CCG Parsing

CCG Parsing

$$
\begin{array}{cccc}
\text{What} & \text{states} & \text{border} & \text{Texas} \\
\hline
(S/(S\backslash NP))/N & N & (S\backslash NP)/NP & NP \\
\lambda f.\lambda g.\lambda x.f(x) \wedge g(x) & \lambda x.state(x) & \lambda x.\lambda y.borders(y,x) & texas \\
\cline{3-4}
 & & \multicolumn{2}{c}{(S\backslash NP) \quad >} \\
 & & \multicolumn{2}{c}{\lambda y.borders(y, texas)}
\end{array}
$$

- "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*)

Zettlemoyer and Collins (2005)

CCG Parsing

$$
\begin{array}{cccc}
\text{What} & \text{states} & \text{border} & \text{Texas} \\
\hline
(S/(S\backslash NP))/N & N & (S\backslash NP)/NP & NP \\
\lambda f.\lambda g.\lambda x.f(x) \wedge g(x) & \lambda x.state(x) & \lambda x.\lambda y.borders(y,x) & texas
\end{array}
$$

$$
\begin{array}{cc}
\xrightarrow{\qquad\qquad\qquad\qquad} > & \xrightarrow{\qquad\qquad\qquad\qquad} > \\
S/(S\backslash NP) & (S\backslash NP) \\
\lambda g.\lambda x.state(x) \wedge g(x) & \lambda y.borders(y, texas)
\end{array}
$$

$$
\begin{array}{c}
\xrightarrow{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad} > \\
S \\
\lambda x.state(x) \wedge borders(x, texas)
\end{array}
$$

- “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

- 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

CCG Parsing

Show me	flights	to	Prague
S/N $\lambda f.f$	N $\lambda x.flight(x)$	(N\N)/NP $\lambda y.\lambda f.\lambda x.f(y) \wedge to(x,y)$	NP PRG

N\N

$\lambda f.\lambda x.f(x) \wedge to(x,PRG)$

N

$\lambda x.flight(x) \wedge to(x,PRG)$

S

$\lambda x.flight(x) \wedge to(x,PRG)$

- "to" needs an NP (destination) and N (parent)
- "Show me" is a no-op

Slide credit: Dan Klein

CCG Parsing

- Many ways to build these parsers

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

What	states	border	Texas
$(S/(S\backslash NP))/N$	N	$(S\backslash NP)/NP$	NP
$\lambda f.\lambda g.\lambda x.f(x) \wedge g(x)$	$\lambda x.state(x)$	$\lambda x.\lambda y.borders(y,x)$	$texas$

  - 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

What states border Texas	λx. state(x) $\wedge$ borders(x, e89)
What borders Texas	λx. borders(x, e89)

…

- What can we learn from these?
- Problem: we don't know the derivation
  - *Texas* corresponds to NP \| e89 in the logical form (easy to figure out)
  - *What* corresponds to (S/(S\NP))/N \| λf.λg.λx. f(x) $\wedge$ 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

What states border Texas $\lambda x.\ \mathtt{state}(x) \wedge \mathtt{borders}(x, \mathtt{e89})$

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

$\textsc{Root} \rightarrow \langle$ "*what states border* $\textsc{StateId}$ *?*",
`answer(NV, (state(V0), next_to(V0, NV), const(V0, stateid(`$\textsc{StateId}$`)))))` $\rangle$

$\textsc{StateId} \rightarrow \langle$ "*texas*", `texas` $\rangle$

$\textsc{StateId} \rightarrow \langle$"*ohio*", `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

Geo
x: *"what is the population of iowa ?"*
```
y: _answer ( NV , (
  _population ( NV , V1 ) , _const (
    V0 , _stateid ( iowa ) ) ) )
```

ATIS
x: *"can you list all flights from chicago to milwaukee"*
```
y: ( _lambda $0 e ( _and
   ( _flight $0 )
   ( _from $0 chicago :  _ci )
   ( _to $0 milwaukee :  _ci ) ) )
```

Overnight
x: *"when is the weekly standup"*
```
y: ( call listValue ( call
    getProperty meeting.weekly_standup
    ( string start_time ) ) )
```

- Prolog
- Lambda calculus
- Other DSLs

- Handle all of these with uniform machinery!

Jia and Liang (2016)

Semantic Parsing as Translation

	GEO	ATIS
Previous Work		
Zettlemoyer and Collins (2007)		**84.6**
Kwiatkowski et al. (2010)	88.9	
Liang et al. (2011)[2]	91.1	
Kwiatkowski et al. (2011)	88.6	82.8
Poon (2013)		83.5
Zhao and Huang (2015)	88.9	84.2
Our Model		
No Recombination	85.0	76.3
ABSENTITIES	85.4	79.9
ABSWHOLEPHRASES	87.5	
CONCAT-2	84.6	79.0
CONCAT-3		77.5
AWP + AE	88.9	
AE + C2		78.8
AWP + AE + C2	**89.3**	
AE + C3		83.3

