A Multi-Neuro Tagger Using Variable Lengths of Contexts

Qing Ma and **Hitoshi Isahara**
Communications Research Laboratory
Ministry of Posts and Telecommunications
588-2, Iwaoka, Nishi-ku, Kobe, 651-2401, Japan
{qma, isahara}@crl.go.jp

Abstract

This paper presents a multi-neuro tagger that uses variable lengths of contexts and weighted inputs (with information gains) for part of speech tagging. Computer experiments show that it has a correct rate of over 94% for tagging ambiguous words when a small Thai corpus with 22,311 ambiguous words is used for training. This result is better than any of the results obtained using the single-neuro taggers with fixed but different lengths of contexts, which indicates that the multi-neuro tagger can dynamically find a suitable length of contexts in tagging.

1 Introduction

Words are often ambiguous in terms of their part of speech (POS). POS tagging disambiguates them, i.e., it assigns to each word the correct POS in the context of the sentence. Several kinds of POS taggers using rule-based (e.g., Brill et al., 1990), statistical (e.g., Merialdo, 1994), memory-based (e.g., Daelemans, 1996), and neural network (e.g., Schmid, 1994) models have been proposed for some languages. The correct rate of tagging of these models has reached 95%, in part by using a very large amount of training data (e.g., 1,000,000 words in Schmid, 1994). For many other languages (e.g., Thai, which we deal with in this paper), however, the corpora have not been prepared and there is not a large amount of training data available. It is therefore important to construct a practical tagger using as few training data as possible.

In most of the statistical and neural network models proposed so far, the length of the contexts used for tagging is fixed and has to be selected empirically. In addition, all words in the input are regarded to have the same relevance in tagging. An ideal model would be one in which the length of the contexts can be automatically selected as needed in tagging and the words used in tagging can be given different relevances. A simple but effective solution is to introduce a multi-module tagger composed of multiple modules (basic taggers) with fixed but different lengths of contexts in the input and a selector (a selecting rule) to obtain the final answer. The tagger should also have a set of weights reflecting the different relevances of the input elements. If we construct such a multi-module tagger with statistical methods (e.g., n-gram models), however, the size of the n-gram table would be extremely large, as mentioned in Sec. 4.4. On the other hand, in memory-based models such as IGtree (Daelemans, 1996), the number of features used in tagging is actually variable, within the maximum length (i.e., the number of features spanning the tree), and the different relevances of the different features are taken into account in tagging. Tagging by this approach, however, may be computationally expensive if the maximum length is large. Actually, the maximum length was set at 4 in Daelemans's model, which can therefore be regarded as one using fixed length of contexts.

This paper presents a multi-neuro tagger that is constructed using multiple neural networks, all of which can be regarded as single-neuro taggers with fixed but different lengths of contexts in inputs. The tagger performs POS tagging in different lengths of contexts based on longest context priority. Given that the target word is more relevant than any of the words in its context and that the words in context may have different relevances in tagging, each

element of the input is weighted with information gains, i.e., numbers expressing the average amount of reduction of training set information entropy when the POSs of the element are known (Quinlan 1993). By using the trained results (weights) of the single-neuro taggers with short inputs as initial weights of those with long inputs, the training time for the latter ones can be greatly reduced and the cost to train a multi-neuro tagger is almost the same as that to train a single-neuro tagger.

2 POS Tagging Problems

Since each input Thai text can be segmented into individual words that can be further tagged with all possible POSs using an electronic Thai dictionary, the POS tagging tasks can be regarded as a kind of POS disambiguation problem using contexts as follows:

$$IPT : (ipt_l_l, \cdots, ipt_l_1, ipt_t, ipt_r_1, \cdots, ipt_r_r) \Rightarrow OPT : POS_t, \quad (1)$$

where ipt_t is the element related to the possible POSs of the target word, $(ipt_l_l, \cdots, ipt_l_1)$ and $(ipt_r_1, \cdots, ipt_r_r)$ are the elements related to the contexts, i.e., the POSs of the words to the left and right of the target word, respectively, and POS_t is the correct POS of the target word in the contexts.

