Improving Graph-based Dependency Parsing with Decision History

Wenliang Chen[†], **Jun'ichi Kazama**[†], **Yoshimasa Tsuruoka**[†‡] **and Kentaro Torisawa**[†]
[†]Language Infrastructure Group, MASTAR Project, NICT
{chenwl, kazama, torisawa}@nict.go.jp
[‡]School of Information Science, JAIST
tsuruoka@jaist.ac.jp

Abstract

This paper proposes an approach to improve graph-based dependency parsing by using decision history. We introduce a mechanism that considers short dependencies computed in the earlier stages of parsing to improve the accuracy of long dependencies in the later stages. This relies on the fact that short dependencies are generally more accurate than long dependencies in graph-based models and may be used as features to help parse long dependencies. The mechanism can easily be implemented by modifying a graph-based parsing model and introducing a set of new features. The experimental results show that our system achieves state-of-the-art accuracy on the standard PTB test set for English and the standard Penn Chinese Treebank (CTB) test set for Chinese.

1 Introduction

Dependency parsing is an approach to syntactic analysis inspired by dependency grammar. In recent years, interest in this approach has surged due to its usefulness in such applications as machine translation (Nakazawa et al., 2006), information extraction (Culotta and Sorensen, 2004).

Graph-based parsing models (McDonald and Pereira, 2006; Carreras, 2007) have achieved state-of-the-art accuracy for a wide range of languages as shown in recent CoNLL shared tasks (Buchholz et al., 2006; Nivre et al., 2007). However, to make parsing tractable, these models are forced to restrict features over a very limited history of parsing decisions (McDonald and Pereira, 2006; McDonald and Nivre, 2007). Previous work showed that rich features over a wide range of decision history can lead to significant improvements in accuracy for transition-based models (Yamada and Matsumoto, 2003a; Nivre et al., 2004).

In this paper, we propose an approach to improve graph-based dependency parsing by using decision history. Here, we make an assumption: the dependency relations between words with a short distance are more reliable than ones between words with a long distance. This is supported by the fact that the accuracy of short dependencies is in general greater than that of long dependencies as reported in McDonald and Nivre (2007) for graph-based models. Our idea is to use decision history, which is made in previous scans in a bottom-up procedure, to help parse other words in later scans. In the bottom-up procedure, short dependencies are parsed earlier than long dependencies. Thus, we introduce a mechanism in which we treat short dependencies built earlier as decision history to help parse long dependencies in later stages. It can easily be implemented by modifying a graph-based parsing model and designing a set of features for the decision history.

To demonstrate the effectiveness of the proposed approach, we present experimental results on English and Chinese data. The results indicate that the approach greatly improves the accuracy and that richer history-based features indeed make large contributions. The experimental results show that our system achieves state-of-the-art accuracy on the data.

2 Motivation

In this section, we present an example to show the idea of using decision history in a dependency parsing procedure.

Suppose we have two sentences in Chinese, as shown in Figures 1 and 2, where the correct dependencies are represented by the directed links. For example, in Figure 1 the directed link from

Coling 2008: Poster Volume, pages 126–134,
Beijing, August 2010

w_3:买(bought) to w_5:书(books) mean that w_3 is the head and w_5 is the dependent. In Chinese, the relationship between clauses is often not made explicit and two clauses may simply be put together with only a comma (Li and Thompson, 1997). This makes it hard to parse Chinese sentences with several clauses.

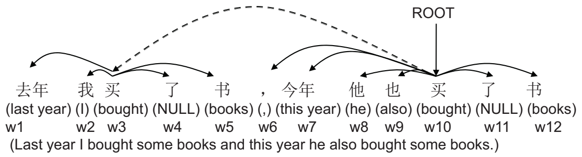

ROOT
去年 我 买 了 书 ， 今年 他 也 买 了 书
(last year) (I) (bought) (NULL) (books) (,) (this year) (he) (also) (bought) (NULL) (books)
w1 w2 w3 w4 w5 w6 w7 w8 w9 w10 w11 w12
(Last year I bought some books and this year he also bought some books.)

