Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)

Joint Word Segmentation, POS-Tagging and Syntactic Chunking*

Chen Lyu[1], **Yue Zhang**[2] and **Donghong Ji**[1†]
[1] Computer School, WuHan University, WuHan 430072, China
[2] Singapore University of Technology and Design, Singapore
{lvchen1989, dhji}@whu.edu.cn, yue_zhang@sutd.edu.sg

Abstract

Chinese chunking has traditionally been solved by assuming gold standard word segmentation. We find that the accuracies drop drastically when automatic segmentation is used. Inspired by the fact that chunking knowledge can potentially improve segmentation, we explore a joint model that performs segmentation, POS-tagging and chunking simultaneously. In addition, to address the sparsity of full chunk features, we employ a semi-supervised method to derive chunk cluster features from large-scale automatically-chunked data. Results show the effectiveness of the joint model with semi-supervised features.

1 Introduction

Chunking is a standard task in natural language processing, with applications to a variety of common tasks, such as syntactic analysis (Zhou, Qu, and Zhang 2012), named entity recognition (Wang, Che, and Manning 2013) and fine-grained sentiment (Yang and Cardie 2013). A challenge unique to character-based languages such as Chinese, Japanese and Thai is that word segmentation is a necessary pre-processing step. Chunking is typically performed on segmentation and POS-tagging outputs, which suffer from error propagation. In this paper, we take syntactic chunking as the task and explore joint models to address the issue. The main reason for our choice is the availability of widely-used benchmark data, and the method can be applied to all chunking tasks, such as name entity and opinion expression chunks.

Syntactic Chunking is also called shallow parsing. The task is to divide a text into syntactically correlated non-overlapping chunks (Abney 1991). Chinese chunking is a difficult task, which was conventionally defined over words, requiring word segmentation and POS-tagging as pre-processing steps (Chen, Zhang, and Isahara 2006; Li, Webster, and Yao 2003; Tan et al. 2004; Tan, Yao, and Chen 2005; Wu et al. 2005; Zhao et al. 2000; Zhou, Qu, and Zhang 2012).

*Work done while the first author was visiting SUTD.

†Corresponding author

Most previous work assumes that gold standard segmentation and POS-tagging have been given before chunking. The performance of typical chunking systems can reach around 91% on the gold input (Chen, Zhang, and Isahara 2006; Zhou, Qu, and Zhang 2012). Our experiments on the standard test corpus, with automatically assigned word segmentation and POS-tags, show that the performance of the chunking system is only around 70%. The contrast demonstrates that there is still a large gap for improvement in realistic settings.

Given a Chinese sentence, word segmentation errors can propagate to later processing stages, namely POS tagging and chunking. On the other hand, chunking information, which consist of chunk boundaries and corresponding chunk types, can also be used to improve the word segmentation and POS-tagging. As a result, the three tasks can be jointly performed to bring potential accuracy improvements on all tasks. Similar ideas have been exploited for joint segmentation and POS-tagging (Zhang and Clark 2010), joint named entity recognition and parsing (Finkel and Manning 2009), and joint segmentation, POS-tagging, and parsing (Hatori et al. 2012; Li and Zhou 2012; Zhang et al. 2013).

In this paper, we make an investigation of character-level Chinese chunking based on a joint segmentation and POS-tagging framework of Zhang and Clark (2010). The character-level chunking system performs word segmentation, part-of-speech (POS) tagging and chunking jointly. Compared to a pipeline system, the advantages of a joint system include reduction of error propagation, and the integration of segmentation, POS tagging and chunk-level features. To our knowledge, this is the first work to develop a system that jointly performs the above three tasks.

Word features, especially chunk-level features, which are used in the transition-based chunking systems, are relatively sparse. The state-of-the-art models (Chen, Zhang, and Isahara 2006; Zhou, Qu, and Zhang 2012) are usually trained on corpora automatically extracted from the Penn Chinese Treebank (CTB) (Nianwen et al. 2000), which consists of tens of thousands of sentences. Given such a limited set of training data (e.g. tens of thousands sentences), the chance of a sparse feature occurring in the training data but not in the test data can be high. As a result, full chunk features are not useful in our experiments. To address this issue, we present a semi-supervised approach that extracts chunk cluster features from large automatically-chunked data to improve chunking.

Standard evaluation shows that the joint model outperforms a baseline pipeline system that consists of a joint segmentation and POS-tagging system, and a transition-based chunker that takes segmented and POS-tagged inputs, both components giving competitive results. In addition, semi-supervised learning effectively reduces the sparsity of full chunk features, which are not useful in fully supervised systems. Our semi-supervised chunk cluster features improve the accuracies of both chunking and segmentation/POS-tagging in the joint chunking system.

