NAIST-IS-DT0061014

Doctor’s Thesis

Clustering Approaches to Text Categorization

HIROYA TAKAMURA

March 24, 2003

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

Doctor’s Thesis
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

HIROYA TAKAMURA

Thesis committee:	Yuji Matsumoto, Professor
	Shunsuke Uemura, Professor
	Shin Ishii, Professor
	Kentaro Inui, Associate Professor

Clustering Approaches to Text Categorization[*]

HIROYA TAKAMURA

Abstract

The aim of this thesis is to improve accuracy of text categorization, which is the basis for various applications such as e-mail classification and Web-page classification. Among the various possible approaches to this aim, two clustering approaches and an application of a new kernel (similarity function) are discussed in this thesis. Although clustering is usually regarded as an unsupervised learning method and categorization as a supervised learning, we show that clustering can be used to improve accuracy of text categorization.

The first clustering approach proposed is *co-clustering of words and texts*. In a number of previous probabilistic approaches, texts in the same category are implicitly assumed to have an identical distribution over words. We empirically show that this assumption is not accurate, and propose a new framework based on a co-clustering technique to alleviate this problem. In this method, training texts are clustered so that the assumption is more likely to be true, and at the same time, features are also clustered in order to tackle the data sparseness problem. We succeeded in improving accuracy of text categorization using the co-clustering method.

The second approach is *constructive induction based on clustering*. In this approach, Support Vector Machines (SVMs) are combined with constructive induction using dimension reduction methods, such as Latent Semantic Indexing (LSI). New features derived by dimension reduction are added to the feature space. Using this method, we succeeded in improving the categorization performance of SVMs in text categorization, especially when a number of extra unlabeled examples other than the given labeled examples are used in the dimesion reduction step.

[*] Doctor's Thesis, Department of Information Processing, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DT0061014, March 24, 2003.

Lastly we discuss the use of a kernel function based on probabilistic models. The TOP kernel is a kernel which can be used with discriminative classifiers on the basis of a probabilistic model. We first view clustering-based constructive induction from the theory of the TOP kernel. We then propose a new TOP kernel which is based on hyperplanes generated by SVMs.

Keywords:

text categorization, clustering, support vector machines, kernel method

Acknowledgements

I am grateful to Professor Yuji Matsumoto for giving me a chance to study natural language processing at the Computational Linguistics laboratory (CL-lab). He gave me lots of advices derived from his vast range of knowledge. I am also grateful to Associate Professor Kentaro Inui, Assistant Professor Edson Tadashi Miyamoto and Assistant Professor Masashi Shimbo for giving me many comments and advices.

I thank the members of the thesis committee including Professor Shunsuke Uemura and Professor Shin Ishii. They gave me useful comments which I had not noticed.

I also thank Hiroyasu Yamada at Japan Advanced Institute of Science and Technology, Taku Kudoh in the CL-lab, Tetsuji Nakagawa at Oki Electric Industry, and other former and current members of the SVM-group at the CL-lab. Discussion with them provided me with deep understanding of machine learning methods and new ideas for my research. I also thank Tsutomu Hirao in NTT Communication Science Laboratories.

Md. Maruf Hasan at Knowledge Media Institute, The Open University in United Kingdom, and Kaoru Matsunaga at IBM Japan also helped me with technical and non-technical aspects.

Dr. Hang Li at Microsoft Research Asia is the person who introduced me to Professor Matsumoto. I learned a lot from him during my 3-month stay at Microsoft Research Asia as a visiting student. I thank him, the researchers and the visiting students in that institute, especially Yunbo Cao, Cong Li and Liqun Chen.

I also thank my former supervisor, Professor Kokichi Sugihara at Tokyo University, and friends in Mathematical Engineering Department of Engineering Faculty in Tokyo University.

I am also grateful to Associate Professor Wolfgang Slany and Research Assistant Nysret Musliu at the Vienna University of Technology.

I also thank friends I met in Vienna, especially Ichiro Ohta and Kiyomi Inoue. I also thank Oliver Harting.

I thank my old friends, Hitoshi Suzuki at the University of Miami, Keisuke Sugiyama at NTT DATA and Shin-ichi Sato at Hamamatsu Photonics.

My life at NAIST was a lot of fun. I owe it to all the members of the staff and all my colleagues, especially the members of the CL-lab Football club and the CL-lab winter sports club.

I also thank the members of the NAIST Football Club.

The members of Komaba Football Club at Tokyo University also encouraged me in many aspects. I thank them as well.

Finally, I thank my family. My father, Kikuo Takamura, and my mother, Kikue Takamura, have made me what I am now. My brothers Shinya and Tomoya have also encouraged and supported me in various aspects.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Text categorization is the task in which texts are classified into one of predefined categories based on their contents. If the texts are newspaper articles, categories could be, for example, economics, politics, sports and so on. This task has various applications such as automatic email classification and web-page categorization. Those applications are becoming increasingly important in today's information-oriented society.

There are mainly two types of approaches to text categorization. One is the rule-based approach. In the rule-based approach, the classification rules are manually created usually by experts in the domain of the texts. Although the rule-based approach can achieve high accuracy, it is costly in terms of labor and time. Moreover, a rule-based system created for one domain can hardly be used in other domains. The second approach involves machine learning techniques, in which classification rules are automatically created using information from labeled (already-categorized) texts. It is cost-saving because we have only to prepare labeled texts. This cost-saving property enables a system for a new domain to be easily constructed. Owing to its wide applicability, we adopt the machine learning approach in this thesis.

Text categorization is also called text classification, document categorization or document classification. It is sometimes confused with text clustering, in which there exist no predefined categories; similar texts are simply put together to form a cluster. These two kinds of tasks should be distinguished.

The goal of this thesis is to effectively use clustering techniques (including text clustering and word clustering) for text categorization. More specifically, given a supervised classifier, we investigate how to improve the categorization performance of the classifier in text categorization by extracting effective features using unsupervised or supervised clustering methods. We propose three methods for this purpose:

1. *co-clustering of words and texts*: in a number of previous probabilistic approaches, texts in the same category are implicitly assumed to have an identical distribution over words. We empirically show that this assumption is not accurate, and propose a new framework based on a co-clustering technique to alleviate this problem. In this method, the training texts are clustered so that the assumption is more likely to be true, and at the same time, features (words) are also clustered in order to tackle the data sparseness problem.

2. *constructive induction based on clustering*: new features extracted by unsupervised dimension reduction methods are added to the feature space. Support Vector Machines can effectively use the expanded feature space.

3. *an application of TOP kernel*: a kernel (similarity) function is constructed on the basis of the hyperplanes of SVMs. The negative class of a binary classification is considered to be a mixture of several categories.

Consider how these three methods relate to previous machine learning approaches to text categorization.

In the early 90's, classical machine learning methods such as k-nearest neighbor classifiers [8, 57], decision tree learners [20, 31] and Naive Bayes classifiers [36, 31, 35] were applied to text categorization. They have been followed by their variants and improvements, including word clustering [3] and feature selection [58]. The first topic in this thesis, co-clustering of words and texts, builds upon such methods.

Since the late 1990s, new machine learning methods, the so called Large Margin classifiers such as Support Vector Machines (SVMs) [54, 9, 6] and AdaBoost [19, 46] have been proposed. In particular, SVMs have been applied and achieved high accuracy in various fields such as object recognition [41] and digit recognition [54]. SVMs have also been applied to tasks in NLP such as text categorization [26], chunk identification [29], named entity extraction [56], and many others [37, 30].

Also for the Large Margin classifiers, many variants have been proposed (e.g., [27, 49]). Moreover, many kernel functions, which can be used in combination with SVMs or other methods, have been proposed. Those include the convolution kernel [22] with application to parsing [7], the string kernel [34] with application to text categorization.

One general trend in recent developments for both classical and Large Margin classifiers is the incorporation of unlabeled data. For example, Naive Bayes classifiers can be enhanced with the Expectation-Maximization (EM) algorithm [38]. Transductive SVMs [27] can use unlabeled examples as labeled, by iteratively relabeling unlabeled examples. The second topic in this thesis, constructive induction based on clustering, is oriented to the use of unlabeled data.

This trend, the use of unlabeled data, has surged over the development of kernel functions. The Fisher kernel [25] is based on probabilistic models, which can be estimated using unlabeled examples. Another kernel based on probabilistic models is the TOP (Tangent vector Of the Posterior log-odds) kernel [52], although the way to use unlabeled examples is not trivial. In the estimation of the model, TOP kernels need to either use labeled data or adopt EM-like methods, because TOP kernels use the posterior probability of categories. In the third topic of this thesis, we view the first two topics from the viewpoint of the TOP kernel and also discuss the use of TOP kernel in text categorization.

Next we look at each of the three proposed methods, in terms of the data that the method uses.

Unlabeled texts provide co-occurrence information for words, which can be used to improve categorization performance. Although unlabeled texts are available from the internet, collecting unlabeled texts which are useful for a text categorization problem is not an easy task because of the wide diversity of texts on the Internet. For example, personal diaries can be noise in the categorization of newspaper articles. Those heterogeneous texts are not useful for text categorization. For this reason, it is useful to consider two types of situations: when only labeled texts are available, and when a large quantity of (unlabeled) homogeneous texts as well as the labeled texts are available. We propose solutions for each situation. The co-clustering approach corresponds to the first situation, and the clustering-based constructive induction and the hyperplane-based TOP kernel correspond to the second situation.

This thesis is organized as follows.

Before going into the main topics, several types of machine learning methods are described in Chapter 4.2.

In Chapter 3, the foundations of text categorization are explained. In particular, through a small experiment, we will see how serious the data-sparseness problem is. In Chapter 4, related work is described.

The co-clustering approach, the first topic, is described in Chapter 5.

Chapter 6 describes the second approach, constructive induction based on clustering.

In Chapter 7, we first view the clustering-based constructive induction from the theory of the TOP kernel. We then discuss the use of the TOP kernel to text categorization.

Finally in Chapter 8, we conclude the thesis.

Chapter 2

Machine Learning Methods

Various machine learning methods used in text categorization are described in this chapter. Naive Bayes classifiers in Section 2.1 and Support Vector Machines in Section 2.2 are supervised classifiers. Kernel functions based on probabilistic models are explained in Section 2.3. Co-clustering described in Section 2.4 is an unsupervised clustering method. The Akaike Information Criterion in Section 2.5 is an information theoretic criterion used for model selection. Constructive Induction is described in Section 2.6. In Section 2.7, methods to apply binary classifiers for multi-class classification are described.

2.1 Naive Bayes Classifiers

The Naive Bayes (NB) classifier [36] is a probabilistic classifier based on the Naive Bayes assumption. Given a feature vector $\mathbf{x}$ $(= (x_1, \cdots, x_n))$, the posterior probability of a category c is, from the Bayes rule,

$$P(c|\mathbf{x}) = \frac{P(c)p(\mathbf{x}|c)}{p(\mathbf{x})}. \tag{2.1}$$

We would like to predict the category c_{max} which yields the maximum value for (2.1). The parameter $P(c)$ is estimated as

$$P(c) = \frac{number\ of\ documents\ in\ c}{number\ of\ documents}. \tag{2.2}$$

The parameter $p(\mathbf{x})$ does not affect the categorization results because it does not depend on the categories. The problem we are faced with is how to estimate $p(\mathbf{x}|c)$. In NB classifiers, $p(\mathbf{x}|c)$ is modeled on the Naive Bayes assumption:

$$p(\mathbf{x}|c) = \prod_i p(x_i|c), \tag{2.3}$$

which is equivalent to assuming that the components of feature vectors are statistically independent of each other.

If the maximum likelihood estimation (Appendix A.1) is used to estimate the parameter $p(x_i|c)$, then

$$p(x_i|c) = \frac{N(x_i,c)}{N(c)}, \tag{2.4}$$

where $N(x,c)$ denotes the joint frequency of x and c, and $N(c) = \sum_x N(x,c)$. However, in this estimation method, if some x_i does not appear in the training data, the probability of any instance containing x_i becomes zero, regardless of the other features in the vector. To avoid zero probability, the parameter $p(x_i|c)$ is estimated using Laplacian prior probabilities, as follows

$$p(x_i|c) = \frac{N(x_i,c)+\lambda}{N(c)+\lambda|V|}, \tag{2.5}$$

where λ is a positive constant, and $|V|$ denotes the number of features. Usually, 1.0 or 0.5 is chosen as the value of λ. This smoothing method is equivalent to adding λ to the frequencies of all the features.

The NB classifier predicts the category c_{max} with the largest posterior probability:

$$\begin{aligned} c_{max} &= \arg\max_c P(c|\mathbf{x}) && (2.6) \\ &= \arg\max_c P(c)p(\mathbf{x}|c). && (2.7) \end{aligned}$$

2.2 Support Vector Machines

Support Vector Machines (SVMs) are binary classifiers which were originally proposed by Vapnik [53]. SVMs have achieved high accuracy in various tasks, such as object recognition [41] and digit recognition [54]. Among several available packages, we use TinySVM[1] in the later experiments.

Suppose a set of ordered pairs consisting of a feature vector and its label

$$\begin{gathered}(\mathbf{x}_1,y_1),(\mathbf{x}_2,y_2),\cdots,(\mathbf{x}_l,y_l), \\ \forall i,\ \mathbf{x}_i \in \mathbf{R}^d, y_i \in \{-1,1\},\end{gathered} \tag{2.8}$$

is given. In SVMs, a separating hyperplane with the largest margin (the distance between the hyperplane and its nearest vectors, see Figure 2.1):

$$f(\mathbf{x}) = \mathbf{w}\cdot\mathbf{x}+b \tag{2.9}$$

is constructed, on the condition that the hyperplane discriminates all the training examples correctly (this condition will relaxed in non-separable case)[2]. Note that the distance from the hyperplane to the nearest positive example should be the same as the distance from the hyperplane to the nearest negative example (otherwise margin is not maximized).

Margin is computed as follows. The condition that all the training examples are classified correctly is formulated as

$$y_i(\mathbf{x}_i\cdot\mathbf{w}+b)-1 \geq 0. \tag{2.10}$$

The equality of the above expression (2.10) must hold for the nearest examples. Those nearest examples form two margin-boundary hyperplanes: one formed by the nearest positive examples, the other formed by the nearest negative examples. Let λ be the distance between these two margin-boundary hyperplanes, and $\bar{\mathbf{x}}$ be a vector on the margin-boundary hyperplane formed by the nearest negative examples.

Then, the following equations hold:

$$-1\times(\bar{\mathbf{x}}\cdot\mathbf{w}+b)-1=0, \tag{2.11}$$

$$1\times((\bar{\mathbf{x}}+\lambda\mathbf{w}/|\mathbf{w}|)\cdot\mathbf{w}+b)-1=0. \tag{2.12}$$

Margin is half of the distance λ and computed as

$$\lambda/2 = 1/|\mathbf{w}|. \tag{2.13}$$

[1] Available from http://cl.aist-nara.ac.jp/˜taku-ku/software/TinySVM/

[2] Some authors define margin for each example, as the distance between the example and the hyperplane. With this definition, the hyperplane that maximizes the minimum *margin* is constructed.

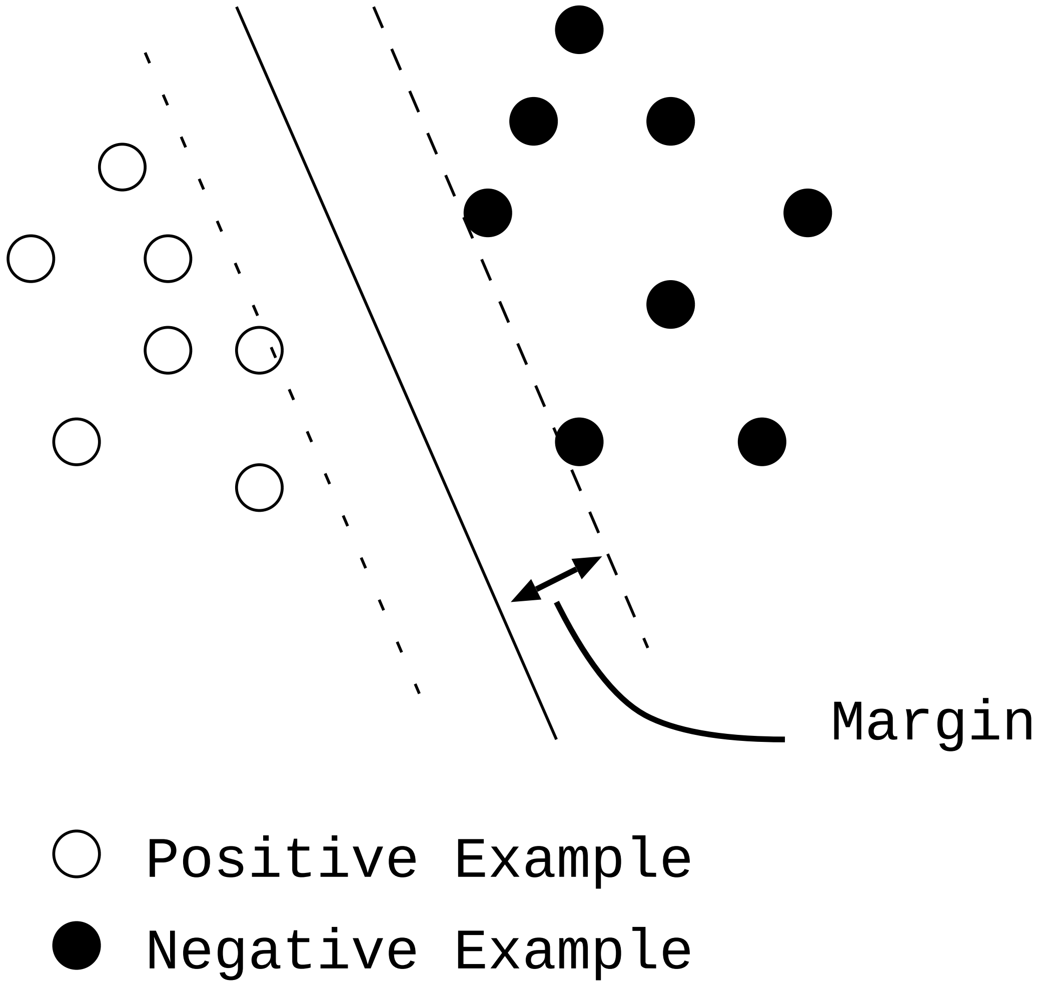

Figure 2.1. Support Vector Machine

Therefore, maximizing the margin is equivalent to minimizing the norm of **w**.

So far, we have seen a general framework for SVMs. In the following two sections, we will see the formulation of SVMs in two different cases.

2.2.1 Separable Case

First a simple case is considered, where the given data is linearly separable. As mentioned in the previous section, finding the largest margin is equivalent to minimizing the norm $|\mathbf{w}|$. This problem is formulated as:

$$\begin{array}{rc} \min. & \frac{1}{2}|\mathbf{w}|^2, \\ \text{s.t.} & \forall i,\ y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \geq 0. \end{array} \tag{2.14}$$

The Lagrangian L of this problem is

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}|\mathbf{w}|^2 - \sum_i \alpha_i (y_i(\mathbf{x}_i \cdot \mathbf{w} - b) - 1), \tag{2.15}$$

where α_i $(\forall i)$ are the Lagrange multipliers.

We would like to minimize $L(\mathbf{w}, b, \alpha)$ with respect to **w** and b under the constraints $\alpha_i \geq 0$ $(\forall i)$. The stationary condition states

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_i \alpha_i y_i \mathbf{x}_i = \mathbf{0}, \tag{2.16}$$

$$\frac{\partial L}{\partial b} = \sum_i \alpha_i y_i = 0. \tag{2.17}$$

Simple substitutions lead us to the following Lagrangian form:

$$L(\mathbf{w}, b, \alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j. \tag{2.18}$$

From the Lagrangian theory, we have only to *maximize* the Lagrangian with respect to α_i $(\forall i)$ to reach the optimum.

This new problem is called the dual problem of (2.14), which is formulated as

$$\begin{array}{rc} \max. & \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \\ \text{s.t.} & \sum_i \alpha_i y_i = 0, \\ & \forall i,\ \alpha_i \geq 0. \end{array} \tag{2.19}$$

The advantage of the dual problem is that this dual problem has a quadratic form which has been widely studied and can be solved more easily than the primal problem (2.14).

Using α_i^* $(\forall i)$ which maximize (2.19), the optimal $\mathbf{w}^*, b^*$ are expressed as

$$\mathbf{w}^* = \sum_i \alpha_i y_i \mathbf{x}_i, \tag{2.20}$$

$$b^* = -\frac{b_{neg} + b_{pos}}{2}, \tag{2.21}$$

where

$$b_{neg} = \max_{i:y_i=-1} (\mathbf{w}^* \cdot \mathbf{x}_i), \tag{2.22}$$

$$b_{pos} = \min_{i:y_i=1} (\mathbf{w}^* \cdot \mathbf{x}_i). \tag{2.23}$$

By substituting (2.20) and (2.21) into (2.9), we obtain

$$f(\mathbf{x}) = \sum_i \alpha_i^* y_i \mathbf{x}_i \cdot \mathbf{x} + b^*. \tag{2.24}$$

Examples are classified according to the sign of (2.24).

2.2.2 Non-separable Case

In the preceding section, we solved the problem under the constraints $y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \geq 0$ $(\forall i)$, which mean that the data must be separated linearly. However, real data is usually not linearly separable and cannot be solved under those constraints. Introducing the so-called slack variables enables the non-separable problems to be solved [9].

The problem is formulated as

$$\begin{array}{rl} \min. & \frac{1}{2}|\mathbf{w}|^2 + C\sum_i \xi_i, \\ \text{s.t.} & \forall i,\ y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 + \xi_i \geq 0, \\ & \forall i,\ \xi_i \geq 0, \end{array} \tag{2.25}$$

where ξ_i $(\forall i)$ are slack variables. The intuition behind this formulation is that as few examples as possible are allowed to penetrate into the margin or even into the other side of the hyperplane.

The Lagrangian L of this problem is

$$\begin{aligned} & L(\mathbf{w}, b, \alpha, \xi, \mu) \\ & = \tfrac{1}{2}|\mathbf{w}|^2 + C\textstyle\sum_i \xi_i - \sum_i \alpha_i(y_i(\mathbf{x}_i \cdot \mathbf{w} - b) - 1 + \xi_i) - \sum_i \mu_i \xi_i, \end{aligned} \tag{2.26}$$

where α_i and μ_i $(\forall i)$ are the Lagrange multipliers, C is a user-given constant.

