposed method explained in the following chapter while this dissertation aims to clarify the effects of richer linguistic information.

3. Word Attribute $N-$gram: Extended Sequence Kernels

3.1 Introduction

Traditionally, research in text classification and information retrieval have mainly deal with the bag-of-words model for text, since it is the simplest but nevertheless powerful model with which to represent text. This model is based on the rough assumption that a set of words appearing in a text is entirely related to the meaning of the text. However, when we consider the text processing tasks described in Chapter 1, this approach requires a lot of information to perform well.

As a simple way of solving this issue, this chapter proposes a method for extracting features from text, called *word attribute N-gram*. It is widely accepted that "expressions" in the text characterize the text better than individual words. Here, an "expression" means the features constructed from a combination of words. The idea for the proposed method comes from this general intuition. More specifically, I take advantage of the various levels of lexical information. Each word generally has certain attributes such as part-of-speech (POS) and meaning. This dissertation abbreviates the attributes of a word such as part-of-speech, semantic information and word itself to "word attributes". These attributes are based on linguistics. The proposed method extracts a concatenation (or N-gram) of mixed word attributes. These N-grams can capture some important linguistic expressions. As a result, the proposed method is expected to improve the performance of text processing.

In the experiment part, I clarify the effect of the proposed method by comparing the performance of several feature sets. Moreover, I analyze the effect of each primitive N-gram to verify that the linguistic features are indeed effective. Finally, this chapter clarifies the effect of linguistic information with experimental results.

3.2 Word Attribute N-grams

This section explains the proposed feature extraction methods, namely word attribute N-grams. First, I introduce the primitive word attributes dealt with in this chapter. Then, I explain in detail how to extract word attribute N-grams from text.

3.2.1 Primitive Word Attributes

In this chapter, I use word, part-of-speech (POS) and *semantic category* for word attributes. Words and POSs are simply obtained from the results of a POS tagger.

The semantic categories of words are obtained from a dictionary. More specifically, we use the semantic attribute system of "Goi-Taikei — A Japanese Lexicon" [Ikehara97]. The semantic attribute system is a sort of hierarchical concept thesaurus represented as a tree structure in which each node is called a *semantic category*. An edge in the tree represents an *is_a* or *has_a* relation between two categories. The semantic attribute system is 12 levels deep and contains 2715 semantic category nodes. More than 300,000 Japanese words are linked to the category nodes. Figure 4 shows how to obtain the semantic category for each word. As shown in the examples in the figure, upper layer categories of each word are also given to the word. That is, if the word "Shinjuku" is located in the semantic category "Location", then semantic categories, "Entity" and "Noun" are also added as attributes of "Shinjuku".

3.2.2 Extraction Method

Figure 5 shows an example of word attribute N-grams. Note that in the case $N = 1$ means the word attribute itself. The proposed method simply extracts all possible concatenations of word attributes.

Then, I extend the proposed method to consider the arbitrary gap between each word attribute, since some text processing tasks require us to deal with gapped N-grams to improve the performance. To accomplish this extension, we introduce a special attribute that represents an "arbitrary" attribute. Figure 6

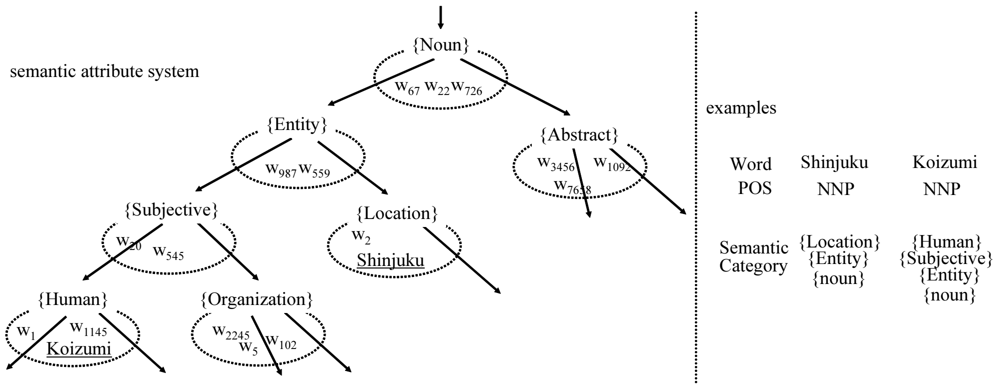


Figure 4. An example of making feature vectors for semantic categories

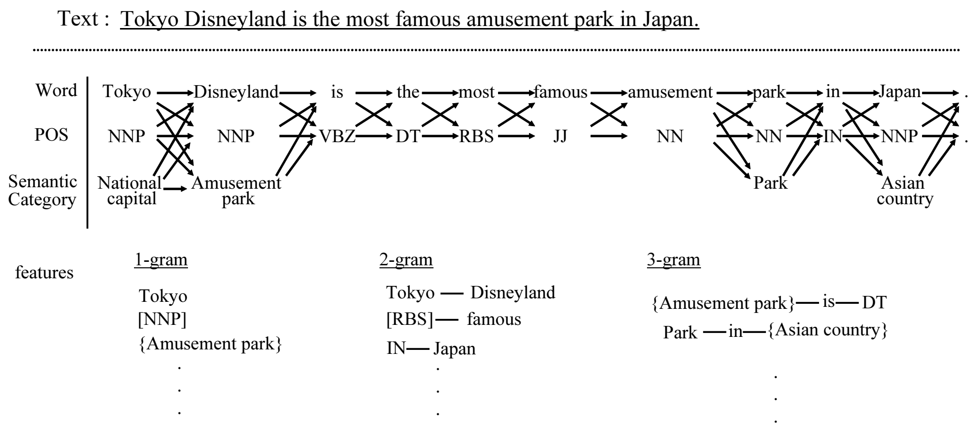

Text : Tokyo Disneyland is the most famous amusement park in Japan.
Word Tokyo Disneyland is the most famous amusement park in Japan .
POS NNP NNP VBZ DT RBS JJ NN NN IN NNP .
Semantic Category National capital Amusement park Park Asian country
features
1-gram
Tokyo
[NNP]
{Amusement park}
2-gram
Tokyo — Disneyland
[RBS]— famous
IN— Japan
3-gram
{Amusement park}— is— DT
Park —in—{Asian country}

Figure 5. Example of word attribute N−grams

shows an extended word attribute N-gram. All words are assumed to have a special word attribute represented as $(*)$.

As shown in the figure, the extended proposed method extracts all the word attribute N-grams including the word attribute $*$. However, $*$ can only appear in the middle of the N-gram, that is, $*$ never appears at the head or tail of N-grams. Since these are equivalent to an N-grams in which the head or tail $*$ has been removed, these N-grams are redundant. Additionally, consecutive $*$ should be

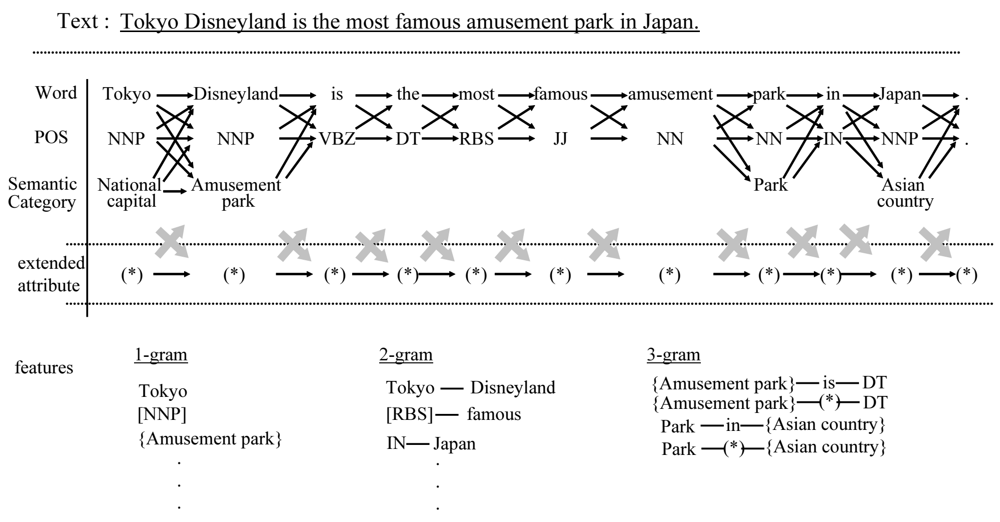


Figure 6. Example of extended word attribute $N-$grams

treated as one $*$ by definition, that is, word attribute N-grams of "A-$*$-C" and "A-$*$-$*$-C" are considered equivalantly. However, "A-C" and "A-$*$-C" are not considered to be equivalantly, since the latter has $*$ and the former has not.

We can easily imagine that the number of extended word attribute N-grams increases exponentially with N. Therefore, in real use, we introduce the threshold t for the number of appearances of each word attribute N-gram. We refuse to extract word attribute N-grams whose appearance are below the threshold. This process easily avoids an explosive increase in the number of features, since the number of appearances of most word attribute N-grams is 1 or very small. This phenomenon can easily be explained by the fact that the number of primitive word attribute is very large, the probability of appearing in an N-gram is very small. Thus, the introduction of threshold t for the appearance of N-grams is sufficient to control the set of features being dealt with.

Finally, in the proposed method, when word attribute N-grams are treated as features of machine learning, they all take a value of 1 or 0; 1 if it appears in the text, and 0 otherwise.

3.3 Experiments

This section verifies that the effectiveness of our proposed method by using text processing tasks. More precisely, I evaluate the proposed method from two different viewpoints, that is,

1. Effect of affinity with SVM classifier and word attribute N-gram
2. Effect of word attribute N-grams as features.

Typicaly, handling a large number of features has an adverse effect on the generalization performance of machine learning. However, SVM is known to be as robust for handling a large number of features. Although the proposed method extracts a huge number of combinational features, SVM can handle this. Thus, the first experiments clarify the effect of combining the proposed method with an SVM classifier.

Then, I verify the effect of word attribute N-gram itself by using SVM.

3.3.1 Data

We used the *question classification* data constructed by [Sasaki01] for question answering test collection. This data set has 10000 questions, and each question has just one question type. In this paper, we call such label a "correct label". I call the final output label from the classifier for each question an "estimated label".

This data set actually has 35 question types. However, we decided to deal only with question types that have more than 100 questions. The others may not learn efficiently by any of the machine learning method, because it has very few samples. Therefore, we deal with 17 question types: 16 question types that have more than 100 questions plus one new question type, "OTHER", which is constructed from all the questions except for the questions in the 16 question types.

Table 1 is a list of question types and the number of questions belonging to each question type.

Table 1. The number of samples used in the experiments

Question type	Abbreviation	# of questions
AGE	AGE	130
DATE	DATE	1885
EVENT	EVEN	165
LOCATION	LOCT	1530
MONEY	MONY	250
NORGANIZATION	NORG	140
NPERSON	NPER	365
ORGANIZATION	ORGN	1605
PERCENT	PCEN	190
PERIOD	PERI	260
PERSON	PERS	1615
PRODUCT	PROD	135
PTITLE	PTIT	270
SUBSTANCE	SUBS	130
TIME	TIME	125
TITLE	TITL	150
OTHER	OTHR	1055
Total		10000

3.3.2 Evaluation Method

Question classification is a task that gives one estimated question type for each given question including question type “OTHER”. The reason for including “OTHER” is that a question classification task is usually used under a setting where the given question is not always classified as a pre-defined question type.

In this settings, I redefine the question classification task that finds one optimal question type for a given question. This setting is derived because the data set used here has the most appropriate question type for each question.

To evaluate the final performance, I used five-fold cross-validation, where I used four-sets for training and the remaining set for the test and iterated this procedure five times by changing test set.

I used two different evaluation measures: one is the average accuracy between the correct and estimated labels for each question and the other is the average F-measure for each question type. The average label accuracy is selected to evaluate global performance of each method, and the average F-measure is used to evaluate detailed performance of each question type. By using these two different evaluation measures, we can discuss the performance of question classification both locally with each question type and globally.

3.3.3 Experimental Settings

Below I summarize the experimental settings.

1. Effect of affinity with SVM classifier and word attribute N-gram:
   rule based method, decision tree, maximum entropy and support vector machine

   (a) comparing the performance of multi-class classification methods:
   one vs. rest model and pairwise model

   (b) effect of kernel parameter:
   degree of polynomial kernels

   (c) learning curve

2. Effect of word attribute N-gram:
   comparison of, (1) word and semantic category, (2) word N-grams, semantic category N-grams, (3) word and semantic category N-grams, (4) combinations of word and semantic category N-grams and (5) combinations of word, semantic category and part-of-speech N-grams

   (a) effect of threshold t with word attribute N-gram

   (b) analysis of effective features for question classification

Effect of affinity with SVM classifier and word attribute

I tested the effect of affinity with the SVM classifier and word attribute N-gram.

This set of experiments compared the performance of our proposed method with different machine learning methods. More precisely, decision tree (C4.5) and maximum entropy methods (MEM) and support vector machine (SVM). [Zukerman01] reports a decision tree based question answering system and [Ittycheriah00, Ittycheriah01b, Ittycheriah01a] describe maximum entropy based methods. Note that the features for C4.5, MEM and SVM are exactly the same, that is, they are the features extracted by a word attribute N-gram. Moreover, as the baseline method, I evaluated the rule based method [Sasaki01] (RULE).

Note that the rule-based method used in this experiments is allowed to output duplicated labels with no ranking. Therefore, I conclude that this method estimated the correct label if one of the output labels is the correct label. That is, I evaluate the performance of the rule-based method by using a relaxed evaluation.

As regards the parameters, C in Equation (6) in Section 2.1 is set at 1 and the threshold t for word attribute N-gram is set at 30.

Since SVM is a binary class classification, we adopted the "one vs. rest model" [Schölkopf95, Blanz96] and adapted it for multi-class classification. This is derived from the result of a comparison of the multi-class classification model for SVM, that is, the one vs. rest with the pairwise model, which showed that the one vs. rest model provided better performance than the pairwise model.

Effect of word attribute N-gram

This set of experiments compares the efficiency of word attribute N-grams.

To accomplish this, we compared the performance of several sets of extracted features, that is, (1) bag-of-words and semantic category (W+S(N=1)), (2) word N-gram (W), (3) semantic category N-gram (S), (4) word N-gram and semantic category N-gram (S+W), (5) N-gram consisting of a combination of word and semantic category (S$*$W), (6) N-gram consisting of combination of word, semantic category and part-of-speech (S$*$W$*$P). The term "+" here means the sum of their features, and "$*$" means the combination of word attributes is considered.

Note that I delete the set of features of the part-of-speech N-gram, because its performance was too poor. Only the part-of-speech N-gram gives any appropriate information for question classification. First, I compare the performance of (S+W(N=1)) and (S+W), to clarify the effect of considering N-gram features. Then, I compare the performance of (W), (S), (S+W), (S$*$W) and (S$*$W$*$P), to determine out the effect of considering N-gram features of a combination of different sources of word attributes.

In these experiments, parameter t is set at 15. This is because $t = 15$ gives the best performance in most of the compared machine learning methods. Moreover, parameter C in Equation (6) is set at 1, which is the same as in the previously described experiments.

3.4 Experimental Results and Discussion

3.4.1 Effect of Combining with SVM

Table 2 shows the performance of each question type: (ave.F) represents the average f-measures and (acc.) represents accuracy of the estimated question type in each question. In this table, (SVM1) and (SVM2) represent an SVM with a polynomial kernel whose degrees are 1 and 2, respectively.

As shown in these tables, SVM performed better than other methods. Specifically, because of the much higher average f-measure, SVM can perform well in all the question types including question types that have only a small number of questions.

The rule-based method performed well for question types that are assumed to be easier to classify, because it seems to possess some typical expression for identifying the types, i.e., “who” with PERSON. However, in our experimental setting, the data set contained a very wide variety of expression, and it is very difficult to make rules to cover them all. Note that with the question type “OTHER”, a high number of correct labels but a low f-measure means the rule-based method always indicates question type “OTHER”, if there is no rule that can be applied to a given question. This fact also indicates that the rule-based method omits many necessary rules for question classification.

Table 2. A comparison of the proposed method with the conventional methods

		RULE	C4.5	MEM	SVM1	SVM2
type	*m*	F measure				
AGE	130	.784	.878	.710	**.904**	.873
DATE	1885	.832	.924	.931	**.965**	.962
EVEN	165	.545	.296	.517	**.585**	.574
LOCT	1530	.616	.575	.738	.744	**.784**
MONY	250	.734	.810	.645	.808	**.829**
NORG	140	**.746**	.654	.637	.727	.722
NPER	365	.853	.834	.836	**.863**	.858
ORGN	1605	.618	.541	.739	.734	**.751**
PCEN	190	**.817**	.765	.759	.812	.800
PERI	260	.439	.734	.627	**.774**	.745
PERS	1615	.816	.707	.873	**.894**	.888
PROD	135	.402	.185	.348	**.587**	.521
PTIT	270	.790	.751	.881	**.886**	.874
SUBS	130	.498	.387	.373	.646	**.647**
TIME	125	.718	.778	.758	.812	**.823**
TITL	150	.316	.299	.335	**.478**	.404
OTHR	1055	.434	.575	.649	.665	**.666**
ave.F		.645	.629	.668	**.758**	.748
acc.		.683	.670	.779	.807	**.813**

The decision tree method performed the worst in this set of experiments. I imagine the number of features is slightly large and this causes the classifier over-fitting in the training data.

For these reason, SVM is more suitable than the other methods.

Table 3. A comparison of the proposed method with the conventional methods based on the number of correct questions

		RULE	C4.5	MEM	SVM1	SVM2
type	m	#. of correct question type				
AGE	130	87	111	77	**118**	107
DATE	1885	1391	1700	**1839**	1837	1832
EVEN	165	81	46	61	**83**	78
LOCT	1530	893	913	1196	1184	**1259**
MONY	250	164	199	127	198	**201**
NORG	140	86	82	72	**97**	95
NPER	365	285	297	288	318	**322**
ORGN	1605	1048	870	**1293**	1177	1217
PCEN	190	133	135	123	**147**	144
PERI	260	160	181	128	**190**	177
PERS	1615	1246	1149	1496	1490	**1509**
PROD	135	39	19	31	**66**	56
PTIT	270	182	203	223	**237**	229
SUBS	130	57	41	31	**72**	66
TIME	125	86	98	83	**104**	100
TITL	150	30	43	31	**60**	44
OTHR	1055	**855**	615	692	695	694
total	10000	6828	6702	7791	8073	**8130**

Effect of Multi-class Classification Methods

This section shows experimental results obtained when comparing multi-class classification methods, one vs. rest and pairwise methods [Weston98, Weston99, KreBel98].

The strategy for deciding final estimated question type for each method is as follows:

- one vs. rest model: a question type that gives the largest value of Equation

Table 4. Classification performance versus multi-class classification method

		one vs. rest		pairwise	
		SVM1	SVM2	SVM1	SVM2
type	m	F-measure			
AGE	130	**.904**	.873	.857	.791
DATE	1885	**.965**	.962	.961	.957
EVEN	165	.585	.574	**.608**	.496
LOCT	1530	.744	**.784**	.763	.769
MONY	250	.808	**.829**	.781	.746
NORG	140	.727	.722	**.737**	.672
NPER	365	**.863**	.858	.856	.843
ORGN	1605	.734	**.751**	.720	.717
PCEN	190	**.812**	.800	.780	.743
PERI	260	**.774**	.745	.759	.688
PERS	1615	**.894**	.888	.880	.878
PROD	135	**.587**	.521	.584	.426
PTIT	270	.886	.874	**.890**	.841
SUBS	130	.646	**.647**	.590	.460
TIME	125	.812	.823	**.825**	.760
TITL	150	**.478**	.404	.464	.301
OTHR	1055	.665	**.666**	.663	.610
ave.F		**.758**	.748	.748	.688
acc.		.807	**.813**	.802	.783

(11) is chosen as the final estimated question type.

- pairwise model: a question type which is voted for most is chosen as the final estimated question type, where each classifier votes 1 for each type classified as positive.

Table 4 shows the result of a comparison of the one vs. rest and pairwise models.

Even though a previous study made it clear that the pairwise method is a better approach than the one vs. rest model [KreBel98], in our experiments, the one vs. rest model performed better than the pairwise method. I assume that when dealing with very high dimensions of feature space, the pairwise model only uses samples of the two target classes, while the one vs. rest model use all the samples to construct each classifier, thus making it difficult to learn the appropriate classifier.

Effect of Kernel Parameter

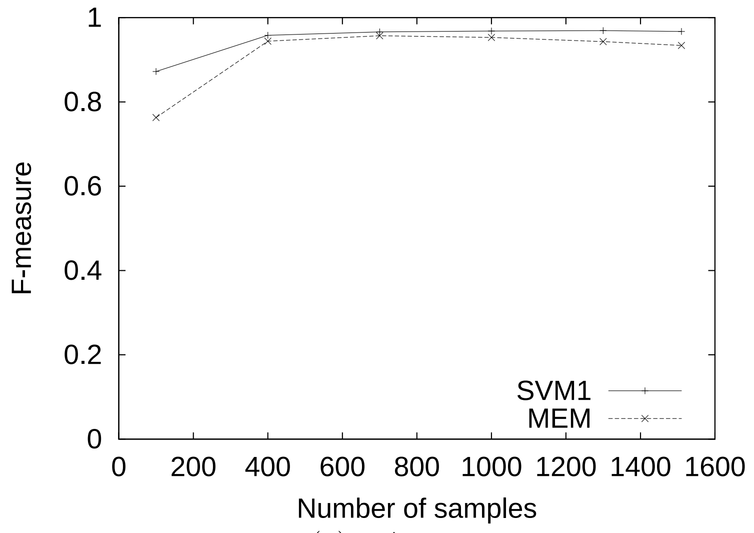
The comparison of the performance of SVM1 and SVM2 in Table 2, 4, shows that SVM2 performed much worse with some question types that have only a small number of questions, i.e., AGE, EVENT, PERCENT, PERIOD, PRODUCT, PTITLE and TITLE. This means that considering the combination of features by polynomial kernels leads to a negative effect. This is because the feature space becomes too sparse, and the positive class is underestimated. Since even SVM1 can easily divide into positive and negative in every classifier in training, we need not use such a high-dimensional feature space constructed by polynomial kernels.

Learning Curve

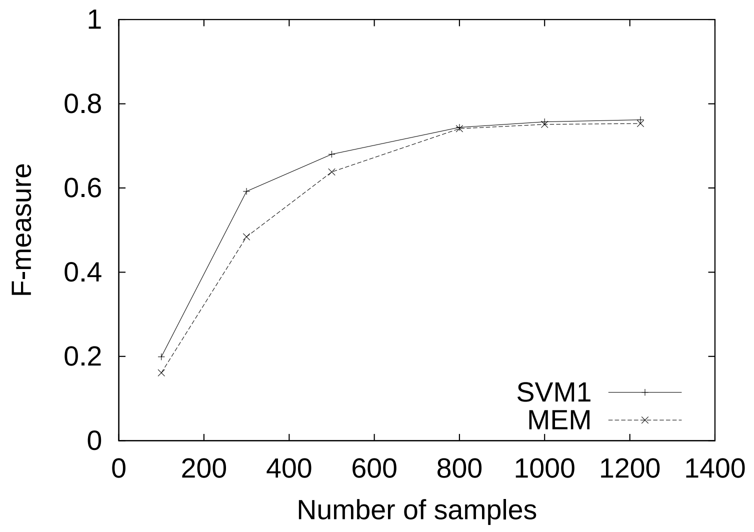
Here, I evaluate the effect of sample size an performance.

Figures 7 and 8 show the f-measure of question types, DATE, LOCATION, AGE and TITLE. These question types are selected for the following reasons; DATE: a large number of samples and high performance, LOCATION: a large number of samples but poor performance, AGE: a small number of samples but high performance, and TITLE: a small number of samples and poor performance. That is, these question types are selected based on the criteria of question size and performance.

The results revealed the difficulty posed by each question type or required number of samples. The performance of question type DATE was near the 90% with only 100 samples. We can say the DATE is an easier question type and the data set has a sufficient number of samples for question classification.



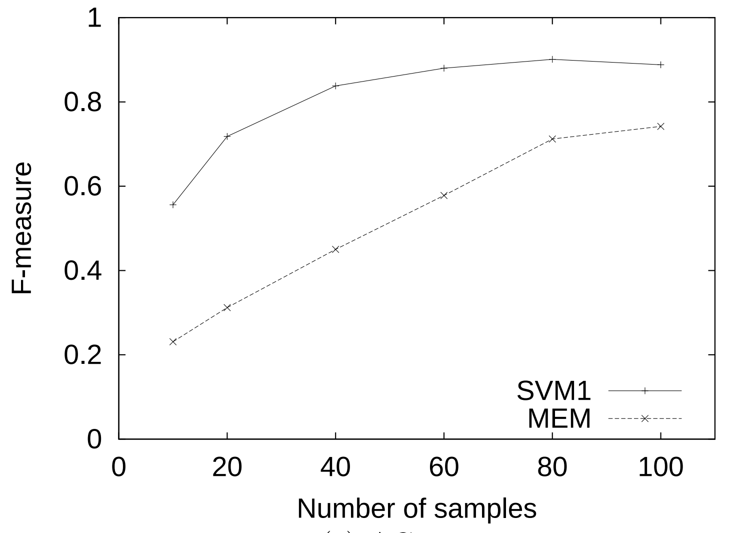
(a) DATE



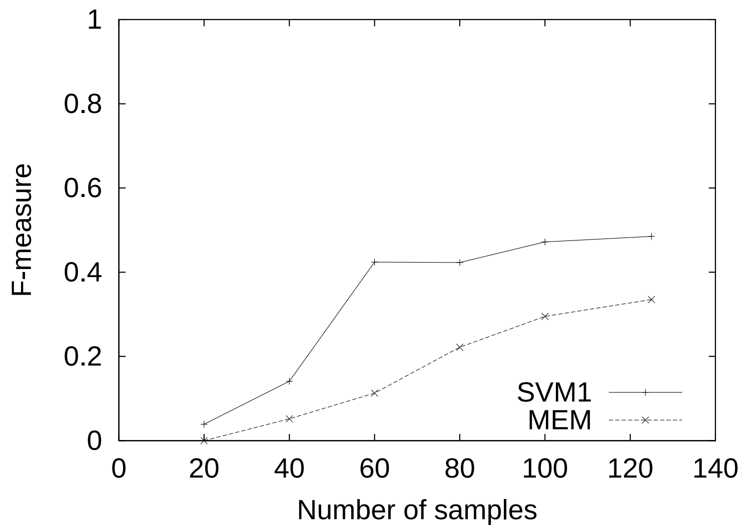
(b) LOCATION

Figure 7. Learning curves of each question type (1/2)

In contrast to DATA, even though LOCATION has about 1200 questions, this is insufficient to classify LOCATION. This means LOCATION is a difficult



(c) AGE



(d) TITLE

Figure 8. Learning curves of each question type (2/2)

question type. Figures 7 and 8 show that increasing the training data improved the performance, however, we can easily guess that it is very difficult to reach

Table 5. Effects of features

		MEM		SVM1		SVM2	
word attributes	# of features	ave.F	acc.	ave.F	acc.	ave.F	acc.
S+W(N=1)	4090	.466	.644	.645	.729	.643	.758
S	7565	.559	.665	.636	.724	.668	.744
W	10114	.592	.733	.673	.763	.651	.750
S+W	17681	.659	.767	.741	.798	.747	.810
S∗W	45525	**.684**	**.792**	.773	**.824**	**.754**	**.818**
S∗W∗P	118509	.670	.777	**.780**	**.824**	.733	.807

90% solely by increasing the number of samples.

We can say the same for question types AGE and TITLE, that is, AGE has a sufficient number of questions, and TITLE does not. Therefore, I confirmed that I could observe an improvement by adding new samples. This is especially for samples such as question type TITLE, which has only a small number of questions and improved when I increased then number of samples.

3.4.2 Effect of Features

Table 5 shows the results of testing the effects of word attribute N-gram features.

As shown in this table, according to the results of S+W(N=1) and S+W, N-gram features include efficient features for question classification.

