A SVM-based Model for Chinese Functional Chunk Parsing

Yingze Zhao
State Key Laboratory of Intelligent Technology and Systems
Dept. of Computer Science and Technology, Tsinghua University
Beijing 100084, P. R. China
zhaoyingze@gmail.com

Qiang Zhou
State Key Laboratory of Intelligent Technology and Systems
Dept. of Computer Science and Technology, Tsinghua University
Beijing 100084, P. R. China
zq-lxd@mail.tsinghua.edu.cn

Abstract

Functional chunks are defined as a series of non-overlapping, non-nested segments of text in a sentence, representing the implicit grammatical relations between the sentence-level predicates and their arguments. Its top-down scheme and complexity of internal constitutions bring in a new challenge for automatic parser. In this paper, a new parsing model is proposed to formulate the complete chunking problem as a series of boundary detection sub tasks. Each of these sub tasks is only in charge of detecting one type of the chunk boundaries. As each sub task could be modeled as a binary classification problem, a lot of machine learning techniques could be applied.

In our experiments, we only focus on the subject-predicate (SP) and predicate-object (PO) boundary detection sub tasks. By applying SVM algorithm to these sub tasks, we have achieved the best F-Score of 76.56% and 82.26% respectively.

1 Introduction

Parsing is a basic task in natural language processing; however, it has not been successful in achieving the accuracy and efficiency required by real world applications. As an alternative, shallow parsing or partial parsing has been proposed to meet the current needs by obtaining only a limited amount of syntactic information needed by the application. In recent years, there has been an increasing interest in chunk parsing. From CoNLL-2000 to CoNLL-2005, a lot of efforts have been made in the identification of basic chunks and the methods of combining them from bottom-up to form large, complex units. In this paper, we will apply functional chunks to Chinese shallow parsing.

Functional chunks are defined as a series of non-overlapping, non-nested functional units in a sentence, such as subjects, predicates, objects, adverbs, complements and so on. These units represent the implicit grammatical relations between the sentence-level predicates and their arguments. Different from the basic chunks defined by Abney (1991), functional chunks are generated from a top-down scheme, and thus their constitutions may be very complex. In addition, the type of a functional chunk could not be simply determined by its constitution, but depends heavily on the context. Therefore, we will have new challenges in the functional chunk parsing.

Ramshaw and Marcus (1995) first introduced the machine learning techniques to chunking problem. By formulating the NP-chunking task as a tagging process, they marked each word with a tag from set {B, I, O}, and successfully applied TBL to it. Inspired by their work, we introduce SVM algorithm to our functional chunking problem. Instead of using the BIO tagging system, we propose a new model for solving this problem. In this model, we do not tag the words with BIO tags, but directly discover the chunk boundaries between every two adjacent functional chunks. Each of these chunk boundaries will be assigned a type to it, which contains the information of the functional chunk types before and after it. Then we further decompose this model into a series of sub modules, each of which is in charge of detecting only one type of the chunk boundaries. As each sub module can be modeled as a binary classifier, various machine learning techniques could be applied.

In our experiments, we focus on the subject-predicate (SP) and predicate-object (PO) boundary detection tasks, which are the most difficult but important parts in our parsing model. By applying SVM algorithm to these tasks, we achieve the best F-Score of 76.56% and 82.26% respectively.

This paper is organized as follows. In section 2, we give a brief introduction to the concept of our functional chunks. In section 3, we propose the parsing model for Chinese functional chunk parsing. In section 4, we compare SVM with several other machine learning techniques, and illustrate how competitive SVM is in our chunking task. In section 5, we build 2 sub modules based on SVM algorithm for SP and PO boundary detection tasks. In section 6, some related work on functional chunk parsing is introduced. Section 7 is the conclusion.

