## **Exploiting Chunk-level Features to Improve Phrase Chunking**

Junsheng Zhou Weiguang Qu Fen Zhang

Jiangsu Research Center of Information Security & Privacy Technology School of Computer Science and Technology Nanjing Normal University. Nanjing, China, 210046 Email:{zhoujs,wgqu}@njnu.edu.cn zf9646@126.com

Abstract

Most existing systems solved the phrase chunking task with the sequence labeling approaches, in which the chunk candidates cannot be treated as a whole during parsing process so that the chunk-level features cannot be exploited in a natural way. In this paper, we formulate phrase chunking as a joint segmentation and labeling task. We propose an efficient dynamic programming algorithm with pruning for decoding, which allows the direct use of the features describing the internal characteristics of chunk and the features capturing the correlations between adjacent chunks. A relaxed, online maximum margin training algorithm is used for learning. Within this framework, we explored a variety of effective feature representations for phrase chunking. The Chinese experimental results show that the use of chunk-level features can lead to significant performance improvement, and that our achieves approach state-of-the-art performance. In particular, our approach is much better at recognizing long and complicated phrases.

## 1 Introduction

Phrase chunking is a Natural Language Processing task that consists in dividing a text into syntactically correlated parts of words. Theses phrases are non-overlapping, i.e., a word can only be a member of one chunk (Abney, 1991). Generally speaking, there are two phrase chunking tasks, including text chunking (shallow parsing), and noun phrase (NP) chunking. Phrase chunking provides a key feature that helps on more elaborated NLP tasks such as parsing, semantic role tagging and information extraction.

There is a wide range of research work on phrase chunking based on machine learning approaches. However, most of the previous work reduced phrase chunking to sequence labeling problems either by using the classification models, such as SVM (Kudo and Matsumoto, 2001), Winnow and voted-perceptrons (Zhang et al., 2002; Collins, 2002), or by using the sequence labeling models, such as Hidden Markov Models (HMMs) (Molina and Pla, 2002) and Conditional Random Fields (CRFs) (Sha and Pereira, 2003). When applying the sequence labeling approaches to phrase chunking, there exist two major problems. Firstly, these models cannot treat globally a sequence of continuous words as a chunk candidate, and thus cannot inspect the internal structure of the candidate, which is an important aspect of information in modeling phrase chunking. In particular, it makes impossible the use of local indicator function features of the type "the chunk consists of POS tag sequence  $p_1...,p_k$ ". For example, the Chinese NP "农业/NN(agriculture) 生产 /NN(production) 和/CC(and) 农村/NN(rural) 经济 /NN(economic) 发展/NN(development)" seems relatively difficult to be correctly recognized by a sequence labeling approach due to its length. But if we can treat the sequence of words as a whole and describe the formation pattern of POS tags of this chunk with a regular expression-like form "[NN]+[CC][NN]+", then it is more likely to be correctly recognized, since this pattern might better express the characteristics of its constituents. As another example, consider the recognition of special terms. In Chinese corpus, there exists a kind of NPs called special terms, such as "『生命

(Life) 禁区 (Forbidden Zone) ] ", which are bracketed with the particular punctuations like " [, ], [, ], [, ], (, )]". When recognizing the special terms, it is difficult for the sequence labeling approaches to guarantee the matching of particular punctuations appearing at the starting and ending positions of a chunk. For instance, the chunk candidate " [ 生命(Life) 禁区(Forbidden Zone)" is considered to be an invalid chunk. But it is easy to check this kind of punctuation matching in a single chunk by introducing a chunklevel feature.

Secondly, the sequence labeling models cannot capture the correlations between adjacent chunks, which should be informative for the identification of chunk boundaries and types. In particular, we find that some headwords in the sentence are expected to have a stronger dependency relation with their preceding headwords in preceding chunks than with their immediately preceding words within the same chunk. For example, in the following sentence:

"[双方/PN(Bilateral)]\_NP [经贸/NN(economic and trade) 关系/NN(relations)]\_NP [正/AD(just) 稳步/AD(steadily) 发展/VV(develop)]\_VP "

if we can find the three headwords "双方", "关系" and "发展" located in the three adjacent chunks with some head-finding rules, then the headword dependency expressed by headword bigrams or trigrams should be helpful to recognize these chunks in this sentence.

In summary, the inherent deficiency in applying the sequence labeling approaches to phrase chunking is that the chunk-level features one would expect to be very informative cannot be exploited in a natural way.

In this paper, we formulate phrase chunking as a joint segmentation and labeling problem, which offers advantages over previous learning methods by providing a natural formulation to exploit the features describing the internal structure of a chunk and the features capturing the correlations between the adjacent chunks.

Within this framework, we explored a variety of effective feature representations for Chinese phrase chunking. The experimental results on Chinese chunking corpus as well as English chunking corpus show that the use of chunk-level features can lead to significant performance improvement, and that our approach performs better than other approaches based on the sequence labeling models.

