Applied Intelligence 19, 65–81, 2003

Efficient Part-of-Speech Tagging with a Min-Max Modular Neural-Network Model

BAO-LIANG LU*
Department of Computer Science, Shanghai Jiao Tong University, 1954 Hua Shan Road, Shanghai 200030, People's Republic of China
blu@cs.sjtu.edu.cn

QING MA
Department of Applied Mathematics and Informatics, Faculty of Science and Technology, Ryukoku University, Seta, Otsu 520-2194, Japan
qma@math.ryukoku.ac.jp

MICHINORI ICHIKAWA
Lab. for Brain-Operative Device, RIKEN Brain Science Institute, 2-1 Hirosawa, Wako-shi 351-0198, Japan
ichikawa@brainway.riken.go.jp

HITOSHI ISAHARA
Communications Research Laboratory, 2-2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0289, Japan
isahara@crl.go.jp

Abstract. This paper presents a part-of-speech tagging method based on a min-max modular neural-network model. The method has three main steps. First, a large-scale tagging problem is decomposed into a number of relatively smaller and simpler subproblems according to the class relations among a given training corpus. Secondly, all of the subproblems are learned by smaller network modules in parallel. Finally, following two simple module combination laws, all of the trained network modules are integrated into a modular parallel tagging system that produces solutions to the original tagging problem. The proposed method has several advantages over existing tagging systems based on multilayer perceptrons. (1) Training times can be drastically reduced and desired learning accuracy can be easily achieved; (2) the method can scale up to larger tagging problems; (3) the tagging system has quick response and facilitates hardware implementation. In order to demonstrate the effectiveness of the proposed method, we perform simulations on two different language corpora: a Thai corpus and a Chinese corpus, which have 29,028 and 45,595 ambiguous words, respectively. We also compare our method with several existing tagging models including hidden Markov models, multilayer perceptrons and neuro-taggers. The results show that both the learning accuracy and generalization performance of the proposed tagging model are better than statistical models and multilayer perceptrons, and they are comparable to the most successful tagging models.

Keywords: part-of-speech tagging, min-max modular neural network, parallel learning, Thai text, Chinese text

*To whom all correspondence should be addressed.

1. Introduction

A remarkable characteristic of natural languages is the large number of words that have more than one syntactic category. For example, over 42% of the text in the Chinese corpus [1] used in this paper consists of words that are syntactically ambiguous. Therefore, how to cope with ambiguity is one of the fundamental problems in natural language processing. Part-of-speech tagging is the task of assigning an appropriate part of speech to each word in a sentence. In the last several years, many tagging systems for different languages [1–3] have been developed that are based on various techniques including probabilistic models [4–7], rule-based methods [2], neural networks [8–11], and hybrid systems [12]. Tagging techniques have been applied to various fields of information processing such as pre-processing for speech synthesis, post-processing for continuous speech recognition, machine translation, and information retrieval.

Multilayer perceptrons [13] have been applied to solve part-of-speech tagging problems. Benello et al. [8] developed a tagging system based on three-layer perceptrons. They trained the network on the Brown Corpus with the back-propagation algorithm [14], and achieved an accuracy of 94.7% on test data including both ambiguous and unambiguous words.

Ma and Isahara proposed a multiple neuro-tagger which consists of several three-layer perceptrons [11]. They introduced three techniques to this tagger to improve its generalization performance. (1) Each element of an input is weighted with information gain [15], a number expressing the average amount of reduction of information entropy for the training set; (2) in the tagging phase, instead of using the original inputs directly, each of the input elements corresponding to the left words in a sentence is replaced by its part-of-speech tag that has already been assigned by the tagging system; (3) variant lengths of contexts for tagging are used based on the longest context priority. Ma and Isahara have applied successfully the multiple neuro-tagger to Thai and Chinese, and obtained an accuracy of 94.3% for a Thai corpus [11] and an accuracy of 91.4% for a Chinese corpus [1]. Here the accuracy was measured using only ambiguous words in the test sets. Their experimental results showed that the multiple neuro-tagger is superior to all existing tagging methods, such as rule-based approaches [2], hidden Markov models [6], and multilayer perceptrons [13].

Up to now, variant multilayer perceptrons have been one of the most successful tagging techniques used because of their massively parallel distributed structure, their ability to learn nonlinearly separable problems, and their ability to generalize on novel data [13]. For developing large-scale, practical tagging systems, however, the multilayer-perceptron based tagging methods suffer from several deficiencies. (1) It is difficult to select a suitable network size (e.g., the number of hidden units) to achieve satisfactory learning accuracy and good generalization performance, because there is no theory or method that can guide this selection; (2) it is difficult to improve learning accuracy for large-scale tagging problems even though a very long training time is used, because there is no efficient algorithm for training large-scale multilayer perceptrons; (3) it is hard to implement the tagging systems in hardware because of their non-modular structure.

In order to overcome the above drawbacks of existing tagging systems, this paper proposes a new part-of-speech tagging method based on a min-max modular (M^3) neural-network model [16–18], an efficient learning framework that is capable of solving large-scale pattern classification problems. The remainder of the paper is organized as follows. In Section 2, we briefly describe the part-of-speech tagging problem. In Section 3, we introduce the M^3 network model. In Section 4, we demonstrate the performance of the proposed method on two different corpora: a Thai corpus (ORCHID) [3] and a Chinese corpus[1] [1]. In Section 5, we present comparison studies. Finally, our conclusions are presented in Section 6.

2. The Problem of Part-of-Speech Tagging

Suppose that a lexicon $\mathcal{V}$ with V registered words is given

$$\mathcal{V} = \{w^1,\ w^2, \ldots, w^V\}, \tag{1}$$

where all possible part-of-speech tags for each word are listed up. Let Λ be a set of part-of-speech tags for the lexicon $\mathcal{V}$

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

where M is the number of part-of-speech tags.

The problem of part-of-speech tagging is to find a string of part-of-speech tags R

$$R = \lambda_1 \lambda_2 \ldots \lambda_S \tag{3}$$

for a given sentence W

$$W = w_1 w_2 \dots w_S, \tag{4}$$

where $\lambda_i \in \Lambda$, and $w_i \in \mathcal{V}$ for $i = 1, \dots, S$.

According to the above description, the tagging problem can be regarded as a pattern classification problem as follows:

$$\varphi_t : W^t \to \lambda_t \tag{5}$$

where t is the index of the target word (the word to be tagged), and W^t is a word sequence with length $l + 1 + r$ centered on the target word,

$$W^t = w_{t-l} \dots w_t \dots w_{t+r} \quad \text{for } t - l \geq 1 \quad \text{and } t + r \leq S. \tag{6}$$

3. Min-Max Modular Neural-Network Model

In this section, we give a brief introduction to the M^3 network model, which is pictured schematically in Fig. 1. The model has three components: task decomposition, concurrent learning, and module combination. The central idea underlying the model is to apply the class relations among training data to both task decomposition and module combination. The M^3 network is similar to committee machines [13, 19] in structure. However, the mechanisms of task decomposition and output combination used in the M^3 network are completely distinct from those of existing committee machines, such as piecewise linear machines [19] and a hierarchical mixture of experts [20].

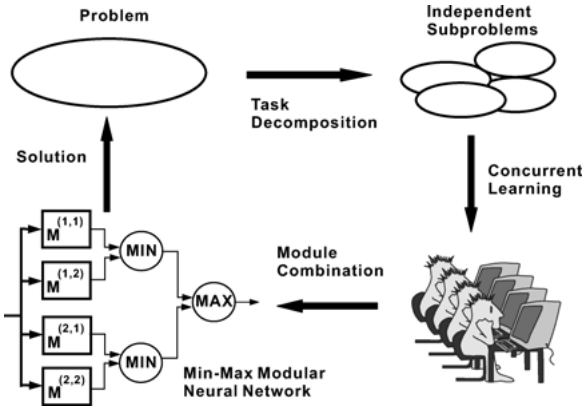

Figure 1. Schematic representation of the min-max modular neural network model. Note that concurrent learning means that the subproblems may be solved either sequentially or in parallel.

