Korean Compound Noun Decomposition Using Syllabic Information Only

Seong-Bae Park, Jeong-Ho Chang, and Byoung-Tak Zhang

School of Computer Science and Engineering
Seoul National University
151-744 Seoul, Korea
{sbpark,jhchang,btzhang}@bi.snu.ac.kr

Abstract. The compound nouns are freely composed in Korean, since it is possible to concatenate independent nouns without a postposition. Therefore, the systems that handle compound nouns such as machine translation and information retrieval have to decompose them into single nouns for the further correct analysis of texts. This paper proposes the GECORAM (GEneralized COmbination of Rule-based learning And Memory-based learning) algorithm for Korean compound noun decomposition using only syllabic information. The merit of rule-based learning algorithms is high comprehensibility, but they shows low performance in many application tasks. To tackle this problem, GECORAM combines the rule-based learning and memory-based learning. According to the experimental results, GECORAM shows higher accuracy than rule-based learning or memory-based learning alone.

1 Introduction

The nouns that appear successively without a postposition can be concatenated to compose a compound noun in Korean. Such compound nouns have more contextual information compared to single nouns [15], and play important role in natural language processing. The critical issue in handling compound nouns is that the number of possible compound nouns is infinite. Because all compound nouns can not be listed in the dictionary, it is required to decompose given a compound noun into single nouns.

When a compound noun is composed of n syllables, there are theoretically 2^{n-1} kinds of decompositions. Thus, the easiest way to decompose a compound noun is to take the most plausible one among 2^{n-1} decompositions. Many previous studies have been proposed based on this idea. Shim used composite mutual information trained from about a corpus of 1.1 million word size [15]. Lee et al. considered this task as part-of-speech tagging, and applied a Markov model [11].

The main drawback of such statistics-based methods is that it is difficult for human to understand the trained results. On the other hand, the rules, whether they are made manually or automatically, have high comprehensibility. Thus, there have been a number of studies that apply rules to compound noun decomposition. For instance, Kang designed four decomposition rules and two

A. Gelbukh (Ed.): CICLing 2004, LNCS 2945, pp. 146–157, 2004.

exception rules [10]. Yoon used both statistics and heuristic rules [16], where the heuristic rules take charge of the nouns with unknown single nouns.

The rule-based methods need a human expert who writes the accurate rules. Since the performance of the rule-based methods depends on the quality of the rules, the rule writer must have profound knowledge about a target task. However, it is very expensive to work with such an expert. Thus, in machine learning community, a number of methods have been proposed that learn the rules from data represented as vectors of feature-value pairs. Clark and Niblett proposed the CN2 program that uses the general-to-specific beam search [3], and Fürnkranz and Widmar proposed the IREP algorithm [9]. Cohen improved the IREP to produce the RIPPER algorithm [4], while Cohen and Singer presented the SILPPER algorithm [5] which adopted a boosting into rule leanring.

The problem of automatically learned rules is low performance compared with other supervised learning algorithms. In our previous work, it is shown that a combination of rules and memory-based learning achieves high accuracy [12]. To apply this idea to the tasks without the previously designed rules, we propose in this paper the *GECORAM* (GEneralized COmbination of Rule-based learning And Memory-based learning) *algorithm* that effectively combines rule-based learning and memory-based learning. Because the rules are basic approach to ILP (Inductive Logic Programming), the improvement of the rules is important not only in natural language processing but also in machine learning.

The rest of this paper is organized as follows. Section 2 surveys the previous rule-based learning algorithms. Section 3 describes the GECORAM algorithm, and Section 4 presents the experimental results. Finally, section 5 draws conclusions.

2 Previous Work on Rule-Based Learning Algorithms

2.1 The IREP Algorithm

Since the GECORAM algorithm is primarily based on the IREP (Incremental Reduced Error Pruning) algorithm, the IREP should be first explained for understanding of the GECORAM algorithm. The algorithm of the IREP is summarized in Figure 1 and consists of two greedy algorithms. The first greedy algorithm constructs a rule at a time, and then removes from the training set all examples covered by a new rule. The principle used in constructing a rule is that more positive examples and less negative examples should be covered by the rule. For this purpose, it partitions given a training set `data` into two subsets: `grow` and `prune`. In general, `grow` is two-thirds of `data`, and `prune` is one-third. `grow` is used to first construct a rule, and `prune` is used to simplify it. The step to grow a rule (function `GrowRule` in Figure 1) repeatedly adds conditions to rule r_0 with an empty antecedent. In each i-th stage, a more specialized rule r_{i+1} is made by adding single condition to r_i. The added condition in constructing r_{i+1} is the one with the largest *information gain* [13] relative to r_i, where the

```
function IREP(data)
begin
  RuleSet := φ
  while ∃ positive examples ∈ data do
    Split data into grow and prune.
    rule := GrowRule(grow)
    rule := PruneRule(prune)
    Add rule to RuleSet.
    Remove examples covered by rule from data.
    if Accuracy(rule) ≤ P/(P+N) then
      return RuleSet
    endif
  endwhile
  return RuleSet
end
```

