Combinatory Categorial Grammar (CCG)

- A lexicalized grammar formalism
Combinatory Categorial Grammar (CCG)

\[
\text{the books which John likes}
\]

\[
\begin{align*}
&\text{NP} / \text{N} \\
&\text{N} \\
\Rightarrow
&\text{NP} \\
\end{align*}
\]

\[
\begin{align*}
&\text{NP} / \text{NP} \\
&\text{NP} \\
\Rightarrow
&(\text{NP} / \text{NP}) / (\text{S} / \text{NP}) \\
\end{align*}
\]

\[
\begin{align*}
&\text{S} / (\text{S} / \text{NP}) \\
\Rightarrow
&\text{S} / \text{NP} \\
\Rightarrow
&\text{B} \\
\end{align*}
\]

\[
\begin{align*}
&\text{NP} \backslash \text{NP} \\
\Rightarrow
&\text{NP} \\
\end{align*}
\]
Combinatory Categorial Grammar (CCG)

- A lexicalized grammar formalism
- Is able to derive typed dependency structures
Combinatory Categorial Grammar (CCG)

\[
\begin{align*}
\text{the} & \quad \text{books} & \quad \text{which} & \quad \text{John} & \quad \text{likes} \\
NP/N & \quad N & \quad (NP\backslash NP)/(S/NP) & \quad NP & \quad (S\backslash NP)/NP \\
\quad & \quad & \quad & S/(S\backslash NP)^T & \quad \Rightarrow_B \\
\quad & \quad & \quad & S/NP & \quad \Rightarrow \\
\quad & \quad & \quad & NP\backslash NP & \quad \Rightarrow < \\
\quad & \quad & \quad & NP & \quad <
\end{align*}
\]
Combinatory Categorial Grammar (CCG)

\[
\begin{array}{c}
\text{the} \quad \text{books} \\
\overrightarrow{\text{NP}/\text{N}} \quad \overrightarrow{\text{N}} \\
\overrightarrow{\text{NP}} \\
\overrightarrow{\text{S}/(\text{NP})} \\
\overrightarrow{\text{S}/\text{NP}} \\
\end{array}
\quad \quad \quad
\begin{array}{c}
\text{which} \\
\overrightarrow{(\text{NP}/\text{NP})/(\text{S}/\text{NP})} \\
\overrightarrow{(\text{S}/\text{NP})/\text{NP}} \\
\overrightarrow{\text{NP}/\text{NP}} \\
\overrightarrow{\text{NP}} \\
\end{array}
\quad \quad \quad
\begin{array}{c}
\text{John} \\
\overrightarrow{\text{NP}} \\
\overrightarrow{\text{S}/(\text{NP})} \\
\overrightarrow{\text{S}/\text{NP}} \\
\end{array}
\quad \quad \quad
\begin{array}{c}
\text{likes} \\
\overrightarrow{(\text{S}/\text{NP})/\text{NP}} \\
\overrightarrow{\text{NP}/\text{NP}} \\
\overrightarrow{\text{NP}} \\
\end{array}
\quad \quad \quad
\begin{array}{c}
\langle \text{the, } \text{NP}/\text{N}_1, 1, \text{books,} \rangle
\end{array}
\]
Combinatory Categorial Grammar (CCG)

\[
\langle \text{the}, NP/N_1, 1, \text{books}, \rangle
\]
\[
\langle \text{likes}, (S\setminus NP_1)/NP_2, 1, \text{John} \rangle
\]
Combinatory Categorial Grammar (CCG)

\[
\begin{align*}
\text{NP} / N & \quad \text{N} \\
\text{NP} & \quad \frac{\text{NP} \setminus \text{NP}}{(S/\text{NP})} \\
\text{S} / (S/\text{NP}) & \quad \frac{\text{NP}}{\text{NP} \setminus \text{NP}} \quad \frac{(S/\text{NP})/\text{NP}}{\text{NP}} \\
\langle \text{the}, \text{NP} / N_1, 1, \text{books}, \rangle & \\
\langle \text{likes}, (S/\text{NP}_1)/\text{NP}_2, 1, \text{John} \rangle & \\
\langle \text{which}, (\text{NP} / \text{NP}_1)/(S/\text{NP})_2, 2, \text{likes} \rangle
\end{align*}
\]
Combinatory Categorial Grammar (CCG)

\[
\begin{align*}
\text{the} & \quad \text{books} \\
NP/N & \quad N & & (NP\backslash NP)/(S/NP) \\
\quad & & NP & (S\backslash NP)/NP \\
\quad & & S/(S\backslash NP) \quad \Rightarrow_T \quad S/NP \\
\quad & & NP\backslash NP & \Rightarrow \quad NP
\end{align*}
\]

\[
\langle \text{the}, \, NP/N_1, \, 1, \, \text{books}, \rangle \\
\langle \text{likes}, \, (S\backslash NP_1)/NP_2, \, 1, \, \text{John} \rangle \\
\langle \text{which}, \, (NP/NP_1)/(S/NP)_2, \, 2, \, \text{likes} \rangle \\
\langle \text{which}, \, (NP/NP_1)/(S/NP)_2, \, 1, \, \text{books} \rangle \\
\langle \text{likes}, \, (S\backslash NP_1)/NP_2, \, 2, \, \text{books} \rangle
\]
Combinatory Categorial Grammar (CCG)

- A lexicalized grammar formalism
- Is able to derive typed dependency structures

[Rimell et al., 2009; Nivre et al., 2010]
- Exhibits "spurious" ambiguity (through the use of type-raising) in order to recover certain dependencies
Combinatory Categorial Grammar (CCG)