3.1. Task Decomposition and Parallel Learning

Breaking up a problem helps human beings deal with complex issues [21]. It has been shown that breaking up a problem is also helpful to neural networks to tackle complex problems [16, 17]. Up to now, various task decomposition methods have been developed. A short survey on this topic can be found in [17].

Let $\mathcal{T}$ be the training set for a K-class classification problem,

$$\mathcal{T} = \{(X_l, Y_l)\}_{l=1}^{L}, \tag{7}$$

where $X_l \in \mathcal{X} \subset \mathbf{R}^n$ is the input vector, $Y_l \in \mathcal{Y} \subset \mathrm{R}^K$ is the desired output, and L is the total number of training data.

We have suggested that a K-class problem defined by (7) can be divided into $K(K-1)$ relatively smaller two-class subproblems based on the class relations among training data [16, 17]. In these $K(K-1)$ two-class subproblems, however, we see that only $\binom{K}{2}$ two-class subproblems need to be learned, and the rest of the $\binom{K}{2}$ two-class subproblems are the same as the former ones from the point of view of pattern classification [16, 17]. The training set for each of the $\binom{K}{2}$ two-class subproblems is given by

$$\mathcal{T}_{ij} = \left\{\left(X_l^{(i)}, 1-\epsilon\right)\right\}_{l=1}^{L_i} \cup \left\{\left(X_l^{(j)}, \epsilon\right)\right\}_{l=1}^{L_j} \quad \text{for } i = 1, \dots, K \text{ and } j = i+1, \dots, K \tag{8}$$

where ϵ is a small real positive number (e.g., ϵ is selected as 0.1 in the simulations below), $X_l^{(i)} \in \mathcal{X}_i$ and $X_l^{(j)} \in \mathcal{X}_j$ are the training inputs belonging to class $\mathcal{C}_i$ and class $\mathcal{C}_j$, respectively, $\mathcal{X}_i$ is the set of training inputs belonging to class $\mathcal{C}_i$, L_i denotes the number of data in $\mathcal{X}_i$, $\bigcup_{i=1}^{K} \mathcal{X}_i = \mathcal{X}$, and $\sum_{i=1}^{K} L_i = L$. The two-class subproblems defined by (8) are called *pairwise classification* in machine learning literature [22]. We would like to emphasize that the partition above is unique for a given set of training data $\mathcal{T}$ because of the uniqueness of $\mathcal{X}_i$.

If some of the two-class problems defined by (8) are still large and hard to be learned, each of these subproblems can be further decomposed into a number of two-class problems as small as the user needs. Assume that $\mathcal{X}_i$ is partitioned into N_i ($1 \leq N_i \leq L_i$) subsets in the form

$$\mathcal{X}_{ij} = \left\{X_l^{(ij)}\right\}_{l=1}^{L_i^{(j)}} \quad \text{for } j = 1, \dots, N_i \text{ and } i = 1, \dots, K, \tag{9}$$

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

According to the above partition of $\mathcal{X}_i$, the two-class problem $\mathcal{T}_{ij}$ defined by (8) can be divided into $N_i \times N_j$ much smaller and simpler two-class subproblems. The training set for each of the two-class subproblems is given by

$$\mathcal{T}_{ij}^{(u,v)} = \left\{\left(X_l^{(iu)},\ 1-\epsilon\right)\right\}_{l=1}^{L_i^{(u)}} \cup \left\{\left(X_l^{(jv)},\ \epsilon\right)\right\}_{l=1}^{L_j^{(v)}} \quad \text{for } u = 1, \ldots, N_i,\ v = 1, \ldots, N_j, \quad i = 1, \ldots, K, \text{ and } j = i+1, \ldots, K, \tag{10}$$

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

If $\mathcal{X}_i$ is partitioned into L_i subsets, the training set $\mathcal{T}_{ij}^{(u,v)}$ has only two different elements in the form

$$\mathcal{T}_{ij}^{(u,v)} = \left\{\left(X_1^{(iu)}, 1-\epsilon\right) \cup \left(X_1^{(jv)}, \epsilon\right)\right\} \quad \text{for } u = 1, \ldots, L_i,\ v = 1, \ldots, L_j, \quad i = 1, \ldots, K, \text{ and } j = i+1, \ldots, K. \tag{11}$$

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

From (8) and (10), we see that a K-class problem can be decomposed into

$$\sum_{i=1}^{K-1} \sum_{j=i+1}^{K} N_i \times N_j \tag{12}$$

two-class subproblems. The decomposition process is simple and straightforward, and no domain specialists or prior knowledge concerning the decomposition of the problem are required. Consequently, any large-scale problem can be easily decomposed into a number of two-class subproblems as small as the user needs.

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

$$L = K \times J, \tag{13}$$

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

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

$$\lceil J/N_i \rceil + \lceil J/N_j \rceil, \tag{14}$$

where $\lceil z \rceil$ denotes the smallest integer greater than or equal to z. Since $\lceil J/N_i \rceil + \lceil J/N_j \rceil \ll K \times J$ 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.

An important feature of the proposed task decomposition method is that each of the two-class subproblems can be treated as a completely independent, non-communicating subproblem in the learning phase. Consequently, all of the subproblems can be learned in parallel. This is an ideal case called *completely parallelizable* in parallel computing literature [23], since learning can achieve linear speedup as processing elements are added.

3.2. Module Combination

After training each of the modules which are assigned to learn associated subproblems, all of the individual trained network modules can be easily integrated into an M^3 network by using the MIN, MAX, or/and INV units according to the following two simple combination laws [16–18], which are called the minimization principle and the maximization principle, respectively.

Theorem 1 (*Minimization Principle*). *Suppose a two-class problem $\mathcal{B}$ is divided into P relatively smaller two-class subproblems, $\mathcal{B}_i$ for $i = 1, \ldots, P$, and also suppose that all the subproblems have the same positive training data and different negative training data. If the P subproblems are correctly learned by the corresponding P individual network modules, M_i for $i = 1, \ldots, P$, then the combination of the P trained network modules with a* **MIN** *unit produces the correct output for all the training inputs in $\mathcal{B}$, where the function of the* **MIN** *unit is to find a minimum value from its multiple inputs.*

Theorem 2 (*Maximization Principle*). *Suppose a two-class problem $\mathcal{B}$ is divided into P relatively smaller two-class subproblems, $\mathcal{B}_i$ for $i = 1, \ldots, P$, and also suppose that all the subproblems have the same negative training data and different positive training data. If the P subproblems are correctly learned by the corresponding P individual network modules, M_i for $i = 1, \ldots, P$, then the combination of the P trained network modules with a* **MAX** *unit produces the correct output for all the training input in $\mathcal{B}$, where the function of the* **MAX** *unit is to find a maximum value from its multiple inputs.*

The proofs of the theorems above are omitted. See [18] for more details.

Let Y denote the actual output vector of the M^3 network for a K-class classification problem, and let $g(X)$ denote the transfer function of the M^3 network. We may then write

$$Y = g(X) = [g_1(X), \ldots, g_K(X)]^T \tag{15}$$

where $Y \in \mathbf{R}^K$, and $g_i(X) \in \mathbf{R}$ 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 $g_i(X)$ of the M^3 network, which is used to learn the $\binom{K}{2}$ two-class subproblems, can be given by

$$g_i(X) = \min\left[\min_{j=i+1}^{K} h_{ij}(X), \min_{r=1}^{i-1} \overline{h_{ri}(X)}\right] \tag{16}$$

where $h_{ij}(X)$ is the activation function of the module M_{ij} trained on $\mathcal{T}_{ij}$ defined by (8), and the term $\overline{h_{ri}(X)}$ denotes the inverse of $h_{ri}(X)$, which can be implemented by an INV unit.

The function of the INV unit is to invert its single input. The transfer function of the INV unit is defined by

$$q = \alpha + \beta - p \tag{17}$$

where α, β, p, and q are the upper and lower limits of input value, input, and output, respectively.

