Montague Semantics

\[ \text{sings}(e470) \]

- function application: apply this to e470

\[ \lambda y. \text{sings}(y) \]

- takes one argument (y, the entity) and returns a logical form \( \text{sings}(y) \)

- We can use the syntactic parse as a bridge to the lambda-calculus representation, build up a logical form (our model) compositionally

Combinatory Categorial Grammar

- Steedman+Szabolcsi (1980s): formalism bridging syntax and semantics
- Parallel derivations of syntactic parse and lambda calculus expression
- Syntactic categories (for this lecture): S, NP, “slash” categories
- S\NP: “if I combine with an NP on my left side, I form a sentence” — verb
- When you apply this, there has to be a parallel instance of function application on the semantics side

Combinatory Categorial Grammar

- Steedman+Szabolcsi (1980s): formalism bridging syntax and semantics
- Syntactic categories (for this lecture): S, NP, “slash” categories
- S\NP: “if I combine with an NP on my left side, I form a sentence” — verb
- (S\NP)/NP: “I need an NP on my right and then on my left” — verb with a direct object

CCG Parsing

- “What” is a very complex type: needs a noun and needs a S\NP to form a sentence. S\NP is basically a verb phrase (border Texas)

Zettlemoyer and Collins (2005)
**CCG Parsing**

- These question are **compositional**: we can build bigger ones out of smaller pieces
  - *What states border Texas?*
  - *What states border states bordering Texas?*

**Training CCG Parsers**

- Training data looks like pairs of sentences and logical forms
  - *What states border Texas*  \( \lambda x. \text{state}(x) \land \text{borders}(x, e89) \)
  - *What borders Texas*  \( \lambda x. \text{borders}(x, e89) \)
  - ... 

- Unlike PCFGs, we don’t know which words yielded which fragments of CCG
- Requires an “unsupervised” approach like Model 1 for word alignment

---

**CCG Parsing**

- Many ways to build these parsers
- One approach: run a “supertagger” (tags the sentence with complex labels), then run the parser

\[
\begin{array}{c c c c}
 \text{What} & \text{states} & \text{border} & \text{Texas} \\
\frac{(S/(S\backslash NP))/N}{\lambda f. \lambda g. \lambda x. f(x) \land g(x)} & \frac{N}{\lambda x. \text{state}(x)} & \frac{(S\backslash NP)/NP}{\lambda x. \lambda y. \text{borders}(y, x)} & \frac{NP}{\text{texas}} \\
\frac{S/(S\backslash NP)}{\lambda g. \lambda x. \text{state}(x) \land g(x)} & \frac{(S\backslash NP)}{\lambda y. \text{borders}(y, \text{texas})} & \frac{S}{\lambda x. \text{state}(x) \land \text{borders}(x, \text{texas})} & > \\
\end{array}
\]

- “What” is a **very** complex type: needs a noun and needs a S\NP to form a sentence. S\NP is basically a verb phrase (border Texas)
- *What in this case knows that there are two predicates (states and border Texas). This is not a general thing*  
  
  Zettlemoyer and Collins (2005)

---

**CCG Parsing**

- Many ways to build these parsers
- One approach: run a “supertagger” (tags the sentence with complex labels), then run the parser

\[
\begin{array}{c c c c}
 \text{What} & \text{states} & \text{border} & \text{Texas} \\
\frac{(S/(S\backslash NP))/N}{\lambda f. \lambda g. \lambda x. f(x) \land g(x)} & \frac{N}{\lambda x. \text{state}(x)} & \frac{(S\backslash NP)/NP}{\lambda x. \lambda y. \text{borders}(y, x)} & \frac{NP}{\text{texas}} \\
\frac{S/(S\backslash NP)}{\lambda g. \lambda x. \text{state}(x) \land g(x)} & \frac{(S\backslash NP)}{\lambda y. \text{borders}(y, \text{texas})} & \frac{S}{\lambda x. \text{state}(x) \land \text{borders}(x, \text{texas})} & > \\
\end{array}
\]

- Parsing is easy once you have the tags, so we’ve reduced it to a (hard) tagging problem

Zettlemoyer and Collins (2005)
**Seq2seq Semantic Parsing**

- Writing down a linearized form of the semantic parse, train seq2seq models to directly translate into this representation
- What are some benefits of this approach compared to grammar-based?
- What might be some concerns about this approach? How do we mitigate them?

---

**Handling Invariances**

- “what states border Texas”
- “what states border Ohio”

  - Parsing-based approaches handle these the same way
  - Possible divergences: features, different weights in the lexicon
  - Can we get seq2seq semantic parsers to handle these the same way?
  - Key idea: do data augmentation by synthetically creating more data from a single example

---

**Semantic Parsing as Translation**

- “what states border Texas”

  \[
  \text{lambda } x \ ( \ \text{state } ( x ) \ \text{and border } ( x , e89 ) )
  \]

  - What are some benefits of this approach compared to grammar-based?
  - What might be some concerns about this approach? How do we mitigate them?

---

**Semantic Parsing as Translation**

- Prolog
- Lambda calculus
- Other DSLs

  - Handle all of these with uniform machinery!

---

**Jia and Liang (2016)**
Semantic Parsing as Translation

<table>
<thead>
<tr>
<th>Previous Work</th>
<th>Geo</th>
<th>ATIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zeitelmseyer and Collins (2007)</td>
<td></td>
<td>84.6</td>
</tr>
<tr>
<td>Kwiatkowski et al. (2010)</td>
<td>88.9</td>
<td></td>
</tr>
<tr>
<td>Liang et al. (2011)</td>
<td>91.1</td>
<td></td>
</tr>
<tr>
<td>Kwiatkowski et al. (2011)</td>
<td>88.6</td>
<td></td>
</tr>
<tr>
<td>Poon (2013)</td>
<td>88.9</td>
<td></td>
</tr>
<tr>
<td>Zhao and Huang (2015)</td>
<td></td>
<td>84.2</td>
</tr>
</tbody>
</table>

- Three forms of data augmentation all help
- Results on these tasks are still not as strong as hand-tuned systems from 10 years ago, but the same simple model can do well at all problems

Jia and Liang (2016)

Applications

- GeoQuery (Zelle and Mooney, 1996): answering questions about states (~80% accuracy)
- Jobs: answering questions about job postings (~80% accuracy)
- ATIS: flight search
- Can do well on all of these tasks if you handcraft systems and use plenty of training data: these domains aren’t that rich

Next Time

- QA from raw text: how do we answer a question about a passage?
- Neural networks for QA
- Final project discussion