# Chinese Word Segmentation as LMR Tagging

**Nianwen Xue**
Inst. for Research in Cognitive Science
University of Pennsylvania
Philadelphia, PA 19104, USA
xueniwen@linc.cis.upenn.edu

**Libin Shen**
Dept. of Computer and Info. Science
University of Pennsylvania
Philadelphia, PA 19104, USA
libin@linc.cis.upenn.edu

## Abstract

In this paper we present Chinese word segmentation algorithms based on the so-called LMR tagging. Our LMR taggers are implemented with the Maximum Entropy Markov Model and we then use Transformation-Based Learning to combine the results of the two LMR taggers that scan the input in opposite directions. Our system achieves F-scores of $95.9\%$ and $91.6\%$ on the Academia Sinica corpus and the Hong Kong City University corpus respectively.

## 1 Segmentation as Tagging

Unlike English text in which sentences are sequences of words delimited by white spaces, in Chinese text, sentences are represented as strings of Chinese characters or *hanzi* without similar natural delimiters. Therefore, the first step in a Chinese language processing task is to identify the sequence of words in a sentence and mark boundaries in appropriate places. This may sound simple enough but in reality identifying words in Chinese is a non-trivial problem that has drawn a large body of research in the Chinese language processing community (Fan and Tsai, 1988; Gan et al., 1996; Sproat et al., 1996; Wu, 2003; Xue, 2003).

The key to accurate automatic word identification in Chinese lies in the successful resolution of ambiguities and a proper way to handle out-of-vocabulary words. The ambiguities in Chinese word segmentation is due to the fact that a *hanzi* can occur in different word-internal positions (Xue, 2003). Given the proper context, generally provided by the sentence in which it occurs, the position of a *hanzi* can be determined. In this paper, we model the Chinese word segmentation as a *hanzi* tagging problem and use a machine-learning algorithm to determine the appropriate position for a *hanzi*. There are several reasons why we may expect this approach to work. First, Chinese words generally have fewer than four characters. As a result, the number of positions is small. Second, although each *hanzi* can in principle occur in all possible positions, not all *hanzi* behave this way. A substantial number of *hanzi* are distributed in a constrained manner. For example, 们, the plural marker, almost always occurs in the word-final position. Finally, although Chinese words cannot be exhaustively listed and new words are bound to occur in naturally occurring text, the same is not true for *hanzi*. The number of *hanzi* stays fairly constant and we do not generally expect to see new *hanzi*.

We represent the positions of a *hanzi* with four different tags (Table 1): LM for a *hanzi* that occurs on the left periphery of a word, followed by other *hanzi*, MM for a *hanzi* that occurs in the middle of a word, MR for a *hanzi* that occurs on the right periphery of word, preceded by other *hanzi*, and LR for *hanzi* that is a word by itself. We call this LMR tagging. With this approach, word segmentation is a process where each *hanzi* is assigned an LMR tag and sequences of *hanzi* are then converted into sequences of words based on the LMR tags. The use of four tags is linguistically intuitive in that LM tags morphemes that are prefixes or stems in the absence of prefixes, MR tags morphemes that are suffixes or stems in the absence of suffixes, MM tags stems with affixes and LR tags stems without affixes. Representing the distributions of *hanzi* with LMR tags also makes it easy to use machine learning algorithms which has been successfully applied to other tagging problems, such as POS-tagging and IOB tagging used in text chunking.

| | Right Boundary (R) | Not Right Boundary (M) |
|---|---|---|
| Left Boundary (L) | LR | LM |
| Not Left Boundary (M) | MR | MM |

Table 1: LMR Tagging

## 2 Tagging Algorithms

Our algorithm consists of two parts. We first implement two Maximum Entropy taggers, one of which scans the input from left to right and the other scans the input from right to left. Then we implement a Transformation Based Algorithm to combine the results of the two taggers.

### 2.1 The Maximum Entropy Tagger

The Maximum Entropy Markov Model (MEMM) has been successfully used in some tagging problems. MEMM models are capable of utilizing a large set of features that generative models cannot use. On the other hand, MEMM approaches scan the input incrementally as generative models do.

The Maximum Entropy Markov Model used in POS-tagging is described in detail in (Ratnaparkhi, 1996) and the LMR tagger here uses the same probability model. The probability model is defined over $H \times T$, where $H$ is the set of possible contexts or "histories" and $T$ is the set of possible tags. The model's joint probability of a history $h$ and a tag $t$ is defined as

$$p(h,t) = \pi\mu \prod_{j=1}^{k} \alpha_j^{f_j^{(h,t)}} \quad (1)$$

where $\pi$ is a normalization constant, $\{\mu, \alpha_1, ..., \alpha_k\}$ are the model parameters and $\{f_1, ..., f_k\}$ are known as features, where $f_j(h,t) \in \{0,1\}$. Each feature $f_j$ has a corresponding parameter $\alpha_j$, that effectively serves as a "weight" of this feature. In the training process, given a sequence of characters $\{c_1, \cdots, c_n\}$ and their LMR tags $\{t_1, ..., t_n\}$ as training data, the purpose is to determine the parameters $\{\mu, \alpha_1, ..., \alpha_k\}$ that maximize the likelihood of the training data using $p$:

$$L(P) = \prod_{i=1}^{n} P(h_i, t_i) = \prod_{i=1}^{n} \pi\mu \prod_{j=1}^{k} \alpha_j^{f_j^{(h_i,t_i)}} \quad (2)$$

