On Closed Task of Chinese Word Segmentation: An Improved CRF Model Coupled with Character Clustering and Automatically Generated Template Matching

Richard Tzong-Han Tsai, Hsieh-Chuan Hung, Cheng-Lung Sung, Hong-Jie Dai, and Wen-Lian Hsu
Intelligent Agent Systems Lab
Institute of Information Science, Academia Sinica
No. 128, Sec. 2, Academia Rd., 115 Nankang, Taipei, Taiwan, R.O.C.
{thtsai,yabt,clsung,hongjie,hsu}@iis.sinica.edu.tw

Abstract

This paper addresses two major problems in closed task of Chinese word segmentation (CWS): tagging sentences interspersed with non-Chinese words, and long named entity (NE) identification. To resolve the former, we apply K-means clustering to identify non-Chinese characters, and then adopt a two-tagger architecture: one for Chinese text and the other for non-Chinese text. For the latter problem, we apply postprocessing to our CWS output using automatically generated templates. The experiment results show that, when non-Chinese characters are sparse in the training corpus, our two-tagger method significantly improves the segmentation of sentences containing non-Chinese words. Identification of long NEs and long words is also enhanced by template-based post-processing. In the closed task of SIGHAN 2006 CWS, our system achieved F-scores of 0.957, 0.972, and 0.955 on the CKIP, CTU, and MSR corpora respectively.

1 Introduction

Unlike Western languages, Chinese does not have explicit word delimiters. Therefore, word segmentation (CWS) is essential for Chinese text processing or indexing. There are two main problems in the closed CWS task. The first is to identify and segment non-Chinese word sequences in Chinese documents, especially in a closed task (described later). A good CWS system should be able to handle Chinese texts peppered with non-Chinese words or phrases. Since non-Chinese language morphologies are quite different from that of Chinese, our approach must depend on how many non-Chinese words appear, whether they are connected with each other, and whether they are interleaved with Chinese words. If we can distinguish non-Chinese characters automatically and apply different strategies, the segmentation performance can be improved. The second problem in closed CWS is to correctly identify longer NEs. Most ML-based CWS systems use a five-character context window to determine the current character's tag. In the majority of cases, given the constraints of computational resources, this compromise is acceptable. However, limited by the window size, these systems often handle long words poorly.

In this paper, our goal is to construct a general CWS system that can deal with the above problems. We adopt CRF as our ML model.

2 Chinese Word Segmentation System

2.1 Conditional Random Fields

Conditional random fields (CRFs) are undirected graphical models trained to maximize a conditional probability (Lafferty *et al.*, 2001). A linear-chain CRF with parameters $\Lambda=\{\lambda_1, \lambda_2, \ldots\}$ defines a conditional probability for a state sequence $\mathbf{y} = y_1 \ldots y_T$, given that an input sequence $\mathbf{x} = x_1 \ldots x_T$ is

$$P_\Lambda(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z_\mathbf{x}} \exp\left(\sum_{t=1}^{T} \sum_k \lambda_k f_k(y_{t-1}, y_t, \mathbf{x}, t) \right) ,(1)$$

where $Z_\mathbf{x}$ is the normalization factor that makes the probability of all state sequences sum to one; $f_k(y_{t-1}, y_t, \mathbf{x}, t)$ is often a binary-valued feature function and λ_k is its weight. The feature functions can measure any aspect of a state transition, $y_{t-1} \rightarrow y_t$, and the entire observation sequence, x, centered at the current time step, t. For example, one feature function might have the value 1 when y_{t-1} is the state B, y_t is the state I, and x_t is the character "国".

