A Boosted Maximum Entropy Model for Learning Text Chunking

Seong-Bae Park SBPARK@BI.SNU.AC.KR
Byoung-Tak Zhang BTZHANG@BI.SNU.AC.KR
School of Computer Science and Engineering, Seoul National University, Seoul 151-742, Korea

Abstract

We propose a hybrid machine learning method to overcome the limitations of maximum entropy models applied to text chunking (feature selection and high computational complexity), and the problem of imbalanced data in natural language resources. This method consists of three main components: (i) generating high-order features for maximum entropy models from the decision tree learned with simple first-order features, (ii) active learning in training a maximum entropy model to reduce the computational cost, and (iii) AdaBoost to boost up the maximum entropy models and adapt them to the problems with highly imbalanced data. The performance of the proposed method is empirically verified on CoNLL-2000 dataset.

1. Introduction

Since Ramshaw and Marcus applied machine learning to text chunking (Ramshaw & Marcus, 1995), many researchers in natural language learning have used statistics-based or machine learning methods to capture the best hypothesis on text chunking (Argamon et al., 1998; Zhang et al., 2001). Maximum entropy models are one of the methods successfully applied to the problem (Koeling, 2000).

While the maximum entropy models have a long history, it is only recently that they are successfully applied to various kinds of problems in natural language processing (Ratnaparkhi, 1998). Koeling showed that the maximum entropy model using very local lexical information also gives encouraging results (Koeling, 2000). However, the maximum entropy models applied to natural language processing have two major problems. The first problem is that they require much *prior* knowledge on the target problem that is required to construct *features* of the maximum entropy models. The other is that they require too much computation. Since millions of training examples are managed in most language learning problems, it is often infeasible to train the maximum entropy models using a straightforward technique.

In this paper, we propose a method that uses the maximum entropy model for text chunking without much prior linguistic knowledge. To overcome the two hurdles stated above,

- We automatically construct the features of maximum entropy models from decision trees. A set of if-then rules generated from a decision tree trained with simple first-order features can be considered as the high-order features with identical weights. Thus, using maximum entropy models, we can enhance the if-then rules by finding the optimal weights of the rules.
- In order to reduce computational complexity in training maximum entropy models, we adopt the *active learning* framework. Rather than using all given training examples, we intelligently select the critical informative examples during the learning process, so that it is possible to provide the necessary amount of information needed to learn the maximum entropy model with a far less number of training examples (Zhang, 1994).

Another severe obstacle is that the data for text chunking are highly imbalanced like other natural language resources. Only a few chunk types occupy most of the cases in the data while there are a number of different chunk types. Such imbalance deteriorates the performance of the most machine learning algorithms due to their low recall (Kubat & Matwin, 1997). We adopt the AdaBoost algorithm based on the trained maximum entropy models to resolve this problem. Combining the outputs of several learners leads to improved performance, because combining learners reduces the variance in boundary locations of the chunk type space. As a result, not only the recall but also the precision are improved by AdaBoost, so that the general performance can be improved.

The rest of this paper is organized as follows. Section 2 gives a brief introduction to the maximum entropy model for text chunking. Section 3 explains how the features for the maximum entropy model are automatically constructed from decision trees. Section 4 describes the active learning method for building the maximum entropy model. Section 5 explains the imbalanced data problem and proposes the AdaBoost algorithm to solve it. Section 6 presents the dataset and reports the experimental results. Finally, Section 7 draws conclusions.

2. A Maximum Entropy Model for Text Chunking

Text chunking can be considered to be a classification problem in machine learning, in which the goal is to observe some linguistic context $x \in \mathcal{X}$ and predict the chunk label $y \in \mathcal{Y}$ of it. A conditional probability distribution $p(y|x)$ is one way to implement the classifier for predicting y.

Maximum entropy models combine diverse pieces of contextual evidence to estimate the probability of y with x. They implement the intuition that if there is no evidence to favor one alternative solution above others, all alternatives should be equally likely. That is, a maximum entropy model has the form:

$$p(y|x) = \frac{\prod_{i=1}^{k} \mu_i^{f_i(x,y)}}{Z},$$

where μ_i are expressed as

$$\mu_i = \exp(\lambda_i)$$

and λ_i are real-valued parameters. Here, Z is a normalizing constant and $f_i(x, y)$ are arbitrary computable *features* of an example (x, y).

