Part of Speech Tagging with Mixed Approaches of Neural Networks and Transformation Rules

Qing Ma, Masaki Murata, Masao Utiyama
Kiyotaka Uchimoto and Hitoshi Isahara
Communications Research Laboratory
Ministry of Posts and Telecommunications, Japan
E-mail: {qma, murata, mutiyama, uchimoto, isahara}@crl.go.jp

Abstract

For the purpose of constructing a practical part of speech tagger that uses as few training data as possible, an approach using neural networks, which uses different lengths of contexts based on longest context priority and takes into account the maximization of information amount, have been proposed so far. To further improve the tagging performance, this paper proposes an integrated approach of using neural network and rule-based methods by introducing a set of transformation-based rules as a post-processor of the neural network tagger.

1 Introduction

A large number of part of speech (POS) taggers using rule-based (e.g., Brill, 1994), statistical (e.g., Merialdo, 1994), decision tree (e.g., Daelemans et al., 1996), and neural network (e.g., Schmid, 1994) models have been proposed so far. These taggers have reached a high accuracy partly because a very large amount of training data has been used (e.g., on the order of 1,000,000 words for English). For many other languages (e.g., Thai, which we treat in this paper), however, it will be not so easy to have large corpora from which large amounts of training data are available. It will therefore be desirable if we can construct a practical tagger which uses as few training data as possible. A multi-neuro tagger (Ma and Isahara, 1998) was therefore proposed for this purpose. This tagger performs POS tagging in different lengths of contexts based on longest context priority and each element of the input is weighted with information gains (Quinlan, 1993) for reflecting different relevances of words in tagging.

This paper proposes an integrated approach of using neural networks (neuro tagger) and a rule-based method. The neuro tagger is used as an initial-state annotator and the rule-based approach is used as a post-processor to correct the outputs of the neuro tagger. The rules are acquired from the training corpus using a set of templates by transformation-based error-driven learning (Brill, 1994). Preliminary computer experiments showed that about 1% tagging accuracy can be improved by adopting such a rule-based corrector for the neuro tagger.

2 POS Tagging Problems

In this paper, 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 up, 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. That is, unknown words that do not exist in the lexicon are not dealt with. The POS tagging problems therefore are to find a string of POSs $T = \tau_1\tau_2\cdots\tau_s (\tau_i \in \Gamma,\ i = 1, \cdots, s)$ at the word level (3) or sentence level (4) when a sentence $W = w_1 w_2 \cdots w_s$ $(w_i \in V,\ i = 1, \cdots, s)$ is given.

$$\varphi_1 : W^t \rightarrow \tau_t,\ t = 1, \cdots, s \tag{3}$$

$$\varphi_2 : W \rightarrow T \tag{4}$$

Here, t is the index of the target word (the word to be tagged) and W^t is the word sequence which is centered by the target word and has length $l+1+r$. That is,

$$W^t = w_{t-l} \cdots w_t \cdots w_{t+r}, \tag{5}$$

where $t - l \geq 1$, $t + r \leq s$. Tagging at the word and sentence levels can thus be regarded as a classification problem (by replacing the POS with class) and a search problem for searching for the optimal state transition (by replacing the POS with state), respectively. They can therefore be handled by using methods like the neural network model and HMM, respectively.

3 The Integrated Approach

The integrated approach of POS tagging uses a neural network (neuro tagger) as an initial-state annotator and a rule-based approach as an corrector that corrects the output of the neuro tagger.

3.1 The Neuro Tagger

Figure 1 shows the neuro tagger which consists of a 3-layer feedforward neural network with elastic input. It performs tagging at the word level [Eq. (3)] using a variable length of the context by training it in a supervised manner with a well-known error backpropagation algorithm (for details see e.g., Haykin, 1994).

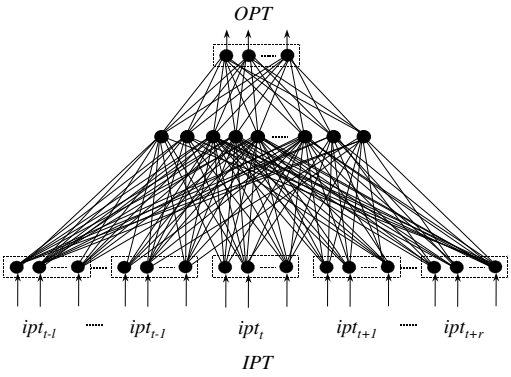

Fig. 1. The neuro tagger.

