Dependency Parse Reranking Based on Subtree Extraction

Mo Shen, Daisuke Kawahara, and Sadao Kurohashi
Graduate School of Informatics，Kyoto University
Yoshida-honmachi, Sakyo-ku, Kyoto, 606-8501, Japan
shen@nlp.ist.i.kyoto-u.ac.jp {dk,kuro}@i.kyoto-u.ac.jp

Abstract

We present a new reranking approach for dependency parsing that can utilize complex subtree representation by applying efficient subtree selection heuristics. We demonstrate the effectiveness of the approach in experiments conducted on the Penn Treebank and the Chinese Treebank. Our system improves the baseline accuracy from 91.88% to 93.37% for English, and in the case of Chinese from 87.39% to 89.16%.

1. Introduction

In dependency parsing, graph-based models are prevalent for their state-of-the-art accuracy and efficiency, which are gained from their ability to combine exact inference and discriminative learning methods. The ability to perform efficient exact inference lies on the factorization technique which breaks down a parse tree into smaller substructures to perform an efficient dynamic programming search. This treatment however restricts the representation of features to a relatively small context.

A remedy approach that can explore complex feature representations for global information is called parse reranking. In its general framework, a K-best list of parse tree candidates is first produced from the base parser; a reranker is then applied to pick up the best parse among these candidates. Improvements on dependency parsing accuracy have been achieved in (Hall, 2007; Hayashi et al., 2011), however the feature sets in these studies explored a relatively small context, either by emulating the feature set in the constituent parse reranking (Charniak and Johnson, 2005), or by factorizing the search space. We think that the desirable approach for fully utilizing the power of K-best list reranking is to encode features on subtrees extracted from the candidate parse with arbitrary orders and structures, as long as the extraction process is tractable. It is still an open question how to design a process for subtree extraction that is able to selects a set of subtrees which provides reliable and concrete linguistic evidences.

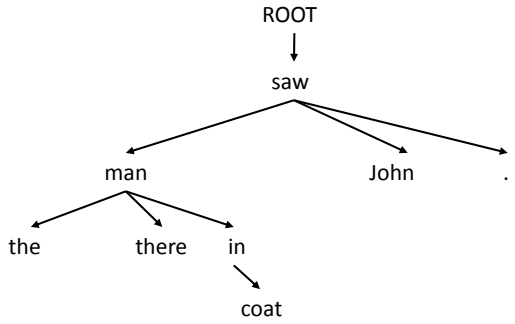

Figure 1. A dependency parse tree of the sentence "the man there in coat saw John."

In this paper, we explore a feature set that captures global information with less restriction in the structure and the size of the subtrees. It exhaustively explores a candidate parse tree for features from the most simple to the most expressive while maintaining the efficiency in the sense that it does not add additional complexities over the K-best parsing.

2. Dependency Parse Reranking

The task of dependency parsing is to find a tree structure for a sentence in which edges represent the head-modifier relationship between words: each word is linked to a unique "head" such that the link forms a semantic dependency while the main predicate of the sentence is linked to a dummy "root". An example of a dependency parse is illustrated in Figure 1.

We formally define the dependency parsing task. Given a sentence x, the best parse tree is obtained by searching for the tree with highest score:

$$\tilde{y} = \text{argmax}_{y \in \mathcal{Y}(x)} Score(y, x)$$

Where $\mathcal{Y}(x)$ is the search space of possible parse trees for x, and y is a parse tree in $\mathcal{Y}(x)$.

Parse reranking is similar with that of parsing instead of that the searching of parse tree is performed on a K-best list with selected parse