## 2 Related Work

In recent years, many chunking systems based on machine learning approaches have been presented. Some approaches rely on k-order generative probabilistic models, such as HMMs (Molina and Pla, 2002). However, HMMs learn a generative model over input sequence and labeled sequence pairs. It has difficulties in modeling multiple nonindependent features of the observation sequence. To accommodate multiple overlapping features on observations, some other approaches view the phrase chunking as a sequence of classification problems, including support vector machines (SVMs) (Kudo and Matsumoto 2001) and a variety of other classifiers (Zhang et al., 2002). Since these classifiers cannot trade off decisions at different positions against each other, the best classifier based shallow parsers are forced to resort to heuristic combinations of multiple classifiers. Recently, CRFs were widely employed for phrase chunking, and presented comparable or better performance than other state-of-the-art models (Sha and Pereira 2003; McDonald et al. 2005). Further, Sun et al. (2008) used the latent-dynamic conditional random fields (LDCRF) to explicitly learn the hidden substructure of shallow phrases, achieving state-of-the-art performance over the NP-chunking task on the CoNLL data.

Some similar approaches based on classifiers or sequence labeling models were also used for Chinese chunking (Li et al., 2003; Tan et al., 2004; Tan et al., 2005). Chen et al. (2006) conducted an empirical study of Chinese chunking on a corpus, which was extracted from UPENN Chinese Treebank-4 (CTB4). They compared the performances of the state-of-the-art machine learning models for Chinese chunking, and proposed some Tag-Extension and novel voting methods to improve performance.

In this paper, we model phrase chunking with a joint segmentation and labeling approach, which offer advantages over previous learning methods by explicitly incorporating the internal structural feature and the correlations between the adjacent chunks. To some extent, our model is similar to Semi-Markov Conditional Random Fields (called a Semi-CRF), in which the segmentation and

labeling can also be done directly (Sarawagi and Cohen, 2004). However, Semi-CRF just models label dependency, and it cannot capture more correlations between adjacent chunks, as is done in our approach. The limitation of Semi-CRF leads to its relatively low performance.

## **3 Problem Formulation**

### 3.1 Chunk Types

Unlike English chunking, there is not a benchmarking corpus for Chinese chunking. We follow the studies in (Chen et al. 2006) so that a more direct comparison with state-of-the-art systems for Chinese chunking would be possible. There are 12 types of chunks: ADJP, ADVP, CLP, DNP, DP, DVP, LCP, LST, NP, PP, QP and VP in the chunking corpus (Xue et al., 2000). The training and test corpus can be extracted from CTB4 with a public tool, as depicted in (Chen et al. 2006).

# **3.2** Sequence Labeling Approaches to Phrase Chunking

The standard approach to phrase chunking is to use tagging techniques with a BIO tag set. Words in the input text are tagged with one of B for the beginning of a contiguous segment, I for the inside of a contiguous segment, or O for outside a segment. For instance, the sentence (word segmented and POS tagged) "他/NR(He) 到达/VV(reached) 北京/NR(Beijing) 机场/NN(airport)。/PU" will be tagged as follows:

Example 1:

S1: [NP 他][VP 到达][NP 北京/机场][O。] S2: 他/B-NP 到达/B-VP 北京/B-NP 机场/I-NP。/O

Here S1 denotes that the sentence is tagged with chunk types, and S2 denotes that the sentence is tagged with chunk tags based on the BIO-based model. With the data representation like the S2, the problem of phrase chunking can be reduced to a sequence labeling task.

# **3.3** Phrase Chunking via a Joint Segmentation and Labeling Approach

To tackle the problems with the sequence labeling approaches to phrase chunking, we formulate it as a joint problem, which maps a Chinese sentence x

with segmented words and POS tags to an output y with tagged chunk types, like the S1 in Example 1. The joint model considers all possible chunk boundaries and corresponding chunk types in the sentence, and chooses the overall best output. This kind of parser reads the input sentences from left to right, predicts whether current segment of continuous words is some type of chunk. After one chunk is found, parser move on and search for next possible chunk.

Given a sentence x, let y denote an output tagged with chunk types, and GEN a function that enumerates a set of segmentation and labeling candidates GEN(x) for x. A parser is to solve the following "argmax" problem:

$$\hat{y} = \arg \max_{y \in GEN(x)} w^{T} \cdot \Phi(y)$$

$$= \arg \max_{y \in GEN(x)} w^{T} \cdot \sum_{i=1}^{|y|} \phi(y_{[1..i]})$$
(1)

where  $\Phi$  and  $\phi$  are global and local feature maps and w is the parameter vector to learn. The inner product  $w^T \cdot \phi(y_{[1,i]})$  can be seen as the confidence score of whether  $y_i$  is a chunk. The parser takes into account confidence score of each chunk, by using the sum of local scores as its criteria. Markov assumption is necessary for computation, so  $\phi$  is usually defined on a limited history.