Next, when compared with S+W and $S * W$, whose difference relates to whether the combination of different information such as word and semantic category are considered or not, N-gram with the combination provides better levels of performance. Some features are better to combine into one with upper level information such as semantic category in this case to generalize the features, however, some features are not. This situation can be handled by dealing with different levels of information and lead to better performance. Note that dealing with the different levels of information together in one feature is one of the main aspects of our method, and our approach is a more natural technique for question classification.

However, our approach to dealing with part-of-speech, that is, S$*$W$*$P, did not improve the performance much with an increase in the number of features. This result indicates the part-of-speech information is not really related to question classification. Moreover, MEM and SVM2 showed that S$*$W$*$P degraded the performance. I believe effect of dealing with part-of-speech is less than the negative effect of increasing the number of features, which sometimes leads to over-fitting in the training data. Therefore, the above discussion shows that the information of part-of-speech is not very important for question classification.

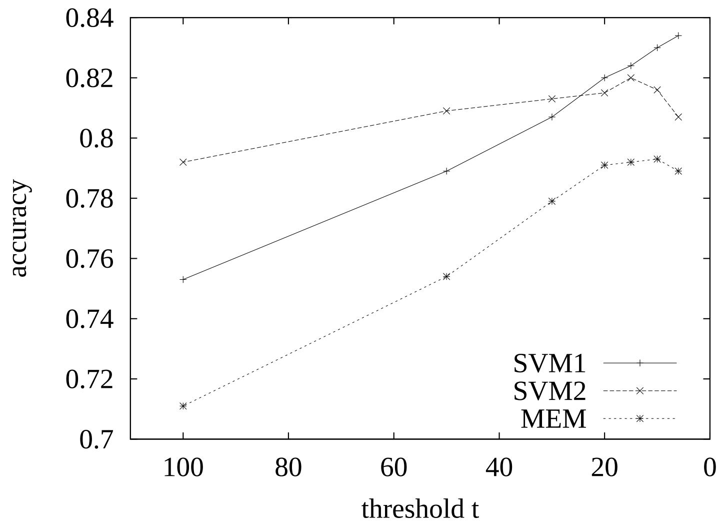
Effect of Threshold t

In this set of experiments, I evaluated the effect of threshold t in word attribute N-grams. The results are shown in Figure 9.

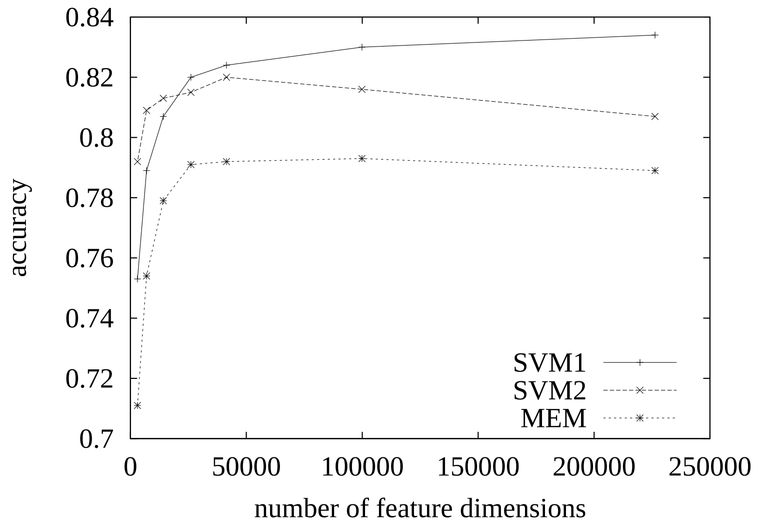
As shown in the figure, decreasing t causes the number of features to increase, and this might have an adverse effect on the generalization performance of the machine learning method. However, SVM can improve the performance. This showed that evidence of certain poor appearance features can also inform the tasks, and also SVM has good generalization performance. From this result if we could select only informative features from all possible features regardless of the number of appearances, we could expect the performance to improve.

In the case of maximum entropy method, the performance worsened with decreasing t. this might be the result of over-fitting to the training data and degraded generalization performance. This result showed one reason for using SVM with the proposed method.

However, using polynomial kernels with degree 2 also degraded the performance, while decreasing t. As discussed previously, the proposed method already provided a large number of features. The polynomial kernels with degree 2 provided too many features and that caused the generalization performance to worsen, as found with the maximum entropy method. Therefore, we have to carefully choose appropriate kernels.



(a) Shift in performance with threshold t



(b) Shift in performance with feature dimensions

Figure 9. Effects of the threshold t on the word attribute $N-$gram

Feature Analysis

The proposed method which extracts large numbers of features, utilized the inherent SVM property, namely that it is robust when using many features. The

Table 6. Examples of effective features for each question type (1/2)

AGE	DATE	EVENT	LOCATION
{ 年齢 }	いつ	{ 挙行 }-(格助詞)	どこ
年齢	{ 時間 }	{ 式・行事等 }	{ 地域（人間活動）}
{ 年齢 }-(副助詞)	{ 日程 }	(格助詞)-(*)-{ 挙行 }	どこ-で
{ 年齢 }-は	いつ-(*)-(格助詞)	イベント	会場
{ 年齢 }-(*)-(助動詞)	{ 時間 }-(*)-(格助詞)	(格助詞)-(*)-{ 行事 }	{ 地域（範囲）}-(格助詞)
{ 年齢 }-(*)-(記号)	{ 日程 }-(*)-(記号)	{ 日 }-(*)-で	{ 名詞 }-(*)-(格助詞)
(格助詞)-(*)-{ 年齢 }	{ 日程 }-(*)-(動詞)	{ 祭 }	{ 位置 }-で
何-歳	{ 日程 }-(格助詞)	{ 会議 }	どこ-(格助詞)
{ 数 }-{ 年齢 }	は-いつ	{ 行事 }	{ 地域（人間活動）}-で
何-{ 年齢 }	は-{ 時間 }	{ 公共機関 }-(*)-{ 日 }	{ 地域（人間活動）}-(格助詞)
MONEY	NORGANIZATION	NPERSON	ORGANIZATION
いくら	(格助詞)-(*)-{ 企業 }	人	どこ
{ 値・額 }	{ 企業 }	何-人	{ 企業 }-を
{ 収入 }	数	{ 数 }-人	の-(*)-{ 企業 }
{ 値・額 }-(格助詞)	{ 企業 }-(*)-{ 数 }	人数	{ 機関 }
いくら-(*)-(助動詞)	{ 公共機関 }-(*)-{ 数 }	の-{ 個数・回数等 }	どこ-(格助詞)
{ 値・額 }-(*)-(記号)	{ 公共機関 }-(格助詞)-{ 数 }	{ 個数・回数等 }	{ 位置 }
{ 負債 }	数-(*)-(記号)	{ 個数・回数等 }-(*)-(記号)	(記号)-(名詞)-(格助詞)
額	幾-つ	数	(名詞)-(*)-(副用語)
総額	幾-{ 単位 }	(格助詞)-{ 個数・回数等 }	{ 事務所 }
(格助詞)-(*)-{ 収入 }	幾-{ 個数・回数等 }	居る	どこ-が

experimental results described so far show that the proposed method is highly suited to text processing tasks. However, we still do not know which primitive features are really informative for the given tasks.

Therefore, this set of experiments clarifies which primitive feature are informative by using one of the features selection methods for SVM, called SVM Recursive Feature Elimination (SVM-RFE)[Guyon02].

SVM-RFE calculates the contribution of each primitive feature by using the value of $\mathbf{w}$ in Equation (11).

Tables 6 and 7 show the top 10 features for each question type derived by SVM-RFE, where { } represents the semantic category from the dictionary, ( ) represents the part-of-speech tags, $*$ represents arbitrary word attributes, and the others are words themselves.

Note that the classification performance shown in the previous experiments is obtained when all the features are dealt with in the training, and only the top 10

Table 7. Examples of effective features for each question type (2/2)

PERCENT	PERIOD	PERSON
{ 計算値 }	期間	誰
(格助詞)-(*)-{ 計算値 }	(格助詞)-(*)-期間	{ 不定称 }
{ 計算値 }-(*)-(記号)	{ 時 }	誰-(格助詞)
の-(*)-{ 計算値 }	期間-(格助詞)	{ 不定称 }-(格助詞)
{ 計算値 }-(副助詞)	の-(*)-期間	{ 偉人 }
割合	どれぐらいの	{ 男 }-{ 競技場 }
{ 計算値 }-(*)-(助動詞)	(副助詞)-(*)-{ 期間（自然・人間活動等）}	(記号)-(*)-(格助詞)-(*)-(記号)
{ 計算値 }-は	期間-(*)-(記号)	{ 人名 }
何-{ 計算値 }	期間-(*)-(助動詞)	{ 人名 }-(*)-(記号)
{ 数 }-{ 計算値 }	{ 時点 }	は-誰

PRODUCT	PTITLE	SUBSTANCE
{ 菓子 }	{ 敬称 }	{ 物質（本体）}
{ 仕事場 }	氏	物質
{ 仕事場 }-(格助詞)	{ 立場 }	{ 属性 }
(格助詞)-{ 菓子 }	役職	(格助詞)-(*)-{ 物質（本体）}
{ 売り }	{ 職業 }	(格助詞)-(*)-物質
{ 乗り物（本体（移動（陸圏)））}-は	(名詞)-氏	{ 物質（本体）}-(*)-(記号)
{ 乗り物（本体（移動（陸圏)））}-(副助詞)	{ 敬称 }-は	は-(*)-{ 物質（本体）}
(名詞)-(格助詞)	{ 敬称 }-(副助詞)	(副助詞)-(*)-{ 物質（本体）}
{ 電気部品 }	(名詞)-{ 敬称 }	(動詞)-(*)-{ 物質（本体）}
{ 企業 }-(格助詞)	どういう	(副用語)-(*)-{ 物質（本体）}

TIME	TITLE
{ 時刻 }	{ 創作物 }
{ 時刻 }-(*)-(助動詞)	{ 出版物 }
{ 時刻 }-(*)-(記号)	作品
(格助詞)-(*)-{ 時刻 }	(格助詞)-(名詞)
何-時	{ 題名 }
{ 時刻 }-を	(動詞)-(*)-{ 出版物 }
{ 時刻 }-(格助詞)	{ 創作物 }-(*)-(格助詞)
の-(*)-{ 時刻 }	{ 創作物 }-(格助詞)
{ 時刻 }-(*)-(動詞)	タイトル
{ 時刻 }-(格助詞)-(*)-(助動詞)	{ 出版物 }-(格助詞)

features cannot provide such high performance. However, this ranking is a good indicator of how informative the features are for the question types.

From an overall viewpoint, there are many semantic categories in the top ranking. Then, from a local viewpoint, for example, the question type AGE has semantic category { **年齢** }, and MONEY has { **値・額** }, PERCENT has { **計算値** }, and TIME has { **時刻** }. These semantic categories are all related to question types that match our intuition. In addition to these examples, there were many semantic categories that were semantically related to each question type. This fact indicates that semantic categories provide a powerful clue as regards classifying text into semantically related classes.

Another observation, **「いつ」** for DATE **「どこ」** for LOCATION and ORGANIZATION, **「いくら」** for MONEY, **「何-人」** for NPERSON and **「何-時」** for TIME are typical expressions for each question type. Of course, these expressions become very powerful rules for each question type. However, as there are many questions that do not use these typical expressions, we cannot obtain a high performance question classifier, solely using these rules.

Then, considering the N of the word attribute N-gram, most of the top features are $N = 1, 2, 3$. I believe that the features of larger N obviously contain the features of smaller N, and it is natural that the smaller N features are selected as they are equally important. Thus, I observed a semantic category of $N = 1$ and semantic categories with particles of $N = 2$ constitute the majority of the appearance patterns.

According to the above discussion, larger N-grams did not appear in the top ten ranking. However, this does not mean that larger N-grams are useless. We can find informative larger N-grams in or around the top hundred.

Table 8 shows similar features to the rule based method [Sasaki01]. The leftmost number in the table shows the ranking. Note that this result does not show a full the comparison of rules and features. As shown in the table, SVM can automatically select the appropriate feature set that is strongly related to our intuition.

According to these data, if we could select only the informative features, then re-train the classifiers, we could improve the performance. However, this process can be difficult because there is no instruction or theoretical proof as to which ranks are really informative for given tasks.

3.5 Related Work: Sequence Kernels

Sequence kernels were proposed in [Lodhi02, Cancedda03]. As regards the kernels, the proposed method can also *kernelize*, and be almost the same as sequence

Table 8. Examples of word attribute $N-$grams typically used in each question type

type	rank	feature
AGE	90	(格助詞)-(*)-{ 年齢 }-(格助詞)
	94	{ 年齢 }-を-(*)-．
	456	は-(*)-幾-つ
DATE	12	{ 日 }-は-？-(文末)
	71	{ 時機 }-を-(*)-(記号)
	113	{ 日 }-(*)-知る-たい-．
	289	は-いつ-(*)-(助動詞)
EVNT	587	{ 催し }-[副助詞]-何-[助動詞]-[終助詞詞]
LOCT	45	ある-(*)-の-{ 名称 }
	73	どこ-で-(*)-(格助詞)
	392	(名詞)-(*)-は-どこ
ORGN	160	{ 企業 }-[格助詞]-名前-[格助詞]
MONY	98	{ 値・額 }-は-？-(文末)
NORG	113	{ 企業 }-は-{ 数 }-{ 個数・回数等 }
NPER	765	{ 数 }-人-[助動詞]-[終助詞詞]
PERI	100	{ 数 }-年-(*)-(助動詞)
	250	は-(*)-{ 年月日 }-くらい
PTIT	437	{ 仕事 }-は-何
SUBS	109	{ 物質（本体） }-(*)-何-です
TIME	58	は-(*)-(格助詞)-いつ
	588	いつ-[動詞]-[助動詞]-[準体助詞]
TITL	127	(格助詞)-(*)-の-{ 題名 }

kernels. However, the proposed method can be defined as *extended sequence kernels (ESK)* because it can handle combinations of different levels of word attributes, and distinguish between consecutive and non-consecutive sequences (See section 3.2.2). That is, the framework of each node of a sequence can have duplicate labels.

We can simply obtain to obtain the ESK by modifying Equations (22) to (25).

Therefore, ESK can be defined as:

$$K^{\text{SK}}(S^1, S^2) = \sum_{i=1}^{|S^1|} \sum_{j=1}^{|S^2|} J(S_i^1, S_j^2). \tag{38}$$

First, I rewrite Equation (23) to handle duplicate labels in one node.

$$I(s_i^1, s_j^2) = \mu \cdot sim(s_i^1, s_j^2). \tag{39}$$

where $sim(S_i^1, S_j^2)$ represents a similarity between S_i^1 and S_j^2, that is, the number of matching labels in S_i^1 and S_j^2.

I rewrite Equations (22), (24) and (25) to realize the framework that distinguishes between consecutive and non-consecutive sub-sequences as different features.

$$J(S_i^1, S_j^2) = (J'(S_i^1, S_j^2) + J(S_{i-1}^1, S_{j-1}^2)) \cdot I(s_i^1, s_j^2) + I(s_i^1, s_j^2). \tag{40}$$

The first term represents the value of non-consecutive sequences and the second term represents the value of consecutive sequences. The following two recursive equations are introduced to obtain the first term of Equation (40).

$$J'(S_i^1, S_j^2) = \begin{cases} 0 \quad \text{if} \quad j = 0, 1, \\ \lambda(J'(S_i^1, S_{j-1}^2) + J''(S_i^1, S_{j-2}^2)) \quad \text{otherwise} \end{cases} \tag{41}$$

$$J''(S_i^1, S_j^2) = \begin{cases} 0 \quad \text{if} \quad i = 0, 1, \\ \lambda(J''(S_{i-1}^1, S_j^2) + J(S_{i-2}^1, S_j^2)) \quad \text{otherwise} \end{cases} \tag{42}$$

The complexity of ESK comes down to $O(|S^1||S^2|)$, while an explicit enumeration of all the features, which I undertook in this chapter, takes the exponential order of the length of a sequence.

3.6 Summary

This chapter presented the "word attribute N-gram", which is a gapped N-gram with a combination of different levels of information, such as word, POS and semantic category derived from a dictionary.

I assumed that the proposed method is well suited to text processing tasks that require linguistic information to achieve higher performance. The experimental results clarified that the feature set, which is constructed by word attribute N-grams, is effectively suited to text processing tasks.

Moreover, when handling a large number of features, SVM is one of the best choices for a machine learning algorithms. SVM is known to be very robust for handling large numbers of features, and a combination consisting of the proposed method and SVM provided the best result.

Finally, the experiments verified most of the relative features to given tasks by using one of the feature selection methods, SVM RFE. These extracted features are also intuitively related to each question type, thus, this fact also indicated that the proposed method can automatically extract efficient features from the texts.

4. Hierarchically Structured Graph Kernels

4.1 Introduction

Recently, the design of appropriate kernel functions for specific tasks has engaged the attention of many researchers in various fields, e.g., natural language processing and bio-informatics. In particular, as most real world data is represented not as numerical vectors, but as discrete structures such as graphs including sequences and trees, research on kernel design is now focused on investigating kernels on discrete structures.

Convolution Kernels [Haussler99, Watkins99] are a pioneering general framework that enables us to build kernels over discrete structures. In this framework, input objects are decomposed into parts, and kernels are defined in terms of the sub-kernels between their parts. In the last five years, a number of kernels for discrete structures, such as sequences, trees and graphs, have been proposed [Lodhi02, Cancedda03, Collins01, Kashima02, Jaakkola00, Leslie04, Kashima03, Gärtner03]. The developments of this line of research 'have made it possible to handle discrete data in their original and natural representations.

Therefore, the motivation for this chapter is to propose kernels, called *Hierarchically Structured Graphs (HS-graphs)*, specifically suited to structured natural language data. To be more precise, I first define Hierarchically Structured Graphs, which are constructed with recursive graph-in-graph structures, to represent integrated syntactic and semantic information within texts. Then, I define kernels on this class of graphs. This approach can be expected to improve the performance of NLP tasks, because it can naturally deal with fully integrated syntactic and semantic information within text.

This chapter is organized as follows. In Section 4.2, I discuss the structures of natural language data. In Section 4.3, I define a class of graph, namely a hierarchically structured graph, which is suited to representing structured NL data. Then, in Section 4.4, I introduce the basic idea and an efficient computation method for kernels of hierarchically structured graphs. In Section 4.5, I compare the performance of conventional methods with that obtained with the proposed method by using real NLP tasks. Moreover, I clarify the advantages of the proposed method according to the experimental results. In Section 4.6, I present some extensions of kernels on hierarchically structured graphs that are more suited for handling structured NL data.

4.2 Structured Natural Language Data

In general, natural language data contain many kinds of syntactic and semantic structures. For example, texts have several levels of syntactic and semantic segments (chunks), such as part-of-speech (POS) tags, named entities (NEs), noun phrases (NPs), sentences, and discourse segments, and these segments have related structures, such as dependency structures, semantic roles, anaphora, coreferences, and discourse structures. These syntactic and semantic structures can provide important information for understanding natural language and, moreover, can tackle real tasks in NLP application areas. It is natural to expect that dealing with all this richer syntactic and semantic information in an integrated way provides much better performance than conventional methods, which lack much of their information or only deal with part of it. That is, the idea arose naturally from the desire to provide a method that can deal with all the syntactic and semantic information that exists inherently in text. The background to the approach is that fundamental natural language analysis tools and electric dictionaries, namely *natural language processing (NLP) resources*, have improved rapidly in recent years, and developed to the point that they can help us to realize real applications.

Figure 10 shows an example of structures within texts analyzed by NLP resources that are currently available and that offer high levels of performance. Note that not only the NLP resources shown in Figure 10, but also semantic roles, discourse structures and anaphora analyzers are now being developed by many researchers and will be available soon.

Then I combine all of these analysis results in a single representation. Figure 11 shows an example of integrated structures within texts. As shown in Figure 11, structures in texts take the form of hierarchical or recursive structures. This means segments that are obtained from syntactic and semantic information are characterized by smaller segments inside them or relationships between

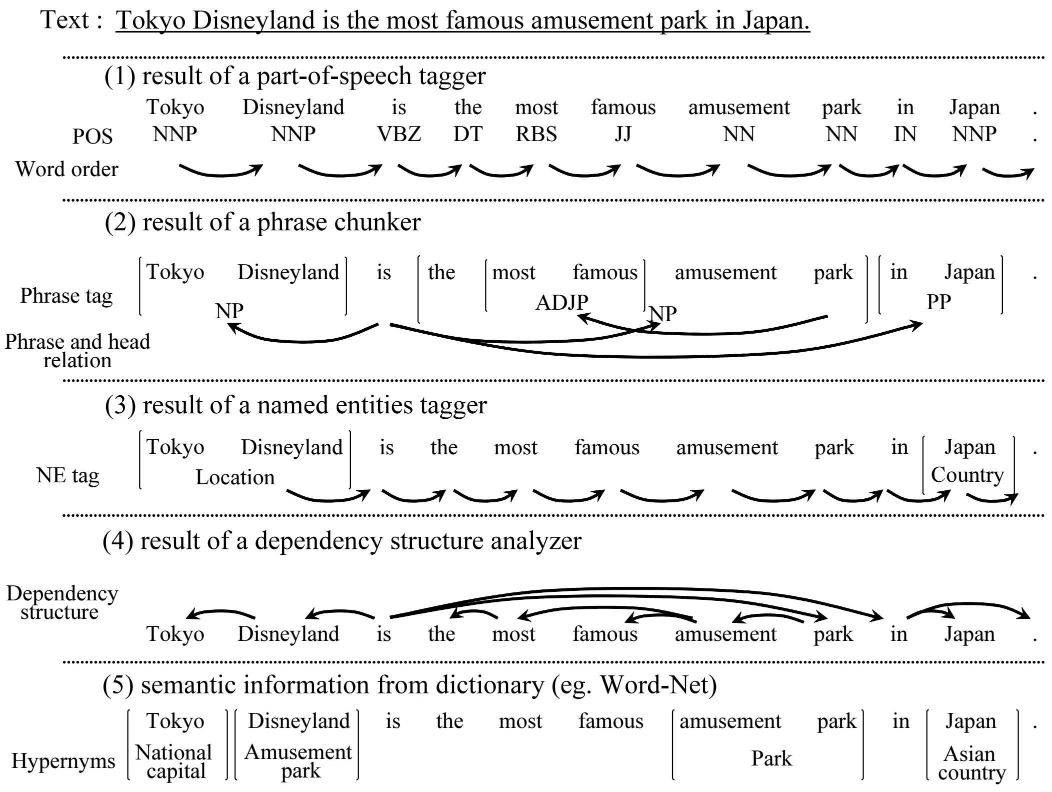

Text : Tokyo Disneyland is the most famous amusement park in Japan.
(1) result of a part-of-speech tagger
Tokyo Disneyland is the most famous amusement park in Japan .
POS NNP NNP VBZ DT RBS JJ NN NN IN NNP .
Word order
(2) result of a phrase chunker
Phrase tag
Tokyo Disneyland NP
is
the most famous ADJP amusement park NP
in Japan PP
Phrase and head relation
(3) result of a named entities tagger
NE tag
Tokyo Disneyland Location
is the most famous amusement park in
Japan Country
(4) result of a dependency structure analyzer
Dependency structure
Tokyo Disneyland is the most famous amusement park in Japan .
(5) semantic information from dictionary (eg. Word-Net)
Hypernyms
Tokyo National capital
Disneyland Amusement park
is the most famous
amusement park Park
in
Japan Asian country

Figure 10. Information obtained from natural language processing resources

them. For example, the segment labeled 'Location' is characterized by the word sequence 'Tokyo Disneyland', and the second noun phrase (NP) is composed of the word sequence 'the most famous amusement park'. Therefore, this kind of hierarchical information is well suited to model structured natural language data.

4.3 Hierarchically Structured Graph (HS-graph)

This section provides a definition of a class of graphs constructed by using hierarchical structures such as those shown in Figure 11, that is, vertices can be characterized by graphs. In other words, vertices can have certain 'special edges' directed to (sub-)graphs. We call this family of graphs hierarchically structured graphs (HS-graph).

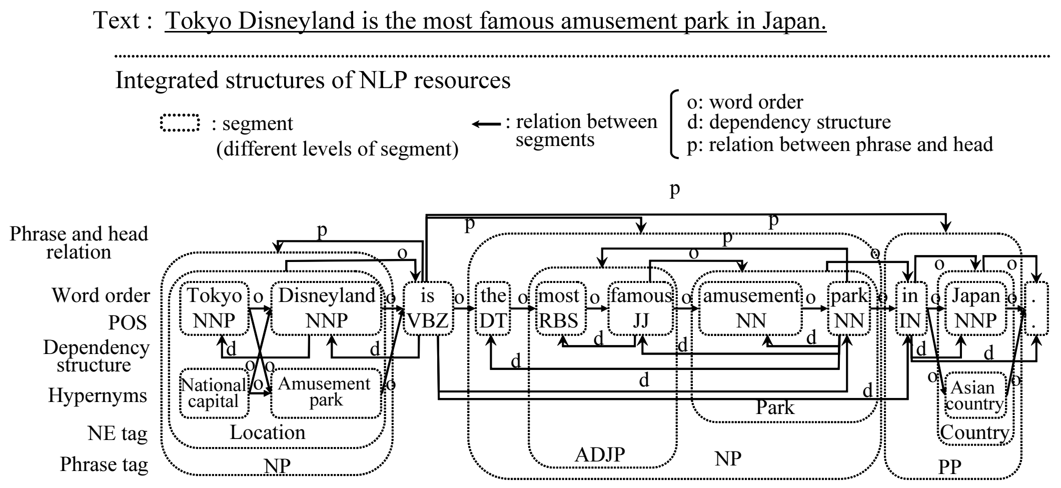


Figure 11. Examples of integrated structure obtained from natural language processing resources

Some similar classes of graphs with hierarchical graph structures have already been proposed, such as hi-graphs [Harel88], clustered graphs [Eades96], compound graphs [Sugiyama91] and hierarchical graphs [Buchsbaum00]. The definitions they use are all slightly different. Therefore, this dissertation provides an original definition of a hierarchically structured graph.

In order to define a hierarchically structured graph, first, I introduce a *vertical edge*, which represents a special edge from a vertex to a (sub-)graph. Let $\mathcal{G} = (V, E)$ be a graph, where V is the set of vertices, and $E \subseteq V \times V$ is a set of edges[3] . Then, let $\mathcal{G}_i = (V_i, E_i)$ be a subgraph in $\mathcal{G} = (V, E)$ where $V_i \subseteq V$ and $E_i \subseteq E$, and $\mathbb{G} = \{\mathcal{G}_i | i = 1 \dots n\}$ be a set of subgraphs in $\mathcal{G}$.

Definition 1 *(Vertical Edge) $F \subseteq V \times \mathbb{G}$ is a set of vertical edges. That is, a vertical edge $f_{i,j}$ is defined as a directed edge from a vertex $v_i \in V$ to a subgraph $\mathcal{G}_j \in \mathbb{G}$.*

A hierarchically structured graph is defined as follows:

[3] This dissertation only discusses the class of directed graphs, since an undirected graph can be identified with a directed graph that has two edges of both directions for each edge.

Definition 2 *(Hierarchically Structured Graph) A hierarchically structured graph can be represented by a 4-tuple, $\mathcal{G} = (V, E, \mathbb{G}, F)$.*

I use the notation $V(\mathcal{G})$, $E(\mathcal{G})$, $\mathbb{G}(\mathcal{G})$, and $F(\mathcal{G})$ to represent the sets of all vertices, edges, subgraphs, and vertical edges of graph $\mathcal{G}$, respectively.

The purpose of this chapter is to propose kernels suited to dealing with structured NL data, therefore the graph generally has labels. Let Γ be a set of labels and $\mathcal{A} : V \cup E \cup F \rightarrow \Gamma$ be a label mapping to the vertices, edges and vertical edges. Finally, a *labeled hierarchically structured graph* is defined as follows.

Definition 3 *(Labeled Hierarchically Structured Graph) A labeled hierarchically structured graph can be represented by a 5-tuple, $\mathcal{G} = (V, E, \mathbb{G}, F, \mathcal{A})$.*

Figure 12 shows examples of hierarchically structured graphs: (a) a standard graph G, (b) an HS-graph $\mathcal{G}^1$ and (c) a labeled HS-graph $\mathcal{G}^2$. Hereafter, this dissertation refers to labeled hierarchically structured graphs simply as hierarchically structured graphs.

