Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia, July 2002, pp. 497-504.

Revision Learning and its Application to Part-of-Speech Tagging

Tetsuji Nakagawa* and **Taku Kudo** and **Yuji Matsumoto**
tetsu-na@plum.freemail.ne.jp,{taku-ku,matsu}@is.aist-nara.ac.jp
Graduate School of Information Science
Nara Institute of Science and Technology
8916−5 Takayama, Ikoma, Nara 630−0101, Japan

Abstract

This paper presents a revision learning method that achieves high performance with small computational cost by combining a model with high generalization capacity and a model with small computational cost. This method uses a high capacity model to revise the output of a small cost model. We apply this method to English part-of-speech tagging and Japanese morphological analysis, and show that the method performs well.

1 Introduction

Recently, corpus-based approaches have been widely studied in many natural language processing tasks, such as part-of-speech (POS) tagging, syntactic analysis, text categorization and word sense disambiguation. In corpus-based natural language processing, one important issue is to decide which learning model to use. Various learning models have been studied such as Hidden Markov models (HMMs) (Rabiner and Juang, 1993), decision trees (Breiman et al., 1984) and maximum entropy models (Berger et al., 1996). Recently, Support Vector Machines (SVMs) (Vapnik, 1998; Cortes and Vapnik, 1995) are getting to be used, which are supervised machine learning algorithm for binary classification. SVMs have good generalization performance and can handle a large number of features, and are applied to some tasks successfully (Joachims, 1998; Kudoh and Matsumoto, 2000). However, their computational cost is large and is a weakness of SVMs. In general, a trade-off between capacity and computational cost of learning models exists. For example, SVMs have relatively high generalization capacity, but have high computational cost. On the other hand, HMMs have lower computational cost, but have lower capacity and difficulty in handling data with a large number of features. Learning models with higher capacity may not be of practical use because of their prohibitive computational cost. This problem becomes more serious when a large amount of data is used.

To solve this problem, we propose a revision learning method which combines a model with high generalization capacity and a model with small computational cost to achieve high performance with small computational cost. This method is based on the idea that processing the entire target task using a model with higher capacity is wasteful and costly, that is, if a large portion of the task can be processed easily using a model with small computational cost, it should be processed by such a model, and only difficult portion should be processed by the model with higher capacity.

Revision learning can handle a general multi-class classification problem, which includes POS tagging, text categorization and many other tasks in natural language processing. We apply this method to English POS tagging and Japanese morphological analysis.

This paper is organized as follows: Section 2 describes the general multi-class classification

* Presently with Oki Electric Industry

problem and the one-versus-rest method which is known as one of the solutions for the problem. Section 3 introduces revision learning, and discusses how to combine learning models. Section 4 describes one way to conduct Japanese morphological analysis with revision learning. Section 5 shows experimental results of English POS tagging and Japanese morphological analysis with revision learning. Section 6 discusses related works, and Section 7 gives conclusion.

2 Multi-Class Classification Problems and the One-versus-Rest Method

Let us consider the problem to decide the class of an example $\mathbf{x}$ among multiple classes. Such a problem is called multi-class classification problem. Many tasks in natural language processing such as POS tagging are regarded as a multi-class classification problem. When we only have binary (positive or negative) classification algorithm at hand, we have to reformulate a multi-class classification problem into a binary classification problem. We assume a binary classifier $f(\mathbf{x})$ that returns positive or negative real value for the class of $\mathbf{x}$, where the absolute value $|f(\mathbf{x})|$ reflects the confidence of the classification.

The one-versus-rest method is known as one of such methods (Allwein et al., 2000). For one training example of a multi-class problem, this method creates a positive training example for the true class and negative training examples for the other classes. As a result, positive and negative examples for each class are generated. Suppose we have five candidate classes A, B, C, D and E , and the true class of $\mathbf{x}$ is B. Figure 1 (left) shows the created training examples. Note that there are only two labels (positive and negative) in contrast with the original problem. Then a binary classifier for each class is trained using the examples, and five classifiers are created for this problem. Given a test example $\mathbf{x}'$, all the classifiers classify the example whether it belongs to a specific class or not. Its class is decided by the classifier that gives the largest value of $f(\mathbf{x}')$. The algorithm is shown in Figure 2 in a pseudo-code.