The optimal distribution p^* is the most uncertain one:

$$p^* = \arg\max_{p \in \mathcal{P}} H(p),$$

where $\mathcal{P}$ is a set of distributions satisfying $E_p[f_i] = E_{\tilde{p}}[f_i]$ for all i. Given a training set $S = \{(x_1, y_1), \ldots, (x_N, y_N)\}$, $E_p[f_i]$ and $E_{\tilde{p}}[f_i]$ are

$$\begin{aligned} E_{\tilde{p}}[f_i] &= \sum_{x,y \in S} \tilde{p}(x, y) f_i(x, y) \\ E_p[f_i] &= \sum_{x,y} \tilde{p}(x) p(y|x) f_i(x, y), \end{aligned}$$

where $\tilde{p}$ is the empirical distribution. The parameters $\vec{\mu} = \{\mu_1, \ldots, \mu_k\}$ of p^* can be found by an iterative procedure such as the Generalized Iterative Scaling (GIS) algorithm (Darroch & Ratcliff, 1972).

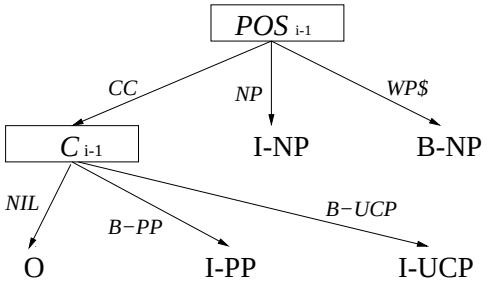


Figure 1. An example decision tree for text chunking. This tree is trained with the features, POS_{i-1} and C_{i-1} for the classification of chunk type C_i of a word W_i. POS_{i-1} is the part-of-speech of the previous word W_{i-1} and C_{i-1} is the chunk type of W_{i-1}.

In using the maximum entropy models, we have two specific issues to consider. One is how to select the features, f_i. The features of the maximum entropy model, in general, are selected by the modeler to capture the aspects of the data. Thus, it is very difficult to construct f_i's if the modeler does not have sufficient knowledge about the problem domain. The other problem is that in training a maximum entropy model it is needed to estimate $E_p[f_i]$ for all f_i's at each iteration of the GIS procedure. Since $E_p[f_i]$ requires a summation over all training examples, estimating it is often infeasible for the problems with a great number of training examples.

In order to overcome these two problems, we use decision trees and active learning. Since decision trees can be represented as a set of if-then rules, the features are automatically constructed by transforming the decision trees into if-then rules. The decision trees are learned with simple linguistic models, such as n-gram. For the complexity problem in estimating $E_p[f_i]$, we use an active learning method in which informative examples are chosen to maximize the learning effect while reducing the training set size.

3. Constructing Features for Maximum Entropy Models

As mentioned above, the trained decision trees are used to construct features in maximum entropy models since they can be easily re-represented as a set of if-then rules. For instance, we can construct five rules from the decision tree in Figure 1 as follows:

1. $f_1(POS_{i-1}, C_{i-1}, C_i)$
   $= \begin{cases} 1 & \text{if } POS_{i-1} = \textit{WP\$} \text{ and } C_i = \textit{B-NP} \\ 0 & \text{otherwise} \end{cases}$

2. $f_2(POS_{i-1}, C_{i-1}, C_i)$

$$= \begin{cases} 1 & \text{if } POS_{i-1}\text{=}NP \text{ and } C_i\text{=}I\text{-}NP \\ 0 & \text{otherwise} \end{cases}$$

3. $f_3(POS_{i-1}, C_{i-1}, C_i)$
$$= \begin{cases} 1 & \text{if } C_{i-1} = B\text{-}UCP \text{ and } POS_{i-1}\text{=}CC \\ & \text{and } C_i\text{=}I\text{-}UCP \\ 0 & \text{otherwise} \end{cases}$$