3 Information Gain

Suppose each element, ipt_x ($x = l_i, t,$ or r_j), in (1) has a weight, w_x, which can be obtained using information theory as follows. Let S be the training set and C_i be the *ith class*, i.e., the *ith* POS ($i = 1, \cdots, n$, where n is the total number of POSs). The entropy of the set S, i.e., the average amount of information needed to identify the class (the POS) of an example in S, is

$$info(S) = -\sum_{i=1}^{n} \frac{freq(C_i, S)}{|S|} \times ln(\frac{freq(C_i, S)}{|S|}), \quad (2)$$

where $|S|$ is the number of examples in S and $freq(C_i, S)$ is the number of examples belonging to class C_i. When S has been partitioned to h subset S_i ($i = 1, \cdots, h$) according to the element ipt_x, the new entropy can be found as the weighted sum over these subsets, or

$$info_x(S) = \sum_{i=1}^{h} \frac{|S_i|}{|S|} \times info(S_i). \quad (3)$$

Thus, the quantity of information gained by this partitioning, or by knowing the POSs of element ipt_x, can be obtained by

$$gain(x) = info(S) - info_x(S), \quad (4)$$

which is used as the weight, w_x, i.e.,

$$w_x = gain(x). \quad (5)$$

4 Multi-Neuro Tagger

4.1 Single-Neuro Tagger

Figure 1 shows a single-neuro tagger (SNT) which consists of a 3-layer feedforward neural network. The SNT can disambiguate the POS of each word using a fixed length of the context by training it in a supervised manner with a well-known error back-propagation algorithm (for details see e.g., Haykin, 1994).

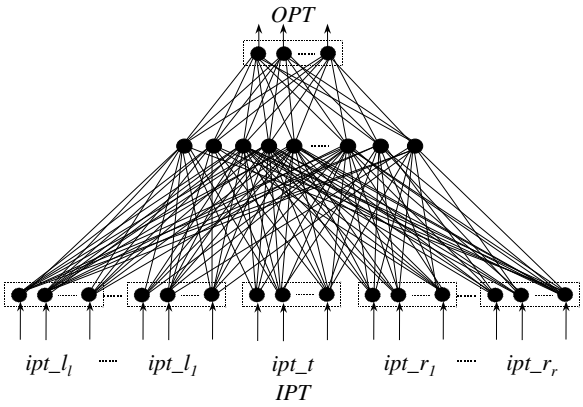

Fig. 1. The single-neuro tagger (SNT).

When word x is given in position y ($y = t$, l_i, or r_j), element ipt_y of input IPT is a weighted pattern defined as

$$ipt_y = w_y \cdot (e_{x1}, e_{x2}, \cdots, e_{xn}), = (I_{x1}, I_{x2}, \cdots, I_{xn}) \quad (6)$$

where w_y is the weight obtained in (5), n is the total number of POSs defined in Thai, and

$I_{xi} = w_y \cdot e_{xi}$ $(i = 1, \cdots, n)$. If x is a known word, i.e., it appears in the training data, each bit e_{xi} is obtained as follows:

$$e_{xi} = Prob(POS_i|x). \tag{7}$$

Here the $Prob(POS_i|x)$ is the prior probability of POS_i that the word x can be and is estimated from the training data as

$$Prob(POS_i|x) = \frac{|POS_i, x|}{|x|}, \tag{8}$$

where $|POS_i, x|$ is the number of times both POS_i and x appear and $|x|$ is the number of times x appears in all the training data. If x is an unknown word, i.e., it does not appear in the training data, each bit e_{xi} is obtained as follows:

$$e_{xi} = \begin{cases} \frac{1}{n_x}, & \text{if } POS_i \text{ is a candidate} \\ 0, & \text{otherwise,} \end{cases} \tag{9}$$

where n_x is the number of POSs that the word x can be (this number can be simply obtained from an electronic Thai dictionary). The OPT is a pattern defined as follows:

$$OPT = (O_1, O_2, \cdots, O_n). \tag{10}$$

The OPT is decoded to obtain a final result RST for the POS of the target word as follows:

$$RST = \begin{cases} POS_i, \text{ if } O_i = 1 \ \& \ O_j = 0 \text{ for } j \neq i \\ Unknown. \quad \text{otherwise} \end{cases} \tag{11}$$

There is more information available for constructing the input for the words on the left because they have already been tagged. In the tagging phase, instead of using (6)-(9), the input may be constructed simply as follows:

$$ipt_l_i(t) = w_l_i \cdot OPT(t - i), \tag{12}$$

where t is the position of the target word in a sentence and $i = 1, 2, \cdots, l$ for $t - i > 0$. However, in the training process the output of the tagger is not correct and cannot be fed back to the inputs directly. Instead, a weighted average of the actual output and the desired output is used as follows:

$$ipt_l_i(t) = w_l_i \cdot (w_{OPT} \cdot OPT(t-i) + w_{DES} \cdot DES), \tag{13}$$

where DES is the desired output

$$DES = (D_1, D_2, \cdots, D_n), \tag{14}$$

whose bits are defined as follows:

$$D_i = \begin{cases} 1 & \text{if } POS_i \text{ is a desired answer} \\ 0. & \text{otherwise} \end{cases} \tag{15}$$

and w_{OPT} and w_{DES} are respectively defined as

$$w_{OPT} = \frac{E_{OBJ}}{E_{ACT}} \tag{16}$$

and

$$w_{DES} = 1 - w_{OPT}, \tag{17}$$

where E_{OBJ} and E_{ACT} are the objective and actual errors, respectively. Thus, the weighting of the desired output is large at the beginning of the training, and decreases to zero during training.