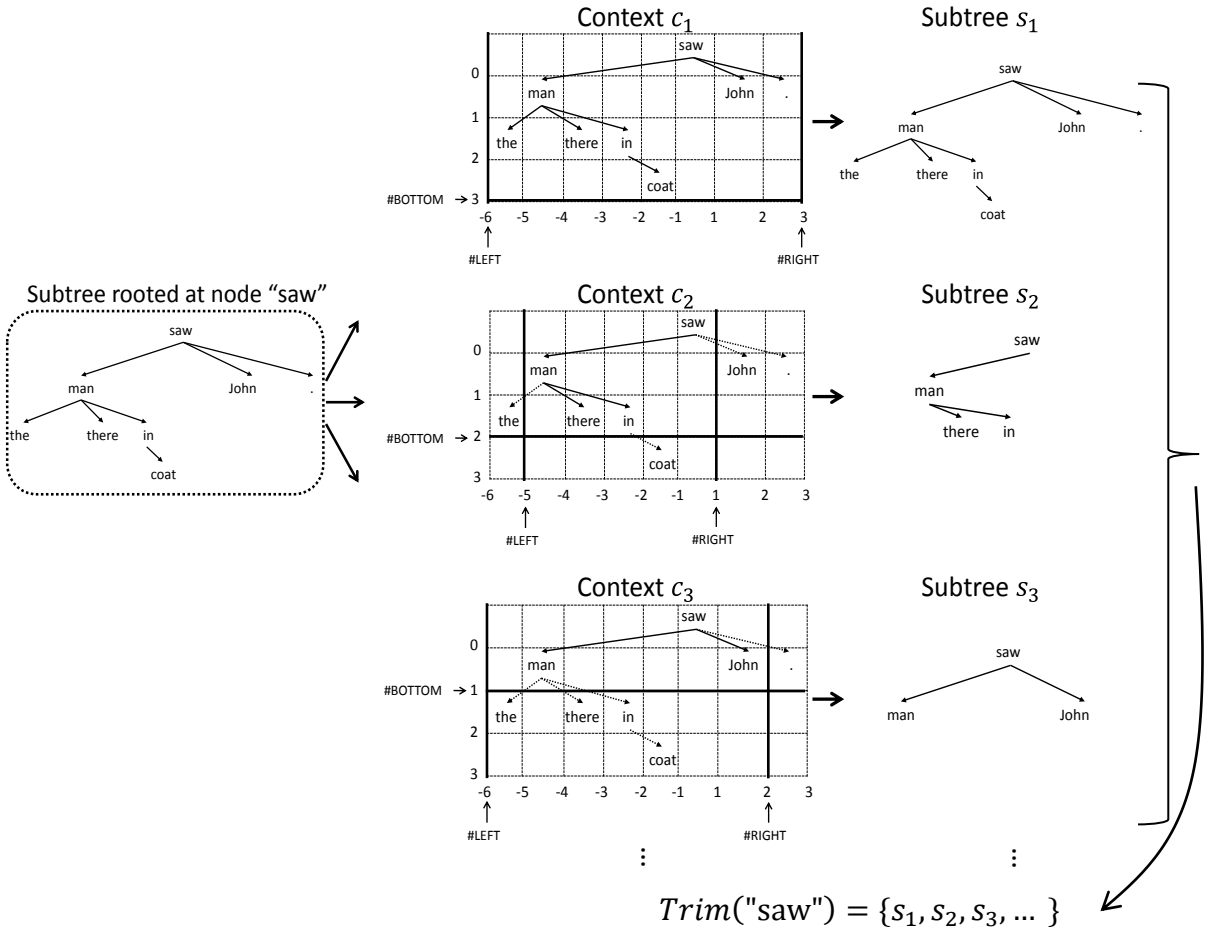

Figure 2. Extraction of trimmed subtrees from the node "saw". "#LEFT", "#RIGHT" and "#BOTTOM" represents the three boundaries that can vary along possible positions on the corresponding axis. Contexts c_1, c_2 and c_3 represnt three instances of possible combinations of boundary positions. s_1, s_2 and s_3 are resulted subtrees that are elements in the trimmed subtrees set of the node "saw".

candidates rather than the entire search space:

$$\tilde{y} = \text{argmax}_{y \in Kbest(x)} Score'(y, x)$$

$$= \text{argmax}_{y \in Kbest(x)} (L(y, x) + w \cdot f(y, x))$$

Where $L(y, x)$ is the score of y output by the base parser, and the score of the reranker is given by an inner product of a feature vector f and its corresponding weight vector w. We define the oracle parse y^+ to be the parse in the K-best list with highest accuracy compared with the gold-standard parse. The goal of reranking is to learn the weight vector so that the reranker can pick up the oracle parse as many times as possible. Note that in the reranking framework, the feature is defined on the entire parse tree which enables the encoding of global information.

2.1 Feature Sets for Reranking

We define three types of subtree structures as the reranking features below.

Trimmed subtree: For each node in a given parse tree, we check its dominated subtrees to see whether they are likely to appear in a good parse tree or not. To efficiently obtain these subtrees, we set a local window that bounds a node from its left side, right side and bottom. We then extract the maximum subtree inside this window, means that we cut off those nodes that are too distant in sequential order or too deep in a tree.