Figure 1: Example A

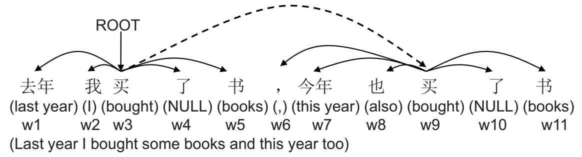
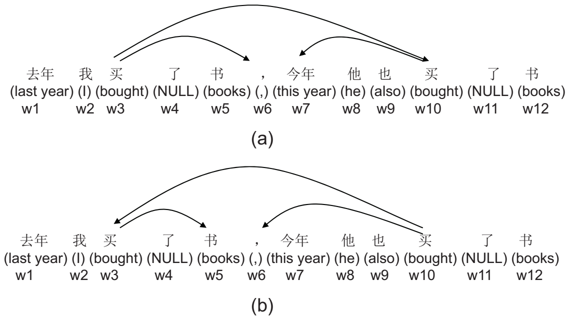

ROOT
去年 我 买 了 书 ， 今年 也 买 了 书
(last year) (I) (bought) (NULL) (books) (,) (this year) (also) (bought) (NULL) (books)
w1 w2 w3 w4 w5 w6 w7 w8 w9 w10 w11
(Last year I bought some books and this year too)

Figure 2: Example B

If we employ a graph-based parsing model, such as the model of (McDonald and Pereira, 2006; Carreras, 2007), it is difficult to assign the relations between w_3 and w_{10} in Example A and between w_3 and w_9 in Example B. For simplicity, we use w_i^A to refer to w_i of Example A and w_i^B to refer to w_i of Example B in what follows.

The key point is whether the second clauses are independent in the sentences. The two sentences are similar except that the second clause of Example A is an independent clause but that of Example B is not. w_{10}^A is the root of the second clause of Example A with subject w_8^A, while w_9^B is the root of the second clause of Example B, but the clause does not have a subject. These mean that the correct decisions are to assign w_{10}^A as the head of w_3^A and w_3^B as the head of w_9^B, as shown by the dash-dot-lines in Figures 1 and 2.

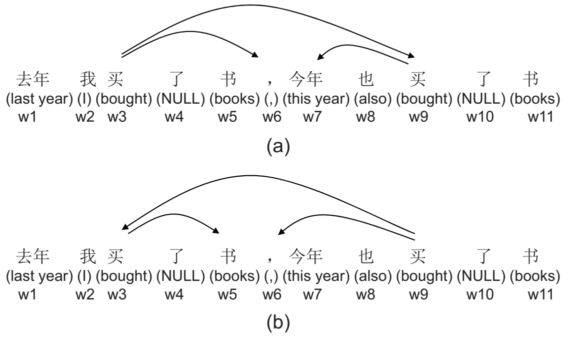
However, the model can use very limited information. Figures 3-(a) and 4-(a) show the right dependency relation cases and Figures 3-(b) and 4-(b) show the left direction cases. For the right direction case of Example A, the model has the information about w_3^A's rightmost child w_5^A and w_{10}^A's leftmost child w_6^A inside w_3^A and w_{10}^A, but it does not have information about the other children (such as w_8^A) of w_3^A and w_{10}^A, which may be useful for judging the relation between w_3^A and w_{10}^A. The parsing model can not find the difference between the syntactic structures of two sentences for pairs (w_3^A, w_{10}^A) and (w_3^B, w_9^B). If we can provide the information about the other children of w_3^A and w_{10}^A to the model, it becomes easier to find the correct direction between w_3^A and w_{10}^A.


去年 我 买 了 书 ， 今年 他 也 买 了 书
(last year) (I) (bought) (NULL) (books) (,) (this year) (he) (also) (bought) (NULL) (books)
w1 w2 w3 w4 w5 w6 w7 w8 w9 w10 w11 w12
去年 我 买 了 书 ， 今年 他 也 买 了 书
(last year) (I) (bought) (NULL) (books) (,) (this year) (he) (also) (bought) (NULL) (books)
w1 w2 w3 w4 w5 w6 w7 w8 w9 w10 w11 w12

(a)

(b)

Figure 3: Example A: two directions


去年 我 买 了 书 ， 今年 也 买 了 书
(last year) (I) (bought) (NULL) (books) (,) (this year) (also) (bought) (NULL) (books)
w1 w2 w3 w4 w5 w6 w7 w8 w9 w10 w11
去年 我 买 了 书 ， 今年 也 买 了 书
(last year) (I) (bought) (NULL) (books) (,) (this year) (also) (bought) (NULL) (books)
w1 w2 w3 w4 w5 w6 w7 w8 w9 w10 w11

(a)

(b)

Figure 4: Example B: two directions

Next, we show how to use decision history to help parse w_3^A and w_{10}^A of Example A.

In a bottom up procedure, the relations between the words inside [w_3^A, w_{10}^A] are built as follows before the decision for w_3^A and w_{10}^A. In the first round, we build relations for neighboring words (word distance[1]=1), such as the relations between w_3^A and w_4^A and between w_4^A and w_5^A. In the second round, we build relations for words of distance 2, and then for longer distance words until all the possible relations between the inside words are built. Figure 5 shows all the possible relations inside [w_3^A, w_{10}^A] that we can build. To simplify, we use undirected links to refer to both directions

[1] Word distance between w_i and w_j is $|j - i|$.

of dependency relations between words in the figure.