Fig. 1. The IREP algorithm. P is the number of positive examples in `data` and N is that of negative examples

information gain is defined as

$$Gain\,(r_{i+1}, r_i) = T_{i+1}^{+} \cdot \left(-\log \frac{T_i^{+}}{T_i^{+} + T_i^{-}} + \log \frac{T_{i+1}^{+}}{T_{i+1}^{+} + T_{i+1}^{-}} \right).$$

Here, T_i^{+} and T_i^{-} are the number of positive and negative examples covered by r_i accordingly. The conditions are added until the information gain becomes 0.

In the second step, the rule constructed in the function `GrowRule` is simplified by removing the conditions one by one. In the function `PruneRule`, the condition that maximizes the function $f(r_{i+1}) = \frac{T_{i+1}^{+} - T_{i+1}^{-}}{T_{i+1}^{+} + T_{i+1}^{-}}$ is removed. After simplifying the rule, the pruned rule is added to `RuleSet`, and all examples covered by it are removed from `data`.

The information gain used in constructing a rule is larger than or equal to 0. Thus, all the generated rules always cover some positive examples in `data` and it guarantees that the algorithm will eventually terminate. However, it is possible that there could be a number of rules that cover only a few positive examples, which causes too much computation for noisy data. To keep these rules from being added to `RuleSet`, the learning process stops if the accuracy of the generated rule is less than $P/(P+N)$, where P is the number of positive examples in `data`, and N is that of negative examples.

2.2 The RIPPER Algorithm

The RIPPER(Repeated Incremental Pruning to Produce Error Reduction) algorithm is an improved model of the IREP algorithm, and Figure 2 gives its pseudo-code. After adding rules by running `IREP`, the RIPPER optimizes the rule set `RuleSet` to reduce the number of the rules within `RuleSet` and improve

```
function Optimize(RuleSet, data)
begin
   for each rule c ∈ RuleSet do
      Split data into grow and prune.
      c' := GrowRule(grow)
      c' := PruneRule(c', data) with RuleSet−c + c'
      c'' := GrowRule(c, grow)
      c'' := PruneRule(c'', prune) with RuleSet−c + c''
      Replace c in RuleSet with best of c, c', and c''.
   endfor
   return RuleSet
end

function RIPPER(data)
begin
   RuleSet := IREP(data)
   repeat twice:
      RuleSet := Optimize(RuleSet, data)
      UncovData := examples ∈ data not covered by the rules in RuleSet
      RuleSet := RuleSet + IREP(UncovData)
   endrepeat
end
```

Fig. 2. The RIPPER algorithm.

performance. In the function `Optimize`, each rule is tested one by one in the order in which they were added. For each rule $c \in$ `RuleSet`, two alternative rules are constructed: c' and c''. c' is made by growing and pruning, where pruning is guided so as to minimize error not of `prune` but of the entire rule set with c' replacing c. c'' is constructed in a similar way except that it is grown from c not empty antecedent. Finally, the best of c, c' and c'' is added to `RuleSet`. Then, which of them is best? To determine this, RIPPER uses the MDL (Minimum Description Length) principle. According to MDL, the best rule is the one with the shortest DL (description length). To measure the DL of a rule, the RIPPER adopts the method proposed by Quinlan [14]. After optimization, because it is highly possible for `RuleSet` to cover less number of positive examples than the original `RuleSet`, new rules are added to `RuleSet` by running the IREP over `UncovData` that contains the examples uncovered by `RuleSet`. This process is performed twice, because the empirical performance is increased when it is repeated twice.