We would like to minimize $L(\mathbf{w}, b, \alpha, \xi, \mu)$ with respect to $\mathbf{w}$, b and ξ under the constraints $\xi_i \geq 0,\ \alpha_i \geq 0,\ \mu_i \geq 0\ (\forall i)$. The stationary condition states

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_i \alpha_i y_i \mathbf{x}_i = \mathbf{0}, \tag{2.27}$$

$$\frac{\partial L}{\partial b} = \sum_i \alpha_i y_i = 0, \tag{2.28}$$

$$\frac{\partial L}{\partial b} = C - \alpha_i - \mu_i = 0. \tag{2.29}$$

Simple substitutions lead us to the following Lagrangian form:

$$L(\mathbf{w}, b, \alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j. \tag{2.30}$$

From the Lagrangian theory, we have only to *maximize* the Lagrangian with respect to α_i $(\forall i)$ to reach the optimum.

We thus obtain the dual problem of (2.25):

$$\begin{array}{ll} \text{max.} & \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \\ \text{s.t.} & \sum_i \alpha_i y_i = 0, \\ & \forall i,\ 0 \leq \alpha_i \leq C. \end{array} \tag{2.31}$$

The constraint $\alpha_i \leq C$ comes from $\mu_i \geq 0$ and $C - \alpha_i - \mu_i = 0$.

Using optimal α_i^* $(\forall i)$, the optimal $\mathbf{w}^*, b^*$ are expressed as in the separable case.

2.2.3 Kernel Methods

Since SVMs are linear classifiers, their separating ability is limited. To compensate for this limitation, the *kernel method* is usually combined with SVMs [53].

In the kernel method, the dot-products in (2.31) and (2.24) are replaced with more general inner-products $K(\mathbf{x}_i, \mathbf{x})$, called the kernel function. The polynomial kernel $(\mathbf{x}_i \cdot$

$\mathbf{x}_j + 1)^d$ $(d \in \mathbf{N}_+)$ and the RBF kernel $\exp\{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma^2\}$ are often used. Using the kernel method means that feature vectors are mapped into a (higher dimensional) Hilbert space and linearly separated there. This mapping structure makes non-linear separation possible, although SVMs are basically linear classifiers.

Another advantage of the kernel method is that although it deals with a high dimensional (possibly infinite) Hilbert space, explicit computation of high dimensional vectors is not required. Only the general inner-products of two vectors are needed. This advantage leads to a relatively small computational overhead.

2.3 Kernels from Probabilistic Models

Recently a new type of kernel which connects generative models of data and the discriminative classifiers such as SVMs, have been proposed: the Fisher kernel [25] and the TOP (Tangent vector Of the Posterior log-odds) kernel [52].

2.3.1 Fisher Kernel

Suppose we have a probabilistic generative model $p(\mathbf{d}|\theta)$ of the data (we denote a sample by **d**). The Fisher score of **d** is defined as $\nabla_\theta \log p(\mathbf{d}|\theta)$, where ∇_θ means partial differentiation with respect to the parameters θ. The Fisher information matrix is denoted by $I(\theta)$ (this matrix defines the geometric structure of the model space). Then, the Fisher kernel at an estimate $\hat{\theta}$ is given by :

$$K(\mathbf{d}^1, \mathbf{d}^2) = (\nabla_\theta \log p(\mathbf{d}^1|\hat{\theta}))^t I^{-1}(\hat{\theta})(\nabla_\theta \log p(\mathbf{d}^2|\hat{\theta})) \tag{2.32}$$

The Fisher score approximately indicates how the model will change if the sample is added to the training data used in the estimation of the model. That means, the Fisher kernel between two samples will be large, if the influences of the two samples are similar and large [51].

The matrix $I(\theta)$ is often approximated by the unit matrix to avoid large computational overhead.

2.3.2 TOP Kernel

On the basis of a probabilistic model of the data, TOP kernel is designed to extract feature vectors $\mathbf{f}_{\hat{\theta}}$ which are considered to be useful for the categorization with a separating hyperplane.

We begin with the proposition that, between the generalization error $R(\mathbf{f}_{\hat{\theta}})$ and the expected error of the posterior probability $D(\mathbf{f}_{\hat{\theta}})$, the following relation holds:

$$R(\mathbf{f}_{\hat{\theta}}) - L^* \leq 2D(\mathbf{f}_{\hat{\theta}}), \tag{2.33}$$

where L^* is the Bayes error. $D(\mathbf{f}_{\hat{\theta}})$ is expressed, using a logistic function $F(t) = 1/(1+\exp(-t))$, as

$$D(\mathbf{f}_{\hat{\theta}}) = \min_{\mathbf{w},b} E_{\mathbf{x}} |F(\mathbf{w}\cdot\mathbf{f}_{\hat{\theta}} - b) - P(y=+1|\mathbf{x},\theta^*)|, \tag{2.34}$$

where θ^* is the actual parameter of the model.

The expression (2.33) means that minimizing $D(\mathbf{f}_{\hat{\theta}})$ leads to reducing the generalization error $R(\mathbf{f}_{\hat{\theta}})$. The TOP kernel extracts features which *can* minimize $D(\mathbf{f}_{\hat{\theta}})$. In other words, we would like to have feature vectors $\mathbf{f}_{\hat{\theta}}$ which satisfy

$$\forall \mathbf{x}, \quad \mathbf{w}\cdot\mathbf{f}_{\hat{\theta}}(\mathbf{x}) - b = F^{-1}(P(y=+1|\mathbf{x},\theta^*)), \tag{2.35}$$

with certain values of $\mathbf{w}$ and b.

For that purpose, we first define a function $v(\mathbf{x},\theta)$:

$$\begin{aligned} v(\mathbf{x},\theta) &= F^{-1}(P(y=+1|\mathbf{x},\theta)) \\ &= \log(P(y=+1|\mathbf{x},\theta)) - \log(P(y=-1|\mathbf{x},\theta)). \end{aligned} \tag{2.36}$$

The first-order Taylor expansion of $v(\mathbf{x},\theta^*)$ around the estimate $\hat{\theta}$ is

$$v(\mathbf{x},\theta^*) \approx v(\mathbf{x},\hat{\theta}) + \sum_i (\theta_i^* - \hat{\theta}_i) \frac{\partial v(\mathbf{x},\hat{\theta})}{\partial \theta_i}. \tag{2.37}$$

If $\mathbf{f}_{\hat{\theta}}$ is of the form:

$$\mathbf{f}_{\hat{\theta}}(\mathbf{x}) = \left(v(\mathbf{x},\hat{\theta}), \frac{\partial v(\mathbf{x},\hat{\theta})}{\partial \theta_1}, \cdots, \frac{\partial v(\mathbf{x},\hat{\theta})}{\partial \theta_p} \right), \tag{2.38}$$

and if **w** and b are properly chosen as

$$\begin{aligned}\mathbf{w} &= (1, \theta_1^* - \hat{\theta}_1, \cdots, \theta_p^* - \hat{\theta}_p), && (2.39)\\ b &= 0, && (2.40)\end{aligned}$$

then (2.35) is approximately satisfied. Thus, the TOP kernel is defined as

$$K(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{f}_{\hat{\theta}}(\mathbf{x}_1) \cdot \mathbf{f}_{\hat{\theta}}(\mathbf{x}_2). \qquad (2.41)$$

Detailed discussion of TOP kernel and its theoretical analysis can be found in a paper by Tsuda et al [52].

2.4 Likelihood-based Hard Clustering

In this section, a hard clustering method based on likelihood is introduced. A detailed explanation of this method is written in a paper by Li [32]. This clustering method is called *co-clustering*, because it clusters two sets of items simultaneously.

Suppose we have co-occurrence data $S = \{(x_1, y_1), (x_2, y_2), \cdots, (x_m, y_m)\}$ and assume that these samples were generated from the probability model:

$$\begin{aligned}P(x,y) = \;& P(C_x, C_y) P(x|C_x) P(y|C_y) && (2.42)\\ & x \in C_x,\; y \in C_y\end{aligned}$$

The log-likelihood of the given data is

$$\begin{aligned}& \textstyle\sum_{(x,y)\in S} \log P(x,y)\\ = \;& \textstyle\sum_{(x,y)\in S} \log P(C_x, C_y) P(x|C_x) P(y|C_y)\\ = \;& \textstyle\sum_{(x,y)} N(x,y) \log P(C_x, C_y) P(x|C_x) P(y|C_y), && (2.43)\end{aligned}$$

where $N(x)$ denotes the frequency of x.

The parameters of this model is computed with the maximum likelihood estimation as follows:

$$\begin{aligned}P(C_x, C_y) &= \frac{N(C_x, C_y)}{|S|}, && (2.44)\\ P(x|C_x) &= \frac{N(x)}{N(C_x)}, && (2.45)\\ P(y|C_y) &= \frac{N(y)}{N(C_y)}. && (2.46)\end{aligned}$$

The clustering method described here is of bottom-up type. At each step, a pair of either two clusters for X (the first dimension) or two clusters for Y (the second dimension) is selected and merged. Merging always causes the non-negative reduction of likelihood, regardless of which pair is selected. Among all the pairs, the pair with the least likelihood reduction is basically selected, although several variants to this method can be created by controlling the selection of pairs. In the algorithm used by Li and Abe [33], given two positive integers k and l, merging for the first dimension X is performed k times, followed by l merges for the second dimension Y. In Chapter 5, several variants to this clustering method are described.

An efficient implementation of this algorithm is described in a paper by Li [32].

2.5 The Akaike Information Criterion

One problem in the clustering algorithm explained in the preceding section is when to stop merging. The Akaike Information Criterion (AIC) is a criterion which determines a stopping point on the basis of information theory.

The maximum likelihood estimation (Appendix A.1) is a method by which the fixed numbers of parameters of a model are estimated. However, if we are in a situation where we need to select a model among several models with different numbers of parameters, the maximum likelihood estimation tends to select a model with larger numbers of parameters, because the model can be closely fitted to the training data by adjusting those parameters. Selecting those models with a larger numbers of parameters often leads to over-fitting; such models do not fit test data. AIC, proposed by Akaike [1], is one of the criteria which help avoid such over-fitting. In AIC, both the likelihood of the data and the complexity of the model are taken into account.

AIC is expressed as

$$AIC = -2L + 2k, \tag{2.47}$$

and the model that minimizes this value should be selected. In our application, we use $AIC = -L + k$, since the multiplication by -2 in (2.47) simply stems from a historical reason.

A similar criterion is the Minimum Description Length (MDL) proposed by Rissanen [44]. According to the MDL principle, the model which minimize the sum of the data description length and the model description length, namely $-L + k \log |D|/2$ is selected, where $|D|$ denotes the number of data samples.

2.6 Constructive Induction

Constructive induction is a type of induction learning in which new features are created from original features. Constructive induction is effective when the original features are not able to express the data [15].

Famous examples of systems in which constructive induction is implemented can be found in papers by Bloedorn and Michalski [5] or Pagallo and Haussler [39]. In the system AQ17-DCI [5], several *expansion operators* are defined, such as *Equivalence* indicating whether or not two features are equivalent, and *Greater-than* indicating whether or not one feature is greater than another feature. The values of these operators for some pairs of features are iteratively added to the feature set after their statistical or empirical tests of effectiveness.

2.7 Binary Classification to Multi-class Classification

Since some classifiers including SVMs are binary classifiers, we need a framework to augment a binary classifier to a multi-class classifier.

There are two well-known methods [18] for that purpose. One is the *one-versus-rest* method, which is also called the *winner-takes-all* method. In this method, classification is performed for each category. We suppose that the classifier used here produces a score indicating how likely an example belongs to each category. Examples are classified into the category that has the maximum score. When SVMs are used, the distance $\mathbf{w} \cdot \mathbf{x} - b$ plays the role of such score.

The other is *pairwise* method, also called *one-against-one* method. In this method, classification is performed for each pair of two categories. After finishing the classification for all the pairs of categories, the category of each example is determined by voting : if a category wins against another category, the former category obtain one vote. For each example, the category which has obtained the largest number of votes is assigned.

Apart from the two methods mentioned above, the use of error-correcting output code has been proposed [14].

Chapter 3

Text Categorization

Text categorization (text classification) is the task in which texts are classified into one of the predefined categories using information from training (labeled) texts. This task has various applications such as automatic email classification and web-page categorization. In this chapter, several properties of text categorization are described.

3.1 Vector Space Model

One of the simplest ways to model texts is Vector Space Model [45]. In this model, a text is represented as a vector whose components are the frequencies of words. Instead of frequencies, binary features are sometimes used indicating whether a word exists in the text or not. Most of the methods for text categorization and Information Retrieval are based on the Vector Space Model. Although the model has lost the information about the order of word occurrences, there is actually no better model.

3.2 Data Description

As in other tasks, there are several common data sets in text categorization. In this section, various properties with two widely-used data sets are described. These data sets are used in the later experiments.

Table 3.1. The categories of Reuters-21578

Category	Number of training texts	Number of test texts
earn	2725	1051
acq	1490	644
money-fx	464	141
grain	399	135
crude	353	164
trade	339	133
interest	291	100
ship	197	87
wheat	199	66
corn	161	48

3.2.1 Reuters-21578

Reuters-21578[1] is probably the most widely used data set for text categorization. All the texts in this data set were collected from the Reuters newswire in 1987. Although the original data set contains 21578 texts, researchers use a data-splitting method to extract a training set and a test set. The most popular data-splitting method is ModApte-split [17], which extracts 9603 training texts and 3023 test texts. However, there are still unsuitable texts in the 9603 training texts. For example, some texts contain only "blah blah blah". We deleted those texts, because those texts can lead us to an inexact conclusion of research. After deletion, we obtained 8815 training texts.

Our training set consisting of 8815 samples has 116 different category labels. Table 3.1 shows the 10 most frequent categories (except for "others") and their numbers of positive examples. These categories are listed in the same order as in the other literature (e.g. [26]).

Texts in Reuters-21578 are allowed to have multiple labels, so in Table 3.1 one text can be counted more than once as a positive example in several different categories.

As features used in the experiments conducted in this thesis, we use the frequencies of nouns, verbs, proper nouns, adjectives and adverbs which occur five times or more

[1] Available from http://www.daviddlewis.com/resources/

in the whole training set. Stemming was conducted using TreeTagger [47].

3.2.2 20-newsgroup

20-newsgroup[2] is also a common data set for text categorization. Although 20-newsgroup is less popular than Reuters-21578, it is still used by many researchers (e.g. [3, 35]). The articles in this data set are postings to some newsgroups, unlike Reuters-21578 are taken from newswire. Another big difference between 20-newsgroup and Reuters-21578 is that texts in 20-newsgroup are not allowed to have multiple category labels. In addition, the category set has a hierarchical structure (e.g. “sci.crypt”, “sci.electronics”, “sci.med” and “sci.space” are subcategories of “sci (science)”).

Table 3.2 shows the categories in 20-newsgroup and their numbers of texts. There is no fixed way to split 20-newsgroup into a training set and a test set. This table also shows that the sizes of categories are relatively uniform compared with those of Reuters-21578.

As features, we use the frequencies of nouns, verbs, proper nouns, adjectives and adverbs which occur 10 times or more in the whole data set. Stemming was conducted using TreeTagger [47].

3.3 Sparsity of Text Categorization Data

Many tasks in natural language processing including text categorization are suffered from the data-sparseness problem. The data-sparseness problem refers to the problem that each feature appears so rarely in training data that estimating reliable parameters becomes difficult. If words are used as features, the words that never appeared in training data can be often seen in test data. Such words can not make any contribution to the categorization.

Through experiments, we illustrate how serious this problem is. Our test set of Reuters-21578 contains 15197 different words (236563 tokens). We first investigate how many of the 15197 words are covered by the training sets with various sizes. We use the data set before deleting low-frequency words. The result is shown in Figure 3.1.

[2] Available from http://kdd.ics.uci.edu/

Table 3.2. The categories of 20-newsgroup

Category	Number of texts
alt.atheism	799
comp.graphics	974
comp.os.ms-windows.misc	985
comp.sys.ibm.pc.hardware	982
comp.sys.mac.hardware	961
comp.windows.x	980
misc.forsale	972
rec.autos	990
rec.motorcycles	994
rec.sport.baseball	994
rec.sport.hockey	999
sci.crypt	991
sci.electronics	981
sci.med	990
sci.space	987
soc.religion.christian	999
talk.politics.guns	910
talk.politics.mideast	940
talk.politics.misc	775
talk.religion.misc	628

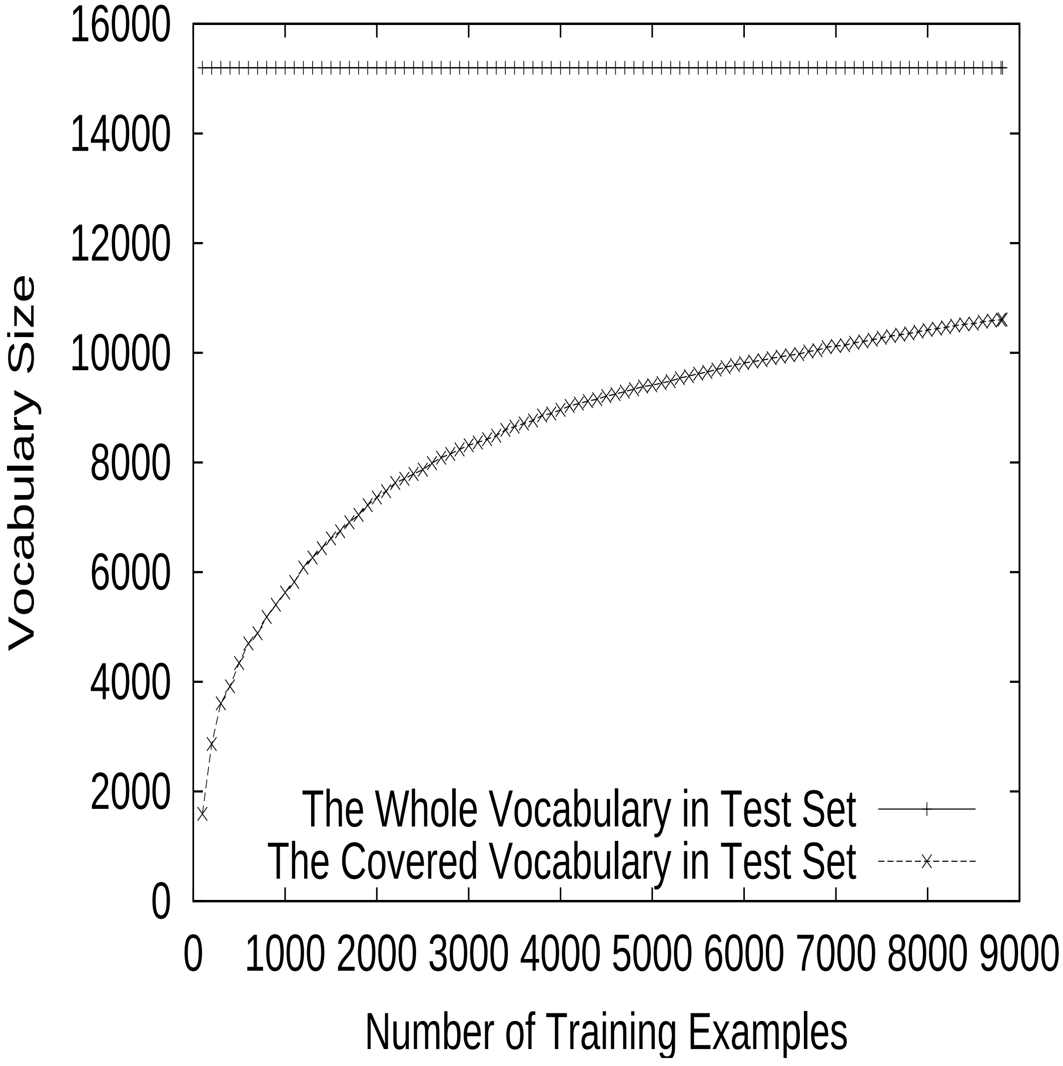

Figure 3.1. Size of vocabulary in test set covered by training set

Figure 3.1 shows that many of the words in the test set are not covered by the training set. Especially when the number of training examples is around 1000, the covered vocabulary size is only about one-third of the whole vocabulary size.

Next we count the numbers of tokens covered by the training sets with various sizes. In this experiment, we set a threshold (0, 10, 50 and 100) and count the number of tokens whose frequencies are more than the threshold. The result is shown in Figure 3.2. Although, for small thresholds, the difference between the numbers of all the tokens and the covered tokens is small, the reliable estimation of parameters for the infrequent words is expected to be hard. This experiment shows that the frequent tokens are not sufficiently covered by the training sets.

It is expected that making good use of those uncovered words leads to improvement of categorization performance.

3.4 Evaluation Methods

There are several evaluation measures for text categorization. Among them, we explain F-measure, which will be used in the experiments, followed by description of averaging methods and statistical tests.

3.4.1 F-measure

We first describe how to evaluate the binary categorization performance for one category. In binary classification problems, we would like to know whether each text belongs to the category or not. The texts belonging to a category are called *positive* (with respect to the category), while the texts not belonging to the category are called *negative* (with respect to the category).

Suppose we have conducted categorization experiments and obtained the result summarized in the *contingency table* (Table 3.3). Let I denote the set of the texts which are actually positive, and J denote the set of the texts which are predicted as positive by a classifier. Table 3.3 means that a is the size of $I \cap J$, b is the size of $I \cap \bar{J}$, c the size of $\bar{I} \cap J$, and d is the size of $\bar{I} \cap \bar{J}$.