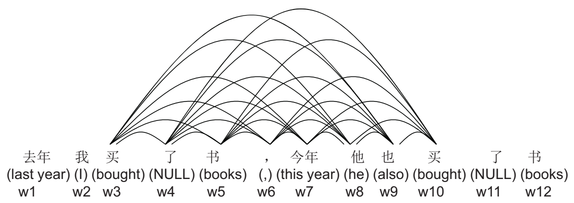

去年 我 买 了 书 ， 今年 他 也 买 了 书
(last year) (I) (bought) (NULL) (books) (,) (this year) (he) (also) (bought) (NULL) (books)
w1 w2 w3 w4 w5 w6 w7 w8 w9 w10 w11 w12

Figure 5: Example A: first step

Then given those inside relations, we choose the inside structure with the highest score for each direction of the dependency relation between w_3^A and w_{10}^A. Figure 6 shows the chosen structures. Note that the chosen structures for two directions could either be identical or different. In Figure 6-(a) and -(b), they are different.

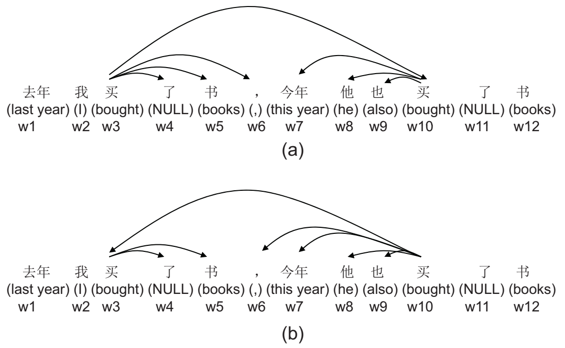

去年 我 买 了 书 ， 今年 他 也 买 了 书
(last year) (I) (bought) (NULL) (books) (,) (this year) (he) (also) (bought) (NULL) (books)
w1 w2 w3 w4 w5 w6 w7 w8 w9 w10 w11 w12
(a)
去年 我 买 了 书 ， 今年 他 也 买 了 书
(last year) (I) (bought) (NULL) (books) (,) (this year) (he) (also) (bought) (NULL) (books)
w1 w2 w3 w4 w5 w6 w7 w8 w9 w10 w11 w12
(b)

Figure 6: Example A: second step

Finally, we use the chosen structures as decision history to help parse w_3^A and w_{10}^A. For example, the fact that w_8^A is a dependent of w_{10}^A is a clue that suggests that the second clause may be independent. This results in w_{10}^A being the head of w_3^A.

This simple example shows how to use the decision history to help parse the long distance dependencies.

3 Background: graph-based parsing models

Before we describe our method, we briefly introduce the graph-based parsing models. We denote input sentence w by $w = (w_0, w_1, ..., w_n)$, where $w_0 = ROOT$ is an artificial root token inserted at the beginning of the sentence and does not depend on any other token in w and w_i refers to a word.

We employ the second-order projective graph-based parsing model of Carreras (2007), which is an extension of the projective parsing algorithm of Eisner (1996).

The parsing algorithms used in Carreras (2007) independently find the left and right dependents of a word and then combine them later in a bottom-up style based on Eisner (1996). A subtree that spans the words in $[s, t]$ (and roots at s or t) is represented by chart item $[s, t, right/left, C/I]$, where right (left) indicates that the root of the subtree is s (t) and C means that the item is *complete* while I means that the item is *incomplete* (McDonald, 2006). Here, *complete item* in the right (left) direction means that the words other than s (t) cannot have dependents outside $[s, t]$ and *incomplete item* in the right (left) direction, on the other hand, means that t (s) may have dependents outside $[s, t]$. In addition, t (s) is the direct dependent of s (t) in the incomplete item with the right (left) direction.

Larger chart items are created from pairs of smaller chart items by the bottom-up procedure. Figure 7 illustrates the cubic parsing actions of the Eisner's parsing algorithm (Eisner, 1996) in the right direction, where s, r, and t refer to the start and end indices of the chart items. In Figure 7-(a), all the items on the left side are complete and represented by triangles, where the triangle of [s, r] is complete item $[s, r, \rightarrow, C]$ and the triangle of [$r + 1$, t] is complete item $[r + 1, t, \leftarrow, C]$. Then the algorithm creates incomplete item $[s, t, \rightarrow, I]$ (trapezoid on the right side of Figure 7-(a)) by combining the chart items on the left side. This action builds the dependency from s to t. In Figure 7-(b), the item of [s, r] is incomplete and the item of [r, t] is complete. Then the algorithm creates complete item $[s, t, \rightarrow, C]$. For the left direction case, the actions are similar. Note that only the actions of creating the incomplete chart items build new dependency relations between words, while the ones of creating the complete items merge the existing structures without building new relations.