The relationship among $\overline{h_{ri}(X)}$, $h_{ir}(X)$ and the INV unit is given by

$$h_{ir}(X) = \overline{h_{ri}(X)} = \mathrm{INV}(h_{ri}(X)), \tag{18}$$

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

$$g_i(X) = \min\left[\min_{j=i+1}^{K}\left[\max_{k=1}^{N_i}\left[\min_{l=1}^{N_j} h_{ij}^{(k,l)}(X)\right]\right], \min_{r=1}^{i-1}\overline{\max_{k=1}^{N_r}\left[\min_{l=1}^{N_i} h_{ri}^{(k,l)}(X)\right]}\right], \tag{19}$$

where the term $\overline{\max_{k=1}^{N_r}[\min_{l=1}^{N_i} h_{ri}^{(k,l)}(X)]}$ denotes the inverse of $\max_{k=1}^{N_r}[\min_{l=1}^{N_i} h_{ri}^{(k,l)}(X)]$ and $h_{ij}^{(k,l)}$ is the activation function of the module $M_{ij}^{(k,l)}$ trained on $\mathcal{T}_{ij}^{(k,l)}$ defined by (10).

In the following, we present a simple example [18] to illustrate the module combination principles. Consider the XOR problem. According to the task decomposition method, this problem (see Fig. 2(a)) was divided into four linearly separable subproblems: $\mathcal{T}^{(1,1)}$, $\mathcal{T}^{(1,2)}$, $\mathcal{T}^{(2,1)}$, and $\mathcal{T}^{(2,2)}$, which are depicted in Figs. 2(b)–(e), respectively. Four perceptrons represented as $M^{(1,1)}$, $M^{(1,2)}$, $M^{(2,1)}$, and $M^{(2,2)}$ were selected to learn $\mathcal{T}^{(1,1)}$, $\mathcal{T}^{(1,2)}$, $\mathcal{T}^{(2,1)}$, and $\mathcal{T}^{(2,2)}$, respectively. The M^3 network for the XOR problem is shown in Fig. 1. The *optimal boundaries* formed by the four perceptrons are shown in Figs. 3(a)–(d), respectively. The responses of the combinations of individual modules and the whole M^3 network are shown in Figs. 3(e)–(g), respectively. Comparing Fig. 2(a) with Fig. 3(g), we see that the M^3 network forms *optimal boundaries* for the XOR problem.

4. Tagging Experiments

4.1. Text Data

We use the Thai [3] and Chinese [1] corpora in the following tagging experiments. Some tagged Thai and Chinese sentences from these corpora are shown in Figs. 4(a) and (b), respectively. The Thai corpus contains 10,452 tagged sentences. Among these sentences,

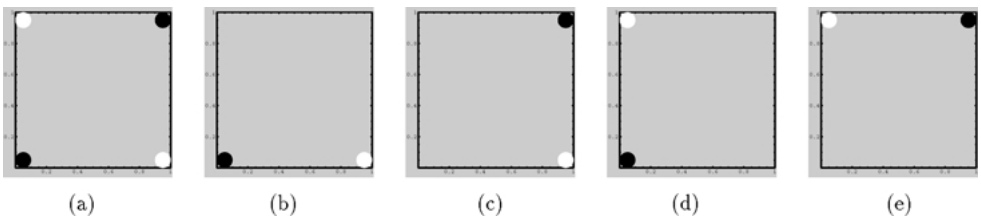

Figure 2. Partition of the XOR problem into four linearly separable subproblems. (a) The training inputs for the original XOR problem, (b) $\mathcal{T}^{(1,1)}$, (c) $\mathcal{T}^{(1,2)}$, (d) $\mathcal{T}^{(2,1)}$, and (e) $\mathcal{T}^{(2,2)}$, respectively. The black and white points represent the inputs whose desired outputs are '0' and '1', respectively, and the grey represents only the background of the figures.

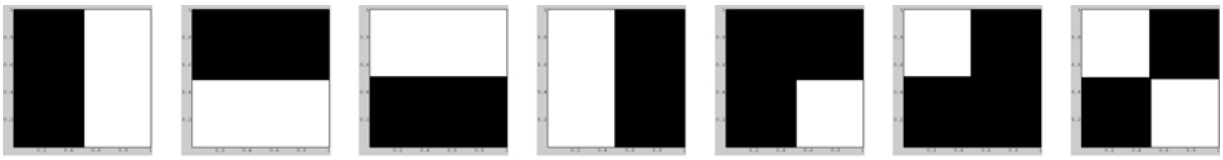

Figure 3. The responses of (a) $M^{(1,1)}$, (b) $M^{(1,2)}$, (c) $M^{(2,1)}$, (d) $M^{(2,2)}$, (e) the combination of $M^{(1,1)}$ and $M^{(1,2)}$ with the MIN unit, (f) the combination of $M^{(2,1)}$ and $M^{(2,2)}$ with the MIN unit, and (g) the whole M^3 network, respectively. The black and white represent the outputs of '0' and '1', respectively.

1 ประเทศไทย@NPRP ก็ได้/EAFF/XVAE@XVBM เริ่ม/VACT@VSTA/XVAM มี/VACT@VSTA นโยบาย@NCMN
ตลอดจน@JCRG การ@FIXN นฺยิบัติ@VACT/VSTA เพื่อ@JSBR พัฒนา@VACT เทคโนโลยี@NCMN ใน@RPRE
ด้าน@NCMN นี้/DDAC/PDMN@DDAN //
2 ซึ่ง@JSBR มี/VACT@VSTA อยู่/VSTA@XVAE หลาย@DIBQ โครงการ@NCMN ที่/NCMN@PREL/RPRE
ประสบ@VSTA ผลสำเร็จ@NCMN เป็น@VSTA ที่/NCMN@PREL/RPRE น่าพอใจ@VSTA //
3 โดย/RPRE@JSBR มี/VACT@VSTA เจตชน@NCMN เข้าร่วม@VACT ดำเนินการ@VACT ด้วย@ADVN //

(a)

1 现在@t 两@mx/qnm 部/ng@qnq 一@mx/wg 社@ng 合并@vg 后@f/t/wg/ng ， @xs 从@pg/wg
体制@ng 上@f/vg/d/j/wg/hm 解决@vg 了@el/y/vg 这个@rn 问题@ng 。 @xs //
2 原来@b/wg 的@ed/y/ng/a 县@ng 供销社@ng 改@vg 为/v1@vg/pg/pe 基层@ng
供销社@ng 的@ed/y/ng/a 联合@vg 社@ng 。@xs //
3 集体@ng/b 商业@ng 在@pz/vg/wg/pg 我@rp 国@ng/j 现@t 阶段@ng ， @xs 包括@vg 由@pg
原来@b/wg 的@ed/y/ng/a 小@a/hg/wg/b 商@ng 小贩@ng 组织@vg/ng 起来@d/vv/vg
的@ed/y/ng/a 合作@vg/ng 店@ng 、 @xp/xs 组@qnq/ng ； @xs //

(b)

Figure 4. Examples of tagged sentences in the Thai corpus (a) and the Chinese corpus (b). Here, the strings, such as 'NPRP', 'EAFF', 'mx', 'qnm' that follow each word and are separated by the symbol '@' or '/', are all possible part-of-speech tags for the word. The string that follows the symbol '@' is the correct part-of-speech tag for the word.

38 kinds of part-of-speech tags are used, while the total number of part-of-speech tags in Thai is 47 [3]. The Chinese corpus contains 5,603 tagged sentences. The sentences in the training set and the first test set (Test-1) are collected from articles in the business field, while the sentences in the second test set (Test-2) are collected from articles in the military field. The number of part-of-speech tags used in the Chinese corpus is 56, while the total number of part-of-speech tags in Chinese is 65 [24]. The data distributions of these two corpora are shown in Table 1. In the experiments, only the ambiguous words in the corpora are used as training and test data.

4.2. Representations of Inputs and Outputs

Each of the training inputs X is organized as a vector consisting of seven $(l+r+1=3+3+1)$ sub-vectors as follows:

$$X = [\boldsymbol{x}_{t-3}, \boldsymbol{x}_{t-2}, \boldsymbol{x}_{t-1}, \boldsymbol{x}_t, \boldsymbol{x}_{t+1}, \boldsymbol{x}_{t+2}, \boldsymbol{x}_{t+3}]^T, \quad (20)$$

where $\boldsymbol{x}_t$ is the sub-vector associated with the central word to be tagged, and $\boldsymbol{x}_{t\pm i}$ for $i = 1, 2$, and 3 are the sub-vectors encoding the other six words (three on either side of the central word) that provide a context for the tagging decision.