A simple accuracy $(a+d)/(a+b+c+d)$ is not appropriate as a measure for binary text categorization. The reason is that in most data sets there are many more positive

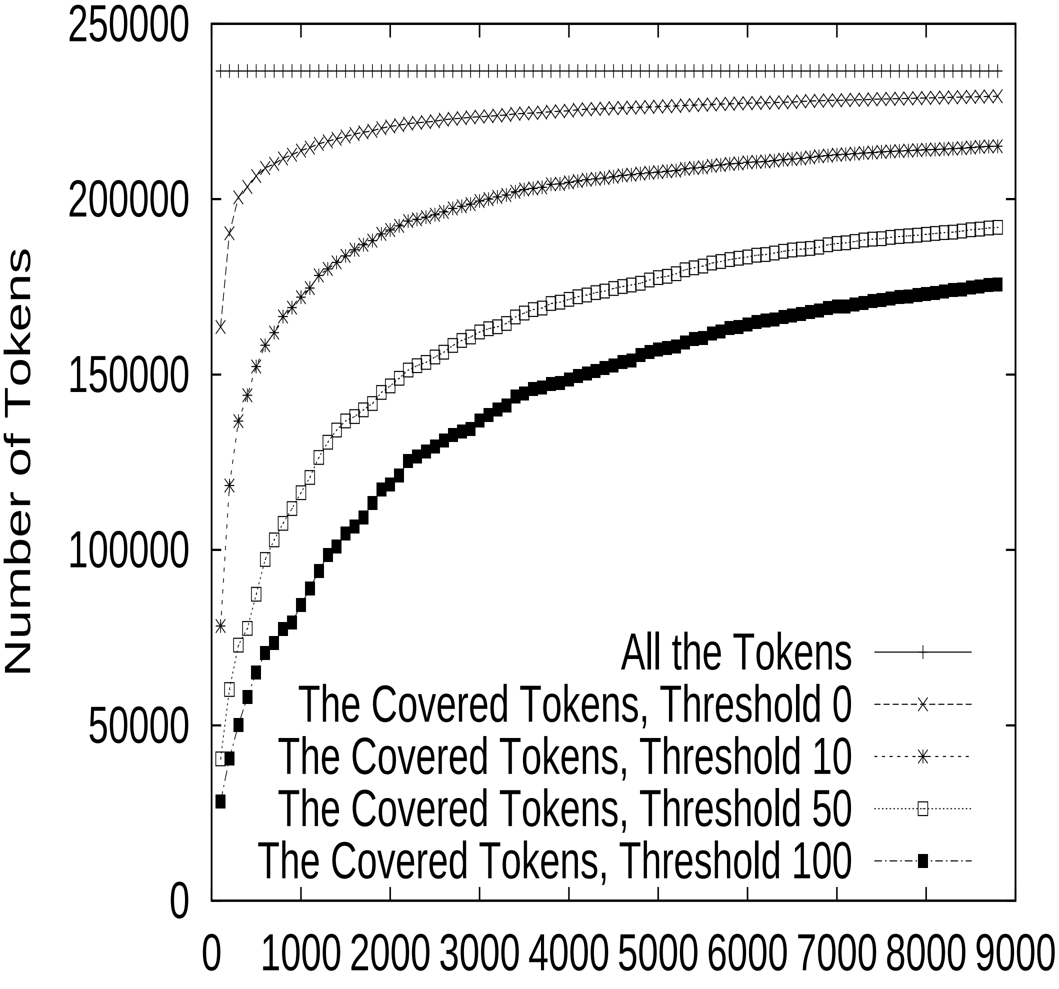

Figure 3.2. Number of tokens in test set covered by training set

Table 3.3. Contingency table of a result

	predicted positive	predicted negative
actual positive	a	b
actual negative	c	d

texts than negative ones; a trivial prediction that all texts are negative can achieve a high accuracy. Therefore, more appropriate measures must be defined.

Now we define two measures, each of which is not sufficient by itself alone. *Precision* P is defined as $a/(a+c)$, indicating how many of the predicted positive texts are actually positive. *Recall* R is defined as $a/(a+b)$, indicating how many of the actual positive texts the prediction covers. The classifier which realizes both high precision and high recall can be regraded as a good classifier. These two measures are combined as follows and make *F-measure*:

$$F_{\beta} = \frac{1}{\beta(1/P) + (1-\beta)(1/R)}. \tag{3.1}$$

Usually β is set as 0.5 and F-measure is simply expressed as

$$F_{0.5} = \frac{2PR}{P+R}. \tag{3.2}$$

Although there are some other widely used measures such as *break-even point*, throughout this thesis, we use F-measure $F_{0.5}$ whenever an evaluation measure for each category is needed.

3.4.2 Averaging F-measures

F-measure is computed for each category. To evaluate the performance across categories, F-measures have to be averaged. There are two main kinds of averaged values, namely, micro average and macro average [57]. Micro average is obtained by first computing the precisions and the recalls for all the categories and then using them to compute the F-measure. Macro average is computed by first calculating the F-measures for all the categories and then taking their arithmetic mean. Micro average tends to be dominated by large-sized categories, and macro average by small-sized ones.

3.4.3 Statistical Tests

After computing performance measures (say, F-measures) for some methods, we have to check whether the difference of the measures is statistically significant or not. In this section, we explain the statistical tests which can be used for that purpose. These tests are, of course, not exclusive to text categorization.

First, we give the definition of p-value. The p-value is the probability that, given the data, the statistics computed under a null hypothesis is realized. In other words, p-value is the smallest significance-level with which the null hypothesis can be rejected. If the p-value is smaller than the significance-level (usually, 0.001 or 0.05 is used), then the null hypothesis is rejected.

Suppose we compare model A with model B in terms of F-measure. A usual null hypothesis is “the F-measure by A and the F-measure by B have an identical distribution”. If the p-value is small enough (smaller than the significance-level), the null hypothesis is rejected. That is, two F-measures have different distributions. Since we would usually like to show that one model is superior to another, small p-values are desired.

Next, we will mention two kinds of tests used to compute p-value and explain when those tests can be used. The prerequisite condition for both tests is that you have matched pairs, such as the F-measures for several categories yielded by two models.

- Wilcoxon Signed Rank Test : This test can be used when a difference between each pair can be defined and those differences can be ranked. For example, a difference of two F-measures can be defined by subtracting one F-measure from the other.

- Sign Test : This test is used when you know only whether one value of each pair is better than, equal to, or worse than the other. For example, if classifier A correctly classified an example and classifier B incorrectly classified it, you know that A is better than B, but you do not know how much A is better than B.

It is known that Wilcoxon Signed Rank Test is more sensitive than Sign Test.

Chapter 4

Related Work

4.1 Text Categorization with SVMs

Joachims first applied SVMs to text categorization [26]. Although the model of the text used in their framework was a simple Vector Space Model (Section 3.1), they achieved an outstanding improvement over other methods. They argue that SVMs are appropriate for text categorization because SVMs can handle high dimensional feature spaces and few relevant features, which are main properties of text categorization.

4.2 Text Categorization with NB classifiers

NB classifiers described in Section can be applied to text categorization in two different ways [35]. One is called *multi-variate Bernoulli model*, and the other is called *multinomial model*. The difference between these two models stems from the interpretation of the probability $p(\mathbf{x}|c)$.

Multi-variate Bernoulli Model : in this model, we focus on whether one word appears in the text or not. A random variable x_i corresponding to a word w_i takes one of two values: this word occur in the text or not occur. Therefore, the probability $p(\mathbf{x}|c)$ is expressed as follows:

$$p(\mathbf{x}|c) \quad = \quad \prod_i p(x_i|c)$$

$$= \prod_i \Big(\delta_i p(w_i|c) + (1-\delta_i)(1-p(w_i|c)) \Big), \qquad (4.1)$$

where δ_i is 1 if w_i appears in the text, otherwise 0.

Multinomial Model in this model, a random variable x_i indicates the occurrence of some word at a specific position in the text. The occurrence of a text of length $|\mathbf{x}|$ is regarded as $|\mathbf{x}|$ trials each of which produces one of $|V|$ values ($|V|$ denotes the size of vocabulary). Therefore, the probability $p(\mathbf{x}|c)$ is expressed as follows:

$$p(\mathbf{x}|c) = P(|\mathbf{x}|)|\mathbf{x}|! \prod_i \frac{p(w_i|c)^{N(i,\mathbf{x})}}{N(i,\mathbf{x})!}, \qquad (4.2)$$

where $P(|\mathbf{x}|)$ denotes the probability that a text of length $|\mathbf{x}|$ occurs, and $N(i,\mathbf{x})$ denotes the number of occurrences of w_i in text $\mathbf{x}$.

4.3 Feature Selection

The dimension of a feature space for text data can be very large if words are used as features. Among those features, some features may provide no contribution to categorization performance and sometimes decrease accuracy. Several methods to select only good features (or eliminate bad features) have been proposed [58]. Elimination of features is beneficial also because it reduces the size of memory required for maintain the data. Some examples of such methods are explained in this section.

4.3.1 Stop-word Elimination

Among many words, some words are too frequent to work as a useful feature. For example, the verbs “be” and “have” can be seen in almost any documents. Such words are called *stop-words* and often removed from the feature set [2]. One problem in stop-word elimination is that a word can be a stop-word for a data set, but can be a useful feature for another data set.

4.3.2 Statistical Methods for Feature Selection

There are several statistical methods for features selection. Those methods provide a measure for usefulness of each word. Some examples are as follows.

Document Frequency : this measure is the number of texts in which the word appears.

Information Gain : this measure IG is computed as

$$\begin{aligned} IG &= -\sum_C P(C)\log P(C) \\ &\quad + P(w)\sum_C P(C|w)\log P(C|w) + P(\bar{w})\sum_C P(C|\bar{w})\log P(C|\bar{w}). \end{aligned} \tag{4.3}$$

Intuitively, IG measures the average entropy-reduction caused by occurrence or not-occurrence of the word.

Mutual Information : this measure MI is computed as

$$MI = -\sum_C P(C)\log\frac{P(C,w)}{P(C)P(w)}. \tag{4.4}$$

Intuitively, MI measures how strongly the word is statistically dependent on categories. Instead of taking summation in 4.4, the following expression is also used [58]:

$$MI = \max_C \log P(C,w)/(P(C)P(w)). \tag{4.5}$$

Other than these three measures, several measures have been proposed. Yang and Pedersen [58] compared feature selection methods.

4.4 Latent Semantic Indexing

Latent Semantic Indexing (LSI) [11] is a method to reduce the dimension n of the feature space. LSI provides a reduced feature space with m $(< n)$ orthogonal axes. This reduced space is optimal in the sense that the reduced space realizes the least square difference from the original space. The reduction with LSI is equivalent to mapping co-occurring words to one axis.

First, Singular Value Decomposition (SVD) is used to decompose the term-document matrix X. X $(n \times l)$ is expressed using two orthogonal matrices T $(n \times r)$ and D $(l \times r)$, and a diagonal matrix S $(r \times r)$, as

$$X = TSD^t, \tag{4.6}$$

where r is the rank of X, and the diagonal elements $\sigma_1, \sigma_2, \cdots, \sigma_r$ of S are the singular values of X ($\sigma_1 < \sigma_2 < \cdots < \sigma_r$).

Now let us take an integer m $(\leq r)$, instead of r. Let S_m denote the diagonal matrix with m largest singular values of X, T_m the matrix consisting of the first m row vectors of T, D_m the matrix consisting of the first m row vectors of D. Then the approximated matrix X_m of X is

$$X_m = T_m S_m D_m^t. \tag{4.7}$$

By mapping X into the m-dimensional subspace, we obtain the matrix $S_m D_m^t$, which can be expressed as $T_m^t X$. This $T_m^t X$ is the reduced expression of the term-document matrix. T_m can be regarded as a reduction operator.

4.5 Semantic Kernels

Kernel functions aimed for capturing semantic information have been proposed by Siolas and d'Alché-Buc [48], and Cristianini et al [10].

Semantic Kernel using Thesaurus : in the kernel function proposed by Siolas and d'Alché-Buc [48], a thesaurus is used to determine the *semantically-smoothing* matrix. Specifically, occurrences of words are smoothed according to semantic proximity computed from the length of two words in the tree structure of thesaurus. The improvement of performance of SVMs and k-nearest neighbor classifiers in text categorization is reported.

Semantic Kernel using LSI : in the kernel function proposed by Cristianini et al [10], LSI is used to determine the mapping from words to conceptual indexes. They proved that the conceptual indexes can be computed with only the inner products of example pairs. Although their method has a wide extensibility, the improvement in their experiments of text categorization was limited to a few categories.

4.6 Class-distributional Clustering

Class-distributional clustering was proposed and applied to text categorization by Baker and McCallum [3]. They theoretically proved the optimality of their clustering method in terms of the Naive Bayes score, and validated it empirically.

In class-distributional clustering, occurrences of categories given a word (or a word cluster) are regarded as a probability distribution $P(C|C_w)$, and words are clustered according to this distribution (here we denote a category by C and a word cluster by C_w). This clustering algorithm is of bottom-up type. At each step, two word clusters that are most similar with each other in terms of Jensen-Shannon (JS) divergence, are merged. JS divergence is also called as "divergence to the mean", because JS divergence is the average of two KL divergences: between clusters before merging and the cluster after merging. JS divergence is expressed as

$$JS = P(C_{w_i})D_{KL}(P(\cdot|C_{w_i})||P(\cdot|C_{w_i \cap w_j})) + P(C_{w_j})D_{KL}(P(\cdot|C_{w_j})||P(\cdot|C_{w_i \cap w_j})), \quad (4.8)$$

where $P(\cdot|C_w)$ denotes a probability distribution over categories given a cluster C_w.

$P(C|C_{w_i \cap w_j})$ is computed as

$$P(C|C_{w_i \cap w_j}) = \frac{P(C_{w_i})}{P(C_{w_i}) + P(C_{w_j})}P(C|C_{w_i}) + \frac{P(C_{w_j})}{P(C_{w_i}) + P(C_{w_j})}P(C|C_{w_j}). \quad (4.9)$$

We should note that class-distributional clustering is a supervised clustering, since it needs the co-occurrence data of categories and words.

4.7 Fisher Kernel based on PLSI

Hofmann [24] applied Fisher kernels to the text categorization under the Probabilistic Latent Semantic Indexing (PLSI) model [23].

In PLSI, the joint probability of a document $\mathbf{d}$ and a word w is :

$$P(\mathbf{d}, w) = \sum_k P(z_k)P(\mathbf{d}|z_k)P(w|z_k), \quad (4.10)$$

where the variables z_k correspond to latent classes. The parameters are estimated using EM-algorithm.

After the estimation of the model, the Fisher kernel for this model is computed.

They use *spherical parametrization* [28] instead of the original parameters in the model. Define $\rho_{jk} = 2\sqrt{P(w_j|z_k)}$ and $\rho_k = 2\sqrt{P(z_k)}$, then,

$$\frac{\partial \log p(\mathbf{d}|\hat{\theta})}{\partial \rho_{jk}} = \frac{\hat{P}(w_j|\mathbf{d})P(z_k|\mathbf{d}, w_j)}{\sqrt{P(w_j|z_k)}}, \quad (4.11)$$

$$\frac{\partial \log p(\mathbf{d}|\hat{\theta})}{\partial \rho_k} \quad \approx \quad \frac{P(z_k|\mathbf{d})}{\sqrt{P(z_k)}}, \tag{4.12}$$

where $\hat{P}(w|\mathbf{d})$ denotes the maximum likelihood estimate (Appendix A.1) of the probability of w given $\mathbf{d}$.

Thus, the Fisher kernel for this model is

$$\begin{aligned} K(\mathbf{d}^1,\mathbf{d}^2) \quad &= \quad \sum_k \frac{P(z_k|\mathbf{d}^1)P(z_k|\mathbf{d}^2)}{P(z_k)} \\ &\quad + \sum_j \hat{P}(w_j|\mathbf{d}^1)\hat{P}(w_j|\mathbf{d}^2) \sum_k \frac{P(z_k|\mathbf{d}^1,w_j)P(z_k|\mathbf{d}^2,w_j)}{P(w_j|z_k)}, \end{aligned} \tag{4.13}$$

where

$$P(z_k|\mathbf{d},w_j) \quad = \quad \frac{P(z_k)P(\mathbf{d}|z_k)P(w_j|z_k)}{\sum_l P(z_l)P(\mathbf{d}|z_l)P(w_j|z_l)} \tag{4.14}$$

$$= \quad \frac{P(z_k)P(\mathbf{d}|z_k)P(w_j|z_k)}{P(\mathbf{d},w_j)}. \tag{4.15}$$

The first term of (4.13) corresponds to the similarity through latent spaces. The second term corresponds to the similarity through the distribution of each word. The number of latent classes z_k can affect the values of the kernel function. In the experiment of [24], they computed the kernels with the different numbers (1 to 64) of z_k and added them together to make a robust kernel without deciding one specific number of latent classes z_k.

They concluded that the Fisher kernel based on PLSI is effective when a large amount of unlabeled examples are available for the estimation of the PLSI model.

Chapter 5

Co-clustering of Words and Texts

We propose a new method to improve the accuracy of text categorization using co-clustering. In a number of previous probabilistic approaches, texts in the same category are implicitly assumed to be generated from an identical distribution over words. We empirically show that this assumption is not accurate, and propose a new framework based on co-clustering to alleviate this problem. In our method, training texts are clustered so that the assumption is more likely to be true, and at the same time, features are also clustered in order to tackle the data sparseness problem. We conduct some experiments to validate this co-clustering method.

5.1 Introduction

One problem in a number of previous simple probabilistic approaches to text categorization is that texts in the same category are assumed to be generated from an identical distribution (we call it the *i.d. assumption*, in this thesis). However, categories are manually defined and there is no predefined probabilistic structure behind them, as discussed in the next section. Another problem with text categorization is the data-sparseness problem caused by the high dimensionality of the feature space. The frequency of each word is usually so small that it is difficult to estimate reliable statistics.

In order to tackle these problems, we propose a new framework based on co-clustering. Before estimating the probability model of each category, we first cluster training texts into several clusters whose elements can be thought as being generated from an identical distribution. The data-sparseness problem is more critical if the number of parameters is larger as in text clustering approach we adopt. Therefore, we alleviate this problem by clustering features (words). That is to say, we *cluster both texts and features simultaneously*. After clustering, examples are classified using a supervised classifier.

Through experiments, we show that our approach works well with Naive Bayes (NB) classifiers.

5.2 Human-made Categories and Probabilistic Structure

As observed in the previous section, a probabilistic structure does not always underlie the categories defined by humans. In order to provide empirical evidence for this observation, we conducted a small experiment using the Reuters-21578 data set. We first clustered a set of texts labeled with “earn”, which is the largest category in this corpus. We obtained seven clusters[1], each of which contains several hundred or more documents. Let us call these clusters A, B, $\cdots$, G. Then, according to the probability of word occurrence given a cluster, $P(word|cluster)$, we computed the KL (Kullback-Leibler) divergence between a cluster (A) in “earn” and the other six clusters (B, $\cdots$, G) in “earn”, and between the same cluster A and the (10 most frequent) categories other than “earn”. This model is almost equivalent to the Naive Bayes model because each word is regarded as an independent random event.

If the occurrences of words in texts in the same category are (approximately) identically distributed, the values of the KL divergence from cluster A to clusters B, C $\cdots$ G should be smaller than those to the other categories, because the clusters are labeled as “earn”. The result is shown in Figure 5.1. The six boxes on the left correspond to the divergences from cluster A to clusters B, $\cdots$, G. The box on the middle corresponds to the divergence from cluster A to the category “earn”. The other boxes correspond to the divergences from A to the other categories. The divergences to B, C and E are larger than the divergences to other categories. The result is unexpected given the i.d. assumption. Furthermore, there are two categories, namely “crude” and “ship”, whose

[1] The number of clusters is determined by the AIC (Akaike Information Criterion).

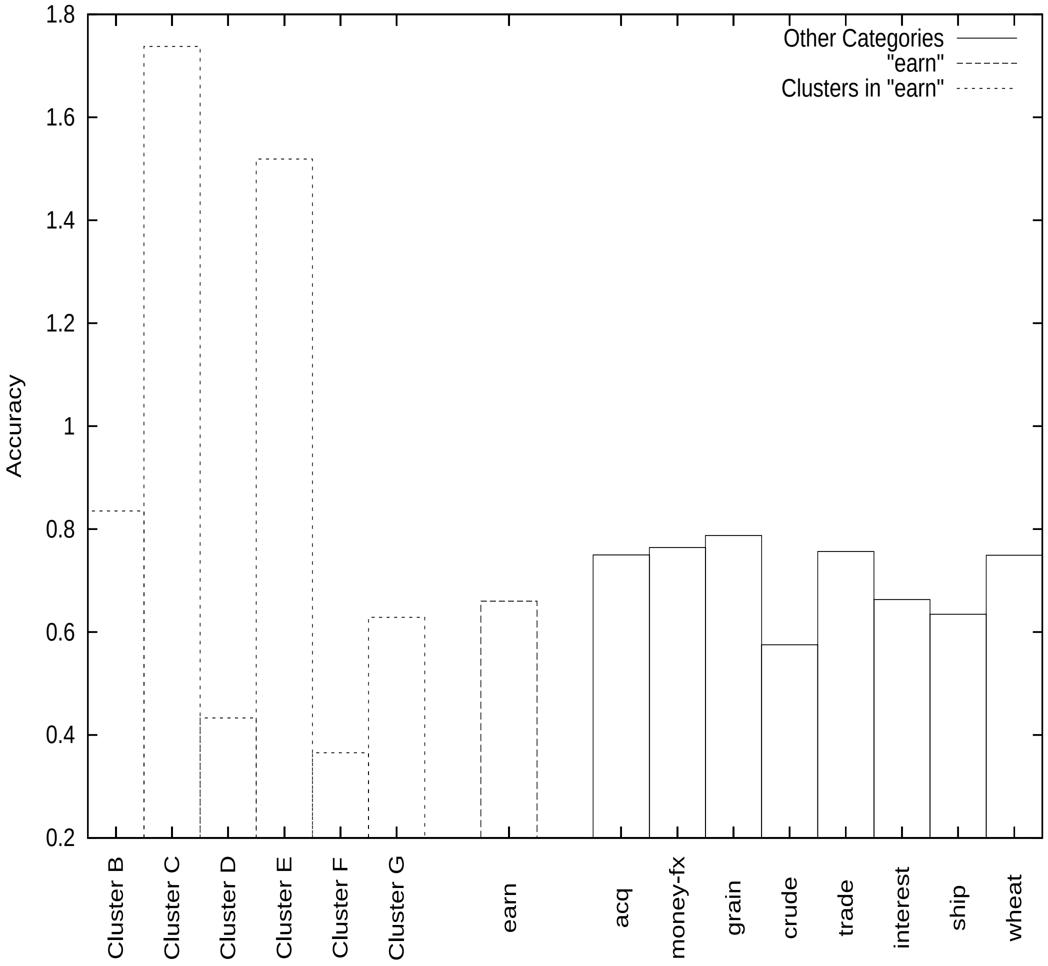

Figure 5.1. KL-divergence

divergence from A is smaller than the divergence from A to “earn”. This fact suggests that, in “earn”, there exists a set of texts (cluster A) which is more similar to another category than to “earn”.

