Text Chunking by Combining Hand-Crafted Rules and Memory-Based Learning

Seong-Bae Park **Byoung-Tak Zhang**
School of Computer Science and Engineering
Seoul National University
Seoul 151-744, Korea
{sbpark,btzhang}@bi.snu.ac.kr

Abstract

This paper proposes a hybrid of hand-crafted rules and a machine learning method for chunking Korean. In the partially free word-order languages such as Korean and Japanese, a small number of rules dominate the performance due to their well-developed postpositions and endings. Thus, the proposed method is primarily based on the rules, and then the residual errors are corrected by adopting a memory-based machine learning method. Since the memory-based learning is an efficient method to handle exceptions in natural language processing, it is good at checking whether the estimates are exceptional cases of the rules and revising them. An evaluation of the method yields the improvement in F-score over the rules or various machine learning methods alone.

1 Introduction

Text chunking has been one of the most interesting problems in natural language learning community since the first work of (Ramshaw and Marcus, 1995) using a machine learning method. The main purpose of the machine learning methods applied to this task is to capture the hypothesis that best determine the chunk type of a word, and such methods have shown relatively high performance in English (Kudo and Matsumoto, 2000; Zhang et. al, 2001). In order to do it, various kinds of information, such as lexical information, part-of-speech and grammatical relation, of the neighboring words is used. Since the position of a word plays an important role as a syntactic constraint in English, the methods are successful even with local information.

However, these methods are not appropriate for chunking Korean and Japanese, because such languages have a characteristic of partially free word-order. That is, there is a very weak positional constraint in these languages. Instead of positional constraints, they have overt postpositions that restrict the syntactic relation and composition of phrases. Thus, unless we concentrate on the postpositions, we must enlarge the neighboring window to get a good hypothesis. However, enlarging the window size will cause the *curse of dimensionality* (Cherkassky and Mulier, 1998), which results in the deficiency in the generalization performance.

Especially in Korean, the postpositions and the endings provide important information for noun phrase and verb phrase chunking respectively. With only a few simple rules using such information, the performance of chunking Korean is as good as the rivaling other inference models such as machine learning algorithms and statistics-based methods (Shin, 1999). Though the rules are approximately correct for most cases drawn from the domain on which the rules are based, the knowledge in the rules is not necessarily well-represented for any given set of cases. Since chunking is usually processed in the earlier step of natural language processing, the errors made in this step have a fatal influence on the following steps. Therefore, the exceptions that are ignored by the rules must be com-

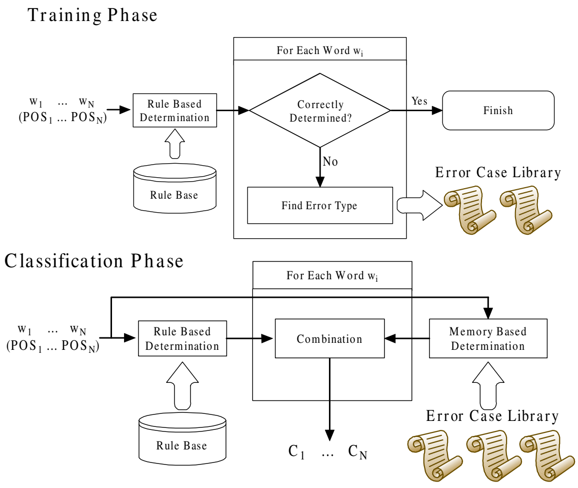

Figure 1: The structure of Korean chunking model. This figure describes a sentence-based learning and classification.

pensated for by some special treatments of them for higher performance.

To solve this problem, we have proposed a combining method of the rules and the k-nearest neighbor (k-NN) algorithm (Park and Zhang, 2001). The problem in this method is that it has redundant k-NNs because it maintains a separate k-NN for each kind of errors made by the rules. In addition, because it applies a k-NN and the rules to each examples, it requires more computations than other inference methods.