Each of the seven sub-vectors $\boldsymbol{x}_i$ for $i = t-3, t-2, \ldots, t+3$ consists of M elements which are used to represent M different kinds of part-of-speech tags, and is represented in the form

$$\boldsymbol{x}_i = [x_{i1}, x_{i2}, \ldots, x_{iM}]^T. \quad (21)$$

In the simulations below, M is set to 46 for the Thai tagging problem and 59 for the Chinese tagging problem according to the actual numbers of input

Table 1. The Thai and Chinese corpora.

Corpus	Data set	No. of tagged sentences	No. of words	No. of ambiguous words	Ambiguity rate (%)
Thai	Training	8322	124331	22311	18
	Test	2130	34544	6717	19
Chinese	Training	2904	70900	30211	43
	Test-1	1622	36631	15383	42
	Test-2	1077	27192	11098	41

dimensions in the Thai corpus and the Chinese corpus. Thus, the dimensions of the training inputs for learning the Thai corpus and the Chinese corpus are $46 \times 7 = 322$ and $59 \times 7 = 413$, respectively.

The elements of $\boldsymbol{x}_i$ for $i = t-3, t-2, \ldots, t+3$ are determined by

$$x_{ip} = \text{Prob}(\lambda_p \mid w_i), \tag{22}$$

where $\text{Prob}(\lambda_p \mid w_i)$ is the probability that the part-of-speech tag λ_p might be assigned to word w_i. It is estimated from the training data as

$$\text{Prob}(\lambda_p \mid w_i) = \frac{|\lambda_p, w_i|}{|w_i|}, \tag{23}$$

where $|\lambda_p, w_i|$ is the number of times that λ_p might be assigned to w_i, and $|w_i|$ is the total number of times that w_i might appear in the training set.

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

$$x_{ip} = \begin{cases} \dfrac{1}{\gamma_i} & \text{if } \lambda_p \text{ is a candidate} \\ 0 & \text{otherwise,} \end{cases} \tag{24}$$

where γ_i is the total number of part-of-speech tags that can be assigned to w_i.

Suppose that grandmother cells are used as the output representation for M^3 networks, that is, K output units can only represent $K+1$ classes of patterns at most, and one and only one output unit is active at a time [25]. The actual output of the M^3 network is judged by the following rule. If the position of the largest element of the actual output is the same as that of the desired output, then the output is considered to be a correct output. Otherwise, it is an incorrect output.

4.3. Decomposition of the Tagging Problems

The problem of part-of-speech tagging of Thai can be regarded as a 38-class pattern classification problem. The number of training and test data belonging to each of the classes are shown in Table 2. According to (8), this tagging problem can be divided into $\binom{38}{2} = 703$ two-class subproblems. From Table 2 and the definition of the two-class subproblems, we see that the number of training data for the smallest two-class subproblem $T_{36,38}$ is only three, while the number of

Table 2. Number of training and test data belonging to each of 38 classes in the Thai corpus.

	No. of ambiguous words			No. of ambiguous words	
Class	Training	Test	Class	Training	Test
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

training data for the largest two-class subproblem $T_{6,8}$ is 6,205. Although these two-class subproblems are smaller than the original problem, they are not adequate for massively parallel computation and efficient learning with the back-propagation algorithm due to the following reasons. (1) The two-class subproblems are rather 'load imbalanced'. Since the speed of parallel learning is limited by the speed of the slowest subproblem, the unduly burdening of even a single subproblem can dramatically degrade the overall performance of the learning; (2) Some of the two-class subproblems are still too big for training; (3) Some of the two-class subproblems are very imbalanced, i.e., the training set contains many more data of the 'dominant' class than the other 'subordinate' class. For example, the two-class subproblem $\mathcal{T}_{8,38}$ is an imbalanced problem. It has been shown that the standard back-propagation algorithm converges very slowly for imbalanced problems [26]. In order to speed-up learning, we should further decompose each of the bigger two-class subproblems into a number of relatively smaller and simpler two-class subproblems.

By using the decomposition method, each of the training input sets for the bigger two-class subproblems is *randomly* divided into a number of relatively smaller subsets in the form as defined by (9). Table 3 shows the number of subsets belonging to each of the following larger classes: $\mathcal{C}_1, \mathcal{C}_3, \mathcal{C}_4, \mathcal{C}_5, \mathcal{C}_6, \mathcal{C}_8, \mathcal{C}_9, \mathcal{C}_{11}, \mathcal{C}_{12}, \mathcal{C}_{13}, \mathcal{C}_{15}$, and $\mathcal{C}_{18}$. For example, the training input set for class $\mathcal{C}_4$ is randomly divided into eight subsets, each of which has just 300 pieces of data. Note that the number of subsets belonging to each of the other 26 smaller classes is one. After performing this partition, the original tagging problem is divided into

$$\sum_{i=1}^{37} \sum_{j=i+1}^{38} N_i \times N_j = 3{,}893 \tag{25}$$

Table 3. Number of subsets belonging to each of 12 larger classes in the Thai corpus.

Class	No. of subsets	Class	No. of subsets
N_1	10	N_9	5
N_3	5	N_{11}	2
N_4	8	N_{12}	2
N_5	5	N_{13}	2
N_6	10	N_{15}	3
N_8	10	N_{18}	3

much smaller and simpler two-class subproblems. It should be noted that decomposing the tagging problem into 3,893 two-class subproblems is not unique in general because of random partition of the training input sets. Among the 3,893 two-class subproblems, the number of training data of the largest subproblem $\mathcal{T}_{10,19}$ is only 957. Clearly, it is far smaller than the original problem.

Following the similar way mentioned above, the problem of part-of-speech tagging of Chinese can be regarded as a 56-class pattern classification problem. Table 4 shows the number of training and test data belonging to each of the 56 classes for the Chinese tagging

Table 4. Number of training and test data belonging to each of 56 classes in the Chinese corpus.

	No. of ambiguous words				No. of ambiguous words		
Class	Training	Test-1	Test-2	Class	Training	Test-1	Test-2
L_1	3712	2015	1574	L_{29}	148	30	73
L_2	1642	871	478	L_{30}	131	82	45
L_3	4146	2005	1069	L_{31}	3	1	0
L_4	61	36	28	L_{32}	94	61	18
L_5	3402	1716	1159	L_{33}	29	21	10
L_6	1261	521	549	L_{34}	115	83	193
L_7	1439	762	611	L_{35}	157	89	126
L_8	856	441	258	L_{36}	283	139	137
L_9	1973	993	505	L_{37}	183	81	31
L_{10}	2337	1046	1131	L_{38}	49	22	12
L_{11}	1	5	40	L_{39}	190	121	60
L_{12}	898	447	189	L_{40}	250	141	103
L_{13}	382	204	209	L_{41}	690	326	141
L_{14}	14	4	1	L_{42}	4	3	25
L_{15}	340	199	202	L_{43}	203	105	62
L_{16}	145	64	38	L_{44}	78	22	17
L_{17}	758	471	261	L_{45}	12	13	1
L_{18}	1000	538	434	L_{46}	103	48	88
L_{19}	465	274	153	L_{47}	30	17	5
L_{20}	106	67	84	L_{48}	68	33	38
L_{21}	646	343	250	L_{49}	2	1	3
L_{22}	1	0	0	L_{50}	118	65	8
L_{23}	108	57	66	L_{51}	7	3	2
L_{24}	346	170	70	L_{52}	26	19	8
L_{25}	648	283	379	L_{53}	61	17	20
L_{26}	251	154	41	L_{54}	74	44	42
L_{27}	14	7	2	L_{55}	41	23	1
L_{28}	102	75	45	L_{56}	7	5	2

Table 5. Number of subsets belonging to each of 16 larger classes in the Chinese corpus.