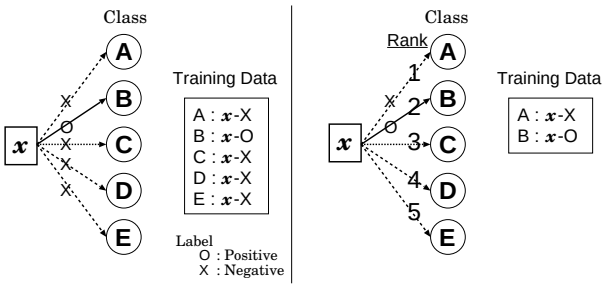

Figure 1: One-versus-Rest Method (left) and Revision Learning (right)

```
# Training Procedure of One-versus-Rest
# This procedure is given training examples
# {(x_i, y_i)}, and creates classifiers.
#   C = {c_0, ..., c_{k-1}}:  the set of classes,
#   x_i:  the ith training example,
#   y_i ∈ C:  the class of x_i,
#   k:  the number of classes,
#   l:  the number of training examples,
#   f_c(·):  the binary classifier for the class c
#            (see the text).
procedure Train_OVR({(x_0, y_0), ..., (x_{l-1}, y_{l-1})})
begin
 # Create the training data with binary label
 for i := 0 to l − 1
 begin
  for j := 0 to k − 1
  begin
   if c_j ≠ y_i then
     Add x_i to the training data for the class c_j as a
     negative example.
   else
     Add x_i to the training data for the class c_j as a
     positive example.
  end
 end
 # Train the binary classifiers
 for j := 0 to k − 1
   Train the classifier f_{c_j}(·) using the training data.
end

# Test Function of One-versus-Rest
# This function is given a test example and
# returns the predicted class of it.
#   C = {c_0, ..., c_{k-1}}:  the set of classes,
#   x:  the test example,
#   k:  the number of classes,
#   f_c(·):  binary classifier trained with the
#            algorithm above.
function Test_OVR(x)
begin
 for j := 0 to k − 1
   confidence_j := f_{c_j}(x)
 return c_{argmax_j confidence_j}
end
```

Figure 2: Algorithm of One-versus-Rest

However, this method has the problem of being computationally costly in training, because the negative examples are created for all the classes other than the true class, and the total number of the training examples becomes large (which is equal to the number of original training examples multiplied by the number of classes). The computational cost in testing is also large, because all the classifiers have to work on each test example.

3 Revision Learning

As discussed in the previous section, the one-versus-rest method has the problem of computational cost. This problem become more serious when costly binary classifiers are used or when a large amount of data is used. To cope with this problem, let us consider the task of POS tagging. Most portions of POS tagging is not so difficult and a simple POS-based HMMs learning [1] achieves more than 95% accuracy simply using the POS context (Brants, 2000). This means that the low capacity model is enough to do most portions of the task, and we need not use a high accuracy but costly algorithm in every portion of the task. This is the base motivation of the revision model we are proposing here.

Revision learning uses a binary classifier with higher capacity to revise the errors made by the stochastic model with lower capacity as follows: During the training phase, a ranking is assigned to each class by the stochastic model for a training example, that is, the candidate classes are sorted in descending order of its conditional probability given the example. Then, the classes are checked in their ranking order to create binary classifiers as follows. If the class is incorrect (i.e. it is not equal to the true class for the example), the example is added to the training data for that class as a negative example, and the next ranked class is checked. If the class is correct, the example is added to the training data for that class as a positive example, and the remaining ranked classes are not taken into consideration (Figure 1, right). Using these training data, binary classifiers are created. Note that each classifier is a pure binary classifier regardless with the number of classes in the original problem. The binary classifier is trained just for answering whether the output from the stochastic model is correct or not.

During the test phase, first the ranking of the candidate classes for a given example is assigned by the stochastic model as in the training. Then the binary classifier classifies the example according to the ranking. If the classifier answers the example as incorrect, the next highest ranked class becomes the next candidate for checking. But if the example is classified as correct, the class of the classifier is returned as the answer for the example. The algorithm is shown in Figure 3.

The amount of training data generated in the revision learning can be much smaller than that in one-versus-rest. Since, in revision learning, negative examples are created only when the stochastic model fails to assign the highest probability to the correct POS tag, whereas negative examples are created for all but one class in the one-versus-rest method. Moreover, testing time of the revision learning is shorter, because only one classifier is called as far as it answers as correct, but all the classifiers are called in the one-versus-rest method.

4 Morphological Analysis with Revision Learning

We introduced revision learning for multi-class classification in the previous section. However, Japanese morphological analysis cannot be regarded as a simple multi-class classification problem, because words in a sentence are not separated by spaces in Japanese and the morphological analyzer has to segment the sentence into words as well as to decide the POS tag of the words. So in this section, we describe how to apply revision learning to Japanese morphological analysis.

