Montague Semantics

sings(e470)

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

function application: apply this to e470

takes one argument \((y, \text{the entity})\) 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

\[
S \quad \text{sings(e728)}
\]
\[
\text{NP} \quad \text{e728}
\]
\[
\lambda y. \text{sings(y)}
\]

Eminem  \quad \text{sings}
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

Eminem sings e728

S\NP

NP e728

λy. sings(y)

S

sings(e728)

NP e101

λx. λy borders(y, x)

S\NP

Oklahoma

borders(e101, e89)

NP e89

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)

Zettlemoyer and Collins (2005)
“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?*
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

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

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

  ... 

- Unlike PCFGs, we don’t know which words yielded which fragments of CCG

- Requires an “unsupervised” approach like Model 1 for word alignment

Zettlemoyer and Collins (2005)
Seq2seq Semantic Parsing
“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: do data augmentation by synthetically creating more data from a single example
Seman,c\textsubscript{c} Parsing as Transla,on

\begin{tabular}{|c|}
\hline
\textbf{GEO} \hfill \textbf{ATIS} \hfill \textbf{Overnight} \\
\hline
\textit{x: “what is the population of iowa?”} \hfill \textit{x: “can you list all flights from chicago to milwaukee”} \hfill \textit{x: “when is the weekly standup”} \\
\textit{y: \_answer ( NV , ( \_population ( NV , V1 ) , \_const ( V0 , \_stateid ( iowa ) ) ) )} \hfill \textit{y: ( \_lambda $0 \ e ( \_and ( \_flight $0 ) ( \_from $0 \ chicago : \_ci ) ( \_to $0 \ milwaukee : \_ci ) ) )} \hfill \textit{y: ( call listValue ( call getProperty meeting.weekly_standup ( string start_time ) ) )} \\
\hline
\end{tabular}

- Prolog
- Lambda calculus
- Other DSLs

\textbf{\textcopyright Jia and Liang (2016)}

\textbf{Handle all of these with uniform machinery!}
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
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