Learning with Multiple Stacking for Named Entity Recognition

Koji Tsukamoto and Yutaka Mitsuishi and Manabu Sassano
Fujitsu Laboratories Ltd.
{tukamoto,mitsuishi-y,sassano}@jp.fujitsu.com

1 Introduction

In this paper, we present a learning method using multiple stacking for named entity recognition. In order to take into account the tags of the surrounding words, we propose a method which employs stacked learners using the tags predicted by the lower level learners. We have applied this approach to the CoNLL-2002 shared task to improve a base system.

2 System Description

Before describing our system, let us see one aspect of the named entity recognition, the outline of our method, and the relation to the previous works.

The task of named entity recognition can be regarded as a process of assigning a named entity tag to each given word, taking into account the patterns of surrounding words. Suppose that a sequence of words is given as below:

$$\ldots\ \mathrm{W}_{-2}, \mathrm{W}_{-1}, \mathrm{W}_0, \mathrm{W}_1, \mathrm{W}_2\ \ldots$$

Then, given that the current position is at word W_0, the task is to assign tag T_0 to W_0.

In the named entity recognition task, an entity is often made up of a sequence of words, rather than a single word. For example, an entity "the United States of America" consists of five words. In order to allocate a tag to each word, the tags of the surrounding words (we call these tags the *surrounding tags*) can be a clue to predict the tag of the word (we call this tag the *current tag*). For the test set, however, these tags are unknown.

In order to take into account the surrounding tags for the prediction of the current tag, we propose a method which employs multiple stacked learners, an extension of stacking method (Wolpert, 1992). Stacking based method for named entity recognition usually employs two or more level learners. The higher level learner uses the current tags predicted by its lower level learners. In our method, by contrast, the higher level learner uses not only the current tag but also the surrounding tags predicted by the lower level learner. Our aim is to leverage the performance of the base system using the surrounding tags as the features.

At least two groups have previously proposed systems which use the predicted surrounding tags. One system, proposed by van Halteren et al. (1998), also uses stacking method. This system uses four completely different types of taggers as the first level learners, because it has been assumed that first level learners should be as different as possible. The tags predicted by the first level learners are used as the features of the second level learner.

The other system, proposed by (Kudo and Matsumoto, 2000; Yamada et al., 2001), uses the "dynamic features". In the test phase, the predicted tags of the preceding (or subsequent) words are used as the features, which are called "dynamic features". In the training phase, the system uses the answer tags of the preceding (or subsequent) words as the features.

More detailed descriptions of our system are shown below:

2.1 Learning Algorithm

As the learning algorithm for all the levels , we use an extension of AdaBoost, the *real AdaBoost.MH* which is extended to handle multiclass problems (Schapire and Singer, 1999). For weak learners, we use decision stumps (Schapire and Singer, 1999), which select only one feature to classify an example.

2.2 Features

We use the following types of the features for the prediction of the tag of the word.

- surface form of W_{-2}, W_{-1}, W_0, W_1 and W_2.

Word Feature	Example Text
Digit	25
Digit+Alphabet	CDG1
Symbol	.
Uppercase	EFE
Capitalized	Australia
Lowercase(word length $>$ 3 characters)	necesidad
Lowercase(word length $\leq$ 3 characters)	del
Other	hoy,

Table 1: Word features and examples

- One of the eight word features in Table 1. These features are similar to those used in (Bikel et al., 1997).
- First and last two/three letters of W_0
- Estimated tag of W_0 based on the word unigram model in the training set.

Additionally, we use the surrounding tag feature. This feature is discussed in Section 2.3.

2.3 Multiple Stacking

In order to take into account the tags of the surrounding words, our system employs stacked learners. Figure 1 gives the outline of the learning and applying algorithm of our system. In the learning phase, the base system is trained at first. After that, the higher level learners are trained using word features (described in Section 2.2), current tag T_0 and surrounding tags $T_{-2,-1,1,2}$ predicted by the lower level learner. While these tag may not be correctly predicted , if the accuracy of the prediction of the lower level learner is improved, the features used in each prediction become accurate. In the applying phase, all of the learners are cascaded in the order.