- A lexicalized grammar formalism
- Is able to derive typed dependency structures
- Remains to be the most competitive formalism for recovering “deep” dependencies (from coordination, control, extraction etc.)
  [Rimell et al., 2009; Nivre et al., 2010]
Combinatory Categorial Grammar (CCG)

- A lexicalized grammar formalism

- Is able to derive typed dependency structures

- Remains to be the most competitive formalism for recovering “deep” dependencies (from coordination, control, extraction etc.) [Rimell et al., 2009; Nivre et al., 2010]

- Exhibits “spurious” ambiguity (through the use of type-raising) in order to recover certain dependencies
Combinatroy Categorial Grammar (CCG)

He reads the book

\[
\begin{array}{c}
NP \\
S/(S\backslash NP) \quad (S\backslash NP)/NP \quad NP/N \quad N \\
\quad NP \\
S/\backslash NP \quad (S\backslash NP)/N \quad \quad NP \quad N \\
\quad NP \\
S/\backslash NP \quad (S\backslash NP)/N \quad \quad NP \quad N \\
\quad NP \\
S/\backslash NP \quad (S\backslash NP)/N \quad \quad NP \quad N \\
\quad NP \\
S
\end{array}
\]

\[
\begin{array}{c}
NP \\
S/(S\backslash NP) \quad (S\backslash NP)/NP \quad NP/N \quad N \\
\quad NP \\
S/\backslash NP \quad (S\backslash NP)/N \quad \quad NP \quad N \\
\quad NP \\
S/\backslash NP \quad (S\backslash NP)/N \quad \quad NP \quad N \\
\quad NP \\
S/\backslash NP \quad (S\backslash NP)/N \quad \quad NP \quad N \\
\quad NP \\
S
\end{array}
\]

⟨the, NP/N₁, 1, book, −⟩
⟨reads, (S\\backslash NP₁)/NP₂, 2, book, −⟩
⟨reads, (S\\backslash NP₁)/NP₂, 1, he, −⟩

In general, exponentially many!
Motivation: Dependency Model

- Should we model the derivations or dependencies?
  - the standard choice: the normal-form model [Hockenmaier, 2003; Clark and Curran 2007]

- The derivation is just a “trace” of the semantic interpretation [Steedman, 2000]
Motivation: Dependency Model

- Should we model the derivations or dependencies?
  - the standard choice: the normal-form model [Hockenmaier, 2003; Clark and Curran 2007]

- The derivation is just a “trace” of the semantic interpretation [Steedman, 2000]

<table>
<thead>
<tr>
<th>[Clark et al., 2002]</th>
<th>C&amp;C (dep)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dep. Model</td>
<td>✓</td>
</tr>
<tr>
<td>Deriv. Feats</td>
<td>✗</td>
</tr>
</tbody>
</table>
Motivation: Dependency Model

• Should we model the derivations or dependencies?
  – the standard choice: the normal-form model [Hockenmaier, 2003; Clark and Curran 2007]

• The derivation is just a “trace” of the semantic interpretation [Steedman, 2000]

<table>
<thead>
<tr>
<th></th>
<th>[Clark et al., 2002]</th>
<th>C&amp;C (dep)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dep. Model</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Deriv. Feats</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>

– an elegant solution to the spurious ambiguity problem
– gold-standard data cheaper to obtain
– optimizing for evaluation
Motivation: Dependency Model

⇒ Open: Shift-Reduce CCG parsing with a Dependency Model

• Shift-Reduce fits with the incremental nature of CCG
  – linear vs. $\mathcal{O}(n^5)$ decoding
  – can include arbitrary features

<table>
<thead>
<tr>
<th></th>
<th>[Clark et al., 2002]</th>
<th>C&amp;C (dep)</th>
<th>Z&amp;C</th>
<th>this work</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shift-Reduce</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Dep. Model</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
</tr>
<tr>
<td>Deriv. Feats</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Shift-Reduce CCG Parsing

- **Input**: pos- and super-tagged words
- **Shift**: next lexical category from the queue
- **Reduce**: the top two categories
- **Unary**: type-raising or type-changing the top category
## Shift-Reduce CCG Parsing

<table>
<thead>
<tr>
<th>step</th>
<th>stack ((s_n, \ldots, s_1, s_0))</th>
<th>queue ((q_0, q_1 \ldots, q_n))</th>
<th>action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>Ms. Haag plays Elianti</td>
<td></td>
</tr>
</tbody>
</table>
## Shift-Reduce CCG Parsing

<table>
<thead>
<tr>
<th>step</th>
<th>stack ( (s_n, \ldots, s_1, s_0) )</th>
<th>queue ( (q_0, q_1 \ldots, q_n) )</th>
<th>action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>Ms. Haag plays Elianti</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>( N/N )</td>
<td>Haag plays Elianti</td>
<td>SHIFT</td>
</tr>
</tbody>
</table>
# Shift-Reduce CCG Parsing

<table>
<thead>
<tr>
<th>step</th>
<th>stack ( (s_n, \ldots, s_1, s_0) )</th>
<th>queue ( (q_0, q_1 \ldots, q_n) )</th>
<th>action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>Ms. Haag plays Elianti</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>( N/N )</td>
<td>Haag plays Elianti</td>
<td>SHIFT</td>
</tr>
<tr>
<td>2</td>
<td>( N/N \ N )</td>
<td>plays Elianti</td>
<td>SHIFT</td>
</tr>
</tbody>
</table>
### Shift-Reduce CCG Parsing