Class	No. of subsets	Class	No. of subsets
N_1	12	N_{10}	8
N_3	5	N_{12}	3
N_4	14	N_{17}	3
N_5	11	N_{18}	3
N_6	4	N_{19}	2
N_7	5	N_{21}	2
N_8	3	N_{25}	2
N_9	7	N_{42}	2

problem. Each of the training input sets belonging to 16 larger classes is randomly broken down into a number of smaller subsets, as shown in Table 5. According to (12), the original Chinese tagging problem is divided into

$$\sum_{i=1}^{55} \sum_{j=i+1}^{56} N_i \times N_j = 7{,}572 \tag{26}$$

much smaller and simpler two-class subproblems.

4.4. Design of Network Modules

After task decomposition, each of the two-class subproblems can be treated as a completely independent problem. Therefore, any pattern classification technique can be used to learn the two-class subproblems. In the experiments, we selected three-layer perceptrons as network modules to solve the tagging problems.

To achieve satisfactory learning accuracy and good generalization performance, it is necessary to select a suitable network architecture. This is because if the network selected is too small, it cannot learn a given task with satisfactory accuracy, but if the network is too large, it will lead to *overtraining* [11] and poor generalization performance. As discussed earlier, for conventional neural networks, such as multilayer perceptrons, it is difficult for the user to select an appropriate network architecture for a given problem, especially for large-scale, complex problems.

One of the most important features of the M^3 network model is that it allows us to easily design a *small* network for a given problem according to the following procedure.

Step 1: Assign the smallest network modules as initial network modules to all the two-class subproblems.
Step 2: Train the network modules up to a given number of epochs.
Step 3: If all of the modules or a prescribed number of modules satisfy a given error tolerance, then stop the procedure. Otherwise, perform the following steps.
Step 4: Add one or more hidden units to each of the unconverged network modules.
Step 5: Retrain the extended network modules up to a given number of epochs, and go back to Step 3.

It is worth noting that the reason why the design procedure above can work efficiently is that each of the network modules are completely independent each other in the learning phase and only the unconverged network modules need to be retrained.

Technically, we can select perceptrons instead of multilayer perceptrons as initial network modules in the above procedure because some of the two-class subproblems might be linearly separable problems when each of the subproblems is very small. In the simulations below, as the initial network modules, we choose three-layer perceptrons with one hidden unit and two hidden units for the problems of tagging Chinese text and Thai text, respectively.

4.5. Learning Algorithm

In the experiments, all of the problems were learned using the back-propagation algorithm [14] in a *batch mode* [13]. The momentums were all set to 0.9, and the learning rates were selected as 0.25 or 0.1. Training is stopped when a given number of epochs is reached or the following criterion is satisfied:

$$\frac{1}{L} \sum_{i=1}^{L} \sum_{j=1}^{K} |d_{ij} - y_{ij}| \leq \delta \tag{27}$$

where d_{ij} and y_{ij} are the desired and actual outputs of the jth output unit associated with the ith training data, respectively, L is the number of training data, K is the number of output units, and δ is a real number which denotes the error tolerance. In the simulations, δ was selected as 0.005 or 0.002. All of the simulations were performed on a Fujitsu VPP700E vector parallel computer.

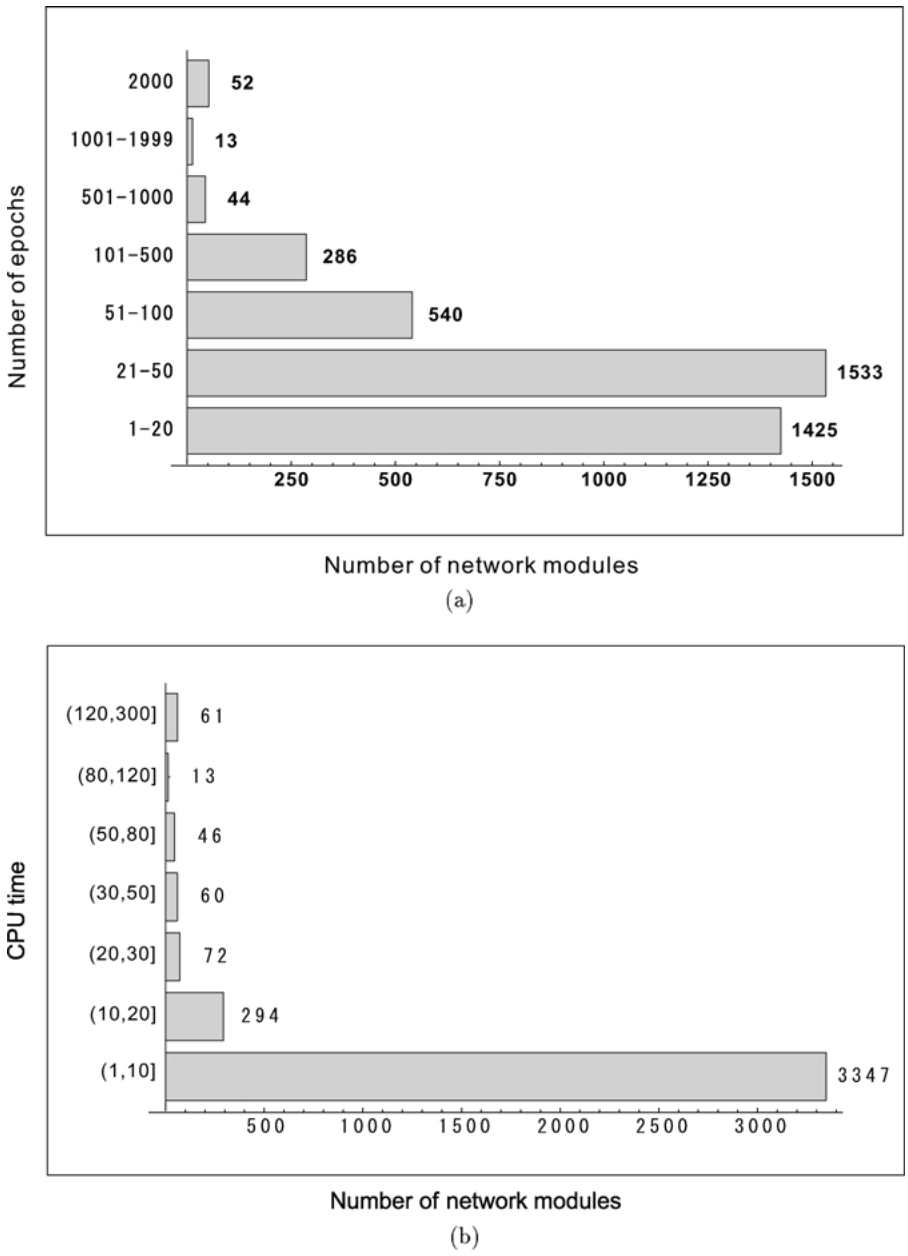

Figure 5. Summary of 3,893 trained network modules for learning the Thai corpus. (a) distributions of trained modules measured by epochs, and (b) distributions of trained modules measured by CPU time in seconds.

4.6. Learning Results

4.6.1. Thai Case. At the beginning of learning, three-layer perceptrons with 322 input, two hidden, and one output units were selected as the initial network modules to learn the corresponding two-class subproblems. The learning rate was set to 0.25. The error tolerance δ was set to 0.005 and the maximum number of epochs was set to 2,000. Figures 5(a) and (b) shows the distributions of the 3,893 trained network modules measured by the number of epochs and CPU times (s.), respectively. From Fig. 5(b), we see that over 93% of the network modules converged within 20 seconds, and the number of unconverged modules was only 52. After the