Compared to the previous systems (van Halteren et al., 1998; Kudo and Matsumoto, 2000; Yamada et al., 2001), our system is: (i) employing more than two levels stacking, (ii) using only one algorithm and training only one learner at each level, (iii) using the surrounding tag given by the lower level learner. (iv) using both the preceding and subsequent tags as the features. (v) using the predicted tags instead of the answer tags in the training phase.

3 Experiments and Results

In this section, the experimental conditions and the results of the proposed method are shown.

In order to improve the performance of the base system, the tag sequence to be predicted is formatted according to IOB1, even though the sequence

Let L_k denote the k th level learner and let $T_i^{(k)}$ denote k th level output tags for W_i.

Learning:

1. Train the base learner L_0 using the features described in Table 1.
2. for k = 1,...,N
   - Get $T^{(k-1)}$ of the training set using L_{k-1}, the features described in section 2.2 (and $T_{-2}^{(k-2)}$, $T_{-1}^{(k-2)}$, $T_0^{(k-2)}$, $T_1^{(k-2)}$, $T_2^{(k-2)}$ if $k > 1$).
   - Train L_k using the features described in section 2.2 and $T_{-2}^{(k-1)}$, $T_{-1}^{(k-1)}$, $T_0^{(k-1)}$, $T_1^{(k-1)}$, $T_2^{(k-1)}$.
3. Output L_0, L_1,...,L_N.

Applying:

1. for k = 0,...,N
   - Get $T^{(k)}$ of test set using L_k, the features described in section 2.2 (and $T_{-2}^{(k-1)}$, $T_{-1}^{(k-1)}$, $T_0^{(k-1)}$, $T_1^{(k-1)}$, $T_2^{(k-1)}$ if $k > 0$).
2. Output $T^{(N)}$.

Figure 1: Outline of multiple stacking algorithm

in the original corpus was formatted according to IOB2 (Tjong Kim Sang and Veenstra, 1999).

To reduce the computational cost, features appearing fewer than three times are eliminated in the training phase.

3.1 Base System

To evaluate the effect of multiple stacking in the next section, the performance of the base system is shown in Figure 2. A performance peak is observed after 10,000 rounds of boosting. Note that a decision stump used in the real AdaBoost.MH takes into account only one feature. Hence the number of features used by real AdaBoost.MH is less than the number of the rounds. In our experiment, because the rounds of boosting are always less than the number of the features (about 40,000), a large proportion of features are not used by the learners. If the rounds of boosting in the base system are not enough, stacking effect may be similar to increasing the rounds of boosting. In Figure 2, however, we can see that 10,000 rounds is enough.

3.2 Multiple Stacking

We examine the effect of multiple stacking compared to the base system.

The $F_{\beta=1}$ score of multiple stacking for the Spanish test set (esp.testa) is shown in Table 2. By stacking learners, the score of each named entity is im-

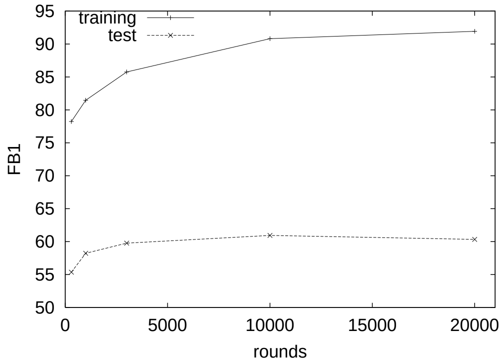

Figure 2: $F_{\beta=1}$ score of the base system

n	overall	LOC	MISC	ORG	PER
L_0	60.94	64.12	33.61	61.37	70.91
L_1	65.68	67.35	40.04	66.26	74.82
L_2	66.91	68.02	41.69	67.38	75.51
L_3	67.24	67.96	41.76	67.95	75.77
L_4	67.35	67.81	42.75	67.91	75.89
L_5	67.35	67.78	42.30	67.95	75.99
L_6	67.30	67.57	42.12	68.01	76.01

