# Massively Parallel Part of Speech Tagging Using Min-Max Modular Neural Networks

Bao-Liang Lu †, Qing Ma ‡, Michinori Ichikawa †, & Hitoshi Isahara ‡
†Lab. for Brain-Operative Device, Brain Science Institute, RIKEN
2-1 Hirosawa, Wako-shi, 351-0198, Japan
{lu;ichikawa}@brainway.riken.go.jp
‡Communications Research Laboratory,
Ministry of Posts and Telecommunications, Kobe, 651-2401, Japan
{qma;isahara}@crl.go.jp

Abstract: This paper presents a massively parallel tagging method for automatically assigning the correct part of speech (POS) tag to each ambiguous word in a sentence in the context of the sentence. This method is based on the min-max modular neural network, an efficient modular neural network model for solving large-scale pattern recognition problems. The method has two attractive features. One is that it can efficiently solve large-scale POS tagging problems in a massively parallel way. The other is that the tagger has quick response and can be easily implemented in hardware. We demonstrate the performance of the proposed method by using a Thai corpus with 22,311 ambiguous words for training and 6,717 ambiguous words for test. The experimental results show that the method is superior to the existing tagging methods in the aspects of training time, learning accuracy and generalization performance.

## 1 Introduction

Part of speech tagging is the problem of assigning the correct part of speech tag to each ambiguous word in a sentence in the context of the sentence. It is believed that POS tagging is one of the basic techniques in natural language processing. In the last several years, various POS tagging systems for different languages [2, 10, 17] have been developed by using statistical model [13], rule-based approach [1], machine learning technique [15], and neural networks [9, 11, 12, 17]. POS tagging techniques have been applied to pre-processing for speech synthesis, post-processing for continuous speech recognition, machine translation, and information retrieval. However, most of the existing POS taggers suffer from the following two deficiencies: (a) it needs long training time for solving large-scale POS tagging problems; and (b) the organization of POS taggers is non-modular structure, and therefore they are difficult to be implemented in hardware.

In order to overcome the above drawbacks of the existing tagging methods, we propose a massively parallel POS tagging method based on min-max modular ($M^3$) neural network. The remainder of the paper is organized as follows. In Section 2, we briefly describe the POS tagging problem. In Section 3, we introduce the min-max modular neural network. In Section 4, we demonstrate the performance of the proposed method by tagging Thai texts. Finally, conclusions are presented in Section 5.

## 2 POS Tagging Problem

Generally speaking, words are ambiguous in term of their POS tags. However, each word in a sentence has only one correct POS tag from grammatical point of view. Consider a sentence

$$W = w_1 w_2 \cdots w_S \tag{1}$$

where $w_i$ for $i = 1, 2, \cdots, S$ denotes a word, and $S$ is the length of the sentence.

Suppose that a set of POS tags is given by

$$\Lambda = \{\lambda_1, \lambda_2, \cdots, \lambda_M\}, \tag{2}$$

where $\lambda_i$ is a POS tag and $M$ is the total number of types of POS tags.

The POS tagging problem is to assign the correct POS tag $\lambda \in \Lambda$ to each ambiguous word $w_i$ in $W$.

## 3 Min-Max Modular Network

### 3.1 Task Decomposition

Let $T$ be the training set for a $K$-class classification problem,

$$T = \{(\boldsymbol{x}_t, \boldsymbol{y}_t)\}_{t=1}^{N}, \tag{3}$$

where $\boldsymbol{x}_t \in \mathbf{R}^n$ is the input vector, $\boldsymbol{y}_l \in \mathrm{R}^K$ is the desired output, and $N$ is the total number of training data.

