Learning compositional semantic theories

Christopher Potts

CS 244U: Natural language understanding
May 5
Plan

1. Review of learning to map to logical forms
2. Discussion of learning from denotations
# Related materials

## Readings

- Liang, Percy and Christopher Potts. 2014. *Bringing machine learning and compositional semantics together.*

## Code

- SEMPRE: Semantic Parsing with Execution
- UW Semantic Parsing Framework

## Data

- Geoquery, Jobsquery, Restaurant Query
- Abstract Meaning Representation Bank
- WebQuestions and Free917
- CCGBank (Penn Treebank in CCG; syntax only)
Linguistic objects

\[ \langle u, t, r, d \rangle \]

- **u**: the utterance (sequence of strings/words)
- **t**: the syntactic structure (tree structure)
- **r**: the semantic representation (a.k.a. logical form)
- **d**: the denotation (meaning)

(The denotation might under-represent or mis-represent the speaker’s intended message. We’ll return to that issue in the context of pragmatics.)
Example interpreted grammar

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Logical form</th>
<th>Denotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>N → one</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>N → two</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>R → plus</td>
<td>+</td>
<td>the $R$ such that $R(x, y) = x + y$</td>
</tr>
<tr>
<td>R → minus</td>
<td>−</td>
<td>the $R$ such that $R(x, y) = x - y$</td>
</tr>
<tr>
<td>R → times</td>
<td>$\times$</td>
<td>the $R$ such that $R(x, y) = x \times y$</td>
</tr>
<tr>
<td>S → minus</td>
<td>$\neg$</td>
<td>the $f$ such that $f(x) = -x$</td>
</tr>
<tr>
<td>N → S N</td>
<td>$\Gamma S^{-}^{-}N^{-}$</td>
<td>$\llbracket \Gamma S^{-}^{-} \rrbracket(\llbracket \Gamma N^{-} \rrbracket)$</td>
</tr>
<tr>
<td>N → N_L R N_R</td>
<td>$(\Gamma R^{-}^{-} \Gamma N_L^{-} \Gamma N_R^{-})$</td>
<td>$\llbracket \Gamma R^{-}^{-} \rrbracket(\llbracket \Gamma N_L^{-} \rrbracket, \llbracket \Gamma N_R^{-} \rrbracket)$</td>
</tr>
</tbody>
</table>

**Table:** An illustrative grammar. $\llbracket u \rrbracket$ is the translation of syntactic expression $u$, and $\llbracket r \rrbracket$ is the denotation of semantic representation $r$. $N$ is the CFG’s start symbol. In the final rule, the $L$ and $R$ subscripts are meta-annotations to ensure deterministic translation and interpretation.
## Examples

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Logical form</th>
<th>Denotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>A. seven minus five</td>
<td>((- 7 5))</td>
<td>2</td>
</tr>
<tr>
<td>B. minus three plus one</td>
<td>((+ -3 1))</td>
<td>(-2)</td>
</tr>
<tr>
<td>C. two minus two times two</td>
<td>(\times (-2 2) 2)</td>
<td>0</td>
</tr>
<tr>
<td>D. two plus three plus four</td>
<td>((+ 2 (+3 4)))</td>
<td>9</td>
</tr>
</tbody>
</table>
Examples

- $7 - 5$
- $R(x, y) = x - y$
Examples

\( N : (−7\; 5) \)

\[
\begin{align*}
N &: 7 & R &: − & N &: 5 \\
\text{seven} &: & \text{minus} &: & \text{five}
\end{align*}
\]

\( 2 \)

\[
\begin{align*}
7 & \quad \text{the } R \text{ such that} \\
R(x, y) &= x − y \\
5 &
\end{align*}
\]
Examples

$N : (- 7 5)$

$N : 7$

$R : -$  

$N : 5$

seven  

minus five

$N : (+ -3 1)$

$N : -3$

$R : +$

$N : 1$

$U : -$  

$N : 3$

plus one

minus three

$R(X, Y) = x - y$

$R(X, Y) = x + y$

$R(x, y) = -x$
Parsing and ambiguity

The grammar determines the candidate space; dynamic programming algorithms efficiently map us to that space.

\[ \text{GEN(} \text{two minus two times two} \text{)} = \]

\[ N : \llbracket (\times (- 2 2) 2) \rrbracket = 0 \]

\[ (- 2 2) \]

\[ \begin{align*}
\text{R :} & \times \\
\text{N :} & 2 \\
\text{two} & \\
\text{minus} & \\
\text{two} & \\
\end{align*} \]

\[ N : \llbracket (- 2 (\times 2 2)) \rrbracket = -2 \]

\[ \begin{align*}
\text{N :} & 2 \\
\text{R :} & - \\
\text{two} & \\
\text{minus} & \\
(\times 2 2) & \\
\end{align*} \]

\[ \begin{align*}
\text{N :} & 2 \\
\text{R :} & \times \\
\text{two} & \\
\text{times} & \\
\text{two} & \\
\end{align*} \]
Direct implementations

- Prominent recent examples: Bos 2005; Bos and Markert 2005
- Excel at inference (via theorem provers).
- Tend to be high precision, low recall — the analyst must anticipate every lexical item and every constructional quirk.

Figure: Prolog representation from Bos 2005: *Mubarak reviewed the blueprints for a number of other huge national projects, known as Egypt's 21st century project.*
Compositionality

The meaning of a phrase is a function of the meanings of its immediate syntactic constituents and the way they are combined.
Compositionality

The meaning of a phrase is a function of the meanings of its immediate syntactic constituents and the way they are combined.