2.2 Character Clustering

In many cases, Chinese sentences may be interspersed with non-Chinese words. In a closed task, there is no way of knowing how many languages there are in a given text. Our solution is to apply a clustering algorithm to find homogeneous characters belonging to the same character clusters. One general rule we adopted is that a language's characters tend to appear together in tokens. In addition, character clusters exhibit certain distinct properties. The first property is that the order of characters in some pairs can be interchanged. This is referred to as *exchangeability*. The second property is that some characters, such as lowercase characters, can appear in any position of a word; while others, such as uppercase characters, cannot. This is referred to as *location independence*. According to the general rule, we can calculate the pairing frequency of characters in tokens by checking all tokens in the corpus. Assuming the alphabet is Σ, we first need to represent each character as a |Σ|-dimensional vector. For each character c_i, we use v_j to represent its j-dimension value, which is calculated as follows:

$$v_j = \alpha + (1-\alpha)[\min(f_{ij}, f_{ji})]^{\gamma} , \quad (2),$$

where f_{ij} denotes the frequency with which c_i and c_j appear in the same word when c_i's position precedes that of c_j. We take the minimum value of f_{ij} and f_{ji} because even when c_i and c_j have a high co-occurrence frequency, if either f_{ij} or f_{ji} is low, then one order does not occur often, so v_j's value will be low. We use two parameters to normalize v_j within the range 0 to 1; α is used to enlarge the gap between non-zero and zero frequencies, and γ is used to weaken the influence of very high frequencies.

Next, we apply the K-means algorithm to generate candidate cluster sets composed of K clusters (Hartigan et al., 1979). Different K's, α's, and γ's are used to generate possible character cluster sets. Our K-means algorithm uses the cosine distance.

After obtaining the K clusters, we need to select the N_1 best character clusters among them. Assuming the angle between the cluster centroid vector and (1, 1, ..., 1) is θ, the cluster with the largest cosine θ will be removed. This is because characters whose co-occurrence frequencies are nearly all zero will be transformed into vectors very close to (α, α, ..., α); thus, their centroids will also be very close to (α, α, ..., α), leading to unreasonable clustering results.

After removing these two types of clusters, for each character c in a cluster M, we calculate the inverse relative distance (IRDist) of c using (3):

$$\text{IRDist}(c) = \log\left(\frac{\sum_i \cos(c, m_i)}{\cos(c, m)}\right) , \quad (3)$$

where m_i stands for the centroid of cluster M_i, and m stands for the centroid of M.

We then calculate the average inverse distance for each cluster M. The N_1 best clusters are selected from the original K clusters.

The above K-means clustering and character cluster selection steps are executed iteratively for each cluster set generated from K-means clustering with different K's, α's, and γ's.

After selecting the N_1 best clusters for each cluster set, we pool and rank them according to their inner ratios. Each cluster's inner ratio is calculated by the following formula:

$$\text{inner}(M) = \frac{\sum_{c_i, c_j \in M} \text{co-occurence}(c_i, c_j)}{\sum_{c_i, c_j} \text{co-occurence}(c_i, c_j)} , \quad (4)$$

where co-occurrence(c_i, c_j) denotes the frequency with which characters c_i and c_j co-occur in the same word.

To ensure that we select a balanced mix of clusters, for each character in an incoming cluster M, we use Algorithm 1 to check if the frequency of each character in $C \cup M$ is greater than a threshold τ.

Algorithm 1 Balanced Cluster Selection

```
Input: A set of character clusters $P=\{M_1, ..., M_K\}$
        Number of selections $N_2$,
Output: A set of clusters $Q=\{M'_1, ..., M'_{N_2}\}$.

1: $C=\{\}$
2: sort the clusters in $P$ by their inner ratios;
3: while |C|<=$N_2$ do
4:    pick the cluster $M$ that has highest inner ratio;
5:    for each character $c$ in $M$ do
6:        if the frequency of $c$ in $C \cup M$ is over threshold $\tau$
7:            $P \leftarrow P - M$;
8:            continue;
9 :       else
```

```
10:         $C \leftarrow C \cup M$;
11:         $P \leftarrow P - M$;
12:     end;
13:  end;
14: end
```