<table>
<thead>
<tr>
<th>step</th>
<th>stack ((s_n, \ldots, s_1, s_0))</th>
<th>queue ((q_0, q_1 \ldots, q_n))</th>
<th>action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>Ms. Haag plays Elianti</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>(N/N)</td>
<td>Haag plays Elianti</td>
<td>SHIFT</td>
</tr>
<tr>
<td>2</td>
<td>(N/N) (N)</td>
<td>plays Elianti</td>
<td>SHIFT</td>
</tr>
<tr>
<td>3</td>
<td>(N)</td>
<td>plays Elianti</td>
<td>REDUCE</td>
</tr>
</tbody>
</table>
### Shift-Reduce CCG Parsing

<table>
<thead>
<tr>
<th>step</th>
<th>stack ((s_n, \ldots, s_1, s_0))</th>
<th>queue ((q_0, q_1 \ldots, q_n))</th>
<th>action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>Ms. Haag plays Elianti</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>(N/N)</td>
<td>Haag plays Elianti</td>
<td>SHIFT</td>
</tr>
<tr>
<td>2</td>
<td>(N/N\ N)</td>
<td>plays Elianti</td>
<td>SHIFT</td>
</tr>
<tr>
<td>3</td>
<td>(N)</td>
<td>plays Elianti</td>
<td>REDUCE</td>
</tr>
<tr>
<td>4</td>
<td>(NP)</td>
<td>plays Elianti</td>
<td>UNARY</td>
</tr>
</tbody>
</table>
# Shift-Reduce CCG Parsing

<table>
<thead>
<tr>
<th>step</th>
<th>stack ((s_n, \ldots, s_1, s_0))</th>
<th>queue ((q_0, q_1, \ldots, q_n))</th>
<th>action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>Ms. Haag plays Elianti</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>(N/N)</td>
<td>Haag plays Elianti</td>
<td>SHIFT</td>
</tr>
<tr>
<td>2</td>
<td>(N/N) (N)</td>
<td>plays Elianti</td>
<td>SHIFT</td>
</tr>
<tr>
<td>3</td>
<td>(N)</td>
<td>plays Elianti</td>
<td>REDUCE</td>
</tr>
<tr>
<td>4</td>
<td>(NP)</td>
<td>plays Elianti</td>
<td>UNARY</td>
</tr>
<tr>
<td>5</td>
<td>(NP \ (S[dcl]\ NP)/NP)</td>
<td>Elianti</td>
<td>SHIFT</td>
</tr>
</tbody>
</table>
## Shift-Reduce CCG Parsing

<table>
<thead>
<tr>
<th>step</th>
<th>stack ((s_n, \ldots, s_1, s_0))</th>
<th>queue ((q_0, q_1 \ldots, q_n))</th>
<th>action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>Ms. Haag plays Elianti</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>(N/N)</td>
<td>Haag plays Elianti</td>
<td>SHIFT</td>
</tr>
<tr>
<td>2</td>
<td>(N/N \ N)</td>
<td>plays Elianti</td>
<td>SHIFT</td>
</tr>
<tr>
<td>3</td>
<td>(N)</td>
<td>plays Elianti</td>
<td>REDUCE</td>
</tr>
<tr>
<td>4</td>
<td>(NP)</td>
<td>plays Elianti</td>
<td>UNARY</td>
</tr>
<tr>
<td>5</td>
<td>(NP \ (S[dcl])\backslash NP)/NP)</td>
<td>Elianti</td>
<td>SHIFT</td>
</tr>
<tr>
<td>6</td>
<td>(NP \ (S[dcl])\backslash NP)/NP \ N)</td>
<td></td>
<td>SHIFT</td>
</tr>
</tbody>
</table>
## Shift-Reduce CCG Parsing

<table>
<thead>
<tr>
<th>step</th>
<th>stack ((s_n, \ldots, s_1, s_0))</th>
<th>queue ((q_0, q_1 \ldots, q_n))</th>
<th>action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>Ms. Haag plays Elianti</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>(N/N)</td>
<td>Haag plays Elianti</td>
<td>SHIFT</td>
</tr>
<tr>
<td>2</td>
<td>(N/N\ N)</td>
<td>plays Elianti</td>
<td>SHIFT</td>
</tr>
<tr>
<td>3</td>
<td>(N)</td>
<td>plays Elianti</td>
<td>REDUCE</td>
</tr>
<tr>
<td>4</td>
<td>(NP)</td>
<td>plays Elianti</td>
<td>UNARY</td>
</tr>
<tr>
<td>5</td>
<td>(NP \ (S[dcl]\ NP) / NP)</td>
<td>Elianti</td>
<td>SHIFT</td>
</tr>
<tr>
<td>6</td>
<td>(NP \ (S[dcl]\ NP) / NP N)</td>
<td></td>
<td>SHIFT</td>
</tr>
<tr>
<td>7</td>
<td>(NP \ (S[dcl]\ NP) / NP NP)</td>
<td></td>
<td>UNARY</td>
</tr>
</tbody>
</table>
## Shift-Reduce CCG Parsing

