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)
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)
These question are *compositional*: we can build bigger ones out of smaller pieces

*What states border Texas?*

*What states border states bordering Texas?*

*What states border states bordering states bordering Texas?*

In general, answering this does require parsing and not just slot-filling
“to” needs an NP (destination) and N (parent)

“Show me” is a no-op
Many ways to build these parsers

One approach: run a “supertagger” (tags the sentence with complex labels), then run the parser

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

Zettlemoyer and Collins (2005)
Training CCG Parsers

- Training data looks like pairs of sentences and logical forms

\[ \text{What states border Texas} \quad \lambda x. \text{state}(x) \land \text{borders}(x, e89) \]
\[ \text{What borders Texas} \quad \lambda x. \text{borders}(x, e89) \] ...

- What can we learn from these?

- Problem: we don’t know the derivation
  - \text{Texas} corresponds to NP | e89 in the logical form (easy to figure out)
  - \text{What} corresponds to (S/(S\(\backslash\)NP))/N | \lambda f.\lambda g.\lambda x. f(x) \land g(x)
  - How do we infer that without being told it?
Lexicon

- GENLEX: takes sentence $S$ and logical form $L$. Break up logical form into chunks $C(L)$, assume any substring of $S$ might map to any chunk.

  \[ \text{What states border Texas} \quad \lambda x. \text{state}(x) \land \text{borders}(x, e89) \]

- Chunks inferred from the logic form based on rules:
  - NP: $e89$
  - $(S\backslash NP)/NP: \lambda x. \lambda y. \text{borders}(x, y)$

- Any substring can parse to any of these in the lexicon:
  - Texas -> NP: $e89$ is correct
  - border Texas -> NP: $e89$
  - What states border Texas -> NP: $e89$

... Zettlemoyer and Collins (2005)
Learning

- Unsupervised learning of correspondences, like word alignment

- Iterative procedure: estimate “best” parses that derive each logical form, retrain the parser using these parses with supervised learning

- Eventually we converge on the right parses at the same time that we learn a model to build them

Zettlemoyer and Collins (2005)
Seq2seq Semantic Parsing
Semantic Parsing as Translation

“what states border Texas”

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

- Write 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?

Jia and Liang (2016)
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: don’t change the model, change the data
- “Data augmentation”: encode invariances by automatically generating new training examples
Data Augmentation

Jia and Liang (2016)

\[
\text{ROOT} \rightarrow \langle \text{"what states border STATEID ?"}, \\
\quad \text{answer}(\text{NV}, (\text{state}(\text{V0}), \text{next_to}(\text{V0}, \text{NV}), \text{const}(\text{V0}, \text{stateid}(	ext{STATEID})))) \rangle \\
\text{STATEID} \rightarrow \langle \text{"texas"}, \text{texas} \rangle \\
\text{STATEID} \rightarrow \langle \text{"ohio"}, \text{ohio} \rangle \\
\]

- Lets us synthesize a “what states border ohio ?” example
- Abstract out entities: now we can “remix” examples and encode invariance to entity ID. More complicated remixes too
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>Zettlemoyer 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>82.8</td>
</tr>
<tr>
<td>Poon (2013)</td>
<td></td>
<td>83.5</td>
</tr>
<tr>
<td>Zhao and Huang (2015)</td>
<td>88.9</td>
<td>84.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Our Model</th>
<th>GEO</th>
<th>ATIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Recombination</td>
<td>85.0</td>
<td>76.3</td>
</tr>
<tr>
<td>AbsEntities</td>
<td>85.4</td>
<td>79.9</td>
</tr>
<tr>
<td>AbsWholePhrases</td>
<td>87.5</td>
<td></td>
</tr>
<tr>
<td>CONCAT-2</td>
<td>84.6</td>
<td>79.0</td>
</tr>
<tr>
<td>CONCAT-3</td>
<td></td>
<td>77.5</td>
</tr>
<tr>
<td>AWP + AE</td>
<td>88.9</td>
<td></td>
</tr>
<tr>
<td>AE + C2</td>
<td></td>
<td>78.8</td>
</tr>
<tr>
<td>AWP + AE + C2</td>
<td>89.3</td>
<td>83.3</td>
</tr>
<tr>
<td>AE + C3</td>
<td></td>
<td></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
Can use for other semantic parsing-like tasks

Predict regex from text

Problem: requires a lot of data: 10,000 examples needed to get ~60% accuracy on pretty simple regexes

Locascio et al. (2016)
SQL Generation

- Convert natural language description into a SQL query against some DB

- How to ensure that well-formed SQL is generated?
  - Three seq2seq models

- How to capture column names + constants?
  - Pointer mechanisms

Question:
How many CFL teams are from York College?

SQL:
SELECT COUNT CFL Team FROM CFLDraft WHERE College = "York"

Zhong et al. (2017)