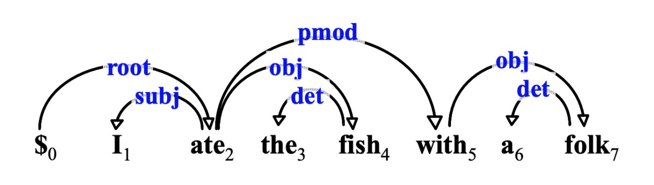
Headed-Span-Based Projective Dependency Parsing

Songlin Yang, Kewei Tu

April 20, 2022

SIST, ShanghaiTech University

Projective dependency parsing



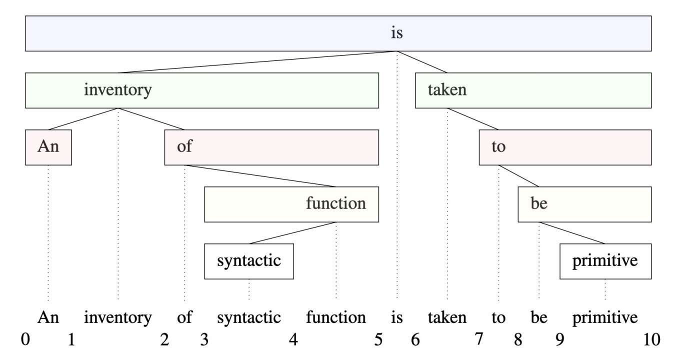
- Dependency parsing aims to produce an acyclic and connected tree where each node has exactly one head.
- There is no crossing arcs in a projective dependency parse tree.
  - More than 99% trees in PTB are projective.
  - Pseudo-projective transformation (Nivre and Nilsson, 2005) can be applied to transform nonprojective trees into projective trees.

Main paradigms

- Graph-based parsers: assign a score to every possible tree and globally find the highest-scoring tree.
  - ☺: Allow global training and decoding. High parsing accuracy.
  - ☹: Cannot capture sufficient subtree information, especially for first-order parsers.
- Transition-based parsers: read the sentence sequentially and conduct a series of local decisions to build the final parse.
  - ☺: Previously parsed subtree information can be exploited. Faster parsing speed.
  - ☹: Most of them cannot perform global optimization. Past decision mistakes lead to error propagation.

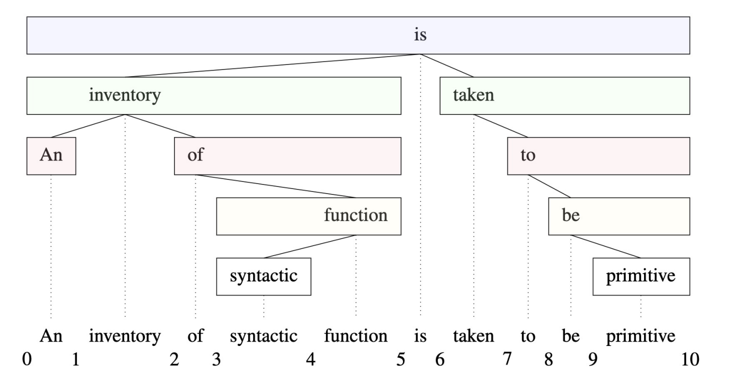
We want to exploit more subtree information, meanwhile maintaining the ability of global optimization.

Headed span



- In a projective parse tree, the whole subtree rooted at a headword forms a contiguous sequence (i.e., span) in the surface order.
- We call such a span-headword pair as **headed span**.
- A projective tree is comprised of a collection of headed spans.

Example



{(0, 1, An), (0, 5, inventory), (2, 5, of), (3, 5, function), (3, 4, syntactic), (0, 10, is), (6, 10, taken), (7, 10, to), (8, 10, be), (9, 10, primitive) }

- Each word corresponds to exactly one headed span.
- One can traverse a gold tree to obtain all headed spans in $O(n)$ time.

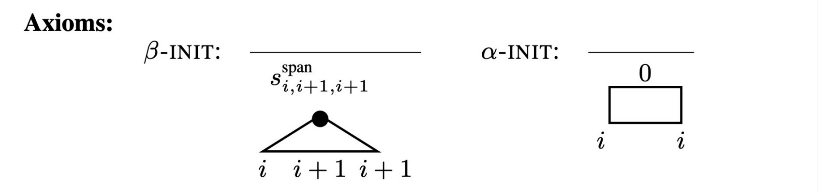
Headed-span-based parsing

- Inspired by span-based constituency parsing, we decompose the score of a dependency tree into the sum of headed span scores:

