Online Learning of Relaxed CCG Grammars for Parsing to Logical Form

Luke Zettlemoyer and Michael Collins

MIT Computer Science and Artificial Intelligence Lab
Learn Mappings to Logical Form

Given training examples like:

Input: List one way flights to Prague.

Output: $\lambda x.\text{flight}(x) \land \text{one\_way}(x) \land \text{to}(x, \text{PRG})$

Challenging Learning Problem:

• Derivations (or parses) are not annotated

Extending previous approach: [Zettlemoyer & Collins 2005]

• Learn a lexicon and parameters for a weighted Combinatory Categorial Grammar (CCG)
Challenge

Learning CCG grammars works well for complex, grammatical sentences:

Input: Show me flights from Newark and New York to San Francisco or Oakland that are nonstop.

Output: \( \lambda x. \text{flight}(x) \land \text{nonstop}(x) \land (\text{from}(x, \text{PRG}) \lor \text{from}(x, \text{NYC})) \land (\text{to}(x, \text{SFO}) \lor \text{to}(x, \text{OAK})) \)

What about sentences that are common given spontaneous, unedited input?

Input: Boston to Prague the latest on Friday.

Output: \( \text{argmax}( \lambda x. \text{from}(x, \text{BOS}) \land \text{to}(x, \text{PRG}) \land \text{day}(x, \text{FRI}), \lambda y. \text{time}(y)) \)

This talk is about an approach that works for both cases.
Outline

• Background
• Relaxed parsing rules
• Online learning algorithm
• Evaluation
Background

- Combinatory Categorial Grammar (CCG)
- Weighted CCGs
- Learning lexical entries: GENLEX
## CCG Lexicon

<table>
<thead>
<tr>
<th>Words</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>flights</td>
<td>(N : \lambda x.\text{flight}(x))</td>
</tr>
<tr>
<td>to</td>
<td>((\text{N/N})/\text{NP} : \lambda x.\lambda f.\lambda y.f(x) \wedge \text{to}(y,x))</td>
</tr>
<tr>
<td>Prague</td>
<td>(\text{NP} : \text{PRG})</td>
</tr>
<tr>
<td>New York city</td>
<td>(\text{NP} : \text{NYC})</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Parsing Rules (Combinators)

Application

- \( X/Y : f \quad Y : a \quad \Rightarrow \quad X : f(a) \)
- \( Y : a \quad X\backslash Y : f \quad \Rightarrow \quad X : f(a) \)

Composition

- \( X/Y : f \quad Y/Z : g \quad \Rightarrow \quad X/Z : \lambda x.f(g(x)) \)
- \( Z\backslash Y : f \quad X\backslash Y : g \quad \Rightarrow \quad X\backslash Z : \lambda x.f(g(x)) \)

Additional rules:

- Type Raising
- Crossed Composition
CCG Parsing

Show me flights to Prague

- S/N \( \lambda f.f \)
- N \( \lambda x.\text{flight}(x) \)
- (N\N) / NP \( \lambda y.\lambda f.\lambda x.f(y) \land \text{to}(x,y) \)
- NP PRG

- N\N \( \lambda f.\lambda x.f(x) \land \text{to}(x,\text{PRG}) \)
- N \( \lambda x.\text{flight}(x) \land \text{to}(x,\text{PRG}) \)
- S \( \lambda x.\text{flight}(x) \land \text{to}(x,\text{PRG}) \)
Weighted CCG

Given a log-linear model with a CCG lexicon $\Lambda$, a feature vector $f$, and weights $w$.