Once the parser has considered the dependency relations between words of distance 1, it goes on to dependency relations between words of distance 2, and so on by the parsing actions. For words of distance 2 and greater, it considers every possible partition of the structures into two parts and chooses the one with the highest score for each direction. The score is the sum of the feature weights of the chart items. The features are designed over edges of dependency trees and the weights are given by model parameters (McDonald and Pereira, 2006; Carreras, 2007). We store the obtained chart items in a table. The chart item includes the information on the optimal splitting point of itself. Thus, by looking up the table, we can obtain the best tree structure (with the highest score) of any chart item.

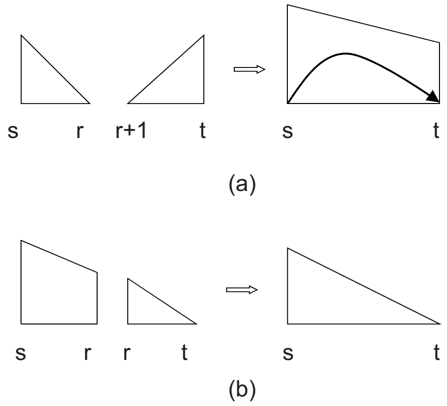


Figure 7: Cubic parsing actions of Eisner (1996)

4 Parsing with decision history

As mentioned above, the actions for creating the incomplete items build the relations between words. In this study, we only consider using history information when creating incomplete items.

4.1 Decision history

Suppose we are going to compute the scores of the relations between w_s and w_t. There are two possible directions for them.

By using the bottom-up style algorithm, the scores of the structures between words with distance $< |s-t|$ are computed in previous scans and the structures are stored in the table. We divide the decision history into two types: history-inside and history-outside. The history-inside type is the decision history made inside [s,t] and the history-outside type is the history made outside [s,t].

4.1.1 History-inside

We obtain the structure with the highest score for each direction of the dependency between w_s and w_t. Figure 8-(b) shows the best solution (with the highest score) of the left direction, where the structure is split into two parts, $[s, r_1, \rightarrow, C]$ and $[r_1 + 1, t, \leftarrow, C]$. Figure 8-(c) shows the best solution of the right case, where the structure is split into two parts, $[s, r_2, \rightarrow, C]$ and $[r_2 + 1, t, \leftarrow, C]$.

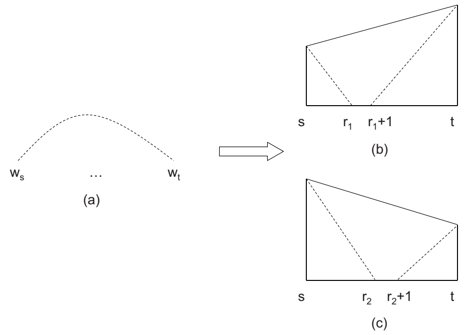


Figure 8: History-inside

By looking up the table, we have a subtree that roots at w_s on the right side of w_s and a subtree that roots at w_t on the left side of w_t. We use these structures as the information on history-inside.

4.1.2 History-outside

For history-outside, we try to obtain the subtree that roots at w_s on the left side of w_s and the one that roots at w_t on the right side of w_t. However, compared to history-inside, obtaining history-outside is more complicated because we do not know the boundaries and the proper structures of the subtrees. Here, we use an simple heuristic method to find a subtree whose root is at w_s on the left side of w_s and one whose root is at w_t on the right side of w_t.

We introduce two assumptions: 1) The structure within a sub-sentence [2] is more reliable than the one that goes across from sub-sentences. 2) More context (more words) can result in a better solution for determining subtree structures.

[2] To simplify, we split one sentence into sub-sentences with punctuation marks.

