Part of Speech Tagging with Min-Max Modular Neural Networks

Qing Ma, Bao-Liang Lu *, and Hitoshi Isahara
Kansai Advanced Research Center
Communications Research Laboratory, MPT
Kobe, 651-2401, Japan
{qma,isahara}@crl.go.jp
*Lab. for Brain-Operative Device, Brain Science Institute, RIKEN
Wako-shi, 351-0198, Japan
lu@brainway.riken.go.jp

Abstract

Part of speech tagging systems using neural networks have been proposed by Ma, et al. They can tag the untrained data at a practical level of accuracy by training a small Thai corpus with ten thousand order words. The multilayer perceptron (MLP) type of neural networks used, however, was found to converge slowly and took a very long time to train even the above mentioned small amount of training data. This paper presents an alternative method for solving the POS tagging problems with the min-max modular neural network proposed by Lu and Ito. By using this modular neural network, the part of speech tagging problems can be broken down into a number of independent smaller and simpler subproblems, and all of the subproblems can be learned by small network modules in parallel.

1 Introduction

Words are often ambiguous in terms of what part of speech (POS) they serve and POS tagging, which disambiguates them in the context of the sentence, is an essential technique in natural language processing. This technique can be widely used in many areas of information processing including pre-processing for speech synthesis, post-processing for OCR and speech recognition, parser, machine translation, and information retrieval. A large number of POS taggers using rule-based (e.g., [1]), statistical (e.g., [2]), decision tree (e.g., [3]), and neural network (e.g., [4]) models have been proposed so far. These taggers have reached a high level of accuracy partly because of the very large amount of training data used (e.g., in the order of 1,000,000 words for English).

To construct a practical tagger that uses as few training data as possible, POS tagging systems that consists of multiple neural networks [5] or a single neural network with elastic input [6] were proposed by Ma, et al. Both of these systems were more than 94% accurate (counting only the ambiguous words in POSs) when tagging untrained data from a small Thai corpus with 22,311 ambiguous words available for training. This accuracy is far higher than that of the Hidden Markov Model (HMM), a major method used for POS tagging [6]. The multilayer perceptron (MLP) type of neural networks used in the systems, however, was found to be slow to converge and took a long time to train even the above mentioned small amount of training data. It would therefore be difficult to train a large amount of data to further improve tagging accuracy.

This paper shows that this problem may be solved by adopting the module neural network proposed by Lu and Ito called a min-max modular network (or M^3 network for short) [7, 8]. The M^3 network can automatically break down complex learning problems into a number of independent smaller and simpler two-class subproblems which can then be automatically recombined into a solution to the original problems.

2 POS Tagging Problems

In this paper, we suppose there is a lexicon:

$$V = (w^1, w^2, \cdots, w^v), \tag{1}$$

where the POSs that can be served by each word are listed, and there is a set of POSs:

$$\Gamma = (\tau^1, \tau^2, \cdots, \tau^\gamma). \tag{2}$$

Here, v is the number of registered words and γ is the number of types of POSs. This means that unknown words that do not exist in the lexicon are not dealt with. The POS tagging problem therefore is to find a POS τ_t for each target word (the word to be tagged) w_t

0-7803-5731-0/99$10.00 ©1999 IEEE

$(w_t \in V,\ t = 1, \cdots, s)$ in a given sentence $w_1 w_2 \cdots w_s$ by using contexts as follows:

$$W^t \rightarrow \tau_t,\ t = 1, \cdots, s \tag{3}$$

Here, W^t is the word sequence which is centered by the target word and has a length $l + 1 + r$, that is,

$$W^t = w_{t-l} \cdots w_t \cdots w_{t+r}, \tag{4}$$

where $t - l \geq 1$, $t + r \leq s$. Tagging can thus be regarded as a classification problem by replacing the POS with class, which can therefore be handled by using a method like the neural network model.

From the word sequence W^t shown in (4), by fixing the number of both the left and right words in three[1], i.e., $l = r = 3$, the input of the neural networks, denoted by X, can be constructed as follows:

$$X = (\mathbf{x}_{t-3}, \cdots, \mathbf{x}_t, \cdots, \mathbf{x}_{t+3}). \tag{5}$$

