NAIST-IS-DT0061207

Doctor's Thesis

Text Categorization using Machine Learning

Hirotoshi Taira

February 5, 2002

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

Hirotoshi Taira

Thesis committee: Yuji Matsumoto, Professor
Hiroyuki Seki, Professor
Shin Ishii, Professor

Text Categorization using Machine Learning*

Hirotoshi Taira

Abstract

With the rapid spread of the Internet and the increase in on-line information, the technology for automatically classifying huge amounts of diverse text information has come to play a very important role. In the 1990s, the performance of computers improved sharply and it became possible to handle large quantities of text data. This led to the use of the machine learning approach, which is a method of creating classifiers automatically from the text data given in a category label. This approach provides excellent accuracy, reduces labor, and ensures conservative use of resources.

This paper discusses the following three points related to text classification using machine learning.

1. How to perform highly precise classification by using a large number of word attributes (Chapter 3).

2. How to utilize the distribution of unlabeled examples for high precision when there are few labeled training examples (Chapter 4).

3. How to achieve a highly precise and efficient classification by assuming the existence of sub-categories and using active labeling (Chapter 5).

* Doctor's Thesis, Department of Information Processing, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DT0061207, February 5, 2002.

Before discussion of the above three points, I describe the present state of text classification technology in Chapter 1. In Chapter 2, I briefly explain a mathematical definition of text classification and the data set used in this paper, as well as the evaluation method, and the feature selection method.

In Chapter 3, text classification using a support vector machine is described. Many word features are generally required to perform highly precise text classification by machine learning. However, when many word features are used as the input of conventional learning techniques, they overfit the training data, and the classification accuracy for unknown data decreases.

Therefore, it is necessary to reduce the dimension of features used in learning by selecting about several hundred features which contain much information. However, it is difficult for the classifier to learn with a high enough accuracy from several hundred features. Consequently, text classification is performed by using a support vector machine, which is a new machine learning technique developed to avoid overfitting. Furthermore, when the support vector machine is used, it evaluates whether feature selection would be effective.

In Chapter 4, text classification using transductive boosting is described. Large margin classifiers such as support vector machines and boosting algorithms are effective in generating classifiers with a high classification accuracy when training data is fully available. However, training data is not sufficiently available in many cases because we have to label the training data by hand and the cost of this is high.

Transduction is a method that takes into consideration the distribution of the unknown data for which a classification label is not given. I investigate transduction as a solution for text classification when training data is not sufficient. Specifically, I propose how to use the transductive method with the boosting algorithm, AdaBoost.

In Chapter 5, text classification using the extended tied mixture model is described. We can often assume a potential sub-category for a given category, for example, the "sports" category could have the sub-categories "baseball" and "soccer." I describe such categorization in this thesis. Furthermore, when there is little training data, a small number of unknown data regarded as important

for classification are mechanically chosen and the "active labeling" technique is applied. This technique efficiently creates a highly precise classifier by assigning the categories for a small number of data by hand.

Finally, in Chapter 6, I conclude this paper and describe future work and directions.

Keywords:

natural language processing, machine learning, stochastic information, text categorization, support vector machine, boosting, extended tied mixture model, EM-algorithm, transduction, feature selection

Acknowledgements

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

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

I am deeply indebted to Dr. Masahiko Haruno of ATR. Since the day I joined the NTT Communication Science Laboratories (NTT CS Labs.), he has continuously encouraged me and generously guided me on the fundamentals of natural language processing as well as machine learning. Without his guidance, this thesis would not have been written.

I would like to express my gratitude to Associate Professor Tsuneaki Kato of University of Tokyo, our former group leader. He supported my research and helped me with various aspects.

I am also indebted to Dr. Naonori Ueda of NTT Communication Science Laboratories. He has continuously encouraged me and generously guided me on probabilistic mixture models.

I am grateful to my colleagues in the Computational Linguistics Laboratory at Nara Institute of Science and Technology. They stimulate me and give me valuable comments.

I completed this thesis at the Intelligent Communication Laboratory (ICL) of NTT CS Labs. I would like to give my appreciation to Dr. Ken'ichiro Ishii,

Director of NTT CS Labs., Dr. Noboru Sugamura, Vice-director of NTT CS Labs., Mr. Toshifumi Yamaki and Dr. Toshiro Kita, the former Director of the ICL and Dr. Shigeru Katagiri, Director of the ICL, for providing me with the opportunity to complete my thesis here at NTT CS Labs.

I wish to thank my colleagues in the ICL, especially Dr. Eisaku Maeda, our group leader and Dr. Yutaka Sasaki. Dr. Maeda supported me and gave me valuable comments. Dr. Sasaki encouraged me and discussed many problems with me. My appreciation also goes to Dr. Hideki Isozaki, Mr. Mitsunori Matsushita, Mr. Tsutomu Hirao, Mr. Hideto Kazawa and members of the ICL. Especially, Mr. Hirao and Mr. Kazawa encouraged me to work and research.

I would also like to thank Mr. Takefumi Yamazaki, a former member of the ICL of NTT CS Labs. He suggested to me how to utilize the text corpus and gave me some text data he processed when I began my research of text categorization.

Special thanks are also due to all of the people who gave me valuable commnets and continuous encouragement. They include Jun'ichi Tsujii, Masaaki Nagata, Gunnar Rätsch, Torsten Joachims, Hang Li, Kazumi Saito, Francis Bond, Jun Muramatsu, and Takafumi Mukouchi. Although I cannot list all of their names, I would like to express my thanks all of them.

Finally, I wish to thank my parents for their continuous encouragement and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the rapid growth of the Internet, it has become natural that we handle a text not as printed but as online information. Today, we can search books and news electronically. Almost all companies and individuals have their own web pages and dispatch their information. When there is some information to find, it is usual to search for the information on the Internet. In this way, a lot of information has become open to the public.

If you have much information, you would need to classify it. When the scale of the target is small or the target is written by yourself, it is possible to classify it. However, in the current situation that many and ordinary people electronically post a lot of texts to the Internet, it is becoming impossible to classify them by hand. Then, it is necessary to classify text information automatically into various fields.

Moreover, new services that have never been considered before appeared along with the growth of online information. They include mail filters, web filters, and online help desks. Mail filters shut out unsolicited business e-mails or spam e-mails sent to many unspecified persons, by classifying e-mail into "ordinary mail" or "spam mail." Web filters mainly prevent children from accessing undesirable website content, such as violence and pornography, by classifying web sites into "harmless" or "detrimental" categories. Online help desks that distribute electronic mails from customers to operators may belong to a very special field. Text classification technology is essential to these services.

In looking back upon the historical development of text classification technology, we find that the rule-base approach, i.e., the method of writing classification rules, was mainly used until the second half of the 1980s. This approach is simple but difficult to create rules by hand for high accuracy, and have to write many rules, as we change the domains. However, in the 1990s, the performance of computers improved rapidly and it became possible to handle a great amount of text data. This led to the use of the machine learning approach, which creates classifiers automatically from the text data which were previously labeled with categories by hand. This approach became popular because of its high classification accuracy, reduction of labor, and conservative use of resources.

Various machine learning methods have been applied to text categorization until today. These include k-nearest-neighbor [Yang94], decision trees [Lewis94] and naive-Bayes [Lewis94]. Using a large number of words as features in these methods, which can potentially contribute to the overall task, is a challenge into machine learning approaches. More specifically, the difficulties in handling a large input space are twofold:

- How to design a learning algorithm with an effective use of large scale feature spaces.
- How to choose an appropriate subset of words for effective classification.

These two factors are interdependent [Yang97]. We should develop learning machines with feature selection criteria because the suitable set of words greatly differs depending on the learning method [Lewis94].

Support Vector Machines (SVMs) [Vapnik95] [Cortes95] construct the optimal hyperplane that separates a set of positive examples from a set of negative examples with a maximum margin. A margin is intuitively the distance from a data point to the classification boundary. SVMs have been shown to achieve a good generalization performance for a wide variety of classification problems that require large-scale input space, such as handwritten character recognition [Vapnik95] and face detection [Osuna98].

Two studies have explored the use of SVMs for text categorization [Joachims98] [Dumais98]. Although they both achieve promising performances, they use com-

pletely different feature (word) selection strategies.

In [Joachims98], words are considered features only if they occur in the training data at least three times and if they are not stop words such as "and" and "or." This approach, then, employs the inverse document frequency (IDF) [Salton88] as a value for each feature. In contrast, [Dumais98] considers only 300 words for each category, which are handled by a threshold for high mutual information [Cover91]. The feature value in this case is assigned as a binary value to indicate whether a word appears in a text or not. A natural question about SVM text categorization occurs to us: how much influence do different feature selection strategies have? Is there one best strategy for choosing appropriate words?

Feature selection becomes especially delicate in agglutinative languages such as Japanese and Chinese because in these languages word identification itself is not a straightforward task. Unknown words output by word-segmentation and part-of-speech tagging systems contain both important keywords (like personal and company names) and useless portions of words. The selection of these unknown words is crucial to these languages.

To address these questions, this thesis investigates the effect of prior feature selection in SVM-based text categorization using Japanese newspaper articles. In our experiments, the number of input spaces was gradually increased by two distinct criteria: mutual information (MI) filtering and part-of-speech (POS) filtering. MI filtering selects discriminating words for a particular category from an information-theoretical viewpoint. Words with higher mutual information are more highly representative in a specific text category. In contrast, POS filtering constructs word input space based on parts of speech.

SVMs [Joachims98] [Dumais98] [Taira99] have been applied to text categorization with remarkable success. However, the inductive approach cannot guarantee a sufficiently high accuracy when there is a great difference between the training and test data distributions. This problem becomes extremely serious if the amount of training data is small. This is often the case under many practical conditions such as the classification of on-line Internet texts. Therefore, it is reasonable to utilize the distribution of unlabeled test data for training as well as the distribution of a small number of labeled training data. Nigam et al. proposed an EM-based method with naive-Bayes to take account of the distribu-

tion of test data under the condition where there is a small amount of training data [Nigam00]. Although the method shows substantial improvement over the performance of the standard naive-Bayes classifier, one of its limitations is the tendency to fall into a local optimum.

In contrast, Joachims adapted a transductive method to SVMs [Joachims99] and obtained an improvement in classification performance. Transduction is a general learning framework that minimizes classification errors only for the test data, while induction tries to minimize classification errors for both the training and test data [Vapnik98]. Transductive SVM (TSVM) achieves a high performance by assuming that the portion of unlabeled examples to be classified into the positive class is determined by the ratio of positive to negative examples in the training data. Nevertheless, the possibility of a decrease in performance remains under different but typical conditions when the ratio of positive to negative examples in the training data is very different from that in the test data, e.g., when a classifier, which learned from articles in 1995, is applied to classify articles related to the Internet in 2000.

The boosting algorithm, AdaBoost [Freund97] is an alternative large margin classifier to SVM also recently noted for its high generalization ability in NLP applications [Haruno99] [Schapire00]. It produces highly accurate classification rules by combining a number of weak hypotheses, each of which is only moderately accurate. The advantage of AdaBoost over SVM is that we can choose any classifier suitable for our own classification applications. Since the original AdaBoost is an inductive learning method, I propose here a novel transductive boosting algorithm to cope with a small number of training data; in particular, under the condition where the ratio of positive to negative examples greatly differs between the training and test data. The experimental results demonstrate that the proposed algorithm not only outperforms SVM and AdaBoost, but is also comparable and sometimes superior to TSVM. The advantage of the method is significant when the number of training data is small and the ratio of positive examples to negative ones in the training data is different from that in the test data. These results confirm that the usefulness of the transductive approach is not limited to SVM but is also effective with a variety of learning methods.

For statistical classifier design using a large number of unclassified text exam-

ples and sub-categories, I also apply a probabilistic model, called an Extended Tied Mixture (ETM) model [Ueda01], with a Bayesian learning algorithm. I also apply a confidence-based "active labeling" method [Ueda01] for text categorization to efficiently select informative samples for labeling. The usefulness of the method is assessed through experimental results.

Recently, based on probabilistic classifier design, several methods using a large number of unlabeled samples gathered at little cost have been proposed to overcome the small sample size problem [Shahshahani94][Miller97][Nigam00]. Before those methods were developed, McLachlan et al. [McLachlan87] originally formulated this kind of problem as partial nonrandom classification. In this thesis, to overcome this small sample size problem, I utilize ETM models, which are natural extensions of McLachlan's basic model, for text categorization. In addition, we can often assume a potential sub-category for a given category, for example, the "sports" category could have the sub-categories "baseball" and "soccer." The ETM models also contain the idea of latent sub-categories. Furthermore, I consider the issue of how we should select informative samples for new labeling from unlabeled samples to be an important practical problem. Active labeling is required when the number of labeled samples is initially too small or the current classification performance is unsatisfactory. This active labeling technique is essentially different from conventional active learning for pattern recognition (e.g., [Cohn96]) in the sense that the label information is completely missing.

Chapter 2

Text Categorization

2.1 A mathematical definition of the text classification task

The text classification task is defined as the automatic classification of a document into two or more predefined classes. Generally, in the text classification task, a document is expressed as a vector of many dimensions,

$$\mathbf{x} = (x_1, x_2, \ldots, x_l).$$

Each feature of a document vector has two values: whether a certain word appears in the document and the real value that is weighted by a suitable method, for example, TF-IDF.

For example, the following two documents,
"All-star game will be held in Boston" (document 1)
and
"Chess is the champion of games" (document 2)
are expressed as $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ (Fig. 2.1)
using the four word features "all-star", "Boston", "chess" and "game".

