Formal Semantics

Jacob Andreas / MIT 6.804–6.864 / Spring 2020
Reminder: Hand in project proposals today if you want feedback in time!
Recap: syntax and question answering
Problem 1

Each of the three girls has a platypus.

Each of the three girls climbed the mountain.

How many platypuses?

How many mountains?
Each of the three girls has a platypus.
Each of the three girls climbed the mountain.
There are 128 cities in South Carolina.

<table>
<thead>
<tr>
<th>name</th>
<th>type</th>
<th>coastal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Columbia</td>
<td>city</td>
<td>no</td>
</tr>
<tr>
<td>Cooper</td>
<td>river</td>
<td>yes</td>
</tr>
<tr>
<td>Charleston</td>
<td>city</td>
<td>yes</td>
</tr>
</tbody>
</table>
Barack Obama was the 44th President of the United States. Obama was born on August 4 in Honolulu, Hawaii. In late August 1961, Obama's mother moved with him to the University of Washington in Seattle for a year...

Is Barack Obama from the United States?
Compositional semantics

It’s not enough to have structured representations of syntax: We also need structured representations of meaning.
It’s not enough to have structured representations of syntax: We also need structured representations of meaning.

Today:

How do we get from language to meaning?
Representing meaning
Meaning in formal languages

\[ a + b = 17 \]
Meaning in formal languages

\[ a + b = 17 \]
Meaning in formal languages

\[ a + b = 17 \]

\[ a = ? \]

\[ b = ? \]
Meanings are sets of valid assignments

\[ a + b = 17 \]

\{a=0, b=0\} \quad \{a=17, b=0\}

\{a=3, b=10\} \quad \{a=10, b=7\}

\{a=5, b=12\} \quad \{a=5, b=5\}
Meanings are sets of valid assignments

\[ a + b = 17 \]

\[
\begin{align*}
\{a=0, \ b=0\} & \quad \times \\
\{a=3, \ b=10\} & \quad \times \\
\{a=5, \ b=12\} & \quad \checkmark \\
\{a=17, \ b=0\} & \quad \checkmark \\
\{a=10, \ b=7\} & \quad \checkmark \\
\{a=5, \ b=5\} & \quad \times
\end{align*}
\]
Meanings are sets of valid assignments

\[ a + 3 = 20 - b \]

\{a=0, b=0\} \times
\{a=3, b=10\} \times
\{a=5, b=12\} \checkmark
\{a=17, b=0\} \checkmark
\{a=10, b=7\} \checkmark
\{a=5, b=5\} \times
Meanings are *functions* that judge validity

\[
\begin{align*}
[a + b &= 17] \\
\{a=5, b=12\} &\checkmark
\end{align*}
\]
Meanings are *functions* that judge validity

\[ a + b = 17 \]

\{a=3, b=10\}
Lessons from math

\[\[a + b = 17\]\]

The meaning of a statement is the set of possible worlds consistent with that statement.

Here, a “possible world” is an assignment of values to variables.

\{a=3, b=10\}
Meaning in natural languages

Pat likes Sal.
Representing possible worlds

Individuals
- Pat
- Sal

Properties
- whale
- sad

Relations
- loves
- contains
Example world

Pat

Sal

Sam

Lou
Different example world

Pat

Sal

Sam

Lou

loves

loves

loves

sad

sad

sad

sad

sad
### Representing possible worlds

<table>
<thead>
<tr>
<th>Individuals</th>
<th>Pat</th>
<th>Sal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Properties</td>
<td>whale={Lou}, sad={Pat,Sal}</td>
<td></td>
</tr>
<tr>
<td>Relations</td>
<td>likes={(Pat,Sal),(Sal,Sam)}</td>
<td></td>
</tr>
</tbody>
</table>
Pat likes Sal.
Interpretations of sentences

Lou is a shark.
Interpretations of sentences

**Sam is inside Lou, a shark.**
The meaning of a sentence is the set of possible worlds it picks out.
Possible worlds and logical forms
Explicit representation is too hard

Pat likes Sal.
Meanings as functions

[[Pat likes Sal]]
Meanings as logical statements

\[
\llbracket \text{Pat likes Sal} \rrbracket
\]

\text{likes(Pat, Sal)}
Expressing functions with logic

*Pat likes Sal*

likes(Pat, Sal)
Meanings as logical statements

Lou is a shark

\(\text{shark}(\text{Lou})\)
Meanings as logical statements

Sam is inside Lou, a shark
Sam is inside Lou, a shark

\[
\text{shark}(\text{Lou}) \land \text{contains}(\text{Lou}, \text{Sam})
\]
Meanings as logical statements