<table>
<thead>
<tr>
<th>step</th>
<th>stack ((s_n, \ldots, s_1, s_0))</th>
<th>queue ((q_0, q_1 \ldots, q_n))</th>
<th>action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>Ms. Haag plays Elianti</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>(N/N)</td>
<td>Haag plays Elianti</td>
<td>SHIFT</td>
</tr>
<tr>
<td>2</td>
<td>(N/N\ \ N)</td>
<td>plays Elianti</td>
<td>SHIFT</td>
</tr>
<tr>
<td>3</td>
<td>(N)</td>
<td>plays Elianti</td>
<td>REDUCE</td>
</tr>
<tr>
<td>4</td>
<td>(NP)</td>
<td>plays Elianti</td>
<td>UNARY</td>
</tr>
<tr>
<td>5</td>
<td>(NP (S[dcl]\ NP)/NP)</td>
<td>Elianti</td>
<td>SHIFT</td>
</tr>
<tr>
<td>6</td>
<td>(NP (S[dcl]\ NP)/NP \ N)</td>
<td></td>
<td>SHIFT</td>
</tr>
<tr>
<td>7</td>
<td>(NP (S[dcl]\ NP)/NP \ NP)</td>
<td></td>
<td>UNARY</td>
</tr>
<tr>
<td>8</td>
<td>(NP S[dcl]\ NP)</td>
<td></td>
<td>REDUCE</td>
</tr>
</tbody>
</table>
Shift-Reduce CCG Parsing

<table>
<thead>
<tr>
<th>step</th>
<th>stack ((s_n, \ldots, s_1, s_0))</th>
<th>queue ((q_0, q_1 \ldots, q_n))</th>
<th>action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>Ms. Haag plays Elianti</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>(N/N)</td>
<td>Haag plays Elianti</td>
<td>SHIFT</td>
</tr>
<tr>
<td>2</td>
<td>(N/N) (N)</td>
<td>plays Elianti</td>
<td>SHIFT</td>
</tr>
<tr>
<td>3</td>
<td>(N)</td>
<td>plays Elianti</td>
<td>REDUCE</td>
</tr>
<tr>
<td>4</td>
<td>(NP)</td>
<td>plays Elianti</td>
<td>UNARY</td>
</tr>
<tr>
<td>5</td>
<td>(NP (S[dcl] \langle NP\rangle) / NP)</td>
<td>Elianti</td>
<td>SHIFT</td>
</tr>
<tr>
<td>6</td>
<td>(NP (S[dcl] \langle NP\rangle) / NP \langle N\rangle)</td>
<td></td>
<td>SHIFT</td>
</tr>
<tr>
<td>7</td>
<td>(NP (S[dcl] \langle NP\rangle) / NP \langle NP\rangle)</td>
<td></td>
<td>UNARY</td>
</tr>
<tr>
<td>8</td>
<td>(NP S[dcl] \langle NP\rangle)</td>
<td></td>
<td>REDUCE</td>
</tr>
<tr>
<td>9</td>
<td>(S[dcl])</td>
<td></td>
<td>REDUCE</td>
</tr>
</tbody>
</table>
Beam-Search Decoding

- Score of an item $\langle s, q \rangle \circ x = w \cdot \phi(\langle s, q \rangle, x)$
Beam-Search Decoding

- Score of an item $\langle s, q \rangle \circ x = w \cdot \phi(\langle s, q \rangle, x)$
### The Dependency Model

<table>
<thead>
<tr>
<th></th>
<th>[Clark et al., 2002]</th>
<th>C&amp;C (dep)</th>
<th>Z&amp;C</th>
<th>this work</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shift-Reduce</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Dep. Model</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Deriv. Feats</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

- The normal-form model [Zhang and Clark, 2011]
  - one unique gold derivation per sentence
The Dependency Model

<table>
<thead>
<tr>
<th></th>
<th>[Clark et al., 2002]</th>
<th>C&amp;C (dep)</th>
<th>Z&amp;C</th>
<th>this work</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shift-Reduce</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Dep. Model</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Deriv. Feats</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

- The normal-form model [Zhang and Clark, 2011]
  - one unique gold derivation per sentence

- The dependency model
  - given only gold dependency structures and exponentially many “correct” derivations hidden
The Dependency Model

<table>
<thead>
<tr>
<th></th>
<th>[Clark et al., 2002]</th>
<th>C&amp;C (dep)</th>
<th>Z&amp;C</th>
<th>this work</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shift-Reduce</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Dep. Model</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Deriv. Feats</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

- The normal-form model [Zhang and Clark, 2011]
  - one unique gold derivation per sentence
- The dependency model
  - given only gold dependency structures and exponentially many “correct” derivations hidden
  - a method to query such an oracle
  - trained using the “early-update” variant of the violation-fixing perceptron [Huang et al., 2012]
  - similar to [Goldberg and Nivre 2012], our oracle to determine valid transition sequences
CCG Parse Forest

- We follow the definitions in [Clark and Curran, 2007; Miyao and Tsujji, 2002]
- Compactly represents all derivation and dependency structure pair
- Grouping together equivalent chart entries
  - identical category, head and unfilled dependencies
  - individual entries are conjunctive nodes and equivalence classes are disjunctive nodes
We follow the definitions in [Clark and Curran, 2007; Miyao and Tsujji, 2002]

- Compactly represents all derivation and dependency structure pair
- Grouping together equivalent chart entries
  - identical category, head and unfilled dependencies
  - individual entries are conjunctive nodes and equivalence classes are disjunctive nodes
The Oracle Forest

• A **subset** of the **complete** forest
  – consistent with the gold-standard dependency structure
  – exponentially-sized and impossible to enumerate

• A dependency structure decomposes over derivations
  – dependencies are realized on conjunctive nodes
  – can count dependencies on-the-fly
The Oracle Forest

• intuition 1: dependencies “live on” conjunctive nodes
The Oracle Forest

- Intuition 1: Dependencies “live on” conjunctive nodes

\[
\langle \text{some, } NP/N_1, 1, \text{ books} \rangle
\]
The Oracle Forest

- intuition 1: dependencies “live on” conjunctive nodes

\[
\langle \text{some}, \, \text{NP}/N_1, \, 1, \, \text{books} \rangle
\]

\[
\langle \text{bought}, \, (\text{S}/\text{NP}_1)/\text{NP}_2, \, 2, \, \text{books} \rangle
\]
The Oracle Forest