In the example above, the document is expressed by a 4 dimensional feature vector. However, it is desirable to use at least 10,000 features, or as many as possible, to classify various documents at a high accuracy. However, when most

	"all-star"	"Boston"	"chess"	"game"
document 1 ($\boldsymbol{x}_1$)	1	1	0	1
document 2 ($\boldsymbol{x}_2$)	0	0	1	1

Figure 2.1: Vector representation of two documents.

machine learning techniques are used, having many features causes overlearning and a very long calculation time. In order to avoid these problems, several feature selection methods have been proposed to cut down the features from 100 to 10,000 by using various evaluation standards such as word appearance frequency, document frequency, mutual information, and information profit. On the other hand, a class label y is given, which stands for which class the document belongs to. The number of classes can be two or more. A two-class case, which solves whether a document belongs to a class or not, is the easiest case and is called a "binary-class problem." A three-class or more case is called a "multi-class problem." Also, the problem can be divided into two cases; namely, the case where a document has only one label and the case where a document has two or more labels, called "multi-label." Generally, multi-class or multi-label classification problems are solved by combining many binary-class classifiers. For example, in Reuters-21578, which is one of the typical benchmarks of text classification, 12,902 news articles of 200 words on average were classified into 118 categories such as corporate, acquisitions, earnings, money market, and grain. One article has 1.2 categories on average. Under the above setup, text classification can be formalized as the problem that calculates function $f(\mathbf{x})$, such that the number of the predicted label different from a true label minimizes,

$$\sum_{f(\mathbf{x_i})\neq y_i,(\mathbf{x_i},y_i)\in S} 1$$

when training data S

$$S = \{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), \ldots, (\mathbf{x_n}, y_n)\}$$

is given. In addition, class label y equals -1 for a positive example and $+1$ for a negative example in binary classification, and the reliability of prediction for a

label equals $|f(\mathbf{x})|$ and the predicted label equals $sign(f(\mathbf{x}))$.

2.2 Data Set

Table 2.1: RWCP corpus for training and test.

Category	training sets	test sets
sports	161	147
criminal law	156	148
government	135	142
educational system	110	124
traffic	112	103
military affairs	110	118
international relations	96	97
communications	76	83
theater	86	95
agriculture	78	72

Experiments were performed using the RWCP corpus [Toyoura96] and the 20 Newsgroups data set [CMUdata]. The RWCP corpus contains 30,207 newspaper articles taken from the Mainichi Shinbun Newspaper published in 1994 [Mainichi95]. Each article was assigned multiple Universal Decimal Classification (UDC) codes, each of which represented a text category. UDC is a hierarchical classification system that has about 60,000 main categories. The text collection has a total of 97,095 categories, among which there are 14,407 different categories, i.e., 3.2 categories per article.

In the rest of this thesis, I will focus on the ten categories that appeared most often in the corpus: sports, criminal law, government, educational system, traffic, military affairs, international relations, communications, theater, and agriculture. The results for other categories were very similar to those of these ten categories. We made binary classifiers for each of the categories whether a sample belonged

to the category or not. The total number of articles used for both training and test were 1,000. Table 2.1 summarizes the number of training and test articles in each category.

These articles were word-segmented and POS tagged by the Japanese morphological analyzing system, Chasen [Matsumoto97]. The process generated 20,490 different words.

The 20 Newsgroups data set consists of 20,017 articles collected from UseNet. They are divided almost evenly among 20 different UseNet discussion groups, postings over a period of several months in 1993. [Nigam00] An article belongs to one newsgroup of twenty: alt.atheism, comp.sys.ibm.pc.hardware, soc.religion.christian, sci.space, sci.med, sci.electronics, rec.sport.hockey, rec.sport.baseball, rec.motorcycles, rec.autos, misc.forsale,comp.sys.mac.hardware, talk.religion.misc, talk.politics.misc, talk.politics.mideast, comp.graphics, comp.os.ms-windows.misc, comp.windows.x, sci.crypt, and talk.politics.guns. I removed common short words including "and" and "or" using a stoplist As a result, there are 62,258 unique words that occur more than once.

In order to compare with Nigam's experiment result, test data was created as follows. A test set of 4,000 articles is selected by posting date, the last 20% of the articles from each newsgroup. An unlabeled set is formed by randomly selecting 10,000 documents from those remaining. Labeled training sets are formed by randomly selecting from the remaining 6,000 documents. The sets are created with equal numbers of documents per class. In this thesis, I will focus on 5 comp.* categories, i.e., comp.windows.x, comp.graphics, comp.sus.mac.hardware, comp.sus.ibm.pc.hardware, and comp.os.ms-windows.misc.

2.3 Feature Selection

2.3.1 Mutual Information

The mutual information (MI) between a word t_i and a category c is defined in equation (2.1). Here, $P(t_i)$ is the probability of the existance of a word t_i in an article, $P(c)$ is the probability that an article belongs in a category c, $P(t_i, c)$

is the joint probability of t_i and c. MI becomes large when the occurrence of t_i is biased to one side between a category c and other categories. Consequently, it can be expected that the words with high mutual information in a category c are keywords in that category. The question we would like to discuss here is whether words with a fixed number of high mutual information can achieve a good generalization over all text categories.

$$MI(t_i, c) = \sum_{t_i \in \{0,1\}} \sum_{c \in \{+,-\}} P(t_i, c) \log \frac{P(t_i, c)}{P(t_i)P(c)} \tag{2.1}$$

Table 2.2 shows 300th, 500th, 1,000th, 5,000th and 10,000th terms of mutual information in each category. In general, up to the 500th or 1,000th term, the words were specific to each category. For example, "screwball" and "golfer," "peace" and "Moscow" are specific to sports and military, respectively. It is also interesting to note that "Kazakhstan" is an unknown word that plays an important role in the category of military affairs. In contrast, after the 1,000th term, words do not seem to be specialized to any specific category.

2.3.2 Part of Speech

I tested the following five feature sets based on parts of speech. The number of different words for each part of speech is summarized in Table 2.3. The total number of different words of these parts of speech is 18,111.

set 1: common nouns

set 2: set 1 + proper nouns

set 3: set 2 + verbal nouns

set 4: set 3 + unknown words

set 5: set 4 + verbs

2.3.3 Evaluation Method

The F-measure was used as the evaluation measure. For every classification, we can calculate
$a =$ (the number of data the classifier evaluates positive for positive data),
$b =$ (the number of data the classifier evaluates positive for negative data),
$c =$ (the number of data the classifier evaluates negative for positive data).
Then, we can calculate precision (P) and recall (R) as

$$P = \frac{a}{a+b}, \quad R = \frac{a}{a+c}$$

By combining precision and recall, the F-measure is defined as follows:

$$F = \frac{1+\beta^2}{\frac{1}{P} + \beta^2 \frac{1}{R}}$$

The F-measure varies between 0 and 1. The larger the F-measure becomes, the higher the classification accuracy gets. β is a weight parameter, and we set $\beta = 1$.

Table 2.2: Words selected with MI.

	words	
Feature	300th	500th
sports	変化球 (screwball)	応援 (cheering)
criminal law	疑惑 (suspicion)	送検 (commit for trial)
government	議会 (parliament)	運輸省 (The Ministry of Transport)
educational system	塾 (cram school)	文相 (Minister of Education)
traffic	大型車 (large-size car)	配達 (delivery)
military	平和 (peace)	モスクワ (Moscow)
international	有事 (emergency)	各国 (countries)
communications	会議 (meeting)	衛星通信 (satellite communications)
theater	台本 (play script)	終演 (the end of a show)
agriculture	イモ (potato)	砂糖 (sugar)

words		
1,000th	5,000th	10,000th
ゴルファー (golfer)	アンケート (questionnaire)	目安 (standard)
地下 (underground)	売る (sell)	増進 (increase)
約束 (promise)	根幹 (basis)	さえぎる (interrupt)
理想的だ (ideal)	涙 (tear)	即 (immediately)
速さ (speed)	池 (pond)	双方向 (bi-direction)
カザフスタン (Kazakhstan)	実際 (practical)	降下 (descend)
大筋 (outline)	年内 (within the year)	裁く (judge)
伝送 (transmission)	正常 (normal)	慎重 (careful)
賞 (prize)	要素 (element)	ロイ (Roy)
飼料 (livestock feed)	改善 (improvement)	変貌 (look different)

Table 2.3: POS distribution of training data.

	POS (part of speech)				
	common noun	proper noun	verbal noun	unknown	verb
Number of words	8629	2725	2829	1634	2294
Percentage (%)	47.6	15.0	16.0	7.4	12.7

Chapter 3

Text Categorization Using Support Vector Machines

3.1 Support Vector Machines

A support vector machine (SVM) is a machine learning method that divides space into a training positive examples side and a negative examples side. It also creates hyperplanes as the margin between the positive and negative examples which becomes the maximum. These hyperplanes serve as the optimum solution based on the concept of structural risk minimization.

The conceptual structure of SVM is shown in Fig. 3.1. SVM calculates the hyperplanes that separate a positive example from a negative example in hyperspace. We call the distance between the positive-side hyperplane nearest the negative examples and the negative-side hyperplane nearest the positive examples the margin. SVM calculates the optimal hyperplanes that supply the maximum margin, where $\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$ is the final border hyperplane for classification. The training examples on $\boldsymbol{w} \cdot \boldsymbol{x} + b = 1$ and $\boldsymbol{w} \cdot \boldsymbol{x} + b = -1$ are called "support vectors." However, when positive and negative examples cannot be separated completely, separated hyperplanes are determined by also taking errors into consideration.

The problem that maximizes the margin in training data can be converted to the problem of quadratic programming that minimizes the purpose function (for-

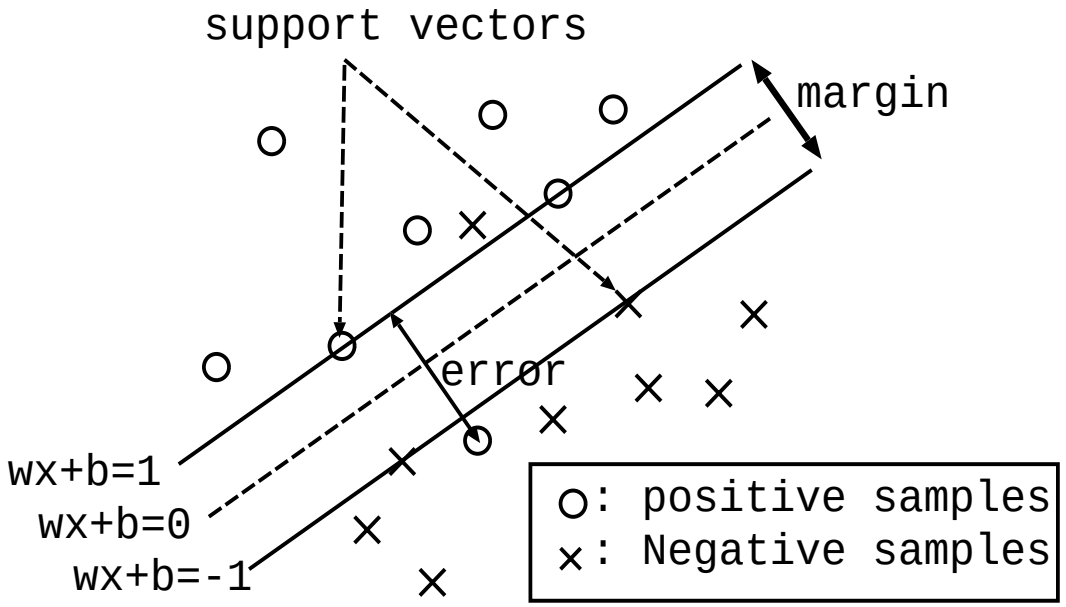

Figure 3.1: Support Vector Machines.

mula 3.1) under the restricted condition of formula (3.2) by introducing Lagrange multiplier α_i.

$$-\sum_{i=1}^{l}\alpha_i + \frac{1}{2}\sum_{i,j=1}^{l}\alpha_i\alpha_j y_i y_j \mathbf{x_i}\cdot\mathbf{x_j} \tag{3.1}$$

$$\sum_{i=1}^{l}\alpha_i y_i = 0, \quad \forall i : \alpha_i \geq 0 \tag{3.2}$$

Here, y_i is a variable showing the label of the example $\mathbf{x_i}$, which is positive or negative. When $\mathbf{x_i}$ is a positive example, $y_i = +1$, and when $\mathbf{x_i}$ is a negative example, $y_i = -1$. We can determine $\boldsymbol{w}$ and b as follows using solved α_i, positive support vectors $\mathbf{x_a}$, and negative support vectors $\mathbf{x_b}$ by quadratic programming:

$$\boldsymbol{w} = \sum_{i=1}^{l}\alpha_i y_i \mathbf{x_i} \tag{3.3}$$

$$b = -\frac{1}{2}(\boldsymbol{w}\mathbf{x_a} + \boldsymbol{w}\mathbf{x_b}) \tag{3.4}$$

Moreover, SVM can also solve nonlinear hypotheses by replacing inner product $\mathbf{x_i}, \mathbf{x_j}$ in formula (3.1) by nonlinear function $K(\mathbf{x_i}, \mathbf{x_j})$, which is called a kernel function.

When learners learn the parameters of a classification model, the phenomenon called "overfitting" arises, where the more complicated the model is, the larger the error classification for new data becomes. On the other hand, when the model better suits the training data, it gets a smaller error for training data. The complexity of a model in SVM is mathematically defined by the measure called VC dimension (Vapnik-Chervonenkis dimension), and an optimum solution can be calculated based on the concept called structural risk minimization [Vapnik95], which minimizes the sum of the classification error and the complexity of the model for training data without overfitting.

Moreover, in machine learning methods before SVM, when the dimension of the data increases, the amount of calculation and storage capacity required increases abruptly, and the so-called "curse of dimensionality" occurs. For this reason, we could not handle high-dimensional space. SVM uses only support vectors for calculation. Thus, SVM excels over conventional machine learning methods in that it can also handle very-high-dimensional input. It has been reported that SVM surpasses Rocchio, decision tree, naive-Bayes and Bayes net in accuracy. [Dumais98]

In addition, SVM is a large margin classifier like boosting. However, the margin in boosting is expressed by the sum of the distance costs of all samples to the classification border, whereas the margin in SVM is the distance between two hyperplanes along support vectors.

