Proceedings of the Workshop on Natural Language Processing in
the Biomedical Domain, Philadelphia, July 2002, pp. 1-8.
Association for Computational Linguistics.

Tuning Support Vector Machines for Biomedical Named Entity Recognition

Jun'ichi Kazama† **Takaki Makino**‡ **Yoshihiro Ohta**∗ **Jun'ichi Tsujii**† ¶

† Department of Computer Science, Graduate School of Information Science and Technology,
University of Tokyo, Bunkyo-ku, Tokyo 113-0033, Japan

‡ Department of Complexity Science and Engineering, Graduate School of Frontier Sciences,
University of Tokyo, Bunkyo-ku, Tokyo 113-0033, Japan

∗ Central Research Laboratory, Hitachi, Ltd., Kokubunji, Tokyo 185-8601, Japan

¶ CREST, JST (Japan Science and Technology Corporation)

Abstract

We explore the use of Support Vector Machines (SVMs) for biomedical named entity recognition. To make the SVM training with the available largest corpus – the GENIA corpus – tractable, we propose to split the non-entity class into sub-classes, using part-of-speech information. In addition, we explore new features such as word cache and the states of an HMM trained by unsupervised learning. Experiments on the GENIA corpus show that our class splitting technique not only enables the training with the GENIA corpus but also improves the accuracy. The proposed new features also contribute to improve the accuracy. We compare our SVM-based recognition system with a system using Maximum Entropy tagging method.

1 Introduction

Application of natural language processing (NLP) is now a key research topic in bioinformatics. Since it is practically impossible for a researcher to grasp all of the huge amount of knowledge provided in the form of natural language, e.g., journal papers, there is a strong demand for biomedical information extraction (IE), which extracts knowledge automatically from biomedical papers using NLP techniques (Ohta et al., 1997; Proux et al., 2000; Yakushiji et al., 2001).

The process called *named entity recognition*, which finds entities that fill the information slots, e.g., proteins, DNAs, RNAs, cells etc., in the biomedical context, is an important building block in such biomedical IE systems. Conceptually, named entity recognition consists of two tasks: *identification*, which finds the region of a named entity in a text, and *classification*, which determines the semantic class of that named entity. The following illustrates biomedical named entity recognition.

> "Thus, CIITA$_{PROTEIN}$ not only activates the expression of class II genes$_{DNA}$ but recruits another B cell-specific coactivator to increase transcriptional activity of class II promoters$_{DNA}$ in B cells$_{CELLTYPE}$."

Machine learning approach has been applied to biomedical named entity recognition (Nobata et al., 1999; Collier et al., 2000; Yamada et al., 2000; Shimpuku, 2002). However, no work has achieved sufficient recognition accuracy. One reason is the lack of annotated corpora for training as is often the case of a new domain. Nobata et al. (1999) and Collier et al. (2000) trained their model with only 100 annotated paper abstracts from the MEDLINE database (National Library of Medicine, 1999), and Yamada et al. (2000) used only 77 annotated paper abstracts. In addition, it is difficult to compare the techniques used in each study because they used a closed and different corpus.

To overcome such a situation, the GENIA corpus (Ohta et al., 2002) has been developed, and at this time it is the largest biomedical annotated corpus available to public, containing 670 annotated abstracts of the MEDLINE database.

Another reason for low accuracies is that biomedical named entities are essentially hard to recognize using standard feature sets compared with the named entities in newswire articles (Nobata et al., 2000). Thus, we need to employ powerful machine learning techniques which can incorporate various and complex features in a consistent way.

Support Vector Machines (SVMs) (Vapnik, 1995) and Maximum Entropy (ME) method (Berger et al., 1996) are powerful learning methods that satisfy such requirements, and are applied successfully to other NLP tasks (Kudo and Matsumoto, 2000; Nakagawa et al., 2001; Ratnaparkhi, 1996). In this paper, we apply Support Vector Machines to biomedical named entity recognition and train them with

the GENIA corpus. We formulate the named entity recognition as the classification of each word with context to one of the classes that represent region and named entity's semantic class. Although there is a previous work that applied SVMs to biomedical named entity task in this formulation (Yamada et al., 2000), their method to construct a classifier using SVMs, *one-vs-rest*, fails to train a classifier with entire GENIA corpus, since the cost of SVM training is super-linear to the size of training samples. Even with a more feasible method, *pairwise* (Kreßel, 1998), which is employed in (Kudo and Matsumoto, 2000), we cannot train a classifier in a reasonable time, because we have a large number of samples that belong to the *non-entity* class in this formulation. To solve this problem, we propose to split the non-entity class to several sub-classes, using part-of-speech information. We show that this technique not only enables the training feasible but also improves the accuracy.