4. $f_4(POS_{i-1}, C_{i-1}, C_i)$
$$= \begin{cases} 1 & \text{if } C_{i-1} = B\text{-}PP \text{ and } POS_{i-1}\text{=}CC \\ & \text{and } C_i\text{=}I\text{-}PP \\ 0 & \text{otherwise} \end{cases}$$

5. $f_5(POS_{i-1}, C_{i-1}, C_i)$
$$= \begin{cases} 1 & \text{if } C_{i-1} = NIL \text{ and } POS_{i-1}\text{=}CC \\ & \text{and } C_i\text{=}O \\ 0 & \text{otherwise} \end{cases}$$

Because this is a decision tree trained with two simple features POS_{i-1} and C_{i-1} to determine the chunk type C_i of a word W_i, all the rules derived from this tree will have three arguments of POS_{i-1}, C_{i-1} and C_i. The number of the rules is equivalent to that of the leaves in the tree. Therefore, once we have trained decision trees we can automatically construct from them the features for maximum entropy models.

In the viewpoint of decision trees, the features try to partition the training set into disjoint equivalent subsets. If they can predict perfectly all training examples, the partitioned subsets are optimal at least for the training examples. In such a case, it is of no use to further investigate the features. However, when they can not classify the training examples perfectly, misclassification training examples takes place since the subsets have no overlap. Thus, we can improve the features by admitting the overlap between the subsets.

In the viewpoint of maximum entropy models, the features that are automatically constructed from decision trees are considered to have identical weights. Since they try to partition the example space with no overlap, each feature is considered to have the same importance weight, meaning that all features have the same value of μ_i. Thus, if we try to correctly classify the examples that are misclassified by decision trees, the values of μ_i should be changed and optimized. Since we formulate the problem in a maximum entropy model, the optimal value of μ_i's can be found by the GIS algorithm (Berger et al., 1996).

For instance, assume that the if-then rules are weighted by a maximum entropy model as follows:

Rule	f_1	f_2	f_3	f_4	f_5
μ_i	27.34	5.35	5.37	10.93	4.21

Let us consider an instance where POS_{i-1} is NP, C_{i-1} is *B-PP*, and C_i is *I-PP*. The chunk type C_i will be predicted to be *I-NP* by the decision tree. However, it will be the correct chunk type, *I-PP* with the weighted rules, because the weight of f_4 is larger than that of f_2 under the same Z. That is,

$$p(I\text{-}PP \mid NP, B\text{-}PP) > p(I\text{-}NP \mid NP, B\text{-}PP).$$

Besides, we can regard decision trees as a generator of high-order features from first-order features for maximum entropy models. For example, in Figure 1 three first-order features, t, C_{i-1}, and C_i, are given. Assume that all features t, C_{i-1}, and C_i can have 100 values respectively. Then, the space of the high-order features consists of 10^6 elements. In this figure, learning from a decision tree produces only five plausible rules, dramatically reducing the space size.

When we do not have sufficient linguistic knowledge about the target language and text chunking, we can use simple language models such as n-gram as the first-order features to train decision trees. In this paper, we use Quinlan's C4.5 release 8 (Quinlan, 1993) for decision trees.

4. Active Learning to Reduce Training Data

Since the complexity of estimating $E_p[f_i]$ in the GIS algorithm is $O(M|\mathcal{Y}|\mathcal{N})$ (Ratnaparkhi, 1998), where M is the average number of features that are active for a given example, N is the number of training examples, and $\mathcal{Y}$ is the set of labels, it can be intractable to calculate $E_p[f_i]$ as N or M increases. The features are generated to partition the sample space by decision trees, so M is fixed in our problem setting. The only way to reduce the computational complexity is to reduce N. We estimate $E_{\tilde{p}}[f_i]$ by using the empirical expected value:

$$E_{\tilde{p}}[f_i] \approx \frac{1}{n} \sum_{j=1}^{n} f_i(x_j, y_j),$$

where $(x_1, y_1), \ldots, (x_n, y_n)$ are random samples from the unknown distribution p and $n \ll N$. Under this estimation, the methods of reducing the computational complexity of $E_{\tilde{p}}[f_i]$ are divided primarily into two approaches. One is to use a statistical approach and the other is to use an information theoretical approach.