The goal of this paper is to provide a new method for chunking Korean by combining the hand-crafted rules and a machine learning method. The chunk type of a word in question is determined by the rules, and then verified by the machine learning method. The role of the machine learning method is to determine whether the current context is an exception of the rules. Therefore, a *memory-based learning* (MBL) is used as a machine learning method that can handle exceptions efficiently (Daelemans et. al, 1999).

The rest of the paper is organized as follows. Section 2 explains how the proposed method works. Section 3 describes the rule-based method for chunking Korean and Section 4 explains chunking by memory-based learning. Section 5 presents the experimental results. Section 6 introduces the issues for applying the proposed method to other problems. Finally, Section 7 draws conclusions.

2 Chunking Korean

Figure 1 shows the structure of the chunking model for Korean. The main idea of this model is to apply rules to determine the chunk type of a word w_i in a sentence, and then to refer to a memory based classifier in order to check whether it is an exceptional case of the rules. In the training phase, each sentence is analyzed by the rules and the predicted chunk type is compared with the true chunk type. In case of misprediction, the error type is determined according to the true chunk type and the predicted chunk type. The mispredicted chunks are stored in the error case library with their true chunk types. Since the error case library accumulates only the exceptions of the rules, the number of cases in the library is small if the rules are general enough to represent the instance space well.

The classification phase in Figure 1 is expressed as a procedure in Figure 2. It determines the chunk type of a word w_i given with the context C_i. First of all, the rules are applied to determine the chunk type. Then, it is checked whether C_i is an exceptional case of the rules. If it is, the chunk type determined by the rules is discarded and is determined again by the memory based reasoning. The condition to make a decision of exceptional case is whether the similarity between C_i and the nearest instance in the error

Procedure Combine
Input : a word w_i, a context C_i, and the threshold t
Output : a chunk type c

[Step 1]	c = Determine the chunk type of w_i using rules.
[Step 2]	e = Get the nearest instance of C_i in error case library.
[Step 3]	**If** Similarity$(C_i, e) \geq t$, **then** c = Determine chunk type of w_i by memory-based learning.

Figure 2: The procedure for combining the rules and memory based learning.

case library is larger than the threshold t. Since the library contains only the exceptional cases, the more similar is C_i to the nearest instance, the more probable is it an exception of the rules.

3 Chunking by Rules

There are four basic phrases in Korean: noun phrase (NP), verb phrase (VP), adverb phrase (ADVP), and independent phrase (IP). Thus, chunking by rules is divided into largely four components.

3.1 Noun Phrase Chunking

When the part-of-speech of w_i is one of determiner, noun, and pronoun, there are only seven rules to determine the chunk type of w_i due to the well-developed postpositions of Korean.

1. **If** $POS(w_{i-1})$ = determiner **and** w_{i-1} does not have a postposition **Then** y_i = `I-NP`.
2. **Else If** $POS(w_{i-1})$ = pronoun **and** w_{i-1} does not have a postposition **Then** y_i = `I-NP`.
3. **Else If** $POS(w_{i-1})$ = noun **and** w_{i-1} does not have a postposition **Then** y_i = `I-NP`.
4. **Else If** $POS(w_{i-1})$ = noun **and** w_{i-1} has a *possessive postposition* **Then** y_i = `I-NP`.
5. **Else If** $POS(w_{i-1})$ = noun **and** w_{i-1} has a *relative postfix* **Then** y_i = `I-NP`.
6. **Else If** $POS(w_{i-1})$ = adjective **and** w_{i-1} has a *relative ending* **Then** y_i = `I-NP`.
7. **Else** y_i = `B-NP`.

Here, $POS(w_{i-1})$ is the part-of-speech of w_{i-1}. `B-NP` represents the first word of a noun phrase, while `I-NP` is given to other words in the noun phrase.