- The best parse is:

$$y^* = \arg \max_y w \cdot f(x, y)$$

Where we consider all possible parses $y$ for the sentence $x$ given the lexicon $\Lambda$. 
Lexical Generation

Input Training Example

Sentence: Show me flights to Prague.
Logic Form: \( \lambda x. \text{flight}(x) \land \text{to}(x, \text{PRG}) \)

Output Lexicon

<table>
<thead>
<tr>
<th>Words</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show me</td>
<td>S/N : ( \lambda f.f )</td>
</tr>
<tr>
<td>flights</td>
<td>N : ( \lambda x. \text{flight}(x) )</td>
</tr>
<tr>
<td>to</td>
<td>(N\N)/NP : ( \lambda x. \lambda f. \lambda y. f(x) \land \text{to}(y, x) )</td>
</tr>
<tr>
<td>Prague</td>
<td>NP : PRG</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
GENLEX: Substrings cross Categories

Input Training Example

| Sentence:                  | Show me flights to Prague. |
| Logic Form:               | $\lambda x.\text{flight}(x) \land \text{to}(x, \text{PRG})$ |

Output Lexicon

All possible substrings:

Show
me
flights
...
Show me
Show me flights
Show me flights to
...

Categories created by rules that trigger on the logical form:

NP : PRG
N : $\lambda x.\text{flight}(x)$
(S\NP)/NP : $\lambda x.\lambda y.\text{to}(y,x)$
(N\N)/NP : $\lambda y.\lambda f.\lambda x. \ldots$
...

[Zettlemoyer & Collins 2005]
Challenge Revisited

The lexical entries that work for:

<table>
<thead>
<tr>
<th>S/NP</th>
<th>NP/N</th>
<th>N</th>
<th>N\N</th>
<th>N\N</th>
<th>N\N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show me the latest flight from Boston to Prague on Friday</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Will not parse:

<table>
<thead>
<tr>
<th>NP</th>
<th>N\N</th>
<th>NP/N</th>
<th>N\N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boston to Prague the latest on Friday</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Relaxed Parsing Rules

Two changes:

- Add application and composition rules that relax word order
- Add type shifting rules to recover missing words

These rules significantly relax the grammar

- Introduce features to count the number of times each new rule is used in a parse
Review: Application

\[
\begin{align*}
X/Y : f & \quad Y : a \quad \Rightarrow \quad X : f(a) \\
Y : a & \quad X\setminus Y : f \quad \Rightarrow \quad X : f(a)
\end{align*}
\]
Disharmonic Application

- Reverse the direction of the principal category:

\[ X \backslash Y : f \quad Y : a \quad \Rightarrow \quad X : f(a) \]
\[ Y : a \quad X / Y : f \quad \Rightarrow \quad X : f(a) \]

\[ \begin{array}{c|c}
\text{flights} & \text{one way} \\
\hline
\lambda x.\text{flight}(x) & \lambda f.\lambda x.f(x) \land \text{one\_way}(x) \\
\hline
\lambda x.\text{flight}(x) \land \text{one\_way}(x) & \\
\end{array} \]
Review: Composition

\[
\begin{align*}
X/Y : f & \quad Y/Z : g \quad \Rightarrow \quad X/Z : \lambda x. f(g(x)) \\
Y\backslash Z : g & \quad X\backslash Y : f \quad \Rightarrow \quad X\backslash Z : \lambda x. f(g(x))
\end{align*}
\]
Disharmonic Composition

- Reverse the direction of the principal category:

\[
\begin{align*}
X \setminus Y & : f \\
Y / Z & : g \\
Y \setminus Z & : g \\
X / Y & : f
\end{align*}
\Rightarrow
\begin{align*}
X / Z & : \lambda x. f(g(x)) \\
X \setminus Z & : \lambda x. f(g(x))
\end{align*}
\]

<table>
<thead>
<tr>
<th>to Prague</th>
<th>the latest</th>
<th>flight</th>
</tr>
</thead>
<tbody>
<tr>
<td>N\N [ \lambda f \cdot \lambda x. f(x) \land \text{to}(x, \text{PRG}) ]</td>
<td>NP\N [ \lambda f. \text{argmax}(\lambda x. f(x), \lambda x. \text{time}(x)) ]</td>
<td>N [ \lambda x. \text{flight}(x) ]</td>
</tr>
<tr>
<td>NP\N [ \lambda f. \text{argmax}(\lambda x. f(x) \land \text{to}(x, \text{PRG}), \lambda x. \text{time}(x)) ]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Missing content words

