Proceedings of the Society for Computation in Linguistics

Volume 2 Article 31

2019

Empty Categories Help Parse the Overt

Weiwei Sun
Peking University, ws@pku.edu.cn

Empty Categories Help Parse the Overt

Weiwei Sun
Institute of Computer Science and Technology
The MOE Key Laboratory of Computational Linguistics
Center for Chinese Linguistics
Peking University
ws@pku.edu.cn

Abstract

This paper is concerned with whether deep syntactic information can help surface parsing, with a particular focus on empty categories. We consider data-driven dependency parsing with both linear and neural disambiguation models. We find that the information about empty categories is helpful to reduce the approximation error in a structured prediction based parsing model, but increases the search space for inference and accordingly the estimation error. To deal with structure-based overfitting, we propose to integrate disambiguation models with and without empty elements. Experiments on English and Chinese TreeBanks indicate that incorporating empty elements consistently improves surface parsing.

1 Introduction

In the last two decades, there was an increasing interest in producing rich syntactic annotations that are not limited to surface analysis. Such analysis, e.g. deep dependency structures (King et al., 2003), is usually coupled with grammars under deep formalisms, e.g. Combinatory Categorial Grammar, Head-driven Phrase-Structure Grammar and Lexical-Functional Grammar. Although deep grammar formalisms allow information beyond local construction to be constructed, it is still not clear whether such additional information is helpful for surface syntactic analysis. This is partly because analysis grounded on different grammar formalisms, e.g. HPSG and CFG, are not directly comparable.

In the Government and Binding (GB; Chomsky (1981)) theory, empty category is a key concept bridging S-Structure and D-Structure, due to its possible contribution to trace *movements*. Following the linguistic insights underlying GB, a traditional dependency analysis can be augmented with empty elements, viz. covert elements (Xue and Yang, 2013). The new representation provides a considerable amount of deep syntactic information, while keeping intact all dependencies of overt words. In this paper, we arguably call dependencies among overt words only surface analysis. See Figure 1 for an English example. Integrating both overt and covert elements in one unified representation provides an effective yet light-weight way to achieve deeper language understanding beyond surface syntax.

This paper studies graph-based parsing models for this new representation with a particular focus on the impact of information about the covert on parsing the overt. The major advantage of the graph-based approach to dependency parsing is that its constrained factorization enables the design of polynomial time algorithms for decoding, especially for projective structures. Following GB, an empty element can only be a dependent. Furthermore, the number and distribution of empty elements in one sentence are highly constrained. These properties make polynomial time decoding for joint empty element detection and dependency parsing still plausible. To exemplify our idea, we extend McDonald and Pereira (2006)'s second- and Koo and Collins (2010)'s third-order models for the new problem.

The influence of incorporating empty elements is twofold. On the one hand, the extra information enriches the structural information of the outputs, which is important to reduce the approximation error in a structured prediction problem. On the other hand, predicting empty elements increases the search space for decoding, and thus increases the difficulty of parameter estimation for disambiguation. To evaluate the impact of empty categories, we implement different parsing models based on global linear and neural models, and conduct experiments on English Penn TreeBank

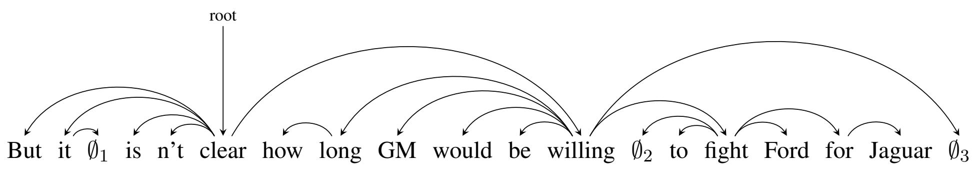
root
But it ∅1 is n't clear how long GM would be willing ∅2 to fight Ford for Jaguar ∅3

Figure 1: An example from PTB. The dependency structure is according to Stanford Dependency. "$\emptyset$" indicates an empty element. "$\emptyset_1$" indicates an expletive construction; "$\emptyset_2$" indicates that the subject for *fight*, i.e. *GM*, is located in another place; "$\emptyset_3$" indicates a *wh*-movement.