As for statistical sampling methods, Markov Chain Monte Carlo (MCMC) methods are most commonly used to generate random samples from p. Gibbs sampling (Pietra et al., 1997), Metropolis sampling (Chen & Rosenfeld, 1999), and perfect sampling (Amaya & Benedi, 2000) all fall into this category. The main

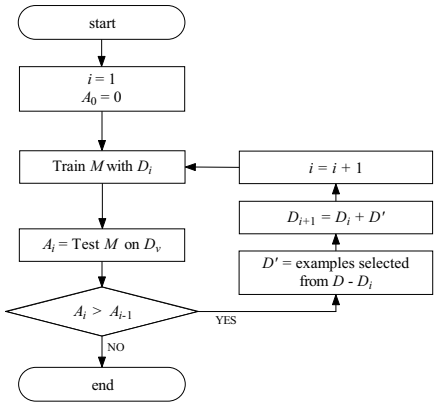


Figure 2. The procedure for active learning. In this figure, M is the training model, D_v is the validation set, and D is the set of total training examples.

drawback of this approach is that the samples are, in practice, drawn from an *estimated* distribution q rather than p, since p is usually unknown. If the distance between p and q is large, these methods will converge very slowly. Thus, in MCMC, the distribution q used to generate new samples must be similar to true distribution p for efficiency. However, it is extremely difficult to find such a distribution in a closed form.

Because most natural language resources are highly redundant, active learning is also widely used to reduce the number of training examples in natural language learning. That is, it is really reasonable to use only some informative samples, in the viewpoint of an information theoretical approach, rather than the whole training examples to calculate $E_{\tilde{p}}[f_i]$.

Figure 2 sketches the procedure for active learning with the maximum entropy models used in this paper. Given D of N training examples, the initial training set D_0 is randomly initialized as $D_0 \subset D$. Assume that we have an independent validation set D_v. At each iteration i, after the model is trained with D_i, the training set is expanded only when the accuracy of the trained model on D_v is higher than the accuracy of the previous iteration. In order to expand the training set, λ examples in $D - D_i$ are selected and appended to current D_i to form D_{i+1}. If A_i, the accuracy of the trained model on D_v does not increase after expanding the training set, the learning process stops.

When selecting an informative example is a computationally expensive process, the advantage of active learning can be lost. Since some informative examples are chosen and the model is retrained, the computational complexity will rather increase if such intermediate cost is larger than the reduction of computational cost obtained by using a smaller number of examples. Adding only one example at each iteration would lead to too many iterations and too many re-learning of models. Thus, the value of a parameter λ should be chosen based on computational complexity and accuracy considerations. In our experiments below λ is chosen as 5% of training examples.

What can be the criterion to select the examples from $D - D_i$? Zhang proved that the active learning which selects the critical examples based on mean square error improves training speed and generalization performance (Zhang, 1994). For the cases where it is too expensive to maintain several classifiers and a single classifier can estimate the uncertainty of an example, the uncertainty estimate can be used to select informative examples. The uncertainty of an example e can be estimated by the distance between class-conditional *a posterior* distribution $P(C|e)$ and the *uniform distribution*. A natural metric to measure the distance between two distributions is the *Kullback-Leibler divergence* (*KL-divergence*), where the KL-divergence between two distributions p and q is defined:

$$KL(p||q) = \sum_{c_i \in C} p(c_i) \ln \frac{p(c_i)}{q(c_i)}.$$

Here, c_i is a disjoint set of classes to which an example belongs. Thus, the most uncertain example e^* is determined by:

$$e^* = \arg\min_{e_j \in D - D_i} KL(U \;||\; P(C|e_j)),$$

where U is the uniform distribution.