We conducted the same experiment for the 10 most frequent categories. The phenomenon that there exists a cluster which is more similar to another category than to its own category, was observed for 3 categories (including “earn”) out of 8 categories (2 other categories had only one cluster).

These experiments suggest that the i.d. assumption does not always hold true. This fact leads to the inaccurate estimation of statistics. For example, the probability distribution over words given a category is estimated in the NB classification (Section 2.1). However, for some texts in the category, a certain word might tend to appear frequently, while for others not. In spite of that, a single value represents the probability of this word for all texts in the category, in the previous NB approaches.

Consequently, categorization accuracy should be improved if we cluster texts in order to make the i.d. assumption more likely to be true.

5.3 Co-clustering of Words and Texts

In this section, we propose a framework to overcome the problem caused by the violation of the i.d. assumption.

Our approach uses a bottom-up clustering technique explained in Section 2.4. The variables X and Y in Section 2.4 are replaced with W and D, that is, words and texts. At the initial stage, each cluster has only one word or one text. At each step, the most similar pair of words or texts is merged into one cluster. As a measure of similarity (or dissimilarity), we use the likelihood decrease caused by merging. This measure is related to Jensen-Shannon divergence.

5.3.1 Clustering Algorithm

In the algorithm by Li [32], given two positive integers k and l, merging for the first dimension is performed k times, followed by l merges for the second dimension.

We propose two variants for Li’s clustering algorithm. In both algorithms, a pair of words or texts is chosen and merged at each step, on the basis of the model described in Section 2.4. The difference is the way to choose the pair of words or texts to be merged. One is what we call *text-first* clustering, in which text clustering is conducted first, followed by word clustering. The other is *greedy* clustering, in which, at each step, the pair with the least likelihood decrease is selected from the word pairs and the text pairs, and merged. In the following, we describe these two algorithms in pseudo-code:

- Text-first Clustering

    1. Initialize
    2. Merge two *text clusters* with the least likelihood decrease repeatedly, while the stopping criterion is not satisfied.
    3. Merge two *word clusters* with the least likelihood decrease repeatedly, while the stopping criterion is not satisfied.

- Greedy Clustering

    1. Initialize
    2. Merge two *text clusters* or two *word clusters* with the least likelihood decrease repeatedly, while the stopping criterion is not satisfied.

We set two constraints. One is that only texts in the same category can be merged (we call *the category-constraint*). The other is that only words with the same part-of-speech can be merged (*pos-constraint*). The category-constraint is indispensable in our method, because of our categorization method which is explained later. Both of these constraints reduce the computational time needed for clustering.

Text-first clustering has the advantage that word clustering can be carried out using the information given by class-distribution[2]. Class-distributional clustering is a special case of text-first clustering; if the stopping criterion of the text clustering phase is set as “no two clusters can be merged without violating the category-constraint”, then text-first clustering is identical to class-distributional clustering.

[2] Precisely speaking, the information used in text-first clustering is different from the information given by class-distribution, but as clustering proceeds, these two types of information become more similar.

5.3.2 The Relation to Jensen-Shannon Divergence

Here we show that using the criterion of the least likelihood decrease is equivalent to selecting the closest pair of clusters in terms of a certain information-theoretic measure, namely the Jensen-Shannon divergence. Let ΔL denote the decrease of the log-likelihood caused by merging word-clusters C_i and C_j. Let $|S|$ denote the number of all the training examples. Using the equation $P(C_{ij},C_d) = P(C_i,C_d) + P(C_j,C_d)$, ΔL divided by $|S|$ is transformed as :

$$\begin{aligned}
\frac{\Delta L}{|S|} &= \sum_{C_d} -P(C_{ij},C_d) \log \frac{P(C_{ij},C_d)}{P(C_{ij})P(C_d)} \\
&\quad + \sum_{C_d} P(C_i,C_d) \log \frac{P(C_i,C_d)}{P(C_i)P(C_d)} + \sum_{C_d} P(C_j,C_d) \log \frac{P(C_j,C_d)}{P(C_j)P(C_d)} \\
&= \sum_{C_d} P(C_i,C_d) \Big(\log \frac{P(C_i,C_d)}{P(C_i)P(C_d)} - \log \frac{P(C_{ij},C_d)}{P(C_{ij})P(C_d)} \Big) \\
&\quad + \sum_{C_d} P(C_j,C_d) \Big(\log \frac{P(C_j,C_d)}{P(C_j)P(C_d)} - \log \frac{P(C_{ij},C_d)}{P(C_{ij})P(C_d)} \Big) \\
&= P(C_i) \sum_{C_d} P(C_d|C_i) \log \frac{P(C_d|C_i)}{P(C_d|C_{ij})} + P(C_j) \sum_{C_d} P(C_d|C_j) \log \frac{P(C_d|C_j)}{P(C_d|C_{ij})} \\
&= P(C_i) D_{KL}\Big(P(\cdot|C_i)\Big|\Big|P(\cdot|C_{ij})\Big) + P(C_j) D_{KL}\Big(P(\cdot|C_j)\Big|\Big|P(\cdot|C_{ij})\Big), \qquad (5.1)
\end{aligned}$$

where $D_{KL}(p||q)$ is the KL-divergence between the probability distributions p and q. The last line of (5.1) is the Jensen-Shannon divergence, which is also known as “KL divergence to the mean”. That is, in our method, the closest pair of clusters in terms of the Jensen-Shannon divergence is merged at each step. Conversely speaking, the clustering method used by Baker and McCallum [3] is valid in terms of the likelihood.

Li mentioned the relation between the log-likelihood decrease and the mutual information [32].

5.3.3 AIC-based Stopping Criterion

As the stopping criterion in the clustering algorithm, we adopt AIC (Akaike Information Criterion) [1]. Li [32] uses MDL (Minimum Description Length) principle [44]. We do not use MDL principle, because it predicted too few clusters in preliminary experiments. For text clustering, it predicted a smaller number of clusters than the number of categories, which is not suitable for our method because of the category-constraints.

AIC is realized as follows. The decrease of the number of parameters caused by merging a pair of clusters is

$$\Delta N_p = \begin{cases} |Number\ of\ text\ clusters| - 1, & \text{(word-merge)} \\ |Number\ of\ word\ clusters| - 1. & \text{(text-merge)} \end{cases} \tag{5.2}$$

According to AIC, the stopping criterion should be

$$-\Delta L + \Delta N_p > 0. \tag{5.3}$$

The first term ΔL denotes the decrease of log-likelihood caused by merging.

Note that, in the algorithm of text-first clustering, there are two points at which AIC is applied. One is the point when text clustering is finished. The other is when word clustering is finished.

5.4 Categorization

Although probabilistic classifiers are expected to yield good results combined with our clustering method, the performance of non-probabilistic classifiers with our method is unpredictable. For this reason, we evaluate our clustering method using NB (Naive Bayes) classifiers (Section 2.1), which are probabilistic classifiers, and SVMs (Support Vector Machines, Section 2.2), which are non-probabilistic classifiers.

For the NB classifier, we use the multinomial model explained in Section 4.2, but ignore the concern of document length, because how to deal with document length has not been investigated in detail, but is not our main point. That is, in our model, the probability $p(\mathbf{x}|c)$ is expressed as

$$p(\mathbf{x}|c) \quad = \quad \prod_i \frac{p(w_i|c)^{N(i,\mathbf{x})}}{N(i,\mathbf{x})!}. \tag{5.4}$$

For the notation and the interpretation of this expression, see Section 4.2.

We use the one-versus-rest method (Section 2.7) to apply SVMs to multi-class classification.

In our method, the training texts are clustered beforehand. Therefore, we first categorize the test texts and predict which cluster each test text belongs to. Then, we assign to each text the label of the category that the predicted cluster belongs to (all the training texts in each cluster are supposed to have the same category tag). When constructing a hyperplane of SVMs for one cluster, the training texts belonging to the other clusters in the same category are removed from the training set.

5.5 Experiments

5.5.1 Experimental Settings

The first data set we used here is Reuters-21578 described in Section 3.2.1. Preprocessing for the data is also done as in Section 3.2.1.

As explained in Section 3.2.1, some of the texts in Reuters-21578 have multiple category-tags. In the clustering phase, we introduced multiple copies of those texts and labeled each text with one of its tags, so that every text has one tag (otherwise the texts with multiple tags can never be merged according to the category-constraint). After clustering, we treat those texts as belonging to multiple clusters in the categorization phase.

We compared our method with the method based on class-distribution clustering, which reportedly shows a good performance [3]. In addition to class-distributional clustering, we also compared our method with word clustering. The word clustering algorithm used here is unsupervised and uses only co-occurrence data of words and texts. This algorithm is equivalent to the co-clustering algorithm without the text clustering phase. As for word clustering, We used word clusters as features, and simply construct a single model for a single category.

The kernel function used is the linear kernel.

The performance of each method is evaluated in terms of accuracy for multi-class classification. Here, accuracy is defined as the ratio of the number of correctly classified test examples to the number of all the test examples. Test texts with several labels are regarded as correctly classified when one of those labels matches the predicted category. Although the performance measures such as F-measure or break-even point are often used, we use accuracy so that the performances of NB classifiers and SVMs can be compared with one another directly and fairly, because NB classifiers are not suitable for binary classification. The probability estimated with the NB classifier are not reliable [16], although it performs well as a classifier. In other words, when all the class posterior probabilities are computed simultaneously, it is not appropriate classifying a text as positive or negative according to whether each posterior probability is larger than a given threshold or not. Even if we train the NB classifier as a binary classifier, the NB classifier still has a disadvantage of large bias between the sizes of positive examples and negative examples. In the experiments by Joachims [26], NB classifiers perform much worse for small categories.

The second data set is 20-newsgroup (Section 3.2.2). With this data set, we conducted complementary experiments to support the conclusion to be drawn by the experiments with Reuters-21578. To clarify the result, we use the higher categories in the category hierarchy of 20-newsgroup. That is, we conduct multi-class classification with 7 categories: “alt”, “comp”, “misc”, “rec”, “sci”, “soc” and “talk”. Since 4 of those 7 categories have subcategories in the original hierarchy, the effect of the co-clustering approach is expected to become more evident. For this data set, we compared the proposed method and the method based on class-distributional clustering.

5.5.2 Results

This section and the next section describe the experiments with Reuters-21578.

The accuracies without clustering are 0.863 and 0.890 for NB classifiers and SVMs, respectively. According to AIC, 141 was selected as the number of text clusters, which is slightly larger than the number 116 of original categories. The categories that have multiple clusters after clustering are “earn (7 clusters)”, “acq (6 clusters)”, “others (8 clusters)”, “crude (3 clusters)”, “money-fx (3 clusters)”, “grain (2 clusters)”, “interest (2 clusters)” and “trade (2 clusters)”. Examples of word clusters can be seen in Appendix B. Those clusters are extracted from the word clusters at the word compression rate predicted by AIC.

Figure 5.2 shows the performance of NB classification combined with text-first clustering, with class-distributional clustering and with word clustering, with various compression rates of words. As for text-first clustering, the accuracies after the text-clustering step are displayed because we want to clarify the influence of the clustering of texts.

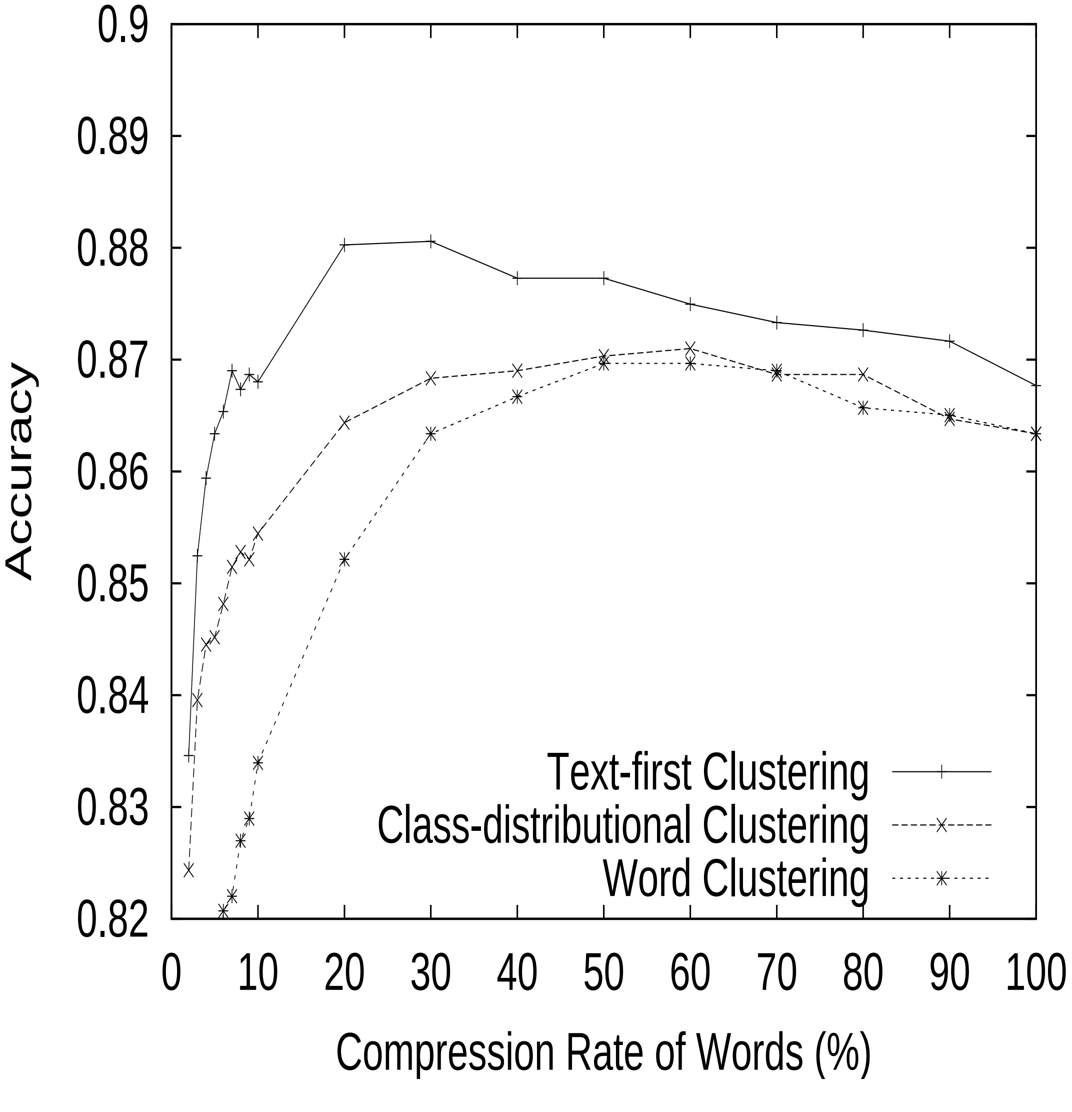

Figure 5.2. Categorization accuracy with NB classifiers (Reuters-21578)

Table 5.1. The AIC-predicted compression rate and the actual best compression rate

Clustering Method	Compression Rate(%)	Accuracy
Class-distributional(AIC)	13.4	0.859
(Actual best)	60	0.871
Text-first(AIC)	16.6	0.880
(Actual best)	30	0.881

Table 5.2. Categorization accuracy with the NB classifier combined with greedy clustering

Word Compression Rate(%)	100.0	90.0	80.0	70.0	60.0	50.0	40.0	30.0	20.0	10.0	9.7(AIC)	9.0	8.0	7.0	6.0	5.0	4.0	3.0	2.0
Text Compression Rate(%)	100.0	94.8	94.3	93.8	93.1	91.8	43.1	29.9	17.0	7.1	6.8	6.2	5.5	4.9	4.1	3.5	2.8	2.3	1.8
Accuracy	0.717	0.718	0.719	0.722	0.731	0.739	0.807	0.834	0.836	0.848	0.848	0.847	0.848	0.849	0.846	0.846	0.843	0.837	0.841

Figure 5.3 shows the performance of SVMs combined with text-first clustering, with class-distributional clustering and with word clustering, with various compression rates of words.

Table 5.1 shows the AIC-predicted compression rates and the corresponding accuracies, together with the actual best compression rates and their accuracies. Word clustering was excluded from this table, because its predicted compression rate was smaller than 1%, which corresponds to a much worse accuracy than others.

Table 5.2 shows the performance of NB classifiers combined with greedy clustering. In the case of greedy clustering, it is necessary to display both word compression rates and text compression rates, so we did not include the results of greedy clustering in Figure 5.2. In Table 5.2, the compression rates predicted by AIC, 9.7% for words and 6.8% for texts, are also displayed.

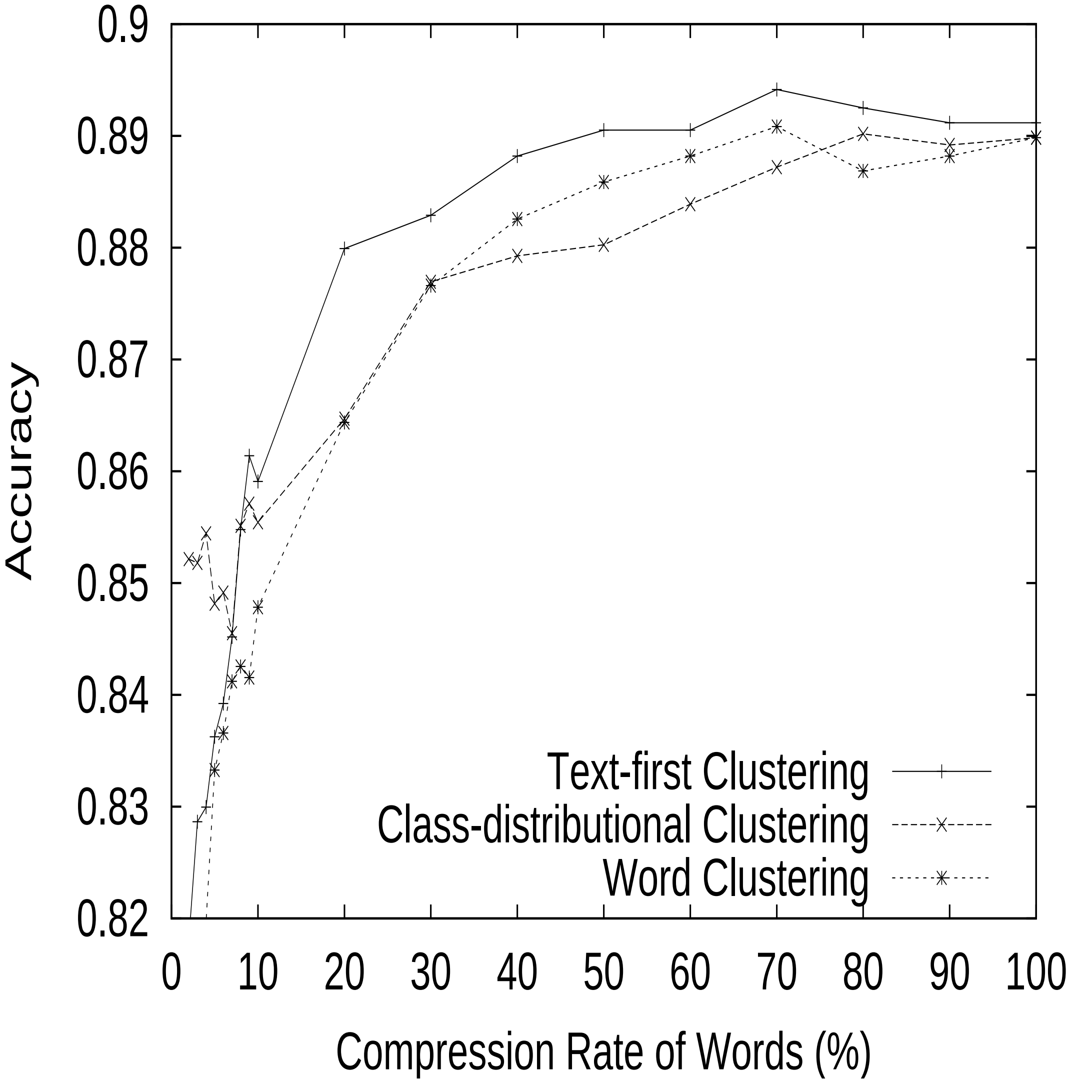

Figure 5.3. Categorization accuracy with SVMs (Reuters-21578)

5.5.3 Discussion

At the point of 100% word compression rate (i.e. every cluster corresponds to only one word) in Figure 5.2, text-first clustering performs better than class-distributional clustering and word clustering, although the difference is small (at this point, the texts have been clustered in text-first clustering). As the word compression rate decreases, the difference between those methods increases. This suggests that the combination of text clustering and word clustering works well.

Figure 5.3 shows that, also for SVMs, text-first clustering outperforms class-distributional clustering and word clustering for most compression rates of words. The apparently slight difference at 100% word compression rate was statistically significant in a sign test at 5% significance-level. Although text-first clustering tends to perform worse for smaller compression rates, the accuracy (0.894) for 70% word compression rate is higher than that for 100%. This difference was statisitically significant at 5% significance-level. On the other hand, class-distributional clustering shows only decrease in accuracy. Word clustering shows little improvement. This result indicates that text-first clustering is better than class-distributional clustering. However, clustering of words is not so effective for both clustering methods as in the case of NB classifiers, because the improvement of accuracy with text-first clustering is small and we currently have no way to predict the best word compression rate (70%, in this experiment).

The predicted compression rates in Table 5.1 are not close to the actual best compression rates, although the corresponding accuracies are close to each other for text-first clustering. The difference of two AIC-predicted accuracies is statistically significant in the sign-test (with 1% significance-level). The difference of the AIC-predicted accuracy of our method and the accuracy without clustering is also statistically significant in the same test with 5% significance-level.

Table 5.2 shows that greedy clustering does not work well. The reason may be that word clustering in the early stage cannot use the information of class-distribution. In addition to that, there are too many clusters in the early stage of greedy clustering, and classifiers are not sufficiently generalized in such situations.

