Combining Segmenter and Chunker for Chinese Word Segmentation

Masayuki Asahara, Chooi Ling Goh, Xiaojie Wang, Yuji Matsumoto
Graduate School of Information Science
Nara Institute of Science and Technology, Japan
{masayu-a,ling-g,xiaoji-w,matsu}@is.aist-nara.ac.jp

Abstract

Our proposed method is to use a Hidden Markov Model-based word segmenter and a Support Vector Machine-based chunker for Chinese word segmentation. Firstly, input sentences are analyzed by the Hidden Markov Model-based word segmenter. The word segmenter produces n-best word candidates together with some class information and confidence measures. Secondly, the extracted words are broken into character units and each character is annotated with the possible word class and the position in the word, which are then used as the features for the chunker. Finally, the Support Vector Machine-based chunker brings character units together into words so as to determine the word boundaries.

1 Methods

We participate in the closed test for all four sets of data in Chinese Word Segmentation Bakeoff. Our method is based on the following two steps:

1. The input sentence is segmented into a word sequence by Hidden Markov Model-based word segmenter. The segmenter assigns a word class with a confidence measure for each word at the hidden states. The model is trained by Baum-Welch algorithm.

2. Each character in the sentence is annotated with the word class tag and the position in the word. The n-best word candidates derived from the word segmenter are also extracted as the features. A support vector machine-based chunker corrects the errors made by the segmenter using the extracted features.

We will describe each of these steps in more details.

1.1 Hidden Markov Model-based Word Segmenter

Our word segmenter is based on Hidden Markov Model (HMM). We first decide the number of hidden states (classes) and assume that the each word can belong to all the classes with some probability. The problem is defined as a search for the sequence of word classes $C = c_1, \ldots, c_n$ given a word sequence $W = w_1, \ldots, w_n$. The target is to find W and C for a given input S that maximizes the following probability:

$$arg \max_{W,C} P(W|C)P(C)$$

We assume that the word probability $P(W|C)$ is constrained only by its word class, and that the class probability $P(C)$ is constrained only by the class of the preceding word. These probabilities are estimated by the Baum-Welch algorithm using the training material (See (Manning and Schütze., 1999)). The learning process is based on the Baum-Welch algorithm and is the same as the well-known use of HMM for part-of-speech tagging problem, except that the number of states are arbitrarily determined and the initial probabilities are randomly assigned in our model.

1.2 Correction by Support Vector Machine-based Chunker

While the HMM-based word segmenter achieves good accuracy for known words, it cannot identify compound words and out-of-vocabulary words. Therefore, we introduce a Support Vector Machine(below SVM)-based chunker (Kudo and Matsumoto, 2001) to cover the errors made by the segmenter. The SVM-based chunker re-assigns new word boundaries to the output of the segmenter.

An SVM (Vapnik, 1998) is a binary classifier. Suppose we have a set of training data for a binary class problem: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$, where $\mathbf{x}_i \in R^n$ is a feature vector of the i th sample in the training data and

$y_i \in \{+1, -1\}$ is the label of the sample. The goal is to find a decision function which accurately predicts y for an unseen $\mathbf{x}$. An SVM classifier gives a decision function $f(\mathbf{x})$ for an input vector $\mathbf{x}$ where

$$f(\mathbf{x}) = sign(\sum_{\mathbf{z}_i \in SV} \alpha_i y_i K(\mathbf{x}, \mathbf{z}_i) + b).$$

$f(\mathbf{x}) = +1$ means that $\mathbf{x}$ is a positive member, and $f(\mathbf{x}) = -1$ means that $\mathbf{x}$ is a negative member. The vectors $\mathbf{z}_i$ are called support vectors, which receive a non-zero weight α_i. Support vectors and the parameters are determined by solving a quadratic programming problem. $K(\mathbf{x}, \mathbf{z})$ is a kernel function which maps vectors into a higher dimensional space. We use a polynomial kernel of degree 2 given by $K(\mathbf{x}, \mathbf{z}) = (1 + \mathbf{x} \cdot \mathbf{z})^2$.

The SVM classifier determines the position tag for each character. We introduce the word class tag as the feature, which is generated by the word segmenter. Since we perform chunking by character units, the feature used by the classifier will be the information for the character unit.

The training data for our SVM-based chunker is constructed from the output of the HMM-based word segmenter defined in the previous section. In the current setting, the HMM produces all the possible tags (class labels) for each of the word within a predefined probability bound. All the words in the output are then segmented into characters, and each of the characters is tagged with pairs of a word class and a position tag. For example, in the paired tag "0-B", "0" is a class label of the word which the character belongs to and "B" indicates the character's position in the word. The number of classes is determined in advance of the HMM learning. The position tag consists of the following four tags (S/B/E/I): S means a single-character word; B is the first character in a multi-character word; E is the last character in a multi-character word; I is the intermediate character in a multi-character word longer than 2 characters. As shown in Figure 1, we set the HMM-based word segmenter to produce the classes of n-best word candidates to take into account multiple possibility of word boundaries.