SVMs are based on Structural Risk Minimization. The idea of structural risk minimization is to find a hypothesis h for which we can guarantee the lowest generalization error. The following upper bound (3.5) connects $error_g(h)$, the generalization error of a hypothesis h with the error of h on the training set $error_t(h)$, and the complexity of h [Vapnik95]. This bound holds with a probability of at least $1 - \eta$. In the second term on the right-hand side, n denotes the number of training examples and λ is the VC dimension, which is a property of

the hypothesis space and indicates its complexity.

$$error_g(h) \leq error_t(h) + 2\sqrt{\frac{\lambda(\ln\frac{2n}{\lambda}+1) - \ln\frac{\eta}{4}}{n}} \tag{3.5}$$

Equation (3.5) reflects the well-known trade-off between the training error and the complexity of the hypothesis space. A simple hypothesis (small λ) would probably not contain good approximating functions and would lead to a high training (and true) error. On the other hand, a too-rich hypothesis space (high λ) would lead to a small training error, but the second term on the right-hand side of (3.5) would be large (overfitting). The right complexity is crucial for achieving good generalization. In the following, we assume that the linear threshold functions represent a hypothesis space in which $\boldsymbol{w}$ and b are parameters of a hyperplane and $\boldsymbol{x}$ is an input vector:

$$h(\boldsymbol{x}) = sign\{\boldsymbol{w}\cdot\boldsymbol{x}+b\} = \begin{cases} +1, & \text{if } \boldsymbol{w}\cdot\boldsymbol{x}+b > 0 \\ \text{-1}, & \text{else} \end{cases}$$

Lemma 1 clarifies the relationship between the dimension of the input space $\boldsymbol{x}$ of a set of hyperplanes and its VC dimension λ.

Lemma 1 (Vapnik) *Consider hyperplanes $h(\boldsymbol{x}) = sign\{\boldsymbol{w}\cdot\boldsymbol{x}+b\}$ as a hypothesis. If all example vectors $\boldsymbol{x_i}$ are contained in a ball of radius R and the following is required such that for all examples $\boldsymbol{x_i}$*

$$|\boldsymbol{w}\cdot\boldsymbol{x}+b| \geq 1, \quad with \quad ||\boldsymbol{w}|| = A,$$

then this set of hyperplanes has a VC dimension λ bounded by

$$\lambda \leq min([R^2A^2], n) + 1 \tag{3.6}$$

Note that the VC dimension of these hyperplanes does not always depend on the number of input features. Instead, the VC dimension depends on the Euclidean length $||\boldsymbol{w}||$ of the weight vector $\boldsymbol{w}$. equation (3.6) supports the possibility that SVM text categorization achieves good generalization even if a huge

number of words are given as an input space. Further experimental evaluations are required because Equations (3.5) and (3.6) both give us only a loose bound.

Basically, SVM finds the hyperplane that separates the training data with the shortest weight vector (i.e., $min||\boldsymbol{w}||$). The hyperplane maximizes the margin between the positive and negative samples. Since the optimization problem is difficult to handle numerically, Lagrange multipliers are introduced to translate the problem into an equivalent quadratic optimization problem. For this kind of optimization problem, efficient algorithms exist that are guaranteed to find the global optimum. The result of the optimization process is a set of coefficients α_i^* for which (3.7) is minimum. These coefficients can be used to construct the hyperplane satisfying the maximum margin requirement.

$$\begin{aligned} Minimize \quad & -\sum_{i=1}^{n} \alpha_i + \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \boldsymbol{x_i} \cdot \boldsymbol{x_j} \\ s.t. \quad & \sum_{i=1}^{n} \alpha_i y_i = 0 \quad \forall i : \alpha_i \geq 0 \end{aligned} \tag{3.7}$$

SVMs can handle nonlinear hypotheses by simply substituting every occurrence of the inner product in equation (3.7) with any kernel function $K(x_1, x_2)$. More precisely, any function which satisfies the Mercer's condition [Vapnik95] can be a kernel function. Among the many types of kernel functions available, I focus on the dth polynomial functions:

$$K_{poly}(x_1, x_2) = (x_1 \cdot x_2 + 1)^d \tag{3.8}$$

3.2 Experimental Results

This section describes our experimental results for two feature selection methods in SVM text categorization: mutual information filtering and part-of-speech filtering. For comparison, we also tested a decision tree induction algorithm C4.5 [Quinlan93] with default parameters. Before going into the details of the results, we first explain the experimental setting.

3.2.1 Experimental Setting

I performed our experiments using the RWCP corpus (see Section 2.2). In the rest of this chapter, I focus on the ten categories that appeared most often in the corpus: sports, criminal law, government, educational system, traffic, military affairs, international relations, communications, theater, and agriculture. The results for other categories were very similar to those of these ten categories. The total number of articles used for both training and test were 1,000.

I used all types of parts of speech in the mutual information filtering and used only common nouns, proper nouns, verbal nouns, unknown words, and verbs (total number of subset words was 18,111) in the part-of-speech filtering. Throughout our experiments, various subsets of these extracted words were used as input feature spaces, and the value for each feature was a binary value that indicated whether a word appeared in a document or not. A binary value was adopted to study the pure effects of each word.

3.2.2 Mutual Information Filtering

Table 3.1: F-measures with MI on C4.5.

Feature	300	500	1000	5000	10000	15000
sports	**87.5**	86.2	85.2	83.6	83.6	83.6
criminal law	67.9	**70.8**	68.9	68.8	68.8	68.8
government	**65.5**	63.0	58.0	57.9	57.9	57.9
educational system	**72.0**	69.2	70.1	70.1	70.1	70.1
traffic	**63.0**	61.0	61.0	61.0	61.0	61.0
military affairs	**75.9**	73.3	69.1	68.8	68.8	68.8
international relations	**50.0**	45.6	42.4	42.4	42.4	42.4
communications	**52.7**	50.3	50.3	50.3	50.3	50.3
theater	**80.9**	80.9	79.5	79.5	79.5	79.5
agriculture	**84.4**	84.4	84.4	83.8	83.8	83.8
avg.	**70.0**	68.5	66.9	66.6	66.6	66.6

Tables 3.1 and 3.2 show the averages of the recall and precision on C4.5 and SVMs, respectively, when the number of words is changed by using various MI thresholds. The order of polynomial d (see equation (3.8)) used is 1 and 2. The boldface values in the tables represent the best performance for each category. It can be easily seen that the best number of words differs greatly from category to category in SVMs, while the best performance in C4.5 is achieved with the smallest number of words. The average performance is best for SVMs when the number of words is 15,000; however, in C4.5 an abrupt drop in average performance is seen at 500 words. It is also notable that on average SVMs significantly outperform C4.5, which indicates that SVMs can make better use of a huge number of input words.

Let us now look more closely at the recall and precision for this data. Figure 3.2 plots the recall on C4.5 and SVMs with $d = 1$. On C4.5, recall is the highest for a small number of features (such as 300 and 500), drops after that, and then maintains a low level. This indicates that it is no use to adopt large input spaces of over 1,000 features for C4.5. On the other hand, recall on SVMs tends to improve as the number of words increase except for the "international relations" category, which monotonically decreases. Thus, we can safely say that increasing the number of words improves recall on SVMs.

In contrast to recall, the change in precision on SVMs is more complicated (Figure 3.3, bottom). For the five categories with the highest precision, the curves decline continuously but only slightly. This is a reasonable phenomenon because the excessive number of keywords may extract irrelevant documents. The other five categories with middle precision differ greatly. Two curves increase monotonically and two others drift, while the remaining one has a peak at 10,000 features. This shows that the increase in features does not involve a large decrease in precision. In other words, the feature selection ability inherent in SVMs can prevent precision from dropping abruptly with an increase in feature space. These results show that good generalization performance with a large number of features (15,000) depends on achieving good precision on SVMs. These results contrast with those on C4.5 in that lower precision continues after 1,000 features. The flatness of recall and precision over 1,000 features on C4.5 means that the words after the 1,000th term are regarded as useless words for recognizing slight

differences in the characteristics among the categories in the measure of mutual information. It is clear that SVMs have much higher ability in finding important words for categorizing texts than MI or the gain ratio used in C4.5.

3.2.3 Part-of-Speech Filtering

Tables 3.3 and 3.4 show the F-measures on C4.5 and SVMs, respectively, when each of the five features mentioned in Section 3.2.1 are used. Boldface numbers represent the best value in each category. It is clear that the best feature set greatly differs from category to category in both cases. The best average performance is achieved on SVMs when all of the words are used (Feature Set 5).

What are the contributions of each part of speech in SVM text categorization? In Table 3.4, common nouns are so powerful that near-optimal performance can be achieved with only one part of speech. Proper nouns, verbal nouns, and verbs improve results in more than half of the categories, while unknown words contribute to improving only three categories. This is probably because the unknown words contain irrelevant portions of a word as well as important keywords for a category. Note that there is no abrupt drop in performance as a result of incrementally adding any parts of speech.

Let us now consider the recall and precision results for this data. Figure 3.4 plots the recall on C4.5 (top) and SVMs (bottom). The situation looks completely different from that of the MI filtering experiment because the number of terms in Feature Set 1 (common nouns) reaches 8,629. Looking at the case for SVMs in Figure 3.2, only after 10,000 features do we notice that POS filtering and MI filtering have the same tendencies: precision curves increase monotonically while recall curves differ among categories. This monotonic increase in the precision curve shows that every part of speech contains powerful keywords specific to one category. This fact was unknown for C4.5 because the huge input space of C4.5, which has a worse generalization ability than SVMs', causes a drop in precision by overfitting, hiding the effect of POS. The recall curve shows not only that the increase in features above 10,000 words does not always improve recall but also that the drop in recall is not serious on SVMs.

3.3 Summary

This paper investigates the effect of prior feature selection in Support Vector Machine (SVM) text categorization. The input space was gradually increased by using mutual information (MI) filtering and part-of-speech (POS) filtering, which determine the portion of words that are appropriate for learning from information-theoretic and linguistic perspectives, respectively. We tested the two filtering methods on SVMs as well as on a decision tree algorithm C4.5. The SVMs' results common to both types of filtering are that 1) the optimal number of features differed completely across categories, and 2) the average performance for all categories was best when all of the words were used. In addition, a comparison of the two filtering methods clarified that POS filtering on SVMs consistently outperformed MI filtering, which indicates that SVMs cannot find irrelevant parts of speech. These results suggest a simple strategy for SVM text categorization: use all of the words found through a rough filtering technique such as part-of-speech tagging.

This chapter has described various aspects of feature selection in SVM text categorization. My experimental results clearly show that SVM text categorization handles large-scale word vectors well but is limited for finding irrelevant parts of speech. This suggests a simple but highly practical strategy for organizing a large number of words found through a rough filtering technique such as part-of-speech tagging.

Table 3.2: F-measures with MI on SVMs.

	Order of polynomial $d = 1$					
Feature	300	500	1000	5000	10000	15000
sports	**91.9**	89.5	90.9	90.8	90.0	90.4
criminal law	71.5	69.2	68.2	72.2	74.3	75.5
government	66.6	68.4	74.4	79.3	76.8	78.2
education	68.4	69.1	71.7	78.1	80.0	**80.1**
traffic	66.6	70.5	**72.1**	70.7	71.0	71.0
military affairs	66.3	71.3	74.5	74.6	75.6	**77.1**
international relations	54.3	60.1	62.9	61.6	61.0	57.1
communications	64.0	65.7	59.3	55.7	53.6	58.2
theater	83.9	**88.7**	86.2	83.6	83.8	83.8
agriculture	85.9	**87.5**	85.7	85.0	85.9	84.1
avg.	71.9	74.0	74.5	75.1	75.2	**75.5**

	Order of polynomial $d = 2$					
Feature	300	500	1000	5000	10000	15000
sports	91.9	89.5	90.9	90.0	89.6	89.6
criminal law	70.7	71.0	70.3	73.0	74.1	**76.4**
government	66.1	68.2	76.4	79.0	78.0	**79.8**
education	68.2	69.7	73.5	77.8	79.8	79.6
traffic	66.6	71.6	71.8	68.3	69.1	71.1
military affairs	68.3	71.9	75.7	74.7	75.9	76.3
international relations	56.9	61.9	**63.5**	60.4	59.2	58.9
communications	64.9	**66.6**	59.3	53.3	50.0	50.0
theater	84.0	83.9	88.2	86.2	82.2	82.4
agriculture	85.2	86.6	85.7	83.2	85.0	84.1
avg.	72.2	74.0	75.5	74.5	74.2	74.8

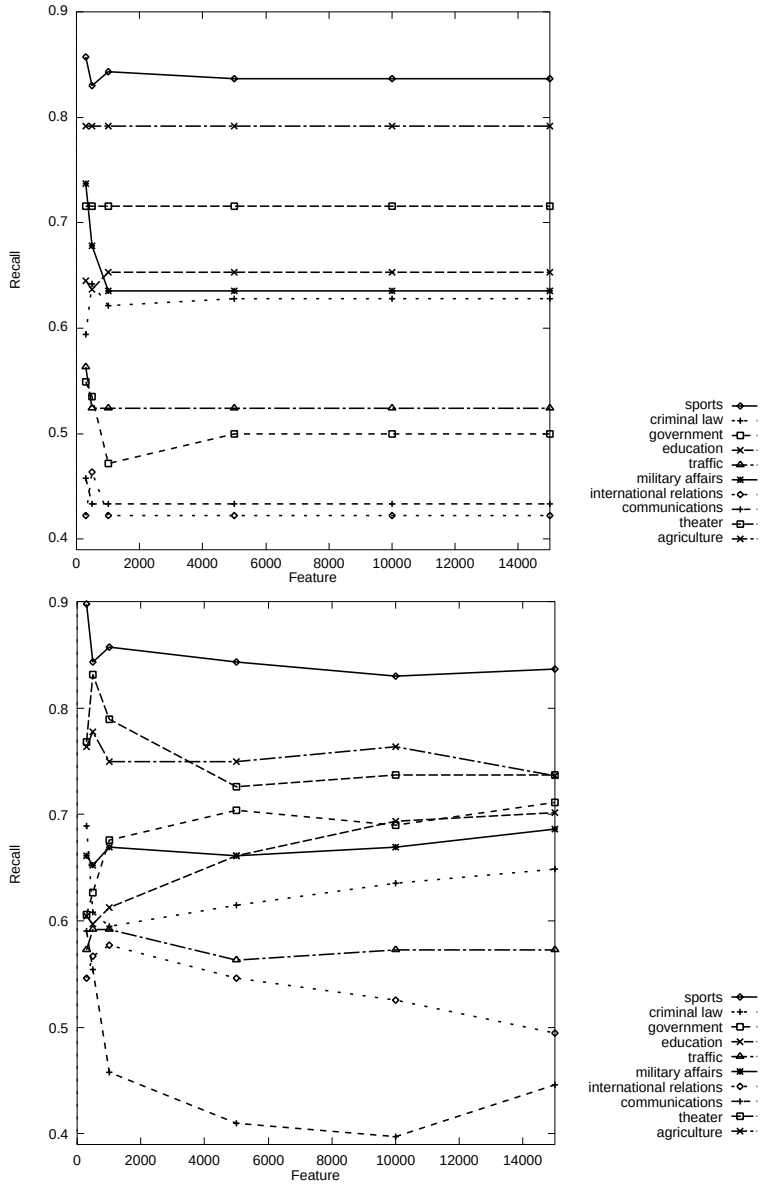

Figure 3.2: Recall with MI features on C4.5 (top) and on SVMs (bottom).