Algorithm 1 Searching for history-outside boundaries

```
Input: $w, s, t$
for $k = s - 1$ to $1$ do
    if(isPunct($w_k$)) break;
    if($s - k >= t - s - 1$) break
end for
$b_s = k$
for $k = t + 1$ to $|w|$ do
    if(isPunct($w_k$)) break;
    if($k - t >= t - s - 1$) break
end for
$b_t = k$
Output: $b_s, b_t$
```

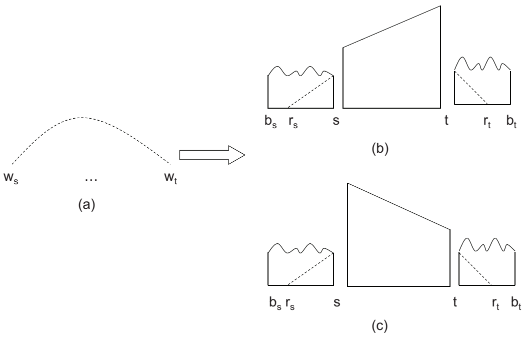


Figure 9: History-outside

Under these two assumptions, Algorithm 1 shows the procedure for searching for history-outside boundaries, where b_s is the boundary for for the descendants on the left side of w_s , b_t is the boundary for searching the descendants on the right side of w_t, and *isPunct* is the function that checks if the word is a punctuation mark. b_s should be in the same sub-sentence with s and $|s - b_s|$ should be less than $|t - s|$. b_t should be in the same sub-sentence with t and $|b_t - t|$ should be less than $|t - s|$.

Next we try to find the subtree structures. First, we collect the part-of-speech (POS) tags of the heads of all the POS tags in training data and remove the tags that occur fewer than 10 times. Then, we determine the directions of the relations by looking up the collected list. For b_s and s, we check if the POS tag of w_s could be the head tag of the POS tag of w_{b_s} by looking up the list. If so, the direction d is $\leftarrow$. Otherwise, we check if the POS tag of w_{b_s} could be the head tag of the POS tag of w_s. If so, d is $\rightarrow$, else d is $\leftarrow$. Finally, we obtain the subtree of w_s from chart item $[b_s, s, d, I]$. Similarly, we obtain the subtree of w_t. Figure 9 shows the history-outside information for w_s and w_t, where the relation between w_{b_s} and w_s and the relation between w_{b_t} and w_t will be determined by the above method. We have subtree $[r_s, s, left, C]$ that roots at w_s on the left side of w_s and subtree $[t, r_t, right, C]$ that roots at w_t on the right side of w_t in Figure 9-(b) and (c).

4.2 Parsing algorithm

Then, we explain how to use these decision history in the parsing algorithm. We use L_{st} to represent the scores of basic features for the left direction and R_{st} for the right case. Then we design history-based features (described in Section 4.3) based on the history-inside and history-outside information, as mentioned above. Finally, we update the scores with the ones of the history-based features by the following equations:

$$L^{+}_{st} = L_{st} + L^{df}_{st} \quad (1)$$

$$R^{+}_{st} = R_{st} + R^{df}_{st} \quad (2)$$

where L^{+}_{st} and R^{+}_{st} refer to the updated scores, L^{df}_{st} and R^{df}_{st} refer to the scores of the history-based features.

Algorithm 2 Parsing algorithm

```
1:  Initialization: $V[s, s, dir, I/C] = 0.0 \; \forall s, dir$
2:  for $k = 1$ to $n$ do
3:      for $s = 0$ to $n - k$ do
4:          $t = s + k$
5:          % Create incomplete items
6:          $L_{st}$=$V[s, t, \leftarrow, I]$= $max_{s \leq r < t} VI(r)$;
7:          $R_{st}$=$V[s, t, \rightarrow, I]$= $max_{s \leq r < t} VI(r)$;
8:          Calculate $L^{df}_{st}$ and $R^{df}_{st}$;
9:          % Update the scores of incomplete chart items
10:         $V[s, t, \leftarrow, I]$=$L^{+}_{st}$=$L_{st}$ + $L^{df}_{st}$
11:         $V[s, t, \rightarrow, I]$=$R^{+}_{st}$=$R_{st}$ + $R^{df}_{st}$
12:         % Create complete items
13:         $V[s, t, \leftarrow, C]$= $max_{s \leq r < t} VC(r)$;
14:         $V[s, t, \rightarrow, C]$= $max_{s < r \leq t} VC(r)$;
15:     end for
16: end for
```

Algorithm 2 is the parsing algorithm with the history-based features, where $V[s, t, dir, I/C]$ refers to the score of chart item $[s, t, dir, I/C]$, $VI(r)$ is a function to search for the optimal sibling and grandchild nodes for the incomplete items (line 6 and 7) (Carreras, 2007) given the splitting point r and return the score of the structure, and $VC(r)$ is a function to search for the optimal grandchild node for the complete items (line 13 and 14). Compared with the parsing algorithms of Carreras (2007), Algorithm 2 uses history information by adding line 8, 10, and 11.

In Algorithm 2, it first creates chart items with distance 1, then goes on to chart items with distance 2, and so on. In each round, it searches for the structures with the highest scores for incomplete items shown at line 6 and 7 of Algorithm 2. Then we update the scores with the history-based features by Equation 1 and Equation 2 at line 10 and 11 of Algorithm 2. However, note that we can not guarantee to find the candidate with the highest score with Algorithm 2 because new features violate the assumptions of dynamic programming.

