Transition-Based Dependency Parsing with Long Distance Collocations

Chenxi Zhu, Xipeng Qiu(✉), and Xuanjing Huang

Shanghai Key Laboratory of Intelligent Information Processing,
School of Computer Science, Fudan University, Shanghai, China
{13210240078,xpqiu,xjhuang}@fudan.edu.cn

Abstract. Long distance dependency relation is one of the main challenges for the state-of-the-art transition-based dependency parsing algorithms. In this paper, we propose a method to improve the performance of transition-based parsing with long distance collocations. With these long distance collocations, our method provides an approximate global view of the entire sentence, which is a little bit similar to top-down parsing. To further improve the accuracy of decision, we extend the set of parsing actions with two more fine-grained actions based on the types of arcs. Experimental results show that our method improve the performance of parsing effectively, especially for long sentence.

1 Introduction

Dependency parsing uses dependency representation of syntactic structure, which directly reflects relationships among the words in a sentence. In recent years, greedy transition-based dependency parsing systems are widely used in a variety of practical tasks, especially for web-scale data, because it runs fast and performs accurately [10,14].

However, it is well known that transition-based parsing systems tend to have lower accuracy for long sentence. The main reason is that they greedily execute shift-reduce actions, and all of their decisions are very local, only two or three input tokens are available to the parser. This leads to error propagation and worse performance when predicting long arcs [9]. For example, when an arc is constructed between two words, the modifier word should have no other child anymore. But it is difficult to check this condition especially in long distance dependency.

An improved approach is to use beam search instead of greedy decoding, in combination with a globally trained model that tries to minimize the loss over the entire sentence instead of a locally trained classifier that tries to maximize the accuracy of single decisions [16]. However, the drawback with this approach is that parsing speed is proportional to the size of the beam, which means that the most accurate transition-based parsers are not nearly as fast as the original greedy transition-based parsers.

Another line of research tries to retain the efficiency of greedy classifier-based parsing by utilizing richer non-local information [17]. However, these non-local

J. Li et al. (Eds.): NLPCC 2015, LNAI 9362, pp. 12–24, 2015.
DOI: 10.1007/978-3-319-25207-0_2

features is still in a context window and just use extra complex structures on the left context. Yamada and Matsumoto [14] also reported that the dependency accuracy depends on the context length, and the longer the right context contributes the performance. However, they also found that the accuracy becomes worse when the length of right context is larger than 5. One reason behind this lies that not all features included in the longer context are effective for parsing.

Therefore, it is important to utilize the non-local information without introducing noises to improve the parsing accuracy of long distance arcs.

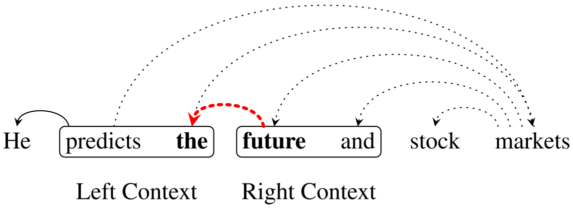


Fig. 1. An example of local decision, with the red dashed arc identifying wrong decision. "the" and "future" are the target nodes. The solid arcs are dependency relations which are already constructed. The black dotted arcs represent the gold (desired) dependency relations.

In this paper, we deal with the predicting problem of the long distance dependency in two ways. One is to utilize the non-local information by incorporating the long distance collocations, which avoids introducing extra noise information. Another is to use more reasonable actions to improve the classification accuracy. The proposed method costs up to $O(n \log(n))$ extra time complexity to find the collocation information. However, our method still runs in almost linear time in practice. The experiments show that our method leads to significant improvements in parsing accuracy, especially for long sentence.

Specifically, we make the two following contributions:

- We improve the transition-based bottom-up parsing with an approximate global view, which greatly reduces the locality of decisions. By utilizing the long distance collocations, our method provides a rough outline of the entire sentence, which is a little bit similar to top-down parsing.
- In addition to the standard parsing actions in transition-based parsing, we provide two extra fine-grained actions to avoid confusion between parsing actions and actual arcs, which can also improve the accuracy of classifier-based parsing decision.