**Insert missing semantic content**

- NP : c  =>$ N\backslash N : \lambda f.\lambda x.f(x) \land p(x,c)$

<table>
<thead>
<tr>
<th>flights</th>
<th>Boston</th>
<th>to Prague</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N\backslash N$</td>
<td>$N\backslash N$</td>
<td>$N\backslash N$</td>
</tr>
<tr>
<td>$\lambda x.\text{flight}(x)$</td>
<td>$\lambda f.\lambda x.f(x) \land \text{from}(x,\text{BOS})$</td>
<td>$\lambda f.\lambda x.f(x) \land \text{to}(x,\text{PRG})$</td>
</tr>
<tr>
<td>$\text{BOS}$</td>
<td>$\lambda f.\lambda x.f(x) \land \text{from}(x,\text{BOS})$</td>
<td>$\lambda x.\text{flight}(x) \land \text{from}(x,\text{BOS}) \land \text{to}(x,\text{PRG})$</td>
</tr>
</tbody>
</table>
Missing content-free words

Bypass missing nouns

- \( N\backslash N : f \Rightarrow N : f(\lambda x.\text{true}) \)

\[
\begin{align*}
\text{Northwest Air} & \quad \text{to Prague} \\
N/N & \quad N/N \\
\lambda f.\lambda x.f(x) \land \text{airline}(x, \text{NWA}) & \quad \lambda f.\lambda x.f(x) \land \text{to}(x, \text{PRG}) \\
N & \quad N \\
\lambda x.\text{to}(x, \text{PRG}) & \quad \lambda x.\text{airline}(x, \text{NWA}) \land \text{to}(x, \text{PRG})
\end{align*}
\]
A Complete Parse

<table>
<thead>
<tr>
<th>Boston</th>
<th>to Prague</th>
<th>the latest</th>
<th>on Friday</th>
</tr>
</thead>
</table>
| NP
| NP/N
| NP/N
| N/N |
| NP/N
| NP/N
| N/N |
| N/N |

\[
\lambda f. \lambda x. f(x) \land to(x, PRG)
\]

\[
\lambda f. \lambda x. f(x) \land from(x, BOS)
\]

\[
\argmax(\lambda x. f(x) \land from(x, BOS) \land to(x, PRG), \lambda x. time(x))
\]

\[
\argmax(\lambda x. from(x, BOS) \land to(x, PRG) \land day(x, FRI), \lambda x. time(x))
\]

\[
\lambda f. \lambda x. f(x) \land day(x, FRI)
\]

\[
\lambda x. day(x, FRI)
\]
A Learning Algorithm

The approach is:

• **Online**: processes data set one example at a time

• **Able to Learn Structure**: selects a subset of the lexical entries from GENLEX

• **Error Driven**: uses perceptron-style parameter updates

• **Relaxed**: learns how much to penalize the use of the relaxed parsing rules
Inputs: Training set \{ (x_i, z_i) \mid i=1...n \} of sentences and logical forms. Initial lexicon \Lambda. Initial parameters \( w \). Number of iterations \( T \).

Computation: For \( t = 1...T, i =1...n \):

Step 1: Check Correctness
- Let \( y^* = \arg \max_y w \cdot f(x_i, y) \)
- If \( L(y^*) = z_i \), go to the next example

Step 2: Lexical Generation
- Set \( \lambda = \Lambda \cup \text{GENLEX}(x_i, z_i) \)
- Let \( \hat{y} = \arg \max_{y \text{ s.t. } L(y)=z_i} w \cdot f(x_i, y) \)
- Define \( \lambda_i \) to be the lexical entries in \( y^* \)
- Set lexicon to \( \Lambda = \Lambda \cup \lambda_i \)

Step 3: Update Parameters
- Let \( y' = \arg \max_y w \cdot f(x_i, y) \)
- If \( L(y') \neq z_i \)
  - Set \( w = w + f(x_i, \hat{y}) - f(x_i, y') \)

Output: Lexicon \( \Lambda \) and parameters \( w \).
Related Work

Semantic parsing with:

- Inductive Logic Prog.  [Zelle, Mooney 1996; Thompson, Mooney 2002]
- Probabilistic CFG Parsing  [Miller et. al, 1996; Ge, Mooney 2006]
- Support Vector Mach.  [Kate, Mooney 2006; Nguyen et al. 2006]