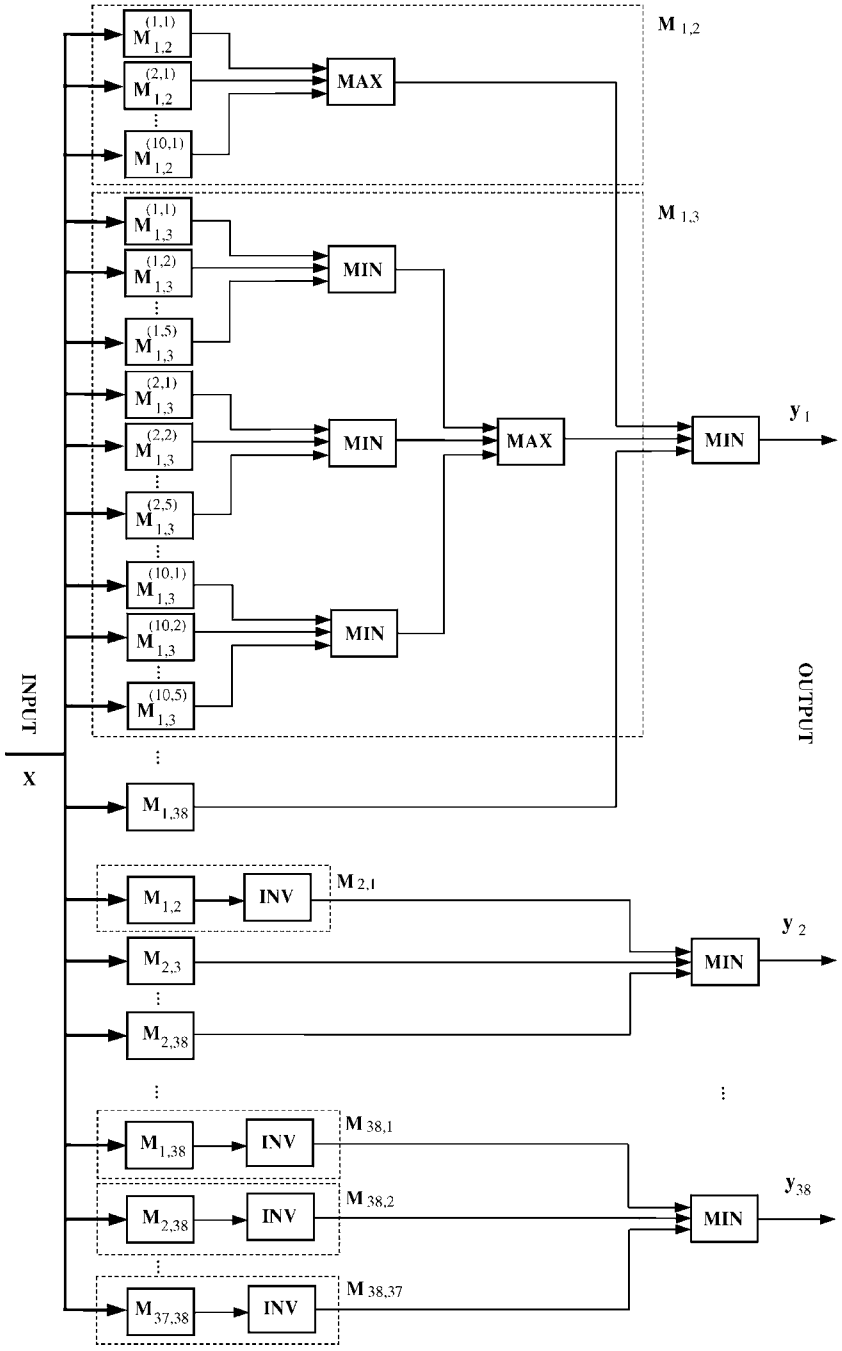

Figure 6. The M^3 network for tagging Thai text. Note that $\mathbf{M}_{1,2}$ and $\mathbf{M}_{1,3}$ are plotted in detail, and the other modules are roughly illustrated.

learning, the 3,893 individual trained network modules were integrated into the M^3 network shown in Fig. 6. Looking at Fig. 6, we can see that the proposed tagging model has two attractive features. (1) The response time of the system is almost independent from the number of modules, that is, the system response time is almost independent of the problem size; (2) The system might be easily implemented in hardware because of its modular structure. It should be noted that this integration is fully guided by two simple module combination laws

Table 6. Performance of the min-max modular neural networks for tagging Thai.

δ	No. of hidden units	No. of converged modules	No. of unconverged modules	Accuracy rate (%)	
				Training	Test
0.005	2	3,841	52	98.4	92.6
0.005	2, 4	3,882	11	98.5	92.9
0.005	2, 4, 6	3,890	3	**99.5**	**93.4**
0.002	2	3,738	155	98.7	93.0
0.002	2, 4	3,797	96	99.3	93.1
0.002	2, 4, 6	3,852	41	**99.8**	**94.2**

[16, 17] and it is performed automatically. The learning accuracy and generalization performance of the M^3 network, i.e., the success rates on training and test data, are shown in the first row of Table 6.

To improve the learning accuracy of the M^3 network, the 52 unconverged subproblems were learned again by a bit big three-layer perceptrons, each of which had four hidden units. After retraining, the number of unconverged network modules was reduced to eleven. In this stage, both the learning accuracy and generalization performance of the M^3 network had been improved slightly. To achieve complete learning, eleven three-layer perceptrons with six hidden units were selected to learn the unconverged two-class subproblems. After this round of training, the learning accuracy reached 99.5% and the tagging accuracy was about 93.4%.

In order to illustrate the ability of the M^3 network model to implement complete learning, the 3,893 subproblems were learned again by selecting a smaller error tolerance. In this simulation, the error tolerance was set to 0.002, and the other parameters were the same as those used in the preceding simulation. All of the results are also shown in Table 6.

4.6.2. Chinese Case. Following the same procedure as mentioned above, three rounds of training were performed to learn the Chinese corpus. 7,572 three-layer perceptrons were used to learn the corresponding 7,572 two-class subproblems. The error tolerance was set to 0.002, and the other parameters used in this simulation were the same as those in solving the tagging problem of Thai. Figures 7(a) and (b) show the convergence of the 7,572 network modules after the first round of training. All of the learning results are shown in Table 7, where the maximum CPU time means the longest time required for training an individual network module in each round of training. From this table, we see that only 344 seconds was required for the complete learning of over 96% of the 7,572 two-class subproblem, provided that all of the two-class subproblems are learned in parallel. In addition, the performance of the M^3 network also demonstrates that the overtraining problem in M^3 networks can be avoided by using small network modules.

5. Comparison Studies

In this section, we experimentally compare the proposed method with several existing tagging approaches including hidden Markov models [7], multilayer perceptrons [8, 13], and neuro-taggers [11, 12, 27]. In all of the comparison experiments, the same Thai and Chinese corpora as mentioned in the preceding sections were used. For neuro-taggers, three-layer perceptrons with p input, $\lceil p/2 \rceil$ hidden, and q output units were

Table 7. Performance of min-max modular neural networks for tagging Chinese.

Maximum CPU time (s.)	No. of converged modules	No. of unconverged modules	No. of hidden units	Accuracy rate (%)		
				Training	Test-1	Test-2
344	7274	298	1	97.2	89.0	86.3
373	7338	234	1, 2	98.7	90.3	88.1
437	7386	186	1, 2, 4	**98.8**	**90.4**	**88.1**