The above algorithm yields the best N_1 clusters in terms of exchangeability. Next, we execute the above procedures again to select the best N_2 clusters based on their location independence and exchangeability. However, for each character c_i, we use v_j to denote the value of its j-th dimension. We calculate v_j as follows:

$$v'_j = \alpha + (1-\alpha)[\min(\overline{f_{ij}}, f'_{ij}, \overline{f_{ji}}, f'_{ji})]^r, \qquad (5)$$

where $\overline{f_{ij}}$ stands for the frequency with which c_i and c_j appear in the same word when c_i is the first character; and f'_{ij} stands for the frequency with which c_i and c_j co-occur in the same word when c_i precedes c_j but not in the first position. We choose the minimum value from $\overline{f_{ij}}$, f'_{ij}, $\overline{f_{ji}}$, and f'_{ji} because if c_i and c_j both appear in the first position of a word and their order is exchangeable, the four frequency values, including the minimum value, will all be large enough.

Type	Cluster	Inner	(K, α, γ)
EX	,.0123456789	0.94	(10, 0.60, 0.16)
	-/ABCDEFGHIKLMNOPRSTUVWabcdefghiklmnoprstuvwxy	0.93	(10, 0.70, 0.16)
EL	－／ABCDEFGHIKLMNOPRSTUVWabcdefghiklmnoprstvwxy	0.84	(10, 0.50, 0.25)
	０１２３４５６７８９	0.76	(10, 0.50, 0.26)

Table 1. Clustering Results of the CTU corpus

Our next goal is to create the best hybrid of the above two cluster sets. The set selected for exchangeability is referred to as the EX set, while the set selected for both exchangeability and location independence is referred to as the EL set. We create a development set and use the best first strategy to build the optimal cluster set from EX∪EL. The EX and EL for the CTU corpus are shown in Table 1.

2.3 Handling Non-Chinese Words

Non-Chinese characters suffer from a serious data sparseness problem, since their frequencies are much lower than those of Chinese characters. In bigrams containing at least one non-Chinese character (referred as non-Chinese bigrams), the problem is more serious. Take the phrase "約莫 20 歲" (about 20 years old) for example. "2" is usually predicted as *I*, (i.e., "約莫" is connected with "2") resulting in incorrect segmentation, because the frequency of "2" in the *I* class is much higher than that of "2" in the *B* class, even though the feature $C_{-2}C_{-1}$="約莫" has a high weight for assigning "2" to the *B* class.

Traditional approaches to CWS only use one general tagger (referred as the G tagger) for segmentation. In our system, we use two CWS taggers. One is a general tagger, similar to the traditional approaches; the other is a specialized tagger designed to deal with non-Chinese words. We refer to the composite tagger (the general tagger plus the specialized tagger) as the GS tagger.

Here, we refer to all characters in the selected clusters as non-Chinese characters. In the development stage, the best-first feature selector determines which clusters will be used. Then, we convert each sentence in the training data and test data into a normalized sentence. Each non-Chinese character *c* is replaced by a cluster representative symbol σ_M, where *c* is in the cluster *M*. We refer to the string composed of all σ_M as *F*. If the length of *F* is more than that of *W*, it will be shortened to *W*. The normalized sentence is then placed in one file, and the non-Chinese character sequence is placed in another. Next, we use the normalized training and test file for the general tagger, and the non-Chinese sequence training and test file for the specialized tagger. Finally, the results of these two taggers are combined.

The advantage of this approach is that it resolves the data sparseness problem in non-Chinese bigrams. Consider the previous example in which σ stands for the numeral cluster. Since there is a phrase "約莫 8 年" in the training data, $C_{-1}C_0$= "莫 8" is still an unknown bigram using the G tagger. By using the GS tagger, however, "約莫 20 歲" and "約莫 8 年" will be converted as "約莫 σσ 歲" and "約莫 σ 年", respectively. Therefore, the bigram feature $C_{-1}C_0$="莫 σ" is no longer unknown. Also, since σ in "莫 σ" is tagged as *B*, (i.e., "莫" and "σ" are separated), "莫" and "σ" will be separated in "約莫 σσ 歲".