Since determiners, nouns and pronouns play the similar syntactic role in Korean, they form a noun phrase when they appear in succession without postposition (Rule 1–3). The words with postpositions become the end of a noun phrase, but there are only two exceptions. When the type of a postposition is *possessive*, it is still in the mid of noun phrase (Rule 4). The other exception is a *relative postfix* '적 (*jeok*)' (Rule 5). Rule 6 states that a simple relative clause with no sub-constituent also constitutes a noun phrase. Since the adjectives of Korean have no definitive usage, this rule corresponds to the definitive usage of the adjectives in English.

3.2 Verb Phrase Chunking

The verb phrase chunking has been studied for a long time under the name of *compound verb processing* in Korean and shows relatively high accuracy. Shin used a finite state automaton for verb phrase chunking (Shin, 1999), while K.-C. Kim used knowledge-based rules (Kim et. al, 1995). For the consistency with noun phrase chunking, we use the rules in this paper. The rules used are the ones proposed by (Kim et. al, 1995) and the further explanation on the rules is skipped. The number of the rules used is 29.

3.3 Adverb Phrase Chunking

When the adverbs appear in succession, they have a great tendency to form an adverb phrase. Though an adverb sequence is not always one adverb phrase, it usually forms one phrase. Table 1 shows this empirically. The usage of the successive adverbs is investigated from STEP 2000 dataset[1] where 270 cases are observed. The 189 cases among them form a phrase whereas the remaining 81 cases form two phrases independently. Thus, it can be said that the possibility that an adverb sequence forms a phrase is far higher than the possibility that it forms two phrases.

When the part-of-speech of w_i is an adjective, its chunk type is determined by the following rule.

1. **If** $POS(w_{i-1})$ = adverb **Then** y_i = `I-ADVP`.
2. **Else** y_i = `B-ADVP`.

[1] This dataset will be explained in Section 5.1.

	No. of Cases	Probability
One Phrase	189	0.70
Two Phrases	81	0.30

Table 1: The probability that an adverb sequence forms a chunk.

3.4 Independent Phrase Chunking

There is no special rule for independent phrase chunking. It can be done only through knowledge base that stores the cases where independent phrases take place. We designed 12 rules for independent phrases.

4 Chunking by Memory-Based Learning

Memory-based learning is a direct descent of the k-Nearest Neighbor (k-NN) algorithm (Cover and Hart, 1967). Since many natural language processing (NLP) problems have constraints of a large number of examples and many attributes with different relevance, memory-based learning uses more complex data structure and different speedup optimization from the k-NN.

It can be viewed with two components: a *learning component* and a similarity-based *performance component*. The learning component involves adding training examples to memory, where all examples are assumed to be fixed-length vectors of n attributes. The similarity between an instance **x** and all examples **y** in memory is computed using a *distance metric*, $\Delta(\mathbf{x}, \mathbf{y})$. The chunk type of **x** is then determined by assigning the most frequent category within the k most similar examples of **x**.

The distance from **x** and **y**, $\Delta(\mathbf{x}, \mathbf{y})$ is defined to be

$$\Delta(\mathbf{x}, \mathbf{y}) \equiv \sum_{i=1}^{n} \alpha_i \delta(x_i, y_i),$$

where α_i is the weight of i-th attribute and

$$\delta(x_i, y_i) = \begin{cases} 0 & \text{if } x_i = y_i, \\ 1 & \text{if } x_i \neq y_i. \end{cases}$$

When α_i is determined by *information gain* (Quinlan, 1993), the k-NN algorithm with this metric is called *IB1-IG* (Daelemans et. al, 2001). All the experiments performed by memory-based learning in this paper are done with IB1-IG.

Table 2 shows the attributes of IB1-IG for chunking Korean. To determine the chunk type of a word w_i, the lexicons, POS tags, and chunk types of surrounding words are used. For the surrounding words, three words of left context and three words of right context are used for lexicons and POS tags, while two words of left context are used for chunk types. Since chunking is performed sequentially, the chunk types of the words in right context are not known in determining the chunk type of w_i.