Nobody likes Lou
Meanings as logical statements

Nobody likes Lou

∀x. ¬likes(x, Lou)
Meanings as logical statements

Everyone who knows Sal is happy
Meanings as logical statements

Everyone who knows Sal is happy

∀x. knows(x, Sal) → happy(x)
Key idea

Collections of possible worlds can be compactly represented with logical forms.
Compositionality of meaning

*Pat likes Sal*  \[\text{likes}(\text{Pat},\text{Sal})\]

*Lou is a shark*  \[\text{shark}(\text{Lou})\]

*Sam is inside Lou, a shark*  \[\text{shark}(\text{Lou}) \land \text{contains}(\text{Lou},\text{Sam})\]

*Nobody likes Lou*  \[\forall x. \neg \text{likes}(x,\text{Lou})\]
Compositionality of meaning

- **Pat** likes **Sal**
  - \text{likes}(\text{Pat, Sal})

- **Lou** is a **shark**
  - \text{shark}(\text{Lou})

- **Sam** is inside **Lou**, a **shark**
  - \text{shark}(\text{Lou}) \land \text{contains}(\text{Lou, Sam})

- Nobody likes **Lou**
  - \( \forall x. \neg \text{likes}(x, \text{Lou}) \)
Compositionality of meaning

Pat likes Sal

likes(Pat, Sal)

Lou is a shark

shark(Lou)

Sam is inside Lou, a shark

shark(Lou) ∧ contains(Lou, Sam)

Nobody likes Lou

∀x.¬likes(x, Lou)
Compositionality of meaning

A Sal le gusta Pat

Lou es un tiburón

Sam está dentro de Lou, un tiburón

A nadie le gusta Lou

likes(Pat, Sal)

shark(Lou)

shark(Lou) ∧ contains(Lou, Sam)

∀x.¬likes(x, Lou)
Compositionality of meaning

\( a_{12} b_{5} c_{67} a_{8} \) likes(Pat, Sal)

\( a_{12} b_{5} c_{0} a_{0} \) shark(Lou)

\( a_{12} b_{16} c_{12} c_{12} \) shark(Lou) \( \land \) contains(Lou, Sam)

\( a_{53} \) \( \forall x. \lnot \) likes(x, Lou)
**KEY IDEA**

Pieces of logical forms correspond to pieces of language
Building a lexicon

Sam is inside Lou, a shark  \( \text{shark}(\text{Lou}) \land \text{contains}(\text{Lou}, \text{Sam}) \)

*Pat*: Pat

*Sal*: Sal

*Sam*: Sam

*Lou*: Lou
Building a lexicon

Sam is inside Lou, a shark  \( \text{shark}(\text{Lou}) \land \text{contains}(\text{Lou}, \text{Sam}) \)

\[\begin{align*}
\text{Pat: } & \text{ Pat} \\
\text{Sal: } & \text{ Sal} \\
\text{Sam: } & \text{ Sam} \\
\text{Lou: } & \text{ Lou}
\end{align*}\]
Building a lexicon

Sam is inside Lou, a shark \[\text{shark}(\text{Lou}) \land \text{contains}(\text{Lou}, \text{Sam})\]

Pat: Pat \hspace{2cm} shark: \(\lambda x.\text{shark}(x)\)

Sal: Sal

Sam: Sam

Lou: Lou
Building a lexicon

Sam is inside Lou, a shark  shark(Lou) ∧ contains(Lou, Sam)

Pat: Pat  shark: \( \lambda x.\text{shark}(x) \)
Sal: Sal  likes: \( \lambda yx.\text{likes}(x, y) \)
Sam: Sam  nobody: \( \lambda f. \forall x. \neg f(x) \)
Lou: Lou  ...
Learning semantic parsers
Seq-to-seq semantic parsing

¬ likes ( Pat , Sal )

Pat doesn’t like Sal .

transformation
Decoder constraints

\[ \forall Sal \neg \text{likes} (Pat, Lou) \]

Pat doesn’t like Sal.
Pat doesn't like Sal.

Syntactically malformed

Doesn't type check
Tree-shaped decoders

Pat doesn’t like Sal’s brother

 ¬

likes

Pat brother

Sal
Tree-shaped decoders

Pat doesn't like Sal's brother

RNN states are updated based on parents and siblings, not arbitrary neighbors.

[e.g. Dong and Lapata 2016]
Learning from denotations

Logical form supervision:

*Pat doesn’t like Lou.*  
\(\neg \text{likes(Pat, Lou)}\)

Answer supervision:

Learn from (question, world, answer) triples without LFs!