4.3 History-based features

In this section, we design features that capture the history information in the recorded decisions.

For a dependency between two words, say s and t, there are four subtrees that root at s or t. We design the features by combining s, t with each child of s and t in the subtrees. The feature templates are shown as follows: (In the following, c means one of the children of s and t, and the nodes in the templates are expanded to their lexical form and POS tags to obtain actual features.):

C+Dir this feature template is a 2-tuple consisting of (1) a c node and (2) the direction of the dependency.

C+Dir+S/C+Dir+T this feature template is a 3-tuple consisting of (1) a c node, (2) the direction of the dependency, and (3) a s or t node.

C+Dir+S+T this feature template is a 4-tuple consisting of (1) a c node, (2) the direction of the dependency, (3) a s node, and (4) a t node.

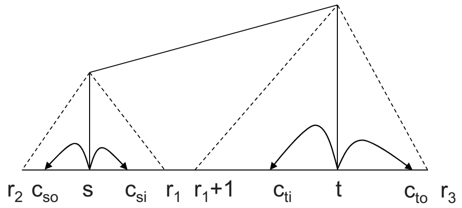


Figure 10: Structure of decision history

We use SHI to represent the subtree of s in the history-inside, THI to represent the one of t in the history-inside, SHO to represent the one of s in the history-outside, and THO to represent the one of t in the history-outside. Based on the subtree types, the features are divided into four sets: F_{SHI}, F_{THI}, F_{SHO}, and F_{THO} refer to the features related to the children that are in subtrees SHI, THI, SHO, and THO respectively.

Figure 10 shows the structure of decision history of a left dependency (between s and t) relation. For the right case, the structure is similar. In the figure, SHI is chart item $[s, r_1, \rightarrow, C]$, THI is chart item $[r_1 + 1, t, \leftarrow, C]$, SHO is chart item $[r_2, s, \leftarrow, C]$, and THO is chart item $[t, r_3, \rightarrow, C]$. We use c_{si}, c_{ti}, c_{so}, and c_{to} to represent a child of s/t in subtrees SHI, THI, SHO, and THO respectively. The lexical form features of F_{SHI} and F_{SHO} are listed as examples in Table 1, where "L" refers to the left direction. We can also expand the nodes in the templates to the POS tags. Compared with the algorithm of Carreras (2007) that only considers the furthest children of s and t, Algorithm 2 considers all the children.

Table 1: Lexical form features of F_{SHI} and F_{SHO}

template	F_{SHI}	F_{SHO}
C+DIR	word-c_{si}+L	word-c_{so}+L
C+DIR+S	word-c_{si}+L+word-s	word-c_{so}+L+word-s
C+DIR+T	word-c_{si}+L+word-t	word-c_{so}+L+word-t
C+DIR +S+T	word-c_{si}+L +word-s+word-t	word-c_{so}+L +word-s+word-t

4.4 Policy of using history

In practice, we define several policies to use the history information for different word pairs as follows:

- All: Use the history-based features for all the word pairs without any restriction.
- Sub-sentences: use the history-based features only for the relation of two words from sub-sentences. Here, we use punctuation marks to split sentences into sub-sentences.
- Distance: use the history-based features for the relation of two words within a predefined distance. We set the thresholds to 3, 5, and 10.

5 Experimental results

In order to evaluate the effectiveness of the history-based features, we conducted experiments on Chinese and English data.

For English, we used the Penn Treebank (Marcus et al., 1993) in our experiments and the tool "Penn2Malt"[3] to convert the data into dependency structures using a standard set of head rules (Yamada and Matsumoto, 2003a). To match previous work (McDonald and Pereira, 2006; Koo et al., 2008), we split the data into a training set (sections 2-21), a development set (Section 22), and a test set (section 23). Following the work of Koo et al. (2008), we used the MXPOST (Ratnaparkhi, 1996) tagger trained on training data to provide part-of-speech tags for the development and the test set, and we used 10-way jackknifing to generate tags for the training set.

For Chinese, we used the Chinese Treebank (CTB) version 4.0[4] in the experiments. We also used the "Penn2Malt" tool to convert the data and created a data split: files 1-270 and files 400-931 for training, files 271-300 for testing, and files 301-325 for development. We used gold standard segmentation and part-of-speech tags in the CTB. The data partition and part-of-speech settings were chosen to match previous work (Chen et al., 2008; Yu et al., 2008).

We measured the parser quality by the unlabeled attachment score (UAS), i.e., the percentage of tokens with the correct HEAD [5]. And we also evaluated on complete dependency analysis.

