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.
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 $(i,j,k)$.

**Axioms:**

$\beta$-INIT: $s_{i,i+1,i+1}^{\text{span}}$

$\alpha$-INIT: $0$

\[\begin{array}{ccc}
i & i+1 & i+1 \\
\end{array}\]
**S-conc**: concatenate two consecutive child spans into a single child span
• **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 \((i, j, k)\).
• **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 \(\approx\) 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 separated.
Our algorithm \( \approx \) 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 algorithm to R-CONC, L-CONC, and CONC using the head-splitting trick.
Our algorithm $\approx$ hook trick + head-splitting trick
### Experiments on PTB and CTB

<table>
<thead>
<tr>
<th>Model</th>
<th>PTB UAS</th>
<th>PTB LAS</th>
<th>CTB UAS</th>
<th>CTB LAS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MFVI2O</strong></td>
<td>95.98</td>
<td>94.34</td>
<td>90.81</td>
<td>89.57</td>
</tr>
<tr>
<td><strong>TreeCRF2O</strong></td>
<td>96.14</td>
<td>94.49</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>HierPtr</strong></td>
<td>96.18</td>
<td>94.59</td>
<td>90.76</td>
<td>89.67</td>
</tr>
<tr>
<td>+BERT&lt;sub&gt;base&lt;/sub&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>RNGTr</strong></td>
<td>96.66</td>
<td>95.01</td>
<td>92.98</td>
<td>91.18</td>
</tr>
<tr>
<td>+BERT&lt;sub&gt;large&lt;/sub&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>MFVI2O</strong></td>
<td>96.91</td>
<td>95.34</td>
<td>92.55</td>
<td>91.69</td>
</tr>
<tr>
<td><strong>HierPtr</strong></td>
<td>97.01</td>
<td>95.48</td>
<td>92.65</td>
<td>91.47</td>
</tr>
<tr>
<td><strong>Biaffine+MM</strong></td>
<td>97.22</td>
<td>95.71</td>
<td>93.18</td>
<td>92.10</td>
</tr>
<tr>
<td><strong>Ours</strong></td>
<td><strong>97.24</strong></td>
<td><strong>95.73</strong></td>
<td><strong>93.33</strong></td>
<td><strong>92.30</strong></td>
</tr>
<tr>
<td>+BERT&lt;sub&gt;base&lt;/sub&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For reference

<table>
<thead>
<tr>
<th>Model</th>
<th>PTB UAS</th>
<th>PTB LAS</th>
<th>CTB UAS</th>
<th>CTB LAS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HPSG&lt;sup&gt;b&lt;/sup&gt;</strong></td>
<td>97.20</td>
<td>95.72</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>HPSG+LAL&lt;sup&gt;b&lt;/sup&gt;</strong></td>
<td>97.42</td>
<td>96.26</td>
<td>94.56</td>
<td>89.28</td>
</tr>
</tbody>
</table>

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

- Our model achieves the state-of-the-art performance.
## Experiments on UD

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