The correct word boundary can be defined by assigning either of two kinds of tags to each of the characters. Look at the rightmost column of Figure 1 named as "Chunker Outputs." The label "B" in this column shows that the character is the first character of a *correct* word, and "I" shows that the character is the other part of a word. This means that the preceding positions of "B" tags are the word boundaries.

Those two tags correspond to the two classes of the SVM chunker: In the training (and test) phrase, we use window size of two characters to the left and right direction to learn (and estimate) the class for a character. For example, the shadowed parts in Figure 1 are used as the features to learn (or estimate) the word boundary tag "I" for the character "判".

Characters	Word Segmenter Outputs				Chunker Outputs
	Best	2nd Best	3rd Best	4th Best	
政	0-B	1-S	*	*	B
治	0-E	3-S	*	*	I
谈	1-B	4-S	*	*	B
判	1-E	4-S	1-S	2-S	I
是	0-S	*	*	*	B
推	0-B	4-S	*	*	B
动	0-E	3-S	*	*	I
两	2-B	4-S	*	*	B
岸	2-E	4-S	2-S	0-S	I
关	3-B	0-S	*	*	B
系	3-I	4-S	*	*	I
发	0-B	2-S	*	*	B
展	0-E	0-S	*	*	I
的	1-S	*	*	*	B
关	1-B	*	*	*	B
键	1-E	*	*	*	I
EOS					

Figure 1: The Extracted Features for the Chunker

2 Model Validation

To find out the best setting of learning, we would like to determine "the number of word classes" and "the depth of n-best word candidates" by using some sort of confidence measure. We perform validation experiments for these two types of parameters by using the training material provided.

2.1 Validation Tests for HMM-based Word Segmenter

The first validation experiment is to determine "the number of word classes" of the HMM. 80% of the material is used for the HMM training, and the other 20% is used as the validation set. We test two settings for the number of classes – 5 & 10. The results are shown in Table 1.

Table 1: Validation Results for HMM

Data	# of classes	Rec.	Prec.	F
AS	5	0.845	0.768	0.804
AS	10	0.900	0.857	0.878
CTB	5	0.909	0.844	0.875
CTB	10	0.912	0.848	0.879
HK	5	0.867	0.742	0.799
HK	10	0.867	0.741	0.799
PK	5	0.942	0.902	0.921
PK	10	0.944	0.905	0.924

In most cases, models perform slightly better with the increasing of the number of classes. When the corpus size is large like the Academia Sinica data, this tendency becomes more significant.

Whereas it is known that the Baum Welch algorithm is very sensitive to the initialization of the classes, we randomly assigned the initial classes without making much effort. There are two reasons: (1) Since the word segmenter outputs are used as the clues to the chunker in our method, we only need some consistent class annotations. (2) The initial classes did not affect on the word segmentation accuracy in our pilot experiments.

2.2 Validation Tests for SVM-based Chunker

The second validation test is for the chunking model to determine both "the number of word classes" and "the depth of the n-best candidates". 80% of the material used for the HMM training, another 10% is used for the chunking model training and the last 10% is used for the validation test. The results are shown in Table 2, 3 and 4. Since the training of this model is time- and resource-consuming, the Academia Sinica data being very large could not get enough time to finish the validation test.

Table 2: Validation Results (CTB) for Chunking

# of classes	n-best	Rec.	Prec.	F
5	1	0.957	0.930	0.943
5	**2**	**0.957**	**0.931**	**0.944**
5	3	0.957	0.930	0.943
5	4	0.957	0.930	0.943
10	1	0.956	0.929	0.943
10	2	0.957	0.928	0.942
10	3	0.956	0.929	0.942
10	4	0.955	0.928	0.941

Table 3: Validation Results (HK) for Chunking

# of classes	n-best	Rec.	Prec.	F
5	1	0.853	0.793	0.822
5	2	0.859	0.799	0.828
5	3	0.859	0.799	0.828
5	**4**	**0.859**	**0.800**	**0.828**
10	1	0.856	0.793	0.823
10	2	0.858	0.797	0.826
10	3	0.857	0.796	0.826
10	4	0.858	0.797	0.826

The results show that the chunker actually improves the word segmentation accuracy compared with the output of the HMM word segmenter for these three data sets. The segmentation errors made by the word segmenter for compound words and unknown words are corrected. The improvement in Chinese Treebank (CTB) data set is significant, because the data set contains many compound words.