Figure 13 shows an example of the structured NL data shown in Figure 11, rewritten as an HS-graph.

4.4 Kernels on Hierarchically Structured Graphs

This section introduces the basic ideas behind kernels on hierarchically structured graphs. Conceptually, I define kernels on HS-graphs as a set of walks[4] on an HS-graph, which we call *hierarchically defined walks*. Therefore, I first define the concept of *hierarchically defined walk*, and then introduce a definition of kernels on HS-graphs.

A *walk* $\omega(\mathcal{G})$ in graph $\mathcal{G} = (V, E)$ is generally defined as a possibly infinite sequence of edges $e_{i,j} \in E$ which is produced by traversing the vertices with connected edges. However, in order to understand the definition of kernels on HS-graphs easily, a walk $\omega(\mathcal{G})$ is written as a possibly infinite alternating sequence of vertices v and edges e, that is,

$$\omega(\mathcal{G}) = \langle v_{i_1}, e_{i_1,i_2}, v_{i_2}, e_{i_2,i_3}, v_{i_3}, \cdots, v_{j_{L-1}}, e_{i_{L-1},i_L}, v_{i_L} \rangle,$$

[4] A walk is called a *path*, if it contains each vertex no more than once.

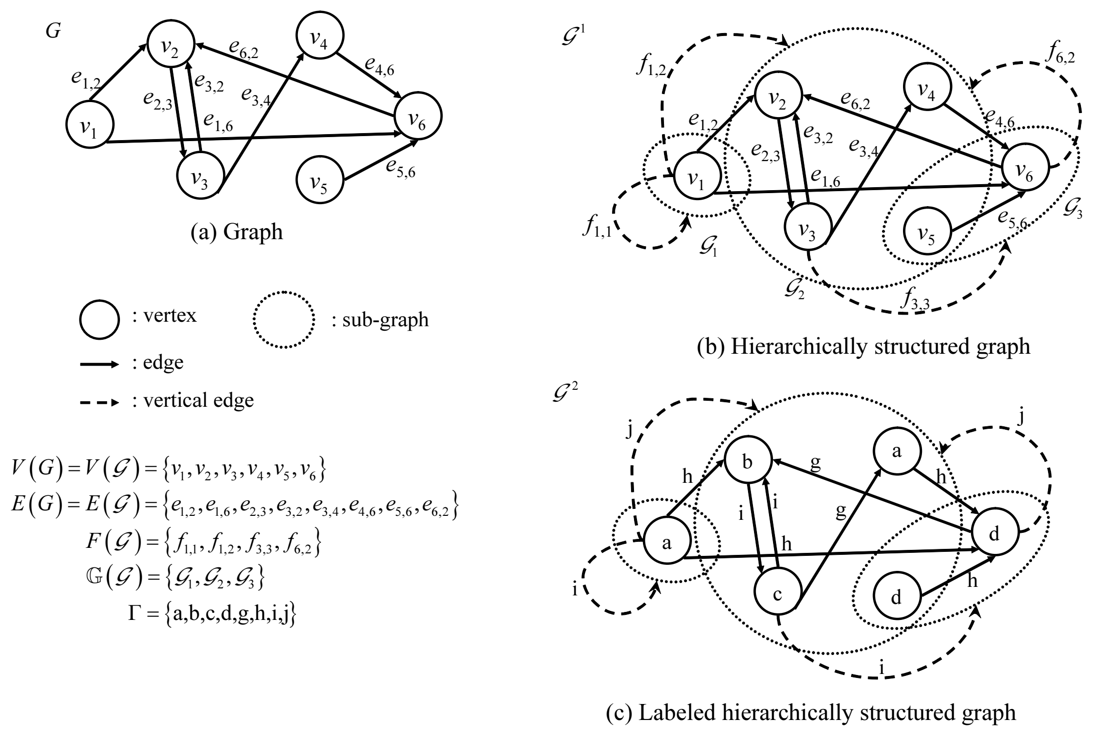

G
(a) Graph
: vertex
: sub-graph
: edge
: vertical edge
V(G) = V(𝒢) = {v1, v2, v3, v4, v5, v6}
E(G) = E(𝒢) = {e1,2, e1,6, e2,3, e3,2, e3,4, e4,6, e5,6, e6,2}
F(𝒢) = {f1,1, f1,2, f3,3, f6,2}
𝔾(𝒢) = {𝒢1, 𝒢2, 𝒢3}
Γ = {a,b,c,d,g,h,i,j}
𝒢¹
(b) Hierarchically structured graph
𝒢²
(c) Labeled hierarchically structured graph

Figure 12. Examples of hierarchically structured graphs: (a) a graph G (b) an HS-graph $\mathcal{G}^1$ (c) a labeled HS-graph $\mathcal{G}^2$

where $v_{i_k} \in V(\mathcal{G}), e_{i_k,i_{k+1}} \in E(\mathcal{G})$ and $\{i_k | k = 1, 2, \dots, L\}$. L represents the length of a walk.

I introduce a *hierarchically defined walk* on HS-graphs. Let $\mathcal{P}(\mathcal{G})$ be a set of all possible hierarchically defined walks in HS-graph $\mathcal{G}$, and $\varpi(\mathcal{G})$ be a primitive hierarchically defined walk, where $\varpi(\mathcal{G}) \in \mathcal{P}(\mathcal{G})$.

Definition 4 *(Hierarchically Defined Walk (Hi-walk)) A hi-walk in HS-graph $\mathcal{G}$ is a possibly infinite alternating sequence of $h(v)$ and edges e, that is,*

$$\varpi(\mathcal{G}) = \langle h(v_{i_1}), e_{i_1,i_2}, h(v_{i_2}), e_{i_2,i_3}, h(v_{i_3}), \cdots, h(v_{i_{L-1}}), e_{i_{L-1},i_L}, h(v_{i_L}) \rangle,$$

where $h(v_x) \in \{v_x\} \cup \{(v_x, f_{x,y}, \varpi(\mathcal{G}_y)) | v_x \in V(\mathcal{G}), f_{x,y} \in F(\mathcal{G}), \varpi(\mathcal{G}_y) \in \mathcal{P}(\mathcal{G}_y)\}$.

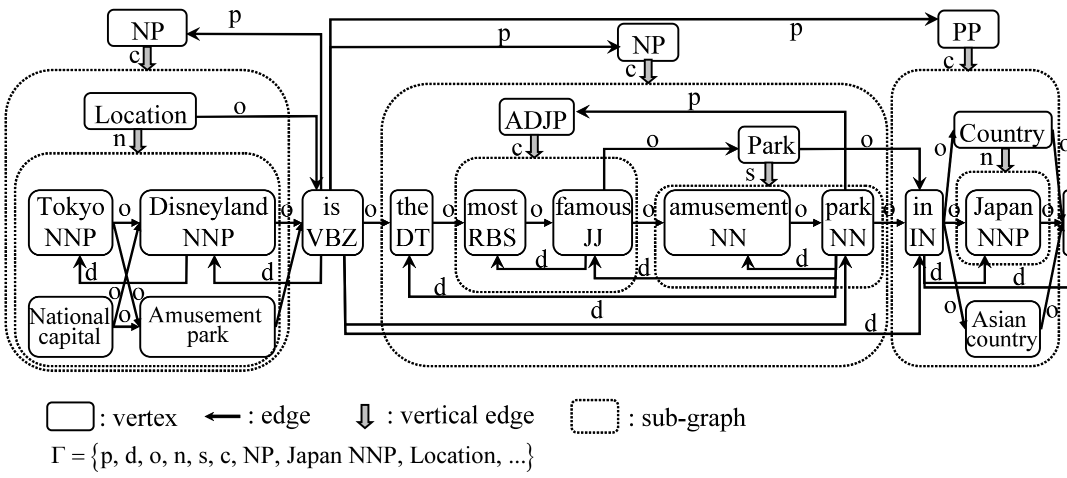


Figure 13. Example of the structured NL data shown in Figure 11 rewritten as an HS-graph

More specifically, $h(v_x)$ represents either a vertex v_x, or a 3-tuple consisting of a vertex, a vertical edge and a hi-walk, $(v_x, f_{x,y}, \varpi(\mathcal{G}_y))$.

According to this definition, a hi-walk has recursive structure, that is, ϖ is defined by h and h possibly consists of ϖ. For example, as an explicit representation, a hi-walk is written as follows:

$$\varpi(\mathcal{G}) = \langle v_2, e_{2,3}, (v_3, f_{3,3}, \langle v_5, e_{5,6}, (v_6, f_{6,2}, \langle v_4 \rangle)\rangle), e_{3,4}, v_4, e_{4,6}, v_6 \rangle.$$

Figure 14 shows graphical images of hi-walk. Intuitively, a hi-walk is allowed to traverse into a sub-graph connected with vertices by vertical edges. Thus, a hi-walk is constructed by a recursive walk in the walk structure.

Now I introduce the concept of *hierarchical label sequences*, which is a primitive feature for defining kernels on HS-graphs. Let $\tau(x)$ where $x \in V \cup E \cup F$ be a function that returns the label allocated in x.

Definition 5 *(Hierarchical label sequence (hi-label sequence)) A hi-label sequence $\tau(\varpi)$ is a sequence of labels associated with a hi-walk ϖ. Namely,*

$$\tau(\varpi(\mathcal{G})) = \langle \tau(h(v_{i_1})), \tau(e_{i_1,j_1}), \tau(h(v_{j_1})), \tau(e_{i_2,j_2}), \cdots, \tau(e_{i_{L-1},j_L}), \tau(h(v_{j_L})) \rangle,$$

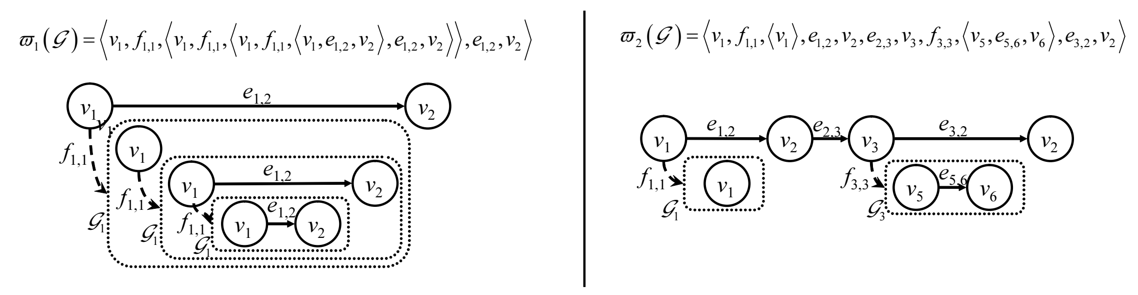

$\varpi_1(\mathcal{G}) = \langle v_1, f_{1,1}, \langle v_1, f_{1,1}, \langle v_1, f_{1,1}, \langle v_1, e_{1,2}, v_2 \rangle, e_{1,2}, v_2 \rangle\rangle, e_{1,2}, v_2 \rangle$
$\varpi_2(\mathcal{G}) = \langle v_1, f_{1,1}, \langle v_1 \rangle, e_{1,2}, v_2, e_{2,3}, v_3, f_{3,3}, \langle v_5, e_{5,6}, v_6 \rangle, e_{3,2}, v_2 \rangle$
v_1
v_2
v_3
v_5
v_6
$e_{1,2}$
$e_{2,3}$
$e_{3,2}$
$e_{5,6}$
$f_{1,1}$
$f_{3,3}$
$\mathcal{G}_1$
$\mathcal{G}_3$

Figure 14. Example graphical images of hi-walk on HS-graph $\mathcal{G}$ in Figure 12

where $\tau(h(v_i)) = \tau(v_i)$ *if* $h(v_i) = v_i$*, and* $\tau(h(v_i)) = (\tau(v_i), \tau(f_{i,j}), \tau(\varpi(\mathcal{G}_j))$ *otherwise.*

A hi-label sequence is also defined recursively: $\tau(\varpi)$ possibly consists of $\tau(\varpi)$. While each vertex, edge and vertical edge has a single label, a hi-label sequence can be derived from a hi-walk. For example, the hi-label sequence extracted from the hi-walk $\varpi_2(\mathcal{G})$ in Figure 14 is $\tau(\varpi_2(\mathcal{G})) = \langle \mathrm{a}, \mathrm{i}, \langle \mathrm{a} \rangle, \mathrm{h}, \mathrm{b}, \mathrm{i}, \mathrm{c}, \mathrm{i}, \langle \mathrm{d}, \mathrm{h}, \mathrm{d} \rangle, \mathrm{i}, \mathrm{b} \rangle$.

Given labels Γ, let Γ^n be a set of all label sequences of maximum length n, that is, $\Gamma^n = \cup_{l=1}^{n} \{\langle a_1, ..., a_l \rangle |\ a_1, ..., a_l \in \Gamma\}$. Γ^* is a case where $n = \infty$. A set of all hi-walks $\langle \Gamma^* \rangle^m$ of maximum depth m is defined as $\langle \Gamma^* \rangle^m = \cup_{d=2}^{m} \cup_{l=1}^{\infty}$ $\{\langle a_1, ..., a_l \rangle | a_1, ..., a_l \in \Gamma \cup \langle \Gamma^* \rangle^{d-1}\}$, where $\langle \Gamma^* \rangle^1 = \Gamma^*$. $\langle \Gamma^* \rangle^*$ is the case where $m = \infty$.

Returning to the standard kernel definition, I define an explicit representation of a numerical feature vector of an HS-graph kernel as:

$$\phi(\mathcal{G}) = (\phi_{\gamma_1}(\mathcal{G}), \ldots, \phi_{\gamma_i}(\mathcal{G}), \ldots),$$

where $\phi(\mathcal{G})$ represents the explicit mapping function from an HS-graph to the feature space, $\phi_\gamma(\mathcal{G})$ represents the value of feature γ, and $\gamma_1, \ldots, \gamma_i \in \langle \Gamma^* \rangle^*$. Then, kernels on HS-graphs can also be written as:

$$K^{HSG}(\mathcal{G}^1, \mathcal{G}^2) = \sum_{\gamma \in \langle \Gamma^* \rangle^*} \phi_\gamma(\mathcal{G}^1)\phi_\gamma(\mathcal{G}^2). \tag{43}$$

According to Equation (43), two input HS-graphs, $\mathcal{G}^1$ and $\mathcal{G}^2$, are mapped by ϕ into feature space. Then the inner product of the weighted common hi-label sequences in $\mathcal{G}^1$ and $\mathcal{G}^2$ is calculated.

More specifically, $\phi_\gamma(\mathcal{G})$ indicates the appearance of γ in $\mathcal{G}$,

$$\phi_\gamma(\mathcal{G}) = \sum_{\varpi \in \mathcal{P}(\mathcal{G})} \delta(\gamma, \tau(\varpi)),$$

where $\tau(\varpi)$ represents a hi-label sequence obtained from hi-walk ϖ. $\delta(x, y)$ is an indicator function that returns 1 if $x = y$, and 0 otherwise.

Therefore, I can rewrite Equation (43) as follows:

$$K^{HSG}(\mathcal{G}^1, \mathcal{G}^2) = \sum_{\varpi^1 \in \mathcal{P}(\mathcal{G}^1)} \sum_{\varpi^2 \in \mathcal{P}(\mathcal{G}^2)} \delta(\tau(\varpi^1), \tau(\varpi^2)), \tag{44}$$

because only hi-label sequences that appeared in $\mathcal{P}(\mathcal{G}^1)$ or $\mathcal{P}(\mathcal{G}^2)$ are evaluated as the kernel value. This equation indicates that the HS-graph kernel calculates the inner product of all the pairs of hi-label sequences associated with hi-walks extracted from each HS-graph.

4.4.1 Efficient Computation

The straightforward calculation of ϕ_γ in Equation (43) is obviously impossible. Moreover, Equation (44) cannot be calculated, because there is an infinite number of possible hi-walks in an HS-graph, i.e. cyclic graph. This means that the possible number of common hi-walks and hi-label sequences that appear between HS-graphs is also infinite.

In order to compute kernels on HS-graphs practically, I rearrange the terms; that is I calculate matching values of all hi-label sequences with respect to the convolution of every pair of vertices in the HS-graphs. Moreover, I apply a convergence condition to the HS-graph kernels for practical calculation, namely the kernel values of the HS-graph kernels converge with respect to the lengths of the hi-walks. To obtain an efficient algorithm, I first introduce (sub-)kernels for labels, vertices, edges and vertical edges.

sub-kernels for labels

$$K_\Gamma(a_i, a_j) = \delta(a_i, a_j), \tag{45}$$

where $\delta(a_1, a_2) = 1$ if $a_1 = a_2$, and 0 otherwise.

sub-kernels for vertices

$$K_V(v_i, v_j) = K_\Gamma(\tau(v_i), \tau(v_j)). \tag{46}$$

sub-kernels for edges

$$K_E(v_i^1, v_j^2, v_k^1, v_l^2) = \begin{cases} 0, \quad \text{if } e_{i,k} \notin E(\mathcal{G}^1) \text{ or } e_{j,l} \notin E(\mathcal{G}^2), \\ K_\Gamma(\tau(e_{i,k}), \tau(e_{j,l})) \quad \text{otherwise} \end{cases} \tag{47}$$

sub-kernels for vertical edges

$$K_F(v_i^1, v_j^2, \mathcal{G}_k^1, \mathcal{G}_l^2) = \begin{cases} 0, \quad \text{if } f_{i,k} \notin F(\mathcal{G}^1) \text{ or } f_{j,l} \notin F(\mathcal{G}^2), \\ K_\Gamma(\tau(f_{i,k}), \tau(f_{j,l})) \quad \text{otherwise} \end{cases} \tag{48}$$

Using these sub-kernels, sub-kernels for $h(v)$ in Definition 4, which corresponds to vertices with subgraphs connected with vertical edges, can be written as follows.

sub-kernels for vertices with subgraphs connected by vertical edges

$$\begin{aligned} H(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) = {} & K_V(v_i^1, v_j^2) + K_V(v_i^1, v_j^2) \\ & \sum_{k=1}^{|\mathbb{G}(\mathcal{G}^1)|} \sum_{l=1}^{|\mathbb{G}(\mathcal{G}^2)|} K_F(v_i^1, v_j^2, \mathcal{G}_k^1, \mathcal{G}_l^2) K^{HSG}(\mathcal{G}_k^1, \mathcal{G}_l^2), \end{aligned} \tag{49}$$

Intuitively, $H(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2)$ represents the sum of all the hi-label sequences between $\mathcal{G}_k^1$ and $\mathcal{G}_l^2$ connected to v_i^1 and v_j^2 with vertical edges $f_{i,k}$ and $f_{j,l}$, respectively.

Suppose we can obtain $H(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2)$ even though it has a term $K^{HSG}(\mathcal{G}_k^1, \mathcal{G}_l^2)$, that means a recursive definition of $K^{HSG}(\mathcal{G}_k^1, \mathcal{G}_l^2)$. Then, let us introduce a function $K_L^{\varpi}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2)$ that returns the sum value between hi-label sequences in $\mathcal{G}^1$ and $\mathcal{G}^2$ that started with labels obtained from vertices v_i^1, v_j^2. L is introduced to represent the number of steps traversing the graph with respect to edges. Namely, the number of $h(v)$ in the hi-walk, that is, $L = 4$ if a hi-walk in graph $\mathcal{G}$ is $\varpi(\mathcal{G}) = \langle h(v_{i_1}), e_{i_1,i_2}, h(v_{i_2}), e_{i_2,i_3}, h(v_{i_3}), e_{i_3,i_4}, h(v_{i_4})\rangle$.

$$\begin{aligned}
K_1^{\varpi}(v_{i_1}^1, v_{j_1}^2, \mathcal{G}^1, \mathcal{G}^2) &= H(v_{i_1}^1, v_{j_1}^2, \mathcal{G}^1, \mathcal{G}^2) \\
K_2^{\varpi}(v_{i_1}^1, v_{j_1}^2, \mathcal{G}^1, \mathcal{G}^2) &= H(v_{i_1}^1, v_{j_1}^2, \mathcal{G}^1, \mathcal{G}^2) \\
&\quad \cdot \left(\sum_{i_2=1}^{|V(\mathcal{G}^1)|} \sum_{j_2=1}^{|V(\mathcal{G}^2)|} K_E(v_{i_1}^1, v_{j_1}^2, v_{i_2}^1, v_{j_2}^2) H(v_{i_2}^1, v_{j_2}^2, \mathcal{G}^1, \mathcal{G}^2) \right) \\
K_3^{\varpi}(v_{i_1}^1, v_{j_1}^2, \mathcal{G}^1, \mathcal{G}^2) &= H(v_{i_1}^1, v_{j_1}^2, \mathcal{G}^1, \mathcal{G}^2) \\
&\quad \cdot \left(\sum_{i_2=1}^{|V(\mathcal{G}^1)|} \sum_{j_2=1}^{|V(\mathcal{G}^2)|} K_E(v_{i_1}^1, v_{j_1}^2, v_{i_2}^1, v_{j_2}^2) H(v_{i_2}^1, v_{j_2}^2, \mathcal{G}^1, \mathcal{G}^2) \right. \\
&\quad \left. \cdot \left(\sum_{i_3=1}^{|V(\mathcal{G}^1)|} \sum_{j_3=1}^{|V(\mathcal{G}^2)|} K_E(v_{i_2}^1, v_{j_2}^2, v_{i_3}^1, v_{j_3}^2) H(v_{i_3}^1, v_{j_3}^2, \mathcal{G}^1, \mathcal{G}^2) \right) \right)
\end{aligned}$$

therefore for $L \geq 2$,

$$\begin{aligned}
&K_L^{\varpi}(v_{i_1}^1, v_{j_1}^2, \mathcal{G}^1, \mathcal{G}^2) \\
&= H(v_{i_1}^1, v_{j_1}^2, \mathcal{G}^1, \mathcal{G}^2) \\
&\quad \cdot \left(\sum_{i_2=1}^{|V(\mathcal{G}^1)|} \sum_{j_2=1}^{|V(\mathcal{G}^2)|} K_E(v_{i_1}^1, v_{j_1}^2, v_{i_2}^1, v_{j_2}^2) H(v_{i_2}^1, v_{j_2}^2, \mathcal{G}^1, \mathcal{G}^2) \right. \\
&\quad \left. \cdot \left(\cdots \left(\sum_{i_L=1}^{|V(\mathcal{G}^1)|} \sum_{j_L=1}^{|V(\mathcal{G}^2)|} K_E(v_{i_{L-1}}^1, v_{j_{L-1}}^2, v_{i_L}^1, v_{j_L}^2) H(v_{i_L}^1, v_{j_L}^2, \mathcal{G}^1, \mathcal{G}^2) \right) \right) \cdots \right). \quad (50)
\end{aligned}$$

I can further simplify Equation (50) as follows:

$$\begin{aligned}
&K_L^{\varpi}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) \\
&= H(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) \cdot \left(\sum_{k=1}^{|V(\mathcal{G}^1)|} \sum_{l=1}^{|V(\mathcal{G}^2)|} K_E(v_i^1, v_j^2, v_k^1, v_l^2) K_{L-1}^{\varpi}(v_k^1, v_l^2, \mathcal{G}^1, \mathcal{G}^2) \right), \quad (51)
\end{aligned}$$

where

$$\begin{aligned}
&K^{\varpi}_{L-1}(v^1_{i_2}, v^2_{j_2}, \mathcal{G}^1, \mathcal{G}^2)\\
&= H(v^1_{i_2}, v^2_{j_2}, \mathcal{G}^1, \mathcal{G}^2)\\
&\cdot\left(\cdots\left(\sum_{i_L=1}^{|V(\mathcal{G}^1)|}\sum_{j_L=1}^{|V(\mathcal{G}^2)|} K_E(v^1_{i_{L-1}}, v^2_{j_{L-1}}, v^1_{i_L}, v^2_{j_L}) H(v^1_{i_L}, v^2_{j_L}, \mathcal{G}^1, \mathcal{G}^2)\right)\cdots\right).
\end{aligned}$$

Now, I can define an efficient algorithm between HS-graphs $\mathcal{G}^1$ and $\mathcal{G}^2$, which is the sum of all the common hi-label sequences whose lengths L are 1 to infinity when derived from each pair of vertices:

$$K^{\text{HSG}}(\mathcal{G}^1, \mathcal{G}^2) = \sum_{i=1}^{|V(\mathcal{G}^1)|}\sum_{j=1}^{|V(\mathcal{G}^2)|} \lim_{L\to\infty}\sum_{L=1}^{L} K^{\varpi}_{L}(v^1_i, v^2_j, \mathcal{G}^1, \mathcal{G}^2). \tag{52}$$

However, H in K^{ϖ}_L cannot be calculated individually because it is defined as the recursive definition of K^{HSG}. That is, H contains K^{HSG} and K^{HSG} contains H. Therefore, I introduce $H_{\mathcal{T}}$ instead of H, which allows us to calculate K^{HSG} recursively. $\mathcal{T}$ represents the "time", in other words the maximum number of steps of walks, that is, $L \leq \mathcal{T}$ always holds.

Now, I redefine HS-graph kernels by using $\mathcal{T}$:

$$K^{\text{HSG}}(\mathcal{G}^1, \mathcal{G}^2) = \sum_{i=1}^{|V(\mathcal{G}^1)|}\sum_{j=1}^{|V(\mathcal{G}^2)|} \lim_{\mathcal{T}\to\infty}\sum_{L=1}^{\mathcal{T}} K^{\varpi}_{L,\mathcal{T}}(v^1_i, v^2_j, \mathcal{G}^1, \mathcal{G}^2). \tag{53}$$

I modify the recursive definition of K^{ϖ}_L in Equation (51) with $\mathcal{T}$, that is,

$$\begin{aligned}
&K^{\varpi}_{L,\mathcal{T}}(v^1_i, v^2_j, \mathcal{G}^1, \mathcal{G}^2)\\
&= H_{\mathcal{T}}(v^1_i, v^2_j, \mathcal{G}^1, \mathcal{G}^2)\cdot\left(\sum_{k=1}^{|V(\mathcal{G}^1)|}\sum_{l=1}^{|V(\mathcal{G}^2)|} K_E(v^1_i, v^2_j, v^1_k, v^2_l) K^{\varpi}_{L-1,\mathcal{T}-1}(v^1_k, v^2_l, \mathcal{G}^1, \mathcal{G}^2)\right).
\end{aligned}\tag{54}$$

In the same way, I also modify H in Equation (49) with $\mathcal{T}$:

$$H_{\mathcal{T}}(v^1_i, v^2_j, \mathcal{G}^1, \mathcal{G}^2) = K_V(v^1_i, v^2_j) + K_V(v^1_i, v^2_j)\sum_{k=1}^{|\mathbb{G}(\mathcal{G}^1)|}\sum_{l=1}^{|\mathbb{G}(\mathcal{G}^2)|} K_F(v^1_i, v^2_j, \mathcal{G}^1_k, \mathcal{G}^2_l)$$

$$\cdot\left(\sum_{m=1}^{|V(\mathcal{G}_k^1)|}\sum_{n=1}^{|V(\mathcal{G}_l^2)|}\sum_{L=1}^{\mathcal{T}-1}K_{L,\mathcal{T}-1}^{\varpi}(v_m^1,v_n^2,\mathcal{G}_k^1,\mathcal{G}_l^2)\right). \tag{55}$$

The function $H_{\mathcal{T}}$ derived from H, where $H=\lim_{\mathcal{T}\to\infty}H_{\mathcal{T}}$;

$$\begin{aligned}
&H(v_i^1,v_j^2,\mathcal{G}^1,\mathcal{G}^2)\\
&=K_V(v_i^1,v_j^2)+K_V(v_i^1,v_j^2)\sum_{k=1}^{|\mathbb{G}(\mathcal{G}^1)|}\sum_{l=1}^{|\mathbb{G}(\mathcal{G}^2)|}K_F(v_i^1,v_j^2,\mathcal{G}_k^1,\mathcal{G}_l^2)K^{HSG}(\mathcal{G}_k^1,\mathcal{G}_l^2)\\
&=K_V(v_i^1,v_j^2)+K_V(v_i^1,v_j^2)\sum_{k=1}^{|\mathbb{G}(\mathcal{G}^1)|}\sum_{l=1}^{|\mathbb{G}(\mathcal{G}^2)|}K_F(v_i^1,v_j^2,\mathcal{G}_k^1,\mathcal{G}_l^2)\\
&\qquad\qquad\cdot\left(\sum_{m=1}^{|V(\mathcal{G}_k^1)|}\sum_{n=1}^{|V(\mathcal{G}_l^2)|}\lim_{\mathcal{T}\to\infty}J_{\mathcal{T}}(v_m^1,v_n^2,\mathcal{G}_k^1,\mathcal{G}_l^2)\right)\\
&=\lim_{\mathcal{T}\to\infty}H_{\mathcal{T}}(v_i^1,v_j^2,\mathcal{G}^1,\mathcal{G}^2)
\end{aligned} \tag{56}$$

I rewrite Equation (53) as

$$K^{\text{HSG}}(\mathcal{G}^1,\mathcal{G}^2)=\sum_{i=1}^{|V(\mathcal{G}^1)|}\sum_{j=1}^{|V(\mathcal{G}^2)|}\lim_{\mathcal{T}\to\infty}J_{\mathcal{T}}(v_i^1,v_j^2,\mathcal{G}^1,\mathcal{G}^2), \tag{57}$$

where

$$J_{\mathcal{T}}(v_i^1,v_j^2,\mathcal{G}^1,\mathcal{G}^2)=\sum_{L=1}^{\mathcal{T}}K_{L,\mathcal{T}}^{\varpi}(v_i^1,v_j^2,\mathcal{G}^1,\mathcal{G}^2). \tag{58}$$

Then, I need to compute $\lim_{\mathcal{T}\to\infty}J_{\mathcal{T}}$ to obtain K^{HSG}.