Liang and Potts (2014)

“the claim of compositionality is that being a semantic interpreter for a language $L$ amounts to mastering the syntax of $L$, the lexical meanings of $L$, and the modes of semantic combination for $L$. This also suggests the outlines of a learning task.”
## Learning tasks

The grammar frames the task; different parts of it can be learned.

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Logical form</th>
<th>Denotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>N → one</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>N → two</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>R → plus</td>
<td>+</td>
<td>the $R$ such that $R(x, y) = x + y$</td>
</tr>
<tr>
<td>R → minus</td>
<td>−</td>
<td>the $R$ such that $R(x, y) = x - y$</td>
</tr>
<tr>
<td>R → times</td>
<td>×</td>
<td>the $R$ such that $R(x, y) = x \times y$</td>
</tr>
<tr>
<td>S → minus</td>
<td>−</td>
<td>the $f$ such that $f(x) = -x$</td>
</tr>
<tr>
<td>N → S N</td>
<td>$\text{⌜S⌝⌜N⌝}$</td>
<td>$\llbracket\text{⌜S⌝}\rrbracket(\llbracket\text{⌜N⌝}\rrbracket)$</td>
</tr>
<tr>
<td>N → N_L R N_R</td>
<td>$\langle\text{⌜R⌝⌜N_L⌝⌜N_R⌝}\rangle$</td>
<td>$\llbracket\text{⌜R⌝}\rrbracket(\llbracket\text{⌜N_L⌝}\rrbracket, \llbracket\text{⌜N_R⌝}\rrbracket)$</td>
</tr>
</tbody>
</table>

- Parsing
- Semantic parsing
- Interpretive
Learning tasks

The grammar frames the task; different parts of it can be learned.

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Logical form</th>
<th>Denotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>N → one</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>N → two</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>R → plus</td>
<td>+</td>
<td>the $R$ such that $R(x, y) = x + y$</td>
</tr>
<tr>
<td>R → minus</td>
<td>−</td>
<td>the $R$ such that $R(x, y) = x - y$</td>
</tr>
<tr>
<td>R → times</td>
<td>×</td>
<td>the $R$ such that $R(x, y) = x \times y$</td>
</tr>
<tr>
<td>S → minus</td>
<td>−</td>
<td>the $f$ such that $f(x) = -x$</td>
</tr>
<tr>
<td>N → S N</td>
<td>$\left[\left[ S \right]\left[ N \right]\right]$</td>
<td>$\left[\left[ S \right]\left[ N \right]\right]$</td>
</tr>
<tr>
<td>N → N_L R N_R</td>
<td>$\left[\left[ R \right]\left[ N_L \right]\left[ N_R \right]\right]$</td>
<td>$\left[\left[ R \right]\left[ N_L \right]\right], \left[\left[ N_R \right]\right]$</td>
</tr>
</tbody>
</table>

- Parsing
- Semantic parsing
- Interpretive
Learning tasks

The grammar frames the task; different parts of it can be learned.

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Logical form</th>
<th>Denotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>N → one</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>N → two</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>R → plus</td>
<td>+</td>
<td>the $R$ such that $R(x, y) = x + y$</td>
</tr>
<tr>
<td>R → minus</td>
<td>−</td>
<td>the $R$ such that $R(x, y) = x − y$</td>
</tr>
<tr>
<td>R → times</td>
<td>×</td>
<td>the $R$ such that $R(x, y) = x * y$</td>
</tr>
<tr>
<td>S → minus</td>
<td>−</td>
<td>the $f$ such that $f(x) = −x$</td>
</tr>
<tr>
<td>N → S N</td>
<td>$S \triangleright N$</td>
<td>$\llbracket S \triangleright \llbracket N \rrbracket \rrbracket$</td>
</tr>
<tr>
<td>N → N_L R N_R</td>
<td>($R \triangleright N_L \triangleright N_R$)</td>
<td>$\llbracket R \triangleright \llbracket N_L \rrbracket, \llbracket N_R \rrbracket \rrbracket$</td>
</tr>
</tbody>
</table>

- Parsing
- Semantic parsing
- Interpretive
### Learning tasks

The grammar frames the task; different parts of it can be learned.

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Logical form</th>
<th>Denotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>N → one</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>N → two</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>R → plus</td>
<td>+</td>
<td>the R such that $R(x, y) = x + y$</td>
</tr>
<tr>
<td>R → minus</td>
<td>−</td>
<td>the R such that $R(x, y) = x - y$</td>
</tr>
<tr>
<td>R → times</td>
<td>×</td>
<td>the R such that $R(x, y) = x \times y$</td>
</tr>
<tr>
<td>S → minus</td>
<td>−</td>
<td>the f such that $f(x) = -x$</td>
</tr>
<tr>
<td>N → S N</td>
<td>⌜S⌝⌜N⌝ ⌜⌜N⌝</td>
<td>$<a href="%5B%5B%5BN%5D%5D%5D">[S]</a>$</td>
</tr>
<tr>
<td>N → N_L R N_R</td>
<td>(⌜R⌝⌜N_L⌝⌜N_R⌝)</td>
<td>$[[R]]([[[N_L]], [[[N_R]]])$</td>
</tr>
</tbody>
</table>

- Parsing
- Semantic parsing
- Interpretive
Learning tasks

The grammar frames the task; different parts of it can be learned.