2 Transition-Based Dependency Parsing

In a typical transition-based parsing process, an input sentence is usually processed from left to right based on shift-reduce algorithm. A set of actions are defined to construct a parse tree [10,14].

We develop our method based on Yamada's algorithm [14], which performs multi-pass scans of a partially built dependency structure. At each point, it focuses on a pair of adjacent nodes in the partial dependency structure and uses a classifier to decide whether to create a dependency link between them or to shift the focus to the next pair of heads.

There are three following deterministic actions:

Shift. No construction of dependency between the target nodes, and the position of focus simply moves to the right.

Left. A dependency relation is constructed between two neighboring nodes where the right node of target nodes becomes a child of the left one.

Right. A dependency relation is constructed between two neighboring nodes where the left node of target nodes becomes a child of the right one.

With these three actions, Yamada' parsing algorithm consists of three steps:

1. Estimate an appropriate parsing action by a classifier with features extracted from contextual information surrounding the target nodes.
2. Execute the estimated action.
3. While there is no construction in one pass, we use the action (**Left** or **Right**) with maximum score from the substitute actions.

The pseudo-code of Yamada's parsing algorithm is shown in Algorithm 1.

We use T to represent the sequence of nodes consisting of m elements $t_m (m \leq n)$, each element of which represents the root node of a sub-tree constructed in the parsing process (initially each t_i is a word w_i).

During the execution, a parsing action $y_i \in \{\textbf{Right}; \textbf{Left}; \textbf{Shift}\}$ is estimated for the focus nodes pair $\langle t_i, t_{i+1} \rangle$. The appropriate parsing action is usually decided by one or more classifiers. The contextual features x are extracted from the context surrounding the focus nodes pair.

Meanwhile, we use y'_i to record the substitute action if $y_i = \textbf{Shift}$. y'_i is the action with second largest score s'_i given by a classifier.

If all actions are **Shift** in one pass(i.e., $no\_construction = true$) we select the action y'_j at position j ($j = \arg\max_i s'_i$) and make the construction.

The complexity of the algorithm is $O(n^2)$ in the worst case because it takes up to $n - 1$ passes to construct a complete dependency tree. However, it runs in linear time in practice.

3 Incorporating Long Distance Collocations

In transition-based parsing systems, all of the decisions are very local, and the strict left-to-right order implies that, while the feature set can use rich structural information from the left of the current attachment point, it is also very restricted in information to the right context. Traditionally, only the next two or three input tokens are available to the parser. This limited look-ahead window leads to error propagation and worse performance for long distant dependencies.

```
input  : Input Sentence: $(w_1, w_2 \cdots w_n)$
output: dependency tree T and $|T| = 1$
Initialize:
$i = 1$ ;
$T = w_1, w_2 \cdots w_n$ ;
no_construction = $true$ ;
while $|T| \geq 1$ do
    if $i == |T|$ then
        if $no\_construction == true$ then
            find the substitute action $y'_j$ with largest score $j = \arg\max_i s'_i$
            construct(T, j, $y'_j$);
        end
        no_construction = true ;
        $i = 1$ ;
    end
    get the contextual features $x$;
    estimate the substitute action $y$;
    construction(T, i, y) ;
    if $y ==$ Left $or$ Right then
        no_construction = false ;
    else
        estimate the substitute action $y'_i$ and its score $s'_i$ ;
    end
end
return $T$ ;
```

Algorithm 1. Parsing Algorithm

As an example, Figure 1 shows a wrong decision due to the restriction of locality. The words “the” and “future” are assumed as the target nodes, and the length of context window is 1 (just for illustration). The decision prefers to make **Right** action if the word “markets” is unseen. But if we know that “future” and ”markets” are collocation and often occur together, a more reasonable decision should be made.