3 GECORAM Algorithm

3.1 Basic Idea

Korean compound noun decomposition can be regarded as a classification task in the viewpoint of machine learning. That is, it is a binary classification task

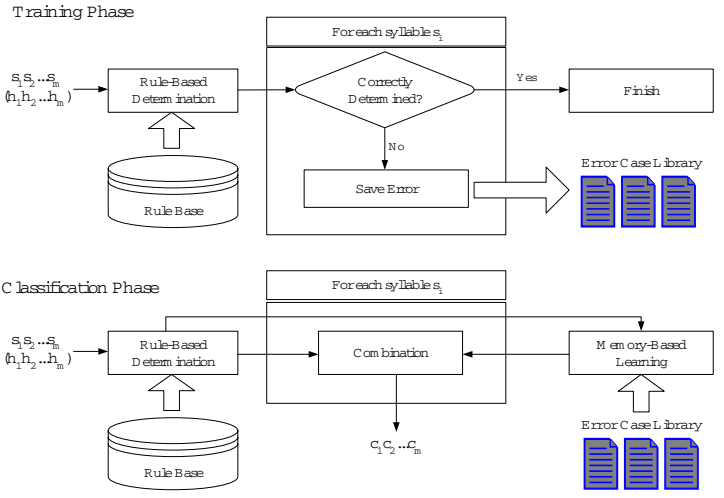

Fig. 3. The basic idea of the GECORAM algorithm.

to determine whether a space is put after a syllable s_i of a compound noun $w = s_1, \ldots, s_m$. The context information h_i of s_i is usually used to determine it. In this paper, n ($n \leq m/2$) syllables in left and right positions are used as context information.

The GECORAM algorithm combines rule-based learning and memory-based learning efficiently to solve classification tasks. The basic idea of this algorithm is expressed in Figure 3. In the training phase, each compound noun is analyzed by the rules trained by a rule-learning algorithm and the classification results are compared with the true labels[1]. In case of misclassification, the errors are stored in the *error case library* with their true labels. Since the error case library accumulates only the exceptions of the rules, the number of examples in it is small if the rules are general and accurate enough to represent the instance space well.

The classification phase determines the decomposition of s_i given with the context h_i. First, the rules are applied to decide whether a space is put after s_i. Then, it is checked if the current context h_i of s_i is an exception of the rules. If it is, the decomposition determined by the rules is discarded and then is determined again by the memory-based classifier trained with the error case library. This is because the rules tend to make errors if h_i is an exception of the

[1] When a space should be put after s_i, the label is 'True'. Otherwise, it is 'False'.

```
function Support(RuleSet, data)
begin
   Err := φ
   for each (⟨s_i, h_i⟩, c_i) ∈ data do
      if RuleSet(⟨s_i, h_i⟩) ≠ c_i then
         Add (⟨s_i, h_i⟩, c_i) into Err.
      endif
   endfor
   MBL := Memory-Based-Learning(Err)
   return MBL
end

function Training-GECORAM(data)
begin
   RuleSet := MODIFIED-IREP(data)
   MBL := Support(RuleSet, data)
   θ := Get-Threshold-MBL(RuleSet, MBL, HeldOutData)
   return RuleSet + MBL + θ
end
```

Fig. 4. The training algorithm of the GECORAM. c_i is the true label for $\langle s_i, h_i \rangle$.

rules. Thus, memory-based learning is a component that handles the errors of the rules.

3.2 GECORAM Algorithm

Figure 4 shows the training phrase of the GECORAM. The first step of GECORAM is to train the rules from a training set `data`. For this purpose, GECORAM uses `MODIFIED-IREP`, a modified version of the IREP. The only difference between `MODIFIED-IREP` and the IREP is that `MODIFIED-IREP` does not have a `PruneRule` function. That is, in `MODIFIED-IREP`, the rules only grow and are never simplified. The role of the function `PruneRule` is played by the memory-based learning explained later. In the next step, the examples that are uncovered by `MODIFIED-IREP` are gathered, and the memory-based learner is trained with them.

Memory-based learning is a direct descent of the k-Nearest Neighbor (k-NN) algorithm [6]. Since many natural language processing tasks 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 k-NN.

The learning process in memory-based learning is simply to store the examples into memory, where all examples are assumed to be fixed-length vectors of n attributes. The similarity between an instance $\mathbf{x}$ and all examples $\mathbf{y}$ in memory is computed using a *distance metric*, $D(\mathbf{x}, \mathbf{y})$. The class of $\mathbf{x}$ is then determined by assigning the most frequent category within the k most similar examples of $\mathbf{x}$.

```
function Classify-GECORAM(x, θ, RuleSet, MBL)
begin
   c := RuleSet(x)
   y := the nearest instance of x in Err.
   if D(x, y) ≥ θ then
      c := MBL(x)
   endif
   return c
end
```

Fig. 5. The classification algorithm of the GECORAM.