In addition, we explore new features such as *word cache* and the states of an unsupervised HMM for named entity recognition using SVMs. In the experiments, we show the effect of using these features and compare the overall performance of our SVM-based recognition system with a system using the Maximum Entropy method, which is an alternative to the SVM method.

2 The GENIA Corpus

The GENIA corpus is an annotated corpus of paper abstracts taken from the MEDLINE database. Currently, 670 abstracts are annotated with named entity tags by biomedical experts and made available to public (Ver. 1.1).[1] These 670 abstracts are a subset of more than 5,000 abstracts obtained by the query "*human* AND *blood cell* AND *transcription factor*" to the MEDLINE database. Table 1 shows basic statistics of the GENIA corpus. Since the GENIA corpus is intended to be extensive, there exist 24 distinct named entity classes in the corpus.[2] Our task is to find a named entity region in a paper abstract and correctly select its class out of these 24 classes. This number of classes is relatively large compared with other corpora used in previous studies, and compared with the named entity task for newswire articles. This indicates that the task with the GENIA corpus is hard, apart from the difficulty of the biomedical domain itself.

[1] Available via http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA/

[2] The GENIA corpus also has annotations for conjunctive/disjunctive named entity expressions such as "human B- or T-cell lines" (Kim et al., 2001). In this paper we ignore such expressions and consider that constituents in such expressions are annotated as a dummy class "temp".

Table 1: Basic statistics of the GENIA corpus

# of sentences	5,109
# of words	152,216
# of named entities	23,793
# of words in NEs	50,229
# of words not in NEs	101,987
Av. length of NEs (σ)	2.11 (1.40)

3 Named Entity Recognition Using SVMs

3.1 Named Entity Recognition as Classification

We formulate the named entity task as the classification of each word with context to one of the classes that represent region information and named entity's semantic class. Several representations to encode region information are proposed and examined (Ramshaw and Marcus, 1995; Uchimoto et al., 2000; Kudo and Matsumoto, 2001). In this paper, we employ the simplest *BIO* representation, which is also used in (Yamada et al., 2000). We modify this representation in Section 5.1 in order to accelerate the SVM training.

In the BIO representation, the region information is represented as the class prefixes "B-" and "I-", and a class "O". B- means that the current word is at the beginning of a named entity, I- means that the current word is in a named entity (but not at the beginning), and O means the word is not in a named entity. For each named entity class *C*, class B-*C* and I-*C* are produced. Therefore, if we have *N* named entity classes, the BIO representation yields $2N + 1$ classes, which will be the targets of a classifier. For instance, the following corresponds to the annotation "Number of glucocorticoid receptors$_{PROTEIN}$ in lymphocytes$_{CELLTYPE}$ and ...".

Number	of	glucocorticoid	receptors
O	O	B-PROTEIN	I-PROTEIN
in	lymphocytes	and	...
O	B-CELLTYPE	O	...

3.2 Support Vector Machines

Support Vector Machines (SVMs) (Cortes and Vapnik, 1995) are powerful methods for learning a classifier, which have been applied successfully to many NLP tasks such as base phrase chunking (Kudo and Matsumoto, 2000) and part-of-speech tagging (Nakagawa et al., 2001).

The SVM constructs a binary classifier that outputs +1 or −1 given a sample vector $\mathbf{x} \in \mathbb{R}^n$. The decision is based on the separating hyperplane as follows.

$$c(\mathbf{x}) = \begin{cases} +1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0, \quad \mathbf{w} \in \mathbb{R}^n, b \in \mathbb{R}, \\ -1 & \text{otherwise} \end{cases}$$

The class for an input $\mathbf{x}$, $c(\mathbf{x})$, is determined by seeing which side of the space separated by the hyperplane, $\mathbf{w} \cdot \mathbf{x} + b = 0$, the input lies on.

Given a set of labeled training samples $\{(y_1, \mathbf{x}_1), \cdots, (y_L, \mathbf{x}_L)\}$, $\mathbf{x}_i \in \mathbb{R}^n$, $y_i \in \{+1, -1\}$, the SVM training tries to find the *optimal hyperplane*, i.e., the hyperplane with the maximum margin. Margin is defined as the distance between the hyperplane and the training samples nearest to the hyperplane. Maximizing the margin insists that these nearest samples (*support vectors*) exist on both sides of the separating hyperplane and the hyperplane lies exactly at the midpoint of these support vectors. This margin maximization tightly relates to the fine generalization power of SVMs.

Assuming that $|\mathbf{w} \cdot \mathbf{x}_i + b| = 1$ at the support vectors without loss of generality, the SVM training can be formulated as the following optimization problem.[3]

$$\begin{aligned} &\text{minimize} \quad \frac{1}{2}\|\mathbf{w}\|^2 \\ &\text{subject to} \quad y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad i = 1, \cdots, L. \end{aligned}$$

The solution of this problem is known to be written as follows, using only support vectors and weights for them.