5.5.4 Complementary Experiments

In the preceding section, we concluded that text-first clustering works well for NB classifiers. In this section, we conduct experiments with another data set, to support the above conclusion. The data set used here is 20-newsgroup described in Section 3.2.2. For this data set, there is no fixed way of splitting the set into training and test sets. So we conduct a five-fold cross-validation. Here only the comparison of text-first clustering and class-distributional clustering was conducted. All other experimental settings are the same as in the earlier experiments.

Results are as follows. The accuracies without clustering are 0.912 and 0.919 for NB classifiers and SVMs, respectively.

Figure 5.4 shows that the NB classifier with text-first clustering outperforms that with class-distributional clustering. This result supports the conclusion in the preceding section.

Figure 5.5 shows that, also for SVMs, the proposed method is better. This result also supports the conclusion in the preceding section. Interestingly, clustering of words is effective for SVMs with 20-newsgroup, and the NB classifier outperforms the SVMs, at the respective best compression rates.

The effectiveness of word clustering presumably depends on data sets. The conditions where word clustering works well for SVMs should be investigated.

5.6 Conclusion of the Chapter

We proposed a new method to improve the accuracy of text categorization using co-clustering. In our method, both training texts and features are clustered before estimating the probabilistic model.

Our approach is motivated by the fact that, in a number of previous probabilistic approaches, one category is assumed to have one identical probability distribution. However, this assumption is not always true, as discussed in Section 5.2. Our co-clustering approach alleviates this problem by splitting performing text clustering on the training data set. At the same time, our approach avoids the data-sparseness problem by performing word clustering on the data set.

Through experiments, we have shown that the co-clustering approach works well with Naive Bayes classifiers and that, for the SVMs, text-first clustering outperforms

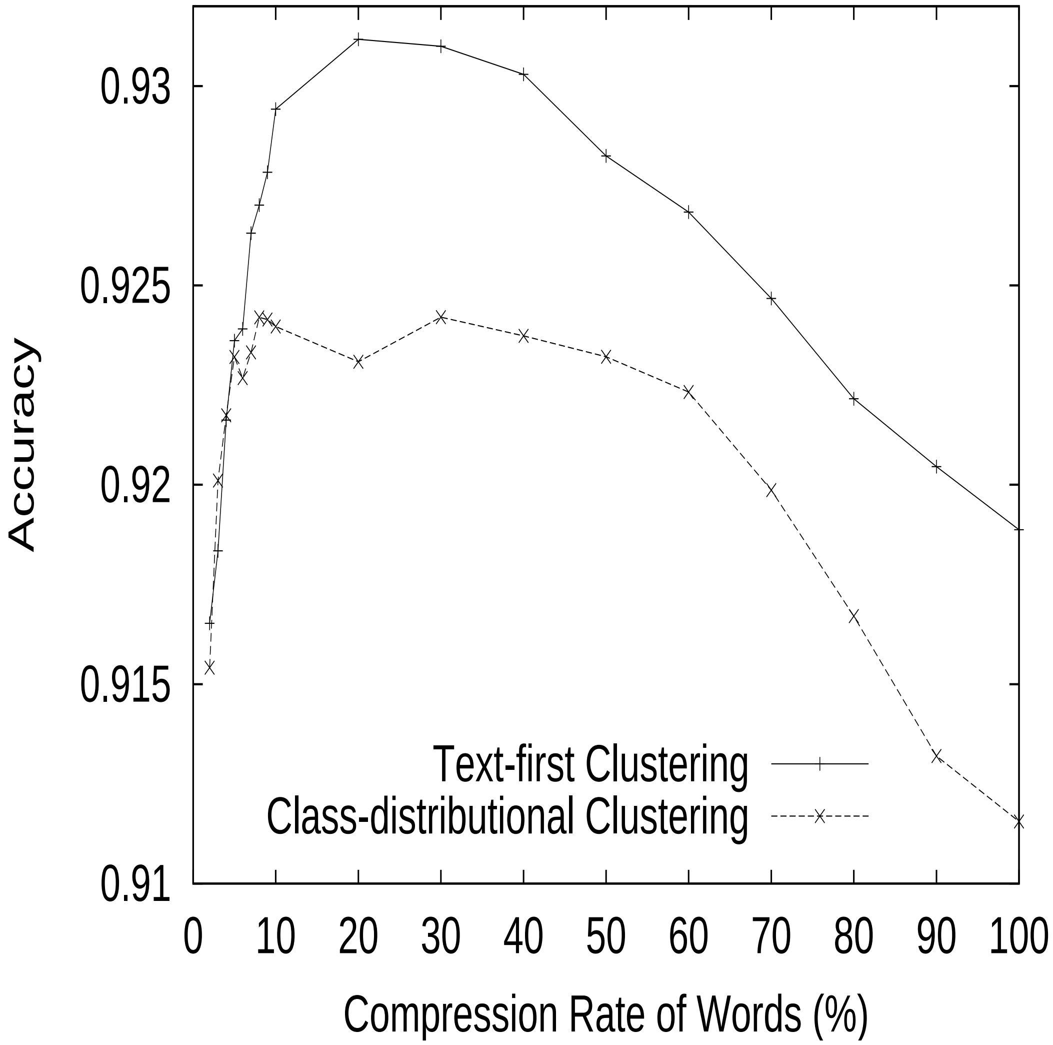

Figure 5.4. Categorization accuracy with NB classifiers (20-newsgroup)

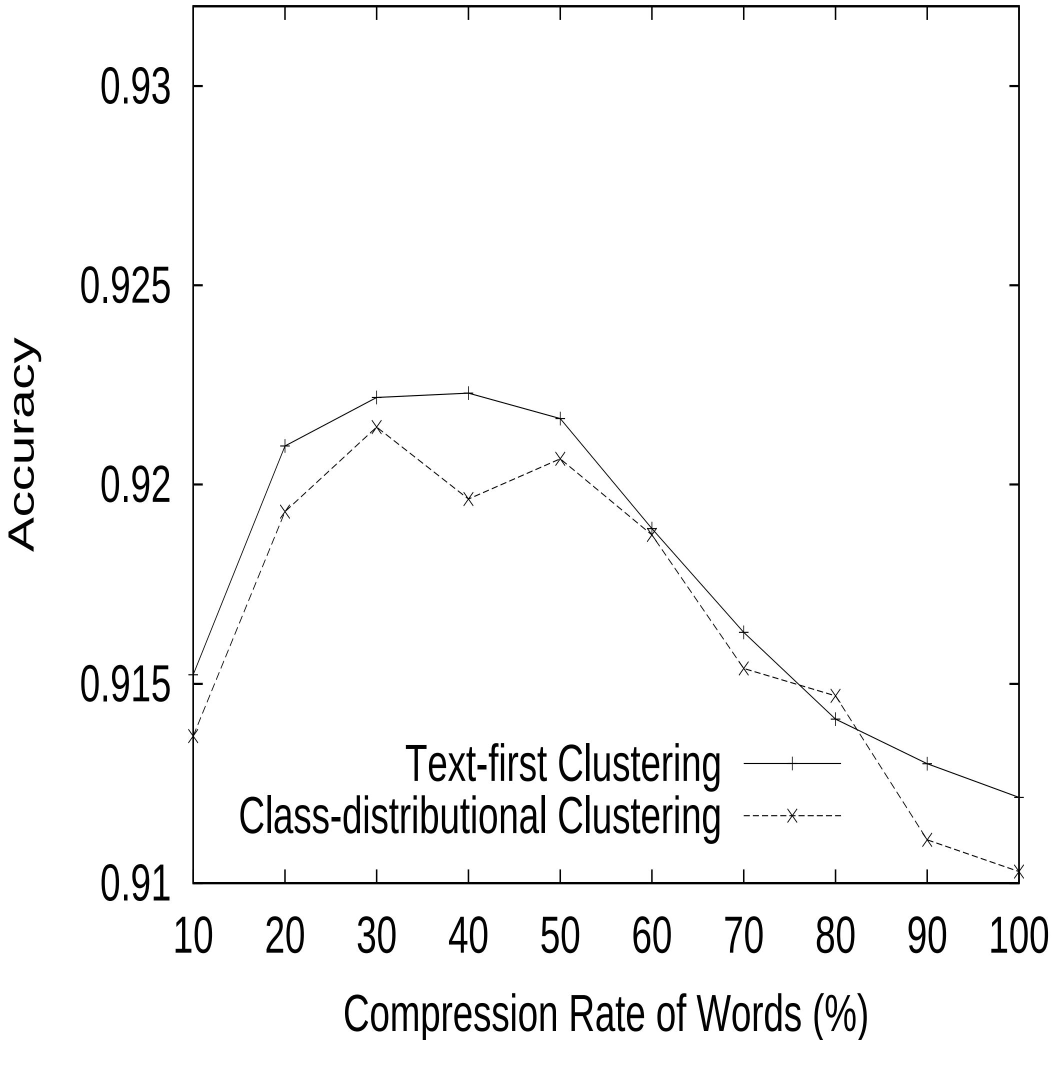

Figure 5.5. Categorization accuracy with SVMs (20-newsgroup)

class-distributional clustering and word clustering, although the effectiveness of word clustering is not clear yet.

Future work includes the following.

We should first investigate in what situations word clustering works well for SVMs.

We used AIC as a stopping criterion of text clustering step in text-first clustering. But we have not investigated whether AIC was valid as a criterion indicating the turning point from text clustering to word clustering, since it needs experiments over two-dimensional parameter space. As a stopping criterion, AIC does not always work well for SVMs. Better criteria should be pursued. In our framework, AIC is targeting the joint probability of words and texts. But, in order to obtain a better stopping criterion, AIC should be incorporated in a more sophisticated way such that it aims at the categorization accuracy itself.

Although we used an agglomerative clustering method, a divisive clustering method [13] may be better in terms of computational time.

Chapter 6

Constructive Induction based on Clustering

In this chapter, we discuss text categorization with Support Vector Machines (SVMs) combined with the constructive induction using dimension reduction methods, such as Latent Semantic Indexing (LSI). It is difficult to improve categorization ability of SVMs only with the usual dimension reduction methods. We show, however, that the categorization ability is improved by adding new features derived by dimension reduction methods. Experiments show that this method works well in text categorization, especially when a number of unlabeled examples are available.

6.1 Introduction

Although several approaches to text categorization have been proposed, most of them rely on a large amount of labeled data. However, text categorization must be realized with a small amount of labeled data because collecting training data incur a high cost. Although several methods aimed at categorization with a small labeled data set have been proposed so far [38], they need to be further developed. For that purpose, we have to take advantage of invaluable information offered by the property of unlabeled data.

The method that we discuss in this chapter is aimed at categorization with a small labeled data set.

This method uses Support Vector Machines (SVMs) combined with constructive induction using dimension reduction methods, such as Latent Semantic Indexing (LSI).

SVMs have been used in many applications such as image processing and natural language processing. The idea to apply SVMs to text categorization was first introduced by Joachims [26]. However, when the size of the labeled data set is small, SVMs often fail to produce a good result, although several efforts to solve this problem have been made. There are two main strategies for improving performance in the case of a limited amount of data. One is to modify the learning algorithm itself [27, 21]. The other is to process training data [55], including the selection of features. Our method belongs to the latter.

The dimension reduction with Latent Semantic Indexing (LSI) is often used with classifiers such as k-nearest neighbors method. LSI [11] can be regarded as a soft-clustering method, which uses co-occurrence information of words to create soft clusters of words related to the same topic. LSI is widely used in NLP and IR.

However, dimension reduction methods such as LSI do not work well with SVMs, as shown in later experiments. Taira and Haruno [49] report that the feature selection with mutual information deteriorates the performance of SVMs.

These observations suggest that the preprocessing to make use of the high classification ability of SVMs should be studied intensively. Our method is a kind of constructive induction, which uses the new features extracted with a dimension reduction method. Specifically, the dimension of the original feature space is reduced with a dimension reduction method, then those reduced features are added to the feature space. The expanded set of new features is used as input to SVMs. This method makes it possible to put weights on a subspace of the feature space without losing information from the original vectors.

The idea of using LSI for constructive induction itself is not novel. Popelinsky and Brazdil [42] used the combination of decision trees and Principal Component Analysis (PCA), which is basically equivalent to LSI, for constructive induction. However, in their paper [42], only small experiments were conducted. We do not know yet whether such constructive induction is effective for high-dimensional and sparse data such as text. Moreover, the question of whether or not SVMs can be improved should be investigated, because SVMs have a good generalization ability, but feature selection or reduction does not work for SVMs.

We also investigate the use of agglomerative hard clustering, which may have different properties compared with LSI.

To validate the effectiveness of the proposed method, we conducted several experiments in text categorization with SVMs using Reuters-21578 (Section 3.2.1) and 20-newsgroup (Section 3.2.2). We tested the following five types of feature spaces:

- the original feature space,
- the feature space reduced by LSI,
- the feature space reduced by hard clustering,
- the feature space expanded by LSI,
- the feature space expanded by hard clustering.

The result shows that the proposed methods (corresponding to the last two feature spaces above) improve the performance of SVMs when a number of unlabeled examples are available.

6.2 The Dimension Reduction Methods

We adopt the Vector Space Model in Section 3.1 to express texts in vector form. We use two types of dimension reduction methods. One is LSI, which is explained in Section 4.4. The other is hard clustering. Hard clustering is also explained in Section 2.4, however the clustering model used here is one-sided (only the words are clustered), so we will give an explanation to clarify the algorithm.

In our model, the joint probability of a word and a text is expressed as

$$\begin{aligned} P(w,\mathbf{d}) = \quad & P(C_w,\mathbf{d})P(w|C_w), \\ & w \in C_w, \end{aligned} \tag{6.1}$$

where w denotes a word and $\mathbf{d}$ denotes a text, and C_w denotes the cluster which w belongs to.

Given co-occurrence data of words and texts:

$$S = \{(w_1,\mathbf{d}_1),(w_2,\mathbf{d}_2),\cdots,(w_k,\mathbf{d}_k)\}, \tag{6.2}$$

its log-likelihood is computed as

$$\begin{aligned}\sum_{(w,\mathbf{d})\in S}\log P(w,\mathbf{d}) = & \quad \Sigma_{(w,\mathbf{d})\in S}\log P(C_w,\mathbf{d})P(w|C_w) \\ = & \quad \Sigma_{(w,\mathbf{d})} N(C_w,\mathbf{d})\log P(C_w,\mathbf{d})P(w|C_w), \end{aligned} \tag{6.3}$$

where $N(x)$ denotes the frequency of x. The parameters of (6.1) are estimated with the maximum likelihood estimation as

$$P(C_w,\mathbf{d}) = \frac{N(C_w,\mathbf{d})}{|S|}, \tag{6.4}$$

$$P(w|C_w) = \frac{N(w)}{N(C_w)}. \tag{6.5}$$

Word clusters are merged iteratively. Two word clusters are selected if their merging causes the least reduction of log-likelihood compared with other pairs.

Although this clustering method looks similar to class-distributional clustering, the two clustering methods are quite different in that class-distributional clustering is a supervised clustering method which needs the co-occurrence data of categories and words, but the above clustering method is an unsupervised clustering method which needs only the co-occurrence data of texts and words.

Let H be a matrix whose (i,j) element is 1 if the i-th cluster contains word w_j and otherwise 0. Using this matrix H, the reduced matrix of a term-document matrix X is expressed as HX.

6.3 Constructive Induction with Clustering

Constructive induction is a type of induction learning in which new features are created from original features. Constructive induction is effective when the original feature set is not rich enough to describe the various properties of the data [15].

We use clustering (dimension reduction) methods as extractors of new features for constructive induction. Thornton [50] strictly distinguishes constructive induction from the others. They insist that a learning method can be called a constructive induction only when the new features make use of relations between features (e.g. *and* or *or*). But we use the word, constructive induction, in a more general sense that some features are created from the original features.

Our approach is described in the following.

First, the feature space is reduced using LSI or hard clustering. Let M denote the reducing matrix T_m^t for LSI or H for hard clustering. The relation between an original feature vector $\mathbf{d}$ and its reduced vector $\mathbf{s}$ is

$$M\mathbf{d} = \mathbf{s}. \tag{6.6}$$

Next, the original vector $\mathbf{d}$ and the reduced vector $\mathbf{s}$ are concatenated:

$$\hat{\mathbf{d}} = \begin{bmatrix} \mathbf{d} \\ \mathbf{s} \end{bmatrix}. \tag{6.7}$$

Then, the texts are categorized with SVMs using $\hat{\mathbf{d}}$ as input.

Expanding the dimension of the feature space as above is equivalent to using a special kernel in the original feature space. We give an explanation for the linear case. Given two vectors $\mathbf{d}_1$ and $\mathbf{d}_2$, the kernel function K in the expanded space is expressed as

$$\begin{aligned} K(\hat{\mathbf{d}}_1, \hat{\mathbf{d}}_2) &= \hat{\mathbf{d}}_1 \cdot \hat{\mathbf{d}}_2 \\ &= \mathbf{d}_1 \cdot \mathbf{d}_2 + \mathbf{s}_1 \cdot \mathbf{s}_2 \\ &= \mathbf{d}_1 \cdot \mathbf{d}_2 + (M\mathbf{d}_1) \cdot (M\mathbf{d}_2). \end{aligned} \tag{6.8}$$

This kernel is different from many popular kernels such as polynomial kernels, in that the form of the mapping to a higher dimensional space depends on the given data.

Expression (6.8) implies that weights are put on the *Latent Semantic Indexes* determined by the clustering methods. We should note that weighting is different from reducing. In the dimension-reduction methods, only the Latent Semantic Indexes are considered. But in our method, the original feature space still directly influences the classification result. This property of our method makes it possible to focus on the information given by the latent semantic space, without losing information given by the original feature space.

As illustrated in Section 3.3, many words in the test set do not appear in the training set. The dimension reduction methods used in this work can include the occurrences of such uncovered words as reduced features, since the methods are unsupervised. On the other hand, a few specific words can greatly influence the categorization results [4]. Simple dimension reduction methods are unable to use the occurrences of those useful words. The proposed method, on the other hand, can make use of both the uncovered words and the useful words. Furthermore, since we use SVMs, the weights on the new features are appropriately determined, and remaining unuseful features expectedly do not deteriorate accuracy. From these observations, the proposed method is expected to improve the categorization performance.

The remaining problem is to determine the compression rate. There exist information theoretic methods such as Minimum Description Length [44] and Akaike Information Criterion [1] which can be used to determine the compression rate. But we avoid this problem simply by adding all the features reduced with various compression rates. High generalization ability of SVMs allows us to choose this simple avoidance method.

6.4 Experiments

We conducted several experiments to evaluate the proposed method. The data sets we used here are Reuters-21578 and 20-newsgroup explained respectively in Section 3.2.1 and Section 3.2.2.

First, we give an explanation about the experiments with Reuters-21578 consisting of 8815 training examples and 3023 test examples. For each of the 10 most frequent categories (Table 3.1), we conducted binary categorization experiments to predict whether each text belongs to the category or not.

For the computation of LSI and hard clustering, we use 8815 training examples or some part of it, but none of the test examples. However, their category labels are not used in the computation, since LSI and hard clustering are unsupervised.

There are four experiments using Reuters-21578. The first experiment is to investigate the categorization performance for each category with a fixed number of training examples (Section 6.4.1). The second is to observe how the performance changes when the number of training examples changes (Section 6.4.2). The third is to observe the performance for a large amount of training data (Section 6.4.3). The last experiment with Reuters-21578 is to investigate the categorization performance with a fixed number of training examples when the number of examples used for the dimension reduction changes (Section 6.4.4).

The result is evaluated using F-measure (see Section 3.4.1).

The experiment is repeated 10 times with different training examples randomly sampled from the whole training set consisting of 8815 examples (but we guarantee that there exists at least one positive example for each category). The values in the result are computed by averaging 10 F-measures with different samples (we take macro average. See Section 3.4.1).

As numbers of clusters, we take 100, 200 and 300 for LSI, and 100, 500 and 1000 for hard clustering. We use different sets of numbers of clusters for each clustering method, because the models of the two clustering methods have different numbers of parameters and comparing these two methods with the same numbers of clusters is not necessarily fair.

The kernel function used for SVMs is the linear kernel.

The experiments using 20-newsgroup in Section 6.4.5 are complementarily conducted to confirm the effectiveness of the proposed method. Experimental settings are the same as for the second experiment with Reuters-21578.

6.4.1 Performance for each Category

We investigate the categorization performance for each category with 1000 training examples.

The result is shown in Table 6.1 and Table 6.2.

The row "Method" indicates how the feature space is constructed. The column "Original" corresponds to the original feature space. The columns "LSI" and "hard" correspond to the feature spaces reduced respectively by LSI and hard clustering. The columns "Original+LSI" and "Original+hard" correspond to the feature spaces expanded respectively by LSI and hard clustering. The row "Dimension" indicates the dimensions of the reduced spaces. In the figures and tables, expressions such as "100+200+300" indicate adding all the reduced features of corresponding dimensions.