- intution 1: dependencies “live on” conjunctive nodes

\[
\langle \text{some}, \text{NP}/N_1, 1, \text{books} \rangle \\
\langle \text{bought}, (S\backslash NP_1)/NP_2, 2, \text{books} \rangle \\
\langle \text{he}, (S\backslash NP_1)/NP_2, 1, \text{bought} \rangle \\
\]
The Oracle Forest

- intuition 1: dependencies “live on” conjunctive nodes

\begin{align*}
\langle \textit{he}, (S\backslash NP_1)/NP_2, 1, \textit{bought} \rangle \\
\langle \textit{bought}, (S\backslash NP_1)/NP_2, 2, \textit{books} \rangle \\
\langle \textit{some}, NP/N_1, 1, \textit{books} \rangle
\end{align*}
- Intuition 1: Dependencies “live on” conjunctive nodes.
- Intuition 2: A conjunctive node that has less than the maximum possible number of gold-standard dependencies is not gold (optimal substructure).

\[
\langle \text{some, } NP/N_1, 1, \text{ books} \rangle
\]
\[
\langle \text{bought, } (S\backslash NP_1)/NP_2, 2, \text{ books} \rangle
\]
\[
\langle \text{he, } (S\backslash NP_1)/NP_2, 1, \text{ bought} \rangle
\]
The Oracle Forest

- Intuition 1: Dependencies “live on” conjunctive nodes
- Intuition 2: A conj. node that has less than the max possible number of gold-standard dependencies is not gold (optimal substructure)

\[
\langle \text{he}, (S \setminus NP_1)/NP_2, 1, \text{bought} \rangle
\]

\[
\langle \text{bought}, (S \setminus NP_1)/NP_2, 2, \text{books} \rangle
\]

\[
\langle \text{some}, NP/N_1, 1, \text{books} \rangle
\]
The Oracle Forest

• intuition 1: dependencies “live on” conjunctive nodes
• intuition 2: a conj. node that has less than the max possible number of gold-standard dependencies is not gold (optimal substructure)

⟨some, NP/N₁, 1, books⟩
⟨bought, (S \ NP₁)/NP₂, 2, books⟩
⟨he, (S \ NP₁)/NP₂, 1, bought⟩
The Dependency Model Oracle

- The dependency oracle

\[
f_d(\langle s, q \rangle, (x, c), \Phi_G) = \begin{cases} 
  \text{true} & \text{if } s' \sim G \text{ or } s' \simeq G \\
  \text{false} & \text{otherwise}
\end{cases}
\]
The Dependency Model Oracle

Canonical Shift-Reduce resembles bottom-up post-order traversal

```
S[dcl]
  NP
    N
      N/N
      Mr.
  (S[dcl]\NP)/NP
    visited
      N
      Paris
  S[dcl]\NP
    NP
      N
      President
```
The Dependency Model Oracle

Canonical Shift-Reduce resembles bottom-up post-order traversal

```
S[dcl]
  NP
    N
      N/N
      Mr. President
  (S[dcl] \ NP) / NP
    visited
          N
            Paris
```
Canonical Shift-Reduce resembles bottom-up post-order traversal

Shift Shift
The Dependency Model Oracle

Canonical Shift-Reduce resembles bottom-up post-order traversal

```
S[dcl]
  NP
    N
      N/N
      Mr. President
    N
      N/N
      President
  S[dcl]\NP
    (S[dcl]\NP)/NP
      visited
      N
      Paris
  NP
```

Shift Shift Reduce
Canonical Shift-Reduce resembles bottom-up post-order traversal

\[
\begin{array}{c}
\text{S[dcl]} \\
\text{NP} \\
\text{N} \\
\text{N/N} \\
\text{Mr.} \\
\text{N/N} \\
\text{N} \\
\text{President} \\
\text{S[dcl]\NP} \\
\text{(S[dcl]\NP)/NP} \\
\text{visited} \\
\text{NP} \\
\text{N} \\
\text{Paris}
\end{array}
\]
The Dependency Model Oracle

Canonical Shift-Reduce resembles bottom-up post-order traversal

```
S[dcl]
  /
NP
  /
  N
  /
N/N N
  Mr. President
```

```
S[dcl]\NP
  /
(S[dcl]\NP)/NP
  /
visited

NP
  /
N
Paris
```

Shift Shift Reduce Unary Shift
Canonical Shift-Reduce resembles bottom-up post-order traversal

```
S[dcl]
   NP
      N
          N/N
          Mr.
          President
       (S[dcl]\NP)/NP
         visited
         N
         Paris
S[dcl]\NP
   NP
```

Shift Shift Reduce Unary Shift Shift Unary Reduce Reduce
The Dependency Model Oracle

But this doesn’t carry over to an oracle forest

He some books bought some books
The Dependency Model Oracle

But this doesn’t carry over to an oracle forest

He bought some books

Shift-NP
But this doesn’t carry over to an oracle forest
The Dependency Model Oracle

But this doesn’t carry over to an oracle forest

He some books
bought
some books

Shift-NP  Shift-(S\NP)/NP
The Dependency Model Oracle

But this doesn’t carry over to an oracle forest

Shift-\textit{NP}  Shift-(S \backslash NP)/NP  Shift-\textit{NP}/N
But this doesn’t carry over to an oracle forest