For a given sentence, a lattice consisting of all possible morphemes can be built using a mor-

[1] HMMs can be applied to either of unsupervised or supervised learning. In this paper, we use the latter case, i.e., visible Markov Models, where POS-tagged data is used for training.

```
# Training Procedure of Revision Learning
# This procedure is given training examples
# {(x_i, y_i)}, and creates classifiers.
#   C = {c_0, ..., c_{k-1}}:  the set of classes,
#   x_i:  the ith training example,
#   y_i ∈ C:  the class of x_i,
#   k:  the number of classes,
#   l:  the number of training examples,
#   n_i:  the ordered indexes of C
#         (see the following code),
#   f_c(·):  the binary classifier for the class c
#           (see the text).
procedure Train_RL({(x_0, y_0), ..., (x_{l-1}, y_{l-1})})
begin
 # Create the training data with binary label
 for i := 0 to l − 1
 begin
  Call the stochastic model to obtain the
  ordered indexes {n_0, ..., n_{k-1}}
  such that P(c_{n_0}|x_i) ≥ ··· ≥ P(c_{n_{k-1}}|x_i).
  for j := 0 to k − 1
  begin
   if c_{n_j} ≠ y_i then
    Add x_i to the training data for the class c_{n_j} as a
    negative example.
   else
   begin
    Add x_i to the training data for the class c_{n_j} as a
    positive example.
    break
   end
  end
 end
 # Train the binary classifiers
 for j := 0 to k − 1
  Train the classifier f_{c_j}(·) using the training data.
end

# Test Function of Revision Learning
# This function is given a test example and
# returns the predicted class of it.
#   C = {c_0, ..., c_{k-1}}:  the set of classes,
#   x:  the test example,
#   k:  the number of classes,
#   n_i:  the ordered indexes of C
#         (see the following code),
#   f_c(·):  binary classifier trained with the
#           algorithm above.
function Test_RL(x)
begin
 Call the stochastic model to obtain the
 ordered indexes {n_0, ..., n_{k-1}}
 such that P(c_{n_0}|x) ≥ ··· ≥ P(c_{n_{k-1}}|x).
 for j := 0 to k − 1
  if f_{c_{n_j}}(x) > 0 then
   return c_{n_j}
 return undecidable
end
```

Figure 3: Algorithm of Revision Learning

pheme dictionary as in Figure 4. Morphological analysis is conducted by choosing the most likely path on it. We adopt HMMs as the stochastic model and SVMs as the binary classifier. For any sub-paths from the beginning of the sentence (BOS) in the lattice, its generative probability can be calculated using HMMs (Nagata, 1999). We first pick up the end node of the sentence as the current state node, and repeat the following revision learning process backward until the beginning of the sentence. Rankings are calculated by HMMs to all the nodes connected to the current state node, and the best of these nodes is identified based on the SVMs classifiers. The selected node then becomes the current state node in the next round. This can be seen as SVMs deciding whether two adjoining nodes in the lattice are connected or not.

In Japanese morphological analysis, for any given morpheme μ, we use the following features for the SVMs:

1. the POS tags, the lexical forms and the inflection forms of the two morphemes preceding μ;
2. the POS tags and the lexical forms of the two morphemes following μ;
3. the lexical form and the inflection form of μ.

The preceding morphemes are unknown because the processing is conducted from the end of the sentence, but HMMs can predict the most likely preceding morphemes, and we use them as the features for the SVMs.

English POS tagging is regarded as a special case of morphological analysis where the segmentation is done in advance, and can be conducted in the same way. In English POS tagging, given a word w, we use the following features for the SVMs:

1. the POS tags and the lexical forms of the two words preceding w, which are given by HMMs;
2. the POS tags and the lexical forms of the two words following w;
3. the lexical form of w and the prefixes and suffixes of up to four characters, the exis-