The main advantage of the joint segmentation and labeling approach to phrase chunking is to allow for integrating both the internal structural features and the correlations between the adjacent chunks for prediction. The two basic components of our model are decoding and learning algorithms, which are described in the following sections.

## 4 Decoding

The inference technique is one of the most important components for a joint segmentation and labeling model. In this section, we propose a dynamic programming algorithm with pruning to efficiently produce the optimal output.

## 4.1 Algorithm Description

Given an input sentence x, the decoding algorithm searches for the highest-scored output with recognized chunks. The search space of combined candidates in the joint segmentation and labeling task is very large, which is an exponential growth in the number of possible candidates with increasing sentence size. The rate of growth is  $O(2^nT^n)$  for the joint system, where n is the length of the sentence and T is the number of chunk types. It is natural to use some greedy heuristic search algorithms for inference in some similar joint problems (Zhang and Clark, 2008; Zhang and Clark, 2010). However, the greedy heuristic search algorithms only explore a fraction of the whole space (even with beam search) as opposed to dynamic programming. Additionally, a specific advantage of the dynamic programming algorithm is that constraints required in a valid prediction sequence can be handled in a principled way. We show that dynamic programming is in fact possible for this joint problem, by introducing some effective pruning schemes.

To make the inference tractable, we first make a first-order Markov assumption on the features used in our model. In other words, we assume that the chunk  $c_i$  and the corresponding label  $t_i$  are only associated with the preceding chunk  $c_{i-1}$  and the label  $t_{i-1}$ . Suppose that the input sentence has nwords and the constant M is the maximum chunk length in the training corpus. Let V(b,e,t) denote the highest-scored segmentation and labeling with the last chunk starting at word index b, ending at word index *e* and the last chunk type being *t*. One way to find the highest-scored segmentation and labeling for the input sentence is to first calculate the V(b,n-1,t) for all possible start position  $b \in (n-1)$ *M*)..*n*-1, and all possible chunk type t, respectively, and then pick the highest-scored one from these candidates. In order to compute V(b,n-1,t), the last chunk needs to be combined with all possible different segmentations of words (b-M)..b-1 and all possible different chunk types so that the highestscored can be selected. According to the principle of optimality, the highest-scored among the segmentations of words (b-M)..b-1 and all possible chunk types with the last chunk being word b'..b-1 and the last chunk type being t' will also give the highest score when combined with the word b..n-1 and tag t. In this way, the search task is reduced recursively into smaller subproblems. where in the base case the subproblems V(0,e,t) for  $e \in 0..M$ -1, and each possible chunk type t, are solved in straightforward manner. And the final highest-scored segmentation and labeling can be found by solving all subproblems in a bottom-up fashion.

The pseudo code for this algorithm is shown in Figure 1. It works by filling an *n* by *n* by *T* table *chart*, where *n* is the number of words in the input sentence *sent*, and *T* is the number of chunk types. *chart*[*b*,*e*,*t*] records the value of subproblem V(b,e,t). *chart*[0, *e*, *t*] can be computed directly for e = 0..M-1 and for chunk type t=1..T. The final output is the best among *chart*[*b*,*n*-1,*t*], with b=n-M..n-1, and t=1..T.

**Inputs**: sentence *sent* (word segmented and POS tagged)

#### Variables:

word index *b* for the start of chunk;

word index *e* for the end of chunk;

word index p for the start of the previous chunk.

chunk type index *t* for the current chunk;

chunk type index t' for the previous chunk;

## Initialization:

for e = 0.. M-1:

for *t* =1..*T*:

*chart* $[0,e,t] \leftarrow$  single chunk *sent*[0,e] and type *t* **Algorithm**:

for *e* = 0..*n*-1:

for b = (e-M)..e:

for t = 1..T:

 $chart[b,e,t] \leftarrow$  the highest scored segmentation and labeling among those derived by combining chart[p,b-1,t'] with sent[b,e]and chunk type t, for p = (b-M)..b-1, t'=1..T.

**Outputs**: the highest scored segmentation and labeling among *chart*[b,n-1,t], for b=n-M..n-1, t =1..T.

Figure 1: A dynamic-programming algorithm for phrase chunking.

#### 4.2 Pruning

The time complexity of the above algorithm is  $O(M^2T^2n)$ , where *M* is the maximum chunk size. It is linear in the length of sentence. However, the constant in the *O* is relatively large. In practice, the search space contains a large number of invalid partial candidates, which make the algorithm slow. In this section we describe three partial output pruning schemes which are helpful in speeding up the algorithm.