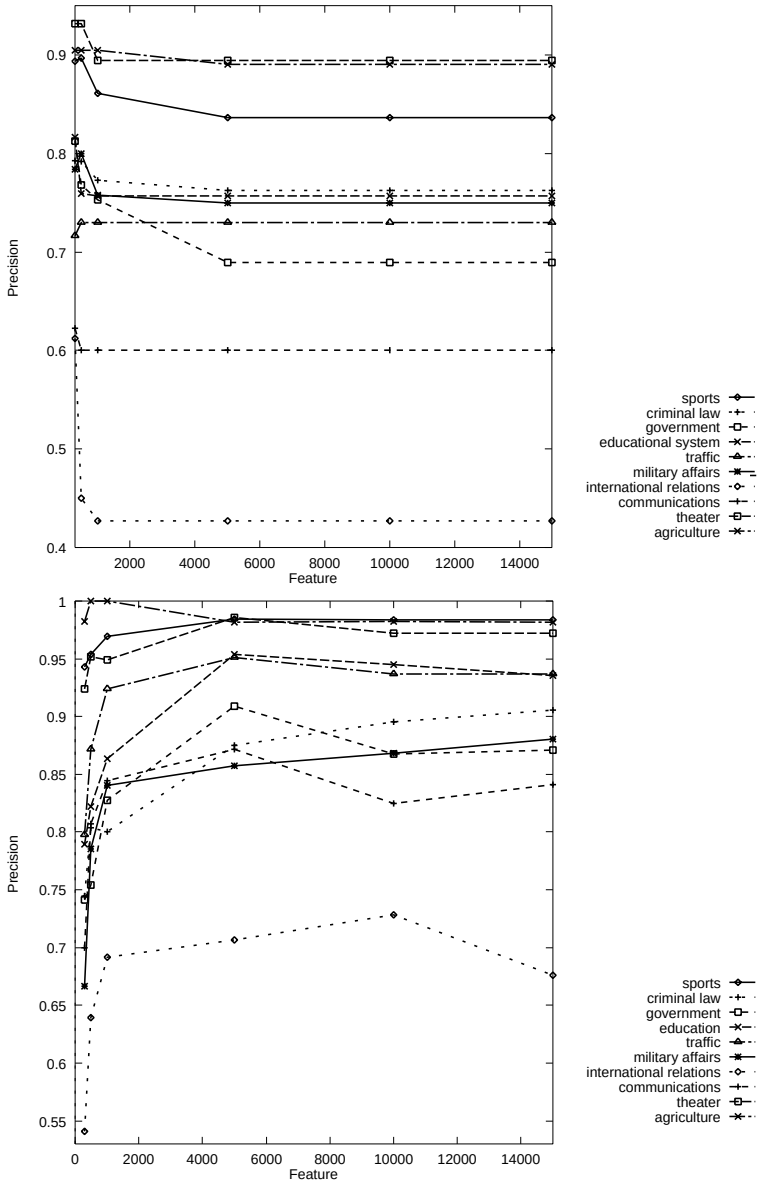

Figure 3.3: Precision with MI features on C4.5 (top) and on SVMs (bottom).

Table 3.3: F-measures with POS filtering on C4.5.

Feature	set 1	set 2	set 3	set 4	set 5
sports	**84.7**	82.9	83.4	83.0	83.4
criminal law	61.5	59.3	**71.3**	71.3	71.3
government	58.0	**62.8**	62.7	62.7	60.4
education	60.2	63.5	62.8	**70.2**	70.2
traffic	58.2	56.4	58.1	58.1	**59.4**
military affairs	**75.5**	71.8	71.8	71.8	71.8
international relations	**49.3**	44.1	48.9	46.4	46.4
communications	49.6	48.5	**51.0**	44.6	44.6
theater	**79.7**	71.3	79.2	79.2	79.2
agriculture	81.2	**81.4**	81.4	81.4	81.4
avg.	65.8	63.9	**67.1**	66.9	66.8

Table 3.4: F-measures with POS filtering on SVMs.

	Order of polynomial $d = 1$				
Feature	set 1	set 2	set 3	set 4	set 5
sports	92.2	**93.2**	92.9	92.0	90.5
criminal law	74.0	73.3	72.5	73.0	**75.2**
government	76.9	78.4	79.3	78.9	**79.6**
education	81.4	80.8	81.4	81.4	81.2
traffic	72.8	76.0	74.8	74.8	73.0
military affairs	80.1	76.1	78.8	77.0	**80.1**
international relations	54.5	59.2	60.7	61.2	**64.0**
communications	65.7	63.8	**69.3**	68.9	65.7
theater	83.8	82.4	85.2	85.2	**87.0**
agriculture	87.5	**88.3**	87.5	86.6	85.0
avg.	76.9	77.5	78.0	77.9	**78.1**
	Order of polynomial $d = 2$				
Feature	set 1	set 2	set 3	set 4	set 5
sports	91.4	92.4	92.4	92.0	90.8
criminal law	73.0	73.6	73.3	73.3	74.9
government	76.7	78.7	79.0	78.2	79.2
education	81.4	80.3	80.9	**81.8**	80.3
traffic	72.0	74.5	**76.0**	75.2	72.2
military affairs	77.3	76.1	76.2	77.0	77.9
international relations	54.6	60.2	61.4	62.2	64.0
communications	67.6	62.3	68.0	67.1	63.2
theater	83.8	82.4	85.2	85.2	85.0
agriculture	87.5	88.3	86.6	86.6	84.8
avg.	76.5	77.2	77.9	77.8	77.2

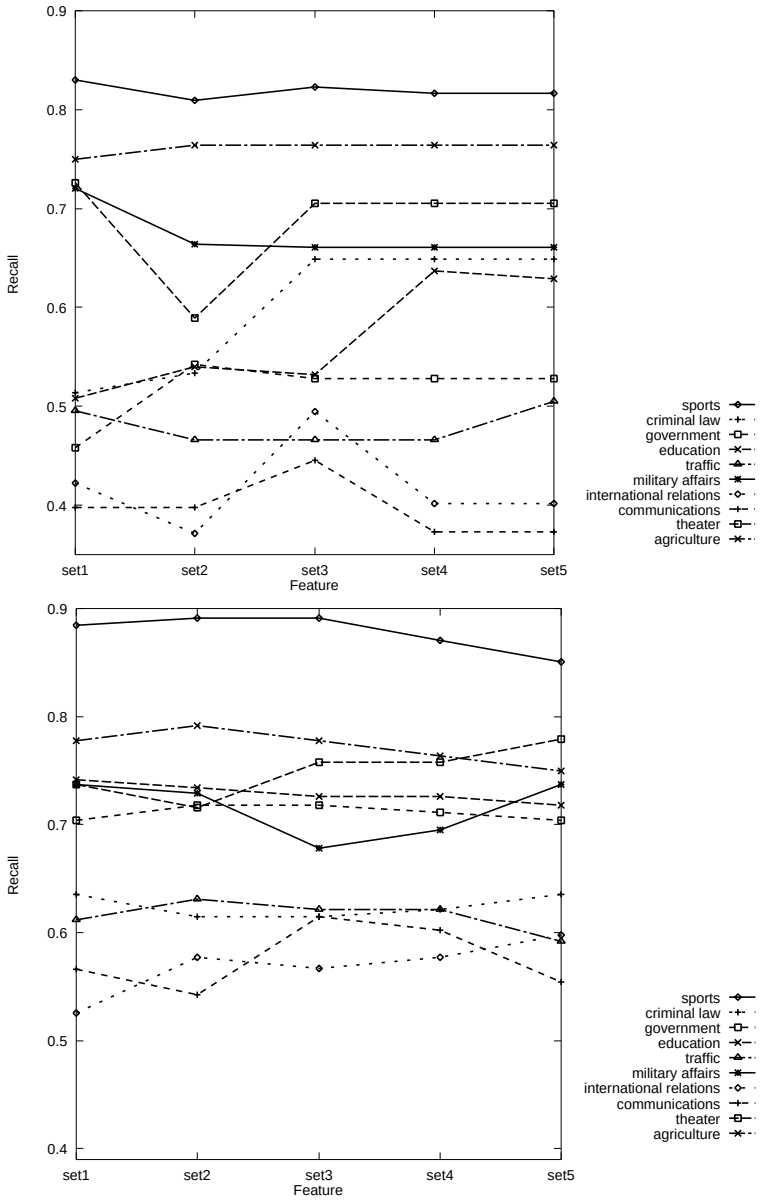

Figure 3.4: Recall with POS features on C4.5 (top) and on SVMs (bottom).

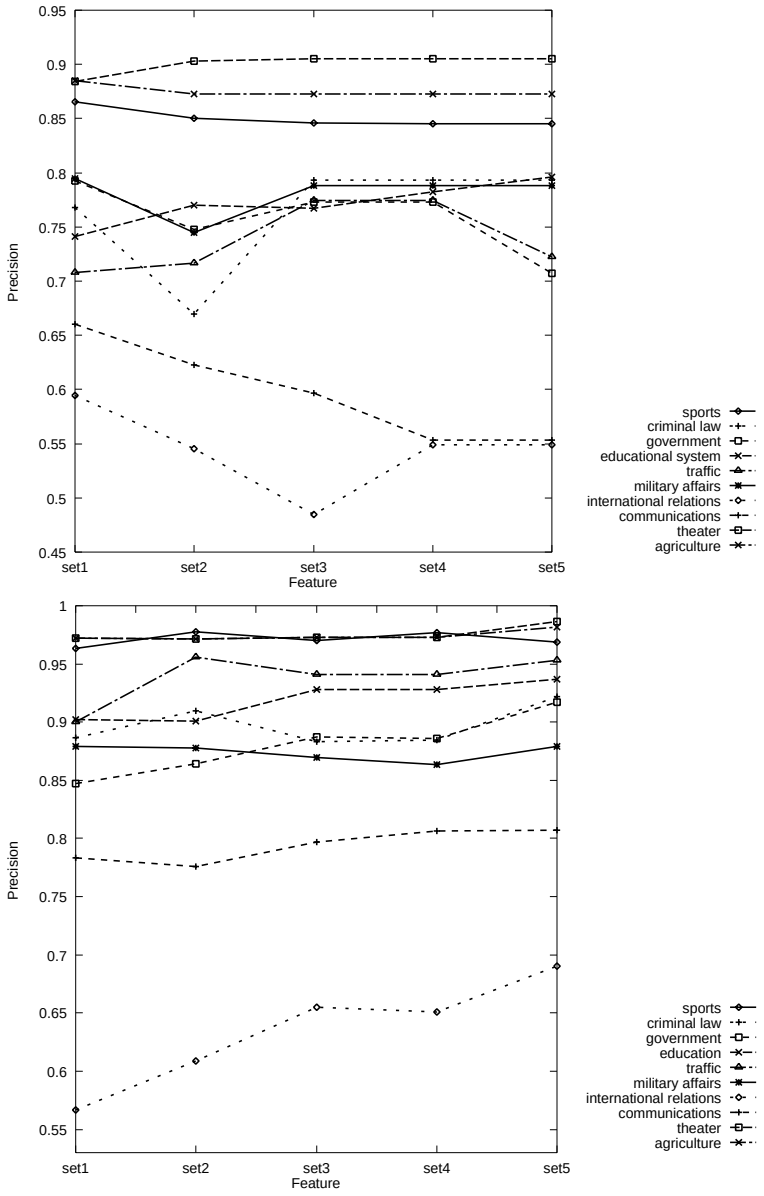

Figure 3.5: Precision with POS features on C4.5 (top) and SVMs (bottom).

Chapter 4

Text Categorization Using Transductive Boosting

4.1 Boosting

Boosting is a technique for creating a highly precise classifier by combining weak learners with somewhat better prediction than random prediction. A conceptual diagram of boosting is shown in Fig. 4.1. In boosting, the same weak learner is repeatedly called T times and hypotheses $h_t(t = 1 \dots T)$ are generated while changing weight D_t for training data. A single operation is called a "round." Finally, boosting linearly sums hypotheses h_t weighted by α_t, which is calculated based on the classification error of each hypothesis ϵ_t, to generate the final hypothesis H, where the sign of the sum is the predicted class label.