3.1.1 Construction of input and output

Input IPT is constructed from word sequence W^t [Eq. (5)], which is centered by target word w_t and has length $l + 1 + r$, as follows:

$$IPT = (ipt_{t-l}, \cdots, ipt_t, \cdots, ipt_{t+r}), \tag{6}$$

provided that the input length $l+1+r$ has elasticity, as described in Sec. 3.1.3. When word w is given in position x ($x = t - l, \cdots, t + r$), element ipt_x of input IPT is a weighted pattern defined as

$$ipt_x = g_x \cdot (e_{w1}, e_{w2}, \cdots, e_{w\gamma}) = (I_{w1}, I_{w2}, \cdots, I_{w\gamma}), \tag{7}$$

where g_x is the weight obtained by the information gain to be described in Sec. 3.1.2, γ is the number of types of POSs, and $I_{wi} = g_x \cdot e_{wi}$ ($i = 1, \cdots, \gamma$). If w is a word that appears in the training data, then each bit e_{wi} is obtained as follows:

$$e_{wi} = Prob(\tau^i | w). \tag{8}$$

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

$$Prob(\tau^i | w) = \frac{C(\tau^i, w)}{C(w)}, \tag{9}$$

where $C(\tau^i, w)$ is the number of times both τ^i and w appear and $C(w)$ 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:

$$e_{wi} = \begin{cases} \frac{1}{\gamma_w} & \text{if } \tau^i \text{ is a candidate} \\ 0 & \text{otherwise,} \end{cases} \tag{10}$$

where γ_w is the number of POSs that the word w can be. The OPT is a pattern defined as follows:

$$OPT = (O_1, O_2, \cdots, O_\gamma), \tag{11}$$

provided that the OPT is decoded as follows:

$$\tau_N(w_t) = \begin{cases} \tau^i & \text{if } O_i = 1 \ \& \ O_j = 0 \text{ for } j \neq i \\ Unknown & \text{otherwise,} \end{cases} \tag{12}$$

where $\tau_N(w_t)$ is the tagging result obtained by the neuro tagger.

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 (7)-(9) the input may be constructed simply as follows:

$$ipt_{t-i} = g_{t-i} \cdot OPT(-i), \tag{13}$$

where $i = 1, \cdots, l$. Here, $OPT(-i)$ means the output of the tagger for the ith word before the target word. 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:

$$ipt_{t-i} = g_{t-i} \cdot (w_{OPT} \cdot OPT(-i) + w_{DES} \cdot DES), \tag{14}$$

where DES is the desired output

$$DES = (D_1, D_2, \cdots, D_\gamma), \tag{15}$$

whose bits are defined as follows:

$$D_i = \begin{cases} 1 & \text{if } \tau^i \text{ is a desired answer} \\ 0 & \text{otherwise} \end{cases} \quad (16)$$

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

$$w_{OPT} = \frac{E_{OBJ}}{E_{ACT}} \quad (17)$$

and

$$w_{DES} = 1 - w_{OPT}, \quad (18)$$

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

3.1.2 Information gain

Weight g_x of each element ipt_x $(x = t-l, \cdots, t+r)$ in (7) can be obtained using information theory as follows. Let S be the training set, C_i be the ith class, i.e., the ith POS $(i = 1, \cdots, \gamma)$, and the set whose members (data) belong to C_i be S_{C_i}. 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

$$info(S) = -\sum_{i=1}^{\gamma} \frac{C(S_{C_i})}{C(S)} \times \ln \frac{C(S_{C_i})}{C(S)}, \quad (19)$$

where $C(X)$ is the number of data in set X. 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