To utilize the non-local information without introducing noises, we incorporate the long distance collocation information to help the greedy parsing decision.

In this paper, we use the term *collocation* (see [6], chapter 5) in a rather loose sense, which refers to any pair of words w_1 and w_2 with dependency relation in a sentence. The occurrence of w_1 in a text makes the appearance of w_2 in the same text more likely. w_1 and w_2 do not need to be consecutive, but they should have a dependency relation.

3.1 Collocations Extraction

Firstly, we search the entire training corpus to build the collocations between words. Here, we just use the first-order dependency relation. Relations that include a preposition are “collapsed” by directly connecting the head and the child of the preposition.

Given a sentence $s = w_1, \cdots, w_n$, if two words w_i, w_j have a dependency relation, they are regarded as a candidate collocation. Thus, we can build a collocation dictionary to store all the candidate collocations by scanning the entire training corpus.

Besides the word forms, we could also build a dictionary to store the collocation (pos_i, pos_j) of part-of-speech (POS) tags for each collocation (w_i, w_j).

In order to score the textual association strength between words and obtain the reliable collocations, we calculate the (pointwise) mutual information for each pair of words which have dependency relation in our training corpus.

Mutual information is one of many measures that seems to be roughly correlated to the degree of semantic relation between words. The mutual information between two words w_1 and w_2 is given by: $I(w_1, w_2) = \frac{Pr(w_1,w_2)}{Pr(w_1)Pr(w_2)}$.

Notice that, directionality is not taken into account (i.e., both (w_l, w_2) and (w_2, w_1) are counted as occurrences of the same bigram).

3.2 Features with Collocation Information

Since the traditional features are extracted from local context, the decision lacks foresight for global information. We wish to improve the decision in each parsing step with collocation information.

Considering an intermediate step in parsing process, the sequence of nodes T consists of m elements $t_m (1 < m \leq n)$, and the focus nodes pair is $\langle t_i, t_{i+1} \rangle$.

The left context is defined as the nodes on the left side of the target nodes: $t_l (l < i)$, and the right context is defined as those on the right: $t_r (i + 1 < r)$. Context length (l, r) represents the numbers of nodes within the left and right contexts.

We find the collocation information outside the context window. For a node t_j $(1 \leq j < i - l$ or $i + r + 1 < j \leq m)$ outside the context, if (t_j, t_i) or (t_j, t_{i+1}) is in collocation dictionary, we add a collocation feature.

The collocation feature can be represented as quadruple (v_1, p_1, v_2, p_2), in which v_1 is the one of the target nodes, $p_1 \in \{L, R\}$ denotes that v_1 is left or right node, v_2 is the collocation node outside the context length, and $p_2 \in \{L, R\}$ denotes that v_2 occurs in the left or right side of v_1.

For the example in Figure 1, the collocation features include (future,R,markets,R), (the,L,markets,R) and so on.

With a discriminative classifier, these collocation features are assigned the corresponding weights according their usefulness.

3.3 Speedup

In each parsing decision, our method need find the collocation information of the target nodes. The parser will search every token in the sentence and then search for its potential relevant token, which is relative slow since the entire sentence is scanned. Fortunately, we can speedup this search process with preprocessing.

When parsing a sentence, we build an index for all collocations in the beginning. The collocation index records position of the token and the positions of its related tokens in the sentence. These two tokens must be included in the word/POS collocation dictionary.

Thus, when extracting the collocation features, our parser will easily use the collocation index to find the relevant token of target nodes as global feature, and it will not cost extra complexity.

After executing a **Left** or **Right** action, we find the position of relevant token of the modifier node, and then delete the modifier node from the collocation index.

By this preprocess, our method just need $O(nlog(n))$ extra time complexity.

4 Enhanced Actions with Fine-Grained "Shift"

Due to the bottom-up regulation, a dependency relation $h \leftarrow d$ can be constructed only after all modifiers of the d have been constructed.