5 Experiments

5.1 Dataset

For the evaluation of the proposed method, all experiments are performed on *STEP 2000 Korean Chunking dataset* (STEP 2000 dataset)[2]. This dataset is derived from the parsed corpus, which is a product of STEP 2000 project supported by Korean government. The corpus consists of 12,092 sentences with 111,658 phrases and 321,328 words, and the vocabulary size is 16,808. Table 3 summarizes the information on the dataset.

The format of the dataset follows that of CoNLL-2000 dataset (CoNLL, 2000). Figure 3 shows an example sentence in the dataset[3]. Each word in the dataset has two additional tags, which are a part-of-speech tag and a chunk tag. The part-of-speech tags are based on KAIST tagset (Yoon and Choi, 1999). Each phrase can have two kinds of chunk types: B-XP and I-XP. In addition to them, there is O chunk type that is used for words which are not part of any chunk. Since there are four types of phrases and one additional chunk type O, there exist nine chunk types.

5.2 Performance of Chunking by Rules

Table 4 shows the chunking performance when only the rules are applied. Using only the rules gives 97.99% of accuracy and 91.87 of F-score. In spite of relatively high accuracy, F-score is somewhat low. Because the important unit of the work in the applications of text chunking is a phrase, F-score is far more important than accuracy. Thus, we have much room to improve in F-score.

[2]The STEP 2000 Korean Chunking dataset is available in http://bi.snu.ac.kr/~sbpark/Step2000.

[3]The last column of this figure, the English annotation, does

Attribute	Explanation	Attribute	Explanation
W_{i-3}	word of w_{i-3}	POS_{i-3}	POS of w_{i-3}
W_{i-2}	word of w_{i-2}	POS_{i-2}	POS of w_{i-2}
W_{i-1}	word of w_{i-1}	POS_{i-1}	POS of w_{i-1}
W_i	word of w_i	POS_i	POS of w_i
W_{i+1}	word of w_{i+1}	POS_{i+1}	POS of w_{i+1}
W_{i+2}	word of w_{i+2}	POS_{i+2}	POS of w_{i+2}
W_{i+3}	word of w_{i+3}	POS_{i+3}	POS of w_{i+3}
C_{i-3}	chunk of w_{i-3}	C_{i-2}	chunk of w_{i-2}
C_{i-1}	chunk of w_{i-1}		

Table 2: The attributes of IB1-IG for chunking Korean.

Information	**Value**
Vocabulary Size	16,838
Number of total words	321,328
Number of chunk types	9
Number of POS tags	52
Number of sentences	12,092
Number of phrases	112,658

Table 3: The simple statistics on STEP 2000 Korean Chunking dataset.

한국	nq	B-NP	Korea
의	jcm	I-NP	*Postposition : POSS*
세종	nq	I-NP	Sejong
기지	ncn	I-NP	base
와	jcj	I-NP	and
그	mmd	I-NP	the
주변	ncn	I-NP	surrounding
기지	ncn	I-NP	base
는	jxt	I-NP	*Postposition: TOPIC*
서남극	ncn	B-NP	western South Pole
남	ncn	B-NP	south
쉐틀란드	nq	I-NP	Shetland
의	jcm	I-NP	*Postposition : POSS*
킹조지섬	nq	I-NP	King George Island
에	jca	I-NP	*Postposition : LOCA*
있	paa	B-VP	is located
다	ef	I-VP	*Ending : DECL*
.	sf	O	

Figure 3: An example of STEP 2000 dataset.

Type	Precision	Recall	F-score
ADVP	98.67%	97.23%	97.94
IP	100.00%	99.63%	99.81
NP	88.96%	88.93%	88.94
VP	92.89%	96.35%	94.59
All	91.28%	92.47%	91.87

Table 4: The experimental results when the rules are only used.