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Logical form</th>
<th>Denotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>N → one</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>N → two</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>R → plus</td>
<td>+</td>
<td>the $R$ such that $R(x, y) = x + y$</td>
</tr>
<tr>
<td>R → minus</td>
<td>−</td>
<td>the $R$ such that $R(x, y) = x - y$</td>
</tr>
<tr>
<td>R → times</td>
<td>×</td>
<td>the $R$ such that $R(x, y) = x * y$</td>
</tr>
<tr>
<td>S → minus</td>
<td>−</td>
<td>the $f$ such that $f(x) = -x$</td>
</tr>
<tr>
<td>N → S N</td>
<td>$\llbracket S \rrbracket \llbracket N \rrbracket$</td>
<td>$\llbracket S \rrbracket (\llbracket N \rrbracket)$</td>
</tr>
<tr>
<td>N → N_L R N_R</td>
<td>$(\llbracket R \rrbracket \llbracket N_L \rrbracket \llbracket N_R \rrbracket)$</td>
<td>$\llbracket R \rrbracket (\llbracket N_L \rrbracket, \llbracket N_R \rrbracket)$</td>
</tr>
</tbody>
</table>

- Parsing
- Semantic parsing
- Interpretive
## Learning tasks

The grammar frames the task; different parts of it can be learned.

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Logical form</th>
<th>Denotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>N → one</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>N → two</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>R → plus</td>
<td>+</td>
<td>the $R$ such that $R(x, y) = x + y$</td>
</tr>
<tr>
<td>R → minus</td>
<td>−</td>
<td>the $R$ such that $R(x, y) = x - y$</td>
</tr>
<tr>
<td>R → times</td>
<td>$\times$</td>
<td>the $R$ such that $R(x, y) = x \times y$</td>
</tr>
<tr>
<td>S → minus</td>
<td>$\neg$</td>
<td>the $f$ such that $f(x) = -x$</td>
</tr>
<tr>
<td>N → S N</td>
<td>$\tilde{\llbracket S \rrbracket} \tilde{\llbracket N \rrbracket}$</td>
<td>$\llbracket \tilde{S} \rrbracket(\llbracket \tilde{N} \rrbracket)$</td>
</tr>
<tr>
<td>N → N&lt;sub&gt;L&lt;/sub&gt; R N&lt;sub&gt;R&lt;/sub&gt;</td>
<td>$(\tilde{\llbracket R \rrbracket} \llbracket \tilde{N}_L \rrbracket \llbracket \tilde{N}_R \rrbracket)$</td>
<td>$\llbracket \tilde{R} \rrbracket(\llbracket \tilde{N}_L \rrbracket, \llbracket \tilde{N}_R \rrbracket)$</td>
</tr>
</tbody>
</table>

- Parsing
- Semantic parsing
- Interpretive
Semantic parsing

\[ \langle u, t, r, d \rangle \]

**Pioneering work**

- Statistical: Zelle and Mooney 1996; Tang and Mooney 2001; Thompson and Mooney 2003; Zettlemoyer and Collins 2005
Basic formulation

<table>
<thead>
<tr>
<th>Utterance</th>
<th>Logical form</th>
</tr>
</thead>
<tbody>
<tr>
<td>seven minus five</td>
<td>(− 7 5)</td>
</tr>
<tr>
<td>five minus seven</td>
<td>(− 5 7)</td>
</tr>
<tr>
<td>three plus one</td>
<td>(− 7 5)</td>
</tr>
<tr>
<td>minus three plus one</td>
<td>(+ −3 1)</td>
</tr>
<tr>
<td>minus three plus one</td>
<td>−(+ 3 1)</td>
</tr>
<tr>
<td>two minus two times two</td>
<td>(× (− 2 2) 2)</td>
</tr>
<tr>
<td>two minus two times two</td>
<td>(− 2 (× 2 2))</td>
</tr>
<tr>
<td>two plus three plus four</td>
<td>(+ 2 (+ 3 4))</td>
</tr>
<tr>
<td>three minus one</td>
<td>?</td>
</tr>
<tr>
<td>three times one</td>
<td>?</td>
</tr>
<tr>
<td>minus six times four</td>
<td>?</td>
</tr>
<tr>
<td>one plus three plus five</td>
<td>?</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table:** Data requirements.

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Logical form</th>
</tr>
</thead>
<tbody>
<tr>
<td>N → one</td>
<td>1</td>
</tr>
<tr>
<td>N → one</td>
<td>2</td>
</tr>
<tr>
<td>N → two</td>
<td>1</td>
</tr>
<tr>
<td>N → two</td>
<td>2</td>
</tr>
<tr>
<td>R → plus</td>
<td>+</td>
</tr>
<tr>
<td>R → plus</td>
<td>−</td>
</tr>
<tr>
<td>R → plus</td>
<td>×</td>
</tr>
<tr>
<td>R → minus</td>
<td>+</td>
</tr>
<tr>
<td>R → minus</td>
<td>−</td>
</tr>
<tr>
<td>R → minus</td>
<td>×</td>
</tr>
<tr>
<td>R → times</td>
<td>+</td>
</tr>
<tr>
<td>R → times</td>
<td>−</td>
</tr>
<tr>
<td>R → times</td>
<td>×</td>
</tr>
<tr>
<td>S → minus</td>
<td>¬</td>
</tr>
</tbody>
</table>

N → S N

N → N_L R N_R

| Table: Crude grammar. |
Learning framework

1. Feature representations: $\phi(x, y) \in \mathbb{R}^d$
2. Scoring: $\text{Score}_w(x, y) = w \cdot \phi(x, y) = \sum_{j=1}^d w_j \phi(x, y)_j$
3. Multiclass hinge-loss objective function:

$$\min_{w \in \mathbb{R}^d} \sum_{(x, y) \in \mathcal{D}} \max_{y' \in \text{GEN}(x)} \left[ \text{Score}_w(x, y') + c(y, y') \right] - \text{Score}_w(x, y)$$

where $\mathcal{D}$ is a set of $(x, y)$ training examples and $c(a, b) = 1$ if $a \neq b$, else 0.