We suggest that a $K$-class problem as defined in (3) can be divided into $\binom{K}{2}$ relatively smaller two-class subproblems. The training set $T_{ij}$ for each of the subproblems is given by

$$T_{ij} = \{(\boldsymbol{x}_l^{(i)},\ 1-\epsilon)\}_{l=1}^{L_i} \cup \{(\boldsymbol{x}_l^{(j)},\ \epsilon)\}_{l=1}^{L_j} \quad \text{for } i = 1,\ \cdots,\ K \text{ and } j = i+1,\ \cdots,\ K, \tag{4}$$

where $\epsilon$ is a small real positive number, $\boldsymbol{x}_l^{(i)} \in \mathcal{X}_i$ and $\boldsymbol{x}_l^{(j)} \in \mathcal{X}_j$ are the training inputs belonging to class $\mathcal{C}_i$ and class $\mathcal{C}_j$, respectively, $\mathcal{X}_i$ is the training input set of class $\mathcal{C}_i$, and $L_i$ denotes the number of data in $\mathcal{X}_i$ for $i = 1,\ \cdots,\ K$.

Assume that $\mathcal{X}_i$ is partitioned into $N_i\,(1 \le N_i \le L_i)$ subsets in the form

$$\mathcal{X}_{ij} = \{\boldsymbol{x}_l^{(ij)}\}_{l=1}^{L_i^{(j)}} \quad \text{for } j = 1,\cdots,\ N_i \text{ and } i = 1,\ \cdots,\ K, \tag{5}$$

where $\cup_{j=1}^{N_i} \mathcal{X}_{ij} = \mathcal{X}_i$.

According to the above partition of $\mathcal{X}_i$, a $K$-class classification problem can be divided into

$$\sum_{i=1}^{K} \sum_{j=i+1}^{K} N_i \times N_j \tag{6}$$

relatively smaller and simpler two-class subproblems.

The training set for each of the subproblems is given by

$$T_{ij}^{(u,v)} = \{(\boldsymbol{x}_l^{(iu)},\ 1-\epsilon)\}_{l=1}^{L_i^{(u)}} \cup \{(\boldsymbol{x}_l^{(jv)},\ \epsilon)\}_{l=1}^{L_j^{(v)}} \quad \text{for } u = 1,\ \cdots,\ N_i,\ v = 1,\ \cdots,\ N_j,\ i = 1,\ \cdots,\ K, \text{ and } j = i+1,\ \cdots,\ K, \tag{7}$$

where $\boldsymbol{x}_l^{(iu)} \in \mathcal{X}_{iu}$ and $\boldsymbol{x}_l^{(jv)} \in \mathcal{X}_{jv}$ are the training inputs belonging to class $\mathcal{C}_i$ and class $\mathcal{C}_j$, respectively.

If the training set $T_{ij}^{(u,v)}$ has only two different elements in the form

$$T_{ij}^{(u,v)} = \{(\boldsymbol{x}_1^{(iu)},\ 1-\epsilon) \cup (\boldsymbol{x}_1^{(jv)},\ \epsilon)\} \quad \text{for } u = 1,\ \cdots,\ L_i,\ v = 1,\ \cdots,\ L_j,\ i = 1,\ \cdots,\ K, \text{ and } j = i+1,\ \cdots,\ K, \tag{8}$$

this problem is obviously a linearly separable problem because any two different training data can always be separated by a hyper-plane.

From (4) and (7), we can see that dividing a $K$-class classification problem into a number of two-class subproblems is simple and straightforward, and no domain specialists or a prior knowledge concerning the decomposition of the problem are required. Consequently, the task decomposition can be performed automatically.

### 3.2 Massively Parallel Learning

An important feature of the proposed task decomposition method is that each of the two-class subproblems obtained can be treated as a completely separate classification problem in the learning phase. Consequently, all of the subproblems can be learned in parallel.

Let $N$ be the total number of training data for a $K$-class classification problem, then

$$N = K \times L, \tag{9}$$

where for simplicity of description, the assumption we made is that each of the classes has the same number of training data $L$.

