Virtual Examples for Text Classification with Support Vector Machines

Manabu Sassano
Fujitsu Laboratories Ltd.
4-1-1, Kamikodanaka, Nakahara-ku,
Kawasaki 211-8588, Japan
sassano@jp.fujitsu.com

Abstract

We explore how virtual examples (artificially created examples) improve performance of text classification with Support Vector Machines (SVMs). We propose techniques to create virtual examples for text classification based on the assumption that the category of a document is unchanged even if a small number of words are added or deleted. We evaluate the proposed methods by Reuters-21758 test set collection. Experimental results show virtual examples improve the performance of text classification with SVMs, especially for small training sets.

1 Introduction

Corpus-based supervised learning is now a standard approach to achieve high-performance in natural language processing. However, the weakness of supervised learning approach is to need an annotated corpus, the size of which is reasonably large. Even if we have a good supervised-learning method, we cannot get high-performance without an annotated corpus. The problem is that corpus annotation is labor intensive and very expensive. In order to overcome this, several methods are proposed, including minimally-supervised learning methods (e.g., (Yarowsky, 1995; Blum and Mitchell, 1998)), and active learning methods (e.g., (Thompson et al., 1999; Sassano, 2002)). The spirit behind these methods is to utilize precious labeled examples maximally.

Another method following the same spirit is one using *virtual examples* (artificially created examples) generated from labeled examples. This method has been rarely discussed in natural language processing. In terms of active learning, Lewis and Gale (1994) mentioned the use of virtual examples in text classification. They did not, however, take forward this approach because it did not seem to be possible that a classifier created virtual examples of documents in natural language and then requested a human teacher to label them.

In the field of pattern recognition, some kind of virtual examples has been studied. The first report of methods using virtual examples with Support Vector Machines (SVMs) is that of Schölkopf et al. (1996), who demonstrated significant improvement of the accuracy in hand-written digit recognition (Section 3). They created virtual examples from labeled examples based on prior knowledge of the task: slightly translated (e.g., 1 pixel shifted to the right) images have the same label (class) of the original image. Niyogi et al. (1998) also discussed the use of prior knowledge by creating virtual examples and thereby expanding the effective training set size.

The purpose of this study is to explore the effectiveness of virtual examples in NLP, motivated by the results of Schölkopf et al. (1996). To our knowledge, use of virtual examples in corpus-based NLP has never been studied so far. It is, however, important to investigate this approach by which it is expected that we can alleviate the cost of corpus annotation. In particular, we focus on virtual examples with Support Vector Machines, introduced by Vapnik (1995). The reason for this is that SVM is one of most successful machine learning methods in NLP. For example, NL tasks to which SVMs have been applied are text classification (Joachims, 1998; Dumais et al., 1998), chunking (Kudo and Matsumoto, 2001), dependency analysis (Kudo and Matsumoto, 2002) and so forth.

In this study, we choose text classification as a first case of the study of virtual examples in NLP because text classification in real world requires minimizing annotation cost, and it is not too complicated to perform some non-trivial experiments. Moreover, there are simple methods, which we propose, to generate virtual examples from labeled examples (Section 4). We show how virtual examples can improve the performance of a classifier with SVM in text classification, especially for small training sets.

2 Support Vector Machines

In this section we give some theoretical definitions of SVMs. Assume that we are given the training data

$$(\boldsymbol{x}_i, y_i), \ldots, (\boldsymbol{x}_l, y_l), \boldsymbol{x}_i \in \boldsymbol{R}^n, y_i \in \{+1, -1\}.$$