2 Functional Chunk Scheme

Functional chunks are defined as a series of non-overlapping, non-nested segments of text at the sentence level without leaving any words outside. Each chunk is labeled with a functional tag, such as subject, predicate, object and so on. These functional chunks in the sentence form a linear structure within which the grammatical relations between sentence-level predicates and their arguments or adjuncts are kept implicitly. Table 1 lists all the tags used in our functional chunk scheme:

Table 1. Functional Chunk Tag Set.

Chunk Tag	Basic Function Description
S	Subject
P	Predicate
O	Object
J	Raised Object
D	Adverbial adjunct
C	Complement
T	Independent constituent
Y	Modal particle

Here, we list some examples to illustrate how these functional tags are used in Chinese sentences.

1. “[D 下午/t (afternoon) ，/， [D 当/p (when) 我/rN (I) 来到/v (come to) 西柏坡村/nS (Xi Bai Po village) 东口/s (eastern entrance) 时/n ，/， [D 已/d (already) [P 有/v (there is) [J 一/m 位/qN (a) 参谋/n (brainman) [D 在/p 那里/rS (there) [P 等候/v (waiting) [Y 了/y 。/。 “

2. “[T 可以说/l (frankly speaking) ，/， [S 那/rN (that) [P 是/vC (was) [O 我/rN (I) 终生/d (lifetime) 不/dN 能/vM (can’t) 忘怀/v (forget) 的/u 。/。 “

3. “[S 时间/n (time) [P 安排/v 得/u (schedule) [C 很/dD (very) 紧/a (tight) 。/。 “

Compared with the basic chunk scheme defined by Abney (1991), our functional chunk scheme has the following two main differences:

(1) Functional chunks are not constituted from bottom-up, but generated from top-down, thus some functional chunks are usually longer and more complex than the basic chunks.

We have a collection of 185 news files as our functional chunk corpus. Each file is manually annotated with functional chunks. There are about 200,000 Chinese words in the corpus. To investigate the complex constitutions of functional chunks, we list the average chunk lengths (ACL) of different types in Table 2:

Table 2. Average Chunk Lengths of Different Types.

Chunk Type	Count	Word Sum	ACL
P	21988	27618	1.26
D	19795	46919	2.37
O	14289	61401	4.30
S	11920	34479	2.89
J	855	2083	2.44
Y	594	604	1.02
T	407	909	2.23
C	244	444	1.82

From the table above, we can find that O chunk has the longest average length of 4.30 words, and S chunk has the second longest average length of 2.89 words, and D chunk has an average length of 2.37 words. Although the average length doesn’t seem so long, the length of a specific chunk varies greatly.

In Table 3, we list some detailed length distributional data of three chunks.

Table 3. Length Distribution of S, O and D Chunks.

Chunk Length	# of S	# of O	# of D
1	5322	3537	12147
2	2093	2228	2499
3	1402	2117	1431
4	917	1624	1010
5	627	1108	696
>5	1559	3675	2013
Sum	11920	14289	19796

From the table above, we can find that there are totally 1559 S chunks with a length of more than 5 words which takes up 13.08% of the total number. And when we refer to the S chunks with more than 3 words, the percentage will increase to 26.03%. These long chunks are usually constituted with several complex phrases or clauses as the modifiers of a head word. Among the O chunks, 25.72% of them have a length of more than 5 words, and 44.84% of them are longer than 3 words. The reason why O chunks have a longer length may be that many of them contain the entire clauses. Although most of the D chunks are less than 5 words, some constituted with complex preposition phrases can still be very long.

The complex constitutions of S, O, D chunks are the main parsing difficulties.

(2) The type of functional chunks can't be simply determined by their constitutions, but depends heavily on their contexts.

As the constitution of a basic chunk is very simple, its type can be largely determined by its head word, but in the case of functional chunks, the relationships between the functional chunks play an important role. For example, a NP phrase before a P chunk can be identified as a subject chunk, but in other sentences, when it follows another P chunk, it will be recognized as an object chunk. Thus we can't determine the type of a functional chunk simply by its constitution.