2 Related Work

Chunking research started with English, with the CoNLL-2000 offering a platform for system comparison (Sang and Buchholz. 2000). The benchmark data was generated from the Penn Treebank. Most previous work reduced chunking to sequence labeling problems. Classification models, including SVMs (Kudo and Matsumoto 2001) and other classifiers (Zhang, Damerau, and Johnson 2002), have been used. Kudo and Matsumoto (2000) applied combinations of multiple SVMs classifiers to English chunking and achieved the best performance in the CoNLL2000 shared task. Sequence labeling models, such as CRFs, were widely used for chunking, and gave state-of-the-art results (Sha and Pereira 2003; Mcdonald, Crammer, and Pereira 2005).

Similar approaches, including classification models and sequence labeling models, were also applied for Chinese chunking (Li, Webster, and Yao 2003; Tan et al. 2004; Tan, Yao, and Chen 2005; Wu et al. 2005; Zhao et al. 2000). Chen, Zhang and Isahara (2006) used CTB4 as a standard benchmark to compare the performances of several state-of-the-art models for Chinese chunking, and proposed ensemble voting methods to improve performance.

Most previous work focused on gold segmentation and POS-tagging input, while there are also some research exploiting the chunking systems that take input with gold segmentation and automatically assigned POS-tags (Sun et al. 2008; Yao, Li, and Huang 2007). To our knowledge, our work is the first to perform chunking on raw text, without assuming that the input has been assigned with gold segmentation.

In terms of features, most previous work performed chunking with word-level features. Zhou, Qu and Zhang (2012) utilized chunk-level features, and achieved state-of-the-art performance. Most of their features are based words within chunks, such as the first word and the last of the chunk. We find that full chunk features are not useful in our fully supervised model due to their sparseness. Yao, Li, and Huang (2007) utilized distributional similarity-based features to solve the sparsity of word features. To our knowledge, our semi-supervised model is the first to solve the sparsity problem of full chunk features.

3 Baseline systems

We use a transition-based model for the baseline chunking system. It consists of a joint word segmentation and POS-tagging model (Zhang and Clark 2010) and a word-based chunker. Both systems are built using the global discriminative learning for structured prediction, and beam search framework of Zhang and Clark (2011).

3.1 Joint Word Segmentation and POS-Tagging

We apply Zhang and Clark (2010) as the baseline word segmentation and POS-tagging system. Because the training and decoding framework is also used for our baseline chunker and joint chunker, we give the model description and training algorithm in details in their generic forms.

The Transition System Given an input sentence, the joint segmentor and POS-tagger builds an output incrementally, one character at a time. At each step, each character can either be attached to the current word or separated as the start of a new word. When the current character starts a new word, a POS-tag is assigned to the new word. When the character is concatenated to a word, the POS-tag of the word remains unchanged.

Formally, a state item in the transition system consists a stack and a queue. The stack contains partially segmented and tagged sentence, and the queue consists of the unprocessed character sequence. The candidate transition action at each step is defined as follows:

- SEPARATE(TAG): remove the front character from the queue, and add it as the start of a new word with the POS *TAG* on the stack.
- APPEND: remove the front character from the queue, and append it to the last partial word on the stack.

Given the sentence “他到达北京机场。” (He reached Beijing airport), the sequence of actions SEPARATE(NR) - SEPARATE(VV) - APPEND - SEPARATE(NR) - APPEND - SEPARATE(NN) - APPEND - SEPARATE(PU) can be used to analyze its structure.

Model A linear model is used to score both partial and full candidate outputs. Given an input x, the score of a candidate output y is computed as:

$$Score(y) = \Phi(y) \cdot \vec{w} \qquad (1)$$

where $\Phi(y)$ is the global feature vector extracted from y, and $\vec{w}$ is the parameter vector of the model.

Given a partial or complete candidate y, its global feature vector $\Phi(y)$ is extracted by instantiating all applicable feature templates for each character in y. The feature templates are taken from Zhang and Clark (2010). Table 1 summarizes the features, where *w*, *t* and *c* are used to represent a word, a POS-tag and a character, respectively.

The subscripts are based on the current character. c_0, c_{-1} and c_{-2} represent the current character and its previous two characters, respectively; w_{-1} and w_{-2} represent the previous two words to the current character, respectively; t_0, t_{-1} and t_{-2} represent the POS tags of the current word and the previous two words, respectively. $start(w)$, $end(w)$ and $len(w)$ represent the first character, the last character and the length of word w, respectively.