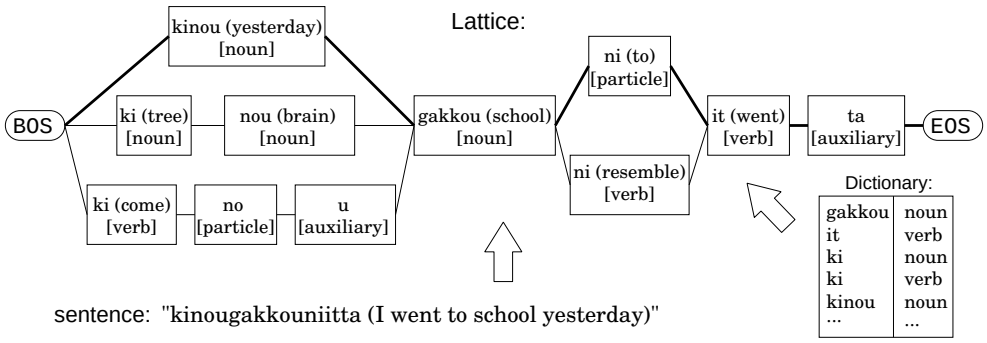

Figure 4: Example of Lattice for Japanese Morphological Analysis

tence of numerals, capital letters and hyphens in w.

5 Experiments

This section gives experimental results of English POS tagging and Japanese morphological analysis with revision learning.

5.1 Experiments of English Part-of-Speech Tagging

Experiments of English POS tagging with revision learning (RL) are performed on the Penn Treebank WSJ corpus. The corpus is randomly separated into training data of 41,342 sentences and test data of 11,771 sentences. The dictionary for HMMs is constructed from all the words in the training data.

T3 of ICOPOST release 0.9.0 (Schröder, 2001) is used as the stochastic model for ranking stage. This is equivalent to POS-based second order HMMs. SVMs with second order polynomial kernel are used as the binary classifier.

The results are compared with TnT (Brants, 2000) based on second order HMMs, and with POS tagger using SVMs with one-versus-rest (1-v-r) (Nakagawa et al., 2001).

The accuracies of those systems for known words, unknown words and all the words are shown in Table 1. The accuracies for both known words and unknown words are improved through revision learning. However, revision learning could not surpass the one-versus-rest. The main difference in the accuracies stems from those for unknown words. The reason for that seems to be that the dictionary of HMMs for POS tagging is obtained from the training data, as a result, virtually no unknown words exist in the training data, and the HMMs never make mistakes for unknown words during the training. So no example of unknown words is available in the training data for the SVM reviser. This is problematic: Though the HMMs handles unknown words with an exceptional method, SVMs cannot learn about errors made by the unknown word processing in the HMMs. To cope with this problem, we force the HMMs to make mistakes by eliminating low frequent words from the dictionary. We eliminated the words appearing only once in the training data so as to make SVMs to learn about unknown words. The results are shown in Table 1 (row "cutoff-1"). Such procedure improves the accuracies for unknown words.

One advantage of revision learning is its small computational cost. We compare the computation time with the HMMs and the one-versus-rest. We also use SVMs with linear kernel function that has lower capacity but lower computational cost compared to the second order polynomial kernel SVMs. The experiments are performed on an Alpha 21164A 500MHz processor. Table 2 shows the total number of training examples, training time, testing time and accuracy for each of the five systems. The training time and the testing time of revision learning are considerably smaller than those of the one-versus-rest. Using linear kernel, the accuracy decreases a little, but the computational cost is much lower than the second order polynomial kernel.

	Accuracy (Known Words / Unknown Words)	Number of Errors
T3 Original	96.59% (96.90% / 82.74%)	9720
with RL	96.93% (97.23% / 83.55%)	8734
with RL (cutoff-1)	96.98% (97.25% / 85.11%)	8588
TnT	96.62% (96.90% / 84.19%)	9626
SVMs 1-v-r	97.11% (97.34% / 86.80%)	8245

Table 1: Result of English POS Tagging

	Total Number of Examples for SVMs	Training Time (hour)	Testing Time (second)	Accuracy
T3 Original	—	0.004	89	96.59%
with RL (polynomial kernel, cutoff-1)	1027840	16	2089	96.98%
with RL (linear kernel, cutoff-1)	1027840	2	129	96.94%
TnT	—	0.002	4	96.62%
SVMs 1-v-r	999984×50	625	55239	97.11%

Table 2: Computational Cost of English POS Tagging

5.2 Experiments of Japanese Morphological Analysis

We use the RWCP corpus and some additional spoken language data for the experiments of Japanese morphological analysis. The corpus is randomly separated into training data of 33,831 sentences and test data of 3,758 sentences. As the dictionary for HMMs, we use IPADIC version 2.4.4 with 366,878 morphemes (Matsumoto and Asahara, 2001) which is originally constructed for the Japanese morphological analyzer ChaSen (Matsumoto et al., 2001).