The decision function g in SVM framework is defined as:

$$g(\boldsymbol{x}) = \mathrm{sgn}(f(\boldsymbol{x})) \quad (1)$$

$$f(\boldsymbol{x}) = \sum_{i=1}^{l} y_i \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b \quad (2)$$

where K is a kernel function, $b \in \boldsymbol{R}$ is a threshold, and α_i are weights. Besides, the weights α_i satisfy the following constraints:

$$\forall i : 0 \leq \alpha_i \leq C \text{ and } \sum_{i=1}^{l} \alpha_i y_i = 0,$$

where C is a misclassification cost. The vectors $\boldsymbol{x}_i$ with non-zero α_i are called Support Vectors. For linear SVMs, the kernel function K is defined as:

$$K(\boldsymbol{x}_i, \boldsymbol{x}) = \boldsymbol{x}_i \cdot \boldsymbol{x}.$$

In this case, Equation 2 can be written as:

$$f(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x} + b \quad (3)$$

where $\boldsymbol{w} = \sum_{i=1}^{l} y_i \alpha_i \boldsymbol{x}_i$. To train an SVM is to find α_i and b by solving the following optimization problem:

$$\text{maximize} \quad \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

$$\text{subject to} \quad \forall i : 0 \leq \alpha_i \leq C \text{ and } \sum_{i=1}^{l} \alpha_i y_i = 0.$$

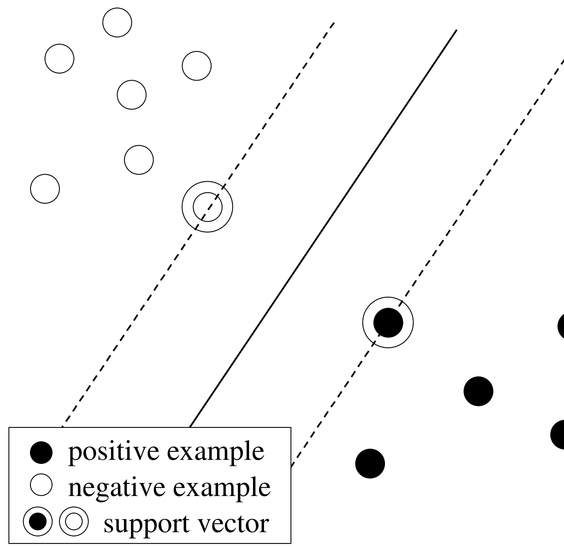

Figure 1: Hyperplane (solid) and Support Vectors

The solution gives an optimal hyperplane, which is a decision boundary between the two classes. Figure 1 illustrates an optimal hyperplane and its support vectors.

3 Virtual Examples and Virtual Support Vectors

Virtual examples are generated from labeled examples.[1] Based on prior knowledge of a target task, the label of a generated example is set to the same value as that of the original example.

For example, in hand-written digit recognition, virtual examples can be created on the assumption that the label of an example is unchanged even if the example is shifted by one pixel in the four principal directions (Schölkopf et al., 1996; DeCoste and Schölkopf, 2002).

Virtual examples that are generated from support vectors are called *virtual support vectors* (Schölkopf

[1]We discuss here only virtual examples which are generated from labeled examples. We do not consider examples, the labels of which are not known.

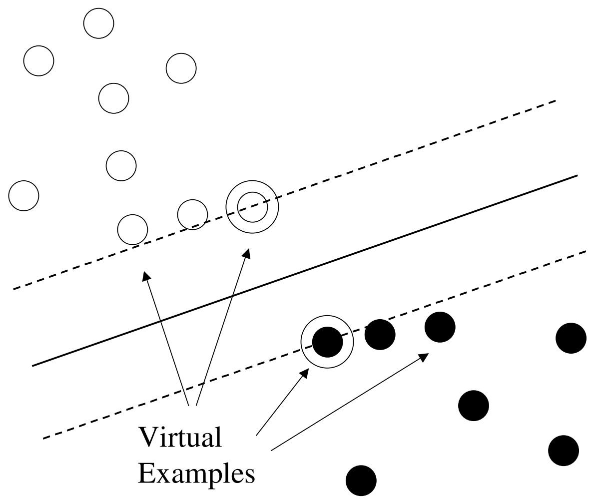

Figure 2: Hyperplane and Virtual Examples

et al., 1996). Reasonable virtual support vectors are expected to give a better optimal hyperplane. Assuming that virtual support vectors represent natural variations of examples of a target task, the decision boundary should be more accurate. Figure 2 illustrates the idea of virtual support vectors. Note that after virtual support vectors are given, the hyperplane is different from that in Figure 1.