By using the recursive relationship for $K_{L,\mathcal{T}}^{\varpi}$, $J_{\mathcal{T}}$ can be obtained by the following equation:

$$\begin{aligned}
&J_{\mathcal{T}}(v_i^1,v_j^2,\mathcal{G}^1,\mathcal{G}^2)\\
&=\sum_{L=1}^{\mathcal{T}}K_{L,\mathcal{T}}^{\varpi}(v_i^1,v_j^2,\mathcal{G}^1,\mathcal{G}^2)\\
&=K_{1,\mathcal{T}}^{\varpi}(v_i^1,v_j^2,\mathcal{G}^1,\mathcal{G}^2)+\sum_{L=2}^{\mathcal{T}}K_{L,\mathcal{T}}^{\varpi}(v_i^1,v_j^2,\mathcal{G}^1,\mathcal{G}^2)
\end{aligned}$$

$$
\begin{aligned}
&= H_{\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) + \sum_{L=2}^{\mathcal{T}} H_{\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) \\
&\qquad \cdot \left(\sum_{k=1}^{|V(\mathcal{G}^1)|} \sum_{l=1}^{|V(\mathcal{G}^2)|} K_E(v_i^1, v_j^2, v_k^1, v_l^2) K^{\varpi}_{L-1,\mathcal{T}-1}(v_k^1, v_l^2, \mathcal{G}^1, \mathcal{G}^2) \right) \\
&= H_{\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) + H_{\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) \\
&\qquad \cdot \left(\sum_{k=1}^{|V(\mathcal{G}^1)|} \sum_{l=1}^{|V(\mathcal{G}^2)|} K_E(v_i^1, v_j^2, v_k^1, v_l^2) \sum_{L=2}^{\mathcal{T}} K^{\varpi}_{L-1,\mathcal{T}-1}(v_k^1, v_l^2, \mathcal{G}^1, \mathcal{G}^2) \right) \\
&= H_{\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) + H_{\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) \\
&\qquad \cdot \left(\sum_{k=1}^{|V(\mathcal{G}^1)|} \sum_{l=1}^{|V(\mathcal{G}^2)|} K_E(v_i^1, v_j^2, v_k^1, v_l^2) J_{\mathcal{T}-1}(v_k^1, v_l^2, \mathcal{G}^1, \mathcal{G}^2) \right), \qquad (59)
\end{aligned}
$$

where $\sum_{L=2}^{\mathcal{T}} K^{\varpi}_{L-1,\mathcal{T}-1} = \sum_{L=1}^{\mathcal{T}-1} K^{\varpi}_{L,\mathcal{T}-1}$, and $\sum_{L=1}^{\mathcal{T}-1} K^{\varpi}_{L,\mathcal{T}-1} = J_{\mathcal{T}-1}$. I define the boundary condition, $J_0(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) = 0$ for all $v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2$, thus $J_1 = H_1$.

Replacing $H_{\mathcal{T}}$ in Equation (59) with Equation (55), The following recursive relationship holds between $J_{\mathcal{T}}$ and $J_{\mathcal{T}-1}$.

$$
\begin{aligned}
&J_{\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) \\
&= K_V(v_i^1, v_j^2) + K_V(v_i^1, v_j^2) \sum_{k=1}^{|\mathbb{G}(\mathcal{G}^1)|} \sum_{l=1}^{|\mathbb{G}(\mathcal{G}^2)|} K_F(v_i^1, v_j^2, \mathcal{G}_k^1, \mathcal{G}_l^2) \\
&\qquad\qquad \sum_{m=1}^{|V(\mathcal{G}_k^1)|} \sum_{n=1}^{|V(\mathcal{G}_l^2)|} J_{\mathcal{T}-1}(v_m^1, v_n^2, \mathcal{G}_k^1, \mathcal{G}_l^2) \\
&\quad + K_V(v_i^1, v_j^2) + K_V(v_i^1, v_j^2) \sum_{k=1}^{|\mathbb{G}(\mathcal{G}^1)|} \sum_{l=1}^{|\mathbb{G}(\mathcal{G}^2)|} K_F(v_i^1, v_j^2, \mathcal{G}_k^1, \mathcal{G}_l^2) \\
&\qquad\qquad \sum_{m=1}^{|V(\mathcal{G}_k^1)|} \sum_{n=1}^{|V(\mathcal{G}_l^2)|} J_{\mathcal{T}-1}(v_m^1, v_n^2, \mathcal{G}_k^1, \mathcal{G}_l^2) \\
&\quad + \Bigg(K_V(v_i^1, v_j^2) + K_V(v_i^1, v_j^2) \sum_{k=1}^{|\mathbb{G}(\mathcal{G}^1)|} \sum_{l=1}^{|\mathbb{G}(\mathcal{G}^2)|} K_F(v_i^1, v_j^2, \mathcal{G}_k^1, \mathcal{G}_l^2)
\end{aligned}
$$

$$\left.\sum_{m=1}^{|V(\mathcal{G}_k^1)|}\sum_{n=1}^{|V(\mathcal{G}_l^2)|} J_{\mathcal{T}-1}(v_m^1, v_n^2, \mathcal{G}_k^1, \mathcal{G}_l^2)\right)$$
$$\cdot\left(\sum_{k=1}^{|V(\mathcal{G}^1)|}\sum_{l=1}^{|V(\mathcal{G}^2)|} K_E(v_i^1, v_j^2, v_k^1, v_l^2) J_{\mathcal{T}-1}(v_k^1, v_l^2, \mathcal{G}^1, \mathcal{G}^2)\right) \tag{60}$$

Assuming that $J_\mathcal{T}$ converges when $\mathcal{T} \to \infty$, we have the following equilibrium equation:

$$\begin{aligned}
&J_\infty(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2)\\
&= K_V(v_i^1, v_j^2) + K_V(v_i^1, v_j^2) \sum_{k=1}^{|\mathbb{G}(\mathcal{G}^1)|}\sum_{l=1}^{|\mathbb{G}(\mathcal{G}^2)|} K_F(v_i^1, v_j^2, \mathcal{G}_k^1, \mathcal{G}_l^2)\\
&\qquad \sum_{m=1}^{|V(\mathcal{G}_k^1)|}\sum_{n=1}^{|V(\mathcal{G}_l^2)|} J_\infty(v_m^1, v_n^2, \mathcal{G}_k^1, \mathcal{G}_l^2)\\
&+K_V(v_i^1, v_j^2) + K_V(v_i^1, v_j^2) \sum_{k=1}^{|\mathbb{G}(\mathcal{G}^1)|}\sum_{l=1}^{|\mathbb{G}(\mathcal{G}^2)|} K_F(v_i^1, v_j^2, \mathcal{G}_k^1, \mathcal{G}_l^2)\\
&\qquad \sum_{m=1}^{|V(\mathcal{G}_k^1)|}\sum_{n=1}^{|V(\mathcal{G}_l^2)|} J_\infty(v_m^1, v_n^2, \mathcal{G}_k^1, \mathcal{G}_l^2)\\
&+\Bigg(K_V(v_i^1, v_j^2) + K_V(v_i^1, v_j^2) \sum_{k=1}^{|\mathbb{G}(\mathcal{G}^1)|}\sum_{l=1}^{|\mathbb{G}(\mathcal{G}^2)|} K_F(v_i^1, v_j^2, \mathcal{G}_k^1, \mathcal{G}_l^2)\\
&\qquad \sum_{m=1}^{|V(\mathcal{G}_k^1)|}\sum_{n=1}^{|V(\mathcal{G}_l^2)|} J_\infty(v_m^1, v_n^2, \mathcal{G}_k^1, \mathcal{G}_l^2)\Bigg)\\
&\quad\cdot\left(\sum_{k=1}^{|V(\mathcal{G}^1)|}\sum_{l=1}^{|V(\mathcal{G}^2)|} K_E(v_i^1, v_j^2, v_k^1, v_l^2) J_\infty(v_k^1, v_l^2, \mathcal{G}^1, \mathcal{G}^2)\right)
\end{aligned} \tag{61}$$

Therefore, the computation of the HS-graph kernel finally comes down to solving Equation (61) and substituting the solutions into Equation (57).

The worst case time complexity when calculating $K^{\text{HSG}}(\mathcal{G}^1, \mathcal{G}^2)$ is nearly quadratic in the total number of vertices in the graph and subgraphs with a linear constant of $\mathcal{T}$, that is,

$$O\left(\mathcal{T} \cdot \left|\sum_{i=0}^{|\mathbb{G}(\mathcal{G}^1)|} V(\mathcal{G}_i^1)\right| \cdot \left|\sum_{j=0}^{|\mathbb{G}(\mathcal{G}^2)|} V(\mathcal{G}_j^2)\right|\right),$$

where $\mathcal{G}_0$ represents $\mathcal{G}$ itself.

4.4.2 Convergence Condition

The convergence condition needed to calculate the HS-graph kernel is as follows:

Theorem 2 *The infinite positive sequence* $\lim_{\mathcal{T}\to\infty} J_{\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2)$ *converges for any* $v_i^1 \in \mathcal{G}^1$ *and* $v_j^2 \in \mathcal{G}^2$*, if the following two inequalities hold for all* $v_i^1 \in \mathcal{G}^1$ *and* $v_j^2 \in \mathcal{G}^2$*, that is,*

$$0 \leq H_{\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) \leq H_{\mathcal{T}+1}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) \leq \mu < 1, \tag{62}$$

and

$$0 \leq \sum_{k=1}^{|V(\mathcal{G}^1)|} \sum_{l=1}^{|V(\mathcal{G}^2)|} K_E(v_i^1, v_j^2, v_k^1, v_l^2) \leq 1. \tag{63}$$

Proof. If I can prove that $J_{\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2)$ is a monotonically increasing sequence bounded above for all $v_i^1 \in \mathcal{G}^1$ and $v_j^2 \in \mathcal{G}^2$, $J_{\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2)$ is sure to converge by the monotone convergence theorem. This is in analogy to the proving of the convergence for Euler's constant, that is, $\lim_{n\to\infty}\left(1+\frac{1}{n}\right)^n = e$.

For this reason, I only need to prove monotonicity:

$$J_{\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) \leq J_{\mathcal{T}+1}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) \quad \forall v_i^1 \in \mathcal{G}^1, v_j^2 \in \mathcal{G}^2, \mathcal{T},$$

and bounded above:

$$J_{\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) < M \quad \forall v_i^1 \in \mathcal{G}^1, v_j^2 \in \mathcal{G}^2, \mathcal{T},$$

where M represents a certain constant.

First, I focus on the monotonically increasing sequence.

Lemma 1 *$J_{\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) \leq J_{\mathcal{T}+1}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2)$ is satisfied for all $v_i^1 \in \mathcal{G}^1, v_j^2 \in \mathcal{G}^2$ and $\mathcal{T}$ if the following inequality holds:*

$$0 \leq H_{\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) \leq H_{\mathcal{T}+1}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) \quad \forall v_i^1 \in \mathcal{G}^1, v_j^2 \in \mathcal{G}^2, \mathcal{T}. \tag{64}$$

Proof. The proof is achieved by induction on $\mathcal{T}$.

According to Equation (59), if $\mathcal{T} = 1$, then

$$J_1(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) - J_0(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) = H_1(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) - 0 \geq 0 \quad \forall v_i^1 \in \mathcal{G}^1, v_j^2 \in \mathcal{G}^2.$$

That is, $J_{\mathcal{T}} \leq J_{\mathcal{T}+1}$ is satisfied for $\mathcal{T} = 1$.

Then, assuming that $J_{\mathcal{T}} \leq J_{\mathcal{T}+1}$ is satisfied for $\mathcal{T} > 1$,

$$J_{\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) - J_{\mathcal{T}-1}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) \geq 0 \quad \forall v_i^1 \in \mathcal{G}^1, v_j^2 \in \mathcal{G}^2. \tag{65}$$

We can show that it is also true for $\mathcal{T} + 1$:

$$\begin{aligned}
& J_{\mathcal{T}+1}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) - J_{\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) \\
= {} & H_{\mathcal{T}+1}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) + H_{\mathcal{T}+1}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) \\
& \qquad \cdot \left(\sum_{k=1}^{|V(\mathcal{G}^1)|} \sum_{l=1}^{|V(\mathcal{G}^2)|} K_E(v_i^1, v_j^2, v_k^1, v_l^2) J_{\mathcal{T}}(v_k^1, v_l^2, \mathcal{G}^1, \mathcal{G}^2) \right) \\
& - H_{\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) + H_{\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) \\
& \qquad \cdot \left(\sum_{k=1}^{|V(\mathcal{G}^1)|} \sum_{l=1}^{|V(\mathcal{G}^2)|} K_E(v_i^1, v_j^2, v_k^1, v_l^2) J_{\mathcal{T}-1}(v_k^1, v_l^2, \mathcal{G}^1, \mathcal{G}^2) \right) \\
= {} & \big(H_{\mathcal{T}+1}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) - H_{\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) \big) \\
& + \sum_{k=1}^{|V(\mathcal{G}^1)|} \sum_{l=1}^{|V(\mathcal{G}^2)|} K_E(v_i^1, v_j^2, v_k^1, v_l^2) \big(H_{\mathcal{T}+1}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) J_{\mathcal{T}}(v_k^1, v_l^2, \mathcal{G}^1, \mathcal{G}^2) \\
& \qquad - H_{\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) J_{\mathcal{T}-1}(v_k^1, v_l^2, \mathcal{G}^1, \mathcal{G}^2) \big) \\
\geq {} & 0, \quad \forall v_i^1 \in \mathcal{G}^1, v_j^2 \in \mathcal{G}^2, \mathcal{T} \geq 2.
\end{aligned}$$

This can be simply derived from the conditions (64) in Lemma 2 and Inequality (65) of the assumption, and the fact that $H_{\mathcal{T}}$ and $J_{\mathcal{T}}$ are always non-negative. That is,

$$H_{\mathcal{T}+1}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2)J_{\mathcal{T}}(v_k^1, v_l^2, \mathcal{G}^1, \mathcal{G}^2) - H_{\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2)J_{\mathcal{T}-1}(v_k^1, v_l^2, \mathcal{G}^1, \mathcal{G}^2) \geq 0,$$

where

$$H_{\mathcal{T}+1}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) - H_{\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) \geq 0,$$

and

$$J_{\mathcal{T}}(v_k^1, v_l^2, \mathcal{G}^1, \mathcal{G}^2) - J_{\mathcal{T}-1}(v_k^1, v_l^2, \mathcal{G}^1, \mathcal{G}^2) \geq 0.$$

Hence the result follows by induction. Thus, I can say that for any $v_i^1 \in \mathcal{G}^1, v_j^2 \in \mathcal{G}^2$, $J_{\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2)$ and $\mathcal{T}$ is a monotonically increasing sequence because $J_{\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) \leq J_{\mathcal{T}+1}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2)$ is always satisfied. □

Next, I consider whether $J_{\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2)$ is bounded above.

Lemma 2 *$J_{\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2)$ for any $v_i^1 \in \mathcal{G}^1$, $v_j^2 \in \mathcal{G}^2$ and $\mathcal{T} = \{1, 2, \ldots, \infty\}$ are bounded above by $\frac{1}{1-\mu}$, that is, $J_{\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) < \frac{1}{1-\mu}$, if the following two inequalities hold*

$$0 < H_{\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) \leq \mu < 1 \quad \forall v_i^1 \in \mathcal{G}^1, v_j^2 \in \mathcal{G}^2, \mathcal{T}, \tag{66}$$

and

$$0 \leq \sum_{k=1}^{|V(\mathcal{G}^1)|} \sum_{l=1}^{|V(\mathcal{G}^2)|} K_E(v_i^1, v_j^2, v_k^1, v_l^2) \leq 1 \quad \forall v_i^1 \in \mathcal{G}^1, v_j^2 \in \mathcal{G}^2. \tag{67}$$

Proof. $J_{\mathcal{T}}$ represents the sum of $K_{L,\mathcal{T}}^{\varpi}$, that is:

$$\begin{aligned} J_{\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) &= \sum_{L=1}^{\mathcal{T}} K_{L,\mathcal{T}}^{\varpi}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) \\ &= K_{1,\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) + K_{2,\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) + \\ &\qquad \ldots + K_{\mathcal{T},\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2). \end{aligned}$$

Each term in the above equation can be individually bounded above by conditions (66) and (67) as follows:

$$\begin{aligned}
K_{1,\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) &= H_{\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) \leq \mu \\
K_{2,\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) &= H_{\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) \\
&\quad \left(\sum_{k=1}^{|V(\mathcal{G}^1)|} \sum_{l=1}^{|V(\mathcal{G}^2)|} K_E(v_i^1, v_j^2, v_k^1, v_l^2) H_{\mathcal{T}-1}(v_k^1, v_l^2, \mathcal{G}^1, \mathcal{G}^2) \right) \leq \mu^2 \\
&\vdots \\
K_{L,\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) &= H_{\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) \\
&\quad \cdot \left(\cdots \left(\sum_{k=1}^{|V(\mathcal{G}^1)|} \sum_{l=1}^{|V(\mathcal{G}^2)|} K_E(v_i^1, v_j^2, v_k^1, v_l^2) H_{\mathcal{T}-(L-1)}(v_k^1, v_l^2, \mathcal{G}^1, \mathcal{G}^2) \right) \cdots \right) \\
&\leq \mu^L.
\end{aligned}$$

When $K_{L,\mathcal{T}}$ has L terms of $H_{\mathcal{T}}$, it can be proved to be bounded by μ^L. Then we can obtain the following inequality:

$$\begin{aligned}
&\sum_{L=1}^{\mathcal{T}} K_{L,\mathcal{T}}^{\varpi}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) \\
&= K_{1,\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) + K_{2,\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) + \ldots + K_{\mathcal{T},\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) \\
&< \mu + \mu^2 + \ldots + \mu^{\mathcal{T}} \\
&= \sum_{L=1}^{\mathcal{T}} \mu^L.
\end{aligned}$$

$\sum_{L=1}^{\mathcal{T}} \mu^L$ is known as a *geometric series* and is known to converge if and only if $|\mu| < 1$. In this case the limit is given by $\lim_{\mathcal{T}\to\infty} \sum_{L=1}^{\mathcal{T}} \mu^L = \frac{1}{1-\mu}$

Therefore, we can obtain the least upper bound of $J_{\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2)$ as follows:

$$\lim_{\mathcal{T}\to\infty} J_{\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) < \lim_{\mathcal{T}\to\infty} \sum_{L=1}^{\mathcal{T}} \mu^L = \frac{1}{1-\mu}, \quad \forall v_i^1 \in \mathcal{G}^1, v_j^2 \in \mathcal{G}^2.$$

Finally, Theorem 1 is proved from Lemma 2 and Lemma 3 that show for any $v_i^1 \in \mathcal{G}^1, , v_j^2 \in \mathcal{G}^2$, $J_{\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2)$ is a monotonically increasing sequence bounded above under conditions (62) and (63) of Theorem 1.

□

To satisfy these convergence conditions, first, I introduce normalized weighting for the edges.

Let $K'_E(v_i^1, v_j^2, v_k^1, v_l^2)$ be a normalized edge of K_E with respect to v_i^1 and v_j^2:

$$K'_E(v_i^1, v_j^2, v_k^1, v_l^2) = \frac{K_E(v_i^1, v_j^2, v_k^1, v_l^2)}{\sum_{m=1}^{|V(\mathcal{G}^1)|} \sum_{n=1}^{|V(\mathcal{G}^2)|} K_E(v_i^1, v_j^2, v_m^1, v_n^2)}. \tag{68}$$

Then, K'_E has a value between 0 and 1, and satisfies

$$\sum_{k=1}^{|V(\mathcal{G}^1)|} \sum_{l=1}^{|V(\mathcal{G}^2)|} K'_E(v_i^1, v_j^2, v_k^1, v_l^2) = 1, \quad \forall v_i^1, v_j^2. \tag{69}$$

$K'_E(v_i^1, v_j^2, v_k^1, v_l^2)$ satisfies the latter condition of Theorem 1.

In the same way, we normalize K_F,

$$K'_F(v_i^1, v_j^2, \mathcal{G}_k^1, \mathcal{G}_l^2) = \frac{K_F(v_i^1, v_j^2, \mathcal{G}_k^1, \mathcal{G}_l^2)}{\sum_{m=1}^{|\mathbb{G}(\mathcal{G}^1)|} \sum_{n=1}^{|\mathbb{G}(\mathcal{G}^2)|} K_F(v_i^1, v_j^2, \mathcal{G}_m^1, \mathcal{G}_n^2)}. \tag{70}$$

Then, I introduce $H'_{\mathcal{T}}$, which is the normalized value of $H_{\mathcal{T}}$:

$$\begin{aligned} & H'_{\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) \\ &= \sigma \cdot K_V(v_i^1, v_j^2) + (1-\sigma) \cdot K_V(v_i^1, v_j^2) \\ &\cdot \sum_{m=1}^{|\mathbb{G}(\mathcal{G}^1)|} \sum_{n=1}^{|\mathbb{G}(\mathcal{G}^2)|} K'_F(v_i^1, v_j^2, \mathcal{G}_m^1, \mathcal{G}_n^2) \sum_{k=1}^{|V(\mathcal{G}^1)|} \sum_{l=1}^{|V(\mathcal{G}^2)|} J'_{\mathcal{T}-1}(v_k^1, v_l^2, \mathcal{G}_m^1, \mathcal{G}_n^2), \end{aligned} \tag{71}$$

where

$$J'_{\mathcal{T}-1}(v_k^1, v_l^2, \mathcal{G}_m^1, \mathcal{G}_n^2) \quad = \frac{J_{\mathcal{T}-1}(v_k^1, v_l^2, \mathcal{G}_m^1, \mathcal{G}_n^2)}{\sqrt{K^{HSG}(\mathcal{G}_m^1, \mathcal{G}_m^1) \cdot K^{HSG}(\mathcal{G}_n^2, \mathcal{G}_n^2)}}, \tag{72}$$

and $\sigma (0 \leq \sigma \leq 1)$, which is introduced as a tunable parameter of the ratio between the evaluation of the matching of the subgraphs connected with vertical edges: if $\sigma = 1$ holds, subgraphs connected with vertical edges are not used at all. The value of $H'_{\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2)$ lies between 0 and 1, $H'_{\mathcal{T}} = 1$ if and only if all the subgraphs connected with vertical edges to v_i^1 and v_j^2 are isomorphic.

Then, I introduce a parameter for the convergence ratio $0 \leq \mu < 1$.

$$H''_{\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) = \mu \cdot H'_{\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) \tag{73}$$

Parameter μ indicates the decay factor of the hi-label sequence size. Moreover, a smaller μ leads to faster convergence.

$H''_{\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2)$ satisfies the condition (62) of Theorem 1, namely the convergence condition for HS-graph kernels, that is,

$$0 \leq H''_{\mathcal{T}}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) \leq H''_{\mathcal{T}+1}(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) \leq \mu < 1 \quad \forall v_i^1 \in \mathcal{G}^1, v_j^2 \in \mathcal{G}^2, \mathcal{T}.$$

By substituting K'_E and $H''_{\mathcal{T}}$ into K_E and $H_{\mathcal{T}}$ of Equation (59), respectively , we can calculate HS-graph kernels that satisfy the convergence condition.

4.4.3 Efficient Computation for Acyclic Graph

Let us now consider a kernel that accepts only *acyclic* HS-graphs. In this case, we can calculate the kernel much more efficiently by using dynamic programming technique.

Before explaining an efficient computation method, I define the meaning of acyclicity for HS-graphs.

Definition 6 *(Hierarchically Structured Directed Acyclic Graph (HDAG)) An HS-graph $\mathcal{G} = (V, E, \mathbb{G}, F, \mathcal{A})$ is an HDAG if and only if there is no hi-walk where the same vertex appears more than twice.*

In other words, when all hi-walks in $\mathcal{G}$ are hi-paths, then $\mathcal{G}$ is an HDAG.

If an HS-graph is acyclic, vertices can be sorted in partial order (topological order) under the following two conditions: (1) $v_i \prec v_j$ for every directed edge $e_{i,j} \in E(\mathcal{G})$, (2) $\{V(\mathcal{G}_j)\} \prec v_i$ for every vertical edges $f_{i,j} \in F(\mathcal{G})$. Therefore, we can employ the *dynamic programming* technique during kernel computation. That is, if we calculate the kernel under the partial order of vertices, the values that are needed to calculate $K(v_i^1, v_j^2)$ have already been calculated in the previous calculation $K(v_k^1, v_l^2)$, where $1 \le k < i \le |\mathcal{G}^1|$ and $1 \le l < j \le |\mathcal{G}^2|$ in the partial order of vertices.

An efficient calculation formula for HDAGs $\mathcal{G}^1$ and $\mathcal{G}^2$ is written as:

$$K^{\text{HDAG}}(\mathcal{G}^1, \mathcal{G}^2) = \sum_{k=1}^{|\mathcal{G}^1|} \sum_{l=1}^{|\mathcal{G}^2|} J(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2),$$

where

$$\begin{aligned} J(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) &= H(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) + H(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) \\ &\quad \cdot \left(\sum_{k=1}^{i} \sum_{l=1}^{j} K_E(v_i^1, v_j^2, v_k^1, v_l^2) J(v_k^1, v_l^2, \mathcal{G}^1, \mathcal{G}^2) \right), \end{aligned} \tag{74}$$

and

$$\begin{aligned} &H(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2) \\ &= K_V(v_i^1, v_j^2) + K_V(v_i^1, v_j^2) \sum_{k=1}^{|\mathbb{G}(\mathcal{G}^1)|} \sum_{l=1}^{|\mathbb{G}(\mathcal{G}^2)|} K_F(v_i^1, v_j^2, \mathcal{G}_k^1, \mathcal{G}_l^2) \sum_{m=1}^{i-1} \sum_{n=1}^{j-1} J(v_m^1, v_n^2, \mathcal{G}_k^1, \mathcal{G}_l^2). \end{aligned}$$

For HDAG, $J(v_m^1, v_n^2, \mathcal{G}_k^1, \mathcal{G}_l^2)$, which is on the right-hand side in equation (74), is calculated in the previous step of $J(v_i^1, v_j^2, \mathcal{G}^1, \mathcal{G}^2)$, because $m < i$ and $n < j$ hold under the partial order defined above.

The worst case time complexity for calculating $K^{\text{HDAG}}(\mathcal{G}^1, \mathcal{G}^2)$ is quadratic in the total number of vertices in the graph and subgraphs, that is,

$$O\left(\left| \sum_{i=0}^{|\mathbb{G}(\mathcal{G}^1)|} V(\mathcal{G}_i^1) \right| \cdot \left| \sum_{j=0}^{|\mathbb{G}(\mathcal{G}^2)|} V(\mathcal{G}_j^2) \right| \right).$$

Therefore, we can calculate the HDAG kernel much more efficiently than the HS-graph kernel. Moreover, we do not need to satisfy the convergence conditions described in the previous section. This is a trade-off between the restriction of the acyclic graph and the convergence conditions.