$$s(y) = \sum_{i=1,\ldots,n} s^{\text{span}}_{l_i, r_i, i}$$

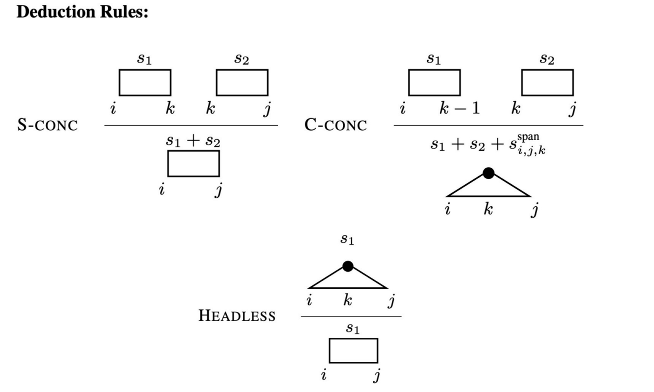
  - ☺: Headed spans contain **more subtree information**, as a headed span corresponds to the largest subtree rooted by the headword.
  - ☺: **Rich span representations** can be exploited.
  - ☺: **Global optimization** can be performed. We design a dynamic programming algorithm to parse in cubic time.

Parsing



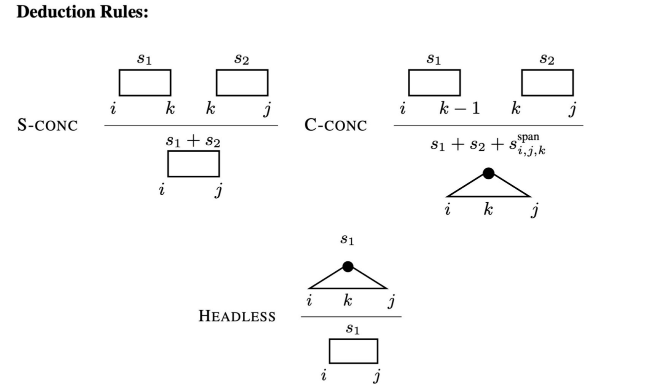
- $\alpha_{i,j}$: the accumulated score of span (i,j) serving as a left or right child span
- $\beta_{i,j,k}$: the accumulated score of the headed span (i,j,k).

Parsing



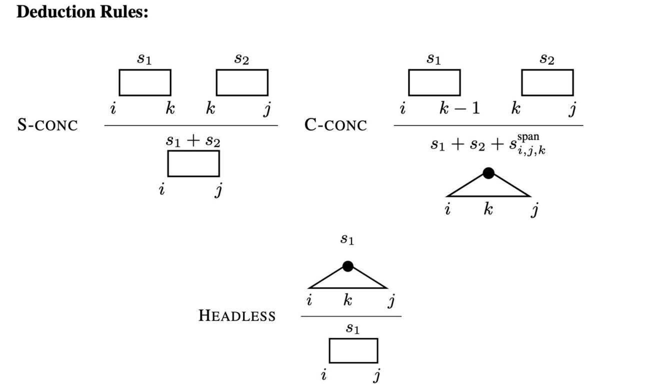
- S-CONC: concatenate two consecutive child spans into a single child span

Parsing



- C-CONC: concatenate left and right child span $(i, k-1)$ and (k, j) along with the root word-span $(k-1, k)$ to form a headed span (i, j, k).

Parsing

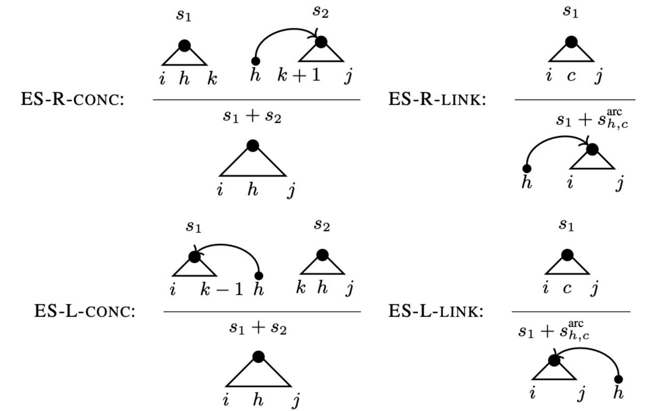


- HEADLESS: obtain a headless child span from a headed span.

Backtracking

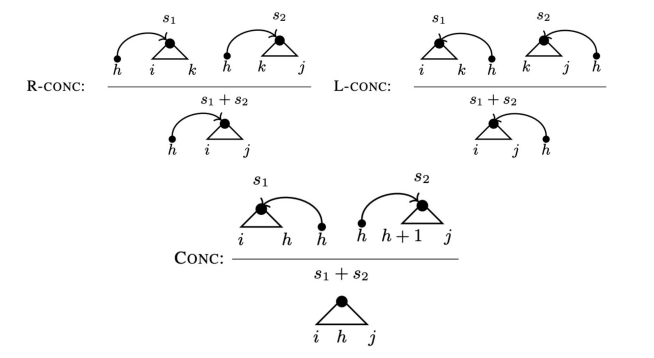
1. Find the root: $\text{argmax}_r\, \beta_{0,n,r}$
2. For a given headed span (i, j, h), **find the best segmentation** regarding child spans, similar to the inference procedure of semi-Markov CRF.
3. For each child span (a, b) within the best segmentation, find its headword: $c = \text{argmax}_r\, \beta_{a,b,r}$ and **add an arc from h to c.**
4. Repeat step 2, 3.