For the example in Figure 2, the decision prefers to make **Left** action between two target words "able" and "handle" if the word "information" is unseen. But if we know that "handle" and "information" are collocation and often occur together, a more reasonable decision (**Shift**) should be made.

Although both the examples in Figure 1 and 2 should execute **Shift** action, they are different essentially. In Figure 1, there is no dependencies relation between the target nodes. However in Figure 1, there is dependency relation between the target nodes, but the modifier node has not been a complete subtree yet.

Therefore, **Shift** action in Yamada's algorithm should be distinguished in two cases. These two cases are more intuitively shown in Figure 3.

ABC is a word sequence in Figure 3. **Shift** action is executed for pair of (A, B) for both cases at the first pass. However, for case (b), it need predict to take **Left** action at node pair (A, B) when the arc between (B, C) is constructed. The difference between two predictions is whether C has been attached as a dependent of node B. This might cause a degradation of performance since that the prediction is often made with linear classifier. The features extracted from

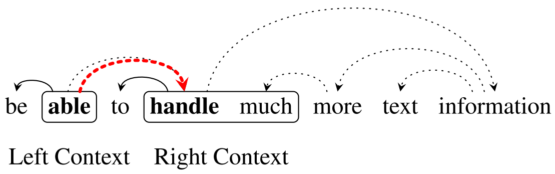


Fig. 2. Another example of wrong local decision, with the red dashed arc identifying wrong decision. "able" and "handle" are the target nodes. The solid arcs are dependency relations which are already constructed. The black dotted arcs represent the gold (desired) dependency relations.

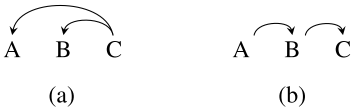


Fig. 3. Two Cases of Shift Action

this two situations have a lot of overlap. So the prediction of (A, B) is very difficult and may cause error propagation. If it predicts **Left** action on (A, B) at the first pass, the predict head of C will be never right due to the bottom-up regulation.

The main reason of the problem is that the parser ignores the potential relation between the target nodes when it predicts a **Shift** action. In order to cope with this problem, Yamada and Matsumoto [14] suggested to divide the action **Shift** into two kinds of actions, **Shift'** and **Wait** for two cases respectively. These new actions are the same behavior in parsing process. However, they do not report any experimental result for their suggestion.

In give more unambiguous definition of parsing actions, we expand the actions by adding two extra actions: **Shift-left** and **Shift-right**. The new set of parsing actions is as follows.

Left let the right node become a child of the left one.

Right let the left node become a child of the right one.

Shift let the point of focus move to the right. And there is no relation between target nodes.

Shift-right let the point of focus move to the right. And there is a right relation between target nodes but we can just shift because dependent child still have further dependent children in the sentence.

Shift-left let the point of focus move to the right. And there is a left relation between target nodes.

Compared with standard set of actions of Yamada's algorithm, we split the original **Shift** action into **Shift**, **Shift-left** and **Shift-right**, which is used to distinguish the relation between the target node pair. The **Shift-left** and **Shift-right** actions can be also called **pseudo Shift** actions.

5 Experiments

In the implementation of our proposed method, we use Margin Infused Relaxed Algorithm (MIRA) [3] to train the parameters of model. Following [2], an averaged strategy is used to avoid the overfitting problem. The training iterations of all parsers are up to 50. The baseline method in our experiments is Yamada's algorithm with MIRA as learning method instead of support vector machine.

5.1 Datasets

To empirically demonstrate the effectiveness of our approach, we use two datasets from different languages (English and Chinese) in our experimental evaluation.

English. For English, we convert the Penn WSJ Treebank (WSJ) [7] constituency trees to dependencies using Yamadas head rules. We then train on the standard PTB split with sections 2-21 as training, section 22 as validation, and section 23 as test. This test data was used in several other previous works, enabling mutual comparison with the methods reported in those works.

Chinese. For Chinese, we use the Chinese dataset in the CoNLL 2009 shared task [5].