5. AdaBoost for Imbalanced Data

Most natural language resources such as the CoNLL-2000 dataset are highly imbalanced. For example, in CoNLL-2000 only four major chunk types (NP, PP, VP, and O) among 12 occupy 95.10% of the training data. The high imbalance deteriorates the performance of the learners because of their low recall (Kubat & Matwin, 1997). We use the AdaBoost algorithm to solve this problem.

Since AdaBoost focuses on the instances that are hard to classify, it gives priority to the rare categories as the iteration passes. The margin-based algorithms such as AdaBoost and support vector machines concentrate on the examples near the boundary of hyperplane, so that they have the effect of biased sampling for rare categories. In addition we can, in general, expect not only higher recall but also better precision by AdaBoost, because it can be regarded as a committee model (Tumer & Ghosh, 1996).

Table 1. The linguistic attributes used to construct feature functions of maximum entropy models.

Attribute	Meaning
POS_{i-2}	Part-of-speech tag of word W_{i-2}
POS_{i-1}	Part-of-speech tag of word W_{i-1}
POS_i	Part-of-speech tag of word W_i
POS_{i+1}	Part-of-speech tag of word W_{i+1}
POS_{i+2}	Part-of-speech tag of word W_{i+2}
W_i	Word W
C_{i-2}	Chunk tag of word W_{i-2}
C_{i-1}	Chunk tag of word W_{i-1}

6. Experiments

6.1. Dataset

We use *CoNLL-2000 Shared Task* data (CoNLL, 2000) as a dataset, which consists of a training set and a test set which are a part of Wall Street Journal (WSJ) corpus. The training set, which are Sections 15–18 of the WSJ corpus, consists of 211,727 words and the test set consists of 47,377 words from Section 20 of the WSJ corpus. Each word in both training and test sets has two additional tags, which are a part-of-speech tag and a chunk tag. The part-of-speech tags are derived by the Brill tagger. The chunk tags contain the name of the chunk type. Most chunk types have two kinds of chunk tags. For instance, B-NP represents the first word of the noun phrase chunk while I-NP is given to other words in noun phrase chunk. In this dataset, they assume 11 types of phrase and one additional chunk tag, O, so that there exist 23 chunk types. The O chunk tag is used for words which are not part of any chunk. To adapt the dataset in our active learning, we divide the given training set into two subsets, a smaller training set D and the validation set Dv. In our experimental settings, D consists of 148,181 words and Dv of 63,546 words.

6.2. First Order Features

To chunk English sentences, we use the mixture of n-gram language models for part-of-speech and chunk type. Table 1 shows the linguistic attributes to learn decision trees. To determine the chunk tag of word W_i, the part-of-speech tags of surrounding words, W_i itself, and the chunk tags of two previous words W_{i-2} and W_{i-1}. For the surrounding words, two words of the left context and two words of the right context are used, since the right context words may give information on chunking. Especially when W_{i+1} or W_{i+2} is the last word of the sentence, it is usually the punctuation mark that makes W_i be the end of a chunk. Since chunking is performed sequentially, the chunk tag of W_{i+1} or W_{i+2} are not known in determining the chunk tag of W_i. Thus, we use only the chunk tags of two previous words.

We do not include any lexical information for the contextual words. Adding many lexical information produces the decision trees with a tremendous number of leaves, which makes the computation of maximum entropy intractable. In addition, we found that using only lexical information of W_i gives better performance in our experimental settings.

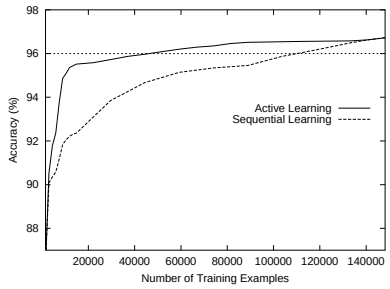
Table 2. The experimental results for text chunking.