Table 4: Validation Results (PK) for Chunking

# of classes	n-best	Rec.	Prec.	F
5	1	0.960	0.934	0.947
5	2	0.961	0.935	0.948
5	**3**	**0.962**	**0.936**	**0.949**
5	4	0.962	0.935	0.948
10	1	0.961	0.932	0.946
10	2	0.962	0.935	0.948
10	3	0.961	0.934	0.947
10	4	0.961	0.934	0.947

There is no significant difference in the results between the different depths of n-best answers. Still, we choose the best model for the test materials among them. If we need to have a faster analyzer, we should employ only the best answer of the word segmentation.

For the HMM, the larger number of classes tends to get better accuracy than smaller ones. However, for the chunking model, the result is the other way round. The model with the smaller number of classes gets slightly better accuracy. So, there should be trade-off between smaller and larger number of classes.

3 Final Models for Test Material

For the final models, 80% of the training material is used for HMM training and 100% of the material is used for the chunking model training. The parameters, namely "the number of word classes" and "the depth of n-best word candidates", are determined by the validation tests described in Section 2. While there is no significant difference between the depths of n-best answers, we choose the best model among them for the testing. The parameters are shown in Table 7.

We cannot create the model using all the original Academia Sinica data because of its large size. Therefore, we use 80% of the data for HMM training (5 classes) and only 10% for chunking model training (with only the best candidates).

Table 7: The Models for the Test Material
– with respect to F-Measure in Our Validation Test

Data	# of classes	n-best	F
AS	5	1	N/A
CTB	5	2	0.943
HK	5	4	0.828
PK	5	3	0.948

Table 5: Throughput Speeds (characters per second)

Data	Word Seg.	(# of words)	Fea. Ext.	(n-best)	Chunker	(# of SV)	Total Speed
AS	57000	(462750)	7640	(Only Best)	279	(96452)	241
CTB	54400	(77324)	4040	(to 2nd Best)	894	(16736)	671
HK	38900	(93231)	3870	(to 4th Best)	649	(14904)	524
PK	57400	(215865)	6209	(to 3rd Best)	254	(49736)	200

Table 6: Results for the Test Materials

Data	T. Rec.	T. Prec.	F	OOV Rec.	IV Rec.	Ranking
AS	0.944	0.945	0.945	0.574	0.952	3rd/6
CTB	0.852	0.807	0.829	0.412	0.949	8th/10
HK	0.940	0.908	0.924	0.415	0.980	5th/6
PK	0.933	0.916	0.924	0.357	0.975	2nd/4

4 Throughput Speeds

As described, our system is based on three modules: HMM-based word segmenter, Feature extractor and SVM-based chunker. The word segmenter is composed by *ChaSen* (written in C/C++) (Matsumoto et. al., 2003) which is adopted for GB/Big5 encoding. The feature extractor is written in Perl. The SVM-based chunker is composed by *YamCha* (written in C++) (Kudo and Matsumoto, 2001).

Table 5 shows the speeds [1] of the three modules individually and of the total system. “# of words” means the size of the word segmenter lexicon. Note that, if a word belongs to more than one class, we regard them as different words in our definition. “# of SV” means the number of support vectors in the chunker. The total system speed depends highly on that of the chunker. It is known that the speed of SVM classifiers depends on the number of support vectors and the number of features.

5 Conclusion

We presented our method for Chinese Word Segmentation Bakeoff in 2nd SIGHAN Workshop. The results for the test materials are shown in Table 6. The proposed method is purely corpus-based statistical/machine learning method. Although we did not incorporate any heuristic rules (e.g. part-of-speeches, functional words and concatenation for numbers) into the model, the method achieved considerable accuracy for the word segmentation task.

Acknowledgments

We thank Mr. Taku Kudo of NAIST for his development of the SVM-based chunker *YamCha*.

[1]The throughput speeds are measured on a machine: Intel(R) Xeon(TM) CPU 2.80GHz $\times$ 2, Memory 4GB, RedHat Linux 9.

References

T. Kudo and Y. Matsumoto. 2001. Chunking with Support Vector Machines. In *Proc. of NAACL 2001*, pages 192–199.

C. D. Manning and H. Schütze. 1999. *Foundation of Statistical Natural Language Processing.* Chapter 9. Markov Models, pages 317–340.

Y. Matsumoto, A. Kitauchi, T. Yamashita, Y. Hirano, K. Takaoka and M. Asahara 2003. Morphological Analyzer ChaSen-2.3.0 Users Manual Tech. Report. Nara Institute of Science and Technology, Japan.

L. A. Ramshaw and M. P. Marcus. 1995 Text chunking using transformation-bases learning In *Proc. of the 3rd Workshop on Very Large Corpora*, pages 83–94.

V. N. Vapnik. 1998. *Statistical Learning Theory.* A Wiley-Interscience Publication.