10. Unsupervised Parsing

Gerald Penn
Department of Computer Science, University of Toronto
Unsupervised parsing

• Parsing without training on parse trees. How could such a thing be learned?
• Well, unsupervised doesn’t always mean no supervision…
  • Parts of speech
  • Binary-branch restrictions
• …and we often lower the bar in terms of what we expect the system to learn:
  • Unlabelled (binary) trees
  • Hierarchical structure without explicit, recursive rules.
PRPN trains a sequence of components that build a parse tree on the way to predicting the next word in a string of words – a fancy language model.

But that means that supervising the whole system in sequence means that we must only provide words in sequence…

• for a parser, that counts as unsupervised!

When we are done, we can break off the later components and use the parser by itself.
Some terminology

• “Corner-dominant”
  • The highest ancestor for which a node is the left corner, e.g.:

```
node

corner-dominant
```

```
corner-dominant’s parent

node’s parent
```

```
S
```

```
NP
```

```
RC
```

```
VP
```

```
NP
```

```
NP
```

```
NP
```

```
NP
```

```
NP
```

```
NP
```

```
NP
```

```
NP
```

```
NP
```

```
NP
```

```
NP
```

```
NP
```

```
NP
```

```
NP
```

```
NP
```

```
NP
```

```
NP
```

```
NP
```

```
NP
```

```
NP
```

```
NP
```

```
NP
```

```
NP
```

```
NP
```

```
NP
```

```
NP
```

```
NP
```

```
NP
```

```
NP
```

```
NP
```

```
NP
```

```
NP
```

```
NP
```

```
NP
```

```
NP
```

```
NP
```

```
NP
```
Some terminology

- “Corner-dominant”
- The highest ancestor for which a node is the left corner, e.g.:
Some terminology

• “Left extent”
• \( l_t = \) the left corner of a pre-terminal node’s corner-dominant’s parent, for \( t > 0 \), e.g.:
Some terminology

- “Left extent”
  - $l_t = \text{the left corner of a pre-terminal node’s corner-dominant’s parent, for } t > 0, \text{ e.g.}:

```
Some terminology

S
  NP  
    RC
      NP
        NP
          PN  N
          Buffalo  buffalo
        NP
          PN  N
          Buffalo  buffalo
        NP
          NP
            V
              V
                PN  N
                Buffalo  buffalo
```
Some terminology

- “Dependency-range gate”
  - \( g_i^t = \begin{cases} 1, & l_t \leq i < t \\ 0, & 0 \leq i < l_t \end{cases} \), labels left extent of \( x_t \), e.g.:

\[ g^2 = \begin{pmatrix} 1 & 1 & - & - & - & - & - & - & - \end{pmatrix} \]
Some terminology

- “Dependency-range gate”
- \[ g_i^t = \begin{cases} 1, & l_t \leq i < t \\ 0, & 0 \leq i < l_t \end{cases} \]

labels left extent of \( x_t \), e.g.:

\[ g^4: 0 \quad 0 \quad 1 \quad 1 \quad -- \quad -- \quad -- \quad -- \quad -- \]
Some terminology

• “Height”
  • \( h(w) = 1 \),
  • \( h(n) = \max_{m \in T_n \setminus n} h(m) + 1 \).

Note: height is not depth, nor is it \( h(\text{root}) \)-depth. Count from the bottom.
Some terminology

- “Roark-Hollingshead distance”
- \( d(i) = d_i = \frac{h(w_{i-1}, w_i) - 2}{h(r) - 1} \).

where \( h(w_{-1}, w_0) = h(w_{L-1}, w_L) = h(r) + 1 \),
\( h(u, v) = h(u \sqcup v) \) everywhere else (trees are CNF).

\[ d(0) = \frac{6 + 1 - 2}{6 - 1} = 1 \]
\[ d(2) = \frac{5 - 2}{6 - 1} = \frac{3}{5} \]
\[ d(4) = \frac{4 - 2}{6 - 1} = \frac{2}{5} \]
Q: How much of this does this preserve?

A: All of it (except labels)!