CCG:

- Multi-modal CCG  [Clark, Curran 2003]
- Wide coverage semantics  [Baldridge 2002]
- CCG Bank  [Bos et al. 2004]
[Hoekenmaier 2003]
Related Work for Evaluation

Hidden Vector State Model: He and Young 2006
- Learns a probabilistic push-down automaton with EM
- Is integrated with speech recognition

\(\lambda\)-WASP: Wong & Mooney 2007
- Builds a synchronous CFG with statistical machine translation techniques
- Easily applied to different languages

Zettlemoyer and Collins 2005
- Uses GENLEX with maximum likelihood batch training and stricter grammar
Two Natural Language Interfaces

ATIS (travel planning)
- Manually-transcribed speech queries
- 4500 training examples
- 500 example development set
- 500 test examples

Geo880 (geography)
- Edited sentences
- 600 training examples
- 280 test examples
Evaluation Metrics

Precision, Recall, and F-measure for:

• Completely correct logical forms

• Attribute / value partial credit

\[ \lambda x. \text{flight}(x) \land \text{from}(x, \text{BOS}) \land \text{to}(x, \text{PRG}) \]

is represented as:

\[ \{ \text{from} = \text{BOS}, \text{to} = \text{PRG} \} \]
Two-Pass Parsing

Simple method to improve recall:

• For each test sentence that can not be parsed:
  • Reparse with word skipping
  • Every skipped word adds a constant penalty
  • Output the highest scoring new parse

We report results with and without this two-pass parsing strategy
ATIS Test Set

Exact Match Accuracy:

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-Pass</td>
<td>90.61</td>
<td>81.92</td>
<td><strong>86.05</strong></td>
</tr>
<tr>
<td>Two-Pass</td>
<td>85.75</td>
<td>84.60</td>
<td>85.16</td>
</tr>
</tbody>
</table>
### ATIS Test Set

**Partial Credit Accuracy:**

<table>
<thead>
<tr>
<th>Method</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-Pass</td>
<td>96.76</td>
<td>86.89</td>
<td>91.56</td>
</tr>
<tr>
<td>Two-Pass</td>
<td>95.11</td>
<td>96.71</td>
<td><strong>95.9</strong></td>
</tr>
<tr>
<td>He &amp; Young 2006</td>
<td>---</td>
<td>---</td>
<td>90.3</td>
</tr>
</tbody>
</table>
Geo880 Test Set

Exact Match Accuracy:

<table>
<thead>
<tr>
<th>Method</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-Pass</td>
<td>95.49</td>
<td>83.20</td>
<td><strong>88.93</strong></td>
</tr>
<tr>
<td>Two-Pass</td>
<td>91.63</td>
<td>86.07</td>
<td>88.76</td>
</tr>
<tr>
<td>Zettlemoyer &amp; Collins 2005</td>
<td>96.25</td>
<td>79.29</td>
<td>86.95</td>
</tr>
<tr>
<td>Wong &amp; Money 2007</td>
<td>93.72</td>
<td>80.00</td>
<td>86.31</td>
</tr>
</tbody>
</table>
# ATIS Development Set

**Exact Match Accuracy:**

<table>
<thead>
<tr>
<th>Method</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full online method</td>
<td>87.26</td>
<td>74.44</td>
<td><strong>80.35</strong></td>
</tr>
<tr>
<td>Without features for new rules</td>
<td>70.33</td>
<td>42.45</td>
<td>52.95</td>
</tr>
<tr>
<td>Without relaxed word order rules</td>
<td>82.81</td>
<td>63.98</td>
<td>72.19</td>
</tr>
<tr>
<td>Without missing word rules</td>
<td>77.31</td>
<td>56.94</td>
<td>65.58</td>
</tr>
</tbody>
</table>
We presented an algorithm that:

• Learns the lexicon and parameters for a weighted CCG
  • Introduces operators to parse relaxed word order and recover missing words
  • Uses online, error-driven updates
• Improves parsing accuracy for spontaneous, unedited inputs
• Maintains the advantages of using a detailed grammatical formalism
The End

Thanks