Table 2: $F_{\beta=1}$ score of stacked learner

proved. Compared to the overall $F_{\beta=1}$ score of L_0, the score of L_1, stacking one learner over the base system, is improved by 4.74 point. Further more, compared to the score of L_1, the score of L_4 is higher by 1.67 point. Through five iterations of stacking, the score is continuously increased. The overall scores for the six tests are briefly shown in Table 3. The effect of two level stacking is higher for the Spanish tests. However, multiple staking effects greater for the Dutch test, especially for the corpus without part of speech. As discussed in Section 3.1, the improvement of the score is not due to the rounds of boosting. Thus, it is due to multiple stacking.

In Table 2, stacking effects for MISC and ORG appear greater than those for LOC and PER. It is reasonable to suppose that MISC and ORG entities consist of a relatively long sequence of words, and the surrounding tags can be good clues for the prediction of the current tag. Indeed, in the Spanish training set, the ratios of entities which consist of more than three words are 9.7%, 22.4%, 4.4% and 3.5% for ORG, MISC, LOC and PER respectively.

Table 4 and 5 show examples of the predicted tags through the stacked level. Let us see how multiple stacking works using the examples in Table 5. Let the word "fin" be the current position. The answer tag is "I-MISC". When we use the base system

	esp.a	esp.b	ned.a*	ned.b*	ned.a	ned.b
L_0	60.94	65.70	55.50	58.04	54.23	57.43
L_1	65.68	69.39	56.72	59.18	56.20	58.84
L_4	67.35	71.49	58.23	60.74	58.83	60.93

Table 3: $F_{\beta=1}$ scores for six tests. "*" indicates use of part of speech tags.

	Answer	L_0	L_1	L_2
:				
Colegio	I-ORG	I-ORG	I-ORG	I-ORG
Plico	I-ORG	I-ORG	I-ORG	I-ORG
Arias	I-ORG	I-LOC	I-LOC	I-ORG
Montano	I-ORG	I-LOC	I-ORG	I-ORG
,	O	O	O	O
de	O	O	O	O
Badajoz	I-LOC	I-LOC	I-LOC	I-LOC
:				

Table 4: Example of the prediction (line 2434 to 2440 in esp.testa)

	Answer	L_0	L_1	L_2	L_3
:					
el	O	O	O	O	O
libro	O	O	O	O	O
"	I-MISC	I-MISC	I-MISC	I-MISC	I-MISC
Una	I-MISC	O	I-MISC	I-MISC	I-MISC
sonrisa	I-MISC	I-MISC	I-MISC	I-MISC	I-MISC
sin	I-MISC	O	I-MISC	I-MISC	I-MISC
fin	I-MISC	O	O	I-MISC	I-MISC
"	I-MISC	O	O	O	I-MISC
,	O	O	O	O	O
:					

Table 5: Example of the prediction (line 16231 to 16239 in esp.testa)

L_0, the predicted tag of the word is "O". In the next level, L_1 uses the surrounding tag features "I-MISC, O, (O,) O, O" and also outputs "O". In the third level, however, L_2 correctly predicts the tag using the surrounding tag features "I-MISC, I-MISC, (O,) O, O". Note that no other feature changes through the levels. The improvement in the example is clearly caused by multiple stacking. As a result, this MISC entity is allocated tags correctly by L_3. The above effect would not be achieved by two level stacking. This result clearly shows that multiple stacking method has an advantage.