Method	Our Method	C4.5
Accuracy	96.72%	94.38%
Precision	91.62%	89.16%
Recall	93.36%	91.25%
F-score	92.48	90.20
No. of Errors	1,554	2,663

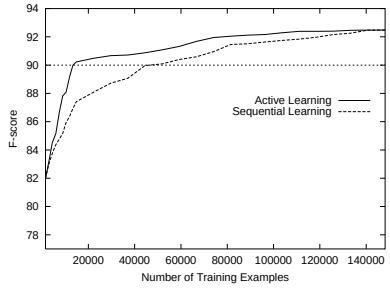
6.3. Experimental Results

Table 2 gives the experimental results of the proposed method in text chunking. The proposed method achieves 96.72% of accuracy and 92.48 of F-score, which are the improvement over C4.5 of 2.34% in accuracy and 2.28 in F-score. The improvement seems to be minute, but the number of errors is reduced by 41.64% from 2,663 to 1,554. This improvement is significant because chunking is in general an intermediate step for full parsing and the errors in this step can not be recovered in the following steps of language processing. When processing a huge text set, and considering an average length per sentence of 10, if the accuracy is about 90% then each sentence, on average, contains one error. Thus, it is important to increase 1∼2% of accuracy in automatic chunk tagging. In addition, this improvement is important since decision trees themselves are strong classifiers. Thus, the proposed method plays a role in enhancing the performance of existing decision trees.

Figure 3 shows how effective the active learning is in reducing the number of training examples. It depicts the learning curves of the proposed method in accuracy and F-score. In this figure, '*Sequential Learning*' implies that the examples in the training set are added in order, while '*Active Learning*' selects the examples by its preference. The Y-axis represents the accuracy and F-score of the text chunking, while the X-axis indicates the number of training examples. The difference between the two lines is the performance improved by using active learning.



(a) Accuracy



(b) F-score

Figure 3. The performance of the proposed method in varying the number of training examples. Both figures compare the active learning and the sequential learning. Plot (a) shows the accuracy curves on the test set, while (b) is for the F-score curves.

To achieve 96% of accuracy on the test set, the active learning needs just about 50,000 training examples, which are far less than the half of those needed for the sequential learning and are only about 34% of the total training examples. After more than 80,000 training examples are used which are just about 54% of the total examples, there is little improvement for the accuracy. Though the model stops learning after 80,000 training examples it shows 96.46% of accuracy, which is the similar accuracy to the case where the whole training examples are used. For F-score, the learning curve shows similar trends. Only a quarter of training examples needed for sequential learning is actually required to achieve 90 in active learning. As is in accuracy, we obtain 92.05 of F-score with only 80,000 training examples in which the learning curve gets flat. This value is rivaling that of the learning with total training examples.

It is interesting to check how many features are generated from decision trees during active learning. Figure 4 shows that more features are constructed from decision trees when using active learning with the same number of training examples. The thick line is the number of features generated in active learning by varying the number of training examples, and the dotted line is that in sequential learning. Since the active learning handles more uncertain examples first, it produces more rules to express them, so that more features are constructed. For example, with 90,000 examples active learning generates 1,700 kinds of rules, while only 1,400 rules are produced in sequential learning.

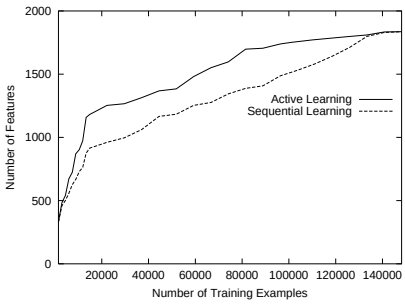


Figure 4. The number of features generated from decision trees in varying the number of training examples.

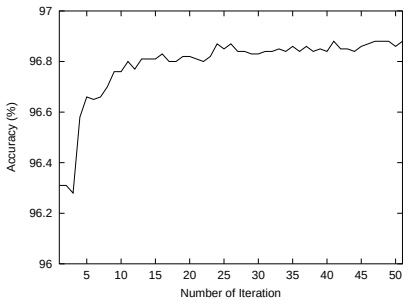
Table 3. The comparison of experimental results with normal maximum entropy models. ME1 is the ME with first-order features, ME2 is the ME with high-order features, and DT is the decision trees.