4 Virtual Examples for Text Classification

We assume on text classification the following:

Assumption 1 *The category of a document is unchanged even if a small number of words are added or deleted.*

This assumption is reasonable. In typical cases of text classification most of the documents usually contain two or more keywords which may indicate the categories of the documents.

Following Assumption 1, we propose two methods to create virtual examples for text classification. One method is to delete some portion of a document. The label of a virtual example is given from the original document. The other method is to add a small number of words to a document. The words to be added are taken from documents, the label of which is the same as that of the document. Although one can invent various methods to create virtual examples based on Assumption 1, we propose here very simple ones.

Document Id	Feature Vector ($\boldsymbol{x}$)	Label (y)
1	$(f_1, f_2, f_3, f_4, f_5)$	$+1$
2	(f_2, f_4, f_5, f_6)	$+1$
3	$(f_2, f_3, f_5, f_6, f_7)$	$+1$
4	$(f_1, f_3, f_8, f_9, f_{10})$	-1
5	$(f_1, f_8, f_{10}, f_{11})$	-1

Table 1: Example of Document Set

Before describing our methods, we describe text representation which we used in this study. We tokenize a document to words, downcase them and then remove stopwords, where the stopword list of freeWAIS-sf[2] is used. Stemming is not performed. We adopt binary feature vectors where word frequency is not used.

Now we describe the two proposed methods: GenerateByDeletion and GenerateByAddition. Assume that we are given a feature vector (a document) $\boldsymbol{x}$ and $\boldsymbol{x}'$ is a generated vector from $\boldsymbol{x}$. GenerateByDeletion algorithm is:

1. Copy $\boldsymbol{x}$ to $\boldsymbol{x}'$.

2. For each binary feature f of $\boldsymbol{x}'$, if rand() $\leq t$ then remove the feature f, where rand() is a function which generates a random number from 0 to 1, and t is a parameter to decide how many features are deleted.

For example, suppose that we have a set of documents as in Table 1. Some possible virtual examples generated from Document 1 by GenerateByDeletion algorithm are $(f_2, f_3, f_4, f_5, +1)$, $(f_1, f_3, f_4, +1)$, or $(f_1, f_2, f_4, f_5, +1)$.

On the other hand, GenerateByAddition algorithm is:

1. Collect from a training set documents, the label of which is the same as that of $\boldsymbol{x}$.

2. Concatenate all the feature vectors (documents) to create an array a of features. Each element of a is a feature which represents a word.

3. Copy $\boldsymbol{x}$ to $\boldsymbol{x}'$.

[2] Available at http://ls6-www.informatik.uni-dortmund.de/ir/projects/freeWAIS-sf/

Category Name	Training	Test
earn	2877	1087
acq	1650	719
money-fx	538	179
grain	433	149
crude	389	189
trade	369	117
interest	347	131
ship	197	89
wheat	212	71
corn	181	56

Table 2: Number of Training and Test Examples

4. For each binary feature f of $\boldsymbol{x'}$, if rand() $\leq t$ then select a feature randomly from a and put it to $\boldsymbol{x'}$.

For example, when we want to generate a virtual example from Document 2 in Table 1 by GenerateByAddition algorithm, first we create an array $a = (f_1, f_2, f_3, f_4, f_5, f_2, f_4, f_5, f_6, f_2, f_3, f_5, f_6, f_7)$. In this case, some possible virtual examples by GenerateByAddition are $(f_1, f_2, f_4, f_5, f_6, +1)$, $(f_2, f_3, f_4, f_5, f_6, +1)$, or $(f_2, f_4, f_5, f_6, f_7, +1)$. An example such as $(f_2, f_4, f_5, f_6, f_{10}, +1)$ is never generated from Document 2 because there are no positive documents which have f_{10}.