4.5 Experiments

This section describes experiments. The aim is to confirm the effectiveness of the richer structural information that reflects the syntactic and semantic information within texts, which can be dealt with now for the first time by using the proposed method. Intuitively, we can say it is a better approach for dealing with richer information. However, no previous paper has shown that richer structural information improves the performance of NLP tasks, and so we do not know if dealing with richer information really improves performance. Therefore, I undertook an experiment to verify the statement.

In order to accomplish this goal I selected an appropriate real NLP task, *question classification*. The question classification task can be regarded as a sort of text classification task and involves mapping a given question into a pre-defined question type. However, it is known that the contextual and semantic features within texts are required if we are to obtain better results [Li02, Suzuki03c], the traditional topic based text classification, because the main point of this task is to classify the *intention* of a question. This task is strongly related to one of the latest topics in NLP, *sentiment classification*, whose purpose is to classify given texts into semantically related classes, such as polarity, subjectivity, intention and emotion. For the above reason, this task is suited to evaluating how well we can evaluate the text contextually and semantically.

4.5.1 Data

I used QC data provided by [Li02] for *English question classification* (EQC) and [Suzuki03c] for *Japanese question classification* (JQC).

The EQC data has 5952 questions in 50 fine question types. Table 9 shows all the question types and the number of questions in each question type in the EQC data.

The JQC data has 5011 questions in 150 question types, which are defined in the CRL QA-data[5] . However, it seems unnecessary to use all 150 question types, because many question types have only a few questions. I selected 16

[5] http://www.cs.nyu.edu/~sekine/PROJECT/CRLQA/

Table 9. Question types in English question classification data (50 types); ABBR: abbreviations, DESC: description and abstract concepts, ENTY: entities, HUM: human beings, LOC: locations and NUM: numeric values

	Question type	# of questions		Question type	# of questions
1	ABBR:abbreviation	17	26	ENTY:term	100
2	ABBR:expression	78	27	ENTY:vehicle	31
3	DESC:definition	544	28	ENTY:word	26
4	DESC:description	281	29	HUM:description	50
5	DESC:manner	278	30	HUM:group	195
6	DESC:reason	197	31	HUM:individual	1017
7	ENTY:animal	128	32	HUM:title	26
8	ENTY:body	18	33	LOC:city	147
9	ENTY:color	50	34	LOC:country	158
10	ENTY:creative	207	45	LOC:mountain	24
11	ENTY:currency	10	36	LOC:other	514
12	ENTY:disease and medicine	105	37	LOC:state	73
13	ENTY:event	58	38	NUM:code	9
14	ENTY:food	107	39	NUM:count	372
15	ENTY:instrument	11	40	NUM:date	265
16	ENTY:language	18	41	NUM:distance	50
17	ENTY:letter	9	42	NUM:money	74
18	ENTY:other	229	43	NUM:order	6
19	ENTY:plant	18	44	NUM:other	64
20	ENTY:product	46	45	NUM:percent	30
21	ENTY:religion	4	46	NUM:period	83
22	ENTY:sport	63	47	NUM:speed	15
23	ENTY:substance	56	48	NUM:temperature	13
24	ENTY:symbol	11	49	NUM:size	13
25	ENTY:technique	39	50	NUM:weight	15
				Total	5952

appropriate question types. Question types are defined in a tree structure, and the 16 question types I selected are located in the second level from the top-node of the tree. Table 10 shows all the question types and the number of questions in each question type in the JQC data.

Table 10. Question types in Japanese question classification data (16 types)

	Question type	# of question
1	NAME-PERSON	824
2	NAME-ORGANIZATION	733
3	NAME-LOCATION	752
4	NAME-FACILITY	147
5	NAME-PRODUCT	564
6	NAME-EVENT	143
7	NAME-TITLE	97
8	TIME_TOP-TIMEX	652
9	TIME_TOP-PERIODX	125
10	NUMEX-MONEY	187
11	NUMEX-PERCENT	104
12	NUMEX-FREQUENCY	27
13	NUMEX-AGE	58
14	NUMEX-MEASUREMENT	133
15	NUMEX-COUNTX	326
16	Other	139
	Total	5011

Table 11. Examples of English question classification data

Question type	Question
NUM:date	When did Hawaii become a state ?
LOC:other	What is the highest dam in the U.S. ?
LOC:city	What is the oldest city in the United States ?

Table 11 shows examples of the English question classification data.

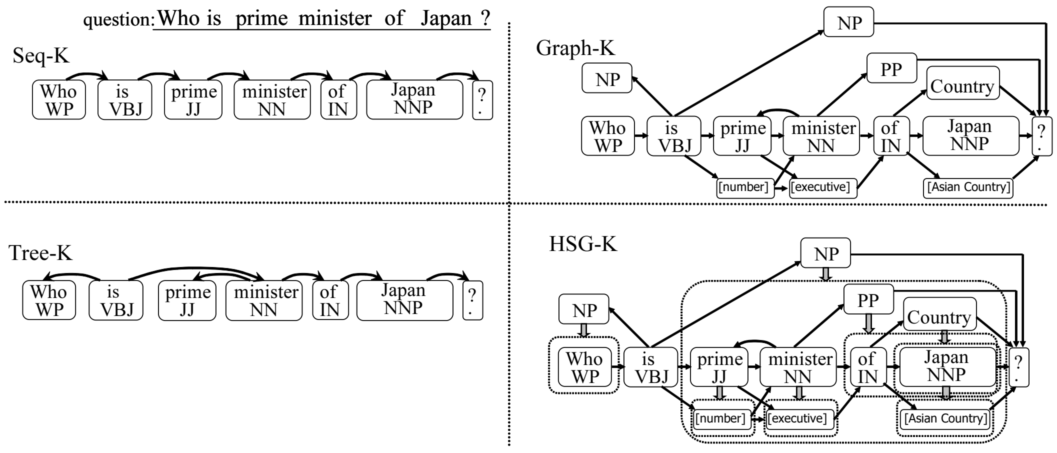


Figure 15. Examples of input data for each method

4.5.2 Comparison Methods

The objectives of the experiments are to observe the efficiency of the richer information in NLP tasks. I compared the performance of the proposed kernel, the HS-graph Kernel (HSG-K), with Graph Kernels (Graph-K) [Kashima03], Tree kernels (Tree-K) [Kashima02], and sequence kernels (Seq-K) [Cancedda03]. Moreover, I evaluated the *bag-of-words* kernel (BOW-K) [Joachims98], that is, the bag-of-words with polynomial kernels, as the baseline method. The main difference between the methods is the ability to deal with syntactic and semantic information within texts.

Figure 15 shows example input objects for each method. Moreover, Table 12 outlines the information (features) of the text dealt with in each of the compared methods. As shown in Table 12, Seq-K only considers word order, Tree-K deals with the dependency structures of words, Graph-K deals with all of the words, phrases, named entities, and semantic categories with dependency structures, and HSG-K deals with the same information as Graph-K plus the sub-structures of each segment. That is, a comparison of the performance of these methods can clarify the impact of the structural information.

I evaluated the performance of HSG-K with the parameter $\mu = \{0.1, 0.3, 0.5, 0.7, 0.9\}$

Table 12. Information dealt with in each comparison method: W: words, N: named entities, P: phrases, S: semantic categories, wo: word orders, dep: dependencies, sb: sub-structures of segments

	Types of label				Types of structure		
Method	W	N	P	S	wo	dep	sb
HSG-K	o	o	o	o	o	o	o
Graph-K	o	o	o	o	o	o	
Tree-K	o					o	
Seq-K	o				o		
BOW-K	o						

in Equation (73) and $\sigma = \{0.2, 0.4, 0.6, 0.8\}$ in Equation (71) for all the experiments in this chapter. The parameter $\mu = \{0.1, 0.3, 0.5, 0.7, 0.9\}$ was also applied with Seq-K, Tree-K and Graph-K, which is a decay factor according to the sub-structure size as well as HSG-K. Moreover, I evaluated the performance of BOW-K with $d = \{1, 2, 3, 4\}$, which represents the degree of polynomial kernels.

I used many NLP resources to obtain these structured input objects. For English data, I first used the OAK system[6] to obtain POS and NE tags. Then, I parsed the input texts by using the Collins-Parser [Collins03]. Finally, I add the hypernym from Word-Net [Fellbaum98].

For Japanese data, I first analyzed input data by using ChaSen [Asahara00] for morphological analysis and a POS tagger. Then, I used CahoCha [Kudo02] as a chunking and dependency analyzer, SVM-NE tagger [Isozaki02] as a named entity tagger, and *Goi-taikei* [Ikehara97] for semantic information.

4.5.3 Performance Evaluation Method

Basically, I evaluated the overall performance by using the average accuracy of *ten fold cross validation.* First, I divided data set into ten sets. Then, I used the Support Vector Machine (SVM) [Vapnik95] as a kernel-based machine learning

[6] http://nlp.cs.nyu.edu/oak/

algorithm to learn a decision function by using nine sets. While a question classification task is generally a multi-class classification task and an SVM is a binary classifier, I used the one vs rest model to determine the final estimated label for a given question. I calculated *label accuracy*, which is the accuracy of estimated labels compared with the correct labels for the remaining set.

I iterated the above procedure ten times with changing the divided sets for training and test data. Finally, I evaluated the performance by using the average label accuracy of the ten iterated evaluations. Additionally, I tested the difference in average label accuracy by using a paired Wilcoxon signed rank test, which is a non-parametric statistical significance test.

4.5.4 Results and Discussion

Comparison with Conventional Kernels

In this set of experiments, I analyze the effect of using richer information within text for NLP tasks. More specifically I compare the performance of the kernels with that of sequence and tree kernels, which substitutes one of the latest methods for handling structural information within text in the NLP field. Since text is easily handled as a sequence of words or a parsed tree, these methods are natural methods for use with text.

The results of these comparisons are described in Tables 13 and 14 for EQC and JQC, respectively. The label accuracy (acc.) is given. It should be noted that these tables also show the standard deviations (S.D.). μ and σ for Seq-K, Tree-K and HSG-K, and d for BOW-K show each of the parameters explained in Section 4.5.2. These tables only show the best results obtained with parameters μ, σ and d. The last column in the tables shows results for the Wilcoxon signed rank test comparing the performance of the proposed method and those of the other methods, where one (*), two (**) and three (***) asterisks represent p-values of $p < 0.05$, $p < 0.01$, and $p < 0.005$, respectively.

According to these results, the proposed methods statistically outperformed both sequence kernels and tree kernels. In the question classification task, a given question is classified as a question type that reflects the intention of the question.

Table 13. Experimental results: Label accuracy for each question (EQC): acc. and S.D. represent the average label accuracy and its standard deviation. μ, σ and d represent the parameters for each method. *, **, and *** represent $p < 0.05$, $p < 0.01$ and $p < 0.005$ of the Wilcoxon signed rank test, respectively.

Comparison methods	Performance		Parameters		Statistical test
	acc.	S.D.	μ	σ	$p < \{0.05, 0.01, 0.005\}$
HSG-K	**0.844**	0.0116	0.3	0.4	
Graph-K	0.838	0.0108	0.3	-	*
Tree-K	0.822	0.0142	0.3	-	**
Seq-K	0.831	0.0131	0.3	-	*
	acc.	S.D.	d		
BOW-K	0.799	0.0153	2		***

Table 14. Experimental results: Label accuracy for each question (JQC): acc. and S.D. represent the average label accuracy and its standard deviation. μ, σ and d represent the parameters for each method. *, **, and *** represent $p < 0.05$, $p < 0.01$ and $p < 0.005$ of the Wilcoxon signed rank test, respectively.

Comparison methods	Performance		Parameters		Statistical test
	acc.	S.D.	μ	σ	$p < \{0.05, 0.01, 0.005\}$
HSG-K	**0.817**	0.0098	0.5	0.2	
Graph-K	0.809	0.0089	0.5	-	*
Tree-K	0.804	0.0129	0.5	-	**
Seq-K	0.803	0.0104	0.5	-	**
	acc.	S.D.	d		
BOW-K	0.749	0.0160	2		***

Therefore, this result indicates that the richer structural information gives more clues with which to identify the differences in intentions inherently contained in the text.

It is interesting to note that the effect of using richer structural information leads to better performance both in English and Japanese. This fact also indicates that the approach using richer structural information is language oriented and we believe that we can improve the performance of NLP tasks for any other language.

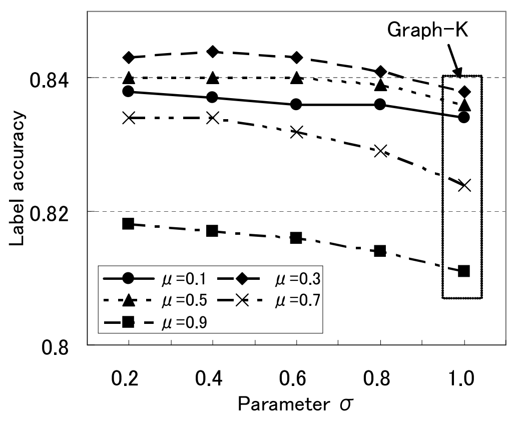
Effect of Hierarchical Structure: Parameter σ

I clarify the effect of using hierarchical structures that are constructed from subgraphs and vertical edges in my method. If we remove whole subgraphs and vertical edges from an HS-graph, it obviously becomes the same as a standard graph. That is, we now compare the performance of HS-graph kernels and graph kernels.

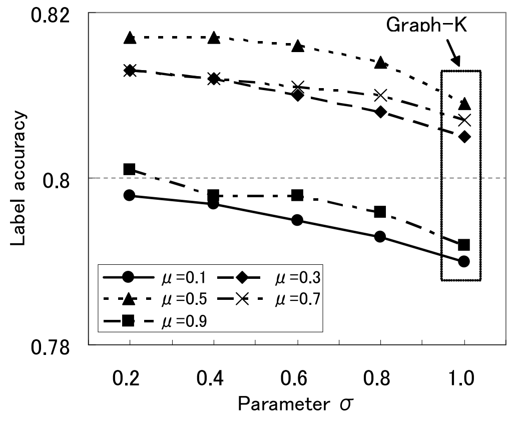
As mentioned earlier, parameter σ gives the relative ratio for evaluating the matching of sub-graphs for connected vertices. If we set $\sigma = 1$ for HS-graph kernels, that is, it can never evaluate the effect of sub-graphs, then it becomes equivalent to evaluating standard graph kernels.

To obtain experimental evidence of the efficiency of hierarchical structures, I evaluated the influence of σ on performance by changing the value. Figure 16 shows the results of this set of experiments in EQC and JQC. As shown in this figure, the performance deteriorates with larger μ, and this tendency is the same for all μ values. Moreover, $\sigma \neq 1$ always provides better performance than $\sigma = 1$. This means that subgraph information is a more certain indicator for identifying the intentions of the text. Generally, the subgraphs information is more precise than the information of the vertex itself that is connected with vertical edges. This may be the reason why evaluating the subgraphs improves the overall performance.

Thus, this result showed further evidence that the proposed method, HS-graph kernels, is a better approach than standard graph kernels for NLP tasks. This indicates that the relation between different levels of segments, such as phrases and named entities, provides informative features for solving real NLP tasks, and moreover, for understanding natural language.



(a) EQC



(b) JQC

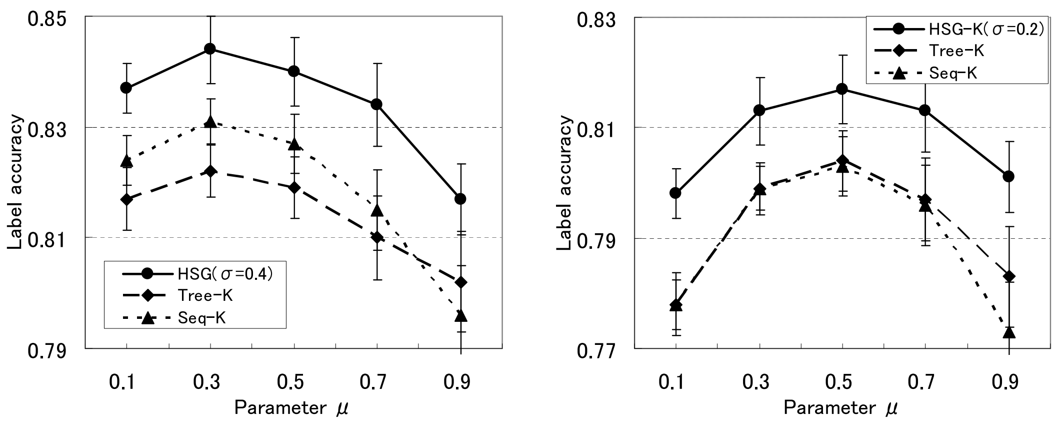
Figure 16. Effect of changing parameter σ

Effect of Kernel Parameter μ

Parameter μ gives the relative weight of the structure size. In previous studies, larger sub-structures are down-weighted compared with smaller ones, to avoid the over-fitting problem from occurring during learning. This parameter is used for sequence kernels and tree kernels as well as the proposed kernels.

My aim in this experiment was to check the influence of μ on performance. This experiment focuses only on $\sigma = 0.2$, which gave the best results in the previous section. Note that the tendency of the influence of μ on the performance obtained with other σ, including Graph-K ($\sigma = 1.0$), is almost the same (See Figure 16). Figure 17 shows the performance when μ was changed from 0.1 to 0.9 in intervals of 0.2.

As shown in the figure, the performance seems to have one peak point, which is nearly equal to optimal μ. Moreover, Seq-K and Tree-K also have the same tendency. This indicates that parameter μ may have one optimal value for a given set of data for each method. This is a positive result because it means if we change μ slightly the performance does not change rapidly and there is a direction for finding optimal μ.



(a) EQC (b) JQC

Figure 17. Effect of changing parameter μ

When we consider a larger μ, this supposed over-fitting problem will arise during SVM training [Collins01, Cancedda03]. On the other hand, with respect to a small μ, these supposed effective features may lack for training.

Precision when selecting optimal parameters (σ and μ)

As in the actual use of question classification, we somehow need to find the optimal (or near optimal) value for the parameters μ and σ. To evaluate the robustness of each parameter simply, we assessed the precision with which the optimal parameters, σ and μ, are chosen in each iteration. If the same parameter gives the best performance in each iteration, this means it is very robust.

At first, we tested the precision when selecting the parameter μ. The results of this set of experiments are shown in Table 15 for EQC and JQC. As shown in these tables, the selection of the best μ value is achieved in almost every iteration. More precisely, in EQC, the best value of $\mu = 0.3$ occurs 8 times out of 10, and in JQC, the best value of $\mu = 0.5$ occurs 6 times out of 10.

I then tested the precision when selecting the parameter σ, and Table 16 shows the results. In EQC, the best value of $\sigma = 0.4$ occurs 5 times out of 10, and in JQC, the best value of $\sigma = 0.4$ occurs 7 times out of 10. However,

μ	1	2	3	4	5	6	7	8	9	10
0.1	$\underline{0.832}$	0.841	0.842	0.839	0.849	$\underline{0.849}$	0.850	$\underline{0.842}$	0.829	0.820
$\underline{0.3}$	0.829	$\underline{0.856}$	$\underline{0.857}$	$\underline{0.844}$	$\underline{0.855}$	$\underline{0.849}$	$\underline{0.852}$	$\underline{0.842}$	$\underline{0.840}$	0.820
0.5	0.829	$\underline{0.856}$	0.855	0.840	0.842	0.839	0.845	0.835	0.839	0.822
0.7	0.819	0.849	0.847	0.834	0.839	0.830	0.835	0.834	0.827	$\underline{0.824}$
0.9	0.812	0.839	0.830	0.825	0.818	0.808	0.807	0.818	0.808	0.800

(a) EQC: $\sigma = 0.4$

μ	1	2	3	4	5	6	7	8	9	10
0.1	0.805	0.816	0.790	0.782	0.796	0.790	0.818	0.802	0.792	0.786
0.3	$\underline{0.819}$	$\underline{0.830}$	0.802	0.802	$\underline{0.824}$	0.798	0.826	0.818	0.810	$\underline{0.800}$
$\underline{0.5}$	$\underline{0.819}$	0.826	$\underline{0.806}$	0.810	$\underline{0.824}$	$\underline{0.810}$	0.834	$\underline{0.822}$	0.820	$\underline{0.800}$
0.7	0.803	0.824	0.802	$\underline{0.812}$	$\underline{0.824}$	0.808	$\underline{0.844}$	0.796	$\underline{0.822}$	0.792
0.9	0.783	0.808	0.786	0.804	0.806	0.800	0.832	0.792	0.812	0.784

(b) JQC: $\sigma = 0.2$

Table 15. Precision of optimal parameter μ for each iteration

the performance for different σ values is much closer than that for different μ values, and it seems to be slightly difficult to chose the optimal parameter in each iteration.

Calculation Speed

I compared the calculation speed for each method. I ran these tests on a "Linux PC with Opteron 2.4GHz". Tables 17 and 18 show the average evaluation time of one iteration in the experiments. In the table, 'ave. $|V(\mathcal{G})|$' indicates the average number of vertices in the training and test data, and '# of SVs.' represents the average number of support vectors in a model constructed by SVM training.

As shown in these tables, the proposed method requires a much greater calculation cost than Seq-K and Tree-K. The main reason for this observation lies in the different numbers of vertices in input objects, since the calculation costs

σ	1	2	3	4	5	6	7	8	9	10
0.2	0.827	$\underline{0.856}$	0.854	$\underline{0.845}$	0.850	0.847	0.845	$\underline{0.845}$	0.840	0.818
$\underline{0.4}$	$\underline{0.829}$	$\underline{0.856}$	0.857	0.844	$\underline{0.855}$	$\underline{0.849}$	$\underline{0.852}$	0.842	0.840	0.820
0.6	0.824	0.852	0.857	0.844	0.852	0.845	0.845	0.840	0.844	$\underline{0.824}$
0.8	0.824	0.849	$\underline{0.859}$	0.839	0.854	0.837	0.842	0.842	$\underline{0.845}$	$\underline{0.824}$
1.0	0.822	0.846	0.855	0.837	0.852	0.830	0.845	0.839	0.834	$\underline{0.824}$

(a) EQC: $\mu = 0.3$

σ	1	2	3	4	5	6	7	8	9	10
$\underline{0.2}$	$\underline{0.821}$	$\underline{0.830}$	0.802	$\underline{0.802}$	0.816	$\underline{0.798}$	$\underline{0.834}$	0.816	$\underline{0.816}$	$\underline{0.802}$
0.4	0.819	$\underline{0.830}$	0.802	$\underline{0.802}$	$\underline{0.824}$	$\underline{0.798}$	0.826	$\underline{0.818}$	0.810	0.800
0.6	0.819	0.828	$\underline{0.804}$	0.800	0.820	$\underline{0.798}$	0.824	$\underline{0.818}$	0.808	0.798
0.8	0.819	$\underline{0.830}$	0.794	0.794	0.820	0.796	0.824	0.812	0.806	0.798
1.0	0.817	0.822	0.792	0.796	0.814	0.796	0.818	0.812	0.810	0.800

(b) JQC: $\mu = 0.5$

Table 16. Precision of optimal parameter σ for each iteration

Table 17. Calculation cost for EQC

	Training		Test		
	time (sec.)	ave. $\lvert V(\mathcal{G})\rvert$	time (sec.)	ave. $\lvert V(\mathcal{G})\rvert$	# of SVs
HSG-K	4112.72	97261.2	493.85	10806.8	24616.8
Graph-K	3852.70	97261.2	455.43	10806.8	24245.0
Tree-K	1442.20	53454.6	150.20	5939.4	30024.0
Seq-K	1397.65	53454.6	143.80	5939.4	29960.1

are obtained as a function of the number of vertices. Since HSG-K and Graph-K are handling richer information than these kernels, the input objects of HSG-K and Graph-K usually have a larger number of vertices than those of Seq-K and Tree-K.

Table 18. Calculation cost for JQC

	Training		Test		
	time (sec.)	ave. $\|V(\mathcal{G})\|$	time (sec.)	ave. $\|V(\mathcal{G})\|$	# of SVs
HSG-K	5394.74	126266.4	586.03	14079.7	14478.5
Graph-K	4512.30	126266.4	513.60	14079.7	14186.3
Tree-K	3339.07	100490.4	329.06	11165.6	16038.7
Seq-K	3039.27	100490.4	321.55	11165.6	16410.7

Additionally, the average number of vertices of HSG-K and Graph-K is nearly double that of Seq-K and Tree-K in EQC, while that in JQC is about 1.3 times larger. This observation arises from the size of 'phrases'. That is, in Japanese, phrases (bunsetsu) are usually composed of more than two or three words, however, in English, most phrases are usually composed of one or two words.

Moreover, HSG-K and Graph-K reduced the number of support vectors. Since the input objects of HSG-K and Graph-K have rich textual information, the SVM seems to be able to find a better model with a small number of SVs. This fact indirectly proves the effect of rich information.

4.6 Extension for Natural Language Processing

We sometimes need to modify the kernel computation to adapt it to the target tasks. This section introduces some extensions of the proposed kernels for use in NLP.

4.6.1 Weights

With NL data, we sometimes would like to take into account with the weight factors of labels, vertices, and edges. This is because we have prior knowledge of the target task that is more important or not important. For example, we may know the score of $tf * idf$ [Salton83] for each label from large scale documents, and types of segments such as words, phrases or named entities.

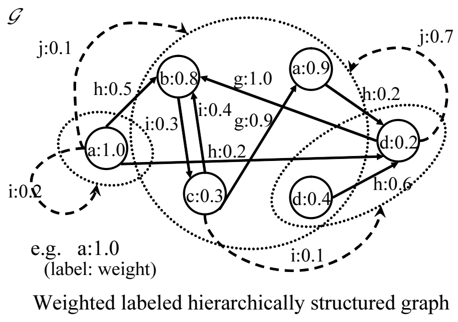


Figure 18. Example of weighted labeled HS-graph

Let $\mathcal{B} : V \cup E \cup F \cup \Gamma \rightarrow \mathbb{R}$ be a weight mapping for the labels, vertices, edges and vertical edges.

Definition 7 *(Weighted Labeled Hierarchically Structured Graph) A weighted labeled hierarchically structured graph can be represented by a 6-tuple, $\mathcal{G} = (V, E, \mathbb{G}, F, \mathcal{A}, \mathcal{B})$.*

Figure 18 shows an example of weighted labeled HS-graph.

Therefore, I redefine the sub-kernels of the labels, vertices, edges and vertical edges, in Equations (45), (46), (47), (48), respectively.

sub-kernels for labels

$$K_\Gamma(a_i, a_j) = w(a_i)w(a_j)\delta(a_i, a_j), \tag{75}$$

sub-kernels for vertices

$$K_V(v_i, v_j) = w(v_i)w(v_j)K_\Gamma(\tau(v_i), \tau(v_j)) \tag{76}$$

sub-kernels for edges

$$K_E(v_i^1, v_j^2, v_k^1, v_l^2) = \begin{cases} 0, & \text{if } e_{i,k} \notin E(\mathcal{G}^1) \text{ or } e_{j,l} \notin E(\mathcal{G}^2), \\ w(e_{i,k})w(e_{j,l})K_\Gamma(\tau(e_{i,k}), \tau(e_{j,l})) & \text{otherwise} \end{cases} \tag{77}$$

sub-kernels for vertical edges

$$K_F(v_i^1, v_j^2, \mathcal{G}_k^1, \mathcal{G}_l^2) = \begin{cases} 0, & \text{if } f_{i,k} \notin F(\mathcal{G}^1) \text{ or } f_{j,l} \notin F(\mathcal{G}^2), \\ w(f_{i,k})w(f_{j,l})K_\Gamma(\tau(f_{i,k}), \tau(f_{j,l})) & \text{otherwise} \end{cases} \tag{78}$$

where $w(x)$ is a function that returns the corresponding weight of $x \in V \cup E \cup F \cup \Gamma$.