Shift-\textit{NP}  Shift-\textit{(S\backslash NP)/NP}  Shift-\textit{NP/\textit{N}}
The Dependency Model Oracle

But this doesn’t carry over to an oracle forest

He some books bought

Shift-\(NP\)  Shift-(\(S\backslash NP\))/\(NP\)  Shift-\(NP/\)\(N\)  Shift-\(N\)
The Dependency Model Oracle

But this doesn’t carry over to an oracle forest

He some books
NP (S\NP)/NP NP/N N
bought
3
2
S/(S\NP)
S/NP
(S\NP)/NP
NP/N
N
Shift-NP Shift-(S\NP)/NP Shift-NP/N Shift-N
The Dependency Model Oracle

But this doesn’t carry over to an oracle forest

```
S[dcl]
   /\          \  (S[dcl]\NP)/NP
  NP        S[dcl]\NP
     /\                  /\        NP
    N    (S[dcl]\NP)/NP   visited   N
   /\                        /\        \\
  N/N        N               visited   N
    |       |                  |        |
  Mr. President      Paris
```

Mr. President

N/N N
But this doesn’t carry over to an oracle forest

\[
S[dcl] \\
NP \\
\quad N \\
\quad N/\ N \\
\quad Mr. President \\
\quad (S[dcl]\NP) /NP \\
\quad visited \\
\quad N \\
\quad (S[dcl]\NP) /NP \\
\quad Paris \\
\quad N/\ N \\
\quad N \\
\quad Mr. President \\
\quad visited \\
\quad (S[dcl]\NP) /NP \\
\quad N \\
\quad N \\
\quad N/\ N
\]
The Dependency Model Oracle

• The dependency oracle

\[ f_d(⟨s, q⟩, (x, c), Φ_G) = \begin{cases} 
  \text{true} & \text{if } s' \sim G \text{ or } s' \simeq G \\
  \text{false} & \text{otherwise}
\end{cases} \]

• Shared ancestor set
  – contains possible valid nodes an item should visit
  – is built on-the-fly during decoding for each action type
  – constructed with each valid action
The Dependency Model

He bought some books.

stack \((s_n, \ldots, s_1, s_0)\) | \(\mathcal{R}(c_{s_0})\)
The Dependency Model

He some books

stack \((s_n, \ldots, s_1, s_0)\)  \(\mathcal{R}(c_{s_0})\)
The Dependency Model

He bought some books

stack \((s_n, \ldots, s_1, s_0)\) \quad \mathcal{R}(c_{s_0})
The Dependency Model

stack \((s_n, \ldots, s_1, s_0)\) | \(\mathcal{R}(c_{s_0})\)
---|---
\(NP\) | ()
\(NP (S\backslash NP)/NP\) |
The Dependency Model

stack \( (s_n, \ldots, s_1, s_0) \) | \( \mathcal{R}(c_{s_0}) \)
---|---
\( NP \) | ()
\( NP \) | (\( S, S \))
The Dependency Model

```
stack (s_n, ..., s_1, s_0)  \mathcal{R}(c_{s_0})
```

<table>
<thead>
<tr>
<th>NP</th>
<th>()</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP</td>
<td>(S, S)</td>
</tr>
<tr>
<td>NP (S \ NP)/NP</td>
<td></td>
</tr>
<tr>
<td>NP (S \ NP)/NP NP</td>
<td></td>
</tr>
<tr>
<td>NP (S \ NP)/NP NP/ N</td>
<td></td>
</tr>
</tbody>
</table>

He bought some books
The Dependency Model

stack \( (s_n, \ldots, s_1, s_0) \)  \( R(c_{s_0}) \)

\( NP \)
\( NP (S\NP)/NP \)
\( NP (S\NP)/NP \ NP/N \)
The Dependency Model

He some books bought

stack \((s_n, \ldots, s_1, s_0)\) \(\mathcal{R}(c_{s_0})\)

\[
\begin{array}{c|c}
NP & () \\
NP (S\backslash NP)/NP & (S, S) \\
NP (S\backslash NP)/NP NP/N & (S \backslash NP, (S\backslash NP)/N) \\
\end{array}
\]
The Dependency Model

He bought some books.

stack \( (s_n, \ldots, s_1, s_0) \)  \( \mathcal{R}(c_{s_0}) \)

\[ \begin{array}{c|c|c}
  NP & () & (S, S) \\
  NP & (S \backslash NP)/NP & (S \backslash NP, (S \backslash NP)/N) \\
  NP & (S \backslash NP)/NP & N
\end{array} \]
The Dependency Model

\[
\begin{array}{c|c}
\text{stack } (s_n, \ldots, s_1, s_0) & \mathcal{R}(c_{s_0}) \\
NP & () \\
NP (S\backslash NP)/NP & (S, S) \\
NP (S\backslash NP)/NP NP/N & (S\backslash NP, (S\backslash NP)/N) \\
NP (S\backslash NP)/NP NP/N N & \\
\end{array}
\]
The Dependency Model

```
stack (s_n, ..., s_1, s_0)        \mathcal{R}(c_{s_0})
NP                              ()
NP (S\NP)/NP                   (S, S)
NP (S\NP)/NP NP/N              (S\NP, (S\NP)/N)
NP (S\NP)/NP NP/N N            (NP)
```

He bought some books
The Dependency Model
The Dependency Model

He bought some books.