ID	**Feature Templates**
1	w_{-1}
2	$w_{-2} \cdot w_{-1}$
3	w_{-1}, where $len(w_{-1}) = 1$
4	$start(w_{-1}) \cdot len(w_{-1})$
5	$end(w_{-1}) \cdot len(w_{-1})$
6	$end(w_{-1}) \cdot c_0$
7	$c_{-1} \cdot c_0$
8	$start(w_{-1}) \cdot end(w_{-1})$
9	$w_{-1} \cdot c_0$
10	$end(w_{-2}) \cdot w_{-1}$
11	$start(w_{-1}) \cdot c_0$
12	$end(w_{-2}) \cdot end(w_{-1})$
13	$w_{-2} \cdot len(w_{-1})$
14	$len(w_{-2}) \cdot w_{-1}$
15	$w_{-1} \cdot t_{-1}$
16	$t_{-1} \cdot t_0$
17	$t_{-2} \cdot t_{-1} \cdot t_0$
18	$w_{-1} \cdot t_0$
19	$t_{-2} \cdot w_{-1}$
20	$w_{-1} \cdot t_{-1} \cdot end(w_{-2})$
21	$w_{-1} \cdot t_{-1} \cdot c_0$
22	$c_{-2} \cdot c_{-1} \cdot c_0 \cdot t_{-1}$, where $len(w_{-1}) = 1$
23	$start(w_0) \cdot t_0$
24	$t_{-1} \cdot start(w_{-1})$
25	$t_0 \cdot c_0$
26	$t_0 \cdot c_0 \cdot start(w_0)$
27	$c \cdot t_{-1} \cdot end(w_{-1})$, where $c \in w_{-1}$ and $c \neq end(w_{-1})$
28	$c_0 \cdot t_0 \cdot c_{-1} \cdot t_{-1}$
29	$c_0 \cdot t_0 \cdot c_{-1}$

Table 1: Feature templates for the joint word segmentation and POS-tagging system (Zhang and Clark 2010).

Decoding We apply beam-search for decoding. An agenda is used to keep the N-best partial outputs at each incremental step. Before decoding starts, the agenda is initialized with the start state item. During the incremental process, existing candidates are extended in all possible ways, and the *N*-best newly generated candidates are used for the next incremental step. When the process is complete, the highest-scored candidate from the agenda is taken as the output.

Pseudocode for the beam-search algorithm is given in Algorithm 1, where the variable *problem* represents a particular task, such as joint word segmentation and POS-tagging, and the variable *candidate* represents a state item, which has a different definition for each task. For example, for the joint word segmentation and POS-tagging task, a candidate is a pair, consisting of the partially segmented and tagged sentence and the remaining input character sequence. The agenda is an ordered list, used to keep all the state items generated at each stage, ordered by the score. The variable *candidates* is the set of state items that can be used to generate new state items, namely the N-best state items from the previous stage. *N* is the number of state items retained at each stage.

STARTITEM initializes the start state item according to the problem; for the joint word segmentation and POS-tagging task, the start state item is a pair consisting of an empty sentence and the complete sequence of characters waiting to be

Algorithm 1 Beam Search Decoding

```
 Input: $problem, agenda, candidates, N$
Output: the highest-scored final state
 1: candidates ← STARTITEM(problem)
 2: agenda ← CLEAR(agenda)
 3: for loop do
 4:     for $s$ in $candidates$ do
 5:         for $action$ in GETACTIONS(s) do
 6:             $agenda \leftarrow$ APPLY($s, action$)
 7:         end for
 8:     end for
 9:     best ← TOP(agenda)
10:     if GOALTEST(problem, best) then
11:         return $best$
12:     end if
13:     candidates ← TOP-N(agenda, N)
14:     agenda ← CLEAR(agenda)
15: end for
```

processed. CLEAR removes all items from the agenda.

GETACTIONS represents all possible actions which one candidate *s* can take to generate new state items, for the joint word segmentation and POS-tagging task, GETACTIONS returns $\{$SEPARATE(TAG), APPEND$\}$. APPLY represents an incremental processing step, which takes a state item *s* and generates new state items from it according to the *action*, and then puts new state items onto the agenda.

TOP returns the highest scoring state item on the agenda. GOALTEST checks whether the incremental decoding process is completed; for the joint word segmentation and POS-tagging task, the process is completed if the state item consists of a fully segmented and tagged sentence and an empty remaining character sequence. TOP-N returns the N-highest scoring state items on the agenda, which are used for the next incremental step.

Training The learning algorithm is based on the generalized perceptron (Collins 2002). Parameter adjustments can be performed at any character during the decoding process, using the early update mechanism (Collins and Roark 2004; Zhang and Clark 2011).

3.2 Word-based Chinese Chunking

We use the same framework as described in Section 3.1 for the baseline word-based chunker. To our knowledge, we are the first to report a transition-based syntactic chunker in the literature, scoring transition action sequences instead of output sentences directly, which most previous work does.

The Transition System Similar to the joint word segmentation and POS-tagging model, the word-based chunking model builds an output incrementally, one word at a time. At each step, each word can either be attached to the current chunk or separated as the start a new chunk. When the current word starts a new chunk, a chunk type is assigned to the new chunk. When the word is concatenated to a chunk, the chunk type of the chunk remains unchanged.

step	action	stack	queue
0	-	Φ	他/NR
1	SEP(NP)	[NP 他/NR]	到达/VV
2	SEP(VP)	[NP 他/NR][VP 到达/VV]	北京/NR
3	SEP(NP)	[NP 他/NR][VP 到达/VV] [NP 北京/NR]	机场/NN
4	APP(NP)	[NP 他/NR][VP 到达/VV] [NP 北京/NR 机场/NN]	。/PU
5	SEP(O)	[NP 他/NR][VP 到达/VV] [NP 北京/NR 机场/NN] [O 。/PU]	Φ

Table 2: Word-based chunking example.