If a $K$-class problem is decomposed into $\binom{K}{2}$ two-class subproblems, the number of training data for each of the subproblems is $2 \times L$. If a $K$-class problem is decomposed into $\sum_{i=1}^{K}\sum_{j=i+1}^{K} N_i \times N_j$ two-class subproblems, the number of of training data for each of the subproblems is about

$$\lceil L/N_i \rceil + \lceil L/N_j \rceil, \tag{10}$$

where $\lceil z \rceil$ denotes the smallest integer greater than or equal to $z$. Since $\lceil L/N_i \rceil + \lceil L/N_j \rceil \ll K \times L$ for a large $K$, i.e., the number of training data for each of the two-class subproblems is much less than the original $K$-class problem. Our experiences indicate that to learn each of the subproblems by using a small network module is much faster than to learn the original large problem by using a large single network [7].

### 3.3 Module Combination

After training each of the modules which were assigned to learn associated subproblems, all of the individual trained modules can be easily integrated into a min-max modular ($\mathrm{M}^3$) network by using the MIN, MAX, or/and INV units according to the module combination principles [7]. Figure 1 illustrates the $\mathrm{M}^3$ network for a four-class pattern classification problem which was decomposed into $\binom{4}{2}$ two-class subproblems according to (4).

Let $\boldsymbol{y}$ denote the actual output vector of the $\mathrm{M}^3$ network for a $K$-class classification problem, and let $g(\boldsymbol{x})$ denote the transfer function of the $\mathrm{M}^3$ network. We may then write

$$\boldsymbol{y} = g(\boldsymbol{x}) = [g_1(\boldsymbol{x}),\ \cdots,\ g_K(\boldsymbol{x})]^T \tag{11}$$

where $\boldsymbol{y} \in \mathbf{R}^K$, and $g_i(\boldsymbol{x}) \in \mathbf{R}$ is called the *discriminant function*, which discriminates the patterns of class $\mathcal{C}_i$ from those of the rest classes.

By replacing the module $M_{st}$ for $s > t$ with the inverse of the output of the module $M_{ts}$, the discriminant functions $g_i(\boldsymbol{x})$ of the $\mathrm{M}^3$ network which is used

to learn the $\binom{K}{2}$ two-class subproblems can be given by

$$g_i(\boldsymbol{x}) = \min\left[\min_{j=i+1}^{K} h_{ij}(\boldsymbol{x}), \min_{r=1}^{i-1}(b - h_{ri}(\boldsymbol{x}))\right] \quad (12)$$

where the term $b - h_{ri}(\boldsymbol{x})$ denotes the inverse of $h_{ri}(\boldsymbol{x})$. which can be implemented by the INV unit, $b$ denotes the upper limit of the output value of each module, and $h_{ij}(\boldsymbol{x})$ is the activation function of the module $M_{ij}$ trained on $\mathcal{T}_{ij}$ as defined in (4).

Similarly, the discriminant function $g_i(\boldsymbol{x})$ of the $M^3$ network which is used to learn $\sum_{i=1}^{K}\sum_{j=i+1}^{K} N_i \times N_j$ two-class subproblems can be expressed as

$$g_i(\boldsymbol{x}) = \min\left[\min_{j=i+1}^{K}\left[\max_{k=1}^{N_i}\left[\min_{l=1}^{N_j} h_{ij}^{(k,l)}(\boldsymbol{x})\right]\right], \min_{r=1}^{i-1}\left(b - \max_{k=1}^{N_r}\left[\min_{l=1}^{N_i} h_{ri}^{(k,l)}(\boldsymbol{x})\right]\right)\right], \quad (13)$$

where the term $b - \max_{k=1}^{N_r}[\min_{l=1}^{N_i} h_{ri}^{(k,l)}(\boldsymbol{x})]$ denotes the inverse of $\max_{k=1}^{N_r}[\min_{l=1}^{N_i} h_{ri}^{(k,l)}(\boldsymbol{x})])$ and $h_{ij}^{(k,l)}$ is the activation function of the module $M_{ij}^{(k,l)}$ trained on $\mathcal{T}_{ij}^{(k,l)}$.