5 Experimental Results and Discussion

5.1 Test Set Collection

We used the Reuters-21578 dataset[3] to evaluate the proposed methods. The dataset has several splits of a training set and a test set. We used here "ModApte" split, which is most widely used in the literature on text classification. This split has 9,603 training examples and 3,299 test examples. More than 100 categories are in the dataset. We use, however, only the most frequent 10 categories. Table 2 shows the 10 categories and the number of training and test examples in each of the categories.

5.2 Performance Measures

We use F-measure (van Rijsbergen, 1979; Lewis and Gale, 1994) as a primal performance measure to evaluate the result. F-measure is defined as:

$$\text{F-measure} = \frac{(1+\beta^2)pq}{\beta^2 p + q} \quad (4)$$

where p is precision and q is recall and β is a parameter which decides the relative weight of precision and recall. The p and the q are defined as:

$$p = \frac{\text{number of positive and correct outputs}}{\text{number of positive outputs}}$$

$$q = \frac{\text{number of positive and correct outputs}}{\text{number of positive examples}}$$

In Equation 4, usually $\beta = 1$ is used, which means it gives equal weight to precision and recall.

When we evaluate the performance of a classifier to a multiple category dataset, there are two ways to compute F-measure: macro-averaging and micro-averaging (Yang, 1999). The former way is to first compute F-measure for each category and then average them, while the latter way is to first compute precision and recall for all the categories and use them to calculate the F-measure.

5.3 SVM setting

Through our experiments we used our original SVM tools, the algorithm of which is based on SMO (Sequential Minimal Optimization) by Platt (1999). We used linear SVMs and set a misclassification cost C to 0.016541 which is $1/($the average of $\boldsymbol{x} \cdot \boldsymbol{x})$ where $\boldsymbol{x}$ is a feature vector in the 9,603 size training set. For simplicity, we fixed C through all the experiments. We built a binary classifier for each of the 10 categories shown in Table 2.

5.4 Results

First, we carried out experiments using GenerateByDeletion and GenerateByAddition separately to create virtual examples, where a virtual example was created per Support Vector. We did not generate virtual examples from non support vectors. We set the parameter t to 0.05[4] for GenerateByDeletion and GenerateByAddition for all the experiments.

To build an SVM with virtual examples we use the following steps:

[3] Available from David D. Lewis's page: http://www.daviddlewis.com/resources/testcollections/reuters21578/

[4] We first tried $t = 0.01, 0.05,$ and 0.10 with GenerateByDeletion using the 9603 size training set. The value $t = 0.05$ yielded best micro-average F-measure for the test set. We used the same value also for GenerateByAddition.

1. Train an SVM.

2. Extract Support Vectors.

3. Generate virtual examples from the Support Vectors.

4. Train another SVM using both the original labeled examples and the virtual examples.

We evaluated the performance of the two methods depending on the size of a training set. We created subsamples by selecting randomly from the 9603 size training set. We prepared seven sizes: 9603, 4802, 2401, 1200, 600, 300, and 150.[5] Micro-average F-measures of the two methods are shown in Table 3. We see from Table 3 that both the methods give better performance than that of the original SVM. The smaller the number of examples in the training set is, the larger the gain is. For the 9603 size training set, the gain of GenerateByDeletion is 0.75 ($= 90.17 - 89.42$), while for the 150 size set, the gain is 6.88 ($= 60.16 - 53.28$). These results suggest that in the smaller training sets there are not enough various examples to give a accurate decision boundary and therefore the effect of virtual examples is larger at the smaller training sets. It is reasonable to conclude that GenerateByDeletion and GenerateByAddition generated good virtual examples for the task and this led to the performance gain.

After we found that the simple two methods to generate virtual support vectors were effective, we examined a combined method which is to use both GenerateByDeletion and GenerateByAddition. Two virtual examples are generated per Support Vector. The performance of the combined method is also shown in Table 3. The performance gain of the combined method is larger than that with either GenerateByDeletion or GenerateByAddition.