The context dependencies of functional chunks bring a new challenge for our chunk parser.

In the next section, we will propose a top-down model for Chinese functional chunk parsing. Since the functional chunk boundaries have the information of linking two adjacent chunks, they will be very helpful in the determination of chunk types.

3 Parsing Model

The Chinese functional chunk parser takes a stream of segmented and tagged words as its input, and outputs all the functional chunk boundaries in a sentence. In this section, we will present a parsing model which formulates the functional chunk parsing problem as a boundary detection task, and then decompose this model into a series of sub modules that are easy to build.

3.1 Formulation

Functional chunks have the property of exhaustibility and no words will be left outside the chunks. Thus we don't need to find the end position for a functional chunk as it could be identified by the start of the next one. In this case, we can simply regard the chunking task as a process of cutting the input sentence into several segments of words, each of which is labeled with a functional tag. Based on this idea, we can model the functional chunk parsing problem as a boundary detection task.

Let S=<W, T> denote the input sentence to be parsed by the functional chunk parser, where $W=w_1w_2w_3…w_n$ is the sequence of words in S, and $T=t_1t_2t_3…t_n$ is sequence of the POS tags assigned to each word in W. If w_i is a punctuation mark, t_i will be equal to w_i.

A chunk boundary is defined as a pair $<C_1, C_2>$ where $C_1, C_2 \in \{S, P, O, J, D, C, T, Y\}$, C_1 is the chunk type before this boundary and C_2 is the chunk type following it. The output of the chunk parser is denoted as O=<B, P> where $B=b_1b_2b_3…b_m$ is the sequence of chunk boundaries generated by the parser, and $P=p_1p_2p_3…p_m$ is the corresponding positions of $b_1b_2b_3…b_m$ in the sentence.

Chinese functional chunk parser can be considered as a function h(S) which maps the input sentence S to the chunk boundary sequence O.

Take the following sentence for example:

"14 核电/n(Nuclear electricity) $_1$ 是/vC(is) $_2$ 一/m(a) $_3$ 种/qN(kind) $_4$ 安全/a(safe) $_5$ 、/、 $_6$ 清洁/a(safe) $_7$ 、/、 $_8$ 经济/a(economical) $_9$ 的/u $_{10}$ 能源/n(energy) $_{11}$ 。/。"

"Nuclear electricity is a kind of safe, clean and economical energy."

In this sentence, there are totally 12 Chinese words (punctuation marks are treated the same way as words) with 11 numbers falling between them indicating the positions where a functional chunk boundary may appear. If the input sentence is parsed correctly by the functional chunk parser, a series of boundaries will arise at position 1 and 2, which are illustrated as below:

"14 核电/n $_{<S, P>}$ 是/vC $_{<P, O>}$ 一/m 种/qN 安全/a 、/、 清洁/a 、/、 经济/a 的/u 能源/n 。/。"

From the information provided by these boundaries, we can easily identify the functional chunks in the sentence:

"14 [S 核电/n [P 是/vC [O 一/m 种/qN 安全/a 、/、 清洁/a 、/、 经济/a 的/u 能源/n 。/。"

3.2 Decomposition of Parsing Model

The functional chunk parser presented above could be further divided into several sub modules, each of which is only in charge of detecting one type of the chunk boundaries in a sentence. The sub module in charge of detecting boundary b could be formulated as a Boolean function $h_b(S, i)$ where S is the input sentence and i is the position between word w_i and w_{i+1}. Function $h_b(S, i)$ will take true if there is a chunk boundary of type b at position i, and it will take false if there's not. Since the Boolean function $h_b(S, i)$ can be treated as a binary classifier, many machine learning techniques could be applied.