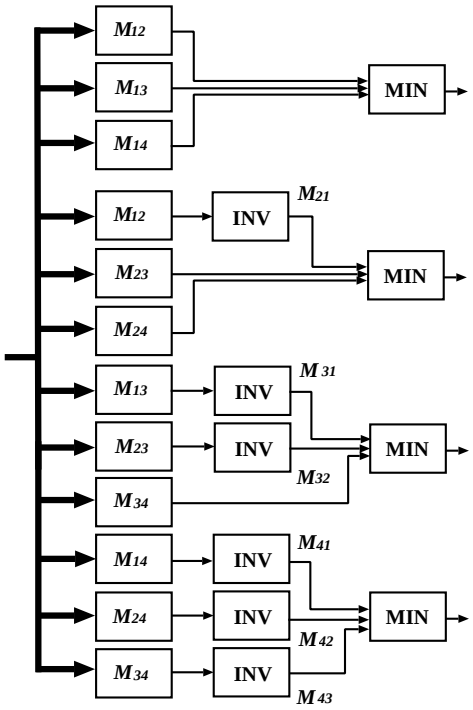


Figure 1: The $M^3$ network for a four-class problem.

## 3.4 Modularity and Parallelism

A neural network is said to be modular if it consists of two or more modules (subnetworks) that operate on distinct inputs without communicating with each other. An output of the modular network is generated by combining the outputs of related modules with integrating units. The function of the integrating unit is to decide which module should output [14].

According to the above definition, the $M^3$ network is a typical modular network. Figures 2(a) and 2(b) represent the structures of the $M^3$ networks for implementing $g_i(x)$ of (12) and $g_i(x)$ of (13), respectively.

The degree of parallelism is the number of subtasks available. According to the task decomposition method presented in Section 2, it is easy to see that any pattern classification problem can be divided into a number of completely independent, non-communicating subtasks. This is ideal case called *completely parallelizable* in parallel computing literature [5], since learning algorithms can achieve linear (i.e., proportional to processing elements) speedup as processing elements (PEs) are added.

## 3.5 Responsiveness

Responsiveness is very important because brain-like computers should have rapid response or real-time response. Let us analyze the responsiveness of the $M^3$ networks.

**Case 1:** If a $K$-class classification problem is just divided into $\binom{K}{2}$ two-class subproblems, the topology of the $M^3$ network for the discriminant function $g_i(x)$ of (12) is a tree with two levels as depicted in Fig. 2(a). The running time for recognizing an input $x$ is given by

$$P_i = P_m + \log(K-1) \quad (14)$$

where $P_m$ is the maximum response time among the $K-1$ individual modules and $\log(K-1)$ is the running time for finding the minimal value from $K-1$ terms with $K(K-1)/2$ PEs [4].

**Case 2:** If a $K$-class classification problem is divided into $\sum_{i=1}^{K}\sum_{j=i+1}^{K} N_i \times N_j$ relatively smaller and simpler two-class problems, the topology of the $M^3$ network for the discriminant function $g_i(x)$ of (13) is a tree with four levels as depicted in Fig. 2(b). Suppose that all the MIN and MAX units are implemented in parallel. The running time for recognizing an input $x$ is given by

$$P_i = P_m + 2\log(N_\alpha) + \log(K-1) \quad (15)$$

where $N_\alpha$ is the maximal value among $N_1, \cdots, N_K$.