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

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

where α_i is the weight of i-th attribute and

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

When α_i is determined by *information gain* [13], the k-NN algorithm with this metric is called *IB1-IG* [7]. All the experiments performed by memory-based learning in this paper are done with IB1-IG.

Since both rules and memory-based learning are used in the GECORAM, it is important to determine when to use rules and when to use memory-based classifier. To determine this, GECORAM has a threshold θ. The optimal value for θ is found by the following procedure. Assume that we have an independent held-out data set `HeldOutData`. Various value for θ is applied to the classification algorithm of the GECORAM described in Figure 5. The optimal value for θ is the one that outputs the best performance over `HeldOutData`.

In `Classify-GECORAM`, if $\mathbf{x}$ and $\mathbf{y}$ are simlar, $\mathbf{x}$ is considered to be an exception of the rules. Since the instances in memory are the ones with which the rules make an error, large $D(\mathbf{x}, \mathbf{y})$ implies that $\mathbf{x}$ is highly possible to be an exception of the rules. Thus, if $D(\mathbf{x}, \mathbf{y})$ is larger than the predefined threshold θ, the rules should not be applied.

Since θ is a threshold value for $D(\mathbf{x}, \mathbf{y})$, $0 \leq \theta \leq \beta$ is always satisfied when $\beta \equiv \sum_{j=1}^{m} \alpha_j$. When $\theta = 0$, the rules are always ignored. In this case, the generalization is done by only memory-based classifier trained with the errors of the rules. Thus, it will show low performance due to data sparseness. In contrast, only the rules are applied when $\theta = \beta$. In this case, the performance of GECORAM is equivalent to that of the rules.

3.3 Comparison to Other Algorithms

The GECORAM algorithm is similar to the RIPPER in that both algorithms are based on the IREP. While the RIPPER is a pure memory-based learning

algorithm for ILP, the GECORAM is a kind of mixture model that combines rule-based learning and memory-based learning. Thus, RIPPER prepares an `Optimize` function to keep too specific rules from being added into a rule set, but the optimizing process is substituted by memory-based learning in the GECORAM. Since memory-based learning is a lazy learning, the GECORAM does not use even a `PruneRule` function in the IREP.

In addition, the GECORAM shares similarities with transformation-based learning (TBL) [2] and AdaBoost [8]. Both methods build classifiers by combining simple rules. However, AdaBoost rests on the firm theoretical foundations while TBL does not. In addition, AdaBoost outperforms TBL in many applications of NLP [1].

The drawbacks of TBL can be summarized by three problems. The first problem is that it is easy to overfit to the training data. In addition, it is sensitive to noise, since it always tries to minimize the number of errors made by the current transformation list by adding an additional transformation. Finally, there is no theoretical foundation which supports that the added transformation improves the performance in the applications.

AdaBoost is similar to TBL in that the learning process is derived by the errors of current classifier. But, it is based on more theoretical background. Assume each $\epsilon_t \leq 1/2$, and let $\gamma_t = 1/2 - \epsilon_t$. Then, it is proven that the error of the final hypothesis h_{fin} has the following upper bound:

$$\frac{1}{N}|\{i : h_{fin}(\mathbf{x}_i) \neq y_i\}| \leq \exp\left(-2\sum_{t=1}^{T}\gamma_t^2\right),$$

where N is the number of training examples and T is the number of weak learners within AdaBoost. It is different from the proposed method in that each weak learner behaves in the same way and more than one learner is used.

The convergence speed of AdaBoost rests on the performance of weak learner. Though the only convergence condition of AdaBoost is that the accuracy of the weak learner is larger than 1/2, the convergence speed depends on the performance of weak learners. The early convergence can be obtained if the weak learners in early stage are strong with the help of prior knowledge. Thus, the proposed method can be considered to be a variation of AdaBoost, where first $(T-1)$ weak learners is substituted by the rules and the last one is substituted by memory-based learning. And, the threshold θ corresponds to weights of the weak learners. Therefore, it is important to design the rules to be strong enough to explain the problem space well. In addition, since the proposed method consists of only two steps rather than T steps, the computational cost is much decreased compared to TBL and AdaBoost.

4 Dataset

4.1 Data Set

In this paper, two kinds of standard data sets are used. The first data set (shim) is designed by Shim and used in [15], and the other (yoon) is the one used in [16].

Table 1. Statistics on data sets for Korean compound noun decomposition.