We performed the Wilcoxon Signed-Rank Test [12] for the averaged F-measure of each method. We test whether the differences of the averaged F-measures of the "Original" and the proposed methods are statistically significant or not. Since we have 10 categories and performed 10 experiments for each category, we have 100 paired F-measures. The results of the tests are shown in Table 6.3. Table 6.3 displays the names of the methods and the corresponding upperbounds of p-values (if an upperbound is less than 0.001, then we simply write 0.001). In parentheses, the dimensions of the

Table 6.1. F-measures for LSI (1000 labeled examples)

Method	Original	LSI			Original+LSI			
Dimension	–	100	200	300	100	200	300	100+200+300
earn	96.8	95.3	94.9	95.2	96.6	96.6	96.5	96.4
acq	88.8	86.8	86.3	86.8	89.5	89.4	89.1	89.5
money-fx	61.3	63.1	61.1	62.3	63.1	62.9	62.3	63.3
grain	70.1	68.1	68.8	70.8	72.0	71.0	71.2	72.0
crude	63.5	68.9	69.3	69.3	67.0	67.2	66.5	69.0
trade	63.3	65.2	62.8	63.0	65.1	63.8	64.1	64.2
interest	58.2	56.4	55.9	56.3	58.9	58.9	58.0	58.6
ship	43.6	45.1	58.5	58.8	49.4	54.2	53.6	57.9
wheat	65.8	66.1	71.3	70.8	68.1	68.6	67.9	70.5
corn	52.1	37.2	46.6	50.4	52.0	52.8	53.2	52.7
Average	66.4	65.2	67.5	68.4	68.2	68.5	68.2	69.4

Table 6.2. F-measures for hard clustering (1000 labeled examples)

Method	Original	hard			Original+hard			
Dimension	–	100	500	1000	100	500	1000	100+500+1000
earn	96.8	92.6	95.4	96.3	96.3	96.4	96.7	96.4
acq	88.8	79.0	84.5	87.2	87.3	87.9	88.7	87.7
money-fx	61.3	56.7	60.5	60.4	63.4	62.0	60.9	63.9
grain	70.1	57.3	70.9	68.6	71.7	73.3	70.5	73.7
crude	63.5	66.1	65.6	66.1	70.4	66.2	65.9	69.8
trade	63.3	58.0	63.3	63.3	67.9	64.1	63.3	66.9
interest	58.2	40.9	55.7	57.1	55.7	57.7	58.1	56.9
ship	43.6	40.1	56.4	57.3	52.9	57.3	54.6	59.9
wheat	65.8	40.7	57.5	65.1	62.8	65.1	65.2	65.0
corn	52.1	34.2	44.8	55.9	50.2	52.8	55.1	53.4
Average	66.4	56.5	65.5	67.7	67.9	68.3	67.9	69.4

Table 6.3. Results of Wilcoxon tests (p-value)

Method	Upperbound of p-value
Original+LSI(100)	0.001
Original+LSI(200)	0.001
Original+LSI(300)	0.001
Original+LSI(100+200+300)	0.001
Original+hard(100)	0.120
Original+hard(500)	0.003
Original+hard(1000)	0.014
Original+hard(100+500+1000)	0.001

reduced spaces are written.

When the number of training examples is 1000, both “Original+LSI” and “Original+hard” outperform “Original” regardless of the dimension of the reduced space. However, the results of statistical tests in Table 6.3 show that the upperbounds of p-value of “Original+hard” are higher than those of “Original+LSI”. Especially when the dimension of the reduced space is 100, the difference is not significant. On the other hand, “Original+LSI” has significant improvements.

Both “Original+LSI (100+200+300)” and “Original+hard (100+500+1000)” have significant differences from “Original”. It shows that adding several reduced features is effective.

6.4.2 Performance for Various Training Set Sizes

The purpose of this experiment is to observe how the performance changes when the number of training examples changes. The number of training examples ranges from 100 to 1000. The averaged numbers of positive examples for 100 training examples are shown in Table 6.4. Since training examples are sampled randomly from 8815 examples with 116 categories, some training examples can belong to the categories not in Table 6.4.

The result is shown in Figure 6.1 and Figure 6.2. Figure 6.1 corresponds to LSI. Figure 6.2 corresponds to hard clustering. In the range shown in Figure 6.1 and Figure

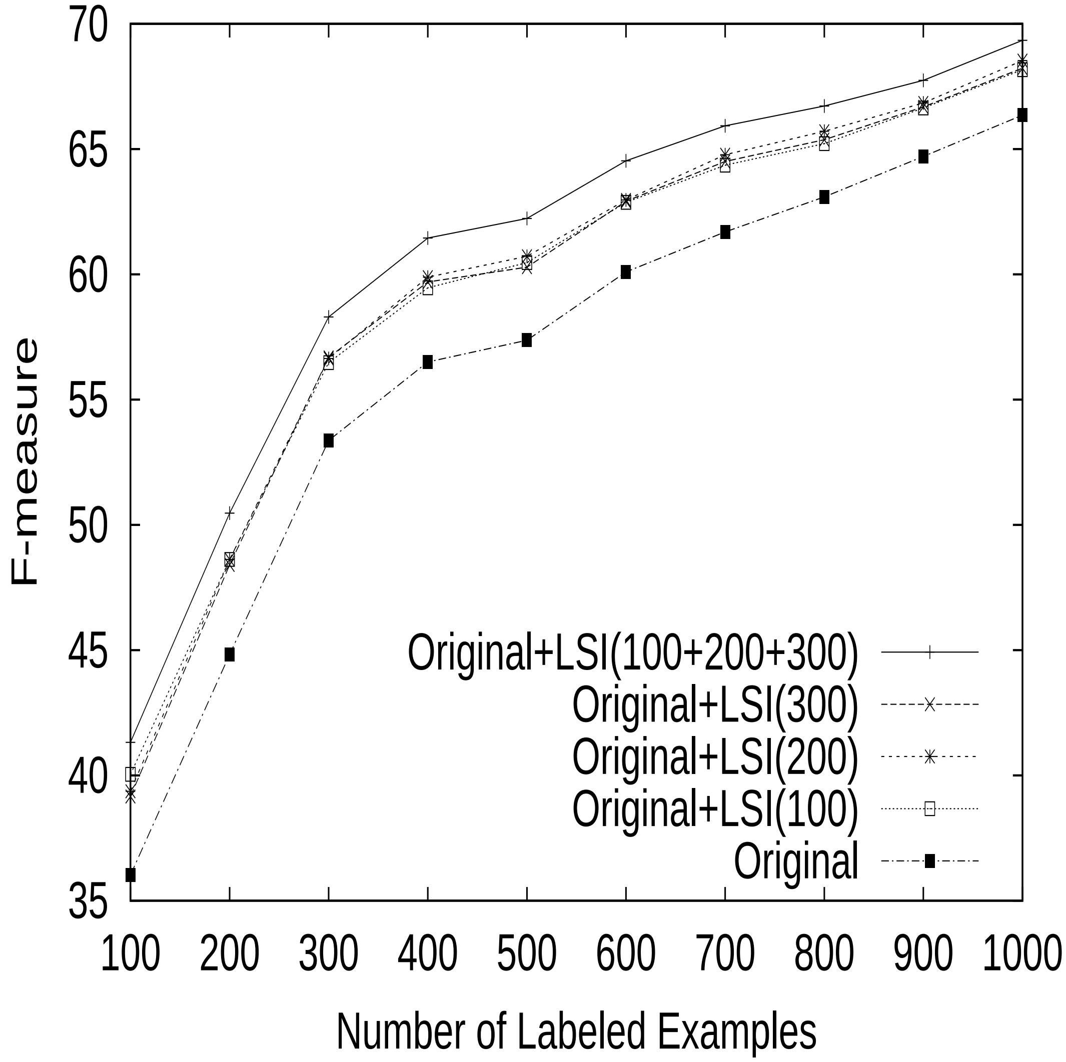

Figure 6.1. Training-data size and performance (LSI)

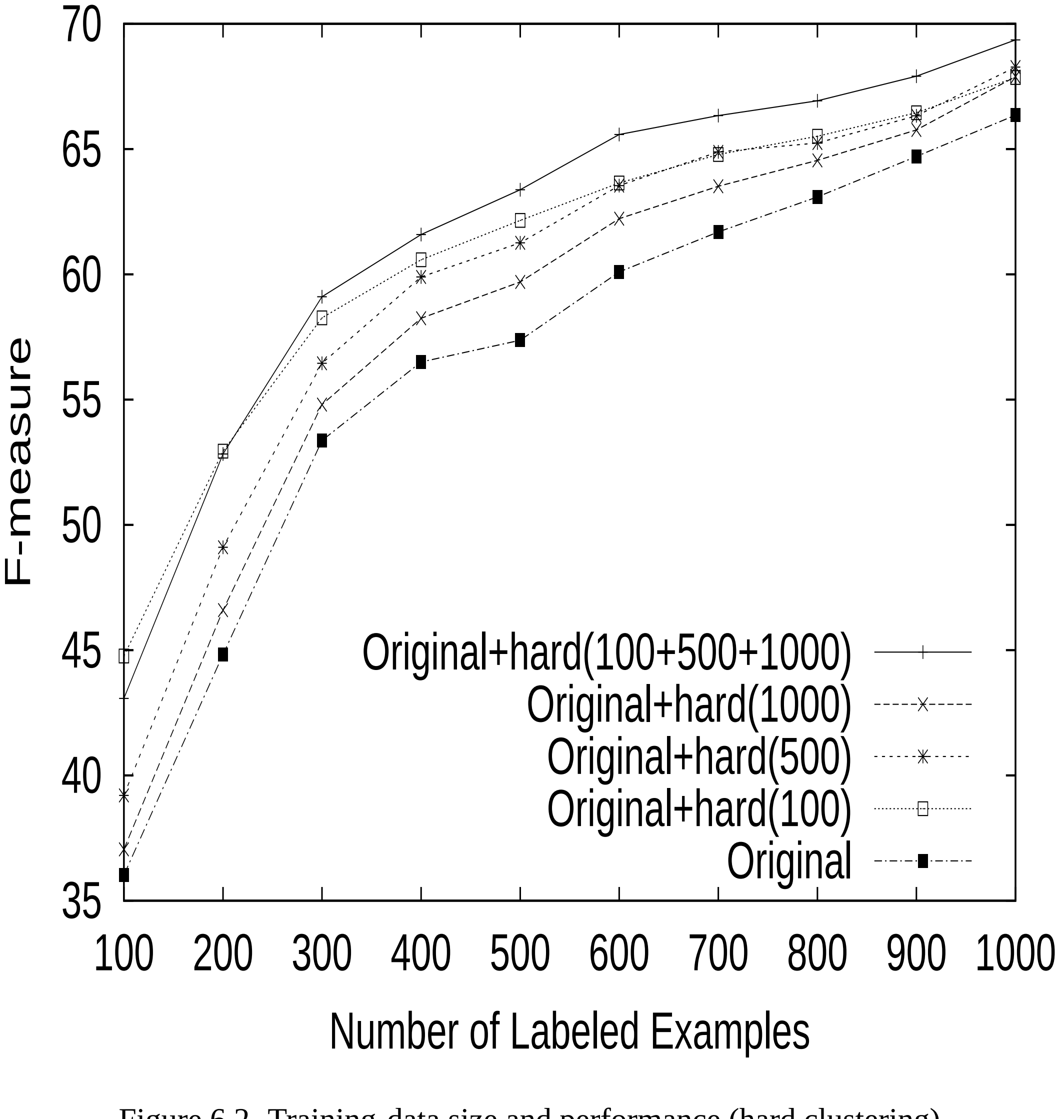

Figure 6.2. Training-data size and performance (hard clustering)

Table 6.4. The averaged numbers of positive examples

Category	Averaged number
earn	31.2
acq	16.3
money-fx	5.3
grain	4.3
crude	4.7
trade	4.5
interest	3.5
ship	2.2
wheat	2.9
corn	1.7

6.2, both cases yield the best F-measure when the three kinds of reduced features are added, regardless of the number of training examples. Moreover, all types of the proposed methods perform better than the original method. This experiment shows that our approach is effective at least in this range.

6.4.3 Performance for a Large Amount of Training Data

Although the proposed method is aimed at a small amount of training data, we test how the proposed method behaves when the number of training examples is large. Specifically, we tested the case of 4000 and 8000 training examples. Since the number of examples used for LSI and hard clustering is 8815, the case of 4000 training examples corresponds to the case where twice as large data is available for dimension reduction and the case of 8000 training examples corresponds to the case where little extra data exist.

The result is shown in Table 6.5 and Table 6.6. In the case of 4000 training examples, F-measures of the proposed methods are not so different from that of “Original”. In the case of 8000 training examples, “Original” mostly performs better than the proposed methods.

This experiment shows that the proposed method is effective only when a large

Table 6.5. F-measures (4000, 8000 labeled examples, LSI)

Method	Original	LSI			Original+LSI			
Dimension	–	100	200	300	100	200	300	100+200+300
4000	77.6	73.1	74.2	74.9	78.1	78.0	77.8	77.9
8000	80.1	76.3	77.6	77.1	80.2	79.9	79.8	79.7

Table 6.6. F-measures (4000, 8000 labeled examples, hard clustering)

Method	Original	hard			Original+hard			
Dimension	–	100	500	1000	100	500	1000	100+500+1000
4000	77.6	64.4	69.0	75.3	76.9	76.3	77.6	76.1
8000	80.1	76.3	77.6	77.1	80.2	79.9	79.8	77.0

amount of unlabeled data used for dimension reduction is available. In the next experiments, we will see this point in detail.

6.4.4 Performance for Various Sizes of Unlabeled Data

The experiments in the preceding sections suggest that the proposed method performs better than the original method for a small training set. The experiments in Section 6.4.3 show that the proposed method does not perform better for a large training set. However, on the basis of the discussion in Section 6.3, we conjecture that the cause of these results is not the number of training examples itself, but the amount of extra data used for dimension reduction. To confirm this conjecture, we observe the behavior of the proposed method when the number of examples used for dimension reduction changes.

In the next experiment, while the number of training examples is fixed as 1000, the number of examples used for dimension reduction ranges from 1000 to 8000, including the 1000 examples used for training (in other words, the number of extra examples is 0 to 7000). The results are shown in Figure 6.3 and Figure 6.4. These figures show that performance does not improve when there is no extra data available. In particular, performance of the method using hard clustering is worsened. However, performance increases as the number of extra examples increases. From this experiment and the preceding experiments, we conclude that the proposed method has the ability to im-

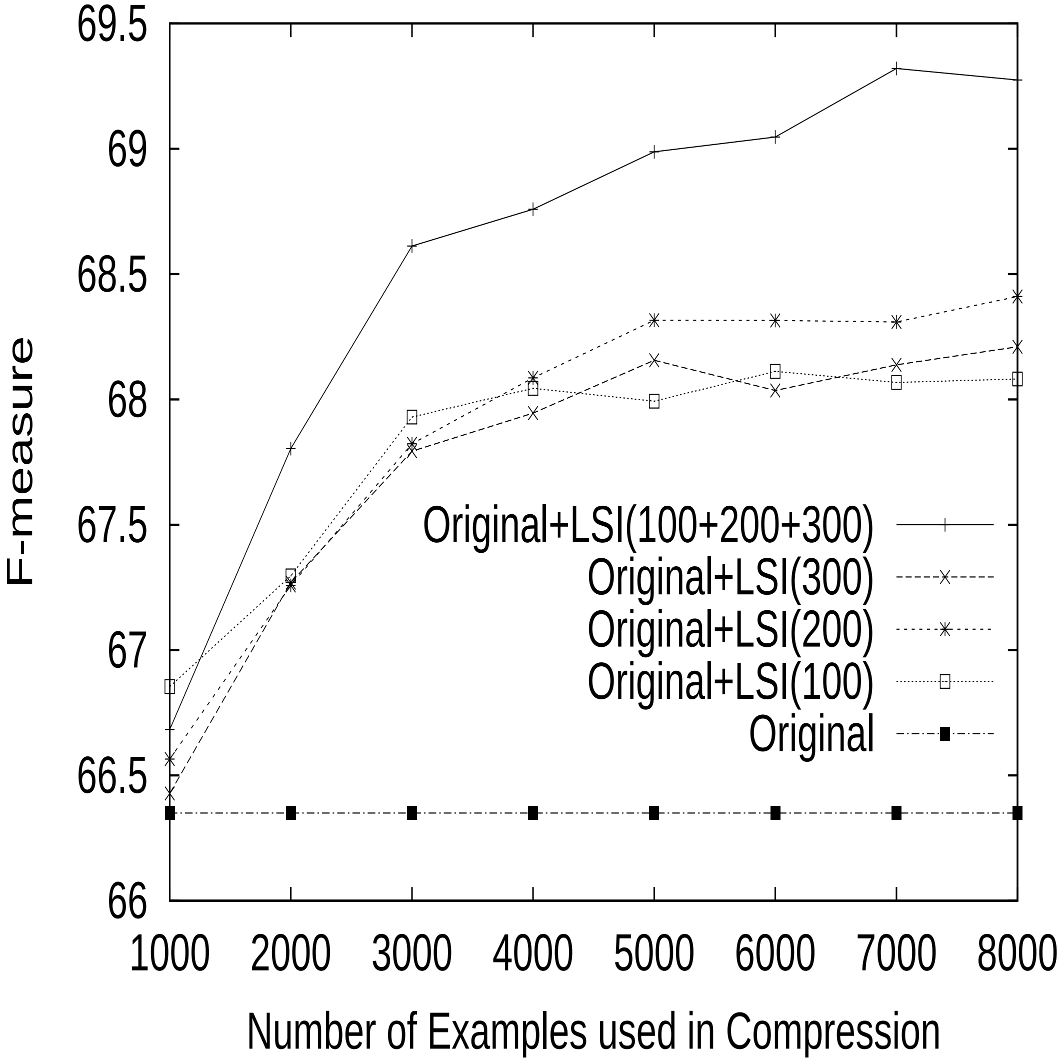

Figure 6.3. Data size used in compression and performance（1000 labeled examples, LSI）

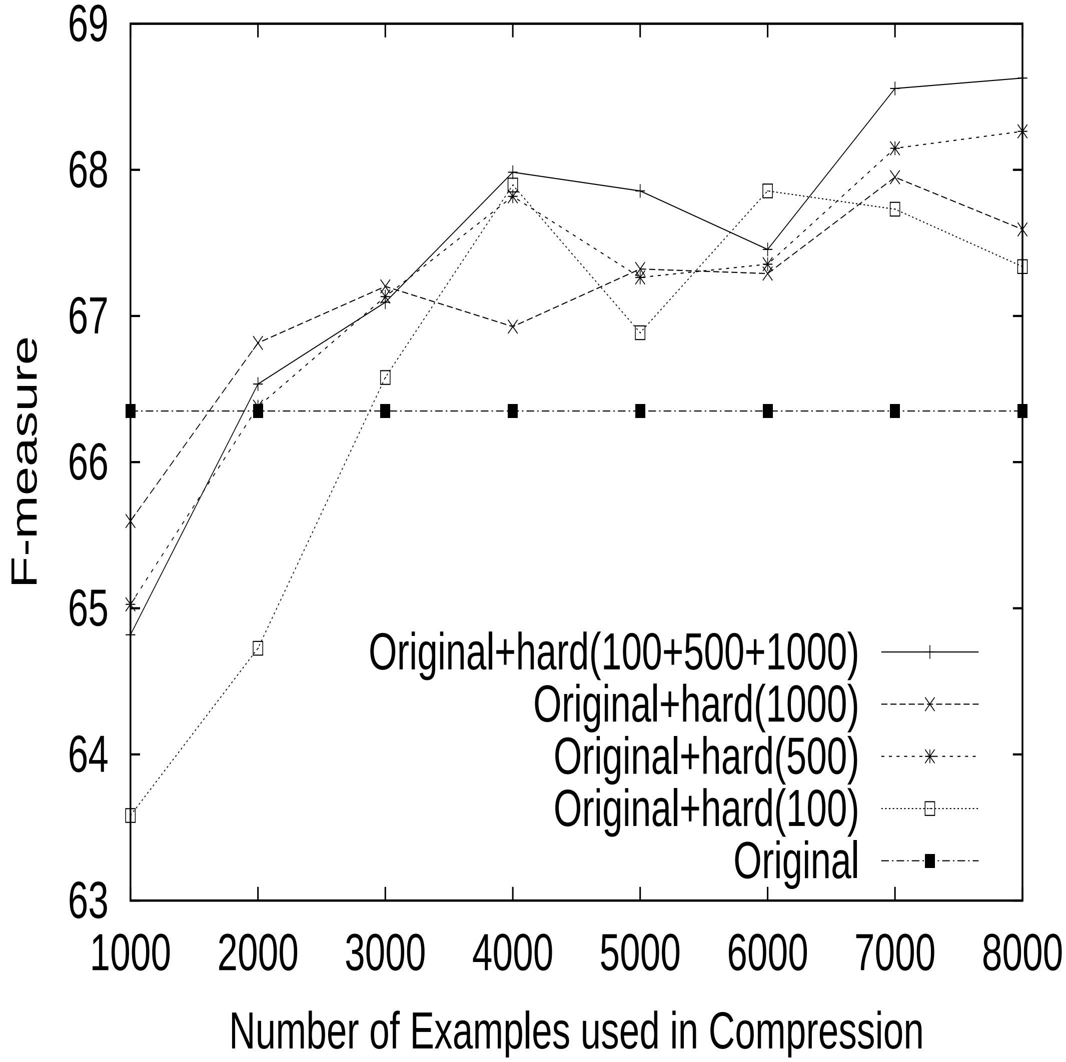

Figure 6.4. Data size used in compression and performance（1000 labeled examples, hard clustering）

prove the performance of SVMs when there exist a large number of examples used for dimension reduction (in our experiment, three times as many examples are needed).

6.4.5 Experiment using Another Data Set

We conduct a complementary experiment to test whether the proposed method works well for another data set. The new data set is 20-newsgroup described in Section 3.2.2. For Reuters-21578, the proposed method performs well when the number of examples for dimension reduction is about three times or more than the number of training examples. Therefore, we conduct an experiment under the condition that “the number of examples for dimension reduction is about three times or more than the number of training examples”. For dimension reduction, we use all the 9420 examples except for test examples, changing the number of training examples from 100 to 3000 (the above condition holds true in this range). We compare the proposed methods “Original+LSI (100+200+300)” and “Original+hard (100+500+1000)” with “Original”, because the two proposed methods were better than others. Other experimental settings are the same as that for the experiments in Section 6.4.2.

The result is shown in Figure 6.5. We can see that the proposed methods outperform “Original” regardless of the number of training examples. This result supports the conclusion that the proposed method has the ability to improve the performance when there exist a large number of extra examples.

6.5 Conclusion of the Chapter

We discussed text categorization with Support Vector Machines (SVMs) combined with the constructive induction using dimension reduction methods, such as Latent Semantic Indexing (LSI).