5.2 Parsing Accuracy

Three measures (Unlabeled Attachment Score (UAS), Labeled Attachment Score (LAS) and Complete Match (CM)) are used to evaluate the parsing performances. In the evaluation, we consistently excluded punctuation marks.

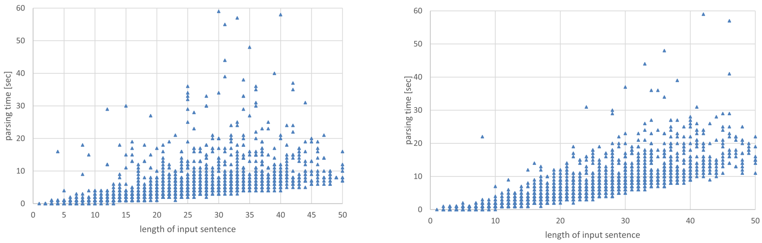
Table 1. Accuracy on English Dataset

	UAS	LAS	CM	Learning Method
Yamada & Matsumoto [14]	90.4	-	38.4	support vector machine
Nivre [11]	87.1	84.4	30.4	memory based learning
Goldberg and Elhadad [4]	89.7	-	37.5	structured perceptron
Baseline(Yamada's algorithm)	90.42	89.26	37.58	MIRA
Baseline + collocation (this work)	90.61	89.43	38.53	MIRA
Baseline + collocation + actions (this work)	**90.91**	**89.77**	**40.03**	MIRA

Table 2. Accuracy on Chinese Dataset

	UAS	LAS	CM	Learning Method
Maltparser	82.31	80.64	28.17	support vector machine
Top CoNLL 2009[12] (Transition-based)	81.22	79.19	-	support vector machine
Baseline(Yamada's algorithm)	83.18	81.51	29.54	MIRA
Baseline + collocation (this work)	83.50	81.73	30.13	MIRA
Baseline + collocation + actions (this work)	**83.95**	**82.25**	**31.18**	MIRA

The experiments results are shown in Table 1 and 2.



(a) baseline

(b) this work

Fig. 4. Parsing time for sentences.

For English dataset, our method achieves 5.1% and 4.7% error reductions in UAS and LAS respectively over baseline, and also outperforms the state-of-the-art greedy transition-based parsing methods.

For Chinese dataset, our method achieves 4.5% and 4% error reductions in UAS and LAS respectively over baseline, and also outperforms the state-of-the-art greedy transition-based parsing methods.

5.3 Parsing Time

Our utilization of the collocations also concerns the speed of our method. Here, we compare its computational time with baseline method on English WSJ dataset.

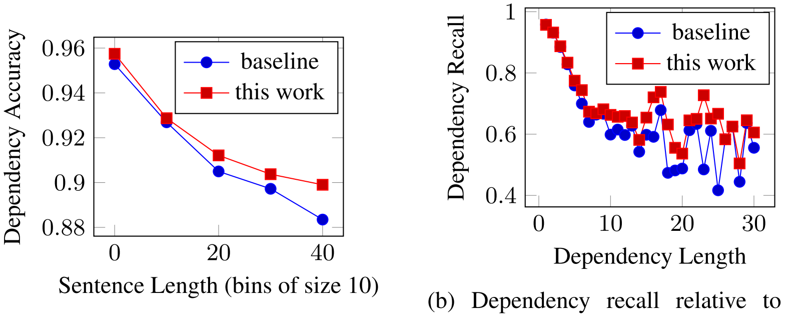
Figure 4 shows plots of the parsing times for all sentences in the test data. The average parsing time for our method was 7.1 sec, whereas that for baseline method was 5.9 sec. Although the worst-case time complexity the baseline method is $O(n^2)$, worst-case situations (e.g., all words having heads on their left) did not appear frequently. Our method costs extra $O(n \log(n))$ time to find the collocations, but it still runs in (almost) linear-time for most cases.

5.4 Result Analysis with Length Factors

It is well known that transition-based parsing systems tend to have lower accuracy for long sentence. Therefore, we evaluate the effects of our method according to different lengths of sentences and arcs. These detailed comparisons are made between our method and baseline on English WSJ dataset.