4.6.2 Multiple Edges

Let $E' \subseteq V \times \Gamma \times V$ and $F' \subseteq V \times \Gamma \times \mathbb{G}$ be a set consisting of a directed edge and a vertical edge allowing plural edges with respect to labels, respectively. Then, a *weighted labeled hierarchically structured graph with multiple edges* can be defined by a 6-tuple, $\mathcal{G} = (V, E', \mathbb{G}, F', \mathcal{A}, \mathcal{B})$.

I only need to redefine the sub-kernels for the edges and vertical edges as follows:

sub-kernels for multiple edges

$$K_{E'}(v_i^1, v_j^2, v_k^1, v_l^2) = \begin{cases} 0, \quad \text{if } \; e_{i,k} \notin E'(\mathcal{G}^1) \; \text{ or } \; e_{j,l} \notin E'(\mathcal{G}^2), \\ \displaystyle\sum_{e_{i,k} \in E'(\mathcal{G}^1)} \sum_{e_{j,l} \in E'(\mathcal{G}^2)} w(e_{i,k}) w(e_{j,l}) K_\Gamma(\tau(e_{i,k}), \tau(e_{j,l})) \\ \qquad\qquad \text{otherwise} \end{cases} \tag{79}$$

sub-kernels for multiple vertical edges

$$K_F(v_i^1, v_j^2, \mathcal{G}_k^1, \mathcal{G}_l^2) = \begin{cases} 0, \quad \text{if } \; f_{i,k} \notin F'(\mathcal{G}^1) \; \text{ or } \; f_{j,l} \notin F'(\mathcal{G}^2), \\ \displaystyle\sum_{f_{i,k} \in F'(\mathcal{G}^1)} \sum_{f_{j,l} \in F'(\mathcal{G}^2)} w(f_{i,k}) w(f_{j,l}) K_\Gamma(\tau(f_{i,k}), \tau(f_{j,l})) \\ \qquad\qquad \text{otherwise} \end{cases} \tag{80}$$

4.6.3 Vertex Skip (Virtual edge)

When we apply kernels to NLP tasks, we must sometimes deal not only with exact structural matching but also with soft structural matching frameworks. This is because text can have the same contextual meaning even if the structure is slightly different.

To overcome this issue, I introduce the *vertex skip* framework during the matching of hi-walks in kernel computation. This framework achieves soft structure matching automatically during kernel computation.

There are several possible ways to realize a vertex skip framework. In this section, I propose a method that adds virtual edges thus allowing us to skip vertices. Virtual edges are created between every pair of vertices only if the edges of the path between the vertices have the same label. The label of a virtual edge

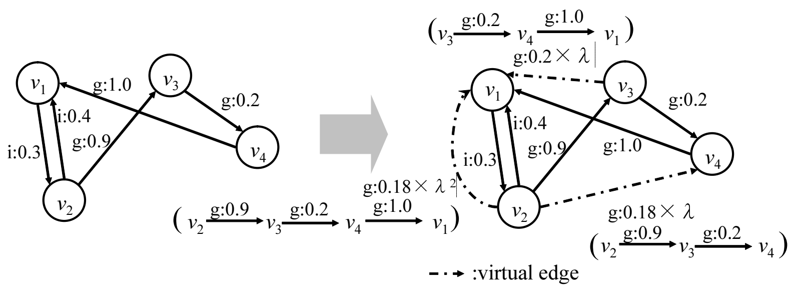

v_1
v_2
v_3
v_4
g:1.0
g:0.2
i:0.4
g:0.9
i:0.3
$(v_2 \xrightarrow{g:0.9} v_3 \xrightarrow{g:0.2} v_4 \xrightarrow{g:1.0} v_1)$
g:0.18× λ²
$(v_3 \xrightarrow{g:0.2} v_4 \xrightarrow{g:1.0} v_1)$
g:0.2× λ
g:0.18× λ
$(v_2 \xrightarrow{g:0.9} v_3 \xrightarrow{g:0.2} v_4)$
:virtual edge

Figure 19. Examples of a vertex skip framework

Comparison methods	Performance		Parameters			
	acc.	S.D.	λ	μ	σ	increase
HSG-K	**0.847**	0.0105	0.3	0.7	0.2	+0.003
Graph-K	0.842	0.0062	0.3	0.7	-	+0.004
Tree-K	0.823	0.0144	0.3	0.3	-	+0.001
Seq-K	0.831	0.0131	0.3	0.3	-	+0.000

Table 19. Effect of vertex skip on performance (EQC)

is the same as that of the corresponding edge and the weight is the multiplied weight of the corresponding edge in the path. Moreover, I introduce a decay function $\Lambda_V(v)(0 < \Lambda_V(v) \leq 1)$, which represents the cost of skipping vertex v, that is, this decay function is multiplied by the weight of the virtual edge.

Figure 19 shows how to form virtual edges. Note that to realize the vertex skip framework, we need a framework for handling weights and multiple edges.

Then, we simply tested the effect of the vertex skip framework. Tables 19 and 20 show the best results under the vertex skip $\lambda = \{0.1, 0.3, 0.5, 0.7, 0.9\}$ with the combination of parameters $\mu = \{0.1, 0.3, 0.5, 0.7, 0.9\}$ and $\mu = \{0.2, 0.4, 0.6, 0.8, 1.0\}$.

As shown in these tables, the performance can be slightly improved by using vertex skips. The reason that the different μ were selected with HSG-K and

Comparison methods	Performance		Parameters			
	acc.	S.D.	λ	μ	σ	increase
HSG-K	**0.818**	0.0146	0.3	0.9	0.2	+0.001
Graph-K	0.810	0.0141	0.3	0.7	-	+0.001
Tree-K	0.804	0.0131	0.3	0.3	-	+0.000
Seq-K	0.805	0.0119	0.3	0.5	-	+0.002

Table 20. Effect of vertex skip on performance (JQC)

Graph-K is following. Since we handled virtual edges for vertex skips, the total number of the edges increased. Then, according to Equation (68), the weight of each edge decreased. As a result, a larger μ was selected with the vertex skip framework.

4.7 Related Work

Since natural language data take the form of sequences of words or parsed trees, sequence kernels [Lodhi02, Cancedda03] and tree kernels [Collins01, Kashima02], have been developed in the NLP field and shown to offer excellent results. Other examples in the bio-informatics field are sequence kernels [Jaakkola00, Leslie04] and graph kernels [Kashima03] that have been proposed mainly for classifying proteins or chemical compounds. Another type of graph kernel [Gärtner03] has also been introduced in the machine learning field. We have presented one kernel for HS-graphs based on a hi-label sequence. Therefore, this chapter is strongly related to these previous studies that designed kernels on discrete structures, especially graph kernels based on *random walks* [Kashima03, Gärtner03]. However, this chapter focused on data (text) representation for tasks such NLP application areas as question answering, text summarization and text classification. With these tasks, richer types of information within texts, such as syntactic and semantic information, are required if we are to obtain improved performance. Moreover, the experiments proved that the relation between different levels of segments provides informative clues for solving NLP tasks. However, integrated syntactic and semantic information are formed by very complex structures that cannot be written in simple structures, such as sequences, trees and standard graphs. Thus, HS-graph kernels is a better approach than standard graph kernels for NLP tasks.

If we remove all the subgraphs and vertical edges from HS-graphs, that is, they become standard labeled graphs, the proposed kernels calculate in the same way as graph kernels. More precisely, parameter σ in Equation (71) gives the relative ratio for evaluating the matching of sub-graphs for connected vertices. If we set $\sigma = 1$ for HS-graph kernels, that is, it never evaluates the effect of sub-graphs, then it becomes equivalent to evaluating standard graph kernels. We can also say that graph kernels are one special example of the proposed kernels, if the input object is constructed in standard graphs. Therefore, my method is a more generalized graph kernel framework.

In terms of data representation ability, sequences and trees are one special class of graphs. The proposed kernels generalize more than previously introduced discrete kernels, that is, sequence, tree and graph kernels. The proposed kernel can accept all structures of sequences, trees, and graphs.

4.8 Summary

This chapter proposed HS-graph kernels, which can handle integrated syntactic and semantic information present within texts. The proposed method provides a very generalized framework for handling natural language data by using a framework of discrete kernels. I evaluated the performance of HS-graph kernels with the English and Japanese question classification tasks, which are the real NLP tasks. The experiments showed that HS-graph kernels provided better performance than sequence, tree and graph kernels, and the baseline method bag-of-words kernels, which have no frameworks with which to handle all the linguistic information inherent in texts. Therefore, this result showed that the richer structural information that reflects syntactic and semantic information within text can provide substantial improvements for NLP tasks. Additionally, the experiments showed that the proposed method is a language independent method: both English and Japanese tasks were improved significantly in the same framework.

5. Statistical Feature Mining for Convolution Kernels

5.1 Introduction

Over the past few years, many machine learning methods have been successfully applied to tasks in *natural language processing* (NLP). In particular, state-of-the-art performance can be achieved with kernel methods, such as *Support Vector Machine* [Cortes95]. Examples include text categorization [Joachims98], chunking [Kudo02] and parsing [Collins01].

Another feature of this kernel methodology is that it not only provide 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 [Lodhi02], tree kernels [Collins01], and graph kernels [Suzuki03a], have been shown to offer excellent results.

These discrete kernels are related to *convolution kernels* [Haussler99], which provides the concept of kernels over discrete structures. Convolution kernels allow us to deal with 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 [Collins01, Cancedda03, Suzuki03b]. That is, since natural language data contain many types of labels, NLP tasks usually deal with extremely high dimension and sparse feature space. As a result, it is difficult ot train efficiently.

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

This chapter discusses this issue of convolution kernels, focusing particularly on sequence and tree kernels, and proposes a new method based on a statistical significance test. The proposed method deals only with those features that are statistically significant to given data (tasks). This means large significant sub-structures can be used without over-fitting. Moreover, by using *sub-structure mining* algorithms, the proposed method can be executed efficiently by embedding it in an original kernel calculation process, which is defined by the *dynamic-programming* (DP) based calculation.

Section 5.2 discusses one problem related to convolution kernels, the main topic of this chapter, and introduces some conventional methods for solving this problem. In Section 5.3, I propose a new approach based on statistical feature selection to offset the issue of convolution kernels using an example consisting of sequence kernels. Section 5.4 compares the performance of conventional methods with that of the proposed method by using several kinds of *semantic classification* tasks. The experimental results clarify the advantages of the proposed method.

5.2 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 (see Figure 2 in Section 2.2.1). The number of sub-structures in an input object usually becomes exponential against the input object size. Moreover, the number of labels $|\Sigma|$ (see definition of sequence and tree kernels in Sections 2.2.1 and 2.2.2) is generally very large. In some cases, even more than 10,000 sub-structures are enumerated, since words are treated as labels. As a result, the dimension of the feature space becomes extremely high, and most kernel values $K(X, Y)$ are very small compared to the kernel value of the object itself, $K(X, X)$. In this situation, it is difficult to train the convolution kernel approach effectively, and it will behave in accordance with the nearest neighbor rule. This means that we obtain a result that is very precise but with very low recall. This issue is described in detail in [Collins01].

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

For sequence kernels, [Cancedda03] 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 length is n or less. In addition to the sequence kernel, [Collins01] proposed a method that restricts the features based on sub-tree depth for tree kernels. These methods seem to work well on the surface, however, good results can only be achieved when n is very small, i.e. $n = 2$ or 3. For example, $n = 3$ showed the best performance for parsing in the experimental results reported by [Collins01], and $n = 2$ showed the best performance for the text classification task reported by [Cancedda03]. 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 useless. 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, offer the possibility of further improving the performance.

The aim of the work described in this chapter is to make it possible to use any significant sub-structure efficiently, regardless of its size, to solve NLP tasks.

5.3 Statistical Feature Mining

This section proposes a new approach to feature selection, which is based on a statistical significant test, in contrast to the conventional methods, which use sub-structure size.

For simplicity's sake, I have restricted the discussion to the two-class (positive and negative) supervised classification problem. This approach tests the statistical deviation of all sub-structures in the training samples between the appearance of positive samples and negative samples, and then, selects only sub-structures that are larger than a certain threshold τ as features. This allows us to select only the statistically significant sub-structures.

Table 21. Contingency table and notation for the chi-squared value

	c	$\bar{c}$	$\sum$ row
u	O_{uc}	$O_{u\bar{c}}$	O_u
$\bar{u}$	$O_{\bar{u}c}$	$O_{\bar{u}\bar{c}}$	$O_{\bar{u}}$
$\sum$ column	O_c	$O_{\bar{c}}$	N

This approach, which uses a statistical metric to select features, is quite natural. I 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, I briefly explain statistical metric, the chi-squared (χ^2) value, and provide an idea of how to select significant features. I then describe a method for embedding statistical feature selection into kernel calculation.

This chapter uses the notations defined in chapter 2.

5.3.1 Statistical Metric

There are many kinds of statistical metrics, including the chi-squared value, the correlation coefficient and mutual information. In this dissertation, I explain the proposed method by using the chi-squared (χ^2) value as a statistical metric.

First, I briefly explain how to calculate the χ^2 value by referring to Table 21. c and $\bar{c}$ represent the names of the positive class and the negative class, respectively. O_{ij}, where $i \in \{u, \bar{u}\}$ and $j \in \{c, \bar{c}\}$, represents the number of samples in each case. $O_{u\bar{c}}$, for instance, represents the number of u that appeared in $\bar{c}$. Let N be the total number of training samples. Since N and O_c are constant for training samples, χ^2 can be obtained as a function of O_u and O_{uc}. The χ^2 value expresses the normalized deviation of the observation from the expectation:

$$\chi^2(O_u, O_{uc}) = \sum_{i\in\{u,\bar{u}\}, j\in\{c,\bar{c}\}} \frac{(O_{ij} - E_{ij})^2}{E_{ij}}, \tag{81}$$

where

$$E_{ij} = n \cdot \frac{O_i}{n} \cdot \frac{O_j}{n}, \tag{82}$$

which represents the expectation. I simply represent $\chi^2(O_u, O_{uc})$ as $\chi^2(u)$.

5.3.2 Feature Selection Criterion

The basic idea of feature selection is quite natural. First, I decide the threshold τ of the statistical metric, that is, in this case, the threshold of the χ^2 value. In the kernel calculation with the statistical feature selection, 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. Therefore, the sequence kernel and the tree kernel with feature selection (FSSK and FSTK) can be defined as follows:

$$K^{\text{FSSK}}(S^1, S^2) = \sum_{\tau \leq \chi^2(u) | u \in \Sigma^*} \sum_{\mathbf{i}:u=S^1[\mathbf{i}]} \sum_{\mathbf{j}:u=S^2[\mathbf{j}]} \mu^{|u|} \mu^{|u|} \lambda^{\zeta(\mathbf{i})} \lambda^{\zeta(\mathbf{j})}, \tag{83}$$

$$K^{\text{TK}}(T^1, T^2) = \sum_{\tau \leq \chi^2(u) | u \in \Sigma^*} \sum_{\mathbf{i}:u=T^1[\mathbf{i}]} \sum_{\mathbf{j}:u=T^2[\mathbf{j}]} \mu^{|u|} \mu^{|u|} \lambda^{\zeta(\mathbf{i})} \lambda^{\zeta(\mathbf{j})}. \tag{84}$$

The difference from their original kernels, namely Equations (20) and (32), is simply the condition of the first summation, which is $\tau \leq \chi^2(u)$.

Figures 20 and 21,respectively, show simple examples of what FSSK and FSTK calculate as regards the kernel value.

5.3.3 Efficient Feature Selection Algorithm

The basic idea of using a statistical metric to select features is quite natural, but it is not a very attractive approach. Note, however, that although kernels are calculated using the DP technique, it is not clear how to calculate that kernels efficiently with a statistical feature selection. It is computationally infeasible to calculate $\chi^2(u)$ for all possible u with a naive exhaustive method. In the approach, I take advantage of *sub-structure mining* algorithms in order to calculate $\chi^2(u)$ efficiently and to embed statistical feature selection in the original DP-based kernel calculation.

Before explaining these algorithms, I introduce the upper bound of the χ^2 value. The upper bound of the χ^2 value of uv, which is the concatenation of

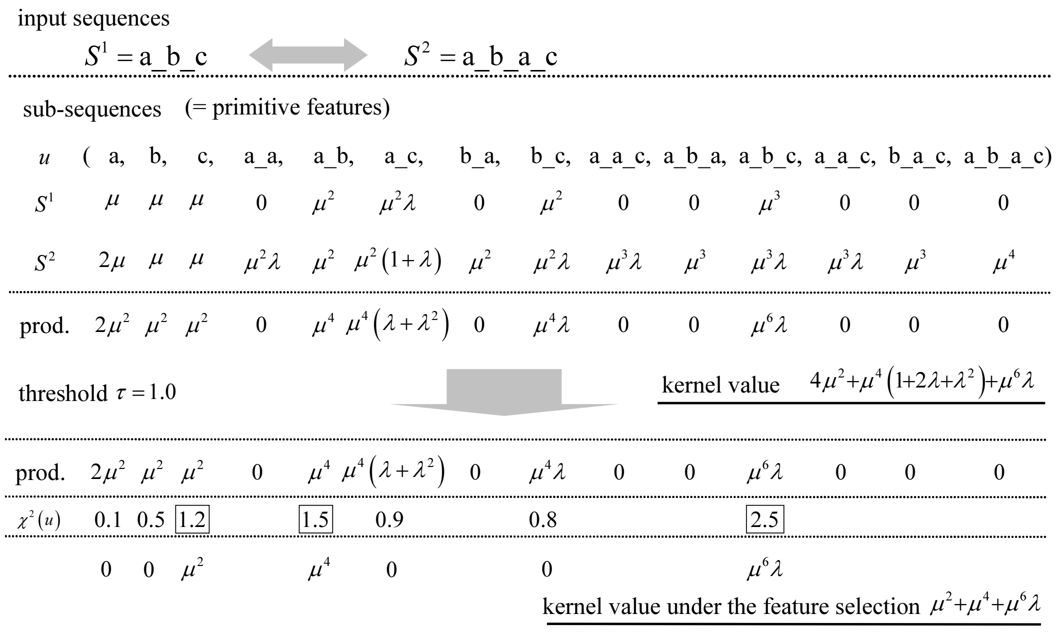

input sequences
S^1 = a_b_c
S^2 = a_b_a_c
sub-sequences (= primitive features)
u (a, b, c, a_a, a_b, a_c, b_a, b_c, a_a_c, a_b_a, a_b_c, a_a_c, b_a_c, a_b_a_c)
S^1 μ μ μ 0 μ^2 $\mu^2\lambda$ 0 μ^2 0 0 μ^3 0 0 0
S^2 2μ μ μ $\mu^2\lambda$ μ^2 $\mu^2(1+\lambda)$ μ^2 $\mu^2\lambda$ $\mu^3\lambda$ μ^3 $\mu^3\lambda$ $\mu^3\lambda$ μ^3 μ^4
prod. $2\mu^2$ μ^2 μ^2 0 μ^4 $\mu^4(\lambda+\lambda^2)$ 0 $\mu^4\lambda$ 0 0 $\mu^6\lambda$ 0 0 0
threshold $\tau = 1.0$
kernel value $4\mu^2+\mu^4(1+2\lambda+\lambda^2)+\mu^6\lambda$
prod. $2\mu^2$ μ^2 μ^2 0 μ^4 $\mu^4(\lambda+\lambda^2)$ 0 $\mu^4\lambda$ 0 0 $\mu^6\lambda$ 0 0 0
$\chi^2(u)$ 0.1 0.5 $\boxed{1.2}$ $\boxed{1.5}$ 0.9 0.8 $\boxed{2.5}$
0 0 μ^2 μ^4 0 0 $\mu^6\lambda$
kernel value under the feature selection $\mu^2+\mu^4+\mu^6\lambda$

Figure 20. Example of statistical feature selection for sequence kernels

sequences u and v, can be defined by the value of u [Morishita00]:

$$\begin{aligned}\chi^2(uv) &\leq \max\left(\chi^2(O_{uc}, O_{uc}), \chi^2(O_u - O_{uc}, 0)\right)\\ &= \widehat{\chi}^2(u).\end{aligned}$$

This inequality indicates that if $\widehat{\chi}^2(u) < \tau$ holds, all sub-sequences uv can be eliminated from the features, since no sub-sequence uv can be $\tau \leq \chi^2(uv)$.

5.3.4 Sequence Kernels with Statistical Feature Mining

Sequence kernels with the proposed feature selection method are defined in the following equations.

$$K^{\text{FSSK}}(S^1, S^2) = \sum_{i=1}^{|S^1|}\sum_{j=1}^{|S^2|} \mathcal{H}_n(S^1_i, S^2_j) \tag{85}$$

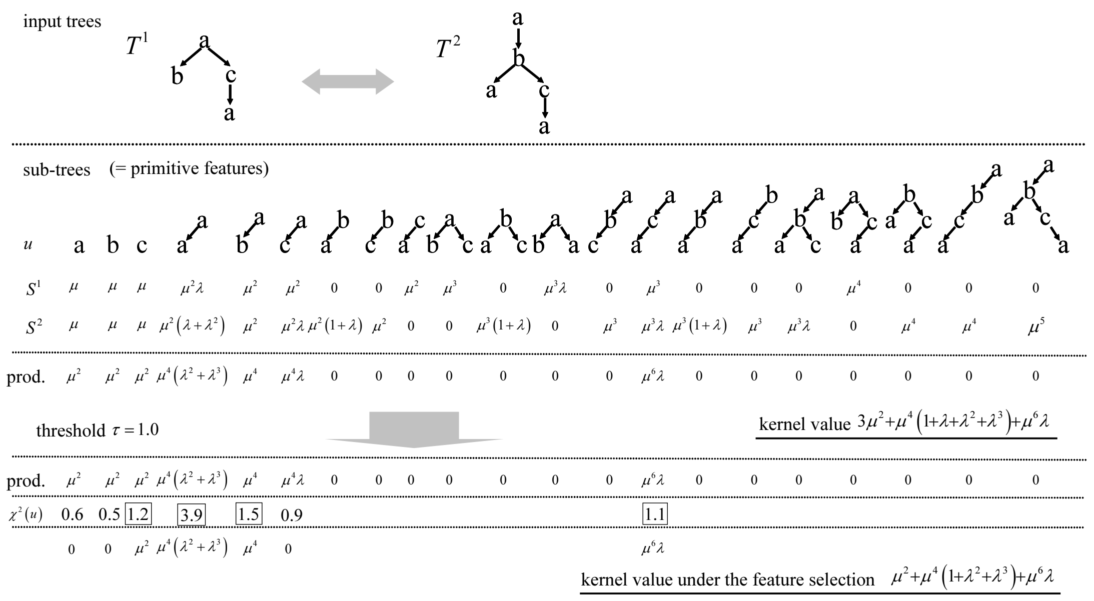

input trees
T^1
T^2
sub-trees (= primitive features)
u
S^1
S^2
prod.
threshold $\tau = 1.0$
kernel value $3\mu^2+\mu^4(1+\lambda+\lambda^2+\lambda^3)+\mu^6\lambda$
$\chi^2(u)$
0.6 0.5 1.2 3.9 1.5 0.9 1.1
kernel value under the feature selection $\mu^2+\mu^4(1+\lambda^2+\lambda^3)+\mu^6\lambda$

Figure 21. Example of statistical feature selection for tree kernels

Let $\mathcal{H}_n(S_i^1, S_j^2)$ be a function that returns the sum value of all statistically significant common sub-sequences u that satisfy $s_i^1 = s_j^2$ and $|u| \leq n$.

$$\mathcal{H}_n(S_i^1, S_j^2) = \sum_{u \in \Gamma_n(S_i^1, S_j^2)} \mathcal{J}_u(S_i^1, S_j^2), \tag{86}$$

where $\Gamma_n(S_i^1, S_j^2)$ represents a set of sub-sequences, where $|u| \leq n$ and that satisfy $\tau \leq \chi^2(u)$. The details of $\Gamma_n(S_i^1, S_j^2)$ are shown in Equation (91).

Then, let $\mathcal{J}_u(S_i^1, S_j^2)$, $\mathcal{J}'_u(S_i^1, S_j^2)$ and $\mathcal{J}''_u(S_i^1, S_j^2)$ be functions that calculate the value of the common sub-sequences between S_i^1 and S_j^2 recursively.

$$\mathcal{J}_{uw}(S_i^1, S_j^2) = \begin{cases} \mathcal{J}'_u(S_i^1, S_j^2) \cdot \mathcal{I}_w(s_i^1, s_j^2) \ \text{ if } \ uw \in \widehat{\Gamma}_n(S_i^1, S_j^2), \\ 0 \quad \text{otherwise}, \end{cases} \tag{87}$$

where

$$\mathcal{I}_w(s_i^1, s_j^2) = \mu \cdot \delta(s_i^1, s_j^2). \tag{88}$$

$\delta(s_i^1, s_j^2)$ is defined as an indicator function that returns 1 iff $s_i^1 = w$ and $s_j^2 = w$, and 0 otherwise. $\widehat{\Gamma}_n(S_i^1, S_j^2)$ is a set of sub-sequences, where $|u| \leq n$, that are candidates for significant sub-sequences. The details of $\widehat{\Gamma}_n(S_i^1, S_j^2)$ are shown in Equation (92).

I introduce a special label Λ to represent an "empty sequence", and define $\Lambda w = w$ and $|\Lambda w| = 1$. Then, $\mathcal{J}'(S_i^1, S_j^2)$ and $\mathcal{J}''(S_i^1, S_j^2)$ are introduced to calculate the common *gapped* sub-sequences between S_i^1 and S_j^2.

$$\mathcal{J}'_u(S_i^1, S_j^2) = \begin{cases} 1 \quad \text{if} \quad u = \Lambda, \\ 0 \quad \text{if} \quad j = 0 \quad \text{and} \quad u \neq \Lambda, \\ \lambda \mathcal{J}'_u(S_i^1, S_{j-1}^2) + \mathcal{J}''_u(S_i^1, S_{j-1}^2) \quad \text{otherwise}, \end{cases} \tag{89}$$

$$\mathcal{J}''_u(S_i^1, S_j^2) = \begin{cases} 0 \quad \text{if} \quad i = 0, \\ \lambda \mathcal{J}''_u(S_{i-1}^1, S_j^2) + \mathcal{J}_u(S_{i-1}^1, S_j^2) \quad \text{otherwise}. \end{cases} \tag{90}$$

Equations (87), (89) and (90) are almost the same as Equations (28), (29) and (30), respectively. The only difference is the selection of the features by $\widehat{\Gamma}_n(S_i^1, S_j^2)$ in the condition of Equation (87).

The following equations are introduced to select a set of significant sub-sequences. $\Gamma_n(S_i^1, S_j^2)$ and $\widehat{\Gamma}_n(S_i^1, S_j^2)$, which are described above, are defined as follows:

$$\Gamma_n(S_i^1, S_j^2) = \{u \mid u \in \widehat{\Gamma}_n(S_i^1, S_j^2), \tau \leq \chi^2(u)\} \tag{91}$$

$$\widehat{\Gamma}_n(S_i^1, S_j^2) = \begin{cases} \Psi_n(\widehat{\Gamma}'_n(S_i^1, S_j^2), s_i^1) \cup \{s_i^1\} \quad \text{if} \quad s_i^1 = s_j^2, \\ \emptyset \quad \text{otherwise}, \end{cases} \tag{92}$$

where $\Psi_n(F, w) = \{uw \mid u \in F, \tau \leq \widehat{\chi}^2(uw), |uw| \leq n\}$, and F represents a set of sub-sequences. Note that $\Gamma_n(S_i^1, S_j^2)$ and $\widehat{\Gamma}_n(S_i^1, S_j^2)$ have only sub-sequences u that satisfy $\tau \leq \chi^2(uw)$ and $\tau \leq \widehat{\chi}^2(uw)$, respectively, iff $s_i^1 = s_j^2$ and $|uw| \leq n$; otherwise they become empty sets.