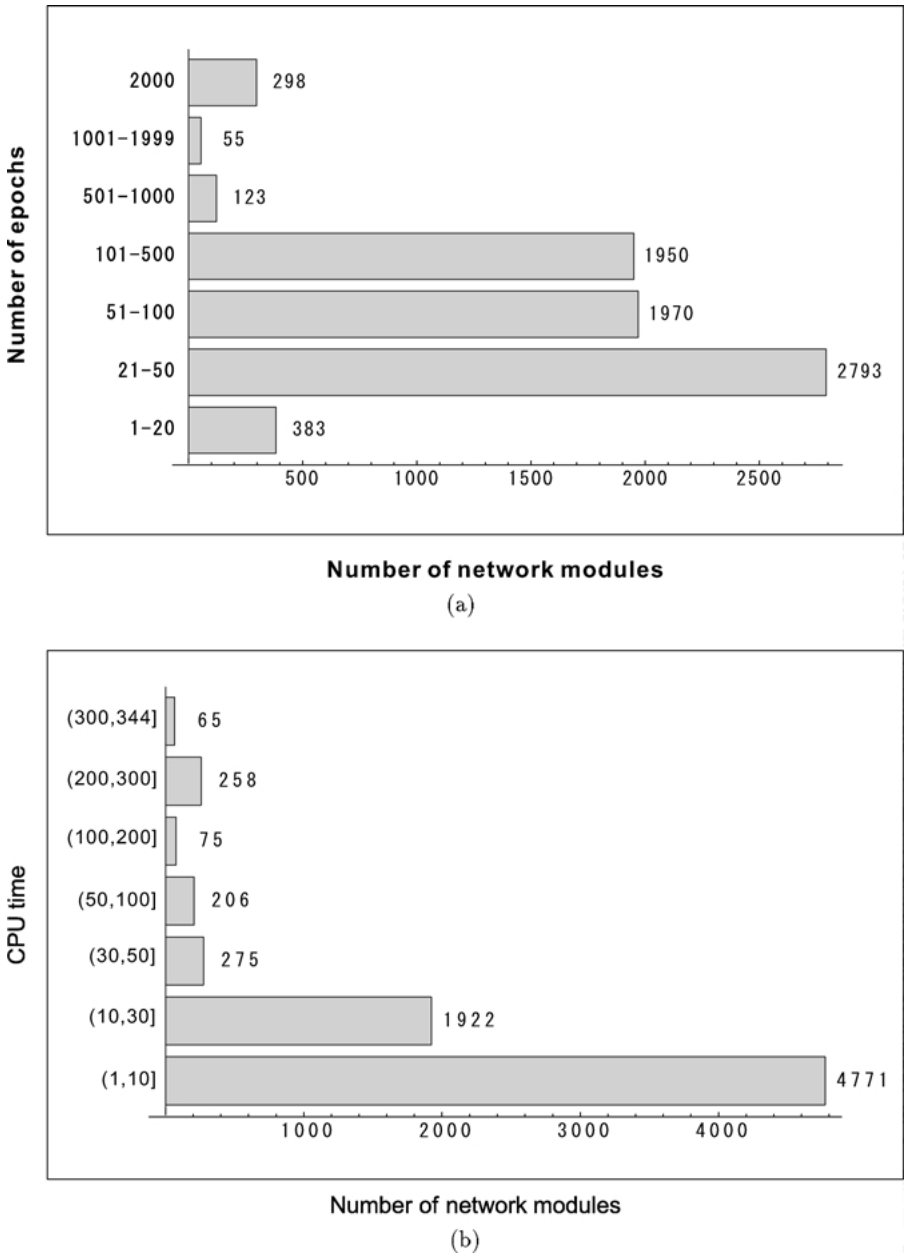

Figure 7. Summary of 7,572 trained network modules for learning the Chinese corpus. (a) distributions of trained modules measured by epochs, and (b) distributions of trained modules measured by CPU time in seconds.

used, where p and q were set to 329 and 47 for tagging Thai text [11], and 441 and 63 for tagging Chinese text [1], respectively.

5.1. Overtraining

To examine overtraining in multilayer-perceptron based tagging systems, two basic tagging models were simulated. One is the conventional three-layer perceptron [13], and the other is the single neuro-tagger [11], which is a simpler version of the multiple neuro-tagger and consists of only one multi-layer perceptron. The difference between the single neuro-tagger and the three-layer perceptron is that the following techniques, as described earlier are used in the single neuro-tagger. (1) Each element of the inputs is weighted with

Table 8. Performance of the single neuro-tagger for tagging Thai text.

	Accuracy rate (%)	
δ	Training	Test
0.005	95.8	**93.0**
0.004	97.0	92.8
0.003	**97.4**	92.6

information gain; (2) In tagging phase, instead of using the original inputs directly, each of the input elements corresponding to the left words in a sentence is replaced by its part-of-speech tag that has already been assigned by the tagging system.

Table 8 shows the generalization performance of the single neuro-tagger for tagging Thai text under three different error tolerance values. From this table, we see that the single neuro-tagger encounters the overtraining problem because the generalization performance decreased when the learning accuracy was improved.

Table 9 shows the performance of conventional three-layer perceptrons with various numbers of hidden units for learning the Chinese corpus. Since our computing source was limited, the maximum number of epochs was set to 300 for each of the simulations. We can see that the largest network with 413 hidden units beats all the other networks in learning accuracy. The network with 53 hidden units, however, had the best generalization performance for both test sets. This result indicates that the conventional multilayer perceptrons also encountered the overtraining problem.

From the above simulation results, we can see that all of the multilayer-perceptron based tagging systems might encounter this overtraining problem. However, how to design a suitable network that avoids this overtraining is still an unsettled problem. On the contrary, the simulation results shown in Tables 6 and 7 demonstrate that the generalization performance of the proposed tagging model had been improved gradually with the increase of learning accuracy in all the simulations. That is, no overtraining occurred. Therefore, we see that the tagging method based on the M^3 network model might avoid the overtraining problem by means of the design procedure presented in Section 4.4.

Table 9. Performance of three-layer perceptrons for tagging Chinese text.

No. of hidden units	CPU time (s.)	Final error	Accuracy rate (%)		
			Training	Test-1	Test-2
13	6298	0.0073	85.0	80.4	78.0
26	6981	0.0043	93.4	87.6	85.4
53	10056	0.0020	96.7	**89.7**	**87.2**
103	26725	0.0013	98.1	88.7	86.2
207	28454	0.0011	98.5	88.4	86.0
310	41239	0.0012	98.6	88.9	86.6
413	59051	0.0012	**98.6**	89.6	86.5

5.2. *Training Time*

To demonstrate the merit of parallel learning used in the proposed tagging method, three-layer perceptrons with various number of hidden units were simulated on the Chinese corpus. The number of hidden units, the training time, the final error, and the performance for each of the networks are shown in Table 9.

Comparing Table 7 with Table 9, we see that our method is about 14 times faster than the three-layer perceptrons in obtaining about the same learning accuracy and generalization performance, providing that learning in our method was performed in parallel. In practice, for multilayer-perceptron-based tagging models, we need to train several networks that have different numbers of hidden units in order to obtain a good generalization performance. Therefore, longer training time might be required.

It has been shown theoretically that learning in multilayer perceptrons is NP-complete [28, 29]. That is, training multilayer perceptrons become intractable as the problem size becomes larger. Experience has also shown that the training process for multilayer perceptrons can be computationally expensive, especially for larger problems with high-dimensional inputs or large data sets [13, 17]. In contrast, by using the power of massively parallel computation, the training time required by our method will not grow with an increase of problem sizes. The simulation results show that our method can scale-up to large-scale problems [18].

5.3. *Generalization*

To compare the generalization performance of the proposed tagging model with existing tagging systems,

Table 10. Performance of different methods for tagging Thai text.

Method	Accuracy rate (%)
Base-line	83.6
HMM	89.1
Single neuro-tagger	93.0
Multiple neuro-tagger	94.3
Elastic neuro-tagger	94.4
M^3	**94.2**

Table 11. Performance of different methods for tagging Chinese text.

	Accuracy rate (%)	
Method	Test-1	Test-2
Base-line	88.1	86.4
HMM	89.9	87.3
MLP	89.7	87.2
Single neuro-tagger	90.1	88.1
Multiple neuro-tagger	91.4	89.2
Elastic neuro-tagger	91.7	89.0
M^3	**90.4**	**88.1**

several experiments were performed on both the Thai and Chinese corpora. The simulation results are shown in Tables 10 and 11. From Table 10, we see that the generalization performance (94.2%) of our model for tagging Thai text is better than statistical models and single neuro-tagger, and approximated the best one (94.4%) obtained by the elastic neuro-tagger. For the tagging problem of Chinese, we can obtain the similar conclusion by examining the results shown in Table 11.

It is important to emphasize that the techniques used in the neuro-taggers were not introduced to our tagging model in the simulations due to the limited computing source. From the simulation results, we can see that the key to increasing the generalization performance of the multiple neuro-tagger and the elastic neuro-tagger is the third technique described earlier, i.e., variant lengths of contexts for tagging are used based on the longest context priority. It is our conjecture that the proposed tagging model might beat both the multiple neuro-tagger and the elastic neuro-tagger in generalization performance when the same techniques are used.

6. Conclusions

In this paper we have presented a novel tagging method based on a modular neural-network model. The advantages of this method over the multilayer-perceptron based approaches are its high modularity, parallelism, and scalability. We have demonstrated that the method is superior to HMM models and multilayer-perceptron-based systems in both learning accuracy and generalization performance. We believe that the proposed method might provide us with an efficient framework for solving large-scale tagging problems. By using the proposed method, we have began doing experiments on a big Chinese corpus which contains 13,111 sentences and 139,397 ambiguous words. As to future work, we would like to apply our method to tagging English text.