When word w is given in position p $(p = t-3, \cdots, t+3)$, element $\mathbf{x}_p$ is a pattern[2] defined as

$$\mathbf{x}_p = (e_{w1}, e_{w2}, \cdots, e_{w\gamma}), \tag{6}$$

where γ is the number of types of POSs. If w is a word that appears in the training data, then each bit e_{wi} is obtained as follows[3]:

$$e_{wi} = Prob(\tau^i | w). \tag{7}$$

Here $Prob(\tau^i|w)$ is the prior probability for τ^i that the word w can be and is estimated from the training data as

$$Prob(\tau^i | w) = \frac{|\tau^i, w|}{|w|}, \tag{8}$$

where $|\tau^i, w|$ is the number of times both τ^i and w appear and $|w|$ is the number of times w appears in all the training data. But if w is a word that does not appear in the training data, then each bit e_{wi} is obtained as follows:

$$e_{wi} = \begin{cases} \frac{1}{\gamma_w} & \text{if } \tau^i \text{ is a candidate} \\ 0 & \text{otherwise,} \end{cases} \tag{9}$$

where γ_w is the number of POSs that the word w can be.

The output of the neural networks, denoted by Y, is defined as follows:

$$Y = (y_1, y_2, \cdots, y_\gamma), \tag{10}$$

provided that the Y is decoded as follows:

$$\tau(w) = \begin{cases} \tau^i & \text{if } y_i \geq 0.5 \ \&\ y_j < 0.5 \text{ for } j \neq i \\ Unknown & \text{otherwise.} \end{cases} \tag{11}$$

where $\tau(w)$ is the tagging result for the word w.

3 POS Tagging with Min-Max Module Networks

3.1 M^3 networks

The central ideas underlying M^3 networks are to use the class relations among the training data and to apply a divide-and-conquer technique to task decomposition and module combination.

With M^3 networks, a K-class classification problem can be decomposed into relatively smaller and simpler $\binom{K}{2}$ two-class subproblems. These two-class subproblems involve discriminating class $\mathcal{C}_i$ from class $\mathcal{C}_j$ where $i = 1, \cdots, K$ and $j = i+1, \cdots, K$, while the existence of training data belonging to the other $K - 2$ classes is ignored. If discriminating the patterns of class $\mathcal{C}_i$ from those of class $\mathcal{C}_j$ remains difficult for the network modules to learn after an initial attempt, the problem can be further divided into as many smaller two-class subproblems as a user requires. Since each of the two-class subproblems can be treated as a completely separate classification problem, all of the two-class subproblems can be learned by different network modules in parallel. Two module combination principles which provide practical guidelines for integrating individual trained modules have been proposed in [7, 8]. After each of the two-class subproblems are learnt by a network module, all of the individual trained modules can be easily integrated into an M^3 network according to the module combination principles. Consequently, a large-scale and complex K-class classification problem can be solved effortlessly and efficiently by learning a series of relatively smaller and simpler two-class subproblems.

3.2 Problem decomposition

The Thai corpus used in this paper contains 10,452 sentences randomly divided into two sets: one with 8,322 sentences for training and another with 2,130 sentences for testing. The training and testing sets used in the computer experiments contain, respectively, 22,311 and 6,717 ambiguous words that serve as more than one POS and are used for training and testing. In Thai, 47 types of POSs are defined [9], i.e., $\gamma = 47$ [Eq. (2)], but 38 types appear in the Thai corpus used in the computer experiments. Since there are 38 kinds of POSs, the POS tagging problem is considered as a 38-class pattern classification problem. For

[1] In Ref. [5, 6], the number of left and right words is variable.

[2] In Ref. [5, 6], the pattern is weighted using information gain (denoted by IG) which can be obtained from the training data using information theory.

[3] There is more information available for constructing the input for the words on the left, because they have already been tagged. In previous systems proposed by Ma, et al., the inputs for the words on the left are constructed using the already tagged results (see details in [5] or [6]).

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	3041	962	L_{20}	4	0
L_2	72	13	L_{21}	76	17
L_3	1444	385	L_{22}	4	7
L_4	2400	701	L_{23}	9	2
L_5	1582	399	L_{24}	57	15
L_6	3008	1011	L_{25}	32	11
L_7	12	0	L_{26}	90	34
L_8	3197	1008	L_{27}	30	5
L_9	1537	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

the 22,311 training and 6,716 testing patterns, their class distributions are shown in Table 1.