Error Type	No. of Errors	Ratio (%)
B-ADVP I-ADVP	89	1.38
B-ADVP I-NP	9	0.14
B-IP B-NP	9	0.14
I-IP I-NP	2	0.03
B-NP I-NP	2,376	36.76
I-NP B-NP	2,376	36.76
B-VP I-VP	3	0.05
I-VP B-VP	1,599	24.74
All	6,463	100.00

Table 5: The error distribution according to the mislabeled chunk type.

Table 5 shows the error types by the rules and their distribution. For example, the error type 'B-ADVP I-ADVP' contains the errors whose true label is B-ADVP and that are mislabeled by I-ADVP. There are eight error types, but most errors are related with noun phrases. We found two reasons for this:

1. It is difficult to find the beginning of noun phrases. All nouns appearing successively without postpositions are not a single noun phrase. But, they are always predicted to be single noun phrase by the rules, though they can be more than one noun phrase.

2. The postposition representing a noun coordination, '와 (*wa*)' is very ambiguous. When '와 (*wa*)' is representing the coordination, the chunk types of it and its next word should be "I-NP I-NP". But, when it is just an adverbial postposition that implies '*with*' in English, the chunk types should be "I-NP B-NP".

	Decision Tree	SVM	MBL
Accuracy	97.95±0.24%	98.15±0.20%	97.79±0.29%
Precision	92.29±0.94%	93.63±0.81%	91.41±1.24%
Recall	90.45±0.80%	91.48±0.70%	91.43±0.87%
F-score	91.36±0.85	92.54±0.72	91.38±1.01

Table 6: The experimental results of various machine learning algorithms.

5.3 Performance of Machine Learning Algorithms

Table 6 gives the 10-fold cross validation result of three machine learning algorithms. In each fold, the corpus is divided into three parts: *training* (80%), *held-out* (10%), *test* (10%). Since *held-out* set is used only to find the best value for the threshold t in the combined model, it is not used in measuring the performance of machine learning algorithms.

The machine learning algorithms tested are (i) memory-based learning (MBL), (ii) decision tree, and (iii) support vector machines (SVM). We use C4.5 release 8 (Quinlan, 1993) for decision tree induction and SVM^{light} (Joachims, 1998) for support vector machines, while *TiMBL* (Daelemans et. al, 2001) is adopted for memory-based learning. Decision trees and SVMs use the same attributes with memory-based learning (see Table 2). Two of the algorithms, memory-based learning and decision tree, show worse performance than the rules. The F-scores of memory-based learning and decision tree are 91.38 and 91.36 respectively, while that of the rules is 91.87 (see Table 4). On the other hand, support vector machines present a slightly better performance than the rules. The F-score of support vector machine is 92.54, so the improvement over the rules is just 0.67.

Table 7 shows the weight of attributes when only memory-based learning is used. Each value in this table corresponds to α_i in calculating $\Delta(\mathbf{x}, \mathbf{y})$. The more important is an attribute, the larger is the weight of it. Thus, the most important attribute among 17 attributes is C_{i-1}, the chunk type of the previous word. On the other hand, the least important attributes are W_{i-3} and C_{i-3}. Because the words make less influence on determining the chunk type of w_i in question as they become more distant from w_i. That is, the order of important lexical attributes is $\langle W_i, W_{i-1}, W_{i+1}, W_{i-2}, W_{i+2}, W_{i+3}, W_{i-3}\rangle$. The same phenomenon is found in part-of-speech (POS) and chunk type (C). In comparing the part-of-speech information with the lexical information, we find out that the part-of-speech is more important. One possible explanation for this is that the lexical information is too sparse.

not exist in the dataset. It is given for the explanation.