Firstly, we collect chunk type transition information between chunk types by observing every pair of adjacent chunks in the training corpus, and record a chunk type transition matrix. For example, from the Chinese Treebank that we used for our experiments, a transition from chunk type ADJP to ADVP does not occur in the training corpus, the corresponding matrix element is set to *false, true* otherwise. During decoding, the chunk type transition information is used to prune unlikely combinations between current chunk and the preceding chunk by their chunk types.

Secondly, a POS tag dictionary is used to record POS tags associated with each chunk type. Specifically, for each chunk type, we record all POS tags appearing in this type of chunk in the training corpus. During decoding, a segment of continuous words that contains only allowed POS tags according to the POS tag dictionary will be considered to be a valid chunk candidate.

Finally, the system records the maximum number of words for each type of chunk in the training corpus. For example, in the Chinese Treebank, most types of chunks have one to three words. The few chunk types that are seen with length bigger than ten are NP, QP and ADJP. During decoding, the chunk candidate whose length is greater than the maximum chunk length associated with its chunk type will be discarded.

For the above pruning schemes, development tests show that it improves the speed significantly, while having a very small negative influence on the accuracy.

#### 5 Learning

#### 5.1 Discriminative Online Training

By defining features, a candidate output y is mapped into a global feature vector, in which each dimension represents the count of a particular feature in the sentence. The learning task is to set the parameter values w using the training examples as evidence.

Online learning is an attractive method for the joint model since it quickly converges within a few iterations (McDonald, 2006). We focus on an online learning algorithm called MIRA, which is a relaxed, online maximum margin training algorithm with the desired accuracy and scalability properties (Crammer, 2004). Furthermore, MIRA

is very flexible with respect to the loss function. Any loss function on the output is compatible with MIRA since it does not require the loss to factor according to the output, which enables our model to be optimized with respect to evaluation metrics directly. Figure 2 outlines the generic online learning algorithm (McDonald, 2006) used in our framework.

MIRA updates the parameter vector w with two constraints: (1) the positive example must have a higher score by a given margin, and (2) the change to w should be minimal. This second constraint is to reduce fluctuations in w. In particular, we use a generalized version of MIRA (Crammer et al., 2005; McDonald, 2006) that can incorporate *k-best* decoding in the update procedure.

**Input**: Training set  $S = \{(x_t, y_t)\}_{t=1}^T$ 

1:  $w^{(0)} = 0$ ; v = 0; i = 02: for *iter* = 1 to *N* do 3: for t = 1 to *T* do 4:  $w^{(i+1)} =$  update  $w^{(i)}$  according to  $(x_b, y_t)$ 5:  $v = v + w^{(i+1)}$ 6: i = i + 17: end for 8: end for 9:  $w = v/(N \times T)$ **Output**: weight vector *w* 

Figure 2: Generic Online Learning Algorithm

In each iteration, MIRA updates the weight vector w by keeping the norm of the change in the weight vector as small as possible. Within this framework, we can formulate the optimization problem as follows (McDonald, 2006):

$$w^{(t+1)} = \arg\min_{w} ||w - w^{(t)}||$$
  
s.t.  $\forall y' \in best_k(x_t; w^{(t)}):$   
 $w^T \cdot \Phi(y_t) - w^T \cdot \Phi(y') \ge L(y_t, y')$  (2)

where  $best_k(x_t; w^{(i)})$  represents a set of top *k-best* outputs for  $x_t$  given the weight vector  $w^{(i)}$ . In our implementation, the top *k-best* outputs are obtained with a straightforward *k-best* extension to the decoding algorithm in section 4.1. The above quadratic programming (QP) problem can be solved using Hildreth's algorithm (Yair Censor, 1997). Replacing Eq. (2) into line 4 of the algorithm in Figure 2, we obtain *k-best* MIRA.

As shown in (McDonald, 2006), parameter averaging can effectively avoid overfitting. The

final weight vector *w* is the average of the weight vectors after each iteration.

## 5.2 Loss Function

For the joint segmentation and labeling task, there are two alternative loss functions: 0-1 loss and F1 loss. 0-1 loss gives credit only when the entire output sequence is correct: there is no notion of partially correct solutions. The most common loss function for joint segmentation and labeling problems is F1 measure over chunks. This is the geometric mean of precision and recall over the (properly-labeled) chunk identification task, defined as follows.

$$L^{F}(y,\hat{y}) \triangleq 1 - \frac{2|y \cap y'|}{|y| + |y'|}$$
(3)

where the cardinality of y is simply the number of chunks identified. The cardinality of the intersection is the number of chunks in common. As can be seen in the definition, one is penalized both for identifying too many chunks (penalty in the denominator) and for identifying too few (penalty in the numerator).

In our experiments, we will compare the performance of the systems with different loss functions.