4. Optimization:

$\text{STOCHASTICGRADIENTDESCENT}(\mathcal{D}, T, \eta)$

1. Initialize $w \leftarrow 0$
2. Repeat $T$ times
3. for each $(x, y) \in \mathcal{D}$ (in random order)
4. $\tilde{y} \leftarrow \arg \max_{y' \in \text{GEN}(x)} \text{Score}_w(x, y') + c(y, y')$
5. $w \leftarrow w + \eta(\phi(x, y) - \phi(x, \tilde{y}))$
6. Return $w$
Example

(a) **Candidates** \(\text{GEN}(x)\) for utterance \(x = \text{two times two plus three}\)

\[
\begin{align*}
\phi(x, y_1) &= \begin{cases}
\text{R:\{times\}} : 1 \\
\text{R:\{plus\}} : 1 \\
\text{top[R:\{+\}]} : 1
\end{cases} \\
\phi(x, y_2) &= \begin{cases}
\text{R:\{times\}} : 1 \\
\text{R:\{plus\}} : 1 \\
\text{top[R:\{+\}]} : 1
\end{cases} \\
\phi(x, y_3) &= \begin{cases}
\text{R:\{times\}} : 1 \\
\text{R:\{plus\}} : 1 \\
\text{top[R:\{+\}]} : 1
\end{cases}
\end{align*}
\]

(b) **Learning from logical forms** (Section 4.1)

<table>
<thead>
<tr>
<th>Iteration 1</th>
<th>Iteration 2</th>
<th>Iteration 3</th>
</tr>
</thead>
</table>
| \(w = \begin{cases}
\text{R:\{times\}} : 0 \\
\text{R:\{plus\}} : 0 \\
\text{R:\{\{+\}} : 0 \\
\text{top[R:\{+\}]} : 0 \\
\text{top[R:\{times\}]} : 0
\end{cases}\) Scores: \([0, 0, 0]\) | \(w = \begin{cases}
\text{R:\{times\}} : 0 \\
\text{R:\{plus\}} : 0 \\
\text{R:\{\{+\}} : 0 \\
\text{top[R:\{+\}]} : 1 \\
\text{top[R:\{times\}]} : -1
\end{cases}\) Scores: \([1, 1, -1]\) | \(w = \begin{cases}
\text{R:\{times\}} : 1 \\
\text{R:\{plus\}} : -1 \\
\text{R:\{\{+\}} : 0 \\
\text{top[R:\{+\}]} : 1 \\
\text{top[R:\{times}]} : -1
\end{cases}\) Scores: \([2, 0, 0]\) |
| \(y = y_1\) \(\tilde{y} = y_3\) (tied with \(y_2\)) | \(y = y_1\) \(\tilde{y} = y_2\) (tied with \(y_1\)) | \(y = y_1\) \(\tilde{y} = y_1\) |
Derivational ambiguity

In the rich grammars of Zettlemoyer and Collins (2005, 2007) and others, a given logical expression might have multiple derivations.

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Logical form</th>
</tr>
</thead>
<tbody>
<tr>
<td>N → one</td>
<td>1</td>
</tr>
<tr>
<td>N → two</td>
<td>2</td>
</tr>
<tr>
<td>R → plus</td>
<td>+</td>
</tr>
<tr>
<td>R → minus</td>
<td>−</td>
</tr>
<tr>
<td>R → times</td>
<td>×</td>
</tr>
<tr>
<td>S → minus</td>
<td>−</td>
</tr>
<tr>
<td>N → S N</td>
<td>ㄱSㄱNㄱ</td>
</tr>
<tr>
<td>N → N_L R N_R</td>
<td>(ㄱRㄱ ㄱN_Lㄱ ㄱN_Rㄱ)</td>
</tr>
<tr>
<td>Q → n</td>
<td>(λf (f ㄱnㄱ))</td>
</tr>
<tr>
<td>N → U Q</td>
<td>(ㄱQㄱ ㄱUㄱ)</td>
</tr>
</tbody>
</table>

Table: Grammar with type-lifting.