(PTB; Marcus et al. (1993)) and Chinese TreeBank (CTB; Xue et al. (2005)). Our experiment shows that the second effect is prominent. The accuracy of predicting dependencies among overt words sometimes declines slightly.

To ensure that predicting the empty elements helps parse the overt, we need to reduce the new estimation error. To this end, we propose to integrate scores from parsing models with and without empty elements and perform *joint* decoding. The intuition is to leverage parameters estimated without empty elements as a backoff, which exhibit better generalization ability. We evaluate two joint decoders: One is based on chart merging and the other is based on dual decomposition. Experiments demonstrate that information about the covert improves surface analysis in this way. Accuracy evaluated using parsers with different factorizations as well as different disambiguation models, and on data sets from different languages is consistently improved. Especially, for those sentences in which there is no empty element, accuracy is improved too. This highlights the fact that empty category can help reduce the approximation error for surface analysis.

This work has been partially published in Zhang et al. (2017) and Chen et al. (2018).

2 Parsing to Dependency Trees with and without Empty Elements

In this section, we formally introduce dependency graphs and graph-based parsing models, as well as the primary notation used in this article. A dependency graph $G = (V, A)$ is a labeled directed graph, such that for sentence $x = w_1, \dots, w_n$ the following holds:

1. $V = \{0, 1, 2, \dots, n\}$,
2. $A \subseteq V \times R \times V$.

The vertex set V consists of $n + 1$ nodes, each of which is represented by a single integer. Especially, 0 represents a virtual root node w_0, while all others corresponded to words in x. The arc set A represents the labeled dependency relations of the particular analysis G. Specifically, an arc $(i, r, j) \in A$ represents a dependency relation r from head w_i to dependent w_j. A dependency graph G is thus a set of labeled dependency relations between the root and the words of x. To simplify the description in this section, we mainly consider unlabeled parsing and assume the relation set R is a singleton. Or taking it another way, we assume $A \subseteq V \times V$.

Considering implementation, we denote the *index set* of all possible dependencies as $\mathcal{I} = \{(i, j) | i, j \in \{1, \cdots, n\}, i \neq j\}$. A dependency parse then can be represented as a vector

$$\boldsymbol{y} = \{y(i, j) : (i, j) \in \mathcal{I}\}$$

where $y(i, j) = 1$ if there is an arc (i, j) in the graph, 0 otherwise. For a sentence x, we define dependency parsing as a search for the highest-scoring analysis of x:

$$\boldsymbol{y}^*(x) = \arg \max_{\boldsymbol{y} \in \mathcal{Y}(x)} \textsc{Score}(x, \boldsymbol{y})$$

Here, $\mathcal{Y}(x)$ is the set of all trees compatible with x and $\textsc{Score}(x, \boldsymbol{y})$ evaluates the event that tree $\boldsymbol{y}$ is the analysis of sentence x. In brief, given a sentence x, we compute its parse $\boldsymbol{y}^*(x)$ by searching for the highest-scored analysis in the set of compatible trees $\mathcal{Y}(x)$; scores are assigned by $\textsc{Score}$.

In general, performing a direct maximization over the set $\mathcal{Y}(x)$ is infeasible, and a common solution used in many parsing approaches is to introduce a part-wise factorization:

$$\textsc{Score}(x, \boldsymbol{y}) = \sum_{p \in \textsc{Part}(\boldsymbol{y})} \textsc{ScorePart}(x, p)$$

Above, we have assumed that the dependency parse $\boldsymbol{y}$ can be factored into a set of parts p, each of which represents a small substructure of $\boldsymbol{y}$; for example, $\boldsymbol{y}$ might be factored into the set of its component dependencies. The parts are evaluated using a *local* scoring function SCOREPART. The factorization thus establishes implicit independence restrictions between parts, which can be exploited to efficiently solve the combinatorial optimization problem involved in the search for the highest-scoring dependency analysis.