The success of the model in tagging depends to a large extent on the selection of suitable features. Given $(h,t)$, a feature must encode information that helps to predict $t$. The features we used in our experiments are instantiations of the feature templates in (1). Feature templates (b) to (e) represent character features while (f) represents tag features. In the following list, $C_{-3}...C_3$ are characters and $T_{-3}...T_3$ are LMR tags.

(1) Feature templates
(a) Default feature
(b) The current character ($C_0$)
(c) The previous (next) two characters ($C_{-2}$, $C_{-1}$, $C_1$, $C_2$)
(d) The previous (next) character and the current character ($C_{-1}C_0$, $C_0C_1$), the previous two characters ($C_{-2}C_{-1}$), and the next two characters ($C_1C_2$)
(e) The previous and the next character ($C_{-1}C_1$)
(f) The tag of the previous character ($T_{-1}$), and the tag of the character two before the current character ($T_{-2}$)

### 2.2 Transformation-Based Learning

One potential problem with the MEMM is that it can only scan the input in one direction, from left to right or from right to left. It is noted in (Lafferty et al., 2001) that non-generative finite-state models, MEMM models included, share a weakness which they call the Label Bias Problem (LBP): a transition leaving a given state compete only against all other transitions in the model. They proposed Conditional Random Fields (CRFs) as a solution to address this problem.

A partial solution to the LBP is to compute the probability of transitions in both directions. This way we can use two MEMM taggers, one of which scans the input from left to right and the other scans the input from right to left. This strategy has been successfully used in (Shen and Joshi, 2003). In that paper, pairwise voting (van Halteren et al., 1998) has been used to combine the results of two supertaggers that scan the input in the opposite directions.

The pairwise voting is not suitable in this application because we must make sure that the LMR tags assigned to consecutive words are compatible. For example, an LM tag cannot immediately follow an MM. Pairwise voting does not use any contextual information, so it cannot prevent incompatible tags from occurring. Therefore, in our experiments described here, we use the Transformation-Based Learning (Brill, 1995) to combine the results of two MEMM taggers. The feature set used in the TBL algorithm is similar to those used in the NP Chunking task in (Ngai and Florian, 2001).

## 3 Experiments

We conducted closed track experiments on three data sources: the Academia Sinica (AS) corpus, the Beijing University (PKU) corpus and the Hong Kong City University (CityU) corpus. We first split the training data from each of the three sources into two portions. $9/10$ of the official training data is used to train the MEMM taggers, and the other $1/10$ is held out as the development test data (the *development set*). The *development set* is used to estimate the optimal number of iterations in the MEMM training. Figure (1), (2) and (3) show the curves of F-scores on the *development set* with respect to the number of iterations in MEMM training.

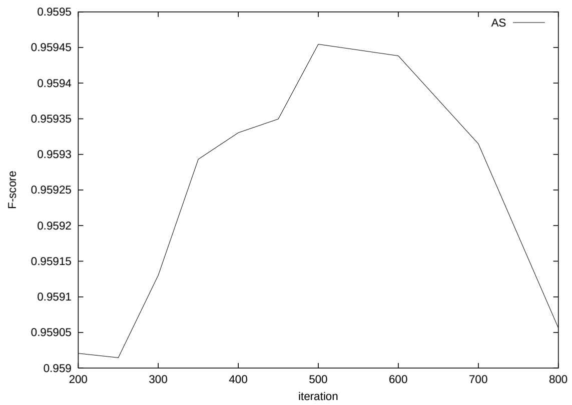


Figure 1: Learning curves on the development dataset of the Academia Sinica corpus. X-axis stands for the number of iteration in training. Y-axis stands for the $F$-score.

Experiments show that the MEMM models

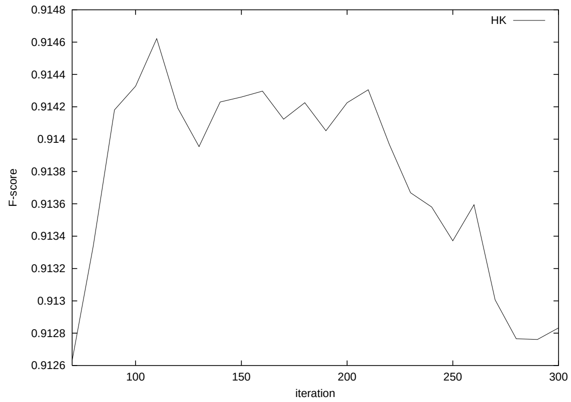


Figure 2: Learning curves on the development dataset of the HK City Univ. corpus.