Training instance: (minus three, −3)

\[
\begin{align*}
N & : \neg 3 \\
U & : \neg \\
N & : 3 \\
Q & : (\lambda f (f 3)) \\
N & : ((\lambda f (f 3)) \neg) \Rightarrow \neg 3 \\
U & : \neg \\
Q & : (\lambda f (f 3)) \\
\end{align*}
\]

(Beta-conversion $\Rightarrow$ is the syntactic counterpart of functional application.)
Derivations as latent variables

- The training instances are \((u, r)\) pairs.
- Since \(r\) might have multiple derivations, derivations are latent variables.
- Zettlemoyer and Collins (2005, 2007) use log-linear latent variable models, but our earlier framework can accommodate them as well.
- Latent support vector machine objective:

\[
\min_{w \in \mathbb{R}^d} \sum_{(x,r) \in \mathcal{D}} \max_{y' \in \text{Gen}(x)} \left[ \text{Score}_w(x, y') + c(r, \text{Root}(y')) \right] - \max_{y'' \in \text{Gen}(x,r)} \text{Score}_w(x, y''),
\]

where \(\mathcal{D}\) is a set of (utterance, formula) pairs; \(c(a, b) = 1\) if \(a \neq b\), else 0; and \(\text{Gen}(x, r) = \{ y \in \text{Gen}(x) : \text{Root}(y) = r \}\)

- Optimization:

\text{STOCHASTICGRADIENTDESCENT}(\mathcal{D}, T, \eta)

1. Initialize \(w \leftarrow 0\)
2. Repeat \(T\) times
3. for each \((x, r) \in \mathcal{D}\) (in random order)
4. \(y \leftarrow \arg \max_{y'' \in \text{Gen}(x,r)} \text{Score}_w(x, y'')\)
5. \(\tilde{y} \leftarrow \arg \max_{y' \in \text{Gen}(x)} \text{Score}_w(x, y') + c(y, y')\)
6. \(w \leftarrow w + \eta(\phi(x, y) - \phi(x, \tilde{y}))\)
7. Return \(w\)
Taming the search space

The complexity issues trace to the fact that the size of $\text{GEN}(x)$ is exponential in the length of $x$.

- Variants of CKY parsing algorithms that track both syntactic and semantic information (Zettlemoyer 2009:Appendix A).
- Assume parts of the lexicon are known (function words, easily specified open-class items).
- Prune the lexicon during training, thereby keeping it small, thereby keeping $\text{GEN}(x)$ small (Zettlemoyer and Collins 2005).
High-level look at results

<table>
<thead>
<tr>
<th>Paper</th>
<th>Recall (LFs)</th>
<th>Recall (Answers)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zettlemoyer and Collins (2005)</td>
<td>79.3</td>
<td>–</td>
</tr>
<tr>
<td>Zettlemoyer and Collins (2007)</td>
<td>81.6</td>
<td>–</td>
</tr>
<tr>
<td>Kwiatkowski et al. (2010)</td>
<td>88.2</td>
<td>–</td>
</tr>
<tr>
<td>Kwiatkowski et al. (2011)</td>
<td>88.6</td>
<td>–</td>
</tr>
</tbody>
</table>

Table: Results for the Geo880 test set (Zelle and Mooney 1996). For a fuller summary, see Liang et al. 2013:435.
Recent developments and extensions

- Zettlemoyer and Collins (2007): grappling with messy data (ATIS travel-planning)
- Artzi and Zettlemoyer (2011): bootstrapping from machine-generated dialog systems
- Kwiatkowski et al. (2010): learning (weights on) the modes of composition
- Matuszek et al. (2012b): mapping to a robot controller language
- Kwiatkowski et al. (2010); Kwiatkowski et al. (2011): multilingual semantic parsing
- Cai and Yates (2013): question-answering with Freebase
Learning from denotations

\[ \langle u, t, r, d \rangle \]

Pioneering work

- Psychological: see Frank et al. 2009 for models and references
- NLP: Clarke et al. (2010); Liang et al. (2011, 2013)
Motivations

**Detailed Supervision**
- doesn’t scale up
- representation-dependent

What is the largest city in California?

```latex
\arg\max\{c : \text{city}(c) \land \text{loc}(c, \text{CA})\}, \text{population}
```

**Natural Supervision**
- scales up
- representation-independent

What is the largest city in California?

```
Los Angeles
```

(Slide from Percy Liang)
Basic formulation

<table>
<thead>
<tr>
<th>Utterance</th>
<th>Denotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>seven minus five</td>
<td>2</td>
</tr>
<tr>
<td>five minus seven</td>
<td>-2</td>
</tr>
<tr>
<td>three plus one</td>
<td>4</td>
</tr>
<tr>
<td>minus three plus one</td>
<td>-2</td>
</tr>
<tr>
<td>two minus two times two</td>
<td>0</td>
</tr>
<tr>
<td>two minus two times two</td>
<td>-2</td>
</tr>
<tr>
<td>two plus three plus four</td>
<td>9</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>three minus one</td>
<td>?</td>
</tr>
<tr>
<td>three times one</td>
<td>?</td>
</tr>
<tr>
<td>minus six times four</td>
<td>?</td>
</tr>
<tr>
<td>one plus three plus five</td>
<td>?</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

Table: Data requirements.

Syntax | Logical form | Denotation |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>N → one</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>N → one</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>N → two</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>N → two</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>R → plus</td>
<td>+</td>
<td>addition</td>
</tr>
<tr>
<td>R → plus</td>
<td>−</td>
<td>subtraction</td>
</tr>
<tr>
<td>R → plus</td>
<td>×</td>
<td>multiplication</td>
</tr>
<tr>
<td>R → minus</td>
<td>+</td>
<td>addition</td>
</tr>
<tr>
<td>R → minus</td>
<td>−</td>
<td>subtraction</td>
</tr>
<tr>
<td>R → minus</td>
<td>×</td>
<td>multiplication</td>
</tr>
<tr>
<td>R → times</td>
<td>+</td>
<td>addition</td>
</tr>
<tr>
<td>R → times</td>
<td>−</td>
<td>subtraction</td>
</tr>
<tr>
<td>R → times</td>
<td>×</td>
<td>multiplication</td>
</tr>
<tr>
<td>S → minus</td>
<td>−</td>
<td>negative</td>
</tr>
<tr>
<td>N → S N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>N → N_L R R_N_R</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table: Crude grammar.
Learning framework