AdaBoost, which is one of the boosting algorithms, is shown in Fig. 4.2. In AdaBoost, the same weights are initially given to all training examples. In each repetition, the weights for the misclassified examples are increased exponentially and more difficult examples are concentrated on for learning. This is called AdaBoost (Adaptive Boosting) because the algorithm adaptively decides the weights for weak hypotheses α_t and the weights in the following round for training examples D_t by using the error rate of classification, ϵ_t. AdaBoost can be implemented easily and has excellent efficiency of calculation. It has been reported that the combination of AdaBoost and a weak learner that tests the existence of a word

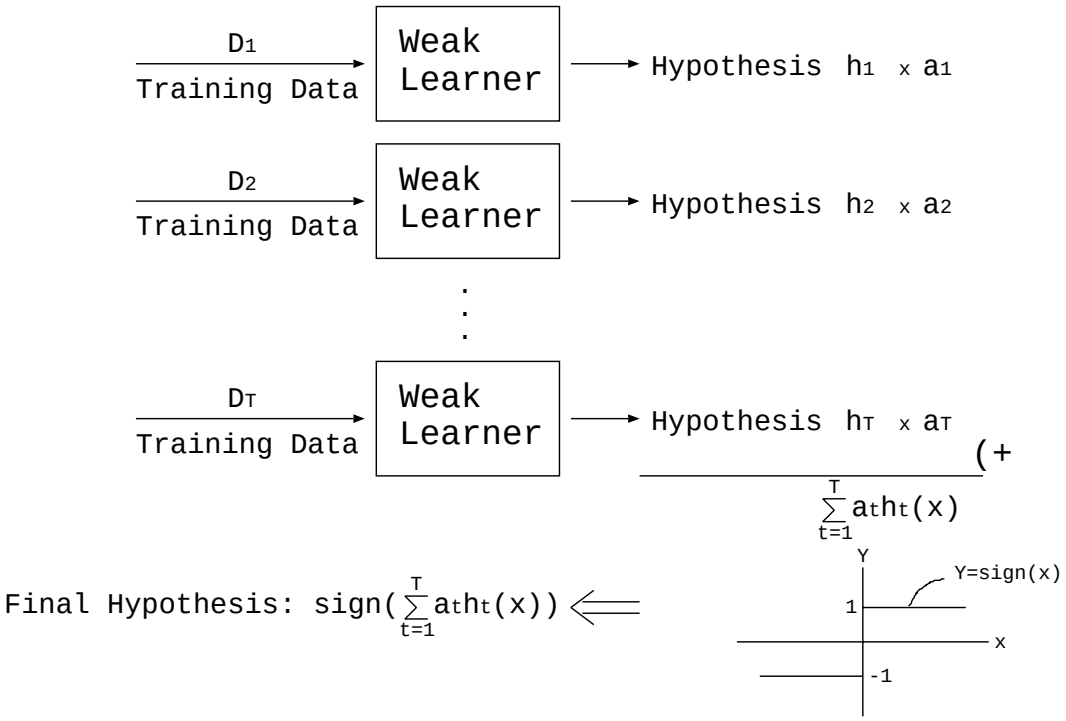

Figure 4.1: Boosting.

training examples: $(x_1, y_1), \ldots, (x_n, y_n) \quad y_i \in \{-1, 1\}$
D_t: distribution of weights at round t

Initializing $D_1(i) = 1/n$
for $t = 1, \ldots, T$

1. Calculating error ϵ_t using h_t and D_t:

$$\epsilon_t = \sum_{i=1}^{n} D_t(i)[h_t(x_i) \neq y_i]$$

2. Updating α_t:

$$\alpha_t = \frac{1}{2} \log \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$$

3. Updating D_t (Z_t is normalizing factor):

$$\begin{aligned} D_{t+1}(i) &= \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases} \\ &= \frac{D_t(i)}{Z_t} \times e^{-\alpha_t \times y_i \times h_t(x_i)} \end{aligned}$$

Final hypothesis H

$$H(x) = sign \left\{ \sum_t \alpha_t h_t(x) \right\}$$

Figure 4.2: AdaBoost algorithm

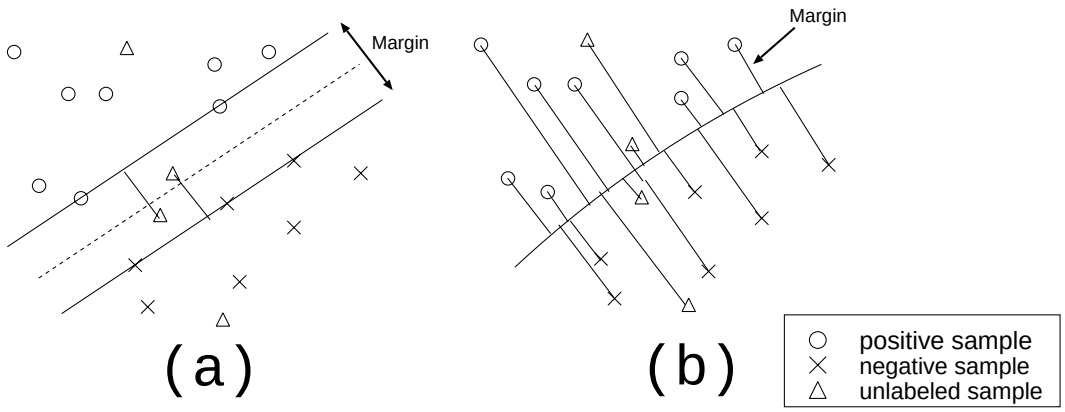

Figure 4.3: TSVM (a) and Transductive Boosting (b).

achieves the highest accuracy in the text classification task [Schapire00].

4.2 Transductive Methods and Text Categorization

4.2.1 Transductive method used in TSVM

I illustrate transductive SVM (TSVM) and novel transductive boosting in Fig. 4.3. The circles, crosses, and triangles denote positive training data, negative training data, and unlabeled test data, respectively. TSVM produces separated hyperplanes by finding the positive examples closest to the negative side and the negative examples closest to the positive side. The "margin" of TSVM is defined as the distance between two separated hyperplanes. TSVM chooses separated hyperplanes such that they maximize the margins while allowing classification errors below some fixed rate.

TSVM first produces a classifier using only the training data by SVM. All of the test data are given temporary classes by the classifier. Then classifiers are iteratively constructed by focusing only on the temporary labeled data. As the figure shows, if a pair of positive and negative examples can be found near the

classification boundary such that an exchange of their temporary classes decreases the classification error, they exchange their classes and re-learn the classifier by SVM. The exchange of classes and re-learning are repeated until there is no pair of test data for which labels have to be exchanged. Finally, a hyperplane fitting the distribution of the test data is obtained.

On the other hand, the margin in boosting is the sum of the distances between each example of every training data and the optimal classification bound, seen in the right side of Fig.4.3. Switching labels is effective in TSVM because the labels of samples near the separating hyperplanes are related to the margin deeply in SVM. However, boosting tries to maximize the average of all margins. My transductive method labels the single most reliable example in every round, and is described in the following sections.

4.2.2 Explanation of Boosting using Gradient Descent Methods

Recently, boosting has been regarded as an algorithm that chooses a weak hypothesis in the direction of the gradient descent of cost functions in a function space [Mason00].

Mason et al. stated that the AdaBoost algorithm corresponds to the algorithm that minimizes the cost function of the $\exp(-M)$ type in MarginBoost, where M is the margin. The cost function of the AdaBoost algorithm is

$$Cost(H(\boldsymbol{x})) = \frac{1}{m}\sum_{i=1}^{m}\exp(-y_i H(\boldsymbol{x}_i))$$

The cost is taken as an average of margins for a power function measure. As a result, a larger cost is given for more classification errors. Minimizing the value of this cost function corresponds to maximizing the margins and to minimizing the global errors as mentioned in the previous section.

Let us consider a class $\mathcal{H}$ of weak classifiers $h : X \to \{+1, -1\}$ (where X is the space of feature vectors). $lin(\mathcal{H})$ is the set of all linear combinations of the functions in $\mathcal{H}$. The inner product is defined by $< F, G > \stackrel{\text{def}}{=} \frac{1}{m}\sum_{i=1}^{m} F(\boldsymbol{x}_i)G(\boldsymbol{x}_i)$ for all $F, G \in lin(\mathcal{H})$. We define the inner product space $(\mathcal{X}, <, >)$ using the

inner product, where $\mathcal{X}$ is a linear space of functions that contains $lin(\mathcal{H})$ and $<,>$ stands for the inner product. Now suppose we have a function $H \in lin(\mathcal{H})$ and we wish to find a new $h \in lin(\mathcal{H})$ to add to H so that $Cost(H+\epsilon h)$ decreases for some small value of ϵ. We define the functional derivative of the cost function of H as

$$\nabla Cost(H)(\boldsymbol{x}) \stackrel{\text{def}}{=} \left.\frac{\partial Cost(H+\alpha 1_x)}{\partial \alpha}\right|_{\alpha=0}$$

where $1_{\boldsymbol{x}}$ is the indicator function of $\boldsymbol{x}$. The cost function can be expanded to the first order in ϵ,

$$Cost(H+\epsilon h) = Cost(H) + \epsilon < \nabla Cost(H), h >$$

Here, we can use the gradient descent method. That is, the greatest reduction in cost will occur for the h maximizing $- < \nabla Cost(H), h >$. After all, we wish to find h_{t+1}, α_{t+1} to minimize

$$\sum_{i=1}^{m} Cost(y_i H_t(\boldsymbol{x}_i) + y_i \alpha_{t+1} h_{t+1}(\boldsymbol{x}_i))$$

in the function space to get the weak hypothesis h_{t+1} at round t.

4.2.3 Transductive Boosting Method

Let us consider the minimization of the cost function mentioned in the former section within the framework of transductive methods. The cost function, including n test examples $\boldsymbol{x}_{m+1}, ..., \boldsymbol{x}_{m+n}$, is described as

$$Cost(H(\boldsymbol{x})) = \frac{1}{m+n}\{\sum_{i=1}^{m} \exp(-y_i H(\boldsymbol{x}_i)) + \sum_{j=m+1}^{m+n} \exp(-y_j^* H(\boldsymbol{x}_j))\}$$

where y_j^* is a temporary class label for $\boldsymbol{x}_j$. y_j^* is unknown, and the initial value of y_j^* is stored with 0. This algorithm aims to label $+1$(positive) or -1(negative) correctly for y_j^*. In the early rounds, the accuracy of the classifiers combining linearly weak learners is low because learning is not sufficient, unlike in SVM. If we labeled the classes for y_j^* based on these classifiers, incorrect labels would

be assigned to the data at a high ratio. Therefore, if boosting were performed with this large amount of wrongly labeled test data, an incorrect gradient descent would be obtained. Moreover, the accuracy of the final classifier would be low, whereas we should perform labeling for the test data at a high accuracy.

Therefore, in every round, we label the class for only the most reliable test data. We label this class assuming that the ratio of positive and negative examples is the same as the ratio of positive and negative examples in the training data. For the test data, we add (step 2) and (step 7) to the AdaBoost algorithm in section 4.1 as follows.

(step 1) Given m training samples $(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_m, y_m)$ with feature vectors $\boldsymbol{x}_1, ..., \boldsymbol{x}_m$ and classification classes $y_1, ..., y_m$ ($+1$ for positive, -1 for negative).
Initialize the weights for training data $D_1(i) = \frac{1}{m}$, where $i = 1, ..., m$.

(step 2) Given n test samples $(\boldsymbol{x}_{m+1}, y^*_{m+1}), ..., (\boldsymbol{x}_{m+n}, y^*_{m+n})$ with feature vectors $\boldsymbol{x}_{m+1}, ..., \boldsymbol{x}_{m+n}$ and classification classes $y^*_{m+1}, ..., y^*_{m+n}$ whose initial values are 0.
Initialize the weights for test data $D_1(j) = 0 \quad (j = m+1, ..., m+n)$.

(step 3) For $t = 1, ..., T$, repeat (step 4)-(step 7).

(step 4) A weak learner learns labeled data (that is, $y_i \neq 0$) under weight D_t, and we get weak hypothesis $h_t(\boldsymbol{x})$, which outputs $+1$ for a positive evaluation for $\boldsymbol{x} = \boldsymbol{x}_i$; -1 for a negative evaluation.

(step 5) Calculate parameter α_t based on $\alpha_t = \frac{1}{2}\ln(\frac{1-\epsilon_t}{\epsilon_t})$, where ϵ_t denotes weighted error rates calculated based on $\epsilon_t = \sum_{i:h_t(\boldsymbol{x}_i)\neq y_i} D_t(i)$.

(step 6) Update every weight of the training data based on

$$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(\boldsymbol{x}_i))}{Z_t}$$

where Z_t, the normalized factor for $\sum_{i=1}^{m} D_{t+1}$, equals 1.

(step 7) Let m^+ be the number of training data with a positive class, $n_{labeled}$ be the number of test data with an already labeled class, and $n^+_{labeled}$ be the number of test data with an already labeled positive class.

(i) If $n_{labeled} = 0$ or $m^+/m \geq n^+_{labeled}/n_{labeled}$, then

we choose the test sample j that maximizes

$$H(\boldsymbol{x}_j) = \sum_{k=1}^{t} \alpha_k h_k(\boldsymbol{x}_j)$$

in the test data such that $y_j = 0$ and give $y_j = +1$ and $D_{t+1}(j) = \epsilon$ (ϵ is a small value, for example, $\epsilon = 0.01$). Then, we update the weight of the data already labeled as follows,

$$D_{t+1}(i) = \frac{D_t(i)}{Z'_t}$$

Here, Z'_t is the normalizing factor such that the sum of the data without j equals $1 - \epsilon$.

(ii) If $n_{labeled} \neq 0$ and $m^+/m < n^+_{labeled}/n_{labeled}$, then we choose the test sample j that minimizes

$$H(\boldsymbol{x}_j) = \sum_{k=1}^{t} \alpha_k h_k(\boldsymbol{x}_j)$$

in the test data such that $y_j = 0$ and give $y_j = -1$ and $D_{t+1}(j) = \epsilon$. Then, we update the weight of the data already labeled as follows,

$$D_{t+1}(i) = \frac{D_t(i)}{Z'_t}$$

where Z'_t is the normalizing factor such that the sum of the data without j equals $1 - \epsilon$.