Attribute	Weight	Attribute	Weight
W_{i-3}	0.03	POS_{i-3}	0.04
W_{i-2}	0.07	POS_{i-2}	0.11
W_{i-1}	0.17	POS_{i-1}	0.28
W_i	0.22	POS_i	0.38
W_{i+1}	0.14	POS_{i+1}	0.22
W_{i+2}	0.06	POS_{i+2}	0.09
W_{i+3}	0.04	POS_{i+3}	0.05
C_{i-3}	0.03	C_{i-2}	0.11
C_{i-1}	0.43		

Table 7: The weights of the attributes in IB1-IG. The total sum of the weights is 2.48.

fold	Precision (%)	Recall (%)	F-score	t
1	94.87	94.12	94.49	1.96
2	93.52	93.85	93.68	1.98
3	95.25	94.72	94.98	1.95
4	95.30	94.32	94.81	1.95
5	92.91	93.54	93.22	1.87
6	94.49	94.50	94.50	1.92
7	95.88	94.35	95.11	1.94
8	94.25	94.18	94.21	1.94
9	92.96	91.97	92.46	1.91
10	95.24	94.02	94.63	1.97
Avg.	94.47±1.04	93.96±0.77	94.21±0.84	1.94

Table 8: The final result of the proposed method by combining the rules and the memory-based learning. The average accuracy is 98.21±0.43.

The best performance on English reported is 94.13 in F-score (Zhang et. al, 2001). The reason why the performance on Korean is lower than that on English is the *curse of dimensionality*. That is, the wider context is required to compensate for the free order of Korean, but it hurts the performance (Cherkassky and Mulier, 1998).

5.4 Performance of the Hybrid Method

Table 8 shows the final result of the proposed method. The F-score is 94.21 on the average which is improvement of 2.34 over the rules only, 1.67 over support vector machines, and 2.83 over memory-based learning. In addition, this result is as high as the performance on English (Zhang et. al, 2001).

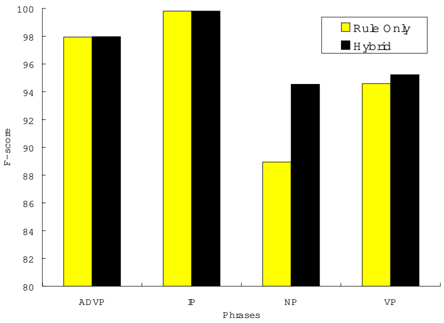

Figure 4: The improvement for each kind of phrases by combining the rules and MBL.

The threshold t is set to the value which produces the best performance on the *held-out* set. The total sum of all weights in Table 7 is 2.48. This implies that when we set $t > 2.48$, only the rules are applied since there is no exception with this threshold. When $t = 0.00$, only the memory-based learning is used. Since the memory-based learning determines the chunk type of w_i based on the exceptional cases of the rules in this case. the performance is poor with $t = 0.00$. The best performance is obtained when t is near 1.94.

Figure 4 shows how much F-score is improved for each kind of phrases. The average F-score of noun phrase is 94.54 which is far improved over that of the rules only. This implies that the exceptional cases of the rules for noun phrase are well handled by the memory-based learning. The performance is much improved for noun phrase and verb phrase, while it remains same for adverb phrases and independent phrases. This result can be attributed to the fact that there are too small number of exceptions for adverb phrases and independent phrases. Because the accuracy of the rules for these phrases is already high enough, most cases are covered by the rules. Memory based learning treats only the exceptions of the rules, so the improvement by the proposed method is low for the phrases.

6 Discussion

In order to make the proposed method practical and applicable to other NLP problems, the following issues are to be discussed:

1. **Why are the rules applied before the memory-based learning?**
   When the rules are efficient and accurate enough to begin with, it is reasonable to apply the rules first (Golding and Rosenbloom, 1996). But, if they were deficient in some way, we should have applied the memory-based learning first.

2. **Why don't we use all data for the machine learning method?**
   In the proposed method, memory-based learning is used not to find a hypothesis for interpreting whole data space but to handle the exceptions of the rules. If we use all data for both the rules and memory-based learning, we have to weight the methods to combine them. But, it is difficult to know the weights of the methods.

3. **Why don't we convert the memory-based learning to the rules?**
   Converting between the rules and the cases in the memory-based learning tends to yield inefficient or unreliable representation of rules.

The proposed method can be directly applied to the problems other than chunking Korean if the proper rules are prepared. The proposed method will show better performance than the rules or machine learning methods alone.