Next we examine the effect of the learning algorithm to multiple stacking. We use the real AdaBoost.MH for 300, 1,000, 3,000, 10,000, 20,000 rounds. Their $F_{\beta=1}$ scores in each stacking level are plotted in Figure 3. The score improves by stacking for all algorithms. The highest score is achieved by 10,000 iterations at every stacking level. The shapes of the curves in Figure 3 are similar to each other. This result suggests that the stacking effect is scarcely affected by the performance of the algorithm.

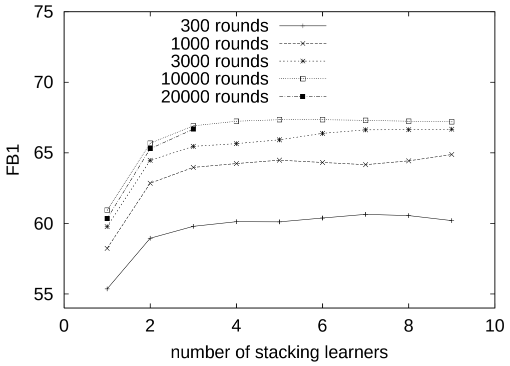

Figure 3: $F_{\beta=1}$ scores of different base system

4 Conclusion

We have presented a new method for recognizing named entity by multiple stacking. This method can leverage the performance of the base system employing multiple stacked learner and using not only the current tag but also the surrounding tags predicted by the lower level learner. By stacking 5 real AdaBoost.MH learners, we can obtain $F_{\beta=1}$ of 67.35 for the Spanish named entity recognition task.

References

D. M. Bikel, S. Miller, R. Schwartz and R. Weischedel. 1997. Nymble: a high-performance learning name-finder. In *Fifth Conference on Applied Natural Language Processing.*

H. van Halteren, J. Zavrel and W. Daelemans. 1998. Improving data driven wordclass tagging by system combination. In *Proceedings of the 17th COLING and the 36th Annual Meeting of ACL.*

T. Kudo and Y. Matsumoto. 2000. Japanese dependency structure analysis based on support vector machines. In *Proceedings of the 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora.*

R. E. Schapire and Y. Singer. 1999. Improved boosting algorithms using confidence-rated predictions. *Machine Learning, 37(3).*

E. F. Tjong Kim Sang and J. Veenstra. 1999. Representing text chunks. In *Proceedings of EACL'99*.

D. H. Wolpert. 1992. Stacked generalization. *Neural Networks, 5.*

H. Yamada, T. Kudo and Y. Matsumoto. 2001. Japanese named entity extraction using support vector machines. *Information Processing Society of Japan, SIG Notes NL 142-17 (in Japanese).*

Spanish dev.	precision	recall	$F_{\beta=1}$
LOC	59.75%	78.38%	67.81
MISC	40.40%	45.39%	42.75
ORG	67.48%	68.35%	67.91
PER	78.26%	73.65%	75.89
overall	65.09%	69.76%	67.35

Spanish test	precision	recall	$F_{\beta=1}$
LOC	70.96%	73.25%	72.08
MISC	41.83%	42.94%	42.38
ORG	68.21%	76.93%	72.31
PER	80.23%	84.49%	82.31
overall	69.04%	74.12%	71.49

Dutch dev.	precision	recall	$F_{\beta=1}$
LOC	54.87%	65.13%	59.56
MISC	59.12%	65.15%	61.99
ORG	67.95%	51.84%	58.81
PER	47.58%	66.53%	55.48
overall	55.92%	62.05%	58.83

Dutch test	precision	recall	$F_{\beta=1}$
LOC	63.79%	73.80%	68.43
MISC	56.89%	59.14%	57.99
ORG	60.16%	53.02%	56.36
PER	52.61%	74.73%	61.75
overall	57.33%	65.02%	60.93

Table 6: Overview of the precision, recall and $F_{\beta=1}$ of multiple stacking with k=4 and 10,000 rounds of boosting. Dutch data is processed without part of speech tags.