- 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

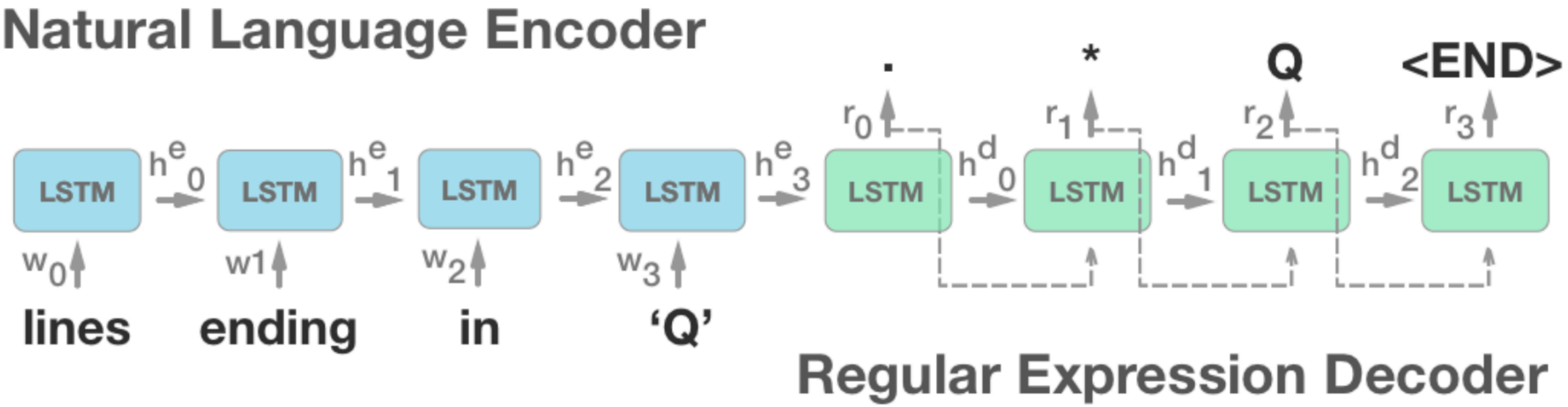
Jia and Liang (2016)

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

Regex Prediction

- Can use for other semantic parsing-like tasks
- Predict regex from text

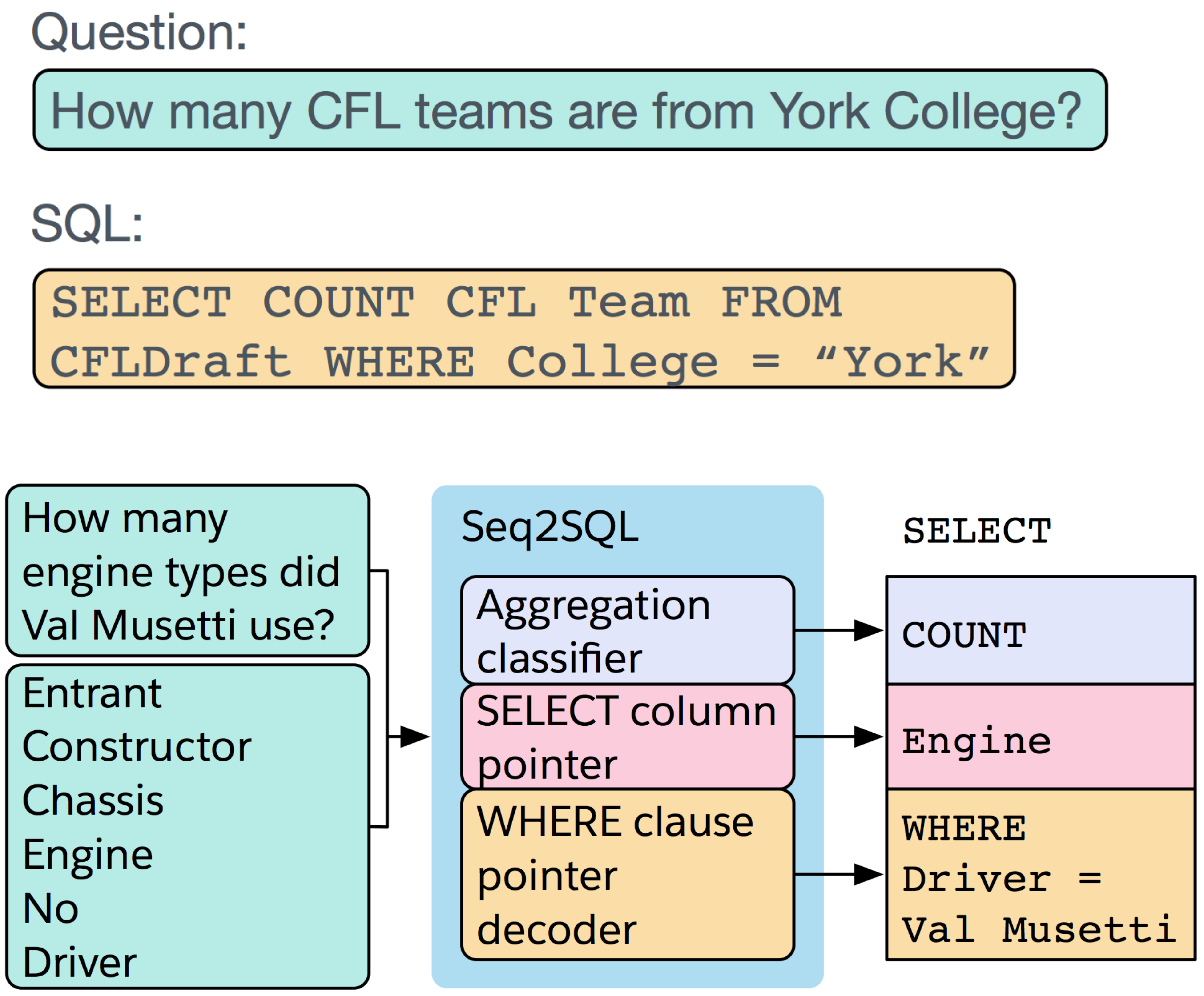


- Problem: requires a lot of data: 10,000 examples needed to get ~60% accuracy on pretty simple regexes

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



Zhong et al. (2017)