The state item in the transition system consists a stack and a queue. The stack contains partially chunked sentence, and the queue consists of the unprocessed word sequence. The candidate transition action at each step is defined as follows:

- SEPARATE(TYPE): remove the front word from the queue, and add it as the start of a new chunk with the label *TYPE* on the stack.
- APPEND: remove the front word from the queue, and append it to the last partial chunk on the stack.

Table 2 gives an example action sequence for the sentence “他/NR(He) 到达/VV(reached) 北京/NR(Beijing) 机场/NN(airport)。”.

The same linear model from Section 3.1 is applied to score candidate outputs. Table 3 summarizes the feature templates. Some templates are adapted from Zhou, Qu and Zhang (2012). In the feature templates, *C*, *T*, *w* and *t* are used to represent a chunk, a chunk type, a word and a POS tag, respectively. *N* is the queue of incoming words. The subscripts are based on the current word. C_0 and C_{-1} represent the current chunk and the previous chunk, respectively. T_0 and T_{-1} represent the chunk types of the current chunk and the previous chunk, respectively. N_0, N_1, and N_2 represent the front items from the queue.

Label(w) represents the position of the word *w* in the current chunk (here refers to label ”B” and ”I” in a BI sequence labeling tag set). *Bigram(w)* denotes the word to the left of *w* and the one to the right of *w*. And the similar meaning is for *biPOS(w)*. *POSset(C)* represents the sequence of POS tags in chunk *C*. *start_word(C)*, *end_word(C)* and *len(C)* represent the first word, the last word, and the length of chunk *C*, respectively. Similarly, *start_POS(C)* and *end_POS(C)* represent the POS tags of the first word and the last word in chunk *C*, respectively.

The training method described in Section 3.1 is used. For the decoding process, GETACTIONS returns $\{$SEPARATE(TYPE), APPEND$\}$.

ID	Feature Templates
1	N_0w
2	N_0t
3	N_1w
4	N_1t
5	N_2w
6	N_2t
7	$N_0w \cdot N_0t$
8	$N_1w \cdot N_1t$
9	$N_2w \cdot N_2t$
10	$N_0w \cdot N_1w$
11	$N_0w \cdot N_1t$
12	$N_0t \cdot N_1w$
13	$N_0w \cdot N_1w \cdot N_0t$
14	$N_0w \cdot N_1w \cdot N_1t$
15	$N_1w \cdot N_2w$
16	$N_1w \cdot N_2t$
17	$N_1t \cdot N_2w$
18	$N_1t \cdot N_2t$
19	$w_1 \cdot N_0 \cdot T_0$, where $len(C_0) = 1$
20	$start\_word(C_0)T_0$
21	$start\_POS(C_0)T_0$
22	$end\_word(C_0)T_0$
23	$end\_POS(C_0)T_0$
24	$w \cdot end\_word(C_0) \cdot T_0$ where $w \in C_0$ and $w \neq end\_word(C_0)$
25	$t \cdot end\_POS(C_0) \cdot T_0$ where $t \in POSset(C_0)$ and $p \neq end\_POS(C_0)$
26	$w \cdot label(w) \cdot T_0$ for all w in C_0
27	$bigram(w) \cdot label(w) \cdot T_0$ for all w in C_0
28	$biPOS(w) \cdot label(w) \cdot T_0$ for all w in C_0
29	$POSset(C_0) \cdot T_0$
30	$T_0 \cdot T_{-1}$
31	$end\_word(C_{-1}) \cdot T_{-1} \cdot start\_word(C_0) \cdot T_0$
32	$end\_word(C_{-1}) \cdot T_{-1} \cdot end\_word(C_0) \cdot T_0$
33	$start\_word(C_{-1}) \cdot T_{-1} \cdot start\_word(C_0) \cdot T_0$
34	$end\_POS(C_{-1}) \cdot T_{-1} \cdot start\_POS(C_0) \cdot T_0$
35	$end\_POS(C_{-1}) \cdot T_{-1} \cdot end\_POS(C_0) \cdot T_0$
36	$start\_POS(C_{-1}) \cdot T_{-1} \cdot start\_POS(C_0) \cdot T_0$
37	$end\_word(C_{-1}) \cdot T_0$; $end\_POS(C_{-1}) \cdot T_0$
38	$T_{-1} \cdot T_0 \cdot start\_word(C_0)$
39	$T_{-1} \cdot T_0 \cdot start\_POS(C_0)$
40	$POSset(C_{-1}) \cdot T_{-1} \cdot POSset(C_0) \cdot T_0$

Table 3: Feature templates for word-based chunking.

4 Character-Level Chinese Chunking

We develop a character-based chunking model under transition-based framework, which jointly performs word segmentation, POS tagging and chunking, by making a change to the state item of the baseline chunker, adding a deque between the stack and the queue to save partial segmented and POS-tagged results.