## 5.3 Features

Table 1 shows the feature templates for the joint segmentation and labeling model. In the row for feature templates, c, t, w and p are used to represent a chunk, a chunk type, a word and a POS tag, respectively. And  $c_0$  and  $c_{-1}$  represent the current chunk and the previous chunk respectively. Similarly,  $w_{-1}$ ,  $w_0$  and  $w_1$  represent the previous word, the current word and the next word, respectively.

Although it is slightly less natural to do so, part of the features used in the sequence labeling models can also be represented in our approach. Therefore the features employed in our model can be divided into three types: the features similar to those used in the sequence labeling models (called SL-type features), the features describing internal structure of a chunk (called Internal-type features), and the features capturing the correlations between the adjacent chunks (called Correlation-type features).

Firstly, some features associated with a single label (here refers to label "B" and "I") used in the

sequence labeling models are also represented in our model. In Table 1, templates 1-4 are SL-type features, where *label(w)* denotes the label indicating the position of the word w in the current chunk; *len(c)* denotes the length of chunk c. For example, given an NP chunk "北京(Beijing) 机场 (Airport)", which includes two words, the value of *label*("北京") is "B" and the value of *label*("机场") is "I". *Bigram(w)* denotes the word bigrams formed by combining the word to the left of w and the one to the right of w. And the same meaning is for *biPOS(w)*. Template *specitermMatch(c)* is used to check the punctuation matching within chunk c for the special terms, as illustrated in *section* 1.

Secondly, in our model, we have a chance to treat the chunk candidate as a whole during decoding, which means that we can employ more expressive features in our model than in the sequence labeling models. In Table 1, templates 5-13 concern the Internal-type features, where start word(c) and end word(c) represent the first word and the last word of chunk c, respectively. Similarly, start POS(c) and end POS(c) represent the POS tags associated with the first word and the last word of chunk c, respectively. These features aim at expressing the formation patterns of the current chunk with respect to words and POS tags. *internalWords(c)* Template denotes the concatenation of words in chunk c, while *internalPOSs(c)* denotes the sequence of POS tags in chunk c using regular expression-like form, as illustrated in section 1.

Finally, in Table 1, templates 14-28 concern the Correlation-type features, where head(c) denotes the headword extracted from chunk c, and headPOS(c) denotes the POS tag associated with the headword in chunk c. These features take into account various aspects of correlations between adjacent chunks. For example, we extracted the headwords located in adjacent chunks to form headword bigrams to express semantic dependency between adjacent chunks. To find the headword within every chunk, we referred to the head-finding rules from (Bikel, 2004), and made a simple modification to them. For instance, the head-finding rule for NP in (Bikel, 2004) is as follows:

## (NP (r NP NN NT NR QP) (r))

Since the phrases are non-overlapping in our task, we simply remove the overlapping phrase tags NP

and QP from the rule, and then the rule is modified as follows:

(NP (r NN NT NR) (r))

Additionally, the different bigrams formed by combining the first word (or POS) and last word (or POS) located in two adjacent chunks can also capture some correlations between adjacent chunks, and templates 17-22 are designed to express this kind of bigram information.

ID	Feature template
1	$wlabel(w) t_0$
	for all $w$ in $c_0$
2	$bigram(w) \ label(w)t_0$
	for all $w$ in $c_0$
3	$biPOS(w) \ label(w)t_0$
	for all $w$ in $c_0$
4	$w_{-1}w_{1}label(w_{0}) t_{0}$ , where $len(c_{0})=1$
5	$start_word(c_0)t_0$
6	$start_{POS(c_0)t_0}$
7	$end\_word(c_0)t_0$
8	$end\_POS(c_0)t_0$
9	wend_word $(c_0) t_0$
	where $w \in c_0$ and $w \neq end \_word(c_0)$
10	$pend\_POS(c_0) t_0$
	where $p \in c_0$ and $p \neq end \_POS(c_0)$
11	$internalPOSs(c_0) t_0$
12	$internalWords(c_0) t_0$
13	$specitermMatch(c_0)$
14	$t_{-1}t_0$
15	$head(c_{-1})t_{-1}head(c_0)t_0$
16	$headPOS(c_{-1})t_{-1}headPOS(c_{0})t_{0}$
17	$end\_word(c_{-1})t_{-1}start\_word(c_{0})t_{0}$
18	$end\_POS(c_{-1})t_{-1}start\_POS(c_{0})t_{0}$
19	$end\_word(c_1)t_1end\_word(c_0)t_0$
20	$end\_POS(c_{-1})t_{-1}end\_POS(c_{0})t_{0}$
21	$start_word(c_1)t_1start_word(c_0)t_0$
22	$start_POS(c_1)t_1start_POS(c_0)t_0$
23	$end\_word(c_{-1})t_0$
24	$end_POS(c_{-1})t_0$
25	$t_1t_0start\_word(c_0)$
26	$t_{-1}t_0 start_POS(c_0)$
27	$internalWords(c_{-1}) t_{-1} internalWords(c_0) t_0$
28	$internalPOSs(c_{-1}) t_{-1} internalPOSs(c_{0}) t_{0}$

Table 1: Feature templates.