Figure 5a shows the accuracies of two methods relative to sentence length (in bins of size 10: 110, 1120, etc.). We can find that our method performs better than baseline on sentences of all lengths. Although our method is to utilize the long distance information, it is helpful for shorter sentences.

Figure 5b shows the dependency recalls of two methods relative to dependency lengths, which measures the percentage of gold standard arcs of length d that are correctly predicted. The length of a dependency from word w_i to word w_j is simply equal to $|i - j|$. We can find that our method is better for long dependency arc.



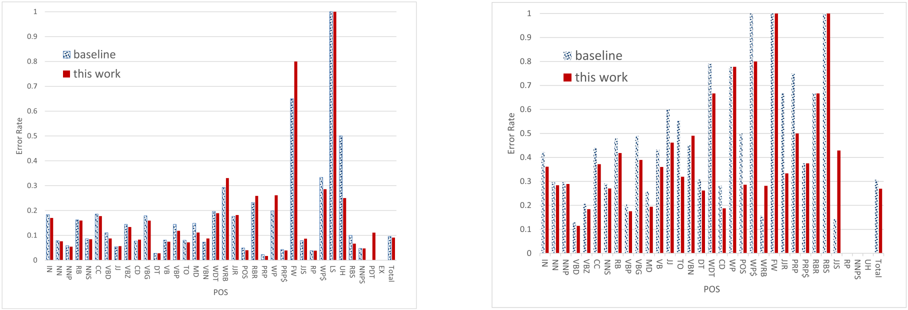
(a) Accuracy relative to sentence length

(b) Dependency recall relative to gold dependency length.

(a) Accuracy relative to sentence length

(b) Dependency recall relative to gold dependency length.

Fig. 5. Performances with different length.



(a) All arcs

(b) Long arcs (length ¿ 5)

(a) All arcs

(b) Long arcs (length ¿ 5)

Fig. 6. Error rates for different parts of speech.

These two behaviors can be explained using the same reason: the long distance collocation is useful for local decision in the greedy parsing procedure and are less tend to error propagation.

5.5 Result Analysis with Different POS Tags

We also evaluate the effects of our method according to different parts of speech. Longer dependencies typically represent modifiers of the root or the main verb in a sentence. Shorter dependencies are often modifiers of nouns such as determiners or adjectives or pronouns modifying their direct neighbours.

These detailed comparisons are made between our method and baseline on the same English WSJ dataset.

Figure 6 shows the error rates of the two models relative to the part of speech of the modifier word in a dependency relation. The error rates of Figure 6a is calculated on all arcs in test dataset, while the error rates of Figure 6b is calculated on the arcs with length larger than 5 in test dataset.

We see that our method has slightly lower error rate for most parts of speech on all arcs. While we just calculate the error rates on the dependencies with length larger than 5, our method is significantly better than baseline. This results are consistent with Figure 5b.

6 Related Works

There are several methods are proposed to avoid the drawback of local decision for the transition-based parsing.

Cheng et al. [1] try to deal with this problem by selecting the words which has not been processed as the global features. However, their method uses all words out of local context and suffers from the noise features which are brought by the unrelated words.

Goldberg and Elhadad [4] build a dependency tree with easy-first non-directional parsing strategy, which iteratively selects the best pair of neighbors to connect at each parsing step. Their methods can capture some, though not all, long-distance relations and therefore incorporate more global information than left-to-right deterministic algorithm.

Zhang and Nivre [17] utilize the richer non-local information to transition-based parsing. However, these non-local features is still in a context window and just use extra complex structures on the left context. The information outside context window is still unseen.

Zhang et al. [15] use a dependency language model (DLM) [13] to enrich the feature representations for the graph-based models [8]. However, the features based on the DLM are difficult to incorporate into the transition-based parser because DLM needs a constructed parsing tree.