1. Feature representations: \( \phi(x, y) \in \mathbb{R}^d \)

2. Scoring: \( \text{Score}_w(x, y) = w \cdot \phi(x, y) = \sum_{j=1}^{d} w_j \phi_j(x, y) \)

3. Latent support vector machine objective:

   \[
   \min_{w \in \mathbb{R}^d} \sum_{(x, d) \in \mathcal{D}} \max_{y' \in \text{Gen}(x)} \left[ \text{Score}_w(x, y') + c(d, \llbracket y' \rrbracket) \right] - \max_{y \in \text{Gen}(x, d)} \text{Score}_w(x, y),
   \]

   where \( \text{Gen}(x, d) = \{ y \in \text{Gen}(x) : \llbracket y \rrbracket = d \} \) is the set of logical forms that evaluate to denotation \( d \).

4. Optimization:

   \text{STOCHASTICGRADIENTDESCENT}(\mathcal{D}, T, \eta)

   1. Initialize \( w \leftarrow 0 \)
   2. Repeat \( T \) times
   3. for each \( (x, d) \in \mathcal{D} \) (in random order)
   4. \( y \leftarrow \arg \max_{y' \in \text{Gen}(x, d)} \text{Score}_w(x, y') \)
   5. \( \tilde{y} \leftarrow \arg \max_{y' \in \text{Gen}(x)} \text{Score}_w(x, y') + c(y, y') \)
   6. \( w \leftarrow w + \eta(\phi(x, y) - \phi(x, \tilde{y})) \)
   7. Return \( w \)
Example

(a) Candidates \( \text{GEN}(x) \) for utterance \( x = \text{two times two plus three} \)

\[
\begin{align*}
\text{y}_1 & : N: (+ (\times 2 2) 3) \Rightarrow 7 \\
\text{y}_2 & : N: (+ 2 2) 3 \Rightarrow 7 \\
\text{y}_3 & : N: (\times 2 (+ 2 3)) \Rightarrow 10
\end{align*}
\]

\[
\begin{align*}
\phi(x, y_1) &= R:\times[\text{times}]: 1 \\
& \quad R:+[\text{plus}]: 1 \\
& \quad \text{top}[R:+]: 1
\end{align*}
\]

\[
\begin{align*}
\phi(x, y_2) &= R:+[\text{times}]: 1 \\
& \quad \text{top}[R:+]: 1
\end{align*}
\]

\[
\begin{align*}
\phi(x, y_3) &= R:\times[\text{times}]: 1 \\
& \quad R:+[\text{plus}]: 1 \\
& \quad \text{top}[R: \times]: 1
\end{align*}
\]

(c) Learning from denotations (Section 4.2)

\[
\begin{align*}
\text{Iteration 1} \\
\mathbf{w} &= \begin{bmatrix} R:\times[\text{times}]: 0 \\
R:+[\text{times}]: 0 \\
R:+[\text{plus}]: 0 \\
\text{top}[R:+]: 0 \\
\text{top}[R: \times]: 0 \\
\end{bmatrix} \\
& \quad \text{Scores: } [0, 0, 0] \\
& \quad \text{GEN}(x, d) = \{y_1, y_2\} \\
& \quad y = y_1 \text{ (tied with } y_2) \\
& \quad \tilde{y} = y_3 \text{ (tied with } y_2) \\
\Rightarrow \quad \text{Iteration 2} \\
\mathbf{w} &= \begin{bmatrix} R:\times[\text{times}]: 0 \\
R:+[\text{times}]: 0 \\
R:+[\text{plus}]: 0 \\
\text{top}[R:+]: 1 \\
\text{top}[R: \times]: -1 \\
\end{bmatrix} \\
& \quad \text{Scores: } [1, 1, -1] \\
& \quad \text{GEN}(x, d) = \{y_1, y_2\} \\
& \quad y = y_1 \text{ (tied with } y_2) \\
& \quad \tilde{y} = y_1 \text{ (tied with } y_2)
\end{align*}
\]

Not pictured: possibility of features on denotations!
Probabilistic formulation

Semantic Parsing: \( p(z \mid x, \theta) \)
(probabilistic)

Interpretation: \( p(y \mid z, w) \)
(deterministic)

(Slide from Percy Liang)
EM-style learning

Objective Function:

\[ p(y | z, w) p(z | x, \theta) \]

Interpretation: Semantic parsing

(Slide from Percy Liang)
EM-style learning

Objective Function:

$$\max_{\theta} \quad p(y \mid z, w) \ p(z \mid x, \theta)$$

Interpretation   Semantic parsing

(Slide from Percy Liang)
EM-style learning

Objective Function:

$$\max_{\theta} \sum_z p(y \mid z, w) p(z \mid x, \theta)$$

Interpretation: Semantic parsing

(Slide from Percy Liang)
EM-style learning

Objective Function:

$$\max_{\theta} \sum_z p(y \mid z, w) p(z \mid x, \theta)$$

Interpretation  Semantic parsing

EM-like Algorithm:

parameters $\theta$

$(0, 0, \ldots, 0)$

(Slide from Percy Liang)
EM-style learning

Objective Function:

\[ \max_\theta \sum_z p(y \mid z, w) p(z \mid x, \theta) \]