#### **6** Experiments

#### 6.1 Data Sets and Evaluation

Following previous studies on Chinese chunking in (Chen et al., 2006), our experiments were performed on the CTB4 dataset. The dataset consists of 838 files. In the experiments, we used the first 728 files (FID from chtb 001.fid to chtb 899.fid) as training data, and the other 110 files (FID from chtb 900.fid to chtb 1078.fid) as testing data. The training set consists of 9878 sentences, and the test set consists of 5920 sentences. The standard evaluation metrics for this task are precision *p* (the fraction of output chunks matching the reference chunks), recall *r* (the fraction of reference chunks returned), and the F-measure given by F = 2pr/(p + r).

Our model has two tunable parameters: the number of training iterations N; the number of top *k-best* outputs. Since we were interested in finding an effective feature representation at chunk-level for phrase chunking, we fixed N = 10 and k = 5 for all experiments. In the following experiments, our model has roughly comparable training time to the sequence labeling approach based on CRFs.

## 6.2 Chinese NP chunking

NP is the most important phrase in Chinese chunking and about 47% phrases in the CTB4 Corpus are NPs. In this section, we present the results of our approach to NP recognition.

Table 2 shows the results of the two systems using the same feature representations as defined in Table 1, but using different loss functions for learning. As shown, learning with F1 loss can improve the F-score by 0.34% over learning with 0-1 loss. It is reasonable that the model optimized with respect to evaluation metrics directly can achieve higher performance.

Loss Function	Precision	Recall	F1
0-1 loss	91.39	90.93	91.16
F1 loss	92.03	90.98	91.50

Table 2: Experimental results on Chinese NP chunking.

#### 6.3 Chinese Text Chunking

There are 12 different types of phrases in the chunking corpus. Table 3 shows the results from

two different systems with different loss functions for learning. Observing the results in Table 3, we can see that learning with F1 loss can improve the F-score by 0.36% over learning with 0-1 loss, similar to the case in NP recognition. More specifically, learning with F1 loss provides much better results for ADJP, ADVP, DVP, NP and VP, respectively. And it yields equivalent or comparable results to 0-1 loss in other categories.

	F1 loss		0-1 loss			
	precision	recall	<i>F1</i>	precision	recall	F1
ADJP	87.86	87.09	87.47	86.74	86.55	86.64
ADVP	90.66	78.73	84.27	91.91	76.68	83.61
CLP	0.00	0.00	0.00	1.32	5.88	2.15
DNP	99.42	99.93	99.68	99.42	99.95	99.69
DP	99.46	99.76	99.61	99.46	99.76	99.61
DVP	99.61	99.61	99.61	99.22	99.61	99.42
LCP	99.74	99.96	99.85	99.74	99.93	99.84
LST	87.50	52.50	65.63	87.50	52.50	65.63
NP	91.87	91.01	91.44	91.34	90.52	90.93
РР	99.57	99.77	99.67	99.57	99.77	99.67
QP	96.45	96.64	96.55	96.45	97.07	96.76
VP	90.14	90.39	90.26	89.92	89.79	89.85
ALL	92.54	91.68	92.11	92.30	91.20	91.75

Table 3: Experimental results on Chinese text chunking.

#### 6.4 Comparison with Other Models

Chen et al. (2006) compared the performance of the state-of-the-art machine learning models for Chinese chunking, and found that the SVMs approach yields higher accuracy than respective CRFs, Transformation-based Learning (TBL) (Megyesi, 2002), and Memory-based Learning (MBL) (Sang, 2002) approaches.

In this section, we give a comparison and analysis between our model and other state-of-theart machine learning models for Chinese NP chunking and text chunking tasks. Performance of our model and some of the best results from the state-of-the-art systems are summarized in Table 4. Row "Voting" refers to the phrase-based voting methods based on four basic systems, which are respectively SVMs, CRFs, TBL and MBL, as depicted in (Chen et al., 2006). Observing the results in Table 4, we can see that for both NP chunking and text chunking tasks, our model achieves significant performance improvement over those state-of-the-art systems in terms of the F1-score, even for the voting methods. For text

chunking task, our approach improves performance by 0.65% over SVMs, and 0.43% over the voting method, respectively.

	Method	<i>F1</i>
	CRFs	89.72
NP	SVMs	90.62
chunking	Voting	91.13
	Ours	91.50
	CRFs	90.74
Text	SVMs	91.46
chunking	Voting	91.68
	Ours	92.11

Table 4: Comparisons of chunking performance for Chinese NP chunking and text chunking.