If we combine every two chunk types in the tag set, we can make a total number of 8*8=64 boundary types in our chunking task. However, not all of them appear in the natural language text, for example, we don't have any SO boundaries in our corpus as S and O chunks can't become neighbors in a sentence without any P chunks between them. In our corpus, we could find 43 boundary types, but only a small number of them are used very frequently. In table 4, we list the 5 most frequently used boundaries in our corpus:

Table 4. The 5 Most Frequently Used Boundaries in the Corpus.

Boundary Type	Count
PO	14209
DP	11459
SD	6156
DD	5238
SP	5233

The top 5 boundaries take up 67.76% of all the 62418 boundaries in our corpus. If we further investigate the chunk types associated with these boundaries, we can find that only four types are involved: P, D, O and S. Referred to Table 2, we can find that these chunks are also the 4 most frequently used chunks in our corpus.

In most cases, S, P, and O chunks constitute the backbone of a Chinese sentence, and they usually contain the most useful information we need. Therefore, we are more concerned about S, P and O chunks. In the following sections, we will focus on the construction of sub modules for SP and PO boundary detection tasks.

4 Statistical Model Selection

After decomposing the parsing model into several sub modules, a lot of machine learning techniques could be applied to the constructions of these sub modules.

SVM[1] is a machine learning technique for solving the binary classification problems. It is well known for its good generalization performance and high efficiency. In this section, we will make a performance comparison between SVM (Vapnik, 1995) and several other machine learning techniques including Naïve Bayes, ID3[2] (Quinlan, 1986) and C4.5[3] (Quinlan, 1993), and then illustrates how competitive SVM is in the boundary detection tasks.

4.1 Experimental Data

The corpus we use here is a collection of 185 news files which are manually corrected after automatic sentence-split, word segmentation and part-of-speech tagging. After these processes, they have been manually annotated with functional chunks. Among the 185 files, 167 of them are taken as the training data and the remaining 18 are left as the test data, which takes up approximately 10% of all the data.

In our experiments, we will use feature templates to describe which features are to be used in the generation of feature vectors. For example, if the current feature template we use is w-1t2, then the feature vector generated at position i will take the first word on the left and the second word tag on the right as its features.

Before we perform any experiments, all the data have been converted to the vectors that are acceptable by different machine learning algorithms. Thus we have a total number of 199268 feature vectors generated from the 185 files. Among them, 172465 vectors are in the training data and 26803 vectors are in the test data. Two sets of training and test data are prepared respectively for the SP and PO boundary detection tasks.

The performance of each experiment is measured with 3 rates: precision, recall and $F_{\beta=1}$, where precision is the percentage of detected boundaries that are correct, recall is the percentage of boundaries in the test data that are found by the parser, and $F_{\beta=1}$ is defined as $F_\beta=(\beta^2+1)$*precision*recall/(β^2*precision + recall) with $\beta=1$.

[1] The software package we use is SVM^{light} v6.00, it is available at http://svmlight.joachims.org/. We use linear kernel function and other default parameters in our experiments.

[2] We use the weka's implementation of Naïve Bayes and ID3 algorithms. Weak 3.4 is available at http://www.cs.waikato.ac.nz/ml/weka/.

[3] We use Quinlan's C4.5 software package with its default parameters in our experiments.

4.2 Algorithm Comparison

We first use t-3t-2t-1t1t2 as the feature template, and list all the experimental results in Table 5 and Table 6. From these results, we can find that SVM has achieved the best precision, recall and F-Score in SP boundary detection task, while C4.5 has an overwhelming advantage in PO boundary detection task. In both tasks, Naïve Bayes algorithm performs the worst, which makes us very disappointed.

Table 5. Results of Different Algorithms in SP Boundary Detection Task.

Algorithm	Precision	Recall	$F_{\beta=1}$
SVM	82.21%	57.10%	67.39%
ID3	67.60%	50.70%	57.94%
C4.5	81.10%	44.60%	57.55%
Naïve Bayes	47.90%	51.00%	49.40%

Table 6. Results of Different Algorithms in PO Boundary Detection Task.