stack \( (s_n, \ldots, s_1, s_0) \) | \( \mathcal{R}(c_{s_0}) \)
---|---
\( NP \) | ()
\( NP \ (S \setminus NP) / NP \) | \( (S, S) \)
\( NP \ (S \setminus NP) / NP \ NP / N \) | \( (S \setminus NP, (S \setminus NP) / N) \) ▼
\( NP \ (S \setminus NP) / NP \ NP / N \ N \) | \( (NP) \)
The Dependency Model

He bought some books

stack \((s_n, \ldots, s_1, s_0)\)  \(\mathcal{R}(c_{s_0})\)

\(NP\)  ()
\((S\backslash NP)/NP\)  \((S, S)\)
\((S\backslash NP)/NP\ NP/N\)  \((S\backslash NP, (S\backslash NP)/N)\)
\((S\backslash NP)/NP\ NP/N\ N\)  \((NP)\)
\((S\backslash NP)/NP\ NP\)  \((S\backslash NP)\)
The Dependency Model

stack \( (s_n, \ldots, s_1, s_0) \) | \( \mathcal{R}(c_{s_0}) \)
---|---
\( NP \) | \( () \)
\( NP (S \backslash NP)/NP \) | \( (S, S) \)
\( NP (S \backslash NP)/NP NP/N \) | \( (S \backslash NP, (S \backslash NP)/N) \)
\( NP (S \backslash NP)/NP NP/N/N \) | \( (NP) \)
\( NP S \backslash NP \) | \( (S \backslash NP) \)
The Dependency Model

stack \((s_n, \ldots, s_1, s_0)\) \quad \mathcal{R}(c_{s_0})

\begin{align*}
NP & (\cdot) \\
NP (S \backslash NP) / NP & (S, S) \\
NP (S \backslash NP) / NP NP / N & (S \backslash NP, (S \backslash NP) / N) \\
NP (S \backslash NP) / NP NP / N N & (NP) \\
NP (S \backslash NP) / NP NP & (S \backslash NP) \\
NP S \backslash NP & (\cdot)
\end{align*}
The Dependency Model

he bought some books

stack \((s_n, \ldots, s_1, s_0)\)  \(R(c_{s_0})\)

\[
\begin{array}{c|c}
NP & () \\
NP \ (S \backslash NP) / NP & (S, S) \\
NP \ (S \backslash NP) / NP \ NP / N & (S \backslash NP, (S \backslash NP) / N) \\
NP \ (S \backslash NP) / NP \ NP / N N & (NP) \\
NP \ (S \backslash NP) / NP \ NP & (S \backslash NP) \\
NP \ S \backslash NP & (S)
\end{array}
\]
The Dependency Model

stack \((s_n, \ldots, s_1, s_0)\) | \(R(c_{s_0})\)
---|---
\(NP\) | ()
\(NP (S\backslash NP)/NP\) | \((S, S)\)
\(NP (S\backslash NP)/NP NP/N\) | \((S\backslash NP, (S\backslash NP)/N)\)
\(NP (S\backslash NP)/NP NP/N N\) | \((NP)\)
\(NP (S\backslash NP)/NP NP\) | \((S\backslash NP)\)
\(NP S\backslash NP\) | \((S)\)
\(S\) | ()
Online Training

- The normal-form model uses the perceptron with early update
  - only one correct sequence
  - “violation” is guaranteed [Huang et al., 2012]
Online Training

- Standard early update no longer valid for the dependency model
  - multiple correct items possible in each beam
  - “violation” is *not* guaranteed [Huang et al, 2012]
Online Training

• Standard early update no longer valid for the dependency model
  – multiple correct items possible in each beam
  – “violation” is not guaranteed [Huang et al, 2012]
  – $w \leftarrow w + \phi(\Pi_G[0]) - \phi(\mathcal{B}_i[0])$

from Heng et al., 2013
Experiments

- Standard split of CCGBank: training (2-21), dev (00) and test (23)
- Parser implemented as an extension of the C&C parser
  - unlike ZPAR, outputs dependencies directly
- Auto-pos for all experiments
- Supertagger prob. cutoff set to .0001 for both training and testing
## Development Results

<table>
<thead>
<tr>
<th></th>
<th>LP</th>
<th>LR</th>
<th>LF</th>
<th>LSent. %</th>
<th>CatAcc. %</th>
<th>coverage %</th>
</tr>
</thead>
<tbody>
<tr>
<td>this parser</td>
<td>86.29</td>
<td>84.09</td>
<td>85.18</td>
<td>34.40</td>
<td>92.75</td>
<td>100</td>
</tr>
<tr>
<td>Z&amp;C</td>
<td><strong>87.15</strong></td>
<td>82.95</td>
<td>85.00</td>
<td>33.82</td>
<td><strong>92.77</strong></td>
<td>100</td>
</tr>
<tr>
<td>C&amp;C (normal-form)</td>
<td>85.22</td>
<td>82.52</td>
<td>83.85</td>
<td>31.63</td>
<td>92.40</td>
<td>100</td>
</tr>
<tr>
<td>this parser</td>
<td>86.76</td>
<td><strong>84.90</strong></td>
<td><strong>85.82</strong></td>
<td><strong>34.72</strong></td>
<td><strong>93.20</strong></td>
<td>99.06 (C&amp;C coverage)</td>
</tr>
<tr>
<td>Z&amp;C</td>
<td><strong>87.55</strong></td>
<td>83.63</td>
<td>85.54</td>
<td>34.14</td>
<td>93.11</td>
<td>99.06 (C&amp;C coverage)</td>
</tr>
<tr>
<td>C&amp;C (hybrid)</td>
<td>–</td>
<td>–</td>
<td>85.25</td>
<td>–</td>
<td>–</td>
<td>99.06 (C&amp;C coverage)</td>
</tr>
<tr>
<td>C&amp;C (normal-form)</td>
<td>85.22</td>
<td>84.29</td>
<td>84.76</td>
<td>31.93</td>
<td>92.83</td>
<td>99.06 (C&amp;C coverage)</td>
</tr>
</tbody>
</table>
Development Results