In particular, for NP chunking task, the F1-score of our approach is improved by 0.88% in comparison with SVMs, the best single system. Further, we investigated the likely cause for performance improvement by comparing the recognized results from our system and SVMs respectively. We first sorted NPs by their length, and then calculated the F1-scores associated with different lengths for the two systems respectively. Figure 3 shows the comparison of F1-scores of the two systems by the chunk length. In the Chinese chunking corpus, the max NP length is 27, and the mean NP length is 1.5. Among all NPs, the NPs with the length 1 account for 81.22%. For the NPs with the length 1, our system gives slight improvement by 0.28% over SVMs. From the figure, we can see that the performance gap grows rapidly with the increase of the chunk length. In particular, the gap between the two systems is 27.73% when the length hits 4. But the gap begins to become smaller with further growth of the chunk length. The reasons may include the following two aspects. First, the number of NPs with the greater length is relatively small in the corpus. Second, the NPs with greater length in Chinese corpus often exhibit some typical rules. For example, an NP with length 8 is given as follows.

"棉花/NN(cotton)、/PU 油料/NN(oil)、/PU 药 材/NN(drug)、/PU 蔬菜/NN(vegetable)等/ETC (et al)".

The NP consists of a sequence of nouns simply separated by a punctuation "、". So it is also easy to be recognized by the sequence labeling approach based on SVMs. In summary, the above investigation indicates that our system is better at recognizing the long and complicated phrases compared with the sequence labeling approaches.



Figure 3: Comparison of F1-scores of NP recognition on Chinese corpus by the chunk length.

#### 6.5 Impact of Different Types of Features

Our phrase chunking model is highly dependent upon chunk-level information. To establish the impact of each type of feature (SL-type, Internaltype, Correlation-type), we look at the improvement in F1-score brought about by adding each type of features. Table 5 shows the accuracy with various features added to the model.

First consider the effect of the SL-type features. If we use only the SL-type features, the system achieves slightly lower performance than CRFs or SVMs, as shown in Table 4. Since the SL-type features consist of the features associated with single label, not including the features associated with label bigrams. Then, adding the Internal-type features to the system results in significant performance improvement on NP chunking and on text chunking, achieving 2.53% and 1.37%, respectively. Further, if Correlation-type features are used, the F1-scores on NP chunking and on text chunking are improved by 1.01% and 0.66%, respectively. The results show a significant impact due to the use of Internal-type features and Correlation-type features for both NP chunking and text chunking.

Task Type	Feature Type	F1
	SL-type	87.96
NP chunking	+Internal-type	90.49
	+Correlation-type	91.50
	SL-type	90.08
Text chunking	+Internal-type	91.45
	+Correlation-type	92.11

Table 5: Test F1-scores for different types of features on Chinese corpus.

#### 6.6 Performance on Other Languages

We mainly focused on Chinese chunking in this paper. However, our approach is generally applicable to other languages including English, except that the definition of feature templates may be language-specific. To validate this point, we evaluated our system on the CoNLL 2000 data set, a public benchmarking corpus for English chunking (Sang and Buchholz 2000). The training set consists of 8936 sentences, and the test set consists of 2012 sentences.

We conducted both the NP-chunking and text chunking experiments on this data set with our approach, using the same feature templates as in Chinese chunking task excluding template 13. To find the headword within every chunk, we referred to the head-finding rules from (Collins, 1999), and made a simple modification to them in a similar way as in Chinese. As we can see from Table 6, our model is able to achieve better performance compared with state-of-the-art systems. Table 6 also shows state-of-the-art performance for both NP-chunking and text chunking tasks. LDCRF's results presented in (Sun et al., 2008) are the stateof-the-art for the NP chunking task, and SVM's results presented in (Wu et al., 2006) are the stateof-the-art for the text chunking task.

Moreover, the performance should be further improved if some additional features tailored for English chunking are employed in our model. For example, we can introduce an orthographic feature type called Token feature and the affix feature into the model, as used in (Wu et al., 2006).

	Method	Precision	Recall	F1
NP	Ours	94.79	94.65	94.72
chunking	LDCRF	94.65	94.03	94.34
Text	Ours	94.31	94.12	94.22
chunking	SVMs	94.12	94.13	94.12

Table 6: Performance on English corpus.

#### 7 Conclusions and Future Work

In this paper we have presented a novel approach to phrase chunking by formulating it as a joint segmentation and labeling problem. One important advantage of our approach is that it provides a natural formulation to exploit chunk-level features. The experimental results on both Chinese chunking and English chunking tasks show that the use of chunk-level features can lead to significant performance improvement and that our approach outperforms the best in the literature.

Future work mainly includes the following two aspects. Firstly, we will explore applying external information, such as semantic knowledge, to represent the chunk-level features, and then incorporate them into our model to improve the performance. Secondly, we plan to apply our approach to other joint segmentation and labeling tasks, such as clause identification and named entity recognition.