Data Set	shim	yoon
Number of examples	9,863	15,096
Number of syllables used	562	557
Average length of compound noun	7.26	4.92

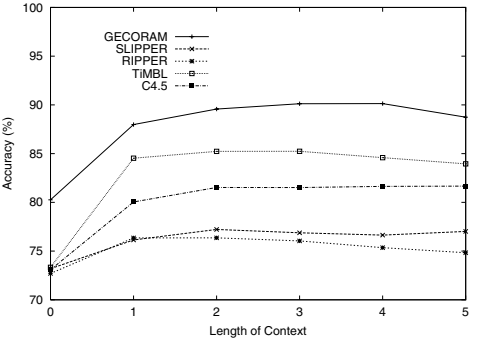

Fig. 6. The accuracy according to the context length on shim data set.

Table 1 summarizes the simple statistics on these two data sets. Each data set is divided into three parts: *training* (80%), *held-out* (10%), and *test* (10%). Since there are not enough examples in both data sets, the 10-fold cross validation is performed.

4.2 Experimental Results

In order to evaluate the performance of the GECORAM, we compare it with RIPPER [4], SLIPPER [5], C4.5 [13], and TiMBL [7]. RIPPER, SLIPPER, and C4.5 are rule-based learning algorithms, and TiMBL is a memory-based learning algorithm.

Figure 6 and Figure 7 shows the accuracy change according to the context length. The GECORAM shows the best performance for both data sets. TiMBL shows higher performance than rule-based learning algorithms such as RIPPER and SLIPPER. In general, the rule-based learning algorithms focus on the comprehensibility, and they have tendency to give lower performance than other supervised learning algorithms.

When the length of context is zero, the context information h_i is not used but only a syllable s_i is used to decompose a compound noun. Even in this case, all algorithms shows over 70% of accuracy. It implies that syllable is very important information for compound noun decomposition. The average syllable length in shim data set is 7.26. When three syllables in left and right context are considered, we will see seven syllables altogether. Thus, when the context length is three, the best performance is obtained. More context information plays role of noise in this data set. Since the average syllable length in yoon data set is 4.92, it

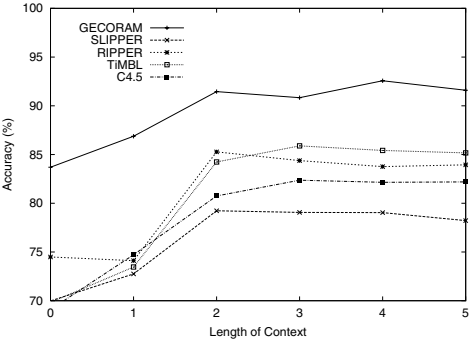

Fig. 7. The accuracy according to the context length on yoon data set.

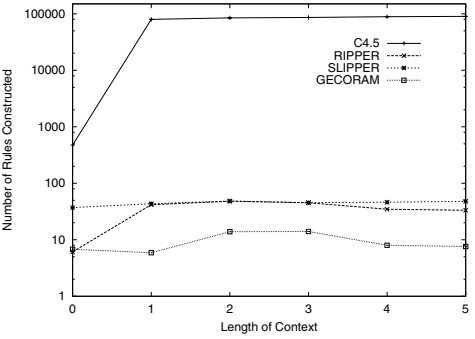

Fig. 8. The number of rules constructed according to the context length on shim data set.

is expected that the best performance is obtained when the context length is two. However, the best performance is actually obtained when it is four. Nevertheless, the difference between them is slight and statistically indistinguishable.

Figure 8 and Figure 9 illustrate the number of rules generated according to the context length. C4.5 constructs the most number of rules, while GECORAM generates the least number of rules. RIPPER and SLIPPER produce less than 100 rules. In the other hand, GECORAM makes only about 20 rules. This is because the errors of the rules are handled by memory-based learning in GECORAM and GECORAM does not have the optimizing process of RIPPER. In the result, GECORAM shows high performance with low computational cost.

Finally, Table 2 gives the accuracy when each algorithm shows the best performance. The GECORAM algorithm shows 90.13% of accuracy for shim data set and 92.57% for yoon data set. This is, on the average, higher than C4.5 by 9.3%, RIPPER by 10.5%, SLIPPER by 13.1%, and TiMBL by 5.8%. Therefore, GECORAM shows higher performance than rule-based learning or memory-based learning alone.