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = \sum_{i \in SVs} y_i \alpha_i \mathbf{x} \cdot \mathbf{x}_i + b \qquad (1)$$

In the SVM learning, we can use a function $k(\mathbf{x}_i, \mathbf{x}_j)$ called a *kernel function* instead of the inner product in the above equation. Introducing a kernel function means mapping an original input $\mathbf{x}$ using $\Phi(\mathbf{x})$, s.t. $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) = k(\mathbf{x}_i, \mathbf{x}_j)$ to another, usually a higher dimensional, feature space. We construct the optimal hyperplane in that space. By using kernel functions, we can construct a non-linear separating surface in the original feature space. Fortunately, such non-linear training does not increase the computational cost if the calculation of the kernel function is as cheap as the inner product. A polynomial function defined as $(s\mathbf{x}_i \cdot \mathbf{x}_j + r)^d$ is popular in applications of SVMs to NLPs (Kudo and Matsumoto, 2000; Yamada et al., 2000; Kudo and Matsumoto, 2001), because it has an intuitively sound interpretation that each dimension of the mapped space is a (weighted) conjunction of d features in the original sample.

[3]For many real-world problems where the samples may be inseparable, we allow the constraints are broken with some penalty. In the experiments, we use so-called 1-norm soft margin formulation described as:

$$\begin{aligned} &\text{minimize} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{L}\xi_i \\ &\text{subject to} \quad y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i, \quad i = 1, \cdots, L, \\ &\qquad\qquad \xi_i \geq 0, \quad i = 1, \cdots, L. \end{aligned}$$

3.3 Multi-Class SVMs

As described above, the standard SVM learning constructs a binary classifier. To make a named entity recognition system based on the BIO representation, we require a multi-class classifier. Among several methods for constructing a multi-class SVM (Hsu and Lin, 2002), we use a pairwise method proposed by Kreßel (1998) instead of the one-vs-rest method used in (Yamada et al., 2000), and extend the BIO representation to enable the training with the entire GENIA corpus. Here we describe the one-vs-rest method and the pairwise method to show the necessity of our extension.

Both one-vs-rest and pairwise methods construct a multi-class classifier by combining many binary SVMs. In the following explanation, K denotes the number of the target classes.

one-vs-rest Construct K binary SVMs, each of which determines whether the sample should be classified as class i or as the other classes. The output is the class with the maximum $f(\mathbf{x})$ in Equation 1.

pairwise Construct $K(K-1)/2$ binary SVMs, each of which determines whether the sample should be classified as class i or as class j. Each binary SVM has one vote, and the output is the class with the maximum votes.

Because the SVM training is a quadratic optimization program, its cost is super-linear to the size of the training samples even with the tailored techniques such as SMO (Platt, 1998) and kernel evaluation caching (Joachims, 1998). Let L be the number of the training samples, then the one-vs-rest method takes time in $K \times O_{SVM}(L)$. The BIO formulation produces one training sample per word, and the training with the GENIA corpus involves over 100,000 training samples as can be seen from Table 1. Therefore, it is apparent that the one-vs-rest method is impractical with the GENIA corpus. On the other hand, if target classes are equally distributed, the pairwise method will take time in $K(K-1)/2 \times O_{SVM}(2L/K)$. This method is worthwhile because each training is much faster, though it requires the training of $(K-1)/2$ times more classifiers. It is also reported that the pairwise method achieves higher accuracy than other methods in some benchmarks (Kreßel, 1998; Hsu and Lin, 2002).

3.4 Input Features

An input $\mathbf{x}$ to an SVM classifier is a feature representation of the word to be classified and its context. We use a bit-vector representation, each dimension

of which indicates whether the input matches with a certain *feature*. The following illustrates the well-used features for the named entity recognition task.

$$w_{k,i} = \begin{cases} 1 & \text{if a word at } k, W_k, \text{ is the } i\text{th word} \\ & \quad \text{in the vocabulary } \mathcal{V} \\ 0 & \text{otherwise} \quad (\textit{word feature}) \end{cases}$$

$$pos_{k,i} = \begin{cases} 1 & \text{if } W_k \text{ is assigned the } i\text{th POS tag} \\ & \quad \text{in the POS tag list } \mathcal{POS} \\ 0 & \text{otherwise} \quad (\textit{part-of-speech feature}) \end{cases}$$

$$pre_{k,i} = \begin{cases} 1 & \text{if } W_k \text{ starts with the } i\text{th prefix} \\ & \quad \text{in the prefix list } \mathcal{P} \\ 0 & \text{otherwise} \quad (\textit{prefix feature}) \end{cases}$$

$$suf_{k,i} = \begin{cases} 1 & \text{if } W_k \text{ starts with the } i\text{th suffix} \\ & \quad \text{in the suffix list } \mathcal{S} \\ 0 & \text{otherwise} \quad (\textit{suffix feature}) \end{cases}$$

$$sub_{k,i} = \begin{cases} 1 & \text{if } W_k \text{ contains the } i\text{th substring} \\ & \quad \text{in the substring list } \mathcal{SB} \\ 0 & \text{otherwise} \quad (\textit{substring feature}) \end{cases}$$

$$pc_{k,i} = \begin{cases} 1 & \text{if } W_k (k < 0) \text{ was assigned } i\text{th class} \\ 0 & \text{otherwise } (\textit{preceding class feature}) \end{cases}$$