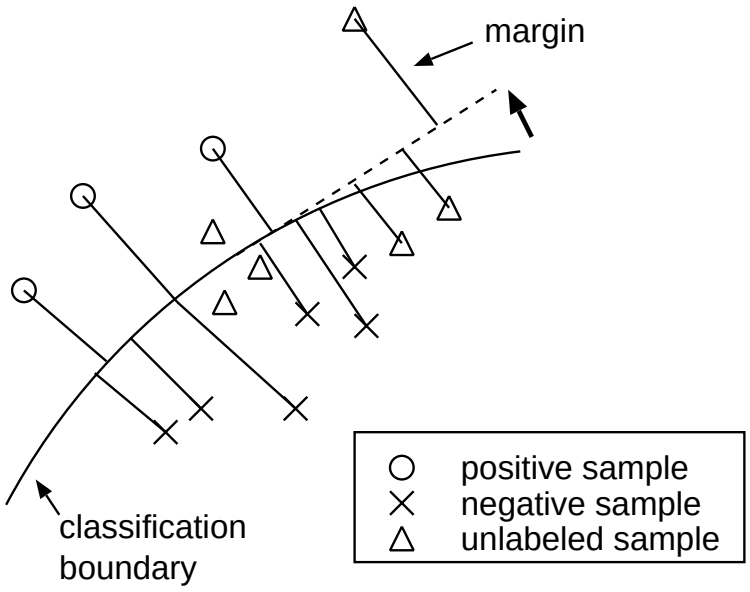

Figure 4.4: Effect of unlabeled samples.

(step 8) Return the final hypothesis, merging weak hypotheses linearly as

$$H(\boldsymbol{x}) = \sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})$$

(step 2) is for initializing the labels and weights of the test samples. (step 7) is for performing labeling such that the ratio of positive and negative test samples always equals that of the training data. A small value is given for the weight for the test samples selected in this step because the reliability of the labels is lower than that of the training samples.

Taking these steps can produce classifiers that minimize the value of the cost function because we can select data to lower the probability of labeling y_j^* wrongly and choose data to maximize $y_j^* H(\boldsymbol{x}_j)$ at a round in a hill-climbing manner with the term of $\exp(-y_j^* H(\boldsymbol{x}_j))$ in the cost function (Figure 4.4).

Another option of our algorithm is that we can use the algorithm after weak learners are produced using only labeled data several times. We can also choose to label several test examples per round by executing (step 7) several times per

round, although the algorithm would need more iterations than the number of test examples.

4.3 Experimental Results

4.3.1 Experimental Settings

My experiments were conducted using the RWCP corpus. I made binary classifiers for each of the categories, which determined whether a sample belonged to the category or not. The total number of articles used for both the training and the test was 1,000. Throughout my experiments, 1,000 words with high mutual information were used as the input feature space because they were keywords that sufficiently characterized the classes.

The iteration T was 1,000 in all of our experiments. The value of 0.01 was used for ϵ in transductive AdaBoost. The value for each feature was a binary value, which indicated whether the word appeared in a document or not. This binary value was employed to study the pure effects of each word.

4.3.2 Relation between Number of Training Data and Accuracy

First, when the ratio of positive to negative examples in the training data was the same as that in the test data, I carried out experiments on text classification using the transductive AdaBoost algorithm with a one-depth decision tree as a weak learner. For comparison, I also performed experiments using the standard AdaBoost with a one-depth decision tree (BoosTexter) [Schapire00], SVM, and TSVM. I increased the number of training data from 75 to 1,000 and then I classified the 1,000 test data. Figure 4.5 shows the results (F-measure) of the averages among ten categories.

Accuracy for the distribution of the 1,000 test data increases significantly when using the transductive method. In particular, when there is a small number of training data, the classification accuracy grows dramatically. When the

Table 4.1: F-measure for the number of training data (Transductive Boosting).

Category \ # of training data	75	100	200	500	750	1000
sports	0.642	0.726	0.766	0.875	0.901	**0.903**
criminal law	0.600	0.571	0.663	0.656	0.743	**0.750**
government	0.723	0.560	0.622	0.689	**0.727**	0.722
educational system	0.459	0.624	0.661	0.675	0.762	**0.778**
traffic	0.495	0.493	0.500	0.638	0.680	**0.698**
military affairs	0.507	0.561	0.688	0.748	0.754	**0.781**
international relations	0.429	0.396	0.363	0.558	0.508	**0.560**
communications	0.493	0.523	0.641	0.612	**0.703**	0.692
theater	0.583	0.749	0.756	0.795	0.857	**0.862**
agriculture	0.750	0.817	0.831	0.805	0.761	**0.853**
avg.	0.569	0.602	0.649	0.705	0.740	**0.760**

number of training data is 75, the F-measure for boosting is 0.438 and that for transductive boosting is 0.569, which is a difference of 0.131. The accuracy of the classification using only 75 training data in the transductive method nearly equals the accuracy of that using 200 training data in the inductive method. This indicates that the transductive method is useful for improving the classification accuracy for a small number of training examples. Compared with SVM and TSVM with a linear kernel function, the classification using transductive boosting is slightly weaker than TSVM but exceeds SVM. The increase in the accuracy from boosting to transductive boosting decreases, while classification accuracy increases monotonically without 1,000 training data in boosting. This might be explained by the fact that when the number of training data is 1,000, the distribution of the test data is similar to that of the training data. The executing speed was from 10 seconds to 1 minute for SVM and TSVM, and several minutes for boosting and transductive boosting using a Linux server with an Athlon 1GHz cpu.

I show the details of the results for every category in Tables 4.1 and 4.2. The boldface numbers denote the best accuracy in each category. The categories with high classification accuracies sometimes show decreases using the

Table 4.2: F-measure for the number of training data (Boosting).

Category \ # of training data	75	100	200	500	750	1000
sports	0.675	0.681	0.826	0.867	0.891	**0.912**
criminal law	0.561	0.402	0.649	0.664	0.681	**0.723**
government	0.607	0.524	0.580	0.683	**0.692**	0.670
educational system	0.287	0.525	0.563	0.646	0.667	**0.714**
traffic	0.514	0.510	0.493	0.647	**0.658**	0.579
military affairs	0.216	0.321	0.550	**0.728**	0.686	0.628
international relations	0.324	0.317	0.233	0.428	**0.490**	0.329
communications	0.119	0.220	0.528	**0.576**	0.561	0.559
theater	0.385	0.645	0.767	0.693	0.800	**0.813**
agriculture	0.690	0.643	0.734	0.855	**0.864**	0.850
avg.	0.438	0.479	0.592	0.679	**0.699**	0.678

transductive method, e.g., 0.912 to 0.903 for sports using 1,000 training data. However, the categories of educational system, military affairs, international relations, and communications, which have low classification accuracies using the inductive method, show dramatic increases using the transductive method.

Let us move on to the second experimental condition. I changed the ratio of positive to negative examples in the training data from 1:1 to 1:4 (the original data distribution depicted in Fig. 4.5 is 1:9). The total number of training examples was changed from 20 to 150. The results (averages of the ten categories) are shown in Fig. 4.6. The performance of transductive boosting is almost the same as that of TSVM and sometimes outperforms TSVM when the training and test distributions are significantly distinct, for example, $N_p : N_n = 1 : 1$ and 150 training samples. This indicates that the good performance of TSVM largely depends on the ratio of positive and negative examples.

4.4 Summary

In natural language tasks like text categorization, we usually have an enormous amount of unlabeled data in addition to a small amount of labeled data. We present here a transductive boosting method for text categorization in order to make efficient use of the large amount of unlabeled data. Our experiments show that the transductive method outperforms conventional boosting techniques that employ only labeled data. We have proposed a transductive boosting method for text classification problems. We carried out experiments in which we varied the number of training data and compared the transductive method to the standard AdaBoost, SVM, and TSVM methods. The results indicated that the transductive boosting method can improve the performance of text categorization in situations where we have an enormous number of unlabeled data in addition to a small number of labeled training data. When the ratio of positive to negative examples in the training data differs from that in the test data, the advantage of our transductive boosting method in terms of performance is significant. This suggests that our transductive boosting method might be particularly useful when we do not know the ratio of positive to negative examples in the test data. Overall, our results show that the transductive approach is effective for a variety of learning methods and potentially promising for other applications.

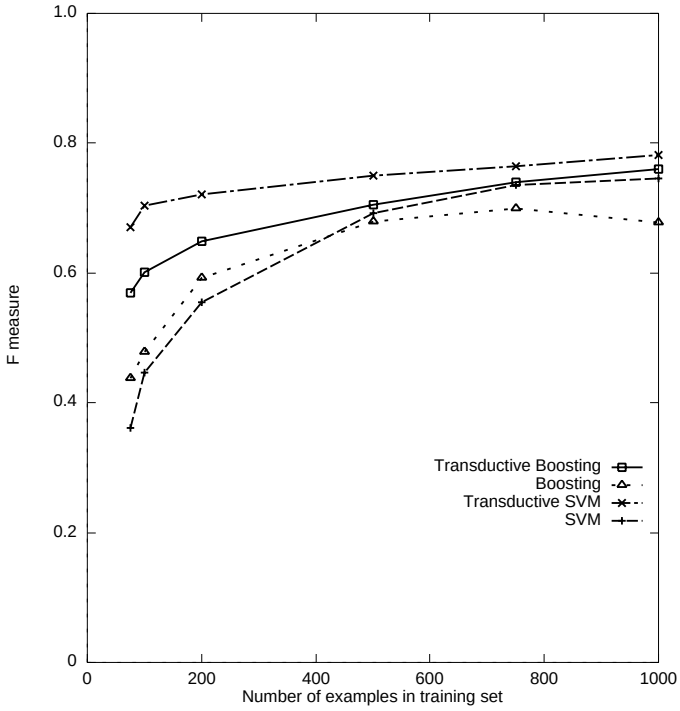

Figure 4.5: F-measure and the number of training data.

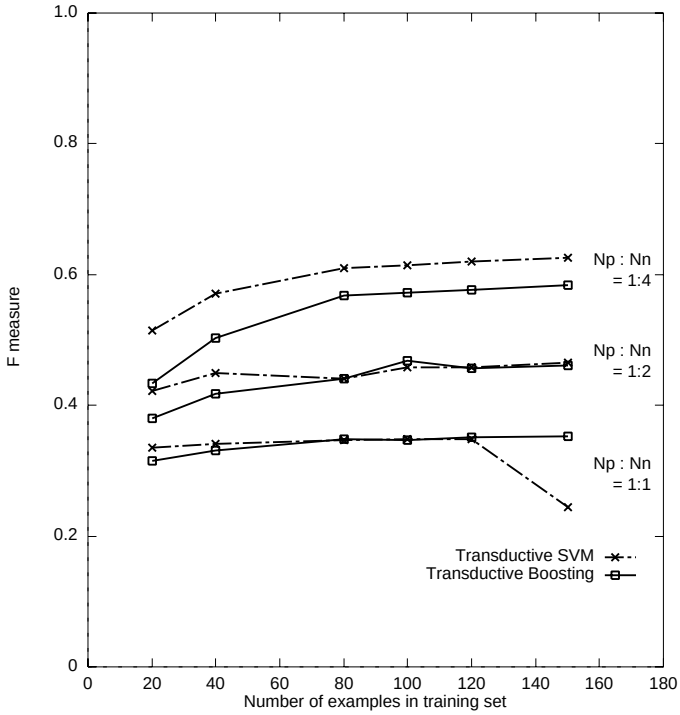

Figure 4.6: F-measure and the ratio of positive and negative examples in the training data.

Chapter 5

Text Categorization Using Extended Tied Mixture Models

5.1 Extended Tied Mixture Models

I assume that the training data set D consists of the labeled subset $D_l = \{(\boldsymbol{x}_j^l, y_j)\}_{j=1}^n$ and the unlabeled subset $D_u = \{\boldsymbol{x}_j^u\}_{j=n+1}^N$, where $\boldsymbol{x}_j^l$ and $\boldsymbol{x}_j^u$ are d-dimensional feature vectors and $y_j \in \{1, \ldots, Y\}$ is the class label for the Y class problem. In the conventional setting [McLachlan87][Shahshahani94][Nigam00], it is further assumed that each class corresponds to a single component density of a mixture model, $p(\boldsymbol{x}|\Theta) = \sum_{k=1}^Y \alpha_k f_k(\boldsymbol{x}|\theta_k)$. Here, α_k is a non-negative mixing parameter satisfying $\sum_{k=1}^Y \alpha_k = 1$. Therefore, in this case, the class label index is equivalent to the component index. The unknown parameter $\Theta = \{\alpha_k, \theta_k\}_{k=1}^Y$ is then estimated by maximizing the joint-data log-likelihood function given by Eq. (5.1) using the EM algorithm [Dempster77].

$$L(\Theta) = \sum_{\boldsymbol{x}_j^l \in D_l} \log \alpha_{y_j} f_{y_j}(\boldsymbol{x}_j^l|\theta_{y_j}) + \sum_{\boldsymbol{x}_j^u \in D_u} \log \sum_{k=1}^Y \alpha_k f_k(\boldsymbol{x}_j^u|\theta_k) \tag{5.1}$$

On the other hand, Miller et al. [Miller97] generalized this model by introducing a new non-negative parameter β_{ky}, representing the probability that the kth component density belongs to class y, so that one component density can

be probabilistically assigned to multiple classes. In this case, the first term on the right hand side of Eq. (5.1) is modified by $\sum_{\boldsymbol{x}_j^l \in D_l} \log \sum_{k=1}^{K} \alpha_k \beta_{ky_j} f_k(\boldsymbol{x}_j^l|\theta_k)$. They experimentally showed the usefulness of this extension.