A POS bigram model and ChaSen version 2.2.8 based on variable length HMMs are used as the stochastic models for the ranking stage, and SVMs with the second order polynomial kernel are used as the binary classifier.

We use the following values to evaluate Japanese morphological analysis:

$$\text{recall} = \frac{\langle \#\text{ of correct morphemes in system's output} \rangle}{\langle \#\text{ of morphemes in test data} \rangle},$$

$$\text{precision} = \frac{\langle \#\text{ of correct morphemes in system's output} \rangle}{\langle \#\text{ of morphemes in system's output} \rangle},$$

$$\text{F-measure} = \frac{2 \times \text{recall} \times \text{precision}}{\text{recall} + \text{precision}}.$$

The results of the original systems and those with revision learning are shown in Table 3, which provides the recalls, precisions and F-measures for two cases, namely segmentation (i.e. segmentation of the sentences into morphemes) and tagging (i.e. segmentation and POS tagging). The one-versus-rest method is not used because it is not applicable to morphological analysis of non-segmented languages directly.

When revision learning is used, all the measures are improved for both POS bigram and ChaSen. Improvement is particularly clear for the tagging task.

The numbers of correct morphemes for each POS category tag in the output of ChaSen with and without revision learning are shown in Table 4. Many particles are correctly revised by revision learning. The reason is that the POS tags for particles are often affected by the following words in Japanese, and SVMs can revise such particles because it uses the lexical forms of the following words as the features. This is the advantage of our method compared to simple HMMs, because HMMs have difficulty in handling a lot of features such as the lexical forms of words.

6 Related Works

Our proposal is to revise the outputs of a stochastic model using binary classifiers. Brill studied transformation-based error-driven learning (TBL) (Brill, 1995), which conducts POS tagging by applying the transformation rules to the POS tags of a given sentence, and has a resemblance to revision learning in that the second model revises the output of the first model.

		Word Segmentation			Tagging			Training Time (hour)	Testing Time (second)
		Recall	Precision	F-measure	Recall	Precision	F-measure		
POS bigram	Original	98.06%	98.77%	98.42%	95.61%	96.30%	95.96%	0.02	8
	with RL	99.06%	99.27%	99.16%	98.13%	98.33%	98.23%	11	184
ChaSen	Original	99.06%	99.20%	99.13%	97.67%	97.81%	97.74%	0.05	15
	with RL	99.22%	99.34%	99.28%	98.26%	98.37%	98.32%	6	573

Table 3: Result of Morphological Analysis

Part-of-Speech	# in Test Data	Original	with RL	Difference
Noun	41512	40355	40556	+201
Prefix	817	781	784	+3
Verb	8205	8076	8115	+39
Adjective	678	632	655	+23
Adverb	779	735	750	+15
Adnominal	378	373	373	0
Conjunction	258	243	243	0
Particle	20298	19686	19942	+256
Auxiliary	4419	4333	4336	+3
Interjection	94	90	91	+1
Symbol	15665	15647	15651	+4
Others	1	1	1	0
Filler	43	36	36	0

Table 4: The Number of Correctly Tagged Morphemes for Each POS Category Tag

However, our method differs from TBL in two ways. First, our revision learner simply answers whether a given pattern is correct or not, and any types of binary classifiers are applicable. Second, in our model, the second learner is applied to the output of the first learner only once. In contrast, rewriting rules are applied repeatedly in the TBL.

Recently, combinations of multiple learners have been studied to achieve high performance (Alpaydm, 1998). Such methodologies to combine multiple learners can be distinguished into two approaches: one is the multi-expert method and the other is the multi-stage method. In the former, each learner is trained and answers independently, and the final decision is made based on those answers. In the latter, the multiple learners are ordered in series, and each learner is trained and answers only if the previous learner rejects the examples. Revision learning belongs to the latter approach. In POS tagging, some studies using the multi-expert method were conducted (van Halteren et al., 2001; Màrquez et al., 1999), and Brill and Wu (1998) combined maximum entropy models, TBL, unigram and trigram, and achieved higher accuracy than any of the four learners (97.2% for WSJ corpus). Regarding the multi-stage methods, cascading (Alpaydin and Kaynak, 1998) is well known, and Even-Zohar and Roth (2001) proposed the sequential learning model and applied it to POS tagging. Their methods differ from revision learning in that each learner behaves in the same way and more than one learner is used in their methods, but in revision learning the stochastic model assigns rankings to candidates and the binary classifier selects the output. Furthermore, mistakes made by a former learner are fatal in their methods, but is not so in revision learning because the binary classifier works to revise them.