Furthermore, we carried out another experiment with a combined method to create two virtual examples with GenerateByDeletion and GenerateByAddition respectively. That is, four virtual examples were generated from a Support Vector. The performance of that setting is shown in Table 3. The best result is achieved by the combined method to create four virtual examples per Support Vector.

[5] Since we selected samples randomly, some smaller training sets of low frequent categories may have had few or even zero positive examples.

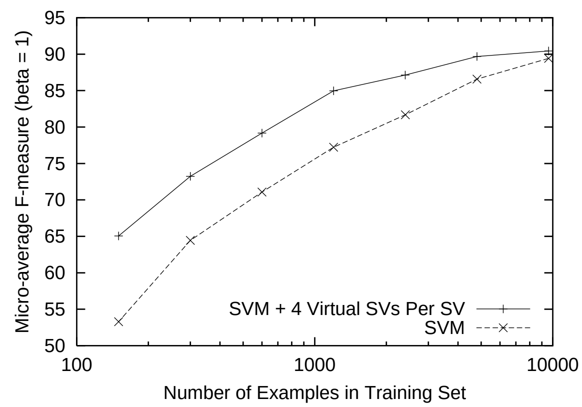

Figure 3: Micro-Average F-Measure versus Number of Examples in the Training Set

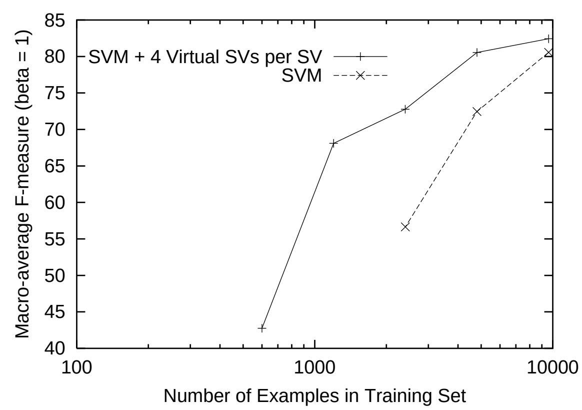

Figure 4: Macro-Average F-Measure versus Number of Examples in the Training Set. For the smaller training sets F-measures cannot be computed because the precisions are undefined.

For the rest of this section, we limit our discussion to the comparison of the results of the original SVM and SVM with four virtual examples per SV (SVM with 4 VSVs). The learning curves of the original SVM and SVM with 4 VSVs are shown in Figures 3 and 4. It is clear that SVM with 4 VSVs outperforms the original SVM considerably in terms of both micro-average F-measure and macro-average F-measure. SVM with 4 VSVs achieves a given level of performance with roughly half of the labeled examples which the original SVM requires. One might suppose that the improvement of F-measure

Method	Number of Examples in Training Set 9603	4802	2401	1200	600	300	150
Original SVM	89.42	86.58	81.69	77.24	71.08	64.44	53.28
SVM + 1 VSV per SV (GenerateByDeletion)	90.17	88.62	84.45	81.11	75.32	70.11	60.16
SVM + 1 VSV per SV (GenerateByAddition)	90.00	88.51	84.48	81.14	75.33	69.59	60.04
SVM + 2 VSVs per SV (Combined)	90.27	89.33	86.27	83.59	77.44	72.81	64.22
SVM + 4 VSVs per SV (Combined)	90.45	89.69	87.12	84.97	79.16	73.25	65.05

Table 3: Comparison of Micro-Average F-measure of Different Methods. "VSV" means virtual SV.