4.2 Multi-Neuro Tagger

Figure 2 shows the structure of the multi-neuro tagger. The individual SNT_i has input IPT_i with length (the number of input elements: $l + 1 + r$) $l(IPT_i)$, for which the following relations hold: $l(IPT_i) < l(IPT_j)$ for $i < j$.

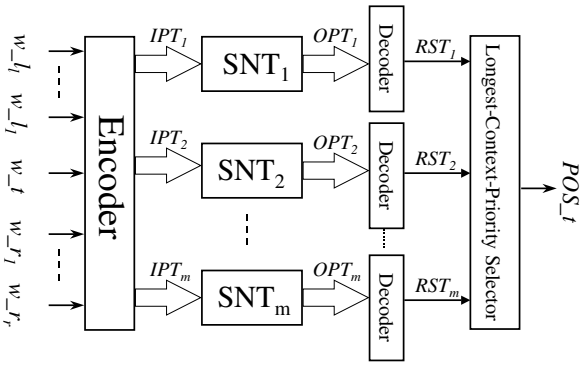

Fig. 2. The multi-neuro tagger.

When a sequence of words ($word_l_l$, $\cdots$, $word_l_1$, $word_t$, $word_r_1$, $\cdots$, $word_r_r$), which has a target word $word_t$ in the center and a maximum length $l(IPT_m)$, is inputed, its subsequence of words, which also has the target word $word_t$ in the center and length $l(IPT_i)$, will be encoded into IPT_i in the same way as described in the previous section. The outputs OPT_i (for

$i = 1, \cdots, m$) of the single-neuro taggers are decoded into RST_i by (11). The RST_i are next inputed into the longest-context-priority selector which obtains the final result as follows:

$$POS_t = \begin{cases} RST_i, & \text{if } RST_j = Unknown \\ & \text{(for } j > i\text{)} \\ & \text{and } RST_i \neq Unknown \\ Unknown. & \text{otherwise} \end{cases} \quad (18)$$

This means that the output of the single-neuro tagger that gives a result being not unknown and has the largest length of input is regarded as a final answer.

4.3 Training

If we use the weights trained by the single-neuro taggers with short inputs as the initial values of those with long inputs, the training time for the latter ones can be greatly reduced and the cost to train multi-neuro taggers would be almost the same as that to train the single-neuro taggers. Figure 3 shows an example of training a tagger with four input elements. The trained weights, w_1 and w_2, of the tagger with three input elements are copied to the corresponding part of the tagger and used as initial values for its training.

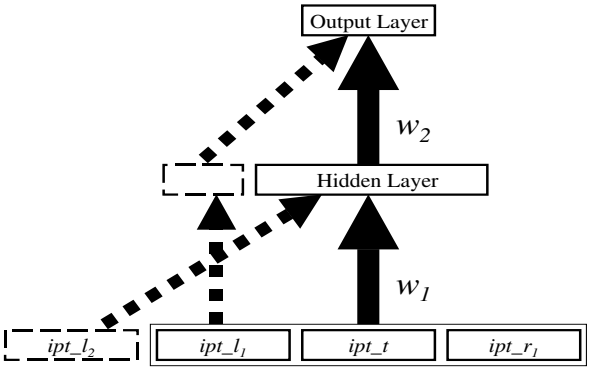

Fig. 3. How to train single-neuro tagger.

4.4 Features

Suppose that at most seven elements are adopted in the inputs for tagging and that there are 50 POSs. The n-gram models must estimate $50^7 = 7.8e + 11$ n-grams, while the single-neuro tagger with the longest input uses only 70,000 weights, which can be calculated by $n_{ipt} \cdot n_{hid} + n_{hid} \cdot n_{opt}$ where n_{ipt}, n_{hid}, and n_{opt} are, respectively, the number of units in the input, the hidden, and the output layers, and n_{hid} is set to be $n_{ipt}/2$. That neuro models require few parameters may offer another advantage: their performance is less affected by a small amount of training data than that of the statistical methods (Schmid, 1994). Neuro taggers also offer fast tagging compared to other models, although its training stage is longer.