The advantage of the multi-expert method is that each learner can help each other even if it has some weakness, and generalization errors can be decreased. On the other hand, the computational cost becomes large because each learner is trained using every training data and answers for every test data. In contrast, multi-stage methods can decrease the computational cost, and seem to be effective when a large amount of data is used or when a learner with high computational cost such as SVMs is used.

7 Conclusion

In this paper, we proposed the revision learning method which combines a stochastic model and a binary classifier to achieve higher performance with lower computational cost. We applied it to English POS tagging and Japanese morphological analysis, and showed improvement of accuracy with small computational cost.

Compared to the conventional one-versus-rest method, revision learning has much lower computational cost with almost comparable accuracy. Furthermore, it can be applied not only to a simple multi-class classification task but also to a wider variety of problems such as Japanese morphological analysis.

Acknowledgments

We would like to thank Ingo Schröder for making ICOPOST publicly available.

References

Erin L. Allwein, Robert E. Schapire, and Yoram Singer. 2000. Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers. In *Proceedings of 17th International Conference on Machine Learning*, pages 9–16.

Ethem Alpaydin and Cenk Kaynak. 1998. Cascading Classifiers. *Kybernetika*, 34(4):369–374.

Ethem Alpaydm. 1998. Techniques for Combining Multiple Learners. In *Proceedings of Engineering of Intelligent Systems '98 Conference*.

Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. 1996. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 22(1):39–71.

Thorsten Brants. 2000. TnT — A Statistical Part-of-Speech Tagger. In *Proceedings of ANLP-NAACL 2000*, pages 224–231.

Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. 1984. *Classification and Regression Trees*. Wadsworth and Brooks.

Eric Brill and Jun Wu. 1998. Classifier Combination for Improved Lexical Disambiguation. In *Proceedings of the Thirty-Sixth Annual Meeting of the Association for Computational Linguistics and Seventeenth International Conference on Computational Linguistics*, pages 191–195.

Eric Brill. 1995. Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part-of-Speech Tagging. *Computational Linguistics*, 21(4):543–565.

Corinna Cortes and Vladimir Vapnik. 1995. Support Vector Networks. *Machine Learning*, 20:273–297.

Yair Even-Zohar and Dan Roth. 2001. A Sequential Model for Multi-Class Classification. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, pages 10–19.

Thorsten Joachims. 1998. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In *Proceedings of the 10th European Conference on Machine Learning*, pages 137–142.

Taku Kudoh and Yuji Matsumoto. 2000. Use of Support Vector Learning for Chunk Identification. In *Proceedings of the Fourth Conference on Computational Natural Language Learning*, pages 142–144.

Lluiís Màrquez, Horacio Rodríguez, Josep Carmona, and Josep Montolio. 1999. Improving POS Tagging Using Machine-Learning Techniques. In *Proceedings of 1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 53–62.

Yuji Matsumoto and Masayuki Asahara. 2001. *IPADIC User's Manual version 2.2.4*. Nara Institute of Science and Technology. (in Japanese).

Yuji Matsumoto, Akira Kitauchi, Tatsuo Yamashita, Yoshitaka Hirano, Hiroshi Matsuda, Kazuma Takaoka, and Masayuki Asahara. 2001. *Morphological Analysis System ChaSen version 2.2.8 Manual*. Nara Institute of Science and Technology.

Masaaki Nagata. 1999. *Japanese Language Processing Based on Stochastic Models*. Kyoto University, Doctoral Thesis. (in Japanese).

Tetsuji Nakagawa, Taku Kudoh, and Yuji Matsumoto. 2001. Unknown Word Guessing and Part-of-Speech Tagging Using Support Vector Machines. In *Proceedings of 6th Natural Language Processing Pacific Rim Symposium*, pages 325–331.

Lawrence R. Rabiner and Biing-Hwang Juang. 1993. *Fundamentals of Speech Recognition*. PTR Prentice-Hall.

Ingo Schröder. 2001. ICOPOST — Ingo's Collection Of POS Taggers. `http://nats-www.informatik.uni-hamburg.de/~ingo/icopost/`.

Hans van Halteren, Jakub Zavrel, and Walter Daelemans. 2001. Improving Accuracy in Word-class Tagging through Combination of Machine Learning Systems. *Computational Linguistics*, 27(2):199–230.

Vladimir Vapnik. 1998. *Statistical Learning Theory*. Springer.