The mode use two types of transition actions, one for joint word segmentation and POS-tagging and the other for chunking. The joint segmentation and POS-tagging actions operate between the deque and the queue, while the chunking actions operate between the stack and the deque. The candidate transition action at each step is defined as follows:

- SEPARATE(TAG): remove the front character from the queue, and add it as the start of a new word with the POS *TAG* on the deque. This action can only be applied when the length of the deque is less than *t* and the last word in the deque is a full word.
- APPENDWORD: remove the front character from the queue, and append it to the last partial word on the deque.

step	action	stack	deque	queue
0	-	Φ	Φ	他 到
1	SEP(NR)	Φ	他/NR	到 达
2	FINW	Φ	他/NR	到 达
3	SEP(NP)	[NP 他/NR]	Φ	到 达
4	SEP(VV)	[NP 他/NR]	到/VV	达 北
5	APPW	[NP 他/NR]	到达/VV	北 京
6	FINW	[NP 他/NR]	到达/VV	北 京
...	...	...	...	
15	APPC	[NP 他/NR] [VP 到达/VV] [NP 北京/NR 机场/NN]	Φ	。
...	...	...	...	

Table 4: Character-based chunking example.

This action can only be applied when the length of the deque is less than *t* and the last word in the deque is a partial word.

- FINISHWORD: mark the last partial word on the deque as a full word. The action is necessary for counting the number of full words in the deque.
- SEPARATE(TYPE): remove the front word from the deque, and add it as the start of a new chunk with the label *TYPE* on the stack. This action can only be applied when the length of the deque is equal with *t*.
- APPENDCHUNK: remove the front word from the deque, and append it to the last partial chunk on the stack. This action can only be applied when the length of the deque is equal with *t*.
- FINISH: finalized the state item. This action can only be applied when both the deque and queue are empty.

Table 4 gives an example action sequence for the sentence "他到达北京机场。" (He reached Beijing airport).

The same linear model from Section 3.1 is applied to score the candidate outputs. The feature templates of our character-based chunking model contains the union of the baseline chunking features in Table 3, and the baseline joint word segmentation and POS tagging features in (Zhang and Clark 2010).

The training method described in Section 3.1 is used. For the decoding process, GETACTIONS returns {SEPARATE(TAG), APPENDWORD, FINISHWORD, SEPARATE(TYPE), APPENDCHUNK, FINISH}.

5 Semi-supervised model

In order to make use full chunk features to improve accuracies, we exploit semi-supervised learning to derive such features with less sparsity. In particular, we use our joint model to segment, POS-tag and chunk raw text, and extract semi-supervised chunk cluster features from the automatically-analyzed data according to the feature templates.

5.1 Features Extraction

We follow the method of Chen et al. (2009) for semi-supervised feature extraction. First, we extract chunks from the automatically-chunked data, and then merge the same chunks into one entry in order to count their frequency. We eliminate all chunks that occur only once in the data.

The extracted chunks are grouped into three sets corresponding to three levels of frequency: "high-frequency (HF)", "middle-frequency (MF)", and "low-frequency (LF)", which correspond to chunks in the TOP-10% most frequent chunks, the TOP-20% chunks and the other chunks, respectively. We store the set ID for each chunk in a map MAP_c. During testing, if a chunk is not included in MAP_c, its set ID is ZERO. Therefore, we have four labels: HF, MF, LF, and ZERO for one chunk.

We generate chunk cluster features for the chunk-based feature templates, replacing the chunk with the indicator functions for set IDs of the retrieved chunk in the chunking process. The set IDs are much less sparse compared to full chunk features. We also extract word bigram list and generate bigram-based features from unlabeled data using the same method.

To our knowledge, we are the first to apply semi-supervised method of Chen et al. (2009) to a transition-based model. When applied to a graph-based model, the mechanism of the cluster features is easy to understand, which scores contextual patterns directly. Here we give some intuition on how it works on our transition-based model. Considering the chunk-based feature templates in the decoding process, if C_0 in the candidate is labeled as HF, the model tends to apply the SEPARATE(TYPE) action to the candidate, and therefore the weight of the SEPARATE(TYPE) action should be higher in the model. In this way, the semi-supervised chunk cluster features can improve the chunking performance in the transition-based model. Similarity, semi-supervised word bigram features can improve word segmentation in the transition-based model.

5.2 Feature Templates

Table 5 summarizes the feature templates. Similar to Table 3, *C*, *T* and *w* are used to represent a chunk, a chunk type and a word, respectively. *N* is the queue of incoming words. C_0 and C_{-1} represent the current chunk and the previous chunk, respectively. T_0 and T_{-1} represent the chunk types of the current chunk and the previous chunk, respectively. w_{-1} and w_{-2} represent the previous two words to the current character, respectively. N_0 represents the front of the queue. *POSset(C)* represents the sequence of POS tags in chunk *C*. *start_word(C)*, *end_word(C)* and *len(C)* represent the first word, the last word, and the length of chunk *C*, respectively. None of these features are included in Table 3, because they do not give higher accuracies. All the features all contain full chunk or word bigram information, and therefore are highly sparse.