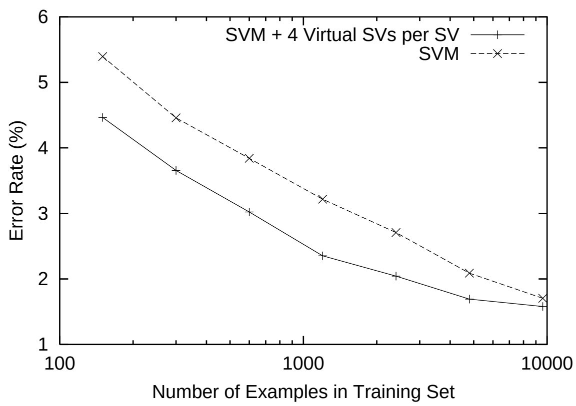

Figure 5: Error Rate versus Number of Examples in the Training Set

is realized simply because the recall gets highly improved while the error rate increases. We plot changes of the error rate for 32990 tests (3299 tests for each of the 10 categories) in Figure 5. SVM with 4 VSVs still outperforms the original SVM significantly.[6]

The performance changes for each of the 10 categories are shown in Tables 4 and 5. SVM with 4 VSVs is better than the original SVM for almost all the categories and all the sizes except for "interest" and "wheat" at the 9603 size training set. For low frequent categories such as "ship", "wheat" and "corn", the classifiers of the original SVM perform poorly. There are many cases where they never output 'positive', i.e. the recall is zero. It suggests that the original SVM fails to find a good hyperplane due to the imbalanced training sets which have very few positive examples. In contrast, SVM with 4 VSVs yields better results for such harder cases.

[6]We have done the significance test which is called "p-test" in (Yang and Liu, 1999), requiring significance at the 0.05 level. Although at the 9603 size training set the improvement of the error rate is not statistically significant, in all the other cases the improvement is significant.

6 Conclusion and Future Directions

We have explored how virtual examples improve the performance of text classification with SVMs. For text classification, we have proposed methods to create virtual examples on the assumption that the label of a document is unchanged even if a small number of words are added or deleted. The experimental results have shown that our proposed methods improve the performance of text classification with SVMs, especially for small training sets. Although the proposed methods are not readily applicable to NLP tasks other than text classification, it is notable that the use of virtual examples, which has been very little studied in NLP, is empirically evaluated.

In the future, it would be interesting to employ virtual examples with methods to use both labeled and unlabeled examples (e.g., (Blum and Mitchell, 1998; Nigam et al., 1998; Joachims, 1999)). The combined approach may yield better results with a small number of labeled examples. Another interesting direction would be to develop methods to create virtual examples for the other tasks (e.g., named entity recognition, POS tagging, and parsing) in NLP. We believe we can use prior knowledge on these tasks to create effective virtual examples.

References

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th COLT*, pages 92–100.

Dennis DeCoste and Bernhard Schölkopf. 2002. Training invariant support vector machines. *Machine Learning*, 46:161–190.

Category	Number of Examples in the Training Set						
	9603	4802	2401	1200	600	300	150
earn	98.06	97.49	97.40	96.39	95.94	94.85	93.73
acq	91.94	89.87	84.43	84.01	78.17	63.10	12.03
money-fx	64.90	61.69	56.03	51.69	17.91	01.11	05.38
grain	86.96	81.68	75.20	59.63	41.27	06.49	-
crude	84.59	81.52	67.11	33.33	01.05	-	-
trade	74.89	64.58	54.86	40.26	12.80	01.69	-
interest	**63.89**	60.29	50.27	35.15	08.57	05.88	-
ship	66.19	44.07	32.73	02.22	-	-	-
wheat	**89.61**	80.60	38.30	08.11	-	-	-
corn	84.62	62.79	10.17	-	-	-	-
Macro-average	80.56	72.46	56.65	-	-	-	-
Micro-average	89.42	86.58	81.69	77.24	71.08	64.44	53.28

Table 4: F-Measures for the Reuters Categories with the Original SVM. The hyphen '-' denotes the case where F-measure cannot be computed because the classifier always says 'negative' and therefore its precision is undefined. The scores in bold means that the score of the original SVM is better than that of SVM with 4 Virtual SVs per SV (shown in Table 5).