This subtree extraction process however often results in very large instances which are extremely sparse in the training data, therefore it is necessary to keep smaller subtrees as back-offs. In most cases, however, it is prohibitively expensive to enumerate all the smaller subtrees. Instead of enumeration, we design a back-off strategy that select subtrees by attempting to leave out nodes that are far away from the subtree's root and keeps those that are nearby. Precisely, after extracted the first subtree of a node, we vary the three boundaries (the left, the right and

the bottom boundary respectively) from their original positions to positions that are closer to the root of the subtree, such that it tightens up the local window. For each possible combination of the variable boundaries, we extract the largest subtree from the new local window and add it to the set of "trimmed subtrees".

This back-off strategy comes from our observation that nodes that are close to the root may provide more reliable information than those that are distant. As it is infeasible to enumerate all small subtrees as back-offs, throwing away the redundant nodes from the outer part of a large subtree is a reasonable choice.

Sibling subtree: We define the sibling subtree structures for reranking as a natural extension of the sibling factorization in (McDonald and Pereira, 2006) from the word-based case to the subtree-based case.

Specifically, for each node m in a candidate parse, its sibling subtree features is the collection of all 3-tuples:

$$\langle h, \mathrm{f}(s, p_1, i_1), \mathrm{f}(m, p_2, i_2) \rangle$$

where h represents the word form, the Part-of-Speech tag, or the combination of the word form and the Part-of-Speech tag of the head node of m; s is the nearest sibling node of m in-between h and m; and the expression $\mathrm{f}(a, p, i)$ represents the i_{th} feature encoded on a trimmed subtree in the set $Trim(a)$, such that the trimmed subtree is the one extracted within the local window p. Here an important point is that we make use of trimmed subtrees extracted in the previous phase. As mentioned before, since we keep the history of trimmed subtree extraction, it eliminates the need to re-compute any subtree structures on the sibling nodes and hence is efficient to encode.

Chain: A chain type feature encodes information for a subtree that each node has exactly one incoming edge and one outgoing edge, except on the two ends (hence a "chain"). We extract all these kind of subtrees from a parse tree in the candidates list with a parameter set to limit the number of edges in the subtree. This type of features emulates the common grandparent-grandchildren structure in dependency parsing, while we loosen the restriction on the order of the subtree. It functions as a complementary for other types of features.

3. Evaluation

We present our experimental results on two datasets, the Penn English and Chinese Treebank respectively. We convert the constituent structure in the Treebank into dependency structure with the tool Penn2Malt and the head-extraction rules being (Yamada and Matsumoto, 2003) for English and (Zhang and Clark, 2008) for Chinese. To align with previous work, for English dataset we use the standard data division: section 02-21 for training, section 24 for development, and section 23 for testing. For Chinese, we choose the Chinese Treebank 5.0 with the following data division: files 1-270 and files 400-931 for training, files 271-300 for testing, and files 301-325 for development. As our system assumes Part-of-Speech tags as input, we use MXPOST, a MaxEnt tagger (Ratnaparkhi, 1996) to automatically tag the English test data which is trained on the same training data. For Chinese we just use the gold standard Part-of-Speech tags in evaluation.

For the evaluation, we apply unlabeled attachment score (UAS) to measure the effectiveness of our method, which is the percentage of words that correctly identified their heads.

We trained a second-order sibling-factored parser emulating models in the MSTParser described in (McDonald and Pereira, 2006) by the averaged perceptron algorithm (Collins, 2002) as our baseline parser; We chose the algorithm described in (Huang and Chiang, 2005) for producing the K-best list for its efficiency; We used 30-way cross-validation on the identical training dataset to provide training data for the rerankers; and we used the following parameter setting for the feature sets throughout the experiments: K=50; the left, right and bottom boundary for the trimmed subtree features are 10, 10 and 5 respectively.