6 Experiments

6.1 Experimental Settings

Unlike English chunking, there is not a most commonly used benchmark corpus for Chinese chunking. We follow Chen, Zhang and Isahara (2006) in dataset selection and

ID	**Feature Templates**
1	C_0
2	$C_0 \cdot T_0$
3	$C_0 \cdot POSset(C_0)$
4	C_0, where $len(C_0) = 1$
5	$C_0 \cdot N_0w$
6	$C_0 \cdot N_0w \cdot T_0$
7	$C_{-1} \cdot C_0$
8	$T_{-1} \cdot C_0$
9	$C_{-1} \cdot T_0$
10	$C_0 \cdot end\_word(C_{-1})$
11	$C_{-1} \cdot len(C_0)$
12	$C_0 \cdot len(C_{-1})$
13	$C_0 \cdot end\_word(C_{-1}) \cdot T0$
14	$C_{-1} \cdot T_{-1} \cdot C_0 \cdot T_0$
15	$w_{-2} \cdot w_{-1}$

Table 5: Additional feature templates for the semi-supervised chunking model.

	Sections	**Sentences**	**Words**
Training	1-300 326-899	9,528	232,085
Dev	301-325	350	6,821
Test	900-1078	5,290	165,862

Table 6: Statistics of the CTB4 corpus.

chunk type defination. The chunking corpus can be extracted from CTB4 with a public tool[1]. We conduct our experiments on the CTB4 corpus following previous studies on Chinese chunking (Chen, Zhang, and Isahara 2006; Zhou, Qu, and Zhang 2012). We split the corpus according to previous work, using files (FID from 301-325) from the training set as the development set from the training set. Table 6 lists the details about the CTB4 data used in this experiment.

We use wikidump20150325[2] as the raw text for the semi-supervised model. All the traditional Chinese pages in Wikipedia are converted to simplified Chinese using the tool OpenCC[3] . After removing duplication, 5.3 million sentences are reserved.

We evaluated the results in the same way as the CONLL2000 share-task, using precision P and recall R. The F1 score is given by $F1 = 2 \times P \times R/(P + R)$.

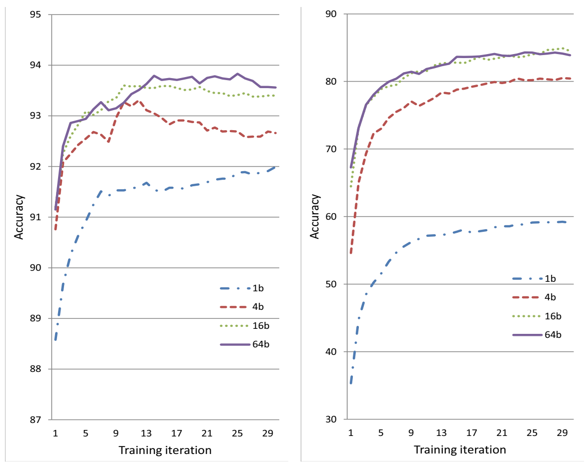
6.2 Development Results

Our development tests are mainly used to decide the size of the beam *b* and the deque size *t* in the transition system. We set the number of training iterations to 30 for all the experiments.

Influence of Deque Size We first adjust the deque size *t* for our character-level chunking model. In this step, we set the beam size *b=16* and the training iteration *n=30*. Table 7 shows the results, according to which we set *t* = 4 for the character-level chunking model for the other experiments.

	SEG	**POS**	**CHUNK**
t=1	95.75	**92.73**	83.77
t=2	95.68	92.40	83.44
t=3	95.50	92.20	83.30
t=4	**95.81**	92.65	**84.52**
t=5	95.70	92.39	83.74

Table 7: The influence of deque size.



(a) Word-based. (b) Character-based.

Figure 1: The influence of beam size.

Influence of Beam Size We use a beam size of 16 for joint segmentation and POS-tagging according to previous work (Zhang and Clark 2010). Figure 1(a) shows the accuracy curves of our word-based chunking model using different beam sizes with respect to the number of training iterations. We use gold word segmentation and POS-tags in this step. Figure 1(b) shows the accuracy curves for our character-level chunking model using different beam sizes with respect to the number of training iterations. We set the beam sizes of both chunkers to 64 according to the the figures.

Comparison with Other Word-based Chinese Chunking Models Chen, Zhang and Isahara (2006) compared the performance of some state-of-the-art machine learning models, including SVMs, CRFs, transformation-based learning and memory-based learning for Chinese chunking on the CTB4 corpus. They proposed ensemble voting methods to improve performance. We evaluate our word-based chunking model using the same dataset and compare our results with Chen, Zhang and Isahara (2006) and Zhou, Qu and Zhang (2012).