## 3.6 Scalability

The simplest definition of *scalability* is that the performance of a computing system increases linearly with respect to the number of PEs used for a given application [5]. However, most of existing neural network models face the *scaling problem*, i.e., training neural

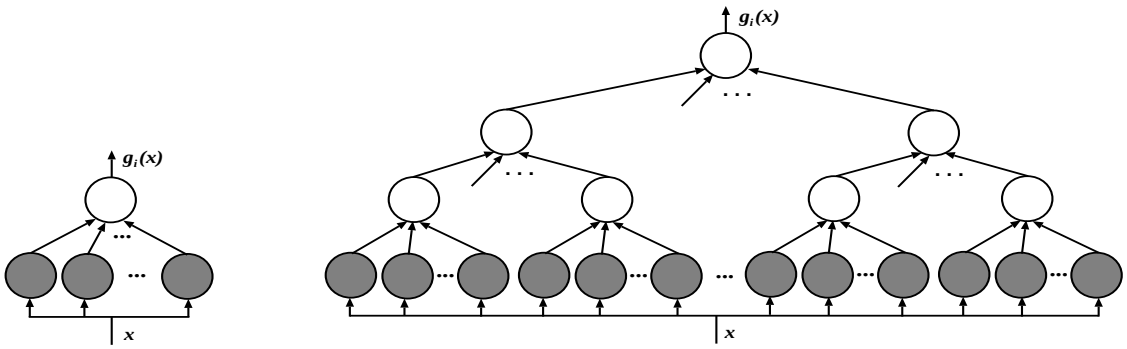


Figure 2: The tree structures of the $M^3$ networks: (a) the tree for the discriminant function as defined in (12) and (b) the tree for the discriminant function as defined in (13), where open circles denote the MIN or MAX units, and dashed circles denote network modules.

networks become intractable as problem sizes get large [6].

The $M^3$ network is able to achieve scalable performance as the learning problem size increases because any problem can be divided into a number of subproblems as small as a user expects and each of the subproblems can be treated as a completely separate problem. Consequently, the proposed learning framework can overcome the scaling problem.

# 4 Tagging Thai Text

## 4.1 Text Data

We use the Thai corpus presented in [3]. It contains 10,452 sentences. Among these sentences, there are 29,028 ambiguous words and 38 kinds of part of speech tags[1]. In this paper, these ambiguous words were randomly divided into two sets. One with 22,311 ambiguous words was used as training set. The other with 6,717 ambiguous words was used as test set.

We consider the tagging problem as a pattern classification problem. According to [12], Each of the training inputs $X_k$ for $k = 1, 2, \cdots, 22,311$ is organized as a vector consisting of seven blocks as follows:

$$X_k = [\boldsymbol{x}_{i-3}\boldsymbol{x}_{i-2}\boldsymbol{x}_{i-1}\boldsymbol{x}_i\boldsymbol{x}_{i+1}\boldsymbol{x}_{i+2}\boldsymbol{x}_{i+3}]^T. \quad (16)$$

where $\boldsymbol{x}_i$ is the block for the word to be tagged, and $\boldsymbol{x}_j$ for $j \in \{i-3,\ i-2,\ i-1,\ i+1,\ i+2,\ i+3\}$ are the blocks for the left and right words.

Each block $\boldsymbol{x}_k$ for $k = i-3, \cdots, i+3$ is also a vector and contains 47 elements, which are used to represent 47 different kinds of part of speech tags. All of the block are in the form

$$\boldsymbol{x}_k = [x_{k1},\ x_{k2},\ \cdots,\ x_{k,47}]^T. \quad (17)$$

[1] The total number of part of speech tags in Thai is 47 [3].

Note that all of the elements in $\boldsymbol{x}_j$ for $j < 1$ or $j > S$ are set to zero.

Suppose that the correct POS tag $\lambda_p \in \Lambda$ has been assigned to the $i$th ambiguous word $w_i$ for $1 \le i \le S$ in a sentence $W$ in the given training set. The elements of the central block $\boldsymbol{x}_i$ and the remaining six blocks at the left and right sides $\boldsymbol{x}_j$ for $j \in \{i-3, i-2, i-1, i+1, i+2, i+3\}$ are determined by

$$x_{ip} = Prob(\lambda_p|w_i), \quad (18)$$

where $Prob(\lambda_p|w_i)$ is the probability that the POS tag $\lambda_p$ can be assigned to word $w_i$. It is estimated from the training data as

$$Prob(\lambda_p|w_i) = \frac{|\lambda_p, w_i|}{|w_i|}, \quad (19)$$

where $|\lambda_p, w_i|$ is the number of times that $\lambda_p$ can be assigned to $w_i$, and $|w_i|$ is the number of times that $w_i$ appears in all the training data.