Our algorithm ≈ hook trick + head-splitting trick



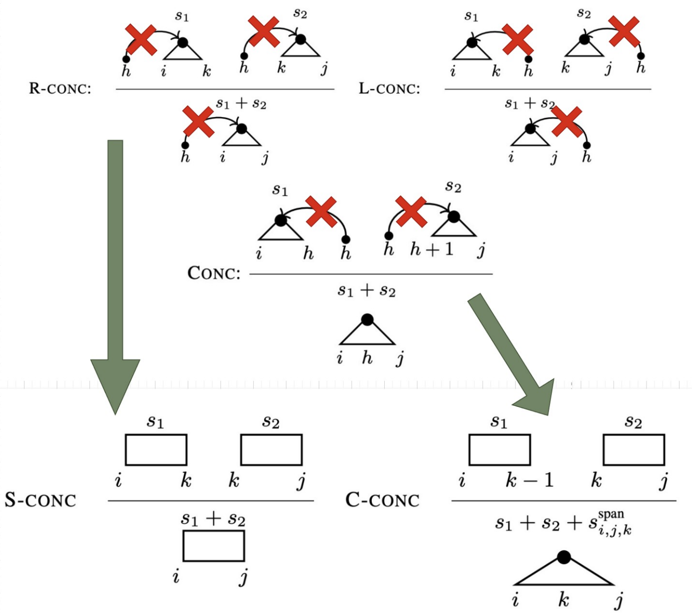
- Deductive rules of the $O(n^4)$ **Eisner-Satta algorithm** (Eisner and Satta, 1999).
- **The hook trick** reduces subtrees into headless spans as the linking of heads and the concatenation of subtrees can be seperated.

Our algorithm ≈ hook trick + head-splitting trick



- The **head-splitting trick** splits each subtree into a left and a right fragment, which is the key of the $O(n^3)$ **Eisner algorithm**.
- We can transform the deductive rules of the **Eisner-Satta** algorihm to R-CONC, L-CONC, and CONC using the **head-splitting trick**.

Our algorithm ≈ hook trick + head-splitting trick



Experiments on PTB and CTB

	PTB		CTB	
	UAS	LAS	UAS	LAS
MFVI2O	95.98	94.34	90.81	89.57
TreeCRF2O	96.14	94.49	-	-
HierPtr	96.18	94.59	90.76	89.67
	+BERT$_{\text{base}}$		+BERT$_{\text{base}}$	
RNGTr	96.66	95.01	92.98	91.18
	+BERT$_{\text{large}}$		+BERT$_{\text{base}}$	
MFVI2O	96.91	95.34	92.55	91.69
HierPtr	97.01	95.48	92.65	91.47
Biaffine+MM†	97.22	95.71	93.18	92.10
Ours	**97.24**	**95.73**	**93.33**	**92.30**
For reference				
	+XLNet$_{\text{large}}$		+BERT$_{\text{base}}$	
HPSG$^{\flat}$	97.20	95.72	-	-
HPSG+LAL$^{\flat}$	97.42	96.26	94.56	89.28

Table 1: Results for different model on PTB and CTB.

- Our model achieves the state-of-the-art performance.

Experiments on UD

	bg	ca	cs	de	en	es	fr	it	nl	no	ro	ru	Avg
TreeCRF2O	90.77	91.29	91.54	80.46	87.32	90.86	87.96	91.91	88.62	91.02	86.90	93.33	89.33
MFVI2O	90.53	92.83	92.12	81.73	89.72	92.07	88.53	92.78	90.19	91.88	85.88	92.67	90.07
						$+$BERT$_{\text{multilingual}}$							
MFVI2O	**91.30**	93.60	92.09	82.00	90.75	92.62	89.32	93.66	91.21	91.74	86.40	92.61	90.61
Biaffine+MM†	90.30	**94.49**	**92.65**	**85.98**	91.13	93.78	**91.77**	94.72	91.04	94.21	87.24	**94.53**	91.82
Ours	91.10	94.46	92.57	85.87	**91.32**	**93.84**	91.69	**94.78**	**91.65**	**94.28**	**87.48**	94.45	**91.96**

Table 2: Labeled Attachment Score (LAS) on twelve languages in UD 2.2.

Questions?