Algorithm	Precision	Recall	$F_{\beta=1}$
C4.5	72.00%	74.70%	73.33%
SVM	67.27%	64.96%	66.09%
ID3	70.70%	59.90%	64.85%
Naïve Bayes	48.10%	60.10%	53.43%

As the feature template we use here is too simple, the results we have got may not seem so persuasive. Therefore we decide to conduct another experiment using a more complex feature template.

In the following experiments, we will use w-2w-1w1w2t-2t-1t1t2 as the feature template. The experimental results are listed in Table 7 and Table 8.

After adding the word information to the feature template, the dimensions of feature vectors used by some algorithms increase dramatically. We remove Naïve Bayes algorithm from the following experiments, as it fails to deal with such high dimensional data.

Table 7. Results of Different Algorithms in SP Boundary Detection Task.

Algorithm	Precision	Recall	$F_{\beta=1}$
SVM	82.25%	61.22%	70.19%
ID3	64.70%	51.70%	57.47%
C4.5	79.70%	37.40%	50.91%

Table 8. Results of Different Algorithms in PO Boundary Detection Task.

Algorithm	Precision	Recall	$F_{\beta=1}$
SVM	74.83%	86.99%	80.45%
C4.5	67.90%	79.90%	73.41%
ID3	75.10%	57.70%	65.26%

After applying the complex feature template, SVM still keeps the first place in SP boundary detection task. In PO boundary detection task, SVM successfully takes the place of C4.5, and achieves the best recall and F-Score among all the algorithms. Although the precision of ID3 is a little better than SVM, we still prefer SVM to ID3. It seems that the word information in the feature vectors is not so beneficial to decision tree algorithms as to SVM.

We also notice that SVM can perform very efficiently even with a large number of features. In the second set experiments, it usually takes several hours to train a decision tree model, but for SVM, the time cost is no more than 20 minutes. In addition, we can expect a better result by adding more information to SVM algorithm without worrying about the dimension disaster problem in other algorithms. Therefore, we decide to base our parsing model on SVM algorithm.

5 The SVM-based Parsing Model

5.1 Baseline Models

In this section, we will build 2 baseline models based on SVM for SP and PO boundary detection tasks respectively. By comprising the results of two different feature templates, we will illustrate how useful the word information is in our SVM based models.

One feature template we use here is the simple template which only takes the POS tag information as its features. The other one is the complex template which takes both word and tag information as its features. To make sure the results are comparable, we restrict the context window to 4 words.

In the SP boundary detection sub task, we got the following results:

Table 9. SP Boundary Detection Results.

Feature template	Precision	Recall	$F_{\beta=1}$
t-2t-1t1t2	76.25%	51.99%	61.83%
w-2w-1w1w2t-2t-1t1t2	82.25%	61.22%	70.19%

In the PO boundary detection sub task, we got the following results:

Table 10. PO Boundary Detection Results.

Feature template	Precision	Recall	$F_{\beta=1}$
t-2t-1t1t2	66.42%	65.27%	65.84%
w-2w-1w1w2t-2t-1t1t2	74.83%	86.99%	80.45%

By taking the complex feature template, we have achieved the best $F_{\beta=1}$ value of 70.19% in SP boundary detection experiment and 80.45% in PO experiment, both of which are much higher than those of the simple feature templates. From these results we can conclude that word information is very helpful in our SVM based models. Thus we will only use the feature templates with word information in the succeeding experiments.

5.2 Expanding the Context Window

In the previous section, the feature templates we use are restricted to a context window of 4 words, which might not be large enough to detect the boundaries between complex chunks. For example, when parsing the sentence “[P 派出/v $_1$ [O 精干/a $_2$ 的/u $_3$ 民族/n $_4$ 政策/n $_5$ 宣传/vN $_6$ 人员 /n”, the algorithm fails to detect the PO boundary at position 1. If we expand the context window to the noun word “民族/n”, some of these errors may disappear. In the following experiments, we will expand the context window from a size of 4 words to 10 words, and make a comparison between the different results.