In the above definitions, k is a relative word position from the word to be classified. A negative value represents a preceding word's position, and a positive value represents a following word's position. Note that we assume that the classification proceeds left to right as can be seen in the definition of the preceding class feature. For the SVM classification, we does not use a dynamic argmax-type classification such as the Viterbi algorithm, since it is difficult to define a good comparable value for the confidence of a prediction such as probability. The consequences of this limitation will be discussed with the experimental results.

Features usually form a group with some variables such as the position unspecified. In this paper, we instantiate all features, i.e., instantiate for all i, for a group and a position. Then, it is convenient to denote a set of features for a group g and a position k as g_k (e.g., w_k and pos_k). Using this notation, we write a feature set as $\{w_{-1}, w_0, pre_{-1}, pre_0, pc_{-1}\}$.[4] This feature description derives the following input vector.[5]

$$\mathbf{x} = \{w_{-1,1}, w_{-1,2}, \cdots, w_{-1,|\mathcal{V}|}, w_{0,1}, \cdots, w_{0,|\mathcal{V}|}, pre_{-1,1}, \cdots, pre_{0,|\mathcal{P}|}, pc_{-1,1}, \cdots, pc_{-1,K}\}$$

[4] We will further compress this as $\{\langle w, pre\rangle_{[-1,0]}, pc_{-1}\}$.

[5] Although a huge number of features are instantiated, only a few features have value one for a given g and k pair.

4 Named Entity Recognition Using ME Model

The Maximum Entropy method, with which we compare our SVM-based method, defines the probability that the class is c given an input vector $\mathbf{x}$ as follows.

$$P(c|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_i \alpha_i^{f_i(c,\mathbf{x})},$$

where $Z(\mathbf{x})$ is a normalization constant, and $f_i(c, \mathbf{x})$ is a *feature function*. A feature function is defined in the same way as the features in the SVM learning, except that it includes c in it like $f(c, \mathbf{x})$ = (c is the jth class) $\wedge$ $w_{i,k}(\mathbf{x})$. If $\mathbf{x}$ contains previously assigned classes, then the most probable class sequence, $\hat{c}_1^T = \text{argmax}_{c_1,\cdots,c_T} \prod_{t=1}^T P(c_t|\mathbf{x}_t)$ is searched by using the Viterbi-type algorithm. We use the maximum entropy tagging method described in (Kazama et al., 2001) for the experiments, which is a variant of (Ratnaparkhi, 1996) modified to use HMM state features.

5 Tuning of SVMs for Biomedical NE Task

5.1 Class Splitting Technique

In Section 3.3, we described that if target classes are equally distributed, the pairwise method will reduce the training cost. In our case, however, we have a very unbalanced class distribution with a large number of samples belonging to the class "O" (see Table 1). This leads to the same situation with the one-vs-rest method, i.e., if L_O is the number of the samples belonging to the class "O", then the most dominant part of the training takes time in $K \times O_{SVM}(L_O)$.

One solution to this unbalanced class distribution problem is to split the class "O" into several subclasses effectively. This will reduce the training cost for the same reason that the pairwise method works.

In this paper, we propose to split the non-entity class according to part-of-speech (POS) information of the word. That is, given a part-of-speech tag set $\mathcal{POS}$, we produce new $|\mathcal{POS}|$ classes, "O-p" $p \in \mathcal{POS}$. Since we use a POS tagger that outputs 45 Penn Treebank's POS tags in this paper, we have new 45 sub-classes which correspond to non-entity regions such as "O-NNS" (plural nouns), "O-JJ" (adjectives), and "O-DT" (determiners).

Splitting by POS information seems useful for improving the system accuracy as well, because in the named entity recognition we must discriminate between nouns in named entities and nouns in ordinal noun phrases. In the experiments, we show this class splitting technique not only enables the feasible training but also improves the accuracy.

5.2 Word Cache and HMM Features

In addition to the standard features, we explore *word cache feature* and *HMM state feature*, mainly to solve the data sparseness problem.