$$info_x(S) = \sum_{i=1}^{h} \frac{C(S_i)}{C(S)} \times info(S_i). \quad (20)$$

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

$$gain(x) = info(S) - info_x(S), \quad (21)$$

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

$$g_x = gain(x). \quad (22)$$

3.1.3 Elasticity

Elastic inputs are used in the neuro tagger so that the feature of the multi-neuro tagger can be inherited, i.e., the length of context is variable in tagging based on longest context priority. In detail, (l, r) is initially set as large as possible for tagging. If $\tau_N(w_t) = Unknown$, then (l, r) is reduced by some constant interval. This process is repeated until $\tau_N(w_t) \neq Unknown$ or $(l, r) = (0, 0)$. On the other hand, to make the same set of connection weights of the neuro tagger with the largest (l, r) available as much as possible when using short inputs for tagging, in training phase the neuro tagger is regarded as a neural network that has gradually grown from small one. The training is therefore performed step by step from small networks to large ones.

3.1.4 Neuro tagger vs. statistical methods

Suppose that at most seven elements are used in the inputs for tagging and that there are 50 POSs, while the n-gram models must estimate $50^7 = 7.8 \times 10^{11}$ n-grams, the neuro tagger uses only 70,000 weights, provided that the number of units in the middle layer is half of that in the input layer. The fact that a neuro model requires few parameters may offer another advantage: its performance is less affected by a small amount of training data than that of the statistical methods (Schmid, 1994). A neuro tagger also offers faster tagging than other models, although its training stage is longer.

3.2 The Rule-Based Corrector

Like the statistical methods such as the n-gram model, what the neuro tagger can basically represent (learn) is rules that have an AND logical relationship, i.e., $(ipt_{t-l}\ \&\ \cdots\ ipt_t\ \&\ \cdots\ ipt_{t+r} \rightarrow OPT)$. In other words, the neuro tagger is not good at[1] representing (learning) other kinds of rules such as those that have an OR logical relationship, e.g., $(ipt_x \parallel ipt_y \rightarrow OPT)$, the monomial relationship[2], e.g., $(ipt_x \rightarrow OPT)$, and the relationships whose inputs consists of only words themselves, e.g., $(w \rightarrow OPT)$ or $(w_1\ \&\ w_2 \rightarrow OPT)$. To make up for these shortcomings of the neuro tagger, a rule-based corrector is adopted as its post-processor.

The rules are acquired from the training corpus using a set of transformation templates, which is designed to be neuro tagger oriented, i.e., designed to complete the above-mentioned relationships that the neuro tagger is not good at representing, by transformation-based error-driven learning

[1] Since neural networks can theoretically learn any relationships, we use the term "not good at" here.

[2] The neuro tagger can also represent a monomial relationship by regarding that it can tag the word using only ipt_t (the input of the target word), in the case of reducing the (l, r) to (0,0) as described in Sec. 3.1.3. The monomial relation described here, however, is a more general case, in which the input can be ipt_x $(x = t-l, \cdots, t+r)$.

Table 1: Transformation Templates Used in Computer Experiments

Change tag τ^a to tag τ^b when:

(monomial input)

(the input consists of a POS)

1. The left (right) word is tagged τ.
2. The second left (right) word is tagged τ.
3. The third left (right) word is tagged τ.

(the input consists of a word)

4. The left (right) word is w.
5. The second left (right) word is w.

(OR logical input)

(the input consists of POSs)

6. The left word is tagged τ_1 or right word is tagged τ_2.
7. The left (right) word is tagged τ_1 or the second left (right) word is tagged τ_2.
8. The left (right) word is tagged τ_1 or the second left (right) is tagged τ_2 or the right (left) word is tagged τ_3.
9. The left word is tagged τ_1 or the second left word is tagged τ_2 or the right word is tagged τ_3 or the second right word is tagged τ_4.
10. The left (right) word is tagged τ_1 or the second left (right) word is tagged τ_2 or the third left (right) word is tagged τ_3.

(the input consists of words)

11. The left (right) word is w_1 or the second left (right) word is w_2.
12. The left word is w_1 or the right word is w_2.

(AND logical input of words)

13. The left (right) word is w_1 and the second left (right) word is w_2.
14. The left word is w_1 and the right word is w_2.

(mixed input of a POS and a word)

15. The target word is w and the left (right) word is tagged τ.
16. The left (right) word is w and the left (right) word is tagged τ.