In the proposed method, the dimension of the original feature space is reduced with a dimension reduction method, then those reduced features are added to the feature space. The expanded new feature vectors are used as inputs to SVMs. We have empirically shown that the proposed method has the ability to improve the performance of SVMs when there exist a large number of extra examples and that using several reduced feature spaces with various dimensions simultaneously improves the

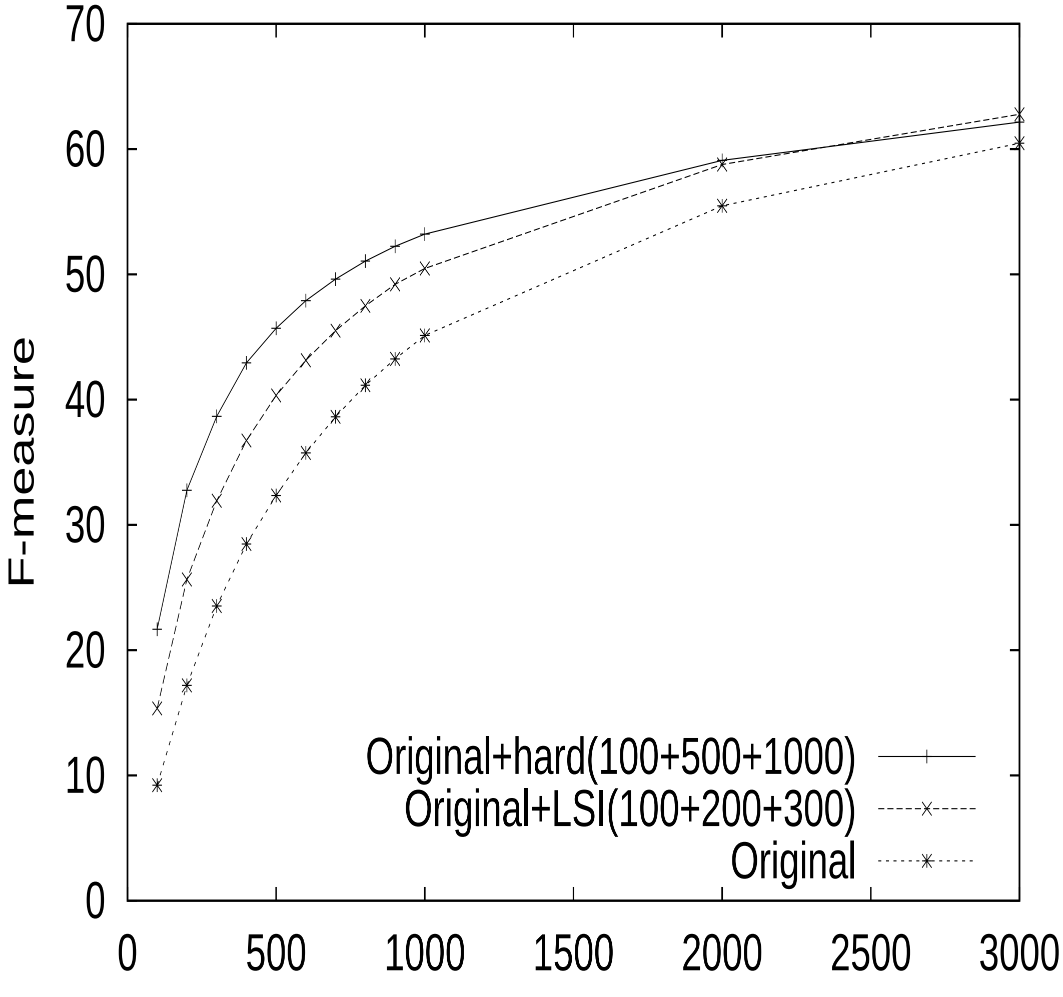

Figure 6.5. Performance for 20-newsgroup

performance.

Similar results are reported by Raskutti et al. [43], in which the similarity values to text clusters are added as a feature and the categorization performance is improved when a large number of unlabeled examples are available.

Although it is still unclear whether performance will be improved by using unlabeled data for dimension reduction when the number of training examples is increased up to around 10000, the discussion in Section 6.3 and the experiments in Section 6.4 suggest a good performance for large training data. The problem of how many unlabeled examples are needed to improve the performance for other data sets still remains. We regard it as future work and would like to address this problem by investigating it theoretically or conducting more experiments with other data sets.

Chapter 7

Application of the TOP Kernel

In this chapter, we discuss the application of TOP kernels to text categorization, which reportedly work well in categorization problems. First we discuss the clustering-based constructive induction in Chapter 6, from the viewpoint of the theory of TOP kernels. Next, we propose a TOP kernel based on the separating hyperplanes of SVMs.

7.1 LSI-based TOP Kernel

In Chapter 6, we saw that adding features extracted with clustering improves performance of SVMs. However, the theoretical underpinnings of the method are not clear. In this chapter, we first discuss the method used in Chapter 6 from the viewpoint of the theory of TOP kernels proposed by Tsuda [52] (explained in Section 2.3.2).

Suppose we have trained SVMs in the reduced feature space where an original feature vector **d** is expressed as $A\mathbf{d}$ and obtained the parameters **w** and b of the separating hyperplane. We first assume that the posterior probability of a positive class is of the form :

$$P(+1|\mathbf{d}) = \frac{1}{1+\exp(\alpha(\mathbf{w}\cdot A\mathbf{d}-b))}. \tag{7.1}$$

α is some (usually negative) constant. This function is used to map distances from the hyperplane to pseudo-probability [40].

The function $v(\mathbf{d},\alpha,\mathbf{w},b)$ used in TOP kernels is

$$v(\mathbf{d},\alpha,\mathbf{w},b) \quad = \quad \log P(+1|\mathbf{d}) - \log P(-1|\mathbf{d}) \tag{7.2}$$

$$
\begin{aligned}
&= \log P(+1|\mathbf{d}) - \log(1 - P(+1|\mathbf{d})) && (7.3)\\
&= \log \frac{P(+1|\mathbf{d})}{1 - P(+1|\mathbf{d})} && (7.4)\\
&= \log \frac{1}{P(+1|\mathbf{d})^{-1} - 1} && (7.5)\\
&= -\log(P(+1|\mathbf{d})^{-1} - 1) && (7.6)\\
&= -\log(1 + \exp(\alpha(\mathbf{w} \cdot A\mathbf{d} - b)) - 1) && (7.7)\\
&= -\alpha(\mathbf{w} \cdot A\mathbf{d} - b). && (7.8)
\end{aligned}
$$

By differentiating $v(\mathbf{d}, \alpha, \mathbf{w}, b)$ with respect to each parameter as follows:

$$
\begin{aligned}
\frac{\partial v(\mathbf{d}, \alpha, \mathbf{w}, b)}{\partial \alpha} &= \mathbf{w} \cdot A\mathbf{d} - b, && (7.9)\\
\frac{\partial v(\mathbf{d}, \alpha, \mathbf{w}, b)}{\partial \mathbf{w}} &= \alpha A\mathbf{d}, && (7.10)\\
\frac{\partial v(\mathbf{d}, \alpha, \mathbf{w}, b)}{\partial a_{cj}} &= -\alpha w_c d_j, && (7.11)\\
\frac{\partial v(\mathbf{d}, \alpha, \mathbf{w}, b)}{\partial b} &= \alpha, && (7.12)
\end{aligned}
$$

we obtain the TOP kernel $K(\mathbf{d}^1, \mathbf{d}^2)$ based on this probabilistic model ($\mathbf{d}^1$ and $\mathbf{d}^2$ denote document vectors)[1] :

$$
\begin{aligned}
K(\mathbf{d}^1, \mathbf{d}^2) &= (\mathbf{w} \cdot A\mathbf{d}^1 - b)(\mathbf{w} \cdot A\mathbf{d}^2 - b) && (7.13)\\
&\quad + \alpha^2 A\mathbf{d}^1 \cdot A\mathbf{d}^2 && (7.14)\\
&\quad + \alpha^2 (\sum_c w_c^2) \mathbf{d}^1 \cdot \mathbf{d}^2 && (7.15)\\
&\quad + \alpha^2 (\mathbf{w} \cdot A\mathbf{d}^1 - b)(\mathbf{w} \cdot A\mathbf{d}^2 - b). && (7.16)
\end{aligned}
$$

We should note that only the two terms (7.14) and (7.15) in the above expression are dot-products of vectors and the others terms are scalar products. Ignoring the terms of scalar products, we obtain $\alpha^2 A\mathbf{d}^1 \cdot A\mathbf{d}^2 + \alpha^2(\sum_c w_c^2)\mathbf{d}^1 \cdot \mathbf{d}^2$, which is similar to the expression (6.8) : $\mathbf{d}_1 \cdot \mathbf{d}_2 + M\mathbf{d}_1 \cdot M\mathbf{d}_2$. Thus, we have shown that the kernel function used in clustering-based constructive induction can be regarded as a type of TOP kernels.

[1] Here we do not use the notation $\mathbf{d}_1$ and $\mathbf{d}_2$, because in such a notation, the indexes for components are easily confused with the indexes for document instances.

TOP kernels of other models lead to a similar expression. The model discussed in this section was one example.

7.2 Hyperplane-based TOP Kernel

The basic idea in Chapter 5 is that one category may have multiple probability distributions. In this section, a similar idea is used to derive a TOP kernel. We focus on negative examples when performing binary classification. Negative examples in text categorization are usually more common than positive examples. There may be several different types of negative examples. Moreover, the categories of negative examples are sometimes explicitly given. In this situation, the probabilistic model of negative examples can be regarded as a mixture of several models. We make efficient use of this property. Although many models can be used, we propose a model based on the separating hyperplanes of categories.

Suppose we have obtained the parameters $\mathbf{w}_c$ and b_c of the separating hyperplane for each category c in the original feature space. We assume that the generative model $P(\mathbf{d}|c)$ of a vector $\mathbf{d}$ given c is

$$\begin{aligned} P(\mathbf{d}|c) &= \lambda q(\mathbf{d}|c) & (7.17)\\ &= \frac{\lambda}{\sqrt{2\pi\sigma_c^2}}\exp\{-\frac{|(\mathbf{w}_c\cdot\mathbf{d}-b_c)-\mu_c|^2}{2\sigma_c^2}\} & (7.18)\end{aligned}$$

with the mean μ_c of a random variable $(\mathbf{w}_c\cdot\mathbf{d}-b_c)$ and the variance σ_c, and a constant λ. Strictly speaking, this λ is not a constant. For the model to be valid as a probability distribution, λ must be 0 in the infinity. However, by assuming λ is a local constant in a neighborhood large enough to enclose all the feature vectors, we can treat λ as a constant. In addition, we assume that λ is statistically independent of categories.

Indeed the assumption that λ is a constant is unlikely to be true, but our purpose is to extract useful feature vectors. So here we ignore this impreciseness.

In the following, c is the positive category among many categories, e is used to denote a category which is not c, and c' is used to denote a category which can be either c or e.

$$v(\mathbf{d},\alpha,\mathbf{w},b) = \log P(+1|\mathbf{d}) - \log P(-1|\mathbf{d})$$

$$
\begin{aligned}
&= \log \frac{P(c)\lambda q(\mathbf{d}|c)}{\sum_{c'} P(c')\lambda q(\mathbf{d}|c')} - \log \frac{\sum_{e\neq c} P(e)\lambda q(\mathbf{d}|e)}{\sum_{c'} P(c')\lambda q(\mathbf{d}|c')} \\
&= \log P(c)q(\mathbf{d}|c) - \log \sum_{e\neq c} P(e)q(\mathbf{d}|e) \\
&= \log P(c)\exp\{\theta_{c1}(\mathbf{w}_c\cdot\mathbf{d}) + \theta_{c2}(\mathbf{w}_c\cdot\mathbf{d})^2 + \frac{\theta_{c1}^2}{4\theta_{c2}} - \frac{1}{2}\log\frac{-\pi}{c2}\}\ \theta \\
&\quad - \log \sum_{e\neq c} P(e)\exp\{\theta_{e1}(\mathbf{w}_e\cdot\mathbf{d}) + \theta_{e2}(\mathbf{w}_e\cdot\mathbf{d})^2 + \frac{\theta_{e1}^2}{4\theta_{e2}} - \frac{1}{2}\log\frac{-\pi}{e2}\},\ \theta
\end{aligned}
\tag{7.19}
$$

where $\theta_{c'1} = \mu'_c/\sigma'^2_c, \theta_{c'2} = 1/2\sigma'^2_c$. We should note that $\theta_{c'1}$ and $\theta_{c'2}$ are not the natural parameters of this model. However, we parameterize this model using the parameters $\theta_{c'1}$, $\theta_{c'2}$, $\mathbf{w}_{c'}$, $b_{c'}$ and $P(c')$ for simplicity.

The partial derivatives of this function with respect to the parameters are the following :

$$\frac{\partial v(\mathbf{d},\theta)}{\partial\theta_{c1}} = \mathbf{w}_c\cdot\mathbf{d} - b_c - \mu_c, \tag{7.20}$$

$$\frac{\partial v(\mathbf{d},\theta)}{\partial\theta_{e1}} = -\frac{P(e)q(\mathbf{d}|e)}{\sum_{c'\neq c} P(c')q(\mathbf{d}|c')}(\mathbf{w}_e\cdot\mathbf{d} - b_e - \mu_e), \tag{7.21}$$

$$\frac{\partial v(\mathbf{d},\theta)}{\partial\theta_{c2}} = (\mathbf{w}_c\cdot\mathbf{d} - b_c)^2 - \mu_c^2 - \sigma_c^2, \tag{7.22}$$

$$\frac{\partial v(\mathbf{d},\theta)}{\partial\theta_{e2}} = \frac{P(e)q(\mathbf{d}|e)}{\sum_{c'\neq c} P(c')q(\mathbf{d}|c')}\{(\mathbf{w}_e\cdot\mathbf{d} - b_e)^2 - \mu_e^2 - \sigma_e^2\}, \tag{7.23}$$

$$\frac{\partial v(\mathbf{d},\theta)}{\partial w_{ci}} = \frac{\mu_c - (\mathbf{w}_c\cdot\mathbf{d} - b_c)}{\sigma_c^2}d_i, \tag{7.24}$$

$$\frac{\partial v(\mathbf{d},\theta)}{\partial w_{ei}} = \frac{P(e)q(\mathbf{d}|e)}{\sum_{c'\neq c} P(c')q(\mathbf{d}|c')}\frac{\mu_e - (\mathbf{w}_e\cdot\mathbf{d} - b_e)}{\sigma_e^2}d_i, \tag{7.25}$$

$$\frac{\partial v(\mathbf{d},\theta)}{\partial b_c} = \frac{\mathbf{w}_c\cdot\mathbf{d} - b_c - \mu_c}{\sigma_c^2}, \tag{7.26}$$

$$\frac{\partial v(\mathbf{d},\theta)}{\partial b_e} = \frac{\mathbf{w}_e\cdot\mathbf{d} - b_e - \mu_e}{\sigma_e^2}, \tag{7.27}$$

$$\frac{\partial v(\mathbf{d},\theta)}{P(c)} = \frac{1}{P(c)}, \tag{7.28}$$

$$\frac{\partial v(\mathbf{d},\theta)}{P(e)} = \frac{P(\mathbf{d}|e)}{\sum_{c'\neq c} P(c')q(\mathbf{d}|c')}. \tag{7.29}$$

Then we can follow the definition (2.41) to obtain our version of the TOP kernel. We call this new kernel a *hyperplane-based TOP (HP-TOP)* kernel. Computing the kernel in this form is time-consuming because the number of components of type (7.25) can be very large : (vocabulary size)×(number of categories). We can avoid this heavy computational cost as follows. Let us compute the dot-product of derivatives (7.25) of two vectors $\mathbf{d}^1$ and $\mathbf{d}^2$:

$$\begin{aligned}
& \sum_{e \neq c} \sum_{i} \frac{\partial v(\mathbf{d}^1, \theta)}{\partial w_{ei}} \frac{\partial v(\mathbf{d}^2, \theta)}{\partial w_{ei}} && (7.30)\\
= & \sum_{e \neq c} \sum_{i} \frac{P(e)^2 q(\mathbf{d}^1|e) q(\mathbf{d}^2|e)}{P_{-c}(\mathbf{d}^1) P_{-c}(\mathbf{d}^2)} \frac{\mu_e - (\mathbf{w}_e \cdot \mathbf{d} - b_e)}{\sigma_e^2} \frac{\mu_e - (\mathbf{w}_e \cdot \mathbf{d} - b_e)}{\sigma_e^2} d_i^1 d_i^2 && (7.31)\\
= & \Big(\sum_{e \neq c} \frac{P(e)^2 q(\mathbf{d}^1|e) q(\mathbf{d}^2|e)}{P_{-c}(\mathbf{d}^1) P_{-c}(\mathbf{d}^2)} \frac{\mu_e - (\mathbf{w}_e \cdot \mathbf{d} - b_e)}{\sigma_e^2} \frac{\mu_e - (\mathbf{w}_e \cdot \mathbf{d} - b_e)}{\sigma_e^2} \Big) \mathbf{d}^1 \cdot \mathbf{d}^2, && (7.32)
\end{aligned}$$

where $P_{-c}(\mathbf{d})$ denotes $\sum_{c' \neq c} P(c') q(\mathbf{d}|c')$. The last expression (7.32) is regarded as the scalar product of two dot-products. Thus, by preserving vectors $\mathbf{d}$ and

$$\left(\frac{P(e) q(\mathbf{d}|e)}{P_{-c}(\mathbf{d})} \frac{\mu_e - (\mathbf{w}_e \cdot \mathbf{d} - b_e)}{\sigma_e^2} \right)_{e \neq c}, \qquad (7.33)$$

we can efficiently compute the dot-product in (7.25); the computational complexity of a kernel function is of the order of vocabulary size (on the condition that the vocabulary size is larger than the number of categories). Thus, from the viewpoint of computational time, our kernel has an advantage over some other kernels such as the PLSI-based Fisher kernel in Section 4.7, which needs the computational complexity of the order of (vocabulary size)×(number of clusters).

According to expression (7.32), two vectors are considered to be more similar, if they have similar distributions over categories. In the PLSI-based Fisher kernel, each word has a probability distribution over latent classes. In this sense, the PLSI-based Fisher kernel is more detailed, but simultaneously has the computational disadvantage mentioned above.

7.3 Experiments

We empirically compare the HP-TOP kernel with the linear kernel and the PLSI-based Fisher kernel. We use Reuters-21578 data set with ModApte-split (Section 3.2.1).

The size of the training data ranges from 1000 to 8000. For each size of the data set, experiments are executed 10 times with randomly selected training examples.

The result is evaluated using F-measures for the 10 most frequent categories (Table 3.1).

The total number of categories is actually 116. However, for small categories, reliable statistics cannot be obtained. For this reason, we regard the remaining categories other than the 10 most frequent categories as one category. Therefore, the model for negative examples is a mixture of 10 models (9 out of the 10 most frequent categories and the new category consisting of the remaining categories).

We assume uniform priors for categories because computing category priors from the given data set can lead to poor estimates [52].

We computed the Fisher kernels with different numbers (10, 20 and 30) of latent classes and added them together to make a robust kernel [24].

The result is shown in Figure 7.1. The HP-TOP kernel outperforms the linear kernel and the PLSI-based Fisher kernel for all numbers of examples. At each number of examples, we conducted a Wilcoxon Signed Rank test with 5% significance-level, for the HP-TOP kernel and the linear kernel, since these two are better than the other. The test shows that the difference between the two methods is significant for the data sizes 1000 to 5000.

In this experimental setting, the PLSI-based Fisher kernel did not work well. However, this Fisher kernel will perform better when the number of labeled examples is small and a number of unlabeled examples are available, as reported by Hofmann[24].

7.4 Conclusion of the Chapter

In this chapter, we first gave an interpretation of the method discussed in Chapter 6 from the theory of the TOP kernel. Then we proposed a TOP kernel based on separating hyperplanes. The proposed kernel is created from one-dimensional Gaussians along the normal directions of the hyperplanes. We have empirically shown that the

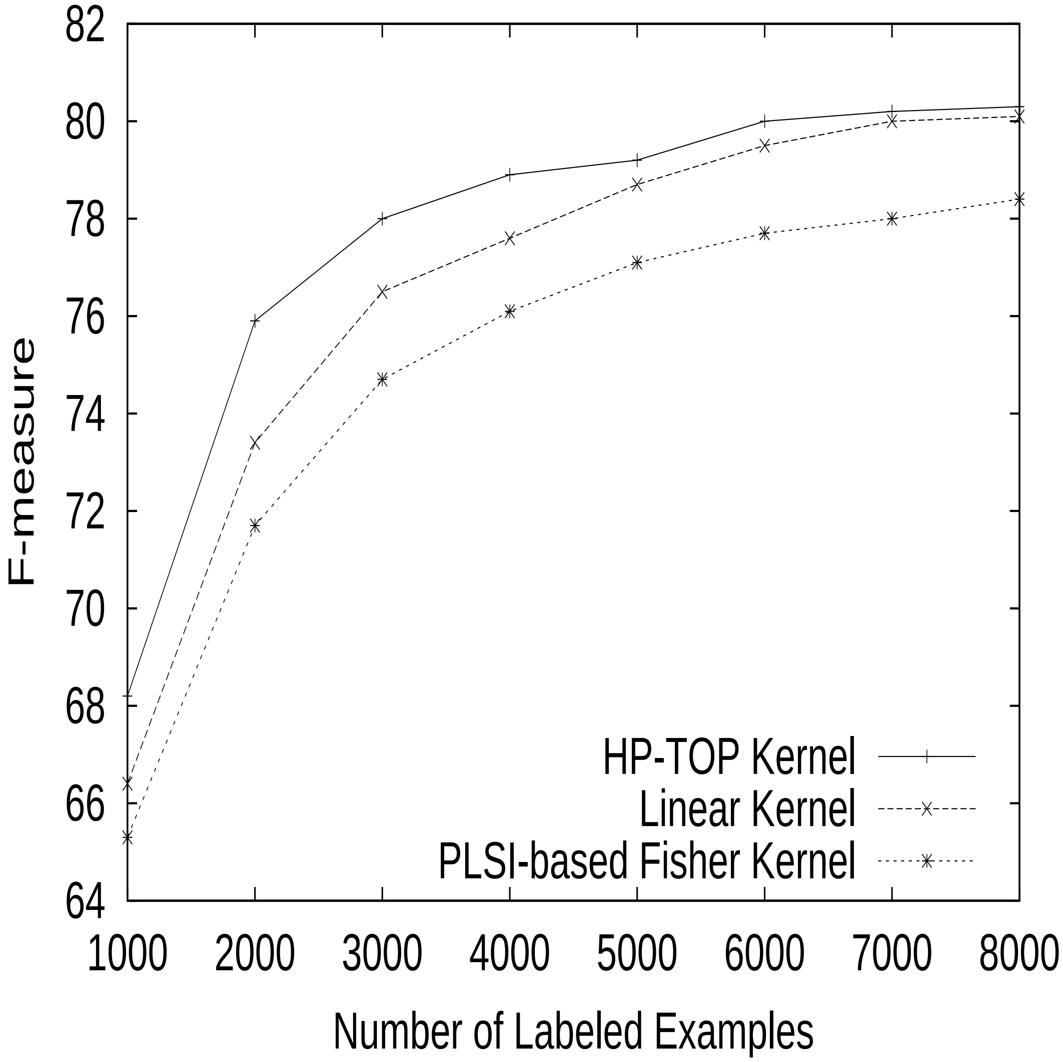

Figure 7.1. Performance of Hyperplane-based TOP kernel

proposed kernel outperforms the linear kernel in text categorization.