7 Conclusion

In this paper, we have proposed a method to improve the accuracy of transition-based parser by utilizing long distance collocations, which achieves 5.1% and 4.5% error reductions in unlabeled attachment score over baseline on English and Chinese datasets respectively. The idea of utilizing collocation information is useful for both long and short sentences.

In future, we will extend the first-order collocation to higher-order and extract multi-word collocations, which should involve more useful information for parsing decision.

Acknowledgments. We would like to thank the anonymous reviewers for their valuable comments. This work was partially funded by the National Natural Science Foundation of China (61472088), National High Technology Research and Development Program of China (2015AA015408), Shanghai Science and Technology Development Funds (14ZR1403200).

References

1. Cheng, Y., Asahara, M., Matsumoto, Y.: Chinese deterministic dependency analyzer: examining effects of global features and root node finder. In: Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing, pp. 17–24 (2005)
2. Collins, M.: Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In: Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (2002)
3. Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., Singer, Y.: Online passive-aggressive algorithms. Journal of Machine Learning Research **7**, 551–585 (2006)
4. Goldberg, Y., Elhadad, M.: An efficient algorithm for easy-first non-directional dependency parsing. In: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, pp. 742–750 (2010)
5. Hajič, J., Ciaramita, M., Johansson, R., Kawahara, D., Martí, M., Màrquez, L., Meyers, A., Nivre, J., Padó, S., Štěpánek, J., et al.: The CoNLL-2009 shared task: syntactic and semantic dependencies in multiple languages. In: Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task, pp. 1–18 (2009)
6. Manning, C., Schütze, H.: Foundations of statistical natural language processing. MIT Press (1999)
7. Marcus, M.P., Marcinkiewicz, M.A., Santorini, B.: Building a large annotated corpus of english: The penn treebank. Computational linguistics **19**(2), 313–330 (1993)
8. McDonald, R., Crammer, K., Pereira, F.: Online large-margin training of dependency parsers. In: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, pp. 91–98 (2005)
9. McDonald, R.T., Nivre, J.: Characterizing the errors of data-driven dependency parsing models. In: EMNLP-CoNLL, pp. 122–131 (2007)
10. Nivre, J.: An efficient algorithm for projective dependency parsing. In: Proceedings of the 8th International Workshop on Parsing Technologies (IWPT) (2003)
11. Nivre, J., Scholz, M.: Deterministic dependency parsing of english text. In: Proceedings of the 20th International Conference on Computational Linguistics, p. 64 (2004)
12. Ren, H., Ji, D., Wan, J., Zhang, M.: Parsing syntactic and semantic dependencies for multiple languages with a pipeline approach. In: Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task, pp. 97–102 (2009)
13. Shen, L., Xu, J., Weischedel, R.M.: A new string-to-dependency machine translation algorithm with a target dependency language model. In: ACL, pp. 577–585 (2008)
14. Yamada, H., Matsumoto, Y.: Statistical dependency analysis with support vector machines. In: Proceedings of the International Workshop on Parsing Technologies (IWPT), vol. 3 (2003)

15. Zhang, M., Chen, W., Duan, X., Zhang, R.: Improving graph-based dependency parsing models with dependency language models. IEEE Transactions on Audio, Speech, and Language Processing **21**(11), 2313–2323 (2013)
16. Zhang, Y., Clark, S.: A tale of two parsers: investigating and combining graph-based and transition-based dependency parsing using beam-search. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 562–571 (2008)
17. Zhang, Y., Nivre, J.: Transition-based dependency parsing with rich non-local features. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, short papers, vol. 2, pp. 188–193 (2011)

Springer

http://www.springer.com/978-3-319-25206-3

Natural Language Processing and Chinese Computing
4th CCF Conference, NLPCC 2015, Nanchang, China,
October 9-13, 2015, Proceedings
Li, J.; Ji, H.; Zhao, D.; Feng, Y. (Eds.)
2015, XX, 610 p. 152 illus. in color., Softcover
ISBN: 978-3-319-25206-3