The 4 feature templates used here are listed below:

T1: w-2w-1w1w2t-2t-1t1t2,

T2: w-3w-2w-1w1w2w3t-3t-2t-1t1t2t3,

T3: w-4w-3w-2w-1w1w2w3w4t-4t-3t-2t-1t1t2t3t4

T4: w-5w-4w-3w-2w-1w1w2w3w4w5t-5t-4t-3t-2t-1t1t2t3t4t5.

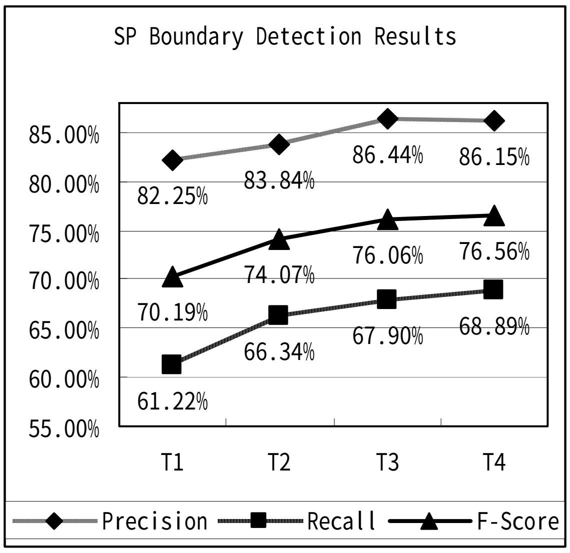

Figure 1. SP Boundary Detection Results.

As we have expected, the performance of SP boundary detection experiment has been improved as the context window expands from a size of 4 words to 8 words. However, the precision value meets its turning point at T3 after which it goes down, while F-Score and recall value still keep rising. From the curves shown in figure 1, we can find that the expansion of context window size from 4 words to 6 words has an obvious improvement for performance, and after that only F-Score and recall could be improved.

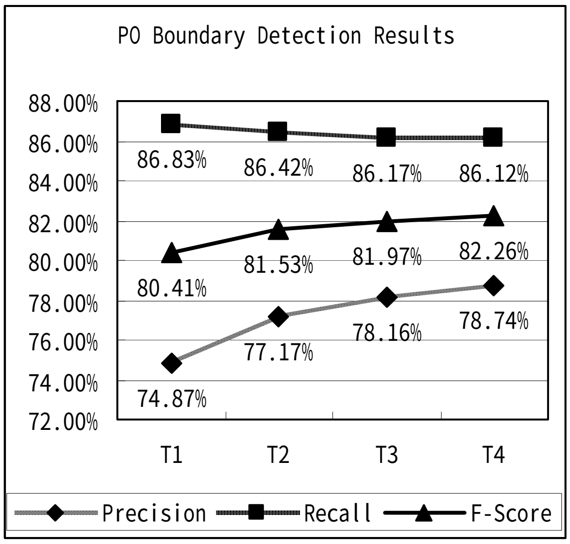

Figure 2. SP Boundary Detection Results.

In contrast to the significant improvement we have achieved in the SP experiments, the results of PO experiments are not so exciting. As the context window expands, the precision value keeps rising while the recall value keeps declining. Fortunately, we have obtained a very slight increase of F-Score from these efforts.

Although it is very difficult to improve the performance of PO boundary detection by simply expanding the context window, we’ve still got a better result than that of SP. If we examine the results of the two tasks carefully, we can find a very interesting difference between them: in SP boundary detection task, it’s very easier to get a better precision than recall, but in PO experiment, as the O chunks have a longer length, they are more likely to be cut into small pieces, and thus it’s easier to get a better recall than precision.

5.3 Error Analysis

In our experiments, the recall value can be simply raised by adding a positive bias value to the SVM classifier. However, we can’t do the same thing to improve the precision value. Thus, in the following analysis, we are only focus on the errors that deter the improvement of precision value.