Acknowledgments

The authors would like to thank the guest Editors and two anonymous reviewers for their useful comments.

Note

1. This Chinese corpus was developed by M.S. Sun of Tsinghua University, Beijing, China.

References

1. 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, 1998, pp. 200–207.
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. 543–565, 1994.
3. T. Charoenporn, V. Sornlertlamvanich, and H. Isahara, "Building a large Thai text corpus—part of speech tagged corpus: ORCHID," in *Proc. Natural Language Processing Pacific Rim Symposium 1997*, Phuket, Thailand, 1997, pp. 509–512.
4. B. Merialdo, "Tagging English text with a probabilistic model," *Computational Linguistics*, vol. 20, no. 2, pp. 155–171, 1994.
5. R. Weischedel, M. Meteer, R. Schwartz, L. Ramshaw, and J. Palmucci, "Coping with ambiguity and unknown words through probabilistic models," *Computational Linguistics*, vol. 19, no. 2, pp. 359–382, 1993.
6. E. Charniak, *Statistical Language Learning*, MIT Press: Cambridge, MA, 1993.
7. C.D. Manning and H. Schutze, *Foundations of Statistical Natural Language Processing*, MIT Press: Cambridge, MA, 1999.
8. J. Benello, A.W. Mackie, and J.A. Anderson, "Syntactic category disambiguation with neural networks," *Computer Speech and Language*, vol. 3, pp. 203–217, 1989.

9. M. Nakamura, K. Maruyama, T. Kawabata, and K. Shikano, "Neural network approach to word category prediction for English texts," in *Proc. COLING'90*, Helsinki University, 1990, pp. 213–218.
10. H. Schmid, "Part-of-speech tagging with neural networks," in *Proc. COLING'94*, Kyoto, Japan, 1994, pp. 172–176.
11. Q. Ma and H. Isahara, "A multi-neuro tagger using variable lengths of contexts," in *Proc. COLING-ACL'98*, Montreal, 1998, pp. 802–806.
12. Q. Ma, K. Uchimoto, M. Murata, and H. Isahara, "Hybrid neuro and rule-based part of speech taggers," in *Proc. COLING-2000*, Saarbrücken, 2000, pp. 509–515.
13. S. Haykin, *Neural Networks*, 2nd edn., Prentice-Hall, Inc., 1999.
14. 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*, edited by D.E. Rumelhart, J.L. McClelland, and PDP Research Group, MIT Press: Cambridge, MA, vol. 1, 1986, pp. 318–362.
15. J. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann: San Mateo, CA, 1993.
16. B.L. Lu and M. Ito, "Task decomposition based on class relations: A modular neural network architecture for pattern classification," in *Biological and Artificial Computation: From Neuroscience to Technology, Lecture Notes in Computer Sciences*, edited by J. Mira, R. Moreno-Diaz, and J. Cabestany, Springer-Verlag: New York, vol. 1240, 1997, pp. 330–339.
17. 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.
18. B.L. Lu and M. Ichikawa, "Emergence of learning: An approach to coping with NP-complete problems in learning," in *Proc. IJCNN'2000*, Como, Italy, 2000, July 24–27, vol. 4, pp. 159–164.
19. N.J. Nilsson, *Learning Machines: Foundations of Trainable Pattern Classifying Systems*, McGraw-Hill: New York, 1965; reissued as *The Mathematical Foundations of Learning Machines*, Morgan Kaufmann, San Mateo, CA, 1990.
20. M.I. Jordan and R.A. Jacobs, "Hierarchical mixtures of experts and the EM algorithm," *Neural Computation*, vol. 6, pp. 181–214, 1994.
21. C.Y. Baldwin and K.B. Clark, *Design Rules: The Power of Modularity*, vol. 1, MIT Press: Cambridge, MA, 2000.
22. J.H. Friedman, "Another approach to polychotomous classification," Technical Report (ftp://stat.stanford.edu/pub/friedman/poly.ps.Z), Stanford University, 1996.
23. G.S. Almasi and A. Gottlieb, *Highly Parallel Computing*, 2nd edn., The Benjamin/Cummings Publishing Company, Inc., 1994.
24. M.S. Sun, "Design of Chinese taggers," *Technical Report*, Tsinghua University, 1996, in Chinese.
25. J.A. Anderson, *An Introduction to Neural Networks*, MIT Press: Cambridge, MA, 1995.
26. R. Anand, K.G. Mehrota, C.K. Mohan, and S. Ranka, "An improved algorithm for neural network classification of imbalanced training sets," *IEEE Trans. Neural Networks*, vol. 4, pp. 962–973, 1993.
27. Q. Ma, K. Uchimoto, M. Murata, and H. Isahara, "Elastic neural networks for part of speech tagging," in *Proc. IJCNN'99*, Washington DC, 1999, pp. 2991–2996.
28. J.S. Judd, *Neural Network Design and the Complexity of Learning*, MIT Press: Cambridge, MA, 1990.
29. A.L. Blum and R.L. Rivest, "Training a 3-node neural network is NP-complete," *Neural Networks*, vol. 5, pp. 117–127, 1992.

Bao-Liang Lu received the B.S. degree in instrument and control engineering from University of Qingdao University of Science and Technology, China, in 1982, the M.S. degree in computer science and engineering from Northwestern Polytechnical University, China, in 1989, and the Dr. Eng. degree in electrical engineering from Kyoto University, Japan, in 1994. From 1982 to 1986, he was the University of Science and Technology at Qingdao. From April 1994 to March 1999, he was a Frontier Researcher at the Bio-Mimetic Control Research Center, the Institute of Physical and Chemical Research (RIKEN), Japan. From April 1999 to August 2002, he was a Research Scientist at the RIKEN Brain Science Institute. Currently, he is a Professor in the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China. His research interests include brain-like computers, neural computation, machine learning, pattern recognition, and natural language processing. He is a member of INNS and a senior member of the IEEE.

Qing Ma received his B.S. degree in electrical engineering from Beijing University of Aeronautics and Astronautics, China, in 1983, and the M.S. and Dr. Eng. degree in computer science from University of Tsukuba, Japan, in 1987 and 1990 respectively. He worked in Ono Sokki Co., Ltd., Japan from 1990 to 1993 and worked in the Communications Research Laboratory, Japan from 1993 to 2003, as a Senior Researcher. Currently, he is a Professor in the Department of Applied Mathematics and Informatics, Faculty of Science and Technology, Ryukoku University. His research interests include neural networks, knowledge representation, and natural language processing. He is a member of the Japanese Neural Network Society, the Association of Natural Language Processing, and the Institute of Electronics, Information and Communication Engineers of Japan.

Michinori Ichikawa received the Dr. Eng. degree in applied physics from University of Tsukuba, Japan, in 1986. From 1986 to 1997, he was with the Electrotechnical Laboratory, where he worked on the mechanism of membrane excitation, optical recording methods for neural activities, and brain-like computers. In 1997, he joined the Brain Science Institute, the Institute of Physical and Chemical Research (RIKEN), as the team leader of the Laboratory for Brain-Operative Device. His research interests include neurophysiology, brain-like computers, robots, and neural computation. Dr. Ichikawa is a member of the Japanese Neural Network Society, the Society for Neuroscience, and the IEEE.

Hitoshi Isahara received his B.E., M.E. degrees in electrical engineering from Kyoto University, Japan, in 1978 and 1980 respectively, and Dr. Eng. degree in electrical engineering from Kyoto University, Japan, in 1995. He worked in Electrotecnical Laboratory, Ministry of International Trade and Industry, Japan, from 1980 to 1995. In 1995, he moved to the Communications Research Laboratory, Ministry of Posts and Telecommunications, Japan, where he was the Chief of the Intelligent Processing Section. He is now working at the Communications Research Laboratory, Independent Administrative Institution, Japan, as the leader of Computational Linguistics Group. His research interests include natural language processing, lexical semantics, and machine translation.