Although the superiority of the proposed method to the linear kernel was shown, the proposed method has to be further investigated. Firstly, for large data sizes (namely 7000 and 8000), the proposed method was not significantly better than the linear kernel. The effectiveness of the proposed method should be confirmed by more experiments and theoretical analysis. Secondly, we have to compare the proposed method with other kernels in order to show the effectiveness of the kernel function consisting of one-dimensional Gaussians normal to the hyperplanes.

This model can be extended to incorporate unlabeled examples, for example using the EM algorithm. When the category structure of the negative examples is not given, the proposed method is not applicable. We should investigate whether unsupervised clustering can substitute for the category structure.

Chapter 8

Conclusion

In this thesis, two clustering approaches and an application of kernel methods to text categorization were studied.

In Chapter 4.2, we introduced machine learning methods used in this thesis. In Chapter 3, several properties of text categorization were described. In particular, we focused on the sparsity of text data.

We proposed a new method to improve the accuracy of text categorization using co-clustering in Chapter 5. In this method, both training texts and features are clustered before estimating the probabilistic model. The development of this method was motivated by the fact that one category may not be expressed by one probabilistic model. We have empirically shown that co-clustering outperforms class-distributional clustering and word clustering. Co-clustering improves the performance of Naive Bayes classifiers, although the effectiveness of the word clustering phase for SVMs is not clear yet.

In Chapter 6, we discussed text categorization with Support Vector Machines combined with the constructive induction using dimension reduction methods, such as Latent Semantic Indexing. The method improves the performance of SVMs when the dimension reduction method can use unlabeled examples to extract new features.

In Chapter 7, we applied a TOP kernel to text categorization on the basis of the probability distributions along the normal lines of the separating hyperplanes. A simple experiment has shown that this hyperplane-based TOP kernel outperforms the linear kernel.

One interesting point for future work is the incorporation of unlabeled data. Although we used unlabeled data in the constructive induction approach in Chapter 6, the framework of co-clustering is based on a supervised clustering technique. The hyperplane-based TOP kernel also uses only labeled data. These supervised approaches can be enhanced with the use of unlabeled data.

References

[1] Hirotsugu Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974.

[2] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.

[3] L. Douglas Baker and Andrew Kachites McCallum. Distributional clustering of words for text classification. In *Proceedings of the Twenty-first ACM International Conference on Research and Development in Information Retrieval (SIGIR98)*, pages 96–103, 1998.

[4] Ron Bekkerman, Ran El-Yaniv, Naftali Tishby, and Yoad Winter. On feature distributional clustering for text categorization. In *Proceedings of SIGIR-01, 24th ACM International Conference on Research and Development in Information Retrieval*, pages 146–153, 2001.

[5] Eric Bloedorn and Ryszard S. Michalski. Data-driven constructive induction. *IEEE Intelligent Systems*, 13(2):30–37, 1998.

[6] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[7] Michael Collins and Nigel Duffy. Convolution kernels for natural language. In *Advances in Neural Information Processing Systems 14 (NIPS 2001)*, volume 1, pages 625–632, 2001.

[8] Robert H. Creecy, Brij M. Masand, Stephen J. Smith, and David L. Waltz. Trading mips and memory for knowledge engineering: Classifying census returns on the connection machine. *Communications of the ACM*, 35(8):48–63, 1992.

[9] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines (and other kernel-based learning methods)*. Cambridge University Press, 2000.

[10] Nello Cristianini, John Shawe-Taylor, and Huma Lodhi. Latent semantic kernels. In Carla Brodley and Andrea Danyluk, editors, *Proceedings of 18th International*

Conference on Machine Learning (ICML-2001), pages 66–73, Williams College, US, 2001. Morgan Kaufmann Publishers, San Francisco, US.

[11] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *Journal of the Society for Information Science*, 41(6):391–497, 1990.

[12] Morris H. DeGroot. *Probability and Statistics*. Addison-Wesley, 1986.

[13] Inderjit S. Dhillon, Subramanyam Mallela, and Rahul Kumar. A divisive information-theoretic feature clustering algorithm for text classification (to appear). *Journal of Machine Learning Research (JMLR)*, 2002.

[14] Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

[15] Thomas G. Dietterich and Ryszard S. Michalski. A comparative review of selected methods for learning from examples. In Ryszard S. Michalski, Jaime G. Carbonell, and Tom M. Mitchell, editors, *Machine Learning *An Artificial Intelligence Approach*, pages 41–82. Morgan Kaufmann, 1983.

[16] Pedro Domingos and Michael Pazzani. Beyond independence: Conditions for the optimality of the simple bayesian classifier. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 105–112, 1996.

[17] Susan T. Dumais, John Platt, David Heckerman, and Mehran Sahami. Inductive learning algorithms and representations for text categorization. In *Proceedings of the Seventh International Conference on Information and Knowledge Management (ACM-CIKM98)*, pages 148–155, 1998.

[18] Ulrich Kreßel. Pairwise classication and support vector machines. In Bernhard Schölkopf, Christopher J. C. Burgesa, and Alexander J. Smola, editors, *Advances in Kernel Methods *Support Vector Learning*, pages 255–268. The MIT Press, 1999.

[19] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[20] Norbert Fuhr and Chris Buckley. A probabilistic learning approach for document indexing. *ACM Transactions on Information Systems*, 9(3):223–248, 1991.

[21] Glenn Fung and Olvi L. Mangasarian. Semi-supervised support vector machines for unlabeled data classification. *Optimization Methods and Software*, 15:29–44, 2001.

[22] David Haussler. Convolution kernels on discrete structures. Technical Report UCS-CRL-99-10, UC Santa Cruz, 1999.

[23] Thomas Hofmann. Probabilistic Latent Semantic Indexing. In *Proceedings of the 22nd Annual ACM Conference on Research and Development in Information Retrieval*, pages 50–57, Berkeley, California, August 1999.

[24] Thomas Hofmann. Learning the similarity of documents: An information geometric approach to document retrieval and categorization. In *Advances in Neural Information Processing Systems, 12*, pages 914–920, 2000.

[25] Tommi Jaakkola and David Haussler. Exploiting generative models in discriminative classifiers, 1998.

[26] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the European Conference on Machine Learning*, pages 137–142, 1998.

[27] Thorsten Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of 16th International Conference on Machine Learning (ICML '99)*, pages 200–209, 1999.

[28] Robert E. Kass and Paul W. Vos. *Geometrical foundations of asymptotic inference*. New York : Wiley, 1997.

[29] Taku Kudo and Yuji Matsumoto. Chunking with support vector machines. In *Proceedings of Second Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL 2001)*, pages 192–199, 2001.

[30] Taku Kudo and Yuji Matsumoto. Japanese dependency analysis using cascaded chunking. In *Proceedings of the 6th Conference on Natural Language Learning 2002 (CoNLL 2002)*, pages 63–69, 2002.

[31] David D. Lewis and Marc Ringuette. A comparison of two learning algorithms for text categorization. In *Proceedings of the 3rd Annual Symposium on Document Analysis and Information Retrieval (SDAIR-94)*, pages 81–93, Las Vegas, US, 1994.

[32] Hang Li. Word clustering and disambiguation based on co-occurrence data. *Natural Language Engineering*, 8(1):25–42, 2002.

[33] Hang Li and Naoki Abe. Word clustering and disambiguation based on co-occurrence data. In *Proceedings of the 36th Annual Meeting of the ACL and the 17th International Conference on Computational Linguistics (COLING-ACL '98)*, pages 749–755, 1998.

[34] Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Christopher J. C. H. Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444, 2002.

[35] Andrew McCallum and Kamal Nigam. A comparison of event models for naive bayes text classification. In *Proceedings of AAAI-98 Workshop on Learning for Text Categorization*, pages 41–48, 1998.

[36] Tom M. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[37] Tetsuji Nakagawa, Taku Kudo, and Yuji Matsumoto. Revision learning and its application to part-of-speech tagging. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002)*, pages 497–504, 2002.

[38] Kamal Nigam, Andrew Mccallum, Sebastian Thrun, and Tom Mitchell. Text classification from labeled and unlabeled documents using em. *Machine Learning*, 39(2/3):103–134, 2000.

[39] Giulia Pagallo and David Haussler. Boolean feature discovery in empirical learning. *Machine Learning*, 5(1):71–99, 1990.

[40] John Platt. Probabilities for sv machines. In A.J. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classiers*, pages 61–74. MIT Press, 1999.

[41] Massimiliano Pontil and Alessandro Verri. Support vector machines for 3D object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(6):637–646, 1998.

[42] Lubomir Popelinsky and Pavel Brazdil. The principal components method as a pre-processing stage for decision tree learning. In *Proceedings of Principles and Practice of Knowledge Discover in Databases, Workshop on Metalearning*, 2000.

[43] Bhavani Raskutti, Herman L. Ferrá, and Adam Kowalczyk. Using unlabeled data for text classification through addition of cluster parameters. In *Proceedings of the 19th International Conference on Machine Learning (ICML-02)*, pages 514–521, 2002.

[44] Jorma Rissanen. Stochastic complexity. *Journal of Royal Statistical Society, Series B*, 49(3):223–239, 1987.

[45] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill Book Company, New York, 1983.

[46] Robert E. Schapire. A brief introduction to boosting. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 1401–1405, 1999.

[47] Helmut Schmid. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of International Conference on New Methods in Language Processing*, pages 44–49, 1994.

[48] Georges Siolas and Florence d'Alché-Buc. Support vectors machines based on a semantic kernel for text categorization. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN-2000)*, volume V, pages 205–209, 2000.

[49] Hirotoshi Taira and Masahiko Haruno. Feature selection in SVM text categorization. In *Proceedings of the 16th National Conference on Artificial Intelligence / Eleventh Conference on Innovative Applications of Artificial Intelligence (AAAI-99/IAAI-99)*, pages 480–486, 1999.

[50] Chris Thornton. What do constructive learners really learn? *Artificial Intelligence Review*, 13(4):249–257, 1999.

[51] Koji Tsuda and Motoaki Kawanabe. The leave-one-out kernel. In *Proceedings of International Conference on Artificial Neural Networks*, pages 727–732, 2002.

[52] Koji Tsuda, Motoaki Kawanabe, Gunnar Rätsch, Sören Sonnenburg, and Klaus-Robert Müller. A new discriminative kernel from probabilistic models. *Neural Computation*, 14(10):2397–2414, 2002.

[53] Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag Berlin, 1995.

[54] Vladimir Vapnik. *Statistical Learning Theory*. John Wiley, New York, 1998.

[55] Jason Weston, Sayan Mukherjee, Olivier Chapelle, Massimiliano Pontil, Tomaso Poggio, and Vladimir Vapnik. Feature selection for svms. In *Advances in Neural Information Processing Systems 13 (NIPS-2000)*, pages 668–674, 2000.

[56] Hiroyasu Yamada and Yuji Matsumoto. Japanese named entity extraction using support vector machine. *Transactions of Information Processing Society Japan*, 43(1):44–53, 2002.

[57] Yiming Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1(1/2):69–90, 1999.

[58] Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In Douglas H. Fisher, editor, *Proceedings of the 14th International Conference on Machine Learning (ICML-97)*, pages 412–420, Nashville, US, 1997. Morgan Kaufmann Publishers, San Francisco, US.

Appendix

A Mathematical Foundations

A.1 Maximum Likelihood Estimation

Suppose we want to estimate the parameters θ of a model $p(x|\theta)$, given a set of data samples $\{x_1, \cdots x_n\}$. The log-likelihood function $L(\{x_1, \cdots x_n\}|\theta)$ is defined as

$$\begin{aligned} L(\{x_1, \cdots x_n\}|\theta) &\equiv \log \prod_{i=1}^{n} p(x|\theta) \\ &= \sum_{i=1}^{n} \log p(x|\theta) \qquad (8.1) \end{aligned}$$

In the maximum likelihood estimation, the parameters maximizing (8.1) are chosen.

Example A.1 *Suppose $p(x|\theta)$ is a gaussian with unknown parameters μ and σ^2. Then, (8.1) is*

$$\begin{aligned} L(\{x_1, \cdots x_n\}|\theta) &= \sum_{i=1}^{n} \log p(x|\theta) \\ &= \sum_{i=1}^{n} \left(-\frac{(x_i-\mu)^2}{2\sigma^2} + \log \frac{1}{\sqrt{2\pi\sigma^2}} \right). \qquad (8.2) \end{aligned}$$

Its partial derivative with respect to μ is

$$\frac{\partial L(\{x_1, \cdots x_n\}|\theta)}{\partial \mu} = \sum_{i=1}^{n} -\frac{x_i - \mu}{2}. \qquad \sigma \qquad (8.3)$$

From the stationary condition stipulating that (8.3) is zero, the maximum likelihood estimator of μ is obtained as follows :

$$\begin{aligned} & \textstyle\sum_{i=1}^{n} -\frac{x_i-\mu}{2} = 0 \qquad \sigma && (8.4) \\ \Rightarrow \quad & \textstyle\sum_{i=1}^{n} (x_i - \mu) = 0 && (8.5) \\ \Rightarrow \quad & \textstyle\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} \mu && (8.6) \\ \Rightarrow \quad & \mu = \frac{\sum_{i=1}^{n} x_i}{n}. && (8.7) \end{aligned}$$

A.2 Kullback-Leibler Divergence

Kullback-Leibler (KL) divergence (or distance) for the probability distributions $P(x)$ and $Q(x)$ is, in the discrete case, defined as

$$KL(P||Q) = \sum_x Q(x) \log \frac{Q(x)}{P(x)}. \tag{8.8}$$

This quantity is used as a *distance* between two probability distributions, though it does not satisfy the conditions for distance.

A.3 Exponential Family

Definition A.1 *The exponential family of distributions is defined as the set of the probability distributions whose density $p(x|\theta)$ can be expressed in the following canonical form :*

$$p(\boldsymbol{x}|\theta) = \exp\{T(\boldsymbol{x})^t\theta - \psi(\theta) + C(\boldsymbol{x})\}. \tag{8.9}$$

The parameters θ are called natural parameters. T, ψ, C are some continuous functions.

The exponential family includes, for example, :

- Binomial distribution.

$$\begin{aligned} p(x|\theta) &= \binom{n}{x} p^x(1-p)^{1-x} && (8.10) \\ &= \exp\left(\log(\frac{p}{1-p})x + n\log(1-p) + \log\binom{n}{x}\right) && (8.11) \end{aligned}$$

$T(x) = x, \theta = \log(\frac{p}{1-p}), \cdots$

- Gaussian

$$\begin{aligned} p(x|\theta) &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{|x-\mu|^2}{2\sigma^2}\right) && (8.12) \\ &= \exp\left(\frac{\mu}{2}x - \frac{1}{2\sigma^2}x^2 - \frac{\mu^2}{2\sigma^2} \underset{\sigma}{+} \log\frac{1}{\sqrt{2\pi\sigma^2}}\right) && (8.13) \end{aligned}$$

$T_1(x) = x, T_2(x) = x^2, \theta_1 = \frac{\mu}{2}, \theta_2 = \frac{1}{2\sigma^2}, \cdots$ σ

B Examples of Clusters

- deliver [VB], handle [VB], charter [VB], destine [VB], halve [VB], mill [VB], feature [VB], date [VB], earmark [VB], reallocate [VB], transact [VB], switch [VB], subsidize [VB]
- export [NN], exporter [NN]
- go [VB], get [VB], do [VB]
- port [NN], ship [NN], vessel [NN], harbour [NN], dwt [NN], liner [NN], yard [NN], charterer [NN], mt [NN]
- increased [JJ], significant [JJ], positive [JJ], overall [JJ], dramatic [JJ], responsible [JJ]
- loan [NP], home [NP], savings [NP]
- apr [NP], mar [NP], feb [NP]
- many [JJ], large [JJ], major [JJ]
- likely [JJ], unlikely [JJ]
- unemployment [NN], employment [NN]
- december [NP], november [NP]
- milk [NN], feeding [NN], convention [NN], livestock [NN], marketings [NN], herd [NN], lock-out [NN], meatpackers [NN], pork [NN], meat [NN], beef [NN], cattle [NN]
- sulphur [NN], fuel [NN], gasoline [NN]
- yemen [NP], guarantee [NP], weekly [NP], egypt [NP], bangladesh [NP], morocco [NP], deputy [NP], poland [NP], sudan [NP]
- electrical [JJ], swedish [JJ], fine [JJ], sophisticated [JJ], compatible [JJ], electronic [JJ], plastic [JJ], 12th [JJ], video [JJ], yearly [JJ], advanced [JJ], technological [JJ]

- copper [NP], smelting [NP], comex [NP], lme [NP], aluminium [NP], metal [NP], alcoa [NP], alcan [NP], aluminum [NP]
- hamburg [NP], chinese [NP], alfred [NP], lord [NP], french [NP], austria [NP], italy [NP], belgium [NP]
- afternoon [NN], morning [NN]
- ask [VB], halt [VB], resolve [VB], speak [VB], urge [VB]
- citibank [NP], banks [NP], midland [NP], trust [NP], bankers [NP], banking [NP]
- talk [NN], negotiation [NN], discussion [NN], failure [NN]
- baa-1 [JJ], prime-2 [JJ], a-1-plus [JJ], ba-2 [JJ], plus [JJ], baa-2 [JJ], ba-3 [JJ], baa-3 [JJ], b-1 [JJ], a-plus [JJ], implied [JJ]
- qtly [JJ], mthly [JJ], qtrly [JJ], cts [JJ]
- president [NN], vice [NN], executive [NN]
- zinc [NN], vein [NN], diamond [NN], mint [NN], tunnel [NN], gram [NN], silver [NN], nickel [NN], hole [NN], mine [NN]
- yield [NN], yields [NN]
- east [RB], north [RB], south [RB], northwest [RB], southwest [RB], nearby [RB], west [RB]
- labour [NN], strikes [NN], employer [NN]
- work [NN], job [NN], people [NN]
- wednesday [NP], monday [NP], tuesday [NP]
- philippine [NP], manila [NP], philippines [NP]
- chip [NN], chipmakers [NN], microchip [NN], microchips [NN], semiconductor [NN]

List of Publication

Journal Papers

1. Hiroya Takamura and Yuji Matsumoto, “Co-clustering for Text Categorization” (in Japanese), Information Processing Society of Japan (IPSJ) Journal, Vol. 44, No. 2, pages 443–450, 2003.

2. Hiroya Takamura and Yuji Matsumoto, “Constructive Induction and Text Categorization with SVMs” (in Japanese), Information Processing Society of Japan (IPSJ) Transactions on Database, Vol. 44, No. SIG 3 (TOD17), 2003 (to appear).

International Conference

1. Hiroya Takamura and Yuji Matsumoto, “Two-dimensional Clustering for Text Categorization”, in Proceedings of Sixth Conference on Natural Language Learning (CoNLL-2002), Taipei, Taiwan, pages 29–35, August-September 2002.

2. Hiroya Takamura, Hiroyasu Yamada, Taku Kudoh, Kaoru Yamamoto and Yuji Matsumoto, “Ensembling based on Feature Space Restructuring with Application to WSD”, in Proceedings of the Sixth Natural Language Processing Pacific Rim Symposium (NLPRS2001), Tokyo, pages 41–48, November 2001.

3. Hiroya Takamura and Yuji Matsumoto, “Feature Space Restructuring for SVMs with Application to Text Categorization”, in Proceedings of 2001 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 51–57, Pittsburgh, June 2001.

Other Publications

1. Hiroya Takamura and Yuji Matsumoto, “Two-dimensional Clustering for Text Categorization” (in Japanese), Information Processing Society of Japan, Special Interest Group on Natural Language Processing (IPSJ-SIGNL), 2002-NL-149, 2002.

2. Hiroya Takamura, Hiroyasu Yamada, Taku Kudo, Kaoru Yamamoto and Yuji Matsumoto, “Word Sense Disambiguation based on Feature Space Restructuring” (in Japanese), Information Processing Society of Japan, Special Interest Group on Natural Language Processing (IPSJ-SIGNL), 2001-NL-144, 2001.

3. Hiroya Takamura and Yuji Matsumoto, “Feature Space Restructuring for SVMs with Application to Text Categorization” (in Japanese), Information Processing Society of

Japan, Special Interest Group on Natural Language Processing (IPSJ-SIGNL), 2001-NL-143, 2001.

4. Hiroya Takamura and Kokichi Sugihara, “Quantification of Syntactic Information and its Application to Language Comparison” (in Japanese), Information Processing Society of Japan, Special Interest Group on Natural Language Processing (IPSJ-SIGNL), 2000-NL-138, 2000.

5. Yuji Matsumoto, Taku Kudo, Hiroya Takamura, Hiroyasu Yamada and Tetsuji Nakagawa, “System Ensemble Methods in Natural Language Processing” (in Japanese), Proceedings of 2001 Workshop on Information-Based Induction Sciences (IBIS2001), pages 19–24, 2001.

6. Hiroshi Sasaki, Hiroya Takamura and Yuji Matsumoto, “Active Learning for Text Categorization using SVMs and Clustering” (in Japanese), in Proceedings of Forum on Information Technology (FIT) 2002, Volume 2, E-9, pages 99–100, 2002.

7. Ryu Iida, Kentaro Inui, Hiroya Takamura and Yuji Matsumoto, “Incorporating Contextual Clues in Trainable Models for Coreference Resolution” (to appear), in Proceedings of Workshop on The Computational Treatment of Anaphora, 2003.

8. Ryu Iida, Kentaro Inui, Hiroya Takamura and Yuji Matsumoto, “One Method for Resolving Japanese Zero Pronouns with Machine Learning Model” (to appear), Information Processing Society of Japan, Special Interest Group on Natural Language Processing (IPSJ-SIGNL), 2003-NL-154, 2003.

9. Ryu Iida, Hiroya Takamura, Kentaro Inui and Yuji Matsumoto, “Zero-pronoun Resolution with Machine Learning Approach Incorporating Contextual Information” (to appear), in Proceedings of the Ninth Annual Meeting of the Association for Natural Language Processing, 2003.