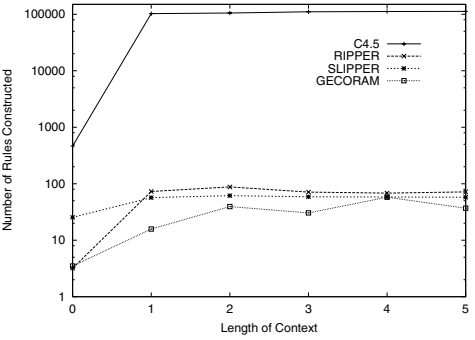

Fig. 9. The number of rules constructed according to the context length on yoon data set.

Table 2. The best accuracy of each algorithm.

Data Set	shim	yoon
C4.5	81.67%	82.37%
TiMBL	85.24%	85.89%
RIPPER	76.36%	85.27%
SLIPPER	77.22%	79.23%
GECORAM	**90.13%**	**92.57%**

5 Conclusions

In this paper we have proposed a new learning algorithm, GECORAM, that combines rule-based learning and memory-based learning. It first learns the rules, and then memory-based learning is performed with the errors of the trained rules. In classification, it is basically based on the rules, and its estimates are verified by a memory-based classifier. 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 trained rules by efficiently handling their exceptions.

We have applied the GECORAM algorithm to Korean compound noun decomposition. The experimental results on two standard data sets showed that it improves the accuracy of RIPPER by 10.5%, SLIPPER by 13.1%, and TiMBL by 5.8%, where RIPPER and SLIPPER are rule-based learning algorithms and TiMBL is a memory-based learning algorithm. Therefore, the proposed algorithm, GECORAM is more efficient than rule-based learning or memory-based learning alone.

In this paper, only syllabic information is used for decomposing compound nouns. Thus, the overall accuracy is lower than that of the previous studies [10, 11,15,16] that use external information such as dictionary and thesaurus. As a future work, we will study how to use such information with the GECORAM algorithm.

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

References

1. S. Abney, R. Schapire, and Y. Singer, "Boosting Applied to Tagging and PP Attachment," In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pp. 38–45, 1999.
2. E. Brill, "Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part of Speech Tagging," *Computational Linguistics*, Vol. 21, No. 4, pp. 543–566, 1995.
3. P. Clark and T. Niblett, "The CN2 Induction Algorithm," *Machine Learning*, Vol. 3, No. 1, pp. 261–284, 1989.
4. W. Cohen, "Fast Effective Rule Induction," In *Proceedings of the 12th International Conference on Mahcine Learning*, pp. 115–123, 1995.
5. W. Cohen and Y. Singer, "A Simple, Fast, and Effective Rule Learner," In *Proceedings of the 16th National Conference on Artificial Intelligence*, pp. 335–342, 1999.
6. T. Cover and P. Hart, "Nearest Neighbor Pattern Classification," *IEEE Transactions on Information Theory*, Vol. 13, pp. 21–27, 1967.
7. W. Daelemans, J. Zavrel, K. Sloot, and A. Bosch, *TiMBL: Tilburg Memory Based Learner, version 4.1, Reference Guide*, ILK 01-04, Tilburg University, 2001.
8. Y. Freund and R. Schapire, "Experiments with a New Boosting Algorithm," In *Proceedings of the 13th International Conference on Machine Learning*, pp. 148–156, 1996.
9. J. Fürnkfranz and G. Widmar, "Incremental Reduced Error Pruning," In *Proceedings of the 11th International Conference on Machine Learning*, pp. 70–77, 1994.
10. S.-S. Kang, "Korean Compound Noun Decomposition Algorithm," *Journal of KISS (B)*, Vol. 25, No. 1, pp. 172–182, 1998.
11. J.-W. Lee, B.-T. Zhang, and Y.-T. Kim, "Compound Noun Decomposition Using a Markov Model," In *Proceedings of MT Summit VII*, pp. 427–431, 1999.
12. S.-B. Park and B.-T. Zhang, "Text Chunking by Combining Hand-Crafted Rules and Memory-Based Learning," In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pp. 497–504, 2003.
13. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publisher, 1993.
14. R. Quinlan, "MDL and Categorical Theories (continued)," In *Proceedings of the 12th International Conference on Machine Learning*, pp. 464–470, 1995.
15. K.-S. Shim, "Segmentation of Compound Nouns using Composite Mutual Information," In *Proceedings of the 3rd Chinese-Korea Joint Symposium on Oriental Language Processing and Character Recognition*, pp. 106–113, 1999.
16. B.-H. Yoon, M.-J. Cho, and H.-C. Rim, "Segmenting Korean Compound Nouns Using Statistical Information and a Preference Rule," *KISS Journal (B)*, Vol. 24, No. 8, pp. 900–909, 1997.