Now consider dependency parsing with empty category detection. We also formulate the issue as an optimization problem. Assume that we are given a sentence x with n normal words. We use an index set $\mathcal{I}_o = \{(i,j)|i,j \in \{1,\cdots,n\}\}$ to denote all possible overt dependency edges, and use $\mathcal{I}_c = \{(i,\phi_j)|i,j \in \{1,\cdots,n\}\}$ to denote all possible covert dependency edges. ϕ_j denotes an empty node that precede the jth word. Then a dependency parse with empty nodes can be represented as a vector:

$$\boldsymbol{z} = \{z(i,j) : (i,j) \in \mathcal{I}_o \cup \mathcal{I}_c\}.$$

Let $\mathcal{Z}$ denote the set of all possible $\boldsymbol{z}$, and PART$(\boldsymbol{z})$ denote the factors in the dependency tree, including edges (and edge siblings in the second-order model). Then parsing with ECD can be defined as a search for the highest-scored $\boldsymbol{z}^*(x)$ in all compatible analyses, just like parsing without empty elements:

$$\begin{aligned}\boldsymbol{z}^*(x) &= \arg\max_{\boldsymbol{z}\in\mathcal{Z}(x)} \text{SCORE}(x,\boldsymbol{z}) \\ &= \arg\max_{\boldsymbol{z}\in\mathcal{Z}(x)} \sum_{p\in\text{PART}(\boldsymbol{z})} \text{SCOREPART}(x,p)\end{aligned}$$

The choice of factorization involves a tradeoff between complexity and expressiveness, and thus design of factorizations that are both expressive and efficiently parsable is of critical importance for practical parsing applications. A number of dynamic programming (DP) algorithms have been designed for first- (Eisner, 1996), second- (McDonald and Pereira, 2006; Carreras, 2007) and third- (Koo and Collins, 2010) factorization. In Zhang et al. (2017), we introduced the basic design of the above algorithms and their extensions to include detection of empty categories.

3 Experiments

We conduct experiments on both English and Chinese treebanks. In particular, PTB and CTB are used. Because PTB and CTB are phrase-structure treebanks, we need to convert them into dependency annotations. To do so, we use the tool provided by Stanford CoreNLP to process PTB, and the tool provided by Xue and Yang (2013) to process CTB 5.0. We use gold-standard POS to derive features for disambiguation. We use standard training, validation, and test splits to facilitate comparisons. Accuracy is measured with unlabeled attachment score for all overt words (UAS$_o$): the percentage of overt words with the correct head. We are also concerned with the prediction accuracy for empty elements. To evaluate performance on empty nodes, we consider the correctness of empty edges. We report the percentage of empty words in right slot with correct head. The i-th slot in the sentence means that the position immediately after the i-th concrete word. So if we have a sentence with length n, we get $n+1$ slots.

Table 1: UAS$_o$ of different individual models on test data. The upper and bottom blocks present results obtained by sibling and tri-sibling models respectively.

Algo	English	Chinese
1	91.73	89.16
3	91.70 (−0.03)	89.20 (+0.04)
4	91.72 (−0.01)	89.28 (+0.12)
2	92.23	90.00
5	92.41 (+0.18)	89.82 (−0.18)

Table 1 lists the accuracy of individual models coupled with different decoding algorithms on the test sets. We focus on the prediction for overt words only. Models coupled with Algorithm 1, 3 and 4 are second-order models, while with 2 and 5 third-order ones. When we take into account empty categories, more information is available. The empirical results suggest that deep linguistic information does not necessarily help surface analysis.

We can see from the definition of the extended algorithms that the search space for decoding is significantly increased. This results in a side effect for practical parsing. Given the limit of available annotations for training, searching for more complicated structures in a larger space is harmful to the generalization ability in structured prediction (Sun, 2014). Incorporating empty elements significantly increases the difficulty for parameter estimation, and therefore it is harder to find a good disambiguation model. To control structure-based overfitting, we propose to combine the two score functions learned from models with and without

Table 2: UAS$_o$ of different joint decoding models on test data. "CM" and "DD" are short for joint decoders based on chart merging and dual decomposition respectively. The upper and bottom blocks present results obtained by sibling and tri-sibling models respectively. All improvements are statistically significant.