In the ETM model, the feature vectors are assumed to be generated from the hierarchical mixture density given by

$$p(\boldsymbol{x}|\Theta) = \sum_{y=1}^{Y} \omega_y \sum_{k=1}^{K} c_{yk} f_k(\boldsymbol{x}|\theta_k) \tag{5.2}$$

where f_k denotes the kth component density parameterized by θ_k, with non-negative mixing parameter (Note that $\sum_k c_{yk} = 1$ in this model, while $\sum_y \beta_{ky} = 1$ in Miller's model) c_{yk}, which corresponds to $P(k|y)$. ω_y is a prior for class y satisfying $\sum_y \omega_y = 1$. $\Theta = \{\omega_y, c_{yk}, \theta_k\}_{k=1,y=1}^{K,Y}$. The probability model of class y is represented by a mixture model $p(\boldsymbol{x}|y, \theta_y) = \sum_k c_{yk} f_k(\boldsymbol{x}|\theta_k)$, while in Miller's model, $p(\boldsymbol{x}|y, \theta_y) = \sum_k \beta_{ky} \alpha_k f_k(\boldsymbol{x}|\theta_k) / \sum_{k'} \beta_{k'y} \alpha_{k'}$. Clearly, since each class model shares K component densities, labeled and unlabeled data can be simultaneously utilized in the ETM model and the representation ability is potentially higher than in McLachlan's basic model. For simplicity, I used the maximum-likelihood (ML) algorithm here. In the case of the ML algorithm, the log-likelihood function is given by Eq. (5.3).

$$L(\Theta|D) = \sum_{\boldsymbol{x}_j^l \in D_l} \log \sum_{k=1}^{Y} \alpha_k f_k(\boldsymbol{x}_j^l|\theta_k) + \sum_{\boldsymbol{x}_j^u \in D_u} \log \sum_{k=1}^{Y} \alpha_k f_k(\boldsymbol{x}_j^u|\theta_k) \tag{5.3}$$

If we specify the component density f_k in Eq. (5.2) with a multinominal distribution over words, the model can be regarded as an extension of the naive-Bayes model [Nigam00]. At that time, f_k is,

$$f_k(\boldsymbol{x}|\theta_k) = \prod_{t=1}^{T} \theta_{t|k}^{\boldsymbol{x}_t} \tag{5.4}$$

Here, t is the index of feature words and T is the number of different words. Since this model presents each class (text category) by a mixture of multinominal distributions, as shown later, it can achieve better classification performance than the original naive-Bayes model.

5.2 Bayesian Learning Algorithm

To overcome the overfitting problem associated with the maximum likelihood approach, a variational Bayesian (VB) algorithm for the ETM model is proposed [Ueda01]. In general, as described in [Attias00][Ueda00], the VB approach tries to maximize the objective function given by

$$\mathcal{F}_K[q] = \Big\langle \log \frac{p(D, Z|\theta, K)}{q(Z|K, D)} \Big\rangle_{Z,\theta} + \Big\langle \log \frac{p(\theta, K)}{q(\theta|K, D)} \Big\rangle_{\theta} \tag{5.5}$$

with regard to the variational posteriors $q(Z|K, D)$, $q(\theta|K, D)$ and the number of mixture components K. The notation $\langle f \rangle_{Z,\theta}$ denotes the expectation of f with regard to the variational posterior distribution $q(Z, \theta|m, D)$ for simplicity.

Let Z_l (Z_u) denote the latent variable set corresponding to D_l (D_u). In the ETM model, the class index and component index are latent for the unlabeled sample, while the component index is latent for the labeled sample. Let $p(D, Z|\theta, K)$ be a complete data likelihood function with parameter θ and model structure K (number of mixture components). Now, using the i.i.d. assumption for the labeled and unlabeled samples, we have

$$p(D, Z|\theta, K) = p(D_l, Z_l|\theta, K)p(D_u, Z_u|\theta, K) \tag{5.6}$$

$$q(Z|K, D) = q(Z_l|K, D)q(Z_u|K, D) \tag{5.7}$$

Substituting Eqs. (5.6) and (5.7) into Eq. (5.5), we obtain the modified objective function for the partial nonrandom classification:

$$\begin{aligned}\mathcal{F}_K[q] = &\Big\langle \log \frac{p(D_l, Z_l|\theta, K)}{q(Z_l|K, D)} \Big\rangle_{Z_l,\theta} + \Big\langle \log \frac{p(D_u, Z_u|\theta, K)}{q(Z_u|K, D)} \Big\rangle_{Z_u,\theta} \\ &+ \Big\langle \log \frac{p(\theta, K)}{q(\theta|K, D)} \Big\rangle_{\theta}\end{aligned} \tag{5.8}$$

Once we have the objective function, according to the general procedure of the VB approach [Attias00], we can derive the following iterative two-phase VB algorithm for the ETM model to estimate variational posteriors [Ueda01]:

Updating posteriors over latent variables:

$$q(k|d^l, K, D)^{(t+1)} \propto \exp\Big\{\langle \log c_{yk} \rangle_{c_{yk}^{(t)}} + \langle \log f_k(\boldsymbol{x}^l|\theta_k) \rangle_{\theta_k^{(t)}}\Big\} \tag{5.9}$$

$$q(k, y|d^u)^{(t+1)} \propto \exp\Big\{\langle \log \omega_y \rangle_{\omega_y^{(t)}} + \langle \log c_{yk} \rangle_{c_{yk}^{(t)}} + \langle \log f_k(\boldsymbol{x}^u|\theta_k) \rangle_{\theta_k^{(t)}}\Big\} \tag{5.10}$$

Updating posteriors over parameters:

$$q(\omega_y|K, D)^{(t+1)} \propto P(\omega_y)\omega_y^{\sum_{k=1}^{K}(\sum_{d^l \in D_{l=y}} q(k|d^l,K,D)^{(t+1)} + \sum_{d^u \in D_u} q(k,y|d^u,K,D)^{(t+1)})} \tag{5.11}$$

$$q(c_{yk}|K, D)^{(t+1)} \propto P(c_{yk})c_{yk}^{(\sum_{d^l \in D_{l=y}} q(k|d^l,K,D)^{(t+1)} + \sum_{d^u \in D_u} q(k,y|d^u,K,D)^{(t+1)})} \tag{5.12}$$

$$q(\theta_k|K, D)^{(t+1)} \propto p(\theta_k) \exp\Big\{ \sum_{d^l \in D_l} q(k|d^l, K, D)^{(t+1)} \log f_k(\boldsymbol{x}^l|\theta_k) + \sum_{d^u \in D^u} q(k, y|d^u, K, D)^{(t)} \log f_k(\boldsymbol{x}^u|\theta_k)\Big\} \tag{5.13}$$

Note that the notation $\langle f \rangle_{s^{(t)}}$ denotes the expectation of f with regard to $q(s|\cdot)^{(t)}$ for simplicity. $P(\omega_y)$, $P(c_{yk})$ and $p(\theta_k)$ are priors.

In the Bayesian setting, an unknown sample $\boldsymbol{x}^*$ is classified using the predictive posterior classification probability defined by

$$P(y|\boldsymbol{x}^*, K^*, D) = \overline{\omega}_y p(\boldsymbol{x}^*|y, K^*, D) \Big/ \sum_{y'=1}^{Y} \overline{\omega}_{y'} p(\boldsymbol{x}^*|y, K^*, D) \tag{5.14}$$

where $\overline{\omega}_y$ is the expectaion value of ω_y, and $p(\boldsymbol{x}^*|y, K^*, D)$ is the predictive density for class y and is defined by

$$p(\boldsymbol{x}^*|y, K^*, D) = \int p(\boldsymbol{x}^*|y, K^*, \Theta) q(\Theta|K^*, D) d\Theta \tag{5.15}$$

If we assume that the component density of the ETM model belongs to the exponential family and the priors are conjugate priors, we can analytically compute the integral.

5.3 Active Labeling

McCallum et al. have recently proposed a Query-by-Committee (QBC)-based active labeling method and successfully applied it to text classification [McCallum98]

using a mixture model [Nigam00]. The method first creates several sets of labeled training samples according to the currently estimated model parameter distribution in order to train a committee of classifier variants. Next, each of the trained committee members classifies unlabeled data and then the disagreements among their classifications are measured. Finally, the sample with the largest disagreement value is selected as a labeling request and the true (not estimated) class label is assigned to the sample. The newly labeled sample is included in the training data and the classifier is retrained to obtain a new classifier (model parameter distribution). These procedures are repeatedly performed until a certain classification performance is reached.

Intuitively, this method tends to select samples near the current class boundaries since the variances of the class posterior probabilities near the class boundaries are large. However, when the current class boundary is far from the true ones, the QBC-based method does not work well.

In addition, this method requires too many extra computational operations, e.g., resampling, training several committee members, and the measuring of disagreements. To overcome these problems, I use a new active labeling method that is quite efficient and works well even when the current class boundaries are far from the true ones. According to the predictive posterior classification probabilities given by Eq. (5.14), it can be assumed that a class label y for $\boldsymbol{x}$ is determined with probability $\pi_y(\boldsymbol{x}) \equiv P(y|\boldsymbol{x}, K^*, D)$ from a multinominal distribution, $\text{Mult}_Y(1, \boldsymbol{\pi}(\boldsymbol{x}))$, consisting of one draw on Y categories (classes) with probabilities $\boldsymbol{\pi}(\boldsymbol{x}) = (\pi_1(\boldsymbol{x}), \ldots, \pi_Y(\boldsymbol{x}))$. In other words, the classification probability for class y can generally be regarded as deriving from a Dirichlet random variable with parameters $\{\pi_1, \ldots, \pi_Y\}$. This is because the Dirichlet distribution is a conjugate distribution of the multinominal parameters.

Let $y_1(y_2)$ denote the class index with the largest (the second largest) posterior classification probability for $\boldsymbol{x}$ (i.e., $y_1 = \arg\max_y \pi_y(\boldsymbol{x})$, $y_2 = \arg\max_{y \neq y_1} \pi_y(\boldsymbol{x})$). Therefore, considering that the mean value and standard deviation for $\boldsymbol{x}$ are then given by $\pi_y(\boldsymbol{x})$ and $\sqrt{0.5\pi_y(\boldsymbol{x})(1 - \pi_y(\boldsymbol{x}))}$ and class boundaries are formed by the intersection of functions $\pi_{y_1}(\boldsymbol{x})$ and $\pi_{y_2}(\boldsymbol{x})$, we define the following a-confidence

region:

$$R_c = \Big\{ \boldsymbol{x} \mid \pi_{y_1}(\boldsymbol{x}) - a\sqrt{0.5\pi_{y_1}(\boldsymbol{x})(1-\pi_{y_1}(\boldsymbol{x}))} \\ \leq \pi_{y_2}(\boldsymbol{x}) + a\sqrt{0.5\pi_{y_2}(\boldsymbol{x})(1-\pi_{y_2}(\boldsymbol{x}))} \Big\}$$

Here, the constant a is usually set to one and can be modified depending on the number of pooled unlabeled samples. We set $a = 1$ in our experiments.

Moreover, the flatter $\pi_{y_1}(\boldsymbol{x})$ and $\pi_{y_2}(\boldsymbol{x})$ are, the less unreliable the intersection of $\pi_{y_1}(\boldsymbol{x})$ and $\pi_{y_2}(\boldsymbol{x})$ becomes. To measure this, I simply use the gradient information defined by $G(\boldsymbol{x}) = ||\partial\pi_{y_1}(\boldsymbol{x})/\partial\boldsymbol{x}|| + ||\partial\pi_{y_2}(\boldsymbol{x})/\partial\boldsymbol{x}||$. Here, the symbol $||\cdot||$ denotes a vector norm. Based on these, in the active labeling method, we select an $\boldsymbol{x} \in D^u$ that belongs to R_c and minimizes $G(\boldsymbol{x})$.

5.4 Experiments

Next, using multinominal distribution as a component density, I applied the ETM model to "comp.*" (five class problem: a subset of the 20 Newsgroups data set). For simplicity, I used the ML algorithm here. The dimensionality was reduced to 200 from about 60,000 by the mutual information criterion. Fifty labeled samples, 2,482 unlabeled samples, and 1,000 test samples were used. The obtained error rate was 31.5% for the mixed data using the ETM model with the optimal number of components ($K^* = 10$). In contrast, the results from using the original model [Nigam00] were 53.0% for the labeled data set and 32.1% for the mixed data. In naive-Bayes, by taking unlabeled data into consideration, the accuracy rose 20.9% and the accuracy rose 0.6 % more by taking into consideration the 10 distribution component which exists potentially using ETM model.

The active labeling method was compared with the QBC-based one by using the ETM model and another "comp.*" data set (sample size was the same as in the experiment above). Fig. 5.1 shows the trajectories of recognition error rates obtained by iteratively adding one sample by the two active labeling methods. Fifty labeled samples, 2,482 unlabeled samples, and 1,000 test samples were used. one unlabeled sample was selected and labeled per one active labeling. Ten active labeling was done. Where the QBC-method could not reduce the error rate, the

proposed method could reduce from 42.1% to 33% drastically. This indicates the proposed method is effective when we have only the small amount of training set.