The following two equations are introduced to calculate $\Gamma_n(S_i^1, S_j^2)$ and $\widehat{\Gamma}_n(S_i^1, S_j^2)$ recursively.

$$\widehat{\Gamma}'_n(S_i^1, S_j^2) = \begin{cases} \emptyset \quad \text{if } j = 0, \\ \widehat{\Gamma}'_n(S_i^1, S_{j-1}^2) \cup \widehat{\Gamma}''_n(S_i^1, S_{j-1}^2) \quad \text{otherwise}, \end{cases} \tag{93}$$

$$\widehat{\Gamma}''_n(S^1_i, S^2_j) = \begin{cases} \emptyset \quad \text{if} \quad i = 0\ , \\ \widehat{\Gamma}''_n(S^1_{i-1}, S^2_j) \cup \widehat{\Gamma}_n(S^1_{i-1}, S^2_j) \quad \text{otherwise.} \end{cases} \tag{94}$$

5.3.5 Implementation

In the implementation, I take advantage of *sub-structure mining* algorithms, specifically a sequential pattern mining technique, PrefixSpan [Pei01], and a statistical metric pruning (SMP) method, Apriori SMP [Morishita00], in order to calculate $\chi^2(u)$ efficiently. By using PrefixSpan with the SMP algorithm, $\chi^2(uw)$ and $\widehat{\chi}^2(uw)$, where uw represents the concatenation of a sequence u and a label w, I can perform a calculation by using a set of pointers of u against data and the number of appearances of w in the suffix of the pointers. I note that the set of pointers of uw can be simply obtained from a previous search of u since uw is simply the concatenation of u and w. In other words, uw will never appear if u does not appear. Counting the number of w appearing in the suffix of u is equivalent to counting the number of uw.

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

With condition 1, sub-sequence uw is selected as a significant feature and stored in $\Gamma_n(S^1_i, S^2_j)$ and $\widehat{\Gamma}_n(S^1_i, S^2_j)$. With condition 2, uw is pruned, that is, all uwv are pruned from the search space of PrefixSpan. With condition 3, uw is not a significant feature, however, uwv can be significant, thus uw is stored in $\widehat{\Gamma}_n(S^1_i, S^2_j)$ and the search is continued to uw. Figure 22 shows an example of searching and pruning the sub-sequences to select significant features by using PrefixSpan with the SMP algorithm, and the internal results represented by a TRIE data structure.

There are certain techniques that make it possible to calculate kernels faster in the implementation. For example, since $\chi^2(u)$ and $\hat{\chi}^2(u)$ are constant against the same data, I only have to calculate them once. I store the internal search results of PrefixSpan with the SMP algorithm in a TRIE structure. After that,

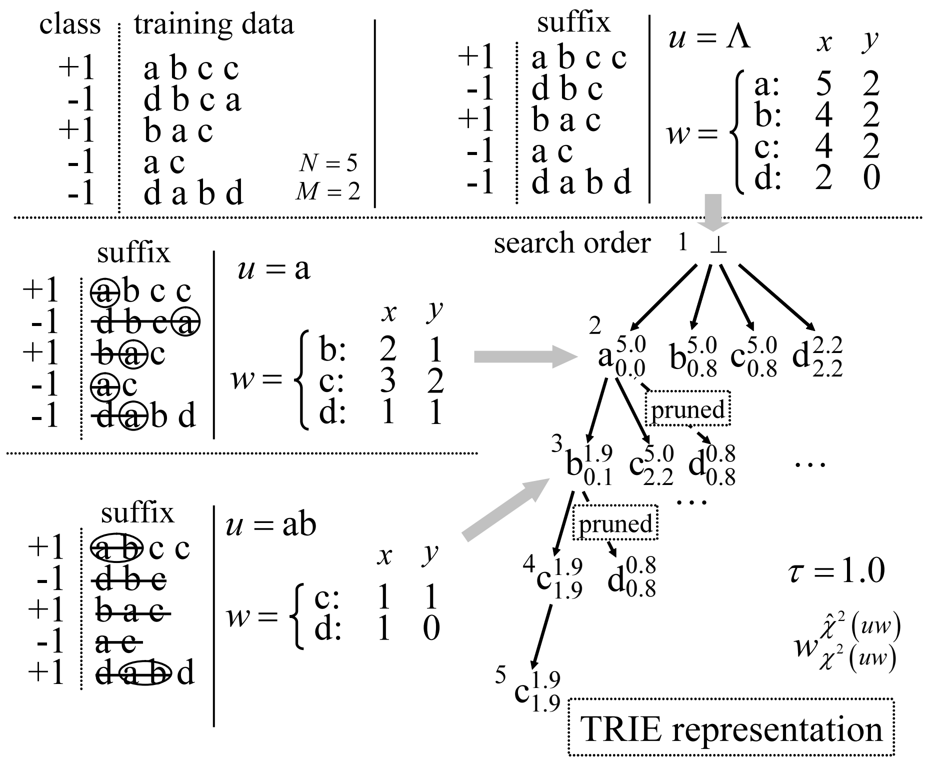


Figure 22. Example of searching and pruning sub-sequences by PrefixSpan with SMP algorithm

I use that results in TRIE instead of calculating $\chi^2(u)$ again when the kernel finds the same sub-sequence. Moreover, I introduce a *transposed index* for the fast evaluation of $\chi^2(u)$ and $\hat{\chi}^2(u)$. By using this approach, I only have to consult that index of w to evaluate whether or not any uw are significant features.

Equations (86) to (90) can be calculated in the same way as the original DP based kernel calculation. The recursive set operations of Equations (92) to (94) can be executed in the same way as Equations (87) to (90). Moreover, calculating $\chi^2(u)$ and $\hat{\chi}^2(u)$ with sub-structure mining algorithms allows us to calculate the same order of the DP based kernel calculation. As a result, statistical feature selection can be embedded in the original sequence kernel calculation based on the DP.

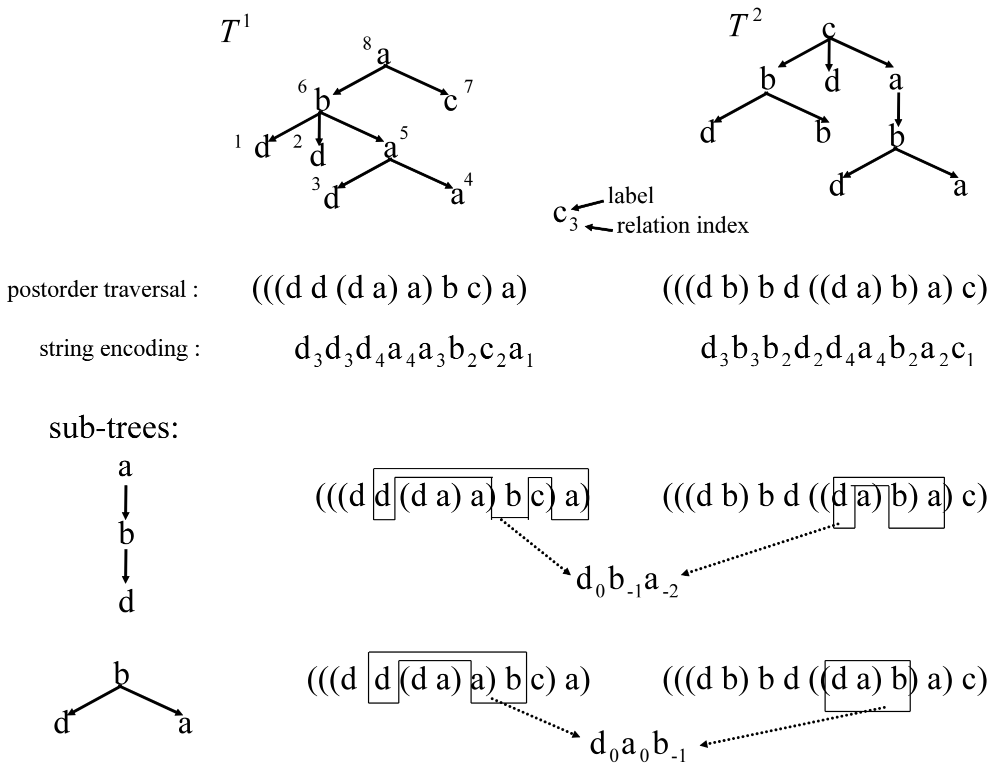


Figure 23. Example of string encoding for trees under the postorder traversal

5.3.6 Tree Kernels with Statistical Feature Mining

The famous tree mining algorithm [Zaki02] cannot be applied as a feature selection method for tree kernels, because the tree mining algorithm executes a preorder search of trees while tree kernels calculate the kernel value in postorder. Thus, I take advantage of the string (sequence) encoding method for trees. In particular, I enumerate the nodes (labels) under the postorder traversal. Figure 23 shows an example of string encoding for trees under the postorder traversal. Each label is converted with a certain number, which we call a 'relation index', written in subscript of each label as shown in Figure 23. These numbers indicate the depths of the node. However, if we consider sub-tree u, the number is calculated from the leftmost node in u. Then, I deal with each label with a relation index as a new label. That is, $t_i^1 = t_j^2$ means that both the label and relation index of

t_i^1 and t_j^2 are the same. For example, $\mathrm{d}_0\mathrm{b}_0\mathrm{a}_{-1}$ and $\mathrm{d}_0\mathrm{b}_{-1}\mathrm{a}_{-2}$ are different.

After converting the tree to the postorder node sequence, the small extension of the FSSK, is used for the FSTK. Then, the proposed method for tree kernels is defined in the following equations.

$$K^{\text{FSTK}}(T^1, T^2) = \sum_{i=1}^{|T^1|} \sum_{j=1}^{|T^2|} \mathcal{H}_m(T_i^1, T_j^2) \tag{95}$$

i and j represent indexes of nodes in the postorder transversal of T^1 and T^2, respectively.

After that, I only show the extension parts of equations in the FSSK, that is, $\Gamma_n(T_i^1, T_j^2)$ and $\Psi_n(F, w)$:

$$\Gamma_n(T_i^1, T_j^2) = \{u \mid u \in \widehat{\Gamma}_n(T_i^1, T_j^2), \tau \leq \chi^2(u), \iota(u_{|u|}) < \min_{i=1\dots|u-1|} \iota(u_i)\}, \tag{96}$$

$$\Psi_n(F, w) = \{uw \mid u \in F, \tau \leq \widehat{\chi}^2(uw), \eta(uw) \leq n, \iota(u_{|u|}) \leq \iota(w) + 1\}, \tag{97}$$

where $\iota(w)$ returns the relation index of w, and $\eta(u)$ returns the depth of u, which is written as $\eta(u) = \max_{i=1\dots|u|} \iota(u_i) - \min_{j=1\dots|u|} \iota(u_j)$. For example, if $u = \mathrm{d}_0\mathrm{b}_{-1}\mathrm{a}_{-2}$, then $u_2 = \mathrm{b}_{-1}$, $\iota(u_2) = -1$ and $\eta(u) = 2$. $\iota(u_{|u|}) < \min_{i=1\dots|u-1|} \iota(u_i)$ is realized to check if u is a complete sub-tree. That is, $\mathrm{d}_0\mathrm{b}_{-1}\mathrm{a}_0$ is not a subtree, because $\mathrm{d}_0\mathrm{b}_{-1}$ and a_0 are not connected. Since the tree kernel does not have parameter λ, I calculate $\mathcal{J}'_u(S_i^1, S_j^2)$ and $\mathcal{J}''_u(S_i^1, S_j^2)$ as $\lambda = 1$.

I note that if we set $\tau = 0$, which means that all features are dealt with by kernel calculation, we can obtain exactly the same kernel value as the original tree kernel.

5.3.7 Properties

The proposed method has several important advantages over conventional methods.

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

Second, according to Equations (85) to (93), 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.

The kernel calculation of the proposed method requires a longer training time due to the feature selection. However, the selected sub-structures are represented as a TRIE data structure. This means the fast calculation technique proposed in [Kudo03] can be applied to the proposed method, which yields the classification faster. For applications, classification speed is much more important than training speed. This is the third advantage of the proposed method.

5.4 Experiments

The goal of the experiments described in this chapter is to clarify the effect of statistical feature selection. To accomplish this goal, I compare the performance of the proposed feature selection method and a conventional approach.

This comparison is performed using the following actual NLP tasks;

- *modality identification (MOD)* (Japanese)
- *subjectivity classification (SUB)* (English)

These tasks are defined as a *sentiment classification* task, which requires richer structural information for improved performance (See Chapter 2).

5.4.1 Data Set

The following describes the data used in the experiments. Table 22 summarizes the data set.

Table 22. Data set for the experiments

	Modality	Subjectivity
# of samples	2095	10000
# of classes	4	2
# of BOW	5995	23435

Modality Identification

The data set was created from articles taken from Mainichi newspaper and one of three modality tags, ‘opinion’, ‘assertion’, ‘description’ and ‘other’ were applied to each sentence. These modality tags follow [Tamura96]. The data size was 2,095 sentences consisting of 195 sentences of ‘opinion’, 627 of ‘assertion’, 1,237 of ‘description’, and 36 of ‘other’.

Subjectivity

This data set was automatically mined from movie-reviews on the Web [Pang04][7] . It contains 5,000 subjective and 5,000 objective sentences about movies. That is, there are two target classes: ‘subjective’ and ‘objective’.

5.4.2 Comparison Methods

Mainly, I compared the performance of sequence kernels (SK) and sequence kernels with the proposed feature selection (FSSK). In the same way, I compared the performance of tree kernels (TK) and tree kernels with the proposed feature selection (FSTK). Moreover, I also evaluated *bag-of-words* (BOW) kernel (BOW-K)[Joachims98] as a baseline method.

There are certain tunable parameters for each method. Parameter n represents the threshold associated with the size of the sub-structure. This means each comparison method only deals with the sub-structures whose size is n or less for

[7] This data is available at http://www.cs.cornell.edu/people/pabo/movie-review-data/.

the features. In the case of BOW-K, n indicates the degree of the polynomial kernels. In the case of SK, TK, FSSK and FSTK, n represents the threshold of the number of labels in the primitive sub-structures. I used $n = \{1, 2, 3, 4\}$ for BOW-K, and $n = \{2, 3, 4, \infty\}$ for SK, TK, FSSK and FSTK, while $n = 1$ of SK and TK is exactly the same as BOW-K. Note that $n = \infty$ means all possible sub-sequences are used.

The decay factor for the sub-structure size μ (See Chapter 2.2.1) is selected from $\mu = \{0.1, 0.3, 0.5, 0.7, 0.9\}$ for the conventional SK and TK. Note that $\mu = 1.0$ indicates that there is no feature weighting.

Then, in the case of the proposed method, there is a threshold of feature selection, τ, as I previously explained in Section 5.3. I used $\tau = \{1.64, 2.71, 3.84, 6.64\}$ for each experiment. These numbers come from the χ^2 significance test, namely 0.1, 0.05, 0.01, 0.005 % levels of significance in the χ^2 distribution with one degree of freedom, respectively.

5.4.3 Performance Evaluation Methods

Basically, I evaluated the overall performance using the average accuracy of a *ten fold cross validation.* I used Support Vector Machine (SVM) [Vapnik95] as a kernel-based machine learning algorithm to learn a decision function by using nine sets. I set a soft margin parameter for SVM, $C = 1000$ in Equation (7). While the modality identification is a multi-class classification task and SVM is a binary classifier, I employed the 'one vs. rest model' to determine a final estimated label. Finally I calculated "label accuracy", which is the accuracy of estimated labels compared with the correct labels of each given text.

I iterated the above procedure ten times while changing the set of test data. Finally, I evaluated the performance by using the average label accuracy of the evaluation. Additionally, I tested the difference in average label accuracy by using a paired Wilcoxon signed rank test, which is a non-parametric statistical significance test.

Table 23. The best MOD results with each comparison method

	Performances		Parameters			Statistical test
	acc.	S.D.	n	μ	α	$p < \{0.05, 0.01, 0.005\}$
FSTK	0.856	0.0228	(∞)	(1.0)	0.05	
TK	0.844	0.0200	(∞)	0.5	-	*
	0.850	0.0200	2	(1.0)	-	*
FSSK	0.865	0.0203	(∞)	(1.0)	0.05	
SK	0.847	0.0108	(∞)	0.7	-	**
	0.858	0.0118	3	(1.0)	-	*
BOW-K	0.730	0.0304	1			

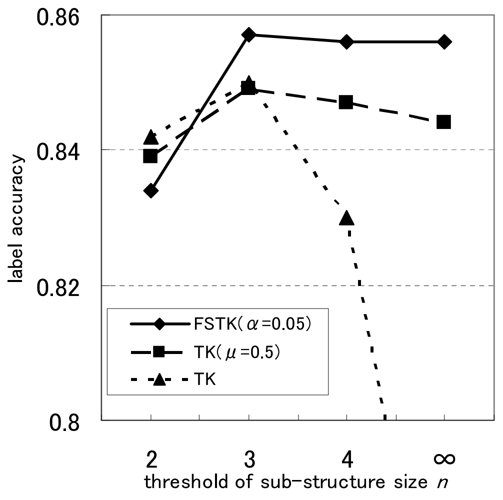
Table 24. The best SUB results with each comparison method

	Performances		Parameters			Statistical test
	acc.	S.D.	n	μ	α	$p < \{0.05, 0.01, 0.005\}$
FSTK	0.918	0.0142	(∞)	(1.0)	0.05	
TK	0.913	0.0123	(∞)	0.5	-	-
	0.904	0.0131	2	(1.0)	-	*
FSSK	0.912	0.0081	(∞)	(1.0)	0.05	
SK	0.917	0.0082	(∞)	0.7	-	-
	0.917	0.0068	2	(1.0)	-	-
BOW-K	0.902	0.1005	2			

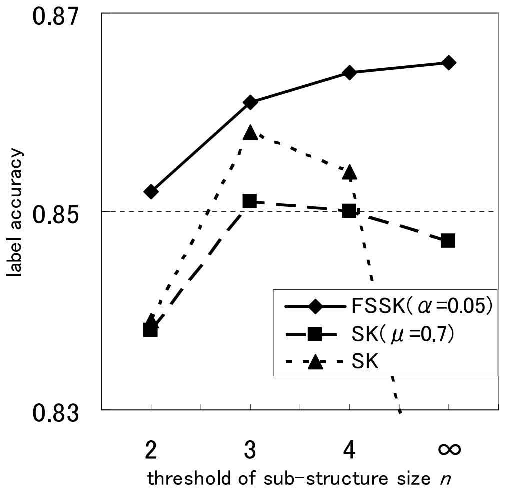
5.4.4 Results and Discussion

Tables 23 and 24, respectively, show the best modality identification (MOD) and subjectivity identification (SUB) results obtained with each comparison method, respectively. Note that n in each table indicates the threshold of the sub-structure size.

Moreover, Figures 24 and 25 show the performance against sub-structure size.

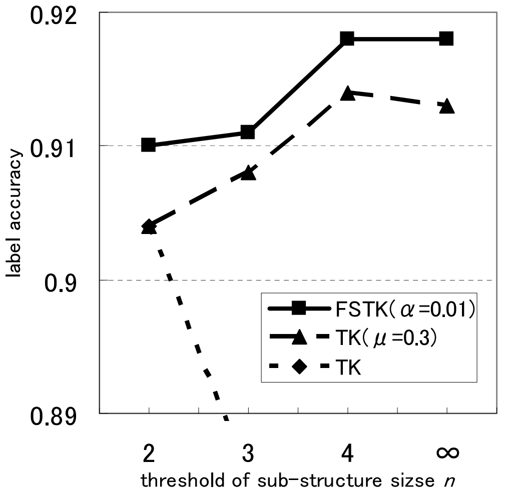


(a) tree kernels

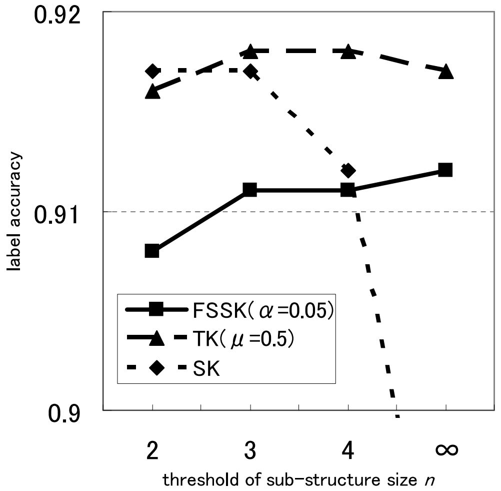


(b) sequence kernels

Figure 24. Effect of structure size with each method (MOD)



(a) tree kernels



(b) sequence kernels

Figure 25. Effect of structure size with each method (SUB)

Effect of Structural Information

In general, according to the results of BOW-K and SK or TK, we can say that the use of structural features can improve the performance of sentiment classification tasks that require the details of the contents to perform well. In fact, under the Wilcoxon signed rank test, there are significant differences ($p < 0.01$) between the best performance of BOW-K and that of SK, and also between that of BOW-K and TK in all the experiments. This indicates that the structural information was very efficient as regards performing these tasks. Again, this fact is strong evidence that structural information substantially improves the performance of sentiment classifications.

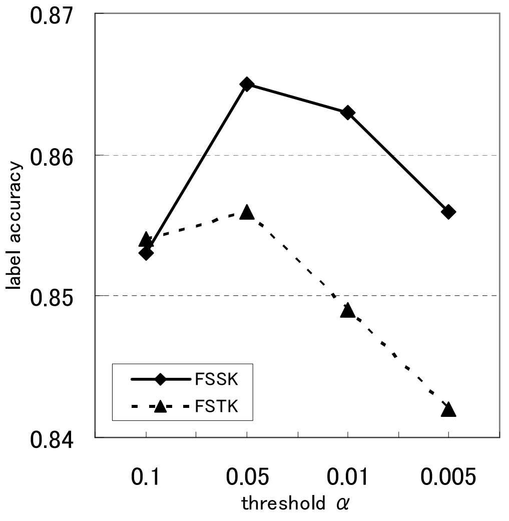
Over-fitting Problem for SK and TK

All the results showed that SK and TK with no feature selection achieve their maximum performance when $n = 2$ or 3. The performance deteriorates considerably once n exceeds 4. Of course, there are significant differences ($p < 0.01$) between the best performance of SK and that of SK with $n = \infty$, as well as TK. This implies that SK and TK with larger sub-structures degrade classification performance. These results show the same tendency as the previous studies discussed in Section 5.2. These experimental results showed additional evidence of over-fitting in learning with standard SK and TK.

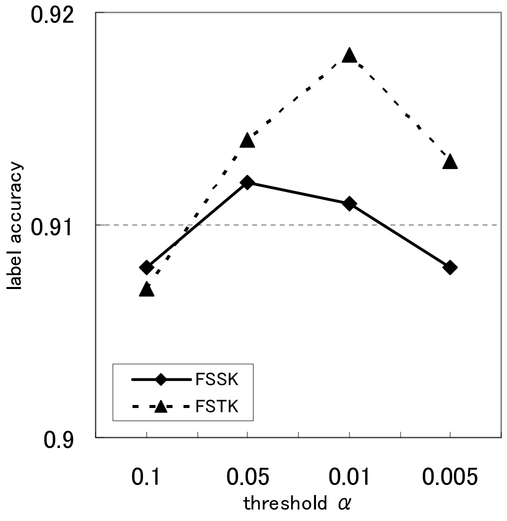
Effect of Statistical Feature Selection

As shown in the experimental results, FSSK and FSTK 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 very large sub-structures can be extremely effective. In the case of the modality identification, a statistical difference ($p < 0.01$) between $n = 2$ and $n = \infty$ was detected for both FSSK and FSTK.

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



(a) tree kernels



(b) sequence kernels

Figure 26. Effect of threshold α

features do exist. Thus, I can improve the performance of some classification problems by dealing with larger substructures. Even if optimum performance was not achieved with $n = \infty$, the difference between the performance levels of smaller n values is very small compared with that of SK and TK. This indicates that the proposed method is very robust as regards sub-structure size; It therefore becomes unnecessary for us to decide the sub-structure size carefully. This indicates that the proposed approach, using large sub-structures, is better than the conventional approach of eliminating sub-structures based on size.

Moreover, these experimental results showed that there are some informative larger-sized sub-substructures. This fact indicates that a method that can handle larger-sized sub-substructures without over-fitting is a better approach for text processing.

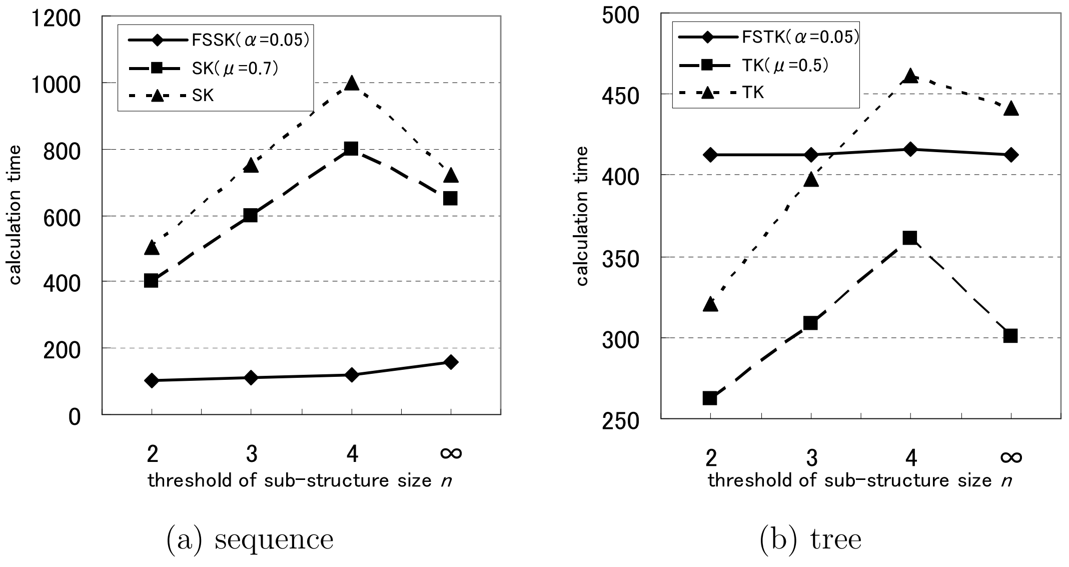


Figure 27. Calculation speed of training in MOD

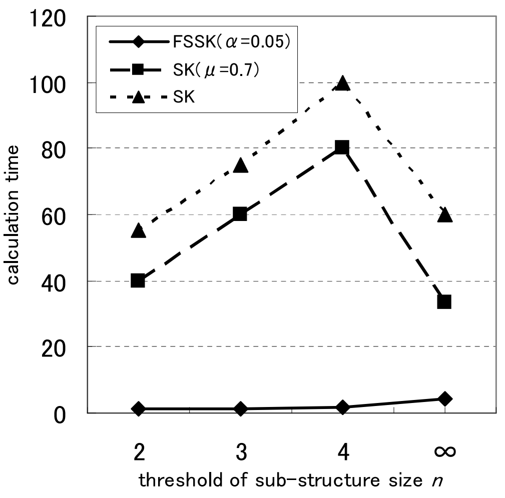
Robustness of statistical metric threshold

Figure 26 shows the performance against the threshold α of a statistical metric. α represents the level of significance. As shown in the figure, the performance curve is convex. This seems to have an optimal threshold for statistical feature selection. In fact, there is a trade-off between performance and the number of features. A small number of features cannot make a good model for a target task, however, a large number of features induces an over-fitting problem.

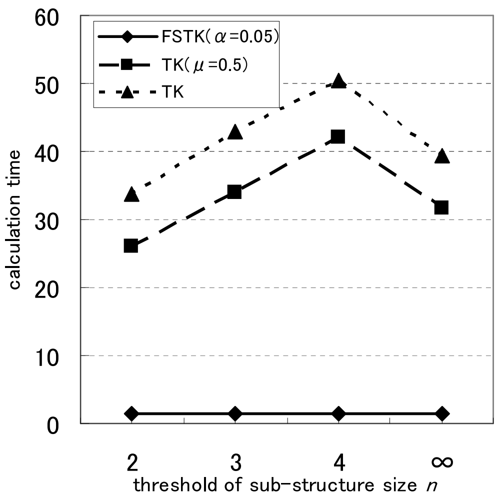
Experimentally, in the case of χ^2 statistical metric, $\alpha = 0.05$ or 0.01, which are usually treated as a threshold for the statistical test, seem to appropriate for the proposed method.

Calculation Speed

I compared the calculation speeds of the proposed method and conventional methods with MOD data. I ran these methods on a "Linux PC with Opteron 2.4GHz". Figures 27 and 28 show the results for the training and test, respectively.