Table 8 shows the results. The results of CRFs-based and SVMs-based chunking systems are taken from Chen, Zhang and Isahara (2006). **Baseline** refers to our word-based chunker and **Pipeline** refers to our pipeline chunking model. Our word-based chunking model give comparable results on the

[1]https://github.com/rainarch/ChunkLinkCTB
[2]http://download.wikipedia.com/zhwiki/20150325/
[3]https://github.com/BYVoid/OpenCC

Method	**CHUNK**
CRFs	90.74
SVMs	91.46
Chen, Zhang and Isahara (2006)	91.68
Zhou, Qu and Zhang (2012)	92.11
Our Baseline	91.43
Pipeline	69.02

Table 8: Results of word-based chunking.

	SEG	**POS**	**CHUNK**
Supervised	89.85	81.94	70.96
Semi-ALL	**91.00**	**82.71**	**72.29**
Semi-C	90.67	82.45	72.09
Semi-C_0	90.71	82.59	71.98
Semi-W	90.72	82.53	71.62

Table 9: Results of the semi-supervised models.

same input with gold segmentation and POS-taggings.

When our word-based chunking model is applied to the automatically segmented and POS-tagged input, the chunking performance decrease significantly. The contrast shows the significance of segmentation errors to chunking accuracies in the realistic setting. To our knowledge, this is the first work to evaluate a pipeline chunking system.

6.3 Results of Semi-supervised Model

We explore the performance of semi-supervised model using different feature templates. The models are listed below:

- **Supervised**: the joint model using the feature templates describe in Section 4.
- **Semi-ALL**: **Supervised** + feature templates in Table 5.
- **Semi-C**: **Supervised** + feature templates 1-14 in Table 5.
- **Semi-C_0**: **Supervised** + feature templates 1 in Table 5.
- **Semi-W**: **Supervised** + feature templates 15 in Table 5.

Table 9 shows the results of the above models on the test data. Chunk cluster features, especially the C_0 feature, improve the chunking performance of the semi-supervised model. When all the semi-supervised features are applied, the **Semi-ALL** system improves the chunking F-score from 70.96% to 72.29%. The results is significant at $p < 10^{-3}$ by pair-wise t-test. In addition, semi-supervised word bigram features also improve the word segmentation performance.

6.4 Comparison between the pipeline and joint models

We apply the semi-supervised method on the pipeline model and the joint model, and explore the effects of the full chunk features.

The models are listed below:

- **Pipeline**: a joint word segmentation and POS-tagging model and a word-based chunker, using feature templates in Table 1 and Table 3, respectively. For the baseline word-based chunker, we assign automatic POS to the training data by 10-way jackknifing.
- **Pipeline-C**: **Pipeline** + feature templates 1-14 in Table 5 for the baseline chunker.
- **Pipeline-Semi-C**: **Pipeline** + templates 1-14 in Table 5 for the semi-supervised model.
- **Joint**: character-level chunking model, using feature templates describe in Section 4.
- **Joint-C**: **Joint** + feature templates 1-14 in Table 5 for the joint chunker.
- **Joint-Semi-C**: **Joint** + feature templates 1-14 in Table 5 for the semi-supervised model.

	SEG	**POS**	**CHUNK**
Pipeline	88.81	80.64	69.02
Pipeline-C	88.81	80.64	68.82
Pipeline-Semi-C	88.81	80.64	69.45
Joint	89.85	81.94	70.96
Joint-C	89.83	81.78	70.63
Joint-Semi-C	**90.67**	**82.45**	**72.09**

Table 10: Comparison between the pipeline and joint models.

Table 10 shows the results of the above models on the test data. From the table, we can see that the joint model is better than the pipeline model on chunking performance, and also give a higher word segmentation and POS-tagging performance. Without using semi-supervised features, the joint model improves the segmentation, POS-tagging and chunking accuracies significantly ($p < 10^{-3}$) from 88.81%, 80.64% and 69.02% to 89.85%, 81.94% and 70.96%, respectively, compared with the pipeline model.

On the other hand, full chunk features can not improve the chunking performance in the supervised model due to its sparsity. In contrast, when applied to the semi-supervised model, they can improve not only the chunking performance, but also the word segmentation and POS-tagging performance of the joint model, showing the effectiveness of reducing feature sparsity. Compared with the joint model without using semi-supervised features, the joint model with all the semi-supervised model improves segmentation, POS-tagging and chunking accuracies significantly ($p < 10^{-3}$).

Compared to the pipelined baseline, our best model **Semi-ALL** gives an error reduction of 19.57%, 10.69% and 10.56% on word segmentation, POS-tagging and chunking, respectively.

7 Conclusions

We studied character-level Chinese chunking using the transition-based framework, which achieved better results compared with a pipelined baseline that performs joint segmentation and POS-tagging, before word-level chunking. Due to the sparsity of full chunk features, they are not effective in a fully supervised model. We exploited a semi-supervised method to solve the sparsity problem by leveraging large-scale automatically-chunked text. The semi-supervised chunk features improved the accuracies of both segmentation and chunking by our joint model.

Acknowledgments

We thank all reviewers for their detailed comments. This work is supported by the National Natural Science Foundation of China (No. 61173062, 61373108, 61133012, 61202193), the Major Projects of the National Social Science Foundation of China (No. 11&ZD189), the Key Program of Natural Science Foundation of Hubei, China (Grant No.2012FFA088), the China Scholarship Council and Singapore Ministry of Education (MOE) AcRF Tier 2 grant T2MOE201301.