There are 2 kinds of errors influencing the precision value of the test results: One is the wrongly detected chunk boundaries (WDB) within chunks (these chunk boundaries are detected by the program, but they don’t exist in the training data). This kind of error tends to cut a large chunk into several small pieces. The other is the misclassification of chunk boundary types (MBT) at the chunk boundaries (There exists a chunk boundary at that position, but chunk boundary type labeled by the program is wrong).

In the following analysis, by comparing the numbers of errors in the test results of T1 (w-2w-1w1w2t-2t-1t1t2) and T4 (w-5w-4w-3w-2w-1w1w2w3w4w5t-5t-4t-3t-2t-1t1t2t3t4t5), we will point out which kind of errors could be effectively eliminated by the expansion of context window and which of them couldn't. Through this analysis, we hope to get some knowledge of what efforts should be made in our further study.

In SP boundary detection task, we list the number of wrongly detected chunk boundaries (#WDB) and the corresponding chunk types (CT) where WDB arises in the following table.

Table 11. Wrongly Detected Chunk Boundaries in the Test Results of T1 and T4.

CT	#WDB of T1	#WDB of T4	T4-T1
O	17	18	1
S	17	18	1
D	7	6	-1
C	0	1	1
P	2	1	-1
T	1	1	0
Sum	44	45	1

From the above table, we find that the number of wrongly detected boundaries seems to be unchanged during the expansion of context window.

But when we refer to the second type of errors, the expansion of context window does help. We list the misclassified boundary types (MBT) and the error numbers (#MB) in the below table. In SP boundary detection task, MBT is wrongly recognized as boundary type SP.

Table 12. Misclassified Chunk Boundaries in the Test Results of T1 and T4.

MBT	#MB of T1	#MB of T4	T4-T1
OP	9	3	-6
JP	8	2	-6
DP	23	20	-3
SD	6	6	0
DS	1	1	0
Sum	47	32	-15

From the above table, we can find that the misclassifications of OP, JP and DP as SP have been largely reduced by expanding the context window, but the misclassifications of DS and SD remain the same. Therefore, we should try some other methods for D chunks in our future work.

In PO boundary detection task, the expansion of context window seems to be very effective. We list all the results in the below table:

Table 14. Wrongly Detected Chunk Boundaries in the Test Results of T1 and T4.

CT	#WDB of T1	#WDB of T4	T4-T1
O	251	196	-55
S	106	76	-30
D	92	55	-37
P	56	64	8
T	4	4	0
C	1	1	0
J	0	1	1
Sum	510	397	-113

It's very exciting to see that by expanding the window size, the number of WDB decreases dramatically from 510 to 397. But it fails to eliminate the WDB errors within P, T, C, and J chunks.

In PO boundary detection task, MBT is wrongly recognized as boundary type PO. We list the error data of T1 and T4 in the below table.

Table 13. Misclassified Chunk Boundaries in the Test Results of T1 and T4.

MBT	#MB of T1	#MB of T4	T4-T1
PJ	17	18	1
PD	9	9	0
PC	8	8	0
SP	6	6	0
PS	5	5	0
SD	5	4	-1
DP	3	2	-1
TS	3	3	0
OD	1	0	-1
PY	1	1	0
Sum	58	56	-2

In contrast to the results of SP boundary detection task, the MBT errors could not be largely reduced by simply expanding the context window. Therefore, we need to pay more attention to these problems in our future work.

6 Related works

After the work of Ramshaw and Marcus (1995) , many machine learning techniques have been applied to the basic chunking task, such as Support Vector Machines (Kudo and Matsumoto, 2001), Hidden Markov Model(Molina and Pla 2002), Memory Based Learning (Sang, 2002), Conditional Random Fields (Sha and Pereira, 2003), and so on. But only a small amount of attention has been paid to the functional chunk parsing problem.