Who does *Pat* like?  
Sal
Maximum likelihood estimation

\[ p(\text{answer} \mid \text{question}) = \sum_{\text{LF}} p(\text{answer} \mid \text{LF}) \ p(\text{LF} \mid \text{question}) \]

deterministic logical evaluation

semantic parser
Maximum likelihood estimation

$p(\text{answer} \mid \text{question}) = \sum_{\text{LF}} p(\text{answer} \mid \text{LF}) \ p(\text{LF} \mid \text{question})$

deterministic logical evaluation

compare:

$p(\text{sentence}) = \sum_{\text{tree}} p(\text{sentence} \mid \text{tree}) \ p(\text{tree})$

semantic parser

syntactic parser
Computational challenges

Can’t efficiently compute this sum: no way to factor scoring fn over pieces of LFs.

\[
p(\text{answer} \mid \text{question}) = \sum_{\text{LF}} p(\text{answer} \mid \text{LF}) \, p(\text{LF} \mid \text{question})
\]

no dynamic program!

dynamic program (CKY)

\[
p(\text{sentence}) = \sum_{\text{tree}} p(\text{sentence} \mid \text{tree}) \, p(\text{tree})
\]
Computational challenges

Hard search problem!

\[ p(\text{answer} \mid \text{question}) = \sum_{L\text{F}} p(\text{answer} \mid \text{L}\text{F}) p(\text{L}\text{F} \mid \text{question}) \]

This is 0 for almost all LFs
Margin losses

\[ L(s, y) = \left[ \max(s_{-y}) - s_y + c \right]_+ \]

- \( s_y \): scores other than \( y \)
- \( [x]_+ := \max(x, 0) \)

Idea: try to make the score of the right label \( s_y \) at least \( c \) greater than the score of every wrong label.
Structured margin

\[ L(s, y) = \left[ \max_{\text{LF}^-, \text{LF}^+} s(\text{LF}^-) - s(\text{LF}^+) + c \right]_+ \]

highest-scoring LF with the wrong answer

highest-scoring LF with the right answer

Each loss computation involves two search problems: solve with whatever heuristic you want!
“Hard EM”

Alternate between:

\[ \text{LF}^* = \arg\max_{\text{LF}} p(\text{answer} \mid \text{LF}) \ p(\text{LF} \mid \text{question}; \theta) \]

\[ \theta^* = \arg\max_{\theta} p(\text{answer} \mid \text{LF}) \ p(\text{LF} \mid \text{question}; \theta) \]

(pick a “pseudo-gold”, treat it as gold, update params)
Lexicon-based semantic parsing

\[ p(\lambda y. \text{likes}(\text{Pat}, y) \mid \text{who does Pat like?}) \]

\[ \propto \exp \{ f(\text{like}, \lambda xy. \text{likes}(x, y)) + f(\text{Pat}, \text{Pat}) + \ldots \} \]
Neural semantic parsing from denotations

Some combination of hard EM and reinforcement learning.

Way less computation / sample efficient than lexicon-based approaches, but better scoring function.

$$\theta^* = \arg\max_{\theta} p(\text{answer} \mid \text{LF}) \ p(\text{LF} \mid \text{question}; \theta)$$
Semantic parsing via paraphrasing

1. Write a rule-based procedure for turning logical forms into sentences

\[ \lambda y. \text{likes}(y, \text{brother(Sal)}) \rightarrow \text{what likes brother of Sal} \]

2. Score LF based on similarity between the input sentence and fake one

\[ p(LF \mid \text{question}) \propto f(\text{who is it that likes Sal’s brother, what likes brother of Sal}) \]

use paraphrase features

[Berant and Liang 2014]
Aside: program synthesis

\[
\max_{\text{LF}: p(\text{answer}|\text{LF}) > 0} f(\text{LF} | \text{question})
\]

Huge amount of work on solving this problem in the programming languages literature!

(not widely used in NLP)
Why not just predict answers directly?

What color is the necktie?

<table>
<thead>
<tr>
<th>name</th>
<th>type</th>
<th>coastal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Columbia</td>
<td>city</td>
<td>no</td>
</tr>
<tr>
<td>Cooper</td>
<td>river</td>
<td>yes</td>
</tr>
<tr>
<td>Charleston</td>
<td>city</td>
<td>yes</td>
</tr>
</tbody>
</table>

Still hard for “unstructured” neural models!
Structured attention mechanisms

<table>
<thead>
<tr>
<th>name</th>
<th>type</th>
<th>coastal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Columbia</td>
<td>city</td>
<td>no</td>
</tr>
<tr>
<td>Cooper</td>
<td>river</td>
<td>yes</td>
</tr>
<tr>
<td>Charleston</td>
<td>city</td>
<td>yes</td>
</tr>
</tbody>
</table>

What city is on the coast?