According to the task decomposition method [7, 8], the 38-class pattern classification problem can be divided into $\binom{38}{2}$ two-class subproblems. Let $\mathcal{T}_{ij}$ be the training set for a two-class subproblem of discriminating class $\mathcal{C}_i$ from class $\mathcal{C}_j$. The two-class subproblems are defined by

$$\mathcal{T}_{ij} = \{(X_l^{(i)},\ 1-\epsilon)\}_{l=1}^{L_i} \cup \{(X_l^{(j)},\ \epsilon)\}_{l=1}^{L_j} \quad (12)$$
$$\text{for } i = 1, \cdots, 38 \text{ and } j = i+1, \cdots, 38,$$

where $X_l^{(i)} \in \mathcal{X}_i$, $X_l^{(j)} \in \mathcal{X}_j$ and L_i is the number of data of $\mathcal{X}_i$. Here, $\mathcal{X}_i$ and $\mathcal{X}_j$ are the input subsets belonging to class $\mathcal{C}_i$ and class $\mathcal{C}_j$, respectively.

From Table 1 and the definition of two-class subproblems, we see that the number of training data for the smallest two-class subproblem ($\mathcal{T}_{36,38}$) is only 3, and the number of training data for the largest two-class subproblem ($\mathcal{T}_{68}$) 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, 8], each of the large two-class subproblems can be divided into a number of relatively smaller ones in the form

$$\mathcal{T}_{ij}^{(u,v)} = \{(X_l^{(iu)},\ 1-\epsilon)\}_{l=1}^{L_i^{(u)}} \cup \{(X_l^{(jv)},\ \epsilon)\}_{l=1}^{L_j^{(v)}} \quad (13)$$
$$\text{for } u = 1, \cdots, N_i,\ v = 1, \cdots, N_j,$$
$$i = 1, \cdots, 38 \text{ and } j = i+1, \cdots, 38,$$

Table 2: Number of subsets belonging to each of 12 large classes

♯	No. of subsets	♯	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

where $X_l^{(iu)} \in \mathcal{X}_{iu}$, $X_l^{(jv)} \in \mathcal{X}_{jv}$, and N_i is the number of subsets of class $\mathcal{C}_i$. Here, $\mathcal{X}_{iu}$ and $\mathcal{X}_{jv}$ are the *uth* and *vth* input subsets belonging to class $\mathcal{C}_i$ and class $\mathcal{C}_j$, respectively.

Table 2 shows the number of subsets belonging to each of the following large classes: $\mathcal{C}_1$, $\mathcal{C}_3$, $\mathcal{C}_4$, $\mathcal{C}_5$, $\mathcal{C}_6$, $\mathcal{C}_8$, $\mathcal{C}_9$, $\mathcal{C}_{11}$, $\mathcal{C}_{12}$, $\mathcal{C}_{13}$, $\mathcal{C}_{15}$, and $\mathcal{C}_{18}$. The number of subsets belonging to each of the remaining 24 classes is 1. For example, the training data of class $\mathcal{C}_8$ is divided into 8 subsets, each of which has just 300 patterns. After performing the above partitions, the original tagging problem is divided into

$$\sum_{i=1}^{38} \sum_{j=i+1}^{38} N_i \cdot N_j = 3,893 \quad (14)$$

smaller two-class subproblems. Thus, among the 3,893 two-class subproblems, the number of training data of the largest subproblem ($\mathcal{T}_{10,19}^{(1,1)}$) is only 957.

3.3 Parallel learning

An important feature of M^3 networks is that each of these two-class subproblems can be treated as a completely separate classification problem in the learning stage. Consequently, all of the two-class subproblems can be learned in parallel. In the first round of learning, 3,893 three-layer MLPs are selected as network modules to learn the 3,893 two-class problems. Each of the modules has 329 ($= \gamma \times (l+1+r) = 47 \times 7$) input, three hidden and one output units. In the simulation, the conventional backpropagation algorithm [11] 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 5,000. After the first round of learning, about 3,100 network modules had achieved the desired learning accuracy and the remaining 800 did not converge. In accordance with the module combination principles [7, 8], the 3,893 individual trained modules were integrated into an M^3 network shown in Figure 1.

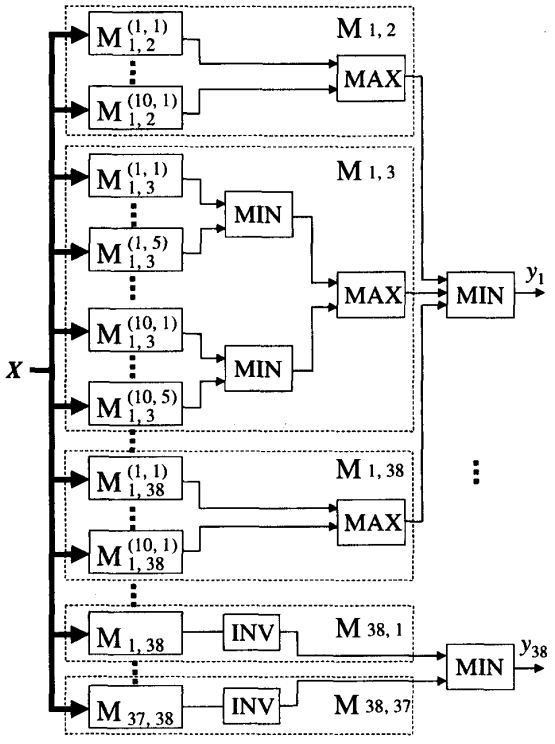

Figure 1: The M^3 Network for part of speech tagging.