Interpretation  Semantic parsing

EM-like Algorithm:

parameters \( \theta \)

enumerate/score DCS trees

(0, 0, \ldots, 0)

(Slide from Percy Liang)
EM-style learning

Objective Function:
\[
\max_{\theta} \sum_z p(y \mid z, w) p(z \mid x, \theta)
\]

Interpretation  Semantic parsing

EM-like Algorithm:
parameters $\theta$

\[ (0, 0, \ldots, 0) \]

enumerate/score DCS trees

$k$-best list

tree1 $\times$
tree2 $\times$
tree3 $\checkmark$
tree4 $\times$
tree5 $\times$

(Slide from Percy Liang)
EM-style learning

Objective Function:
\[
\max_{\theta} \sum_z p(y | z, w) p(z | x, \theta)
\]

Interpretation Semantic parsing

EM-like Algorithm:
- parameters $\theta$
- $(0.2, -1.3, \ldots, 0.7)$
- enumerate/score DCS trees
- numerical optimization (L-BFGS)

$k$-best list
- tree1
- tree2
- tree3
- tree4
- tree5

(Slide from Percy Liang)
EM-style learning

Objective Function:

\[
\max_{\theta} \sum_z p(y \mid z, w) p(z \mid x, \theta)
\]

Interpretation: Semantic parsing

EM-like Algorithm:

- Parameters \( \theta \)
- \((0.2, -1.3, \ldots, 0.7)\)
- Enumerate/score DCS trees
- Numerical optimization (L-BFGS)

\(k\)-best list:
- tree3 ✓
- tree8 ✓
- tree6 ✗
- tree2 ✗
- tree4 ✗

(Slide from Percy Liang)
EM-style learning

Objective Function:

$$\max_{\theta} \sum_x p(y | z, w) \cdot p(z | x, \theta)$$

Interpretation: Semantic parsing

EM-like Algorithm:

- parameters $\theta$
- enumerate/score DCS trees
- numerical optimization (L-BFGS)
- $k$-best list
  - tree3
  - tree8
  - tree6
  - tree2
  - tree4

(Slide from Percy Liang)
**EM-style learning**

**Objective Function:**

$$\max_{\theta} \sum_z p(y \mid z, w) p(z \mid x, \theta)$$

**Interpretation**

Semantic parsing

**EM-like Algorithm:**

- Parameters $\theta$
- $k$-best list
- Enumerate/score DCS trees
- Numerical optimization (L-BFGS)

(Slide from Percy Liang)
Basic Dependency-based Compositional Semantics (DCS)

A sub-logic of the full version in Liang et al. 2013:§2.5:

$$[[P_n]] = \{ \langle x_1, \ldots, x_n \rangle, \ldots \}$$

$$[[a \ i \ j \ b]] = \{ x \in [[a]] : x_i = y_j \text{ for some } y \in [[b]] \}$$

$$[[a \ i \ j \ b \ k \ p \ c]] = \{ x \in [[a]] : x_i = y_j \text{ for some } y \in [[b]] \} \cap \{ x \in [[a]] : x_k = z_p \text{ for some } z \in [[b]] \}$$

$$\sum_{b} = \{ [[b]] \}$$
Basic DCS examples

\[ [\text{lisa}] = \{ \text{Lisa} \} \]

\[ [\text{admire}] = \{ \{ \text{lisa}, \text{boy} \}, \{ \text{lisa}, \text{boy} \}, \{ \text{lisa}, \text{boy} \} \} \]

\[
\begin{array}{c}
\text{admire} \quad 1 \quad 1 \\
\text{lisa}
\end{array}
\]

\[ = \{ x \in [\text{admire}] : x_1 = y_1 \text{ for some } y \in \text{Lisa} \} \]

\[ = \{ \{ \text{lisa}, \text{boy} \}, \{ \text{lisa}, \text{boy} \}, \{ \text{lisa}, \text{boy} \} \} \]

\[
\begin{array}{c}
\text{admire} \quad 2 \quad 1 \\
\text{lisa}
\end{array}
\]

\[ = \{ x \in [\text{admire}] : x_2 = y_1 \text{ for some } y \in \text{Lisa} \} \]

\[ = \{ \{ \text{lisa}, \text{boy} \}, \{ \text{lisa}, \text{boy} \}, \{ \text{lisa}, \text{boy} \} \} \]

\[
\begin{array}{c}
\text{admire} \quad 2 \quad 1 \\
\text{lisa}
\end{array}
\]

\[ = \{ x \in [\text{admire}] : x_2 = y_1, y \in [\text{lisa}] \cap \{ x \in [\text{admire}] : x_1 = z_1, z \in [\text{boy}] \} \]

\[ = \{ \{ \text{lisa}, \text{boy} \}, \{ \text{lisa}, \text{boy} \}, \{ \text{lisa}, \text{boy} \} \} \cap \{ \{ \text{lisa}, \text{boy} \}, \{ \text{lisa}, \text{boy} \}, \{ \text{lisa}, \text{boy} \} \} \]
DCS, mark/execute, and scope ambiguity

Some river traverses every city.