If $w_i$ is a word that does not appear in the training set, then each element $x_{ip}$ is calculated by

$$x_{ip} = \begin{cases} \frac{1}{\gamma_i} & \text{if } \lambda_p \text{ is a candidate} \\ 0 & \text{otherwise,} \end{cases} \quad (20)$$

where $\gamma_i$ is the total number of POS tags that can be assigned to $w_i$.

## 4.2 Decomposition of Tagging Problem

Table 1 shows the data distributions in the training and test data sets. According to the task decomposition method [7], the tagging problem can be uniquely divided into $\binom{38}{2}$ two-class subproblems. From Table 1 and the definition of the two-class subproblems defined in (4), we see that the number of training data for the smallest two-class subproblem ($T_{36,38}$) is only 3, and the number of training data for the largest two-class

Table 1: Number of data belonging to each of 38 classes in the Thai corpus

| ♯ | No. of Instances | | ♯ | No. of Instances | |
|---|---|---|---|---|---|
| | Training | Testing | | Training | Testing |
| $L_1$ | 3,041 | 962 | $L_{20}$ | 4 | 0 |
| $L_2$ | 72 | 13 | $L_{21}$ | 76 | 17 |
| $L_3$ | 1,444 | 385 | $L_{22}$ | 4 | 7 |
| $L_4$ | 2,400 | 701 | $L_{23}$ | 9 | 2 |
| $L_5$ | 1,582 | 399 | $L_{24}$ | 57 | 15 |
| $L_6$ | 3,008 | 1011 | $L_{25}$ | 32 | 11 |
| $L_7$ | 12 | 0 | $L_{26}$ | 90 | 34 |
| $L_8$ | 3,197 | 1008 | $L_{27}$ | 30 | 5 |
| $L_9$ | 1,537 | 475 | $L_{28}$ | 6 | 1 |
| $L_{10}$ | 481 | 176 | $L_{29}$ | 88 | 23 |
| $L_{11}$ | 705 | 233 | $L_{30}$ | 2 | 1 |
| $L_{12}$ | 787 | 226 | $L_{31}$ | 177 | 58 |
| $L_{13}$ | 601 | 108 | $L_{32}$ | 6 | 0 |
| $L_{14}$ | 124 | 38 | $L_{33}$ | 8 | 0 |
| $L_{15}$ | 906 | 328 | $L_{34}$ | 17 | 1 |
| $L_{16}$ | 90 | 30 | $L_{35}$ | 20 | 3 |
| $L_{17}$ | 213 | 49 | $L_{36}$ | 2 | 0 |
| $L_{18}$ | 875 | 214 | $L_{37}$ | 131 | 37 |
| $L_{19}$ | 476 | 145 | $L_{38}$ | 1 | 0 |

subproblem ($T_{68}$) is 6,205. Although these two-class subproblems are smaller than the original problem, some of them are still too large for training. Therefore, the large two-class subproblems should be further decomposed.

By using the fine decomposition method [7], each of the large two-class subproblems can be divided into a number of relatively smaller ones in the form as defined in (7). Table 2 shows the number of subsets belonging to each of the following large classes: $\mathcal{C}_1$, $\mathcal{C}_3$, $\mathcal{C}_4$, $\mathcal{C}_5$, $\mathcal{C}_6$, $\mathcal{C}_8$, $\mathcal{C}_9$, $\mathcal{C}_{10}$, $\mathcal{C}_{11}$, $\mathcal{C}_{12}$, $\mathcal{C}_{13}$, $\mathcal{C}_{15}$, $\mathcal{C}_{17}$, $\mathcal{C}_{18}$, and $\mathcal{C}_{19}$. The number of subsets belonging to each of the remaining 23 classes is 1. For example, the training data for class $\mathcal{C}_4$ were divided into 16 subsets, each of which has just 150 patterns. After performing the above partitions, the original tagging problem is divided into

$$\sum_{i=1}^{38} \sum_{j=i+1}^{38} N_i \cdot N_j = 13,081 \qquad (21)$$

relatively smaller two-class subproblems. Thus, among the 13,081 two-class subproblems, the number of training data of the largest subproblem ($\mathcal{T}_{19,31}^{(2,1)}$) is only 336.