Sandra and Erhard (2001) tried to construct the function-argument structures based on the pre-chunked input. They proposed a similarity based algorithm to assign the functional labels to complete syntactic structures, and achieved a precision of 89.73% and 90.40% for German and English respectively. Different from our top-down scheme, their function-argument structures are still constituted from bottom-up, and the pre-chunked input helps simplify the chunking process.

Elliott and Qiang Zhou (2001) used the BIO tagging system to identify the functional chunks in a sentence. In their experiments, they used C4.5 algorithm to build the parsing model, and focused their efforts on the selection of feature sets. After testing 5 sets of features, they have achieved the best f-measure of 0.741 by using feature set E which contains all the features in other feature sets. Instead of using BIO tags in our chunking task, we introduced chunk boundaries to help us identify the functional chunks, which could provide more relational information between the functional chunks.

7 Conclusions and Future Works

In this paper, we have applied functional chunks to Chinese shallow parsing. Since the functional chunks have the properties of linearity and exhaustibility, we can formulate the functional chunk parsing problem as a boundary detection task. By applying the divide-and-conquer strategy, we have further decomposed the parsing model into a series of sub modules, each of which is only in charge of one boundary type. In this way, we provide a very flexible framework within which different machine learning techniques could be applied. In our experiments, we build two sub modules based on SVM for solving the SP and PO boundary detection tasks. Thanks to the good generalization performance and high efficiency of SVM, we can successfully deal with a large number of features. By expanding the context window, we have achieved the best F-Score of 76.56% and 82.26 for SP and PO boundary detection tasks.

The 2 sub modules we have built are only parts of the Chinese functional chunk parser. Although the results we have got here seem somewhat coarse, they could already be used in some simple tasks. In the future, we will build the other sub modules for the remaining types of the chunk boundaries. After all these work, there may be some inconsistent chunk boundaries in the results, thus we need to solve the inconsistency problems and try to identify all the functional chunks in a sentence by combining these chunk boundaries.

Acknowledgements

This work was supported by the Chinese National Science Foundation (Grant No. 60573185, 60520130299).

References

Elliott Franco Drábek and Qiang Zhou. 2001. Experiments in Learning Models for Functional Chunking of Chinese Text. *IEEE International Workshop on Natural Language processing and Knowledge Engineering*, Tucson, Arizona, pages 859-864.

E.F. Tjong Kim Sang. 2002. Memory-based shallow parsing, *Journal of Machine Learning Research 2*, pages 559-594.

F. Sha and F. Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of Human Language Technology Conference / North American Chapter of the Association for Computational Linguistics annual meeting*.

Ian H. Witten and Eibe Frank. 2005. *Data Mining: Practical machine learning tools and techniques*, 2nd Edition, Morgan Kaufmann, San Francisco.

Taku Kudo and Yuji Matsumoto. 2001. Chunking with support vector machines. *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics*. Pittsburgh, PA.

Lance Ramshaw and Mitch Marcus. 1995. Text chunking using transformation-based learning. In *Proceedings of the Third Workshop on Very Large Corpora*, pages 82—94.

Quinlan, J. Ross. 1986. Induction of decision trees. *Machine Learning*, 1(1), pages 81-106.

Quinlan, J. Ross. 1993. C4.5: *Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann.

Steven Abney. 1991. Parsing by chunks. In *Principle-Based Parsing*, Kluwer Academic Publishers, Dordrecht, pages 257—278.

Sandra Kübler and Erhard W. Hinrichs. 2001. From chunks to function-argument structure: A similarity-based approach. In *Proceedings of ACL/EACL 2001*, Toulouse, France, 2001, pages 338 - 345.

Thorsten Joachims. 1999. *Advances in Kernel Methods - Support Vector Learning*, chapter Making large-Scale SVM Learning Practical. MIT-Press.

Vladimir N. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer, New York.