Although the GENIA corpus is the largest annotated corpus for the biomedical domain, it is still small compared with other linguistic annotated corpora such as the Penn Treebank. Thus, the data sparseness problem is severe, and must be treated carefully. Usually, the data sparseness is prevented by using more general features that apply to a broader set of instances (e.g., disjunctions). While polynomial kernels in the SVM learning can effectively generate feature conjunctions, kernel functions that can effectively generate feature disjunctions are not known. Thus, we should explicitly add dimensions for such general features.

The word cache feature is defined as the disjunction of several word features as:

$$wc_{\mathbf{k}\{k_1,\cdots,k_n\},i} \equiv \vee_{k\in\mathbf{k}} w_{k,i}$$

We intend that the word cache feature captures the similarities of the patterns with a common key word such as follows.

(a) "human W_{-2} W_{-1} W_0" and "human W_{-1} W_0"
(b) "W_0 gene" and "W_0 W_1 gene"

We use a left word cache defined as $lwc_{k,i} \equiv wc_{\{-k,\cdots,0\},i}$, and a right word cache defined as $rwc_{k,i} \equiv wc_{\{1,\cdots,k\},i}$ for patterns like (a) and (b) in the above example respectively.

Kazama et al. (2001) proposed to use as features the Viterbi state sequence of a hidden Markov model (HMM) to prevent the data sparseness problem in the maximum entropy tagging model. An HMM is trained with a large number of unannotated texts by using an unsupervised learning method. Because the number of states of the HMM is usually made smaller than $|\mathcal{V}|$, the Viterbi states give smoothed but maximally informative representations of word patterns tuned for the domain, from which the raw texts are taken.

The HMM feature is defined in the same way as the word feature as follows.

$$hmm_{k,i} = \begin{cases} 1 & \text{if the Viterbi state for } W_k \text{ is} \\ & \quad \text{the } i\text{th state in the HMM's states } \mathcal{H} \\ 0 & \text{otherwise} \quad (\textit{HMM feature}) \end{cases}$$

In the experiments, we train an HMM using raw MEDLINE abstracts in the GENIA corpus, and show that the HMM state feature can improve the accuracy.

5.3 Implementation Issues

Towards practical named entity recognition using SVMs, we have tackled the following implementation issues. It would be impossible to carry out the experiments in a reasonable time without such efforts.

Parallel Training: The training of pairwise SVMs has trivial parallelism, i.e., each SVM can be trained separately. Since computers with two or more CPUs are not expensive these days, parallelization is very practical solution to accelerate the training of pairwise SVMs.

Fast Winner Finding: Although the pairwise method reduces the cost of training, it greatly increases the number of classifications needed to determine the class of one sample. For example, for our experiments using the GENIA corpus, the BIO representation with class splitting yields more than 4,000 classification pairs. Fortunately, we can stop classifications when a class gets $K-1$ votes and this stopping greatly saves classification time (Kreßel, 1998). Moreover, we can stop classifications when the current votes of a class is greater than the others' possible votes.

Support Vector Caching: In the pairwise method, though we have a large number of classifiers, each classifier shares some support vectors with other classifiers. By storing the bodies of all support vectors together and letting each classifier have only the weights, we can greatly reduce the size of the classifier. The sharing of support vectors also can be exploited to accelerate the classification by caching the value of the kernel function between a support vector and a classifiee sample.

6 Experiments

To conduct experiments, we divided 670 abstracts of the GENIA corpus (Ver. 1.1) into the training part (590 abstracts; 4,487 sentences; 133,915 words) and the test part (80 abstracts; 622 sentences; 18,211 words).[6] Texts are tokenized by using Penn Treebank's tokenizer. An HMM for the HMM state features was trained with raw abstracts of the GENIA corpus (39,116 sentences).[7] The number of states is 160. The vocabulary for the word feature is constructed by taking the most frequent 10,000 words from the above raw abstracts, the prefix/suffix/prefix list by taking the most frequent 10,000 prefixes/suffixes/substrings.[8]

The performance is measured by *precision*, *recall*, and *F-score*, which are the standard measures for the

[6] Randomly selected set used in (Shimpuku, 2002). We do not use paper titles, while he used.

[7] These do not include the sentences in the test part.

[8] These are constructed using the training part to make the comparison with the ME method fair.