2.4 Generating and Applying Templates

Template Generation

We first extract all possible word candidates from the training set. Given a minimum word length *L*, we extract all words whose length is greater than or equal to *L*, after which we align all word pairs. For each pair, if more than fifty percent of the characters are identical, a template will be generated to match both words in the pair.

Template Filtering

We have two criteria for filtering the extracted templates. First, we test the matching accuracy of each template t on the development set. This is calculated by the following formula:

$$A(t) = \frac{\text{\# of matched strings with no separators}}{\text{\# of all matched strings}}.$$

In our system, templates whose accuracy is lower than the threshold τ_1 are discarded. For the remaining templates, we apply two different strategies. According to our observations of the development set, most templates whose accuracy is less than τ_2 are ineffective. To refine such templates, we employ the character class information generated by character clustering to impose a class limitation on certain template slots. This regulates the potential input and improves the precision. Consider a template with one or more wildcard slots. If any string matched with these wildcard slots contains characters in different clusters, this template is also discarded.

Template-Based Post-Processing (TBPP)

After the generated templates have been filtered, they are used to match our CWS output and check if the matched tokens can be combined into complete words. If a template's accuracy is greater than τ_2, then all separators within the matched strings will be eliminated; otherwise, for a template t with accuracy between τ_1 and τ_2, we eliminate all separators in its matched string if no substring matched with t's wildcard slots contains characters in different clusters. Resultant words of less than three characters in length are discarded because CRF performs well with such words.

3 Experiment

3.1 Dataset

We use the three larger corpora in SIGHAN Bakeoff 2006: a Simplified Chinese corpus provided by Microsoft Research Beijing, and two Traditional Chinese corpora provided by Academia Sinica in Taiwan and the City University of Hong Kong respectively. Details of each corpus are listed in Table 2.

Corpus	Training Size		Test Size	
	Types	Words	Types	Words
CKIP	141 K	5.45 M	19 K	122 K
City University (CTU)	69 K	1.46 M	9 K	41 K
Microsoft Research (MSR)	88 K	2.37 M	13 K	107 K

Table 2. Corpora Information

3.2 Results

Table 3 lists the best combination of n-gram features used in the G tagger.

Uni-gram	Bigram
C_{-2}, C_{-1}, C_0, C_1	$C_{-2}C_{-1}, C_{-1}C_0, C_0C_1, C_{-3}C_{-1}, C_{-2}C_0, C_{-1}C_1$

Table 3. Best Combination of N-gram Features

Table 4 compares the baseline G tagger and the enhanced GST tagger. We observe that the GST tagger outperforms the G tagger on all three corpora.

Conf	R	P	F	R_{OOV}	R_{IV}
CKIP-g	0.958	0.949	0.954	0.690	0.969
CKIP-gst	0.961	0.953	0.957	0.658	0.974
CTU-g	0.966	0.967	0.966	0.786	0.973
CTU-gst	0.973	0.972	0.972	0.787	0.981
MSR-g	0.949	0.957	0.953	0.673	0.959
MSR-gst	0.953	0.956	0.955	0.574	0.966

Table 4 Performance Comparison of the G Tagger and the GST Tagger

4 Conclusion

The contribution of this paper is two fold. First, we successfully apply the K-means algorithm to character clustering and develop several cluster set selection algorithms for our GS tagger. This significantly improves the handling of sentences containing non-Chinese words as well as the overall performance. Second, we develop a post-processing method that compensates for the weakness of ML-based CWS on longer words.

References

Hartigan, J. A., & Wong, M. A. (1979). A K-means Clustering Algorithm. *Applied Statistics, 28*, 100-108.

Lafferty, J., McCallum, A., & Pereira, F. (2001). *Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data.* Paper presented at the ICML-01.