## 4.3 Massively Parallel Training

Since each of 13,081 two-class subproblems can be treated as a completely separate two-class pattern classification problem in the learning stage, all of the subproblems can be learned in parallel. In the simulation, the conventional backpropagation algorithm [16] was used. The momentums were all set to 0.9, and the learning rates were all selected as 0.025. Training was stopped when the sum of the squared error was smaller than 0.05 or the total number of epochs reached the given maximum values.

Table 2: Number of subsets belonging to each of 15 large classes

| ♯ | No. of subsets | ♯ | No. of subsets |
|---|---|---|---|
| $N_1$ | 20 | $N_{11}$ | 5 |
| $N_3$ | 10 | $N_{12}$ | 6 |
| $N_4$ | 16 | $N_{13}$ | 4 |
| $N_5$ | 11 | $N_{15}$ | 6 |
| $N_6$ | 20 | $N_{17}$ | 6 |
| $N_8$ | 22 | $N_{18}$ | 6 |
| $N_9$ | 10 | $N_{19}$ | 3 |
| $N_{10}$ | 4 | | |

In the first round of learning, 13,081 three-layer MLPs were selected as network modules to learn the corresponding two-class subproblems. Each of the modules has 329 ($47 \times 7$) input, one hidden unit and one output unit. In this round, 12,169 network modules successfully learned the corresponding subproblems within the given maximum number of epochs. The number of average epochs used is 293. Since many network modules converged very fast, we guess that a number of subproblems are linearly separable problems. After the first round of learning, there were 912 unconverged network modules. Following the module combination principles [7], the 13,081 individual trained modules were integrated into an $M^3$ network. Its learning accuracy and generalization performance, i.e., the success rates on training and test data, are shown in Table 3.

In order to improve the tagging accuracy, the unsolved 912 subproblems were learned again by a bit big three-layer MLPs, each of which has two hidden units. The maximum number of epochs was set to 20,000. After the second round of learning, there are only 91 unconverged network modules. In this stage, the learning accuracy reached 99.7% and the success rate on test data has been improved slightly. After the third round of learning, the success rate on the training data was near to 100%, and the generalization performance was about 93.7%. All of the learning results obtained in the three rounds of learning are shown in Table 3.

In order to compare the performance of the proposed method with the existing tagging approaches, Table 4 shows the experimental results obtained by three different methods. From Table 4, we can see that the proposed method is superior to HMM models and single neural networks in both learning accuracy and generalization performance. In addition, the training time can be dramatically reduced because each of the

Table 3: Three rounds for learning 13,081 two-class subproblems and the related performance.

| ♯ round of learning | ♯ maximum epochs | ♯ minimum epochs | ♯ average epochs | ♯ converged modules | ♯ unconverged modules | ♯ hidden units | Success rate (%) | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | training | test |
| 1 | 2,000 | 7 | 293 | 12169 | 912 | 1 | 98.7 | 93.2 |
| 2 | 20,000 | 207 | 3432 | 821 | 91 | 2 | 99.7 | 93.5 |
| 3 | 50,000 | 351 | 6424 | 57 | 34 | 3 | 99.8 | 93.7 |

Table 4: Accuracy of different methods for tagging Thai text

| Method | No. of Modules | Accuracy (%) | |
|---|---|---|---|
| | | Training | Testing |
| HMM | | 88.3 | 89.1 |
| MLP | 1 | 96.7 | 92.6 |
| $M^3$ network | 13,081 | 99.9 | 93.7 |

subproblems is far smaller than the original problem and learning can be performed in a massively parallel way.