Table 2: Training time and accuracy with/without the class splitting technique. The number of training samples includes SOS and EOS (special words for the start/end of a sentence).

	no splitting		**splitting**	
training samples	time (sec.)	acc. (F-score)	time (sec.)	acc. (F-score)
16,000	2,809	37.04	5,581	36.82
32,000	13,614	40.65	9,175	41.36
48,000	21,174	42.44	9,709	42.49
64,000	40,869	42.52	12,502	44.34
96,000	-	-	21,922	44.93
128,000	-	-	36,846	45.99

Table 3: Effect of each feature set assessed by adding/subtracting (F-score). Changes in bold face means positive effect.

feature set	(A) adding	(B) sub. (k=2)	(C) sub. (k=3)
Base	42.86	47.82	49.27
Left cache	43.25 (**+0.39**)	47.77 (**-0.05**)	49.02 (**-0.25**)
Right cache	42.34 (-0.52)	47.81 (**-0.01**)	49.07 (**-0.20**)
HMM state	44.70 (**+1.84**)	47.25 (**-0.57**)	48.03 (**-1.24**)
POS	44.82 (**+1.96**)	48.29 (+0.47)	48.75 (**-0.52**)
Prec. class	44.58 (**+1.72**)	43.32 (**-4.50**)	43.84 (**-5.43**)
Prefix	42.77 (-0.09)	48.11 (+0.29)	48.73 (**-0.54**)
Suffix	45.88 (**+3.02**)	47.07 (**-0.75**)	48.48 (**-0.79**)
Substring	42.16 (-0.70)	48.38 (+0.56)	50.23 (+0.96)

named entity recognition. Systems based on the BIO representation may produce an inconsistent class sequence such as "O B-DNA I-RNA O". We interpret such outputs as follows: once a named entity starts with "B-*C*" then we interpret that the named entity with class "*C*" ends only when we see another "B-" or "O-" tag.

We have implemented SMO algorithm (Platt, 1998) and techniques described in (Joachims, 1998) for soft margin SVMs in C++ programming language, and implemented support codes for pairwise classification and parallel training in Java programming language. To obtain POS information required for features and class splitting, we used an English POS tagger described in (Kazama et al., 2001).

6.1 Class Splitting Technique

First, we show the effect of the class splitting described in Section 5.1. Varying the size of training data, we compared the change in the training time and the accuracy with and without the class splitting. We used a feature set $\{\langle w, pre, suf, sub, pos\rangle_{[-2,\cdots,2]}, pc_{[-2,-1]}\}$ and the inner product kernel.[9] The training time was measured on a machine with four 700MHz PentiumIIIs and 16GB RAM. Table 2 shows the results of the experiments. Figure 1 shows the results graphically. We can see that without splitting we soon suffer from super-linearity of the SVM training, while with splitting we can handle the training with over 100,000 samples in a reasonable time. It is very important that the splitting technique does not sacrifice the accuracy for speed, rather improves the accuracy.

6.2 Word Cache and HMM State Features

In this experiment, we see the effect of the word cache feature and the HMM state feature described in Section 3.4. The effect is assessed by the accuracy gain observed by adding each feature set to a base feature set and the accuracy degradation observed by subtracting it from a (complete) base set. The first column (A) in Table 3 shows an adding case where the base feature set is $\{w_{[-2,\cdots,2]}\}$. The columns (B) and (C) show subtracting cases where the base feature set is $\{\langle w, pre, suf, sub, pos, hmm\rangle_{[-k,\cdots,k]}, lwc_k, rwc_k, pc_{[-2,-1]}\}$ with $k = 2$ and $k = 3$ respectively. The kernel function is the inner product. We can see that word cache and HMM state features surely improve the recognition accuracy. In the table, we also included the accuracy change for other standard features. Preceeding classes and suffixes are definitely helpful. On the other hand, the substring feature is not effective in our setting. Although the effects of part-of-speech tags and prefixes are not so definite, it can be said that they are practically effective since they show positive effects in the case of the maximum performance.