Key-value attention tailored for tabular world representations

[Yin et al. 2016]
Does the blue cylinder have the same material as the big block on the right side of the red metallic thing?

\[
\lambda w \ \exists xyz. \\
\text{eq}(w, \\
\text{eq}(\text{material}(x), \\
\text{material}(y))) \\
\text{blue_cylinder}(x) \\
\text{big_block}(y) \\
\text{red_metallic}(z) \\
\text{right_side}(y, z)
\]

[e.g. Andreas et al. 2016, Mao et al. 2019]
Module networks

Does the blue cylinder have the same material as the big block on the right side of the red metallic thing?

Yes

No need to hand-write “logical” primitives!

[e.g. Andreas et al. 2016, Mao et al. 2019]
Compositional Semantic Parsing on Semi-Structured Tables

Panupong Pasupat
Computer Science Department
Stanford University
ppasupat@cs.stanford.edu

Percy Liang
Computer Science Department
Stanford University
pliang@cs.stanford.edu

Abstract
Two important aspects of semantic parsing for question answering are the breadth of the knowledge source and the depth of logical compositionality. While existing work trades off one aspect for another, this paper simultaneously makes progress on both fronts through a new task: answering complex questions on semi-structured tables using question-answer pairs as supervision. The central challenge arises from two compounding factors: the broader domain results in an open-ended set of relations, and the deeper compositionality results in a combinatorial explosion in the space of logical forms. We propose a logical-form driven parsing algorithm guided by strong typing constraints and show that it obtains significant improvements over natural baselines. For evaluation, we created a new dataset of 22,033 complex questions on Wikipedia tables, which is made publicly available.

1 Introduction
In semantic parsing for question answering, natural language questions are converted into logical forms, which can be executed on a knowledge source to obtain answer denotations. Early semantic parsing systems were trained to answer highly compositional questions, but the knowledge sources were limited to small closed-domain databases (Zelle and Mooney, 1996; Wong and Mooney, 2007; Zettlemoyer and Collins, 2007; Kwiatkowski et al., 2011). More recent work sacrifices compositionality in favor of using more open-ended knowledge bases such as Freebase (Cai and Yates, 2013; Berant et al., 2013; Fader et al., 2014; Reddy et al., 2014). However, even these broader knowledge sources still define a rigid schema over entities and relation types, thus restricting the scope of answerable questions.

To simultaneously increase both the breadth of the knowledge source and the depth of logical compositionality, we propose a new task (with an associated dataset): answering a question using an HTML table as the knowledge source. Figure 1 shows several question-answer pairs and an accompanying table, which are typical of those in our dataset. Note that the questions are logically quite complex, involving a variety of operations such as comparison ($x_2$), superlatives ($x_3$), aggregation ($x_4$), and arithmetic ($x_5$).

The HTML tables are semi-structured and not normalized. For example, a cell might contain multiple parts (e.g., "Beijing, China" or "200 km"). Additionally, we mandate that the training and test tables are disjoint, so at test time, we will see relations (column headers; e.g., "Nations") and entities (table cells; e.g., "St. Louis").

Table: Olympic Games

<table>
<thead>
<tr>
<th>Year</th>
<th>City</th>
<th>Country</th>
<th>Nations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1896</td>
<td>Athens</td>
<td>Greece</td>
<td>14</td>
</tr>
<tr>
<td>1900</td>
<td>Paris</td>
<td>France</td>
<td>24</td>
</tr>
<tr>
<td>1904</td>
<td>St. Louis</td>
<td>USA</td>
<td>12</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>2004</td>
<td>Athens</td>
<td>Greece</td>
<td>201</td>
</tr>
<tr>
<td>2008</td>
<td>Beijing</td>
<td>China</td>
<td>204</td>
</tr>
<tr>
<td>2012</td>
<td>London</td>
<td>UK</td>
<td>204</td>
</tr>
</tbody>
</table>

Figure 1: Our task is to answer a highly compositional question from an HTML table. We learn a semantic parser from question-table-answer triples $\{(x_i, t_i, y_i)\}$.
move forward twice to the chair
\[ \lambda a.\text{move}(a) \land \text{dir}(a, \text{forward}) \land \text{len}(a, 2) \land \text{to}(a, \text{ix.chair}(x)) \]
at the corner turn left to face the blue hall
\[ \lambda a.\text{pre}(a, \text{ix.corner}(x)) \land \text{turn}(a) \land \text{dir}(a, \text{left}) \land \text{post}(a, \text{front}(you, \text{ix.blue}(x) \land \text{hall}(x))) \]
Other aspects of meaning: pragmatics

I ate some of the cookies.

Do you know what time it is?
Next class: dialogue