In our experiments, we implemented our systems on the MSTParser[6] and extended with the parent-child-grandchild structures (McDonald and Pereira, 2006; Carreras, 2007). For the baseline systems, we used the first- and second-order (parent-sibling) features that were used in McDonald and Pereira (2006) and other second-order features (parent-child-grandchild) that were used in Carreras (2007). In the following sections, we call the second-order baseline systems Baseline and our new systems OURS.

5.1 Results with different feature settings

In this section, we test our systems with different settings on the development data.

Table 2: Results with different policies

	Chinese	English
Baseline	89.04	92.43
D_1	88.73	92.27
D_3	88.90	92.36
D_5	89.10	92.59
D_{10}	89.32	92.57
D_{sub}	89.57	92.63

Table 2 shows the parsing results when we used different policies defined in Section 4.4 with all the types of features, where D_{sub} refers to applying the policy: sub-sentence, D_1 refers to applying the policy: all, and $D_{3|5|10}$ refers to applying the policy: distance with the predefined distance 3, 5, or 10. The results indicated that the accuracies of our systems decreased if we used the history information for short distance words. The system with D_{sub} performed the best.

Table 3: Results with different types of Features

	Chinese	English
Baseline	89.04	92.43
$+F_{SHI}$	89.14	92.53
$+F_{THI}$	89.33	92.35
$+F_{SHO}$	89.25	92.47
$+F_{THO}$	88.99	92.54

Then we investigated the effect of different types of the history-based features. Table 3 shows the results with policy D_{sub}. From the table, we found that F_{THI} provided the largest improvement for Chinese and F_{THO} performed the best for English.

In what follows, we used D_{sub} as the policy for all the languages, the features $F_{SHI} + F_{THI} + F_{SHO}$ for Chinese, and the features $F_{SHI} + F_{SHO} + F_{THO}$ for English.

5.2 Main results

The main results are shown in the upper parts of Tables 4 and 5, where the improvements by OURS over the Baselines are shown in parentheses. The results show that OURS provided better performance over the Baselines by 1.02 points for Chi-

[3] http://w3.msi.vxu.se/˜nivre/research/Penn2Malt.html

[4] http://www.cis.upenn.edu/˜chinese/.

[5] As in previous work, English evaluation ignores any token whose gold-standard POS tag is one of {'' `` : , .} and Chinese evaluation ignores any token whose tag is "PU".

[6] http://mstparser.sourceforge.net

Table 4: Results for Chinese

	UAS	Complete
Baseline	88.41	48.85
OURS	89.43(+1.02)	50.86
OURS+STACK	89.53	49.42
Zhao2009	87.0	–
Yu2008	87.26	–
STACK	88.95	49.42
Chen2009	89.91	48.56

nese and 0.29 points for English. The improvements of (OURS) were significant in McNemar's Test with $p < 10^{-4}$ for Chinese and $p < 10^{-3}$ for English.

5.3 Comparative results

Table 4 shows the comparative results for Chinese, where Zhao2009 refers to the result of (Zhao et al., 2009), Yu2008 refers to the result of Yu et al. (2008), Chen2009 refers to the result of Chen et al. (2009) that is the best reported result on this data, and STACK refers to our implementation of the combination parser of Nivre and McDonald (2008) using our baseline system and the MALTParser[7]. The results indicated that OURS performed better than Zhao2009, Yu2008, and STACK, but worse than Chen2009 that used large-scale unlabeled data (Chen et al., 2009). We also implemented the combination system of OURS and the MALTParser, referred as OURS+STACK in Table 4. The new system achieved further improvement. In future work, we can combine our approach with the parser of Chen et al. (2009).

Table 5 shows the comparative results for English, where Y&M2003 refers to the parser of Yamada and Matsumoto (2003b), CO2006 refers to the parser of Corston-Oliver et al. (2006), Z&C 2008 refers to the combination system of Zhang and Clark (2008), STACK refers to our implementation of the combination parser of Nivre and McDonald (2008), KOO2008 refers to the parser of Koo et al. (2008), Chen2009 refers to the parser of Chen et al. (2009), and Suzuki2009 refers to the parser of Suzuki et al. (2009) that is the best reported result for this data. The results shows that OURS outperformed the first two systems that were based on single models. Z&C 2008 and STACK were the combination systems of graph-based and transition-based models. OURS performed better than Z&C 2008, but worse than STACK. The last three systems that used large-scale unlabeled data performed better than OURS.