Category	Number of Examples in the Training Set						
	9603	4802	2401	1200	600	300	150
earn	**98.07**	**98.02**	**97.56**	**97.37**	**97.14**	**96.00**	**95.46**
acq	**94.20**	**93.06**	**91.71**	**88.81**	**88.92**	**78.70**	**59.92**
money-fx	**70.83**	**73.10**	**62.86**	**65.68**	**47.91**	**32.43**	**33.76**
grain	**89.20**	**84.72**	**85.11**	**80.44**	**60.79**	**44.10**	**01.00**
crude	**84.93**	**86.33**	**76.92**	**74.36**	**15.53**	**02.00**	-
trade	**75.83**	**73.21**	**62.31**	**43.53**	**37.58**	**18.32**	**01.65**
interest	62.73	**63.16**	**65.77**	**63.35**	**59.11**	**37.50**	**11.92**
ship	**73.68**	**67.14**	**50.79**	**30.48**	**06.45**	**02.22**	-
wheat	87.42	**82.61**	**87.94**	**68.91**	**10.67**	-	-
corn	**87.50**	**84.11**	**46.75**	**68.09**	**03.45**	-	-
Macro-average	**82.44**	**80.55**	**72.77**	**68.10**	**42.76**	-	-
Micro-average	**90.45**	**89.69**	**87.12**	**84.97**	**79.16**	**73.25**	**65.05**

Table 5: F-Measures for the Reuters Categories with SVM with 4 Virtual SVs per SV. The scores in bold means that the score of SVM with 4 Virtual SVs per SV is better than that of the original SVM (shown in Table 4).

Susan Dumais, John Platt, David Heckerman, and Mehran Sahami. 1998. Inductive learning algorithms and representations for text categorization. In *Proceedings of the ACM CIKM International Conference on Information and Knowledge Management*, pages 148–155.

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

Thorsten Joachims. 1999. Transductive inference for text classification using support vector machines. In *Proceedings of the 16th International Conference on Machine Learning*, pages 200–209.

Taku Kudo and Yuji Matsumoto. 2001. Chunking with support vector machines. In *Proceedings of NAACL 2001*, pages 192–199.

Taku Kudo and Yuji Matsumoto. 2002. Japanese dependency analysis using cascaded chunking. In *Proceedings of CoNLL-2002*, pages 63–69.

David D. Lewis and William A. Gale. 1994. A sequential algorithm for training text classifiers. In *Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 3–12.

Kamal Nigam, Andrew McCallum, Sebastian Thrun, and Tom Mitchell. 1998. Learning to classify text from labeled and unlabeled documents. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, pages 792–799.

Partha Niyogi, Federico Girosi, and Tomaso Poggio. 1998. Incorporating prior information in machine learning by creating virtual examples. In *Proceedings of IEEE*, volume 86, pages 2196–2207.

John C. Platt. 1999. Fast training of support vector machines using sequential minimal optimization. In Bernhard Schölkopf, Christopher J.C. Burges, and Alexander J. Smola, editors, *Advances in Kernel Methods: Support Vector Learning*, pages 185–208. MIT Press.

Manabu Sassano. 2002. An empirical study of active learning with support vector machines for Japanese word segmentation. In *Proceedings of ACL-2002*, pages 505–512.

Bernhard Schölkopf, Chris Burges, and Vladimir Vapnik. 1996. Incorporating invariances in support vector learning machines. In C. von der Malsburg, W. von Seelen, J.C. Vorbrüggen, and B. Sendhoff, editors, *Artificial Neural Networks – ICANN'96, Springer Lecture Notes in Computer Science, Vol. 1112*, pages 47–52.

Cynthia A. Thompson, Mary Leaine Califf, and Raymond J. Mooney. 1999. Active learning for natural language parsing and information extraction. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 406–414.

C.J. van Rijsbergen. 1979. *Information Retrieval*. Butterworths, 2nd edition.

Vladimir N. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer-Verlag.

Yiming Yang and Xin Liu. 1999. A re-examination of text categorization methods. In *Proceedings of SIGIR-99, 2nd ACM International Conference on Research and Development in Information Retrieval*, pages 42–49.

Yiming Yang. 1999. An evaluation of statistical approaches to text categorization. *Journal of Information Retrieval*, 1(1/2):67–88.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of ACL-1995*, pages 189–196.