**Figure 15**
Denotation of Figure 8(c) before the execute relation is applied.

- Execute $x_{12}$ processes column 3, then column 2: wide-scope some river
- Execute $x_{21}$ processes column 2, then column 3: wide-scope every city

See also Percy’s slides from last year.
### Lambda DCS (Liang 2013)

<table>
<thead>
<tr>
<th>Lambda DCS</th>
<th>Lambda DCS type</th>
<th>Lambda expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>e</td>
<td>$\lambda x \ (x = a)$</td>
</tr>
<tr>
<td>R</td>
<td>$\langle e, \langle e, t \rangle \rangle$</td>
<td>$\lambda x \ (\lambda y \ R(x, y))$</td>
</tr>
<tr>
<td>R . a</td>
<td>$\langle e, t \rangle$</td>
<td>$\lambda x \ \exists y \ (R(x, y) \land a(y))$</td>
</tr>
<tr>
<td>P △ Q</td>
<td>$\langle e, t \rangle$</td>
<td>$\lambda x \ (P(x) \land Q(x))$</td>
</tr>
<tr>
<td>P ▽ Q</td>
<td>$\langle e, t \rangle$</td>
<td>$\lambda x \ (P(x) \lor Q(x))$</td>
</tr>
<tr>
<td>¬P</td>
<td>$\langle e, t \rangle$</td>
<td>$\lambda x \ \neg P(x)$</td>
</tr>
<tr>
<td>$\mu x \ (R.S.x)$</td>
<td>$\langle e, t \rangle$</td>
<td>$\lambda x \ \exists y \ (R(x, y) \land S(y, x))$</td>
</tr>
</tbody>
</table>

Table: Language definition.
### Lambda DCS (Liang 2013)

<table>
<thead>
<tr>
<th>Lambda DCS</th>
<th>Lambda DCS type</th>
<th>Lambda expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>e</td>
<td>$\lambda x \ (x = a)$</td>
</tr>
<tr>
<td>R</td>
<td>$\langle e, \langle e, t \rangle \rangle$</td>
<td>$\lambda x \ (\lambda y \ R(x, y))$</td>
</tr>
<tr>
<td>R.a</td>
<td>$\langle e, t \rangle$</td>
<td>$\lambda x \ \exists y \ (R(x, y) \land a(y))$</td>
</tr>
<tr>
<td>P $\sqcap$ Q</td>
<td>$\langle e, t \rangle$</td>
<td>$\lambda x \ (P(x) \land Q(x))$</td>
</tr>
<tr>
<td>P $\sqcup$ Q</td>
<td>$\langle e, t \rangle$</td>
<td>$\lambda x \ (P(x) \lor Q(x))$</td>
</tr>
<tr>
<td>$\neg P$</td>
<td>$\langle e, t \rangle$</td>
<td>$\lambda x \ \neg P(x)$</td>
</tr>
<tr>
<td>$\mu x \ (R.S.x)$</td>
<td>$\langle e, t \rangle$</td>
<td>$\lambda x \ \exists y \ (R(x, y) \land S(y, x))$</td>
</tr>
</tbody>
</table>

**Table**: Language definition.

<table>
<thead>
<tr>
<th>Lambda DCS</th>
<th>Lambda expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>peru</td>
<td>$\lambda x \ (x = \text{peru})$</td>
</tr>
<tr>
<td>Birthplace</td>
<td>$\lambda x \ (\lambda y \ \text{Birthplace}(x, y))$</td>
</tr>
<tr>
<td>Birthplace.peru</td>
<td>$\lambda x \ \exists y \ (\text{Birthplace}(x, y) \land \text{peru}(y))$</td>
</tr>
<tr>
<td>Birthplace.peru $\sqcap$ Linguist</td>
<td>$\lambda x \ (\text{Birthplace.peru}(x) \land \text{Linguist}(x))$</td>
</tr>
<tr>
<td>$\mu x \ (\text{Student.Influenced}.x)$</td>
<td>$\lambda x \ \exists y \ (\text{Student}(x, y) \land \text{Influenced}(y, x))$</td>
</tr>
</tbody>
</table>

**Table**: Examples.
### High-level look at results

<table>
<thead>
<tr>
<th>Paper</th>
<th>Recall (LFs)</th>
<th>Recall (Answers)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zettlemoyer and Collins (2005)</td>
<td>79.3</td>
<td>–</td>
</tr>
<tr>
<td>Zettlemoyer and Collins (2007)</td>
<td>81.6</td>
<td>–</td>
</tr>
<tr>
<td>Kwiatkowski et al. (2010)</td>
<td>88.2</td>
<td>–</td>
</tr>
<tr>
<td>Kwiatkowski et al. (2011)</td>
<td>88.6</td>
<td>–</td>
</tr>
<tr>
<td>Liang et al. (2011, 2013)</td>
<td>–</td>
<td>87.9</td>
</tr>
<tr>
<td>Liang et al. (2011, 2013) with $L^+$</td>
<td>–</td>
<td>91.4</td>
</tr>
</tbody>
</table>

**Table:** Results for the Geo880 test set (Zelle and Mooney 1996). For a fuller summary, see Liang et al. 2013:435. ‘$L^+$’ here involves 22 pre-specified training instances for semantically complex predicates like size.
Recent developments and extensions

- **Learning from large databases**: Clarke et al. 2010; Berant et al. 2013; Berant and Liang 2014; Kwiatkowski et al. 2013.

- **Computer programming tasks**: Kushman and Barzilay 2013; Lei et al. 2013

- **Computer games**: Branavan et al. 2010, 2011

- **Learning via perception**: Matuszek et al. 2012a; Tellex et al. 2011; Krishnamurthy and Kollar 2013
References


References II


References III