7 Conclusion

In this paper we have proposed a new method to learn chunking Korean by combining the handcrafted rules and a memory-based learning. Our method is based on the rules, and the estimates on chunks by the rules are verified by a memory-based learning. Since the memory-based learning is an efficient method to handle exceptional cases of the rules, it supports the rules by making decisions only for the exceptions of the rules. That is, the memory-based learning enhances the rules by efficiently handling the exceptional cases of the rules.

The experiments on STEP 2000 dataset showed that the proposed method improves the F-score of the rules by 2.34 and of the memory-based learning by 2.83. Even compared with support vector machines, the best machine learning algorithm in text chunking, it achieved the improvement of 1.67.

The improvement was made mainly in noun phrases among four kinds of phrases in Korean. This is because the errors of the rules are mostly related with noun phrases. With relatively many instances for noun phrases, the memory-based learning could compensate for the errors of the rules. We also empirically found the threshold value t used to determine when to apply the rules and when to apply memory-based learning.

We also discussed some issues in combining a rule-based method and a memory-based learning. These issues will help to understand how the method works and to apply the proposed method to other problems in natural language processing. Since the method is general enough, it can be applied to other problems such as POS tagging and PP attachment. The memory-based learning showed good performance in these problems, but did not reach the state-of-the-art. We expect that the performance will be improved by the proposed method.

Acknowledgement

This research was supported by the Korean Ministry of Education under the BK21-IT program and by the Korean Ministry of Science and Technology under NRL and BrainTech programs.

References

V. Cherkassky and F. Mulier. 1998. *Learning from Data: Concepts, Theory, and Methods*, John Wiley & Sons, Inc.

CoNLL. 2000. *Shared Task for Computational Natural Language Learning* (CoNLL), http://lcg-www.uia.ac.be/conll2000/chunking.

T. Cover and P. Hart. 1967. Nearest Neighbor Pattern Classification, *IEEE Transactions on Information Theory*, Vol. 13, pp. 21–27.

W. Daelemans, A. Bosch and J. Zavrel. 1999. Forgetting Exceptions is Harmful in Language Learning, *Machine Learning*, Vol. 34, No. 1, pp. 11–41.

W. Daelemans, J. Zavrel, K. Sloot and A. Bosch. 2001. TiMBL: Tilburg Memory Based Learner, version 4.1, Reference Guide, ILK 01-04, Tilburg University.

A. Golding and P. Rosenbloom. 1996. Improving Accuracy by Combining Rule-based and Case-based Reasoning, *Artificial Intelligence*, Vol. 87, pp. 215–254.

T. Joachims. 1998. *Making Large-Scale SVM Learning Practical*, LS8, Universitaet Dortmund.

K.-C. Kim, K.-O. Lee, and Y.-S. Lee. 1995. Korean Compound Verbals Processing driven by Morphological Analysis, *Journal of KISS*, Vol. 22, No. 9, pp. 1384–1393.

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

S.-B. Park and B.-T. Zhang. 2001. Combining a Rule-based Method and a *k*-NN for Chunking Korean Text, In *Proceedings of the 19th International Conference on Computer Processing of Oriental Languages*, pp. 225–230.

R. Quinlan. 1993. *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers.

L. Ramshaw and M. Marcus. 1995. Text Chunking Using Transformation-Based Learning, In *Proceedings of the Third ACL Workshop on Very Large Corpora*, pp. 82–94.

H.-P. Shin. 1999. Maximally Efficient Syntatic Parsing with Minimal Resources, In *Proceedings of the Conference on Hangul and Korean Language Infomration Processing*, pp. 242–244.

J.-T. Yoon and K.-S. Choi. 1999. *Study on KAIST Corpus*, CS-TR-99-139, KAIST CS.

T. Zhang, F. Damerau and D. Johnson. 2001. Text Chunking Using Regularized Winnow, In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pp. 539–546.