#### Acknowledgments

This research is supported by Projects 61073119, 60773173 under the National Natural Science Foundation of China, and project BK2010547 under the Jiangsu Natural Science Foundation of China. We would also like to thank the excellent

and insightful comments from the three anonymous reviewers.

#### References

- Steven P. Abney. 1991. Parsing by chunks. In Robert C. Berwick, Steven P. Abney, and Carol Tenny, editors, Principle-Based Parsing, pages 257-278. Kluwer Academic Publishers.
- Daniel M, Bikel. 2004. On the Parameter Space of Generative Lexicalized Statistical Parsing Models. Ph.D. thesis, University of Pennsylvania.
- Wenliang Chen, Yujie Zhang, and Hitoshi Isahara. 2006. An empirical study of Chinese chunking. In Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions, pages 97-104.
- Michael Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In Proc. EMNLP-02.
- Michael Collins. 1999. Head-Driven Statistical Models for Natural Language Parsing. Ph.D. thesis, University of Pennsylvania.
- Koby Crammer. 2004. Online Learning of Complex Categorial Problems. Hebrew University of Jerusalem, PhD Thesis.
- Taku Kudo and Yuji Matsumoto. 2001. Chunking with support vector machines. In Proceedings of NAACL01.
- Koby Crammer, Ryan McDonald, and Fernando Pereira. 2005. Scalable large-margin online learning for structured classification. In NIPS Workshop on Learning With Structured Outputs.
- Heng Li, Jonathan J. Webster, Chunyu Kit, and Tianshun Yao. 2003. Transductive hmm based chinese text chunking. In Proceedings of IEEE NLPKE2003, pages 257-262, Beijing, China.
- Ryan McDonald, Femando Pereira, Kiril Ribarow, and Jan Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In Proceedings of HLT/EMNLP, pages 523-530.
- Ryan. McDonald, K. Crammer, and F. Pereira, 2005. Flexible Text Segmentation with Structured Multilabel Classification. In Proceedings HLT/EMNLP, pages 987- 994.
- Ryan McDonald. 2006. Discriminative Training and Spanning Tree Algorithms for Dependency Parsing. University of Pennsylvania, PhD Thesis.
- Beata Megyesi. 2002. Shallow parsing with pos taggers and linguistic features. Journal of Machine Learning Research, 2:639-668.

- Antonio Molina and Ferran Pla. 2002. Shallow parsing using specialized hmms. Journal of Machine Learning Research., 2:595-613.
- E.F.T.K Sang and S. Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In Proceedings CoNLL-00, pages 127-132.
- Sunita Sarawagi and W. Cohen. 2004. Semi-markov conditional random fields for information extraction. In Proceedings of NIPS 17, pages 1185–1192.
- Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In Proceedings of HLT-NAACL03.
- Xu Sun, Louis-Philippe Morency, Daisuke Okanohara, and Jun'ichi Tsujii. 2008. Modeling Latent-Dynamic in Shallow Parsing: A Latent Conditional Model with Improved Inference. In Proceedings of the 22nd International Conference on Computational Linguistics, pages 841–848.
- Yongmei Tan, Tianshun Yao, Qing Chen, and Jingbo Zhu. 2004. Chinese chunk identification using svms plus sigmoid. In IJCNLP, pages 527-536.
- Yongmei Tan, Tianshun Yao, Qing Chen, and Jingbo Zhu. 2005. Applying conditional random fields to chinese shallow parsing. In Proceedings of CICLing-2005, pages 167-176.
- Erik F. Tjong Kim Sang. 2002. Memory-based shallow parsing. JMLR, 2(3):559-594.
- Yu-Chieh Wu, Chia-Hui Chang, and Yue-Shi Lee. 2006. A general and multi-lingual phrase chunking model based on masking method. In Proceedings of 7th International Conference on Intelligent Text Processing and Computational Linguistics, pages 144-155.
- Nianwen Xue, Fei Xia, Shizhe Huang, and Anthony Kroch. 2000. The bracketing guidelines for the penn chinese treebank. Technical report, University of Pennsylvania.
- Stavros A. Zenios Yair Censor. 1997. Parallel Optimization: Theory, Algorithms, and Applications. Oxford University Press.
- Tong Zhang, F. Damerau, and D. Johnson. 2002. Text chunking based on a generalization of winnow. Journal of Machine Learning Research, 2:615-637.
- Yue Zhang and Stephen Clark. 2008. Joint word segmentation and POS tagging using a single perceptron. In Proceedings of ACL/HLT, pages 888-896.
- Yue Zhang and Stephen Clark. 2010. A fast decoder for joint word segmentation and POS-tagging using a

single discriminative model. In Proceedings of EMNLP, pages 843-852.