Method	ME1	ME2	DT	Our Method
Accuracy	95.5%	N/A	94.87%	96.29%
Precision	N/A	92.08%	89.59%	90.57%
Recall	N/A	91.86%	89.94%	92.63%
F-score	90.5	91.97	88.75	91.59

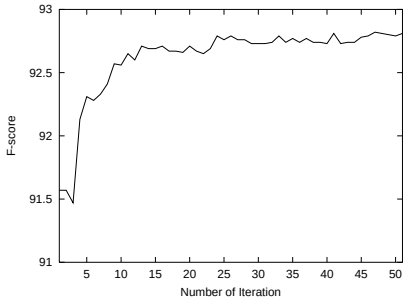
6.4. Comparison with the Normal Maximum Entropy Model

Koeling reported the experimental results using maximum entropy models with first order lexical information (Koeling, 2000), which are a left context of 3 words and a right context of 2 words. The POS tags of the all surrounding words and the words themselves are used as the first order features. The lexical information used consists of the previous word, the current word and the next word. Using only these first order features, the maximum entropy model scores an accuracy of about 95.5% and an F-score of about 90.5, while the decision tree shows an accuracy of 94.87% and an F-score of 88.75.

Even though we obtained the best result with the features in Table 1, we use the same first order features with Koeling's in this section to make the comparison of our method to that of Koeling's in perspective. With these features, the proposed method achieves an accuracy of 96.29% and an F-score of 91.59 (Table 3).



(a) Accuracy



(b) F-score

Figure 5. The effect of AdaBoost for the proposed method. Figure (a) is measured on the accuracy while (b) is on the F-score.

This results are obtained by training with full training examples given in the CoNLL-2000 dataset. The performance is far higher than the decision trees and the maximum entropy model with only first-order features. Although it shows a little worse precision than the maximum entropy model with the selected high-order features, its recall is higher, so that the proposed method gives similar performance to those of the maximum entropy model with high-order features. Though more features than in Table 1 are used, the performance of the proposed method rather decreased by 0.89 in F-score. This implies that some features used in Koeling's experiments are irrelevant at least in our experimental settings.

6.5. AdaBoost Effect

Figure 5 shows the effect of AdaBoost for text chunking on the CoNLL-2000 dataset. Until the 13th iteration, both the accuracy and F-score get higher and higher after each iteration. After that, the performance gets flat though there is some minor change in both measures. Consequently, both the accuracy and F-score get higher by boosting and the improvement in F-score is far larger than that of accuracy. This is because the precision is also improved by boosting, in addition to the expectation that the recall will be improved.

Table 4. The final results. The accuracy is 96.88%.

Type	Precision	Recall	F-score
ADJP	62.23%	65.07%	63.62
ADVP	74.48%	78.87%	76.61
CONJP	40.00%	66.67%	50.00
INTJ	100.00%	50.00%	66.67
LST	0.00%	0.00%	0.00
NP	92.49%	94.75%	93.61
PP	96.63%	97.71%	97.17
PRT	73.74%	68.87%	71.22
SBAR	90.89%	87.66%	89.25
VP	92.67%	93.58%	93.12
All	91.96%	93.69%	92.82

Table 4 is the performance measured when the AdaBoost gives the best performance. It reports 96.88% of accuracy and 92.82 of F-score. Compared with Table 2, the F-score is improved by up to 0.34 by the AdaBoost while the accuracy is improved by 0.16%. The effect of boosting does not seem to be significant. One possible explanation is that the maximum entropy model is already strong enough to boost up. However, the performance in Table 4 is far better than that of the maximum entropy model of Koeling's (see Table 3). Though this is not the best result reported using CoNLL-2000 dataset (Zhang et al., 2001), it shows that the proposed method can overperform the maximum entropy models with the features selected carefully.