Precision %
Dependency length (bins of 5)
C&C
zpar
this parser

Recall %
Dependency length (bins of 5)
C&C
zpar
this parser
## Development Results

<table>
<thead>
<tr>
<th>category</th>
<th>LP % (t)</th>
<th>LP % (z)</th>
<th>LR % (t)</th>
<th>LR % (z)</th>
<th>LF % (t)</th>
<th>LF % (z)</th>
<th>freq.</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/N</td>
<td>95.53</td>
<td>95.77</td>
<td>95.83</td>
<td>95.79</td>
<td>95.68</td>
<td>95.78</td>
<td>7288</td>
</tr>
<tr>
<td>NP/N</td>
<td>96.53</td>
<td>96.70</td>
<td>97.12</td>
<td>96.59</td>
<td>96.83</td>
<td>96.65</td>
<td>4101</td>
</tr>
<tr>
<td>(NP\NP)/NP</td>
<td>81.64</td>
<td>83.19</td>
<td>90.63</td>
<td>89.24</td>
<td>85.90</td>
<td>86.11</td>
<td>2379</td>
</tr>
<tr>
<td>(NP\NP)/NP</td>
<td>81.70</td>
<td>82.53</td>
<td>88.91</td>
<td>87.99</td>
<td>85.15</td>
<td>85.17</td>
<td>2174</td>
</tr>
<tr>
<td>((S\NP)(S\NP))/NP</td>
<td>77.64</td>
<td>77.60</td>
<td>72.97</td>
<td>71.58</td>
<td>75.24</td>
<td>74.47</td>
<td>1147</td>
</tr>
<tr>
<td>((S\NP)(S\NP))/NP</td>
<td>75.78</td>
<td>76.30</td>
<td>71.27</td>
<td>70.60</td>
<td>73.45</td>
<td>73.34</td>
<td>1058</td>
</tr>
<tr>
<td>((S[dl])\NP)/NP</td>
<td>83.94</td>
<td>85.60</td>
<td>86.04</td>
<td>84.30</td>
<td>84.98</td>
<td>84.95</td>
<td>917</td>
</tr>
<tr>
<td>PP/NP</td>
<td>77.06</td>
<td>73.76</td>
<td>73.63</td>
<td>72.83</td>
<td>75.31</td>
<td>73.29</td>
<td>876</td>
</tr>
<tr>
<td>((S[dl])\NP)/NP</td>
<td>82.03</td>
<td>85.32</td>
<td>83.26</td>
<td>82.00</td>
<td>82.64</td>
<td>83.63</td>
<td>872</td>
</tr>
<tr>
<td>((S\NP)(S\NP))</td>
<td>86.42</td>
<td>84.44</td>
<td>86.19</td>
<td>86.60</td>
<td>86.31</td>
<td>85.51</td>
<td>746</td>
</tr>
</tbody>
</table>
## Final Results

<table>
<thead>
<tr>
<th></th>
<th>LP %</th>
<th>LR %</th>
<th>LF %</th>
<th>LSent. %</th>
<th>CatAcc. %</th>
<th>coverage %</th>
</tr>
</thead>
<tbody>
<tr>
<td>our parser</td>
<td>87.03</td>
<td>85.08</td>
<td>86.04</td>
<td>35.69</td>
<td>93.10</td>
<td>100</td>
</tr>
<tr>
<td>Z&amp;C</td>
<td>87.43</td>
<td>83.61</td>
<td>85.48</td>
<td>35.19</td>
<td>93.12</td>
<td>100</td>
</tr>
<tr>
<td>C&amp;C (normal-form)</td>
<td>85.58</td>
<td>82.85</td>
<td>84.20</td>
<td>32.90</td>
<td>92.84</td>
<td>100</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>LP %</th>
<th>LR %</th>
<th>LF %</th>
<th>LSent. %</th>
<th>CatAcc. %</th>
<th>coverage %</th>
</tr>
</thead>
<tbody>
<tr>
<td>our parser</td>
<td>87.04</td>
<td>85.16</td>
<td>86.09</td>
<td>35.84</td>
<td>93.13</td>
<td>99.58 (C&amp;C coverage)</td>
</tr>
<tr>
<td>Z&amp;C</td>
<td>87.43</td>
<td>83.71</td>
<td>85.53</td>
<td>35.34</td>
<td>93.15</td>
<td>99.58 (C&amp;C coverage)</td>
</tr>
<tr>
<td>C&amp;C (hybrid)</td>
<td>86.17</td>
<td>84.74</td>
<td>85.45</td>
<td>32.92</td>
<td>92.98</td>
<td>99.58 (C&amp;C coverage)</td>
</tr>
<tr>
<td>C&amp;C (normal-form)</td>
<td>85.48</td>
<td>84.60</td>
<td>85.04</td>
<td>33.08</td>
<td>92.86</td>
<td>99.58 (C&amp;C coverage)</td>
</tr>
</tbody>
</table>

+0.5 over Z&C, and +0.6 and +1.5 over C&C hybrid and normal-form models, resp.
Conclusions

• Introduced the first dependency model for shift-reduce CCG parsing
  – only gold-standard dependencies are needed for training
  – the oracle encodes \textit{exponentially} many derivations
  – achieved the best accuracy for shift-reduce CCG parsing
The End

Thank You!