Table 5: Results for English

	UAS	Complete
Baseline	91.92	44.28
OURS	92.21 (+0.29)	45.24
Y&M2003	90.3	38.4
CO2006	90.8	37.6
Z&C2008	92.1	45.4
STACK	92.53	47.06
KOO2008	93.16	–
Chen2009	93.16	47.15
Suzuki2009	93.79	–

6 Related work

There are several studies that tried to overcome the limited feature scope of graph-based dependency parsing models .

Nakagawa (2007) proposed a method to deal with the intractable inference problem in a graph-based model by introducing the Gibbs sampling algorithm. Compared with their approach, our approach is much simpler yet effective. Hall (2007) used a re-ranking scheme to provide global features while we simply augment the features of an existing parser.

Nivre and McDonald (2008) and Zhang and Clark (2008) proposed stacking methods to combine graph-based parsers with transition-based parsers. One parser uses dependency predictions made by another parser. Our results show that our approach can be used in the stacking frameworks to achieve higher accuracy.

7 Conclusions

This paper proposes an approach for improving graph-based dependency parsing by using the decision history. For the graph-based model, we design a set of features over short dependencies computed in the earlier stages to improve the accuracy of long dependencies in the later stages. The results demonstrate that our proposed approach outperforms baseline systems by 1.02 points for Chinese and 0.29 points for English.

[7] http://www.maltparser.org/

References

Buchholz, S., E. Marsi, A. Dubey, and Y. Krymolowski. 2006. CoNLL-X shared task on multilingual dependency parsing. *Proceedings of CoNLL-X.*

Carreras, X. 2007. Experiments with a higher-order projective dependency parser. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 957–961.

Chen, WL., D. Kawahara, K. Uchimoto, YJ. Zhang, and H. Isahara. 2008. Dependency parsing with short dependency relations in unlabeled data. In *Proceedings of IJCNLP 2008.*

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

Corston-Oliver, S., A. Aue, Kevin. Duh, and Eric Ringger. 2006. Multilingual dependency parsing using bayes point machines. In *HLT-NAACL2006.*

Culotta, A. and J. Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of ACL 2004*, pages 423–429.

Eisner, J. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proc. of COLING 1996*, pages 340–345.

Hall, Keith. 2007. K-best spanning tree parsing. In *Proc. of ACL 2007*, pages 392–399, Prague, Czech Republic, June. Association for Computational Linguistics.

Koo, T., X. Carreras, and M. Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of ACL-08: HLT*, Columbus, Ohio, June.

Li, Charles N. and Sandra A. Thompson. 1997. *Mandarin Chinese - A Functional Reference Grammar.* University of California Press.

Marcus, M., B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguisticss*, 19(2):313–330.

McDonald, R. and J. Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of EMNLP-CoNLL*, pages 122–131.

McDonald, R. and F. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proc. of EACL2006.*

McDonald, Ryan. 2006. *Discriminative Training and Spanning Tree Algorithms for Dependency Parsing.* Ph.D. thesis, University of Pennsylvania.

Nakagawa, Tetsuji. 2007. Multilingual dependency parsing using global features. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 952–956.

Nakazawa, T., K. Yu, D. Kawahara, and S. Kurohashi. 2006. Example-based machine translation based on deeper NLP. In *Proceedings of IWSLT 2006*, pages 64–70, Kyoto, Japan.

Nivre, J. and R. McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proceedings of ACL-08: HLT*, Columbus, Ohio, June.

Nivre, J., J. Hall, and J. Nilsson. 2004. Memory-based dependency parsing. In *Proc. of CoNLL 2004*, pages 49–56.

Nivre, J., J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 915–932.

Ratnaparkhi, A. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of EMNLP*, pages 133–142.

Suzuki, Jun, Hideki Isozaki, Xavier Carreras, and Michael Collins. 2009. An empirical study of semi-supervised structured conditional models for dependency parsing. In *Proc. of EMNLP 2009*, pages 551–560, Singapore, August. Association for Computational Linguistics.

Yamada, H. and Y. Matsumoto. 2003a. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT2003*, pages 195–206.

Yamada, H. and Y. Matsumoto. 2003b. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT2003*, pages 195–206.

Yu, K., D. Kawahara, and S. Kurohashi. 2008. Chinese dependency parsing with large scale automatically constructed case structures. In *Proceedings of Coling 2008*, pages 1049–1056, Manchester, UK, August.

Zhang, Y. and S. Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *Proceedings of EMNLP 2008*, pages 562–571, Honolulu, Hawaii, October.

Zhao, Hai, Yan Song, Chunyu Kit, and Guodong Zhou. 2009. Cross language dependency parsing using a bilingual lexicon. In *Proceedings of ACL-IJCNLP2009*, pages 55–63, Suntec, Singapore, August. Association for Computational Linguistics.