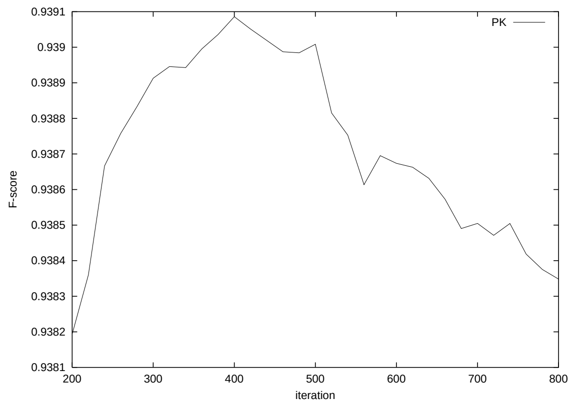


Figure 3: Learning curves on the development dataset of the Beijing Univ. corpus.

achieve the best results after 500 and 400 rounds (iterations) of training on the AS data and the PKU data respectively. However, the results on the CityU data is not very clear. From Round 100 through 200, the F-score on the *development set* almost stays unchanged. We think this is because the CityU data is from three different sources, which differ in the optimal number of iterations. We decided to train the MEMM taggers for 160 iterations the HK City University data.

We implemented two MEMM taggers, one scans the input from left to right and one from right to left. We then used these two MEMM taggers to tag both the training and the development data. We use the LMR tagging output to train a Transformation-Based learner, using *fast TBL* (Ngai and Florian, 2001). The middle in Table 2 shows the F-score on the *development set* achieved by the MEMM tagger that scans the input from left to right and the last column is the results after the Transformation-Based Learner is applied. The results show that using Transformation-Based learning only give rise to slight improvements. It seems that the bidirectional approach does not help much for the LMR tagging. Therefore, we only submitted the results of our left-to-right MEMM tagger, retrained on the entire training sets, as our official results.

| F-score | MEMM | MEMM+TBL |
|---|---|---|
| AS | 0.9595 | 0.9603 |
| HK | 0.9143 | N/A |
| PK | 0.9391 | 0.9398 |

Table 2: F-score on development data

The results on the official test data is similar to what we have got on our *development set*, except that the F-score on the Beijing Univ. corpus is over 2% lower in absolute accuracy than what we expected. The reason is that in the training data of Beijing University corpus, all the numbers are encoded in GBK, while in the test data many numbers are encoded in ASCII, which are unknown to our tagger. With this problem fixed, the results of the official test data are compatible with the results on our *development set*. However, we have withdrawn our segmentation results on the Beijing University corpus.

| corpus | R | P | F | $R_{OOV}$ | $R_{IV}$ |
|---|---|---|---|---|---|
| AS | 0.961 | 0.958 | 0.959 | 0.729 | 0.966 |
| HK | 0.917 | 0.915 | 0.916 | 0.670 | 0.936 |

Table 3: Official Bakeoff Outcome

## 4 Conclusions and Future Work

Our closed track experiments on the first Sighan Bakeoff data show that the LMR algorithm produces promising results. Our system ranks the second when tested on the Academia Sinica corpus and third on the Hong Kong City University corpus. In the future, we will try to incorporate a large word list into our tagger to test its performance on open track experiments. Its high accuracy on $R_{OOV}$ makes it a good candidate as a general purpose segmenter.

## References


E. Brill. 1995. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–565.

C. K. Fan and W. H. Tsai. 1988. Automatic word identification in chinese sentences by the relaxation technique. *Computer Processing of Chinese and Oriental Languages*, 4(1):33–56.

Kok-Wee Gan, Martha Palmer, and Kim-Teng Lua. 1996. A statistically emergent approach for language processing: Application to modeling context effects in ambiguous chinese word boundary perception. *Computational Linguistics*, 22(4):531–53.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for stgmentation and labeling sequence data. In *Proceedings of ICML 2001.*

G. Ngai and R. Florian. 2001. Transformation-based learning in the fast lane. In *Proceedings of NAACL-2001*, pages 40–47.

Adwait Ratnaparkhi. 1996. A maximum entropy part-of-speech tagger. In *Proceedings of the Empirical Methods in Natural Language Processing Conference*, University of Pennsylvania.

L. Shen and A. K. Joshi. 2003. A SNoW based supertagger with application to NP chunking. In *Proceedings of ACL 2003*.

R. Sproat, Chilin Shih, William Gale, and Nancy Chang. 1996. A stochastic finite-state word-segmentation algorithm for chinese. *Computational Linguistics*, 22(3):377–404.

H. van Halteren, J. Zavrel, and W. Daelmans. 1998. Improving data driven wordclass tagging by system combination. In *Proceedings of COLING-ACL 98.*

Andi Wu. 2003. Customizable segmentation of morphologically derived words in chinese. *Computational Linguistics and Chinese Language Processing*.

Nianwen Xue. 2003. Chinese word segmentation as character tagging. *Computational Linguistics and Chinese Language Processing*.