Table 3: Accuracy of different tagging methods

Method	No. of Modules	Accuracy (%)	
		Training	Testing
HMM		88.3	89.1
MLP	1	96.7	92.6
Elastic MLP[4]	1	97.4	94.3
M^3 network-1	3,893	98.2	93.2
M^3 network-2	3,893	99.0	93.9

Table 4: Computational complexity of different tagging methods

Method	No. of Data	No. of Iterations	Network size
MLP	22,311	79	$329 \times \frac{329}{2} \times 38$
Elastic MLP	22,311	1055[5]	$141 \times \frac{141}{2} \times 38$
M^3 network-1	957	5000	$329 \times 3 \times 1$
M^3 network-2	699	20,000	$329 \times 6 \times 1$

4 Experimental Results

Table 3 shows the results of experiments comparing the M^3 network and other tagging methods. As shown in the table, the accuracy of the M^3 network, trained in the manner described above (denoted by M^3 network-1) was 98.2% for training data and 93.2% for the testing data. The M^3 network was much more accurate in tagging the training data than were the other tagging methods and more accurate for tagging the testing data than were the HMM and single MLP methods. In order to improve the tagging accuracy for the testing data, the unconverged 800 two-class subproblems were learned again by bigger three-layer MLPs, each of which had 6 hidden units. The learning rates were reduced to 0.011 and the total number of epochs was increased to 20,000. After the second round of learning, only about 90 modules did not converge. In this case (denoted by M^3 network-2), the training data accuracy reached 99.0% and the testing data accuracy reached 93.9%.

Table 4 shows the computational complexity of the M^3 network and the existing neuro taggers in terms of three factors: the number of training data, the number of iterations, and the size of networks. Because the computational complexity directly determines the computational time, the computational time of the M^3 network is therefore much fewer than that of the existing neuro taggers. The data of M^3 network shown in the table is for one module that takes most long computational time, which equals to that of the whole network because of the parallel learning. The data of elastic MLP is for one training stage that takes almost the whole computational time (see details in [6]). The number of iterations of the M^3 network is much larger than that of MLP. It's training accuracy, however, was much higher than that of MLP instead.

5 Conclusion

Although the POS neuro taggers previously proposed by Ma, et al. have already reached a high accuracy when using a small amount of training data with ten thousand order words, they face the *scaling problem*, i.e., the training of neural networks becomes intractable as the problem size becomes too large due to the use of conventional MLP neural networks. This paper has shown that an M^3 network can deal with this problem by dividing a large-scale POS tagging problem into relatively smaller and simpler subproblems. The simulation results indicate that the M^3 network can obtain almost the same generalization performance as the existing neuro taggers. In addition,

[4] Here shows the case of without using IG.

[5] This data is the case of without using IG. The number of iterations is reduced to the half by using IG.

the M^3 network is superior to the existing neuro taggers in convergence speed and training accuracy. We belive that the M^3 networks may provide an effective resolution for large-scale POS tagging problems. Our next work is to increse the order of the amount of the training data and see whether the M^3 network can learn the data well and obtain a better tagging accuracy for untrained data.

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, 1994, pp. 543-565.

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

[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, 1996, pp. 1-14.

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

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

[6] 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.

[7] B. L. Lu and M. Ito, "Task decomposition based on class relations: a modular neural network architecture for pattern classification", *Biological and Artificial Computation: From Neuroscience to Technology, Lecture Notes in Computer Science*, J. Mira, R. Moreno-Diaz and J. Cabestany, Eds., vol. 1240, Springer, 1997, pp. 330-339.

[8] B. L. Lu and M. Ito, "Task decomposition and module combination based on class relations: a modular neural network for pattern classification", accepted for publication in *IEEE Trans. Neural Networks.*

[9] 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,1997, pp. 509-512.

[10] B. L. Lu, M. Ichikawa, and S. Hosoe, "A modular massively parallel learning framework for brain-like computers", to appear in *Proc. of IEEE SMC'99*, October, 1999, Tokyo, Japan.

[11] 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, 1986, pp. 318-362.