(a) sequence



(b) tree

Figure 28. Calculation speed of test in the MOD

As shown in these tables, the increases in calculation time against n in SK and TK have a linear relation, which is explained in Chapter 2. By comparison the proposed method seems to have no correlation with size n. The main reason for this observation arises from the pruning architecture with the proposed method. Since the feature space is very sparse, we can usually prune most of features (sub-structures) in the kernel calculation. As a result, even if the worst time complexity of the proposed method becomes exponential against the size of an input object, we can efficiently calculate kernels with the proposed feature selection.

Interestingly, the test calculation time was from 20 to 40 times faster than with the conventional methods. There is a technique of efficient representation a model constructed by SVM. While the proposed kernel calculation uses a TRIE structure, I made an SVM model by this TRIE representation of features. This model representation allows us to evaluate test data very quick.

Table 25. The number of features extracted by the proposed method

(a) MOD

	# of features			
	2	3	4	∞
FSTK	3185	4027	4474	4966
FSSK	3090	3457	3618	3767

(b) SUB

	# of features			
	2	3	4	∞
FSTK	33110	33879	33954	33959
FSSK	27854	29901	30306	30408

Table 26. Examples of typical features in modality classification

Class	Weight	Feature (sub-structure)	size
Opinion	0.0846282	で-は-ない-か-。	5
	0.542663	思え-ない-。	3
Assertion	4.15498	で-は-ない-。	4
	3.41099	で-ある-。	3
	2.0444	だっ-た-。	3
Description	0.160708	なっ-た-。	3
	0.705698	に-ある-。	3
	0.156255	で-は-ない	3

Feature Analysis

Now, I focus on the features that are extracted by the proposed method. First, Table 25 shows the numbers of features which are extracted. This table provides evidence of the calculation speed as discussed in the previous section. Increasing n does not greatly affect the increase in the number of features.

Table 27. Examples of typical features in SUB

Class	Weight	Feature (sub-structure)	size
subjective	0.522993	this is	2
	0.00407342	this is the kind of	5
	0.00407342	the story is	3
objective	0.977668	this is the story of	5
	0.977668	is the movie of	4

Tables 27 and 26 show typical examples of large-sized features (sub-structures) extracted by the proposed method. In the tables, ‘weight’ represents the weight in a model that is constructed by SVM training.

For example, in the modality identification, the expression “**ではないか**” is related to ‘opinion’, e.g. “**〜 考えるべきではないか**”, while “**ではない。**” is related to ‘assertion’, e.g. “**〜 に向けられた言葉ではない**”. Moreover, if certain word follows “**ではない**”, for example “**ではない** (**と**)“, then it is a feature of ‘description, e.g. “**〜 便宜供与があったのではないかということだ**”. This observation proves that the similar expressions can characterize different classes in the modality identification. In the proposed method, base words such as “**ない**” and “**ではない**” which have no information for classifying text, are eliminated by the feature selection.

With the subjectivity identification, for example, “This is” was assigned as subjective expression, because people present some opinion following this expression, for example, “this is one of the biggest disappointments of the year.” However, “This is the story of ...” seems to be a typical expression for introducing an objective sentence while the data set is about reviews of movies, such as “this is the story of their lives.” Moreover, the words “story” and ‘movie’ appear frequently in this data set. In fact, the words “story” and ‘movie’ have no effect in terms of subjective or objective classification. However, “the story/movie is” is strongly related subjective, while “is the story/movie of” objective. This result indicates that certain expressions (large sub-structures) are important features for this family of text classification tasks.

Table 28. Examples of topside large sub-structures in MOD

Class	Weight	Feature (sub-structure)	size
Opinion	0.366795	を-感じる-。	3
	0.209057	て-ほしい	2
	1.17468	か-。	2
	0.0846282	で-は-ない-か-。	5
	0.318182	、-私-たち-は	4
	0.155752	は-避け-て-通れ-ない	5
	0.0895081	と-言っ-て-いる-の-だろ-う	7
Assertion	4.15498	で-は-ない-。	4
	3.41099	で-ある-。	3
	2.0444	だっ-た-。	3
	1.81021	だ-。	2
	1.33073	で-あっ-た-。	4
	0.117897	み-て-は-なら-ない-。	6
	0.0471337	に-する-必要-が-あっ-た-。	7
Description	1.28043	が-ある-。	3
	0.705698	に-ある-。	3
	0.561427	し-た-。	3
	0.358281	て-き-た	3
	0.325913	て-いる-。	3
	0.160708	なっ-た-。	3
	0.306159	こと-に-なる-。	4

Additionally, Tables 29 and 28 show examples of the top ranking of large sub-structures in a model.

5.5 Summary

This chapter proposed a statistical feature mining method for sequence and tree kernels.

Table 29. Examples of typical features in SUB

Class	Weight	Feature (sub-structure)	size
Subjective	0.55796	and amazing	2
	0.546071	in the right place	4
	0.548833	under the skin of	4
	0.522993	this is	2
	0.440011	ought to be	3
	0.417562	the film is	3
	0.404369	in the right place .	4
	0.400738	manages to be	3
	0.333458	one of the great	4
Objective	1.46837	the story of	3
	0.849323	a story about	4
	0.600706	has a way of	4
	0.53318	the making of	3
	0.458776	following the	2
	0.386632	is going to	3
	0.360019	about love	2
	0.336961	cast of characters	3
	0.321236	a series of	3
	0.295688	is a documentary about	4
	0.268486	in an effort to	4
	0.258472	the first time .	4
	0.249223	relationship with the	3

The proposed method can select statistically significant features automatically based on a statistical significance test. The proposed method can be embedded in the original DP based kernel calculation process by using sub-structure mining algorithms.

The experiments demonstrated that the proposed method is superior to conventional methods, which generally eliminate or down-weight larger sized sub-structures. Moreover, the experimental results indicate that larger sized sub-structures can provide substantial information for improving the performance of sentimental classification. Therefore, the proposed method, which can handle larger sized sub-structure without creating over-fitting problems, yields benefits in terms of concept and performance.

6. Conclusion

6.1 Summary

The goals of this dissertation were;

1. To **propose efficient methodologies that are capable of handling rich information** derived from syntactic and semantic analysis

2. to **confirm the effectiveness of the rich information** provide by the proposed method.

To achieve the outlined goals, this dissertation discussed the following methodologies:

1. Effectively handling different levels of word attribute, such as words, semantic information obtained from dictionaries and part-of-speech (Chapter 3)

2. Effectively handling richer structural information derived from integrated syntactic and semantic analysis (Chapter 4)

3. The effect of statistical feature mining for structural features (Chapter 5).

These proposed methods were evaluated by using certain *sentiment classification tasks*, that demand a text to be interpreted semantically or contextually.

In chapter 3, I described a feature extraction method, called *word attribute N-gram*, which is capable of dealing with a combination of several levels of word attributes. Additionally, I showed that this method can be regarded as extended sequence kernels:namely sequence kernels with a framework that allows the existence of duplicated labels in one node. The experimental results showed that the proposed method significantly improved the performance of question classification tasks. This result showed that the feature sets constructed by word attribute N-grams are effectively suited to text processing tasks. Moreover, this chapter verified most of the relative features to given tasks by using one of the feature selection methods, SVM RFE.

Chapter 4 proposed HS-graph kernel, that can handle integrated structural information derived from syntactic and semantic analysis. The text processing tasks such as sentiment classification are considered to be tasks that require deep linguistic information within text to obtain good results. Therefore, the proposed method presents one way of handling this information. I evaluated the performance of HS-graph kernels using English and Japanese question classification tasks. The experiments showed that HS-graph kernels offered better performance than sequence, tree and graph kernels, and the baseline method, bag-of-words kernels, which has no framework for handling all the linguistic information inherent in texts. Therefore, this result revealed that the richer structural information that reflects syntactic and semantic information within text can provide substantial improvements as regards text processing. Additionally, the experiments showed that the proposed method is a language independent method: both English and Japanese tasks were improved significantly in the same framework.

Chapter 5 focused on one critical issue of convolution kernels and proposed a statistical feature selection method for sequence kernels and tree kernels. This provided a method that selects statistically significant sub-structures automatically based on a statistical significance test. Moreover, the proposed method was embedded into the original DP-based kernel calculation by using sub-structure mining algorithms, and I was able to compute the proposed method efficiently. Experiments were undertaken on some sentimental classification tasks to confirm the problem with conventional methods and to evaluate the effect of the proposed method. The experimental results demonstrated that the proposed method is superior to conventional methods. Moreover, the results indicated that some of the larger-sized sub-structures can provide effective information for text processing tasks. The proposed method can use these sub-structures without creating overfitting problems and this yields benefits in terms of concept and performance.

The obtained experimental results showed that this dissertation verified that handling richer information derived from syntactic and semantic analysis can provide excellent improvements as regards text processing tasks; that is, the goal of this dissertation. Moreover, this dissertation also made it clear that dealing with only significant sub-structures can further improve the performance of text processing tasks.

6.2 Future Directions

In this dissertation, I focused solely on sentiment classification tasks. However, the methods that I proposed are generalized methods, because they are all defined as *kernel functions*, which is a very generalized mathematical framework. As a result, they can be applied to many tasks, other than classification, for example clustering, regression and ranking. Question answering, text summarization and machine translation are possible tasks to which the proposed methods can be applied. This is because these tasks are generally broken down into small tasks that can be regarded as simple classification, regression, ranking or clustering tasks.

Moreover, the proposed methods can apply not only to NLP tasks, but also to tasks in any number of field, if they consider the structures of input objects. Bioinformatics is one good example of a science that involves handling structured objects, such as chemical compounds and proteins. In the future, I plan to explore opportunities to apply this method to the proposed method to other research fields.

References

[Asahara00] Asahara, A. and Matsumoto, Y.: Extended Models and Tools for High-performance Part-of-speech Tagger, *Proceedings of the 18th International Conference on Computational Linguistics (COLING2000)*, pp. 21–27 (2000).

[Blanz96] Blanz, V., Schölkopf, B., Bulthoff, H. H., Burges, C., Vapnik, V. and Vetter, T.: Comparison of View-Based Object Recognition Algorithms Using Realistic 3D Models, *ICANN*, pp. 251–256 (1996).

[Buchsbaum00] Buchsbaum, A. and Westbrook, J.: Maintaining hierarchical graph views, *Proc. 11th ACM-SIAM Symposium on Discrete Algorithms (SODA'00)*, pp. 566–575 (2000).

[Cancedda03] Cancedda, N., Gaussier, E., Goutte, C. and Renders, J.-M.: Word-Sequence Kernels, *Journal of Machine Learning Research*, Vol. 3, pp. 1059–1082 (2003).

[Collins01] Collins, M. and Duffy, N.: Convolution Kernels for Natural Language, *Proc. of Neural Information Processing Systems (NIPS'2001)* (2001).

[Collins03] Collins, M.: Head-Driven Statistical Models for Natural Language Parsing, *Computational Linguistics*, Vol. 29, No. 4, pp. 589 – 637 (2003).

[Cortes95] Cortes, C. and Vapnik, V. N.: Support Vector Networks, *Machine Learning*, Vol. 20, pp. 273–297 (1995).

[Cristianini00] Cristianini, N. and Shawe-Taylor, J.: *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*, Cambridge University Press (2000).

[Eades96] Eades, P. and Feng, Q.-W.: Multilevel Visualization of Clustered Graphs, *Proc. Graph Drawing, GD*, No. 1190, pp. 101–112 (1996).

[Fellbaum98] Fellbaum, C.: *WordNet: An Electronic Lexical Database*, MIT Press (1998).

[Fukumoto03] Fukumoto, J., Kato, T. and Masui, F.: Question Answering Challenge (QAC1): An Evaluation of QA Tasks at the NTCIR Workshop 3, *Proc. of AAAI Spring Symposium: New Directions in Question Answering*, pp. 122–133 (2003), http:// www.nlp.is.ritsumei.ac.jp/˜qac/index-j.html.

[Fukushima01] Fukushima, T. and Okumura, M.: Text Summarization Challenge: Text Summarization Evaluation in Japan, *Proc. of the NAACL2001 Workshop on Automatic summarization*, pp. 51–59 (2001).

[Gärtner03] Gärtner, T., Flach, P. and Wroble, S.: On Graph Kernels: Hardness Results and Efficient Alternatives, *Proc. 16th Annual Conference on Learning Theory (COLT 2003)*, pp. 129–143 (2003).

[Guyon02] Guyon, I., Westion, J., Barnhill, S. and Vapnik, V. N.: Gene Selection for Cancer Classification using Support Vector Networks, *Machine Learning*, Vol. 46, pp. 389–422 (2002).

[Harel88] Harel, D.: On visual formalisms, *Communications of the ACM*, Vol. 31, No. 5, pp. 514 – 530 (1988).

[Harman04] Harman, D. and Over, P.: The Effects of Human Variation in DUC Summarization Evaluation, *Proc. of the ACL2004 Workshop on Text Summarization Braches Out*, pp. 10–17 (2004).

[Haussler99] Haussler, D.: Convolution Kernels on Discrete Structures, *Technical Report UCS-CRL-99-10*, UC Santa Cruz (1999).

[Herbrich02] Herbrich, R.: *Learning Kernel Classifiers*, MIT Press (2002).

[Ikehara97] Ikehara, S., Miyazaki, M., Shirai, S., Yokoo, A., Nakaiwa, H., Ogura, K., Oyama, Y. and Hayashi, Y. (eds.): *The Semantic Attribute System,* Goi-Taikei — A Japanese Lexicon, Vol. 1, Iwanami Publishing (1997), (in Japanese).

[Isozaki02] Isozaki, H. and Kazawa, H.: Efficient Support Vector Classifiers for Named Entity Recognition, *Proc. of the 19th International Conference on Computational Linguistics (COLING 2002)*, pp. 390–396 (2002).

[Ittycheriah00] Ittycheriah, A., Franz, M., Zhu, W. and Ratnaparkhi, A.: IBM's statistical question answering system, *Proc. of TREC-9*, NIST (2000).

[Ittycheriah01a] Ittycheriah, A., Franz, M. and Roukos, S.: IBM's Statistical Question Answering System – TREC-10, *Proc. of TREC 2001*, NIST (2001).

[Ittycheriah01b] Ittycheriah, A., Franz, M., Zhu, W. and Ratnaparkhi, A.: Question Answering Using Maximum-Entropy Components, *Proc. of NAACL 2001*, pp. 33–39, ACL (2001).

[Jaakkola00] Jaakkola, T., Diekhans, M. and Haussler, D.: A discriminative framework for detecting remote protein homologies, *Journal of Computational Biology*, Vol. 7, No. 1,2, pp. 95–114 (2000).

[Joachims98] Joachims, T.: Text Categorization with Support Vector Machines: Learning with Many Relevant Features, *Proc. of European Conference on Machine Learning (ECML '98)*, pp. 137–142 (1998).

[Kashima02] Kashima, H. and Koyanagi, T.: Kernels for Semi-Structured Data, *Proc. 19th International Conference on Machine Learning (ICML2002)*, pp. 291–298 (2002).

[Kashima03] Kashima, H., Tsuda, K. and Koyanagi, T.: Marginalized Kernels Between Labeled Graph, *Proc. 20th International Conference on Machine Learning (ICML2003)*, pp. 321–328 (2003).

[Kimeldorf71] Kimeldorf, G. S. and Wahba, G.: "A Correspondence between Bayesian Estimation on Stochastic Proccesses and Smoothing by Splines" (1971).

[KreBel98] KreBel, Y. H.-G.: *Advances in Kernel Methods, Pairwise Classification and Support Vector Machines*, pp. 255–268, MIT Press (1998).

[Kudo02] Kudo, T. and Matsumoto, Y.: Japanese Dependency Analysis Using Cascaded Chunking, *Proc. of the 6th Conference on Natural Language Learning (CoNLL 2002)*, pp. 63–69 (2002).

[Kudo03] Kudo, T. and Matsumoto, Y.: Fast Methods for Kernel-based Text Analysis, *Proc. of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-2003)*, pp. 24–31 (2003).

[Leslie04] Leslie, C., Eskin, E., Cohen, A., Weston, J. and Noble, W.: Mismatch string kernels for discriminative protein classification, *Journal of Bioinformatics*, Vol. 20, No. 4, pp. 467–476 (2004).

[Li02] Li, X. and Roth, D.: Learning Question Classifiers, *Proc. of the 19th International Conference on Computational Linguistics (COLING 2002)*, pp. 556–562 (2002).

[Lodhi02] Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N. and Watkins, C.: Text Classification Using String Kernel, *Journal of Machine Learning Research*, Vol. 2, pp. 419–444 (2002).

[Morishita00] Morishita, S. and Sese, J.: Traversing Itemset Lattices with Statistical Metric Pruning, *Proc. of ACM SIGACT-SIGMOD-SIGART Symp. on Database Systems (PODS'00)*, pp. 226–236 (2000).

[Pang02] Pang, B., Lee, L. and Vaithyanathan, S.: Thumbs up? sentiment classification using machine learning techniques, *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (*EMNLP*), pp. 79–86 (2002).

[Pang04] Pang, B. and Lee, L.: A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts, *In Proc. of the 42st Annual Meeting of the Association for Computational Linguistics (ACL-2004)*, pp. 271–278 (2004).

[Pei01] Pei, J., Han, J., Mortazavi-Asl, B. and Pinto, H.: PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth, *Proc. of the 17th International Conference on Data Engineering (ICDE 2001)*, pp. 215–224 (2001).

[Salton75] Salton, G., Wong, A. and Yang, C.: A Vector Space Model for Automatic Indexing, *Communication of the ACM*, Vol. 11, No. 18, pp. 613–620 (1975).

[Salton83] Salton, G. and McGill, M. J.: *Introduction to Modern Information Retrieval*, McGraw-Hill (1983).

[Sasaki01] Sasaki, Y., Isozaki, H., Taira, H., Hirao, T., Kazawa, H., Suzuki, J., Kokuryo, K. and Maeda, E.: SAIQA: A Japanese QA System Based on Large-Scale Corpus (in Japanese), *IPSJ SIG-NL NLP-145*, pp. 77–, IPSJ (2001).

[Schölkof01] Schölkof, B. and Smola, A. J.: *Learning with Kernel*, MIT Press (2001).

[Schölkopf95] Schölkopf, B., Burges, C. and Vapnik, V.: Extracting support data for a given task, *Proc. of First International Conference on Knowledge Discovery & Data Mining*, AAAI Press. (1995).

[Sugiyama91] Sugiyama, K. and Misue, K.: Visualization of structural information: Automatic drawing of compound digraphs, *IEEE Trans. Systems, Man and Cybernetics*, Vol. 21, No. 4, pp. 876–892 (1991).

[Suzuki03a] Suzuki, J., Hirao, T., Sasaki, Y. and Maeda, E.: Hierarchical Directed Acyclic Graph Kernel: Methods for Natural Language Data, *Proc. of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-2003)*, pp. 32–39 (2003).

[Suzuki03b] Suzuki, J., Sasaki, Y. and Maeda, E.: Kernels for Structured Natural Language Data, *Proc. of the 17th Annual Conference on Neural Information Processing Systems (NIPS2003)* (2003).

[Suzuki03c] Suzuki, J., Taira, H., Sasaki, Y. and Maeda, E.: Question Classification using HDAG Kernel, *Workshop on Multilingual Summarization and Question Answering (MSQA-2003)*, pp. 61–68 (2003).

[Tamura96] Tamura, N. and Wada, K.: Text Structureing by Composition and Decomposition of Segments, *Journal of Natural Language Processing*, Vol. 5, No. 1, pp. 59–78 (1996).

[Vapnik95] Vapnik, V. N.: *The Nature of Statistical Learning Theory*, Springer (1995).

[Vapnik98] Vapnik, V. N.: *Statistical Learning Theory*, John Wiley (1998).

[Voorhees99] Voorhees, E. M. and Tice, D. M.: The TREC-8 Question Answering Track Evaluation, *Proc. of the 8th Text Retrieval Conference (TREC-8)* (1999).

[Watkins99] Watkins, C.: Dynamic Alignment Kernels, *Technical Report CSD-TR-98-11*, Royal Holloway, University of London Computer Science Department (1999).

[Weston98] Weston, J. and Watkins, C.: Multi-class support vector machines, Technical Report CSD-TR-98-04, Department of Computer Science, Royal Holloway, University of London, Egham (1998).

[Weston99] Weston, J. and Watkins, C.: Support Vector Machines for Multi-Class Pattern Recognition, *Proc. of 7th European Symposium on Artificial Neural Networks* (1999).

[Zaki02] Zaki, M. J.: Efficiently Mining Frequent Trees in a Forest, *Proc. of the 8th International Conference on Knowledge Discovery and Data Mining (KDD'02)*, pp. 71–80 (2002).

[Zukerman01] Zukerman, I. and Horvitz, E.: Toward Understanding WH-Questions: A Statistical Analysis, *Proc. of Association for Computational Linguistics (ACL-2001)*, ACL (2001).

Acknowledgements

I would like to express my sincere appreciation to Professor Yuji Matsumoto of Nara Institute of Science and Technology for supervising this dissertation. I would heartily appreciate continuous supports, various criticisms and timely advice he has given to me. His encouragement helped shape the direction of my work.

I would like to express my gratitude to Professor Shin Ishii and Professor Hiroyuki Seki of Nara Institute of Science and Technology for their valuable suggestions and helpful comments.

I am indebted to Associate Professor Kentaro Inui, Dr. Masashi Shimbo and Dr. Masayuki Asahara of Nara Institute of Science and Technology for constructive and fruitful discussions.

I am also indebted to Professor Masakazu Nakanishi and Dr. Hiroaki Saito of Keio University, who were my supervisors when I was an undergraduate and master course student at Keio University.

I am greatful to my coleagues in the computational linguistics laboratory at Nara Institute of Science and Technology.

I completed this dissertion at the Intelligent Commumication Laboratory (ICL) of NTT Communication Scuence Laboratories (NN CS Labs.). I would like to gave my appreciation to Dr. Noboru Sugamura, Director of NTT CS Labs. Dr. Sigeru Katagiri, Vice-director of NTT CS Labs., Dr. Naonori Ueda, the Executive Manager of the ICL, Dr. Eisaku Maeda, the formaer group leader and Dr. Hideki Isozaki, our group leader, for providing me the opportunity to complete this dissertation at NTT CS Labs.

I wish to thank my colleagues in the ICL, especially Dr. Eisaku Maeda, Dr. Hideki Isozaki and Dr. Yutaka Sasaki. Dr. Maeda supported me and gave me valuable comments. Dr. Isozaki and Dr. Sasaki encouraged me and discussed many problems with me. Without their guideance, this dissertation would not have been written. My appreciation also goes to members of the ICL. Especially, Mr. Hajime Tsukada, Dr. Hirotoshi Taira, Mr. Hideto Kazawa, Dr. Tsutomu Hirao and Dr. Taku Kudo encouraged me to work and research.

Special thanks are also due to all of the people who gave me valuable comments and continuours encouragemant. They include ... Although I cannot list all of their names, I would like to express my thanks all of them.

Finally, I wish to thank my parents, Tadashi and Yoshiko Imaichi, for their continuous encouragements and supports.

List of Publications

Journal Papers

[1] Suzuki, J., Sasaki, Y. and Maeda, E.: "Question Type Classification Using Word Attribute N-gram and Statistical Machine Learning (in Japanese)," *Journal of IPSJ*, Vol.44, No.11, pp.2839–2853, November 2003.

[2] Sasaki, Y., Isozaki, H., Suzuki, J., Kokuryo, K., Hirao, T., Kazawa, H. and Maeda, E.: "SAIQA-II: A Trainable Japanese QA System with SVM (in Japanese)," *Journal of IPSJ*, Vol.45, No.2, pp.635–646, February 2004.

[3] Suzuki, J., Sasaki, Y. and Maeda, E.: "Hierarchical Directed Acyclic Graph Kernels (in Japanese)," *Journal of IEICE*, Vol.J88-DII, No.2, pp. 230–240, February 2005.

[4] Suzuki, J. and Sasaki, Y.: "Kernels on Hierarchical Structured Graph for Natural Language Data," *Journal of JMLR*, (submitted).

Conference Papers

[1] Suzuki, J., Sasaki, Y. and Maeda, E.: "SVM Answer Selection for Open-Domain Question Answering," In *Proceedings of 19st COLING*, pp.974–980, August 2002.

[2] Suzuki, J., Taira, H., Sasaki, Y. and Maeda, E.: "Question Classification using HDAG Kernel," In *Proceedings of 2nd MSQA*, pp.161–68, July 2003.

[3] Suzuki, J., Hirao, T., Sasaki, Y. and Maeda, E.: "Hierarchical Directed Acyclic Graph Kernel: Methods for Natural Language Data," In *Proceedings of 41st ACL*, pp.32–39, July 2003.

[4] Suzuki, J., Sasaki, Y. and Maeda, E.: "Kernels for Structured Natural Language Data," In *Proceedings of 17th NIPS*, December 2003.

[5] Suzuki, J., Isozaki, H. and Maeda, E.: "Convolution Kernels with Feature Selection for Natural Language Processing Tasks," In *Proceedings of the 42nd ACL*, pp.119–126, July 2004.

[6] Hirao, T., Suzuki, J., Isozaki, H., and Maeda, E.: "Dependency-based Sentence Alignment for Multiple Document Summarization," In *Proceedings of 20th COLING*, pp., August 2004.

List of Other Publications

[1] Suzuki, J., Hirao, T., Sasaki, Y. and Maeda, E.: "Question Type Classification using Statistical Machine Learning (in Japanese)", Forum on Information Technology (FIT), Volume 1, pp.89–90, 2002.

[2] Suzuki, J., Hirao, T., Sasaki, Y. and Maeda, E.: "Efficient Calculation for Similarity between Texts using Hierarchical Structure (in Japanese)," In *Proceedings of IPSJ SIG-NL, NL-154-15*, pp.101–108, March 2003.

[3] Suzuki, J., Hirao, T., Sasaki, Y. and Maeda, E.: "Efficient Calculation for Similarity between Texts using Structures of Texts (in Japanese)," In *Proceedings of NLP*, pp.101–108, March 2003.

[4] Suzuki, J., Hirao, T., Isozaki, H. and Maeda, E.: "String Kernel with Feature Selection Function (in Japanese)," In *Proceedings of IPSJ SIG-NL, NL-157-6*, pp.41–48, September 2003.

[5] Suzuki, J., Hirao, T., Sasaki, Y. and Maeda, E.: "Text Analysis using Hierarchically Structured Graph Kernel (in Japanese)," In *Proceedings of 6th IBIS*, pp.217–222, November 2003.

[6] Hirao, T., Suzuki, J., Isozaki, H. and Maeda, E.: "NTT's Multiple Document Summarization System for DUC2003," In *Proceedings of the 3rd Document Understanding Conference (DUC-2003)*, pp.129–133, 2003.

[7] Hirao, T., Suzuki, J., Isozaki, H. and Maeda, E.: "Multiple Document Summarization using Sequential Pattern Mining (in Japanese)", In *Proceedings of IPSJ SIG-NL, NL-158-6*, pp.31–38, November 2003.

[8] Isozaki, H., Hirao, T. and Suzuki, J.: "On Selection Criteria of Combinatorial Features for Machine Learning (in Japanese)", In *Proceedings of IPSJ SIG-NL, NL-158-10*, pp.63–68, November 2003.

[9] Kazawa, H., Suzuki, J. and Maeda, E.: SVMAP - A Large Margin Map Approximation (in Japanese) in *Proceeding of IBIS*, pp.205–210, November, 2003.

[10] Hirao, T., Suzuki, J., Isozaki, H. and Maeda, E.: "NTT's Multiple Document Summarization System for DUC2004," In *Proceedings of the 3rd Document Understanding Conference (DUC-2004)*, May 2004.

Abbreviations

IEICE The Institute of Electronics, Information and Communication Engineers

IPSJ Information Processing Society of Japan

- **SIG-NLP** Special Interested Group on Natural Language Processing
- **SIG-FI** Special Interested Group on Foundation of Informatics

JMLR Journal of Machine Learning Research

ACL Annual Meeting of the Association for Computational Linguistics

NIPS Annual Conference on Neural Information Processing Systems

COLING International Conference on Computational Linguistics

MSQA Workshop on Multilingual Summarization and Question Answering

NLP Annual Meeting of The Association for Natural Language Processing

IBIS Information-Based Induction Sciences