5 Experimental Results

The Thai corpus used in the computer experiments contains 10,452 sentences that are 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 contain, respectively, 22,311 and 6,717 ambiguous words that serve as more than one POS and were used for training and testing. Because there are 47 types of POSs in Thai (Charoenporn et al., 1997), n in (6), (10), and (14) was set at 47. The single neuro-taggers are 3-layer neural networks whose input length, $l(IPT)$ $(=l+1+r)$, is set to 3-7 and whose size is $p \times \frac{p}{2} \times n$, where $p = n \times l(IPT)$. The multi-neuro tagger is constructed by five (i.e., $m = 5$) single-neuro taggers, SNT_i $(i = 1, \cdots, 5)$, in which $l(IPT_i) = 2 + i$.

Table 1 shows that no matter whether the information gain (IG) was used or not, the multi-neuro tagger has a correct rate of over 94%, which is higher than that of any of the single-neuro taggers. This indicates that by using the multi-neuro tagger the length of the context need not be chosen empirically; it can be selected dynamically instead. If we focus on the single-neuro taggers with inputs greater than four, we can see that the taggers with information gain are superior to those without information gain. Note that the correct rates shown in the table were obtained when only counting the ambiguous words in the testing set. The correct rate of the multi-neuro tagger is 98.9% if all the words in the testing set (the ratio of ambiguous words was 0.19) are counted. Moreover, although the overall performance is not improved

Table 1. Results of POS Tagging for Testing Data

Taggers	"single-neuro"					"multi-neuro"
$l(IPT_i)$	3	4	5	6	7	
with IG	0.915	0.920	0.929	0.930	0.933	0.943
without IG	0.924	0.927	0.922	0.926	0.926	0.941

much by adopting the information gains, the training can be greatly speeded up. It takes 1024 steps to train the first tagger, SNT_1, when the information gains are not used and only 664 steps to train the same tagger when the information gains are used.

Figure 4 shows learning (training) curves in different cases for the single-neuro tagger with six input elements. Thick line shows the case in which the tagger is trained by using trained weights of the tagger with five input elements as initial values. The thin line shows the case in which the tagger is trained independently. The dashed line shows the case in which the tagger is trained independently and does not use the information gain. From this figure, we know that the training time can be greatly reduced by using the previous result and the information gain.

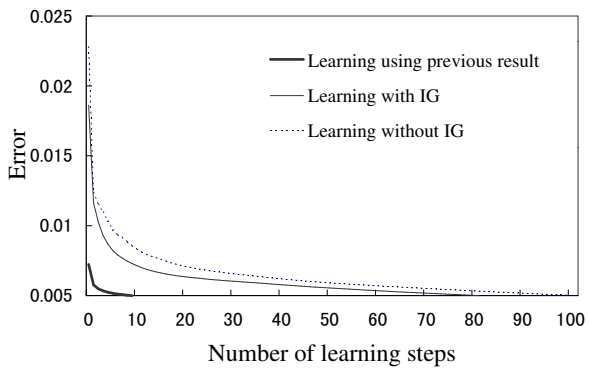

Fig. 4. Learning curves.

6 Conclusion

This paper described a multi-neuro tagger that uses variable lengths of contexts and weighted inputs for part of speech tagging. Computer experiments showed that the multi-neuro tagger has a correct rate of over 94% for tagging ambiguous words when a small Thai corpus with 22,311 ambiguous words is used for training. This result is better than any of the results obtained by the single-neuro taggers, which indicates that that the multi-neuro tagger can dynamically find suitable lengths of contexts for tagging. The cost to train a multi-neuro tagger was almost the same as that to train a single-neuro tagger using new learning methods in which the trained results (weights) of the previous taggers are used as initial weights for the latter ones. It was also shown that while the performance of tagging can be improved only slightly, the training time can be greatly reduced by using information gain to weight input elements.

References

Brill, E., Magerman, D., and Santorini, B.: Deducing linguistic structure from the statistics of large corpora, *Proc. DARPA Speech and Natural Language Workshop*, Hidden Valley PA, pp. 275-282, 1990.

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

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

Haykin, S.: *Neural Networks*, Macmillan College Publishing Company, Inc., 1994.

Merialdo, B.: Tagging English text with a probabilistic model, *Computational Linguistics*, vol. 20, No. 2, pp. 155-171, 1994.

Quinlan, J.: *C4.5: Programs for Machine Learning*, San Mateo, CA: Morgan Kaufmann, 1993.

Schmid, H.: Part-of-speech tagging with neural networks, *Proc. Int. Conf. on Computational Linguistics*, Japan, pp. 172-176, 1994.