Very cool, because this is a “local” statistic.
Some terminology

- “Dependency range”
- \( \alpha_i^t = \frac{\text{sign}(d_t - d_{i+1}) + 1}{2} \), where \( i < t \).
PRPN’s big idea

\[ g_i^t = P(l_t \leq i) \approx \prod_{j=i+1}^{t-1} \alpha_j^t. \]
PRPN’s big idea

\[ g_i^t = P(l_t \leq i) \approx \prod_{j=i+1}^{t-1} \alpha_j^t. \]
PRPN’s big idea

\[ g_i^t = P(l_t \leq i) \approx \prod_{j=i+1}^{t-1} \alpha_j^t. \]

\[ x_4 \cup x_5 \rightarrow S \]

\[ \alpha^5: 1 \ 1 \ 1 \ 1 \ \frac{1}{2} \ -- \ -- \ -- \]

\[ g^5: 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ -- \ -- \ -- \]
PRPN’s big idea

- But if \( P(l_t \leq i) \approx \prod_{j=i+1}^{t-1} \alpha_j^t \), then:

\[
P(l_t \leq i) \approx \prod_{j=i+2}^{t-1} \alpha_j^t \cdot \alpha_{i+1}^t \approx P(l_t \leq i + 1) \cdot \alpha_{i+1}^t,
\]

so:

\[
\alpha_{i+1}^t \approx P(l_t \neq i + 1 \mid l_t \leq i + 1)
\]

\[
1 - \alpha_i^t \approx P(l_t = i \mid l_t \leq i), \text{ and:}
\]

\[
P(l_t = i) = P(l_t = i \mid l_t \leq i) \cdot P(l_t \leq i)
\]

\[
\approx (1 - \alpha_i^t) \cdot \prod_{j=i+1}^{t-1} \alpha_j^t.
\]

- This is an example of the well-known **stick-breaking process**. When \( \alpha_j = 1 - \beta_j \), and the \( \beta_j \) are samples from beta distributions, this process is an instance of a Dirichlet process.
PRPN’s big idea

• But if $P(l_t \leq i) \approx \prod_{j=i+1}^{t-1} \alpha_j^t$, then:

$$P(l_t \leq i) \approx \prod_{j=i+2}^{t-1} \alpha_j^t \cdot \alpha_{i+1}^t \approx P(l_t \leq i + 1) \cdot \alpha_{i+1}^t,$$

so:

$$\alpha_{i+1}^t \approx P(l_t \neq i + 1 | l_t \leq i + 1)$$

$$1 - \alpha_i^t \approx P(l_t = i | l_t \leq i),$$

and:

$$P(l_t = i) = P(l_t = i | l_t \leq i) \cdot \prod_{j=i+1}^{t-1} \alpha_j^t.$$

• This is very well suited to modelling the dependence of $l_t$ upon as many words/pre-terminals as we want.
• Soften up “Dependency range:”
  \[ \alpha_i^t = \frac{\text{sign}(d_t - d_{i+1}) + 1}{2}, \]  where \( i < t \), becomes:
  \[ \alpha_i^t = \frac{\text{hardtanh}((d_t - d_{i+1}) \cdot \tau)}{2}, \]  where \( \tau \) is temperature,
  
  and \( \text{hardtanh}(x) = \max(-1, \min(1, x)) \).

• Then learn RH distance with a 2-layer convolution:
  \[ q_t = \text{ReLU} \left( W_q \begin{bmatrix} e_{t-L} \\ \vdots \\ e_{t-L+1} \\ e_t \end{bmatrix} + b_q \right), \]

  \[ d_t = \text{ReLU}(W_d q_t + b_d). \]

• But we’re not going to supervise this with \( d_t \) from actual trees…
Instead, we couple the input to memory states $m_i$ and use RH distance to interpolate mixtures of previous time steps into “summary vectors” that predict subsequent memory states:

- $k_t = W_mm_{t-1} + W_e e_t,$
- $\bar{s}_i^t = \text{softmax} \left( \frac{m_i k_t^T}{\sqrt{\text{dim}(k)}} \right),$
- $s_i^t = \frac{g_i^t}{\sum_j g_j^t} s_i^t,$
- $[\overline{m}_t] = \sum_{i=1}^{t-1} s_i^t \cdot [m_i],$
- $\overline{c}_t = \sigma_{i=1}^{t-1} s_i^t \cdot [c_i],$
- $\overline{m}_t = \sigma_{i=1}^{t-1} s_i^t \cdot [m_i].$
PRPN: predict