3.1 Experimental Results

We show the experimental results for both English and Chinese in Table 1. Each row in this table shows the UAS of the corresponding system. "McDonald06" stands for the second-order model in the MSTParser (McDonald and Pereira, 2006). "Yu08" stands for the probabilistic Chinese parser in (Yu et al., 2008). "Koo10" stands for the Model 1 in (Koo and Collins, 2010) which is a third-order model. "Martins10" stands for the turbo parser proposed in (Martins et al., 2010). "Baseline" is our duplicated implementation of "McDonald06" and is used as the base parsers for our reranker, denoted as "Reranked" in this table. "Baseline+Hall07" stands for the baseline parser cascading our implementation of the reranker described in (Hall, 2007), which adopted a feature set similar with the constituent

System	English UAS	Chinese UAS
McDonald06	91.5	
Yu08		87.26
Koo10	93.04	
Martins10	93.26	
Baseline	91.88	87.39
Baseline+Hall07	91.91	
Reranked	**93.37**	**89.16**
Chen09^{+}	93.16	**89.91**
Suzuki09^{+}	**93.79**	

Table 1. English and Chinese UAS of previous work, our baseline parsers, and reranked results. "$^{+}$": semi-supervised parsers.

parse reranker (Charniak and Johnson, 2005); we add this system for a comparison with our reranking approach. "Chen09" and "Suzuki09" are parsers using semi-supervised methods (Chen et al., 2009; Suzuki et al., 2009).

As we can see from the results, for English, the accuracy increased from 91.88%("Baseline") to 93.37%("Reranked") for the second-order parse reranker. This result is much higher than "Baseline+Hall07" which yielded an accuracy of 91.91% that is only slightly higher than the baseline. For Chinese, the accuracy increased from 87.39% to 89.16%. It also shows that our reranking systems obtain the highest accuracy among supervised systems. For English, our reranker even slightly outperforms "Martins10", the turbo parser which to the best of our knowledge achieved the highest accuracy in Penn Treebank. Although our rerankers are beaten by some semi-supervised systems "Suzuki09" and "Chen09", but as our reranking approach is totally orthogonal with these semi-supervising methods, it is promising to further improve the accuracy by combing these techniques.

4. Conclusion

We have proposed a novel approach for dependency parse reranking that extracts complex structures for collecting linguistic evidence, and efficient feature back-off strategy is proposed to relieve data sparsity. Experiments have demonstrated effectiveness of our method, and significant improvement over the baseline system as well as other known systems have been observed.

References

E. Charniak and M. Johnson. 2005. Coarse-to-fine N-best Parsing and MaxEnt Discriminative Reranking. In Proceedings of the 43rd ACL.

M. Collins. 2002. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In Proceedings of the 7th EMNLP, pages 1–8.

W. Chen, J. Kazama, K. Uchimoto and K. Torisawa. 2009. Improving Dependency Parsing with Subtrees from Auto-Parsed Data, In Proceedings of EMNLP2009, pages 570-579.

K. Hall. 2007. K-best Spanning Tree Parsing. In Proceedings of ACL 2007.

K. Hayashi, T. Watanabe, M. Asahara and Y. Matsumoto. 2011. Third-order Variational Reranking on Packed-Shared Dependency Forests. In Proceedings of EMNLP 2011, pages 1479-1488.

L. Huang and D. Chiang. 2005. Better K-best Parsing. In Proceedings of the IWPT, pages 53–64.

T. Koo and M. Collins. 2010. Efficient Third-order Dependency Parsers. In Proceedings of the 48th ACL, pages 1–11.

A. F. T. Martins, N. A. Smith, and E. P. Xing. 2010. Turbo Parsers: Dependency Parsing by Approximate Variational Inference. In Proceedings of EMNLP 2010, pages 34–44.

R. McDonald and F. Pereira. 2006. Online Learning of Approximate Dependency Parsing Algorithms. In Proceedings of the 11th EACL, pages 81–88.

A. Ratnaparkhi. 1996. A Maximum Entropy Model for Part-Of-Speech Tagging. In Proceedings of the 1st EMNLP, pages 133–142.

J. Suzuki, H. Isozaki, X. Carreras, and M. Collins. 2009. An Empirical Study of Semi-supervised Structured Conditional Models for Dependency Parsing. In Proceedings of EMNLP 2009, pages 551–560.

H. Yamada and Y. Matsumoto. 2003. Statistical Dependency Analysis with Support Vector Machines. In Proceedings of the IWPT 2003, pages 195–206.

K. Yu, D. Kawahara, and S. Kurohashi. 2008. Chinese Dependency Parsing with Large Scale Automatically Constructed Case Structures. In Proceedings of Coling 2008, pages 1049–1056.

Y. Zhang 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.