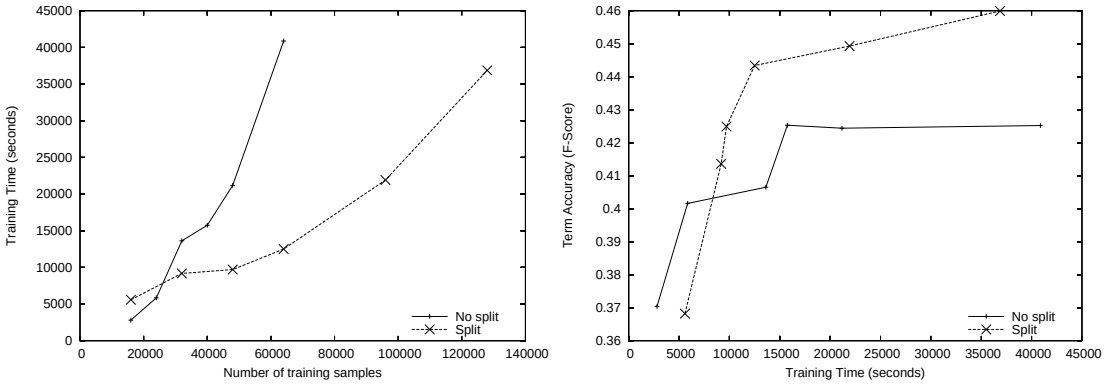
6.3 Comparison with the ME Method

In this set of experiments, we compare our SVM-based system with a named entity recognition system based on the Maximum Entropy method. For the SVM system, we used the feature set $\{\langle w, pre, suf, pos, hmm\rangle_{[-3,\cdots,3]}, lwc_3, rwc_3, pc_{[-2,-1]}\}$, which is shown to be the best in the previous experiment. The compared system is a maximum entropy tagging model described in (Kazama et al., 2001). Though it supports several character type features such as *number* and *hyphen* and some conjunctive features such as word *n*-gram, we do not use these features to compare the performance under as close a condition as possible. The feature set used in the maximum entropy system is expressed as $\{\langle w, pre, suf, pos, hmm\rangle_{[-2,\cdots,2]}, pc_{[-2,-1]}\}$.[10] Both systems use the BIO representation with splitting.

Table 4 shows the accuracies of both systems. For the SVM system, we show the results with the inner product kernel and several polynomial kernels. The row "All (id)" shows the accuracy from the view-

[9] Soft margin constant C is 1.0 throughout the experiments.

[10] When the width becomes $[-3,\cdots,3]$, the accuracy degrades (53.72 to 51.73 in F-score).



(a) Training size vs. time

(b) Training time vs. accuracy

Figure 1: Effect of the class splitting technique.

point of the identification task, which only finds the named entity regions. The accuracies for several major entity classes are also shown. The SVM system with the 2-dimensional polynomial kernel achieves the highest accuracy. This comparison may be unfair since a polynomial kernel has the effect of using conjunctive features, while the ME system does not use such conjunctive features. Nevertheless, the facts: we can introduce the polynomial kernel very easily; there are very few parameters to be tuned;[11] we could achieve the higher accuracy; show an advantage of the SVM system.

It will be interesting to discuss why the SVM systems with the inner product kernel (and the polynomial kernel with $d = 1$) are outperformed by the ME system. We here discuss two possible reasons. The first is that the SVM system does not use a dynamic decision such as the Viterbi algorithm, while the ME system uses it. To see this, we degrade the ME system so that it predicts the classes deterministically without using the Viterbi algorithm. We found that this system only marks 51.54 in F-score. Thus, it can be said that a dynamic decision is important for this named entity task. However, although a method to convert the outputs of a binary SVM to probabilistic values is proposed (Platt, 1999), the way to obtain meaningful probabilistic values needed in Viterbi-type algorithms from the outputs of a multi-class SVM is unknown. Solving this problem is certainly a part of the future work. The second possible reason is that the SVM system in this paper does not use any cut-off or feature truncation method to remove data noise, while the ME system uses a simple feature cut-off method.[12] We observed that the ME system without the cut-off only marks 49.11 in F-score. Thus, such a noise reduction method is also important. However, the cut-off method for the ME method cannot be applied without modification since, as described in Section 3.4, the definition of the features are different in the two approaches. It can be said the features in the ME method is "finer" than those in SVMs. In this sense, the ME method allows us more flexible feature selection. This is an advantage of the ME method.

The accuracies achieved by both systems can be said high compared with those of the previous methods if we consider that we have 24 named entity classes. However, the accuracies are not sufficient for a practical use. Though higher accuracy will be achieved with a larger annotated corpus, we should also explore more effective features and find effective feature combination methods to exploit such a large corpus maximally.

7 Conclusion

We have described the use of Support Vector Machines for the biomedical named entity recognition task. To make the training of SVMs with the GENIA corpus practical, we proposed to split the non-entity class by using POS information. In addition, we explored the new types of features, word cache and HMM states, to avoid the data sparseness problem. In the experiments, we have shown that the class splitting technique not only makes training feasible but also improves the accuracy. We have also shown that the proposed new features also improve the accuracy and the SVM system with the polynomial kernel function outperforms the ME-based system.

Acknowledgements

We would like to thank Dr. Jin-Dong Kim for providing us easy-to-use preprocessed training data.

[11] C, s, r, and d

[12] Features that occur less than 10 times are removed.

Table 4: Comparison: The SVM-based system and the ME-based system. (precision/recall/F-score)