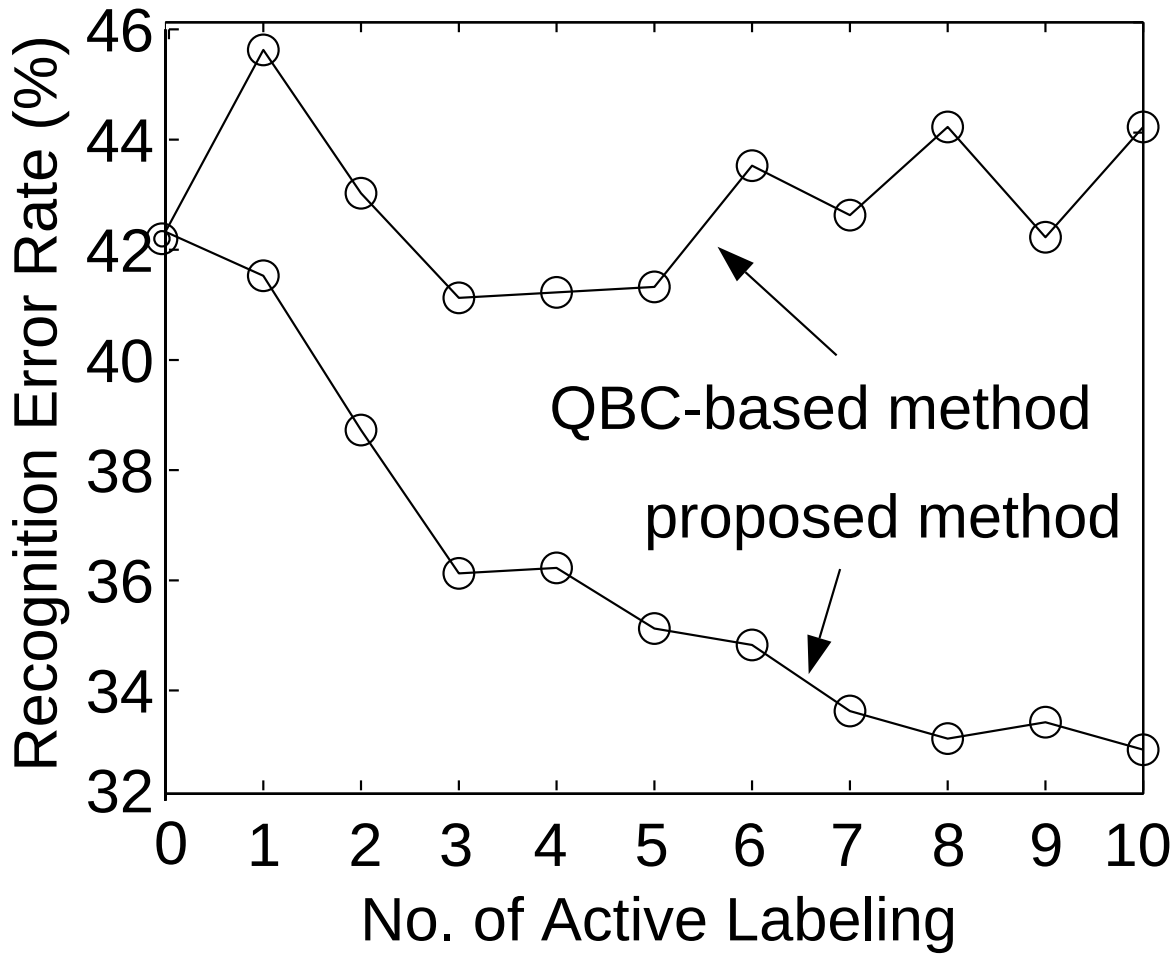

Figure 5.1: Recognition error rates of QBC-based method and proposed method

5.5 Summary

The results above indicate that the ETM model can effectively utilize the unlabeled samples to improve classification performance. Moreover, the active labeling method works well. In this chapter, to design a statistical classifier using unlabeled data, I focused on an approach based on a mixture model.

Chapter 6

Conclusions

6.1 Summary

This thesis discussed the following three points related to text classification using machine learning.

1. How to perform a highly precise classification by using a large number of word attributes (Chapter 3).
2. How to utilize the distribution of unlabeled examples for high precision when there are few labeled training examples (Chapter 4).
3. How to achieve a highly precise and efficient classification by assuming the existence of sub-categories and using active labeling (Chapter 5).

In Chapter 3, text classification using a support vector machine was described. Many word features are generally required to perform highly precise text classification by machine learning. However, when many word features are used as the input of conventional learning techniques, they overfit the training data, and the classification accuracy for unknown data decreases.

Therefore, it was necessary to reduce the dimension of features used in learning by selecting a moderate number of features which contain much information. However, it was difficult for the classifier to learn with a sufficiently high accuracy only with several hundred features. Consequently, text classification was

performed by using a support vector machine, which is a new machine learning technique developed to avoid overfitting. I evaluated how feature selection affects performance of SVM.

Classification accuracy is very high without feature selection and the classifier created by SVM with part-of-speech filtering has the highest accuracy.

In Chapter 4, text classification using transductive boosting was described. Large margin classifiers such as support vector machines and boosting algorithms are effective in generating classifiers with a high classification accuracy when training data is fully available. However, a large size of training data is not always available in many cases because we have to label the training data by hand and the cost of doing this is high.

Transduction is a method that takes into consideration the distribution of the unknown data for which a classification label is not given. I investigated transduction as a solution for text classification. Specifically, I proposed how to use the transductive method with the boosting algorithm, AdaBoost.

In Chapter 5, text classification using the Extended Tied Mixture model was described. We can often assume potential sub-categories in a given category, for example, the "sports" category could include the sub-categories "baseball" and "soccer." I described a method that takes into account such an assumption. Furthermore, when the size of the training data is limited, adding a small number of good unknown data is important to effectively improve the classification accuracy. To mechanically chose such good examples the "active labeling" technique should be applied. Such a technique could efficiently create a highly precise classifier by assigning the categories for a small number of data by hand. I presented such a method in that chapter.

What we could conclude from those results are the following: If we have a sufficient amount of labeled samples and it is modeled as a binary-class problem, it is better to use support vector machines with part-of-speech filtering. If we have only a small amount of labeled samples, transduction for large margin classifiers is effective. Furthermore, when we have a few labeled samples, active labeling for ETM models is useful for improving the classification performance.

6.2 Future Directions

Although all of the text classification methods introduced in this thesis are flat or two-level layered classifications, it is natural that the structure of categories forms a network such as the link structure of the Internet. Furthermore, although I assumed only static structure of categories in this thesis, there are many cases of changing the structure of categories and updating texts in the categories. I intend to aim my research toward machine learning for the network structure of texts and online learning for dynamic structures.

Bibliography

[Attias00] Attias, H.: A Variational Bayesian Framework for Graphical Models, *Proc. of the 12th Advances in Neural Information Processing Systems (NIPS-99)*, pp. 209–215 (2000).

[CMUdata] 20 Newsgroups Data Set, http://www.cs.cmu.edu/textlearning.

[Cohn96] Cohn, D., Ghahramani, Z. and Jordan, M.: Active Learning with Statistical Models, *Journal of Artificial Intelligence Research*, Vol. 4, pp. 129–145 (1996).

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

[Cover91] Cover, T. and Thomas, J.: *Elements of Information Theory*, John Wiley & Sons (1991).

[Dempster77] Dempster, A. P., Laird, N. M. and Rubin, D. B.: Maximum Likelihood from Incomplete Data via the EM Algorithm, *Journal of Royal Statistical Society, Series B*, Vol. 39, pp. 1–38 (1977).

[Dumais98] Dumais, S., Platt, J., Heckerman, D. and Sahami, M.: Inductive Learning Algorithm and Representation for Text Categorization, *Proc. of the Seventh International Conference on Information and Knowledge Management (CIKM-98)*, pp. 148–155 (1998).

[Freund97] Freund, Y. and Schapire, R.: A Decision-theoretic Generalization of On-line Learning and an Application to Boosting, *Journal of Computer and System Sciences*, Vol. 55, No. 1, pp. 119–139 (1997).

[Haruno99] Haruno, M., Shirai, S. and Ooyama, Y.: Using Decision Trees to Construct a Practical Parser, *Machine Learning*, Vol. 34, pp. 131–150 (1999).

[Joachims98] Joachims, T.: Text Categorization with Support Vector Machines: Learning with Many Relevant Features, *Proc. of 10th European Conference on Machine Learning (ECML-98)*, pp. 137–142 (1998).

[Joachims99] Joachims, T.: Transductive Inference for Text Classification using Support Vector Machines, *Proc. of the 16th International Conference on Machine Learning (ICML'99)*, pp. 200–209 (1999).

[Lewis94] Lewis, D. and Ringuette, M.: A Comparison of Two Learning Algorithms for Text Categorization, *Proc. of Third Annual Symposium on Document Analysis and Information Retrieval*, pp. 81–93 (1994).

[Mainichi95] *Mainichi Newspaper 94 CD-ROM*, Nichigai Associates Co. (1995).

[Mason00] Mason, L., Baxter, J., Bartlett, P. and Frean, M.: Boosting Algorithms as Gradient Descent, *Proc. of the 12th Advances in Neural Information Processing Systems (NIPS-99)*, pp. 512–518 (2000).

[Matsumoto97] Matsumoto, Y., Kitauchi, A., Yamashita, T., Hirano, Y., Imaichi, O. and Imamura, T.: *Japanese Morphological Analysis System Chasen Manual* (1997), NAIST Technical Report NAIST-IS-TR97007.

[McCallum98] McCallum, A. K. and Nigam, K.: Employing EM and Pool-based Active Learning for Text Classification, *Proc. of the 15th International Conference on Machine Learning (ICML'98)*, pp. 350–358 (1998).

[McLachlan87] McLachlan, G. J. and Basford, K. E.: *Mixture Models*, Marcel Dekker (1987).

[Miller97] Miller, D. J. and Uyar, H. S.: A Mixture of Experts Classifier with Learning Based on Both Labelled and Unlabeled Data, *Proc. of the Ninth Advances in Neural Information Processing Systems (NIPS-96)*, pp. 571–577 (1997).

[Nigam00] Nigam, K., McCallum, A., Thrun, S. and Mitchell, T.: Text Classification from Labeled and Unlabeled Documents using EM, *Machine Learning*, Vol. 39, pp. 103–134 (2000).

[Osuna98] Osuna, E., Freund, R. and Girosi, F.: Training Support Vector Machines: An Application to Face Detection, *Proc. of Computer Vision and Pattern Recognition '97*, pp. 130–136 (1998).

[Quinlan93] Quinlan, J.: *C4.5: Programs for Machine Learning*, Morgan Kaufmann (1993).

[Salton88] Salton, G. and Buckley, C.: Term Weighting Approaches in Automatic Text Retrieval, *Information Processing and Management*, Vol. 24, No. 5, pp. 513–523 (1988).

[Schapire00] Schapire, R. E. and Singer, Y.: BoosTexter: A Boosting-based System for Text Categorization, *Machine Learning*, Vol. 39, pp. 135–168 (2000).

[Shahshahani94] Shahshahani, B. and Landgrebe, D.: The Effect of Unlabeled Samples in Reducing the Small Sample Size Problem and Mitigating the Hughes Phenomenon, *IEEE Trans. on Geoscience and Remote Sensing*, Vol. 32, No. 5, pp. 1087–1095 (1994).

[Taira99] Taira, H. and Haruno, M.: Feature Selection in SVM Text Categorization, *Proc. of 16th National Conference on Artificial Intelligence (AAAI-99)*, pp. 480–486 (1999).

[Toyoura96] Toyoura, J., Tokunaga, T., Isahara, H. and Oka, R.: Development of a RWC Text Database Tagged with Classification code (in Japanese), *NLC96-13. IEICE*, pp. 89–96 (1996).

[Ueda00] Ueda, N. and Ghahramani, Z.: Optimal Model Inference for Bayesian Mixture of Experts, *Proc. of IEEE Neural Networks for Signal Processing (NNSP2000)*, pp. 145–154 (2000).

[Ueda01] Ueda, N.: Extended Tied Mixture Models, *Proc. of 2001 Workshop on Information-Based Induction Science (IBIS-2001)*, pp. 89–94 (2001).

[Vapnik95] Vapnik, V.: *The Nature of Statistical Learning Theory*, Springer-Verlag, New York (1995).

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

[Yang94] Yang, Y.: Expert Network: Effective and Efficient Learning from Human Decisions in Text Categorization and Retrieval, *Proc. of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 13–22 (1994).

[Yang97] Yang, Y. and Pederson, J.: A Comparative Study on Feature Selection in Text Categorization, *Machine Learning: Proc. of the 14th International Conference (ICML'97)*, pp. 412–420 (1997).

List of Publications

Journal Papers

[1] Taira, H. and Haruno, M.: "Feature Selection in SVM Text Categorization (in Japanese)," *Journal of IPSJ*, Vol. 41, No. 4, pp. 1113-1123, 2000.

[2] Taira, H. and Haruno, M.: "Text Categorization Using Transductive Boosting (in Japanese)," *Journal of IPSJ*, 2002. (to appear)

Conference Papers

[1] Taira, H. and Haruno, M.: "Feature Selection in SVM Text Categorization," *Proc. of the 16th National Conference on Artificial Intelligence (AAAI-99)*, pp. 480-486, 1999.

[2] Taira, H. and Haruno, M.: "Text Categorization Using Transductive Boosting," *Proc. of the 12th European Conference on Machine Learning (ECML-2001)*, pp. 454-465, 2001.

Other Publications

[1] Taira, H., Mukouchi, T. and Haruno, M.: "Text Categorization Using Support Vector Machines (in Japanese)," *IPSJ SIG-NLP*, NL-128, pp.173-180, 1998.

[2] Taira, H. and Haruno, M.: "SVM Text Categorization (in Japanese)," *Proc. of 1999 Workshop on Information-Based Induction Sciences (IBIS-99)*, pp. 233-238, 1999.

[3] Taira, H. and Haruno, M.: "Text Categorization Using a Transductive Boosting Method (in Japanese)," *IPSJ SIG-NLP*, NL-139, pp. 69-76, 2000.

[4] Nagata, M. and Taira, H.: "Text Categorization - the Showcase of Learning Theory - (in Japanese)," *IPSJ Magazine*, Vol. 42, No. 1, pp. 32-37, 2001.

[5] Taira, H. and Haruno, M.: "Text Categorization Using Transductive Boosting (in Japanese)," *Proc. of 2001 Workshop on Information-Based Induction Sciences (IBIS-2001)*, pp. 43-48, 2001.

[6] Taira, H., Matsushita, M. and Iida, T.: "Acquisition of Knowledge for Specifying Users' Requirements (in Japanese)," *IPSJ SIG-NLP*, NL-123, pp. 41-47, 1998.

[7] Makino, T., Matsushita, M., Taira, H. and Iida, T.: "Requests Understanding Method on the Knowledge Provider (in Japanese)," *12th Annual Conference of JSAI*, 98-24-04, pp. 384-385, 1998.

[8] Sasaki, Y., Isozaki, H., Taira, H., Hirota, K., Kazawa, H., Hirao, T., Nakajima, H. and Kato, T.: "An Evaluation and Comparison of Japanese Question Answering Systems (in Japanese)," *Technical Report of IEICE*, NLC2000-10, pp. 17-24, 2000.

[9] Sasaki, Y., Isozaki, H., Taira, H., Hirao, T., Kazawa, H., Suzuki, J., Kokuryo, K. and Maeda, E.: "SAIQA: A Japanese QA System Based on a Large-Scale Corpus (in Japanese)," *IPSJ SIG-NLP*, FI-64/NL-145, pp. 77-82, 2001.

Abbreviations

IEICE Association for Natural Language Processing

IPSJ Information Processing Society of Japan

SIG-NLP Special Interest Group on Natural Language Processing

JSAI Japanese Society for Artificial Intelligence