References

Abney, S. P. 1991. Parsing by chunks. In *Robert C. Berwick, Steven P. Abney, and Carol Tenny, editors, Principle-Based Parsing*, 257–278.

Chen, W.; Kazama, J.; Uchimoto, K.; and Torisawa, K. 2009. Improving dependency parsing with subtrees from auto-parsed data. In *Proceedings of EMNLP*, 570–579.

Chen, W.; Zhang, Y.; and Isahara, H. 2006. An empirical study of chinese chunking. In *Proceddings of ACL*, 97–104.

Collins, M., and Roark, B. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of ACL*, 111–118.

Collins, M. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*, 1–8.

Finkel, J. R., and Manning, C. D. 2009. Nested named entity recognition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, 141–150.

Hatori, J.; Matsuzaki, T.; Miyao, Y.; and Tsujii, J. 2012. Incremental joint approach to word segmentation, pos tagging, and dependency parsing in chinese. In *Proceddings of ACL*, volume 1, 1045–1053.

Kudo, T., and Matsumoto, Y. 2000. Use of support vector learning for chunk identification. In *Proceedings of CoNLL-2000 and LLL-2000*, 142–144.

Kudo, T., and Matsumoto, Y. 2001. Chunking with support vector machines. *Journal of Natural Language Processing* 9(107):3–21.

Li, Z., and Zhou, G. 2012. Unified dependency parsing of chinese morphological and syntactic structures. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 1445–1454. Stroudsburg, PA, USA: Association for Computational Linguistics.

Li, H.; Webster, J. J.; and Yao, T. 2003. Transductive hmm based chinese text chunking. In *Proceedings of IEEE NLP-KE2003*, 257–262.

Mcdonald, R.; Crammer, K.; and Pereira, F. 2005. Flexible text segmentation with structured multilabel classification. In *In Proceddings of HLT-EMNLP*, 987–994.

Nianwen, X.; Fei, X.; Shizhe, H.; and Anthony, K. 2000. The bracketing guidelines for the penn chinese treebank. *Technical report, University of Pennsylvania.*

Sang, T. K., and Buchholz., S. 2000. Introduction to the conll-2000 shared task: Chunking. In *Proceedings of Conference on Computational Natural Language Learning Conll.*

Sha, F., and Pereira, F. 2003. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, 213–220.

Sun, G.; Liu, Y.; Qiao, P.; and Lang, F. 2008. Chinese chunking algorithm based on cascaded conditional random fields. In *Proceedings of the 11th Joint Conference on Information Sciences*, 2509 – 2513.

Tan, Y.; Yao, T.; Chen, Q.; and Zhu, J. 2004. Chinese chunk identification using svms plus sigmoid. In *Proceedings of IJCNLP2004*, 527–536.

Tan, Y.; Yao, T.; and Chen, Q. 2005. Applying conditional random fields to chinese shallow parsing. In *Proceedings of CICLing2005*, 167–176.

Wang, M.; Che, W.; and Manning, C. D. 2013. Joint word alignment and bilingual named entity recognition using dual decomposition. In *Proceedings of ACL*, 1073–1082.

Wu, S. H.; Shih, C. W.; Wu, C. W.; Tsai, T. H.; and Hsu, W. L. 2005. Applying maximum entropy to robust chinese shallow parsing. In *Proceedings of ROCLING2005.*

Yang, B., and Cardie, C. 2013. Joint inference for fine-grained opinion extraction. In *Proceedings of ACL*, 1640–1649.

Yao, L.; Li, M.; and Huang, C. 2007. Improving chinese chunking with enriched statistical and morphological knowledge. In *Proceedings of IEEE NLP-KE2007*, 149 – 156.

Zhang, Y., and Clark, S. 2010. A fast decoder for joint word segmentation and pos-tagging using a single discriminative model. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, 843–852. Stroudsburg, PA, USA: Association for Computational Linguistics.

Zhang, Y., and Clark, S. 2011. Syntactic processing using the generalized perceptron and beam search. *Computational Linguistics* 37(1):105–151.

Zhang, M.; Zhang, Y.; Che, W.; and Liu, T. 2013. Chinese parsing exploiting characters. In *Proceedings of ACL.*

Zhang, T.; Damerau, F.; and Johnson, D. 2002. Text chunking based on a generalization of winnow. *The Journal of Machine Learning Research* 2:615–637.

Zhao, T.; Yang, M.; Liu, F.; Yao, J.; and Yu, H. 2000. Statistics based hybrid approach to chinese base phrase identification. In *Proceedings of Second Chinese Language Processing Workshop*, 73–77.

Zhou, J.; Qu, W.; and Zhang, F. 2012. Exploiting chunk-level features to improve phrase chunking. In *Proceedings of EMNLP-CONLL*, 557–567. Stroudsburg, PA, USA: Association for Computational Linguistics.