		SVM				**ME**
		inner product	polynomial ($s = 0.01, r = 1.0$))			
type	#		$d = 1$	$d = 2$	$d = 3$	
All	(2,782)	50.7/49.8/50.2	54.6/48.8/51.5	**56.2**/52.8/**54.4**	55.1/51.5/53.2	53.4/**53.0**/53.2
All(id)		71.8/70.4/71.1	75.0/67.1/70.8	**75.9**/71.4/**73.6**	75.3/70.3/72.7	73.5/**72.9**/73.2
protein	(709)	47.2/55.2/50.8	45.7/64.9/53.6	49.2/66.4/56.5	48.7/64.7/55.6	49.1/62.1/54.8
DNA	(460)	39.9/37.6/38.7	48.2/31.5/38.1	49.6/37.0/42.3	47.9/37.4/42.0	47.3/39.6/43.1
cell line	(121)	54.8/47.1/50.7	61.2/43.0/50.5	60.2/46.3/52.3	62.2/46.3/53.1	58.0/53.7/55.8
cell type	(199)	67.6/74.4/70.8	67.4/74.9/71.0	70.0/75.4/72.6	68.6/72.4/70.4	69.9/72.4/71.1
lipid	(109)	77.0/61.5/68.4	83.3/50.5/62.9	82.7/61.5/70.5	79.2/56.0/65.6	68.9/65.1/67.0
other names	(590)	52.5/53.9/53.2	60.2/55.9/58.0	59.3/58.0/58.6	58.9/57.8/58.3	59.0/61.7/60.3

References

A. L. Berger, S. A. Della Pietra, and V. J. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

N. Collier, C. Nobata, and J. Tsujii. 2000. Extracting the names of genes and gene products with a hidden Markov model. In *Proc. of COLING 2000*, pages 201–207.

C. Cortes and V. Vapnik. 1995. Support vector networks. *Machine Learning*, 20:273–297.

C. Hsu and C. Lin. 2002. A comparison of methods for multiclass Support Vector Machines. In *IEEE Transactions on Neural Networks*. to appear.

T. Joachims. 1998. Making large-scale support vector machine learning practical. In *Advances in Kernel Methods*, pages 169–184. The MIT Press.

J. Kazama, Y. Miyao, and J. Tsujii. 2001. A maximum entropy tagger with unsupervised hidden markov models. In *Proc. of the 6th NLPRS*, pages 333–340.

J. Kim, T. Ohta, Y. Tateisi, H. Mima, and J. Tsujii. 2001. XML-based linguistic annotation of corpus. In *Proc. of the First NLP and XML Workshop*.

U. Kreßel. 1998. Pairwise classification and support vector machines. In *Advances in Kernel Methods*, pages 255–268. The MIT Press.

T. Kudo and Y. Matsumoto. 2000. Use of support vector learning for chunk identification. In *Proc. of CoNLL-2000 and LLL-2000*.

T. Kudo and Y. Matsumoto. 2001. Chunking with Support Vector Machines. In *Proc. of NAACL 2001*, pages 192–199.

T. Nakagawa, T. Kudoh, and Y. Matsumoto. 2001. Unknown word guessing and part-of-speech tagging using support vector machines. In *Proc. of the 6th NLPRS*, pages 325–331.

National Library of Medicine. 1999. MEDLINE. available at http://www.ncbi.nlm.nih.gov/.

C. Nobata, N. Collier, and J. Tsujii. 1999. Automatic term identification and classification in biology texts. In *Proc. of the 5th NLPRS*, pages 369–374.

C. Nobata, N. Collier, and J. Tsujii. 2000. Comparison between tagged corpora for the named entity task. In *Proc. of the Workshop on Comparing Corpora (at ACL'2000)*, pages 20–27.

Y. Ohta, Y. Yamamoto, T. Okazaki, I. Uchiyama, and T. Takagi. 1997. Automatic construction of knowledge base from biological papers. In *Proc. of the 5th ISMB*, pages 218–225.

T. Ohta, Y. Tateisi, J. Kim, H. Mima, and Tsujii J. 2002. The GENIA corpus: An annotated research abstract corpus in molecular biology domain. In *Proc. of HLT 2002*.

J. C. Platt. 1998. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods*, pages 185–208. The MIT Press.

J. C. Platt. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers*.

D. Proux, F. Prechenmann, and L. Julliard. 2000. A pragmatic information extraction strategy for gathering data on genetic interactions. In *Proc. of the 8th ISMB*, pages 279–285.

L. A. Ramshaw and M. P. Marcus. 1995. Text chunking using transformation-based learning. In *Proc. of the 3rd ACL Workshop on Very Large Corpora*.

A. Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*, pages 133–142.

S. Shimpuku. 2002. A medical/biological term recognizer with a term hidden Markov model incorporating multiple information sources. A master thesis. University of Tokyo.

K. Uchimoto, M. Murata, Q. Ma, H. Ozaku, and H. Isahara. 2000. Named entity extraction based on a maximum entropy model and transformation rules. In *Proc. of the 38th ACL*, pages 326–335.

V. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer Verlag.

A. Yakushiji, Y. Tateisi, Y. Miyao, and J. Tsujii. 2001. Event extraction from biomedical papers using a full parser. In *Proc. of PSB 2001*, pages 408–419.

H. Yamada, T. Kudo, and Y. Matsumoto. 2000. Using substrings for technical term extraction and classification. *IPSJ SIGNotes*, (NL-140):77–84. (in Japanese).