# 5 Conclusions

In this paper we have presented a massively parallel tagging method. The advantages of the method over the existing approaches are its high modularity, parallelism, and scalability. We have demonstrated that the method is superior to HMM models and single neural networks in both learning accuracy and generalization performance. We believe that the proposed method may provide us with an efficient framework for solving very large-scale tagging problems. By using the proposed method, we are doing computer simulations on a Chinese corpus which contains 13,111 sentences, 139,397 ambiguous words, and 63 different tags.

# References


[1] 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. 543-565, 1994.

[2] T. Charoenporn, V. Sornlertlamvanich, and H. Isahara: "Building a large Thai text corpus - part of speech tagged corpus: ORCHID." *Proc. Natural Language Processing Pacific Rim Symposium 1997*, Phuket, Thailand, pp. 509-512, 1997.

[3] W. Daelemans, J. Zavrel, P. Berck, and S. Gillis, "MBT: A memory-based part of speech tagger-generator," *Proc. 4th Workshop on Very Large Corpora*, Copenhagen, Denmark, pp. 1-14, 1996.

[4] M. Garzon, *Models of Massive Parallelism*, Springer, 1995.

[5] K. Hwang, *Advanced Computer Architecture: Parallelism, Scalability, Programmability*, MiGraw-Hill, Inc., 1993.

[6] J. H. Lin and J. S. Vitter, "Complexity results on learning by neural nets", *Machine Learning*, vol.6 , pp211-230, Kluwer Academic Publishers, Boston, 1991.

[7] B. L. Lu and M. Ito, "Task decomposition and module combination based on class relations: a modular neural network for pattern classification", *IEEE Trans. Neural Networks*. vol. 10, no. 5, pp. 1244-1256, 1999.

[8] B. L. Lu, M. Ichikawa, S. Hosoe, "A modular massively parallel learning framework for brain-like computers", *Proc. of IEEE SMC'99*, Tokyo, Oct. 12-15, vol. 5, pp. 332-337, 1999.

[9] Q. Ma and H. Isahara, "A multi-neuro tagger using variable lengths of contexts," *Proc. COLING-ACL'98*, Montreal, pp. 802-806, 1998.

[10] Q. Ma, M. Sun, and H. Isahara, "A multi-neuro tagger applied in Chinese texts", in *Proc. of 1998 Int. Conf. Chinese Info. Processing*, Beijing, Nov. 18-20, pp. 200-207, 1998.

[11] Q. Ma, K. Uchimoto, M. Murata, and H. Isahara, "Elastic neural networks for part of speech tagging," to appear in *Proc. IJCNN'99*, Washington DC., July, 1999.

[12] Q. Ma, B. L. Lu, and H. Isahara, "Part of speech tagging with min-max neural networks", *Proc. of IEEE SMC'99*, Tokyo, Oct. 12-15, vol. 5, pp. 356-360, 1999.

[13] B. Merialdo, "Tagging English text with a probabilistic model," *Computational Linguistics*, Vol. 20, No. 2, pp. 155-171, 1994.

[14] D. N. Osherson, S. Weinstein, and M. Stob, "Modular Learning", *Computational Neuroscience*, E. L. Schwartz, Ed., MIT Press, 1990.

[15] D. Roth and D. Zelenko, "Part of speech tagging using a network of linear separators", in *Proc. of COLING-ALC'98*, Montreal, pp. 1136-1142, 1998.

[16] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation", in *Parallel Distributed Processing: Exploration in the Microstructure of Cognition*, vol. 1, D. E. Rumelhart, J. L. McClelland and PDP Research Group Eds. Cambridge, MA: MIT Press, pp. 318-362, 1986.

[17] H. Schmid, "Part-of-speech tagging with neural networks," *Proc. COLING'94*, Kyoto, Japan, pp. 172-176, 1994.