- Instead, we now predict $e_{t+1}$, given $m_0, \ldots, m_t$, which in turn depend upon $e_0, \ldots, e_t$:
  - $k_t = W_mm_{t-1} + W_e e_t$,
  - $s^t_i = \text{softmax} \left( \frac{m_i k^T_t}{\sqrt{\text{dim}(k)}} \right)$,
  - $r^t_i = \frac{g_{t+1}^i}{\sum_j g_{t+1}^j} s^t_i$, Depends on $d_{t+1}$
  - $l_t = \sum_{i=l_{t+1}} r^t_i \cdot m_i$, Stick-breaking process: also depends on $d_{t+1}$
  - Estimate $d_{t+1} \approx \text{ReLU} (\tilde{W}_d m_t + \tilde{b}_d)$,
  - then estimate $\tilde{e}_{t+1} = \tanh (W_f \left[ l_t \right] + b_f)$.
CCM: brackets and spans

- Predict syntax directly, but not with trees.
- Instead, use bracket matrices and “spans,” which here consist also of yields and contexts:
CCM: brackets and spans

- Predict syntax directly, but not with trees.
- Instead, use bracket matrices and “spans,” which here consist also of yields and contexts.

\[ P(S, B) = P(B)P(S|B) \]
\[ P(S|B) = \prod_{\langle i,j \rangle} P(\alpha_{ij}|B_{ij})P(x_{ij}|B_{ij}) \]

- Then, use Expectation Maximization:
  - E-step: calculate \( P(B|S, \Theta) \)
  - M-step: fixing those, calculate:
    \[ \arg\max_{\Theta} \sum_{\hat{B}} P(B|S, \Theta) \log P(S, B|\hat{\Theta}). \]
- \( P(B) \) is not recalculated – it is a uniform distribution over tree-consistent bracketings.
# Performance on WSJ10

<table>
<thead>
<tr>
<th>Model</th>
<th>$\text{UF}_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>LBRANCH</td>
<td>28.7</td>
</tr>
<tr>
<td>RANDOM</td>
<td>34.7</td>
</tr>
<tr>
<td>DEP-PCFG (Carroll &amp; Charniak, 1992)</td>
<td>48.2</td>
</tr>
<tr>
<td>RBRANCH</td>
<td>61.7</td>
</tr>
<tr>
<td>CCM (Klein &amp; Manning, 2002)</td>
<td>71.9</td>
</tr>
<tr>
<td>DMV+CCM (Klein &amp; Manning, 2005)</td>
<td>77.6</td>
</tr>
<tr>
<td>UML-DOP (Bod, 2006)</td>
<td><strong>82.9</strong></td>
</tr>
<tr>
<td>PRPN</td>
<td>70.02</td>
</tr>
<tr>
<td>UPPER BOUND</td>
<td><strong>88.1</strong></td>
</tr>
</tbody>
</table>
## Performance on PTB30+

<table>
<thead>
<tr>
<th>Model</th>
<th>PTB</th>
<th>CTB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>Max</td>
</tr>
<tr>
<td>PRPN (Shen et al., 2018)</td>
<td>37.4</td>
<td>38.1</td>
</tr>
<tr>
<td>ON (Shen et al., 2019)</td>
<td>47.7</td>
<td>49.4</td>
</tr>
<tr>
<td>URNNG† (Kim et al., 2019)</td>
<td></td>
<td>45.4</td>
</tr>
<tr>
<td>DIORA† (Drozdov et al., 2019)</td>
<td></td>
<td>58.9</td>
</tr>
<tr>
<td>Left Branching</td>
<td>8.7</td>
<td></td>
</tr>
<tr>
<td>Right Branching</td>
<td>39.5</td>
<td></td>
</tr>
<tr>
<td>Random Trees</td>
<td>19.2</td>
<td>19.5</td>
</tr>
<tr>
<td>PRPN (tuned)</td>
<td>47.3</td>
<td>47.9</td>
</tr>
<tr>
<td>ON (tuned)</td>
<td>48.1</td>
<td>50.0</td>
</tr>
<tr>
<td>Scalar PCFG</td>
<td>&lt; 35.0</td>
<td></td>
</tr>
<tr>
<td>Neural PCFG</td>
<td>50.8</td>
<td>52.6</td>
</tr>
<tr>
<td>Compound PCFG</td>
<td>55.2</td>
<td>60.1</td>
</tr>
<tr>
<td>Oracle Trees</td>
<td>84.3</td>
<td></td>
</tr>
</tbody>
</table>