7. Conclusion

In this paper we have proposed a new method for learning text chunking using maximum entropy models. This method resolves three major limitations in applying maximum entropy models to text chunking, where two limitations come from the maximum entropy model itself and the remaining one is from highly imbalanced distribution of chunk types in the corpus. The first limitation is the shortage of prior knowledge for the target problem in constructing the features for maximum entropy models. This limitation is solved by automatically inducing the features for the maximum entropy models from decision trees trained with a simple n-gram language model. This method can also be considered to generate high-order features from the simple first-order features using decision trees. This is important because the need for experts is removed who has much knowledge on the problem domain and is required to make the proper features for maximum entropy models. The experiments on the CoNLL-2000 dataset show that the proposed method achieves 92.48

of F-score, which is superior to either the decision trees or the maximum entropy models alone with the features selected by human experts.

The proposed method also addresses the problem of the high computational complexity in learning maximum entropy models. This problem is solved by employing the active learning paradigm, which samples the informative examples first. By active learning we obtain the performance comparable to that of the model using whole examples, but with only 54% of the total examples. Since the natural language resources are usually highly redundant, active learning is useful for most machine learning approaches to this domain.

Another limitation is the high imbalance of chunk types in the corpus. By applying AdaBoost, both the precision and the recall are much improved while there is minor improvement in accuracy. This improvement is important since in text chunking the combined score of recall and precision, such as F-score, is far more important than the accuracy. As a final result, we achieved 92.82 of F-score, which is comparable to the best result reported in the text chunking competition.

One of the main contribution of this paper is that the proposed method opens a new way to automatic selection of proper high-order features for maximum entropy models from a simple language model. This method minimizes the interference of human experts in learning maximum entropy models for text chunking. From a practical point of view the task of the experts is simplified into selecting the first-order features. This is done efficiently, leading to a reduced time in finding proper features. Other machine learning methods, such as active learning and AdaBoost, improve the performance of the maximum entropy models in addition to reducing the model construction time.

Acknowledgements

This research was supported by the Korean Ministry of Education under the BK21-IT Program, by BrainTech program sponsored by the Korean Ministry of Science and Technology, and by the Korea Science and Engineering Foundation (KOSEF) through the Advanced Information Technology Research Center (AITrc).

References

Amaya, F., & Benedi, J. M. (2000). Using perfect sampling in parameter estimation of a whole sentence maximum entropy language model. *Proceedings of CoNLL-2000 and LLL-2000* (pp. 79–82).

Argamon, S., Dagan, I., & Krymolowski, Y. (1998). A memory-based approach to learning shallow natural language patterns. *Proceedings of COLING/ACL 1998* (pp. 67–73).

Berger, A., Pietra, S., & Pietra, V. (1996). A maximum entropy approach to natural language processing. *Computational Linguistics*, *22*, 39–71.

Chen, S., & Rosenfeld, R. (1999). Efficient sampling and feature selection in whole sentence maximum entropy language models. *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing* (pp. 549–552).

CoNLL (2000). *Shared task for computational natural language learning (CoNLL).* http://lcg-www.uia.ac.be/conll2000/chunking.

Darroch, J., & Ratcliff, D. (1972). Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, *43*, 1470–1480.

Koeling, R. (2000). Chunking with maximum entropy models. *Proceedings of CoNLL-2000 and LLL-2000* (pp. 139–141).

Kubat, M., & Matwin, S. (1997). Addressing the curse of imbalanced training sets: One-sided selection. *Proceedings of the 14th International Conference on Machine Learning* (pp. 179–186).

Pietra, S., Pietra, V., & Lafferty, J. (1997). Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*, 380–393.

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

Ramshaw, L., & Marcus, M. (1995). Text chunking using transformation-based learning. *Proceedings of the Third ACL Workshop on Very Large Corpora* (pp. 82–94).

Ratnaparkhi, A. (1998). *Maximum entropy models for natural language ambiguity resolution.* Doctoral dissertation, Univ. of Pennsylvania.

Tumer, K., & Ghosh, J. (1996). Analysis of decision boundaries in linearly combined neural classifiers. *Pattern Recognition*, *29*, 341–348.

Zhang, B.-T. (1994). Accelerated learning by active example selection. *International Journal of Neural Systems*, *5*, 67–75.

Zhang, T., Damerau, F., & Johnson, D. (2001). Text chunking using regularized winnow. *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics* (pp. 539–546).