	Algo	English	Chinese
CM	1+3	91.94 (+0.21)	89.53 (+0.37)
	1+4	91.88 (+0.15)	89.44 (+0.28)
DD	1+3	91.96 (+0.23)	89.53 (+0.37)
	1+4	91.94 (+0.21)	89.53 (+0.37)
CM	2+5	92.60 (+0.37)	90.35 (+0.35)
DD	2+5	92.71 (+0.48)	90.38 (+0.38)

empty elements.

We evaluate two joint decoders: chart merging and dual decomposition. Table 2 lists the accuracy of different joint decoding models on the test sets. We can see that the joint decoding framework is effective to deal with structure-based overfitting. This time, the accuracy of analysis for overt words is consistently improved across a wide range of conditions. Especially, the third-order model is improved more. We use the Hypothesis Tests method (Berg-Kirkpatrick et al., 2012) to evaluate the improvements. When the *p-value* is set to 0.05, all improvements in Figure 2 is statistically significant.

Table 3: UAS$_o$ of different neural parsing models on test data. Results are obtained by sibling models. The joint decoder is CM.

Algo	English	Chinese
1	93.59	90.70
3	93.49	90.45
1+3	93.96	92.00

We also evaluate the impact of empty categories as well as joint decoding on neural dependency parsing. Table 3 summarizes the results. Due to the complexity of third-order parsing model as well as its marginal improvement, we only report results obtained by second-order parsing. Without structure regularization, i.e. joint decoding in our case, incorporating information about empty category cannot improve surface parsing. In fact, the parsing accuracy decreases slightly. When the model for parsing without empty elements is incorporated, empty category always improves surface parsing. The improvement is statistically significant.

References

Taylor Berg-Kirkpatrick, David Burkett, and Dan Klein. 2012. An empirical investigation of statistical significance in NLP. In *Proceedings of the 2012 Joint Conference on EMNLP and CoNLL*.

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *In Proc. EMNLP-CoNLL*.

Yufei Chen, Yuanyuan Zhao, Weiwei Sun, and Xiaojun Wan. 2018. Pre- and in-parsing models for neural empty category detection. In *Proceedings of the ACL*. http://aclweb.org/anthology/P18-1250.

Noam Chomsky. 1981. *Lectures on Government and Binding*. Foris Publications, Dordecht.

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: an exploration. In *Proceedings of the 16th conference on Computational linguistics - Volume 1*. pages 340–345.

Tracy Holloway King, Richard Crouch, Stefan Riezler, Mary Dalrymple, and Ronald M. Kaplan. 2003. The PARC 700 dependency bank. In *In Proceedings of the 4th International Workshop on Linguistically Interpreted Corpora (LINC-03)*. pages 1–8.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of ACL*. Association for Computational Linguistics, Uppsala, Sweden, pages 1–11.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: the Penn treebank. *Comput. Linguist.* 19(2):313–330.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL-2006*. volume 6, pages 81–88.

Xu Sun. 2014. Structure regularization for structured prediction. In *Advances in Neural Information Processing Systems 27*, pages 2402–2410.

Naiwen Xue, Fei Xia, Fu-dong Chiou, and Marta Palmer. 2005. The Penn Chinese treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering* 11:207–238.

Nianwen Xue and Yaqin Yang. 2013. Dependency-based empty category detection via phrase structure trees. In *Proceedings of NAACL-2013*. Atlanta, Georgia, pages 1051–1060.

Xun Zhang, Weiwei Sun, and Xiaojun Wan. 2017. The covert helps parse the overt. In *Proceedings of the CoNLL*. pages 343–353. http://aclweb.org/anthology/K17-1035.