Table 2: Procedure for Learning Transformation Rules

1. apply the neuro tagger to the training corpus, which is then updated
2. compare the tagged results with the desired ones and find the error places
3. match the templates in all error places and obtain a set of transformation rules
4. select the best rule with the maximum value of $(cnt_good - h \cdot cnt_bad)$ through the entire corpus, where
   cnt_good: the number that transforms incorrect tags to correct ones,
   cnt_bad: the number that transforms correct tags to incorrect ones,
   h: a weight to control the strictness of generating the rule
5. apply the best rule to the training corpus, which is then updated
6. append the best rule to the ordered list of transformation rules
7. repeat steps 2 through 6 until no best rule can be selected, i.e., $cnt_good - h \cdot cnt_bad \leq 0$

(Brill, 1994). The set of templates used in the preliminary computer experiments is shown in Table 1 and the learning procedure is shown in Table 2.

When a corpus to be tagged is given, it is at first tagged by the neuro tagger. The tagged corpus is then corrected by using the ordered list of transformation rules. The correcting process is a repetition of applying rules in order to the corpus, which is then updated, until all rules have been applied.

4 Experimental Results

The Thai corpus used in the computer experiments 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 contain, respectively, 22,311 and 6,717 ambiguous words that serve as more than one POS and were used for training and testing. In the Thai corpus, 47 types of POSs are defined (Charoenporn et al., 1997).

Neuro tagger: It is constructed by a 3-layer neural networks whose input-middle-output layers have $p - \frac{p}{2} - \gamma$ units, respectively. Here $p = \gamma \times (l + 1 + r)$. The $(l + 1 + r)$ has the following elasticity. In training, the (l, r) was increased step by step as (1,1) $\rightarrow$ (2,1) $\rightarrow$ (2,2) $\rightarrow$ (3,2) $\rightarrow$ (3,3) and gradual training from a small network to a large network was performed. In tagging, on the other hand, the (l, r) was inversely reduced step by step as (3,3) $\rightarrow$ (3,2) $\rightarrow$ (2,2) $\rightarrow$ (2,1) $\rightarrow$ (1,1) $\rightarrow$ (1,0) $\rightarrow$ (0,0) as needed, provided that the number of units in the middle layer was fixed to the maximum value.

Rule-based corrector: The weight h (used in step 7 of the learning procedure shown in Table 2) for controlling the strictness of generating a rule was set to 100 to avoid as much as possible generating incorrect rules by regarding the neuro tagger as already having high accuracy and using the rule-based corrector as a tuner for fine-adjustment. An ordered list of 524 transformation rules was acquired by learning.

Computer experiments showed that the accuracy (counted only for the ambiguous words) of the integrated approach was 95.5%, which is 1.1% higher than that using only the neuro tagger, and the accuracy (counted for all the words) of the integrated approach reached 99.1%. In detail, by using the rule-based corrector, 87.6% of the total of 733 errors made by the neuro tagger for the training data and 19.1% of the total of 376 errors made by the neuro tagger for the testing data were corrected. In addition, the accuracy of the neuro tagger with elastic input was slightly higher than that (94.3%) of the previously proposed multi-neuro tagger, which is higher than that of any of the neuro taggers with fixed length of inputs (for details see Ma and Isahara, 1998). This indicates that by using the neuro tagger with elastic inputs, the length of the context need not be chosen empirically; it can be selected dynamically instead.

5 Conclusion

An integrated approach of using neural networks and rule-based method for part of speech tagging is proposed. The neural network tagger can dynamically find suitable lengths of contexts in tagging and the rule-based approach can supply the rules that the neural network tagger cannot learn well. Preliminary computer experiments show that about 19% of errors made by the neural network tagger could be corrected by the transformation rules and about 1% tagging accuracy could therefore be improved.

References

[1] Brill, E.: Transformation-based error-driven learning and natural language processing: a case study in part-of-speech tagging, *Computational Linguistics*, Vol. 21, No. 4, pp. 543-565, 1994.

[2] 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*, Phuket, Thailand, pp. 509-512, 1997.

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

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

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

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

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

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