Lecture 14: Compositional Semantics

Luke Zettlemoyer - University of Washington

[Most slides from Dan Klein]
Topics / Announcements

- **Today**
  - compositional semantics
  - learning to parse to meaning

- **After that**
  - discourse
  - machine translation

- **Only two weeks left!**
  - don’t leave the projects to the last minute
Semantic Interpretation

- **Back to meaning!**
  - A very basic approach to computational semantics
  - Truth-theoretic notion of semantics (Tarskian)
  - Assign a “meaning” to each word
  - Word meanings combine according to the parse structure
  - People can and do spend entire courses on this topic
  - We’ll spend about an hour!

- **What’s NLP and what isn’t?**
  - Designing meaning representations?
  - Computing those representations?
  - Reasoning with them?

- **Supplemental reading on the web page.**
Semantics

“Meaning”
- What is meaning?
  - “The computer in the corner.”
  - “Bob likes Alice.”
  - “I think I am a gummi bear.”
- Knowing whether a statement is true?
- Knowing the conditions under which it’s true?
- Being able to react appropriately to it?
  - “Who does Bob like?”
  - “Close the door.”

A distinction:
- Linguistic (semantic) meaning
  - “The door is open.”
- Speaker (pragmatic) meaning

Today: assembling the semantic meaning of sentence from its parts
Entailment and Presupposition

Some notions worth knowing*:

Entailment:
- A entails B if A being true necessarily implies B is true
- ? “Twitchy is a big mouse” → “Twitchy is a mouse”
- ? “Twitchy is a big mouse” → “Twitchy is big”
- ? “Twitchy is a big mouse” → “Twitchy is furry”

Presupposition:
- A presupposes B if A is only well-defined if B is true
- “The computer in the corner is broken” presupposes that there is a (salient) computer in the corner
- “When did you get better?” presupposes that there was a time when you were sick.

*Technically, this is pragmatics
Truth-Conditional Semantics

- **Linguistic expressions:**
  - “Bob sings”

- **Logical expressions:**
  - `sings(bob)`
  - Could be `p_1218(e_397)`

- **Denotation:**
  - `[[bob]] =` some specific person (in some context)
  - `[[sings(bob)]] = ???` (for truth-value)

- **Types on logical expressions:**
  - `bob : e` (for entity)
  - `sings(bob) : t` (for truth-value)
Truth-Conditional Semantics

- **Proper names:**
  - Refer directly to some entity in the world
  - Bob : \(\text{bob} \quad \text{[[bob]]}^W \rightarrow ???\)

- **Sentences:**
  - Are either true or false (given how the world actually is)
  - Bob sings : \(\text{sings(bob)}\)

- **So what about verbs (and verb phrases)?**
  - \(\text{sings}\) must combine with \(\text{bob}\) to produce \(\text{sings(bob)}\)
  - The \(\lambda\)-calculus is a notation for functions whose arguments are not yet filled.
  - \(\text{sings} : \lambda x.\text{sings}(x)\)
  - This is *predicate* – a function which takes an entity (type e) and produces a truth value (type t). We can write its type as \(e \rightarrow t\).
  - Nouns? Adjectives? Other cases?
Compositional Semantics

- So now we have meanings for the words
- How do we know how to combine words?
- Associate a combination rule with each grammar rule:
  - $S : \beta(\alpha) \rightarrow NP : \alpha \quad VP : \beta$ (function application)
  - $VP : \lambda x . \alpha(x) \land \beta(x) \rightarrow VP : \alpha \quad$ and $: \emptyset \quad VP : \beta$ (intersection)

- Example:
Denotation

What do we do with logical translations?

- Translation language (logical form) has fewer ambiguities
- Can check truth value against a database
  - Denotation ("evaluation") calculated using the database
- More usefully: assert truth and modify a database
- Questions: check whether a statement in a corpus entails the (question, answer) pair:
  - "Bob sings and dances" → "Who sings?" + "Bob"
- Chain together facts and use them for comprehension
Nouns and Adjectives (Intersection)

- **Nouns:**
  - ball : \( \lambda x. \text{ball}(x) \)
  - One-place predicates of type \( e \rightarrow t \).
  - Define a set of things (for each different world)

- **Adjectives:**
  - green : \( \lambda x. \text{green}(x) \)
  - Also defines a set of things?
  - Same as a noun?

- **Modification (how to parse):**
  - \( N : \lambda x . \alpha(x) \land \beta(x) \rightarrow ADJ : \alpha \ N : \beta \) (intersection)
Other Cases

- Transitive verbs:
  - likes: $\lambda x.\lambda y.\text{likes}(y,x)$
  - Two-place predicates of type $e \rightarrow (e \rightarrow t)$.
  - likes Amy: $\lambda y.\text{likes}(y,Amy)$ is just like a one-place predicate.

- Quantifiers:
  - What does “Everyone” mean here?
  - Everyone: $\lambda f.\forall x.f(x)$
  - Mostly works, but some problems
    - Have to change our NP/VP rule.
    - Won’t work for “Amy likes everyone.”
  - “Everyone likes someone.”
  - This gets tricky quickly!
Indefinites

- First try
  - “Bob ate a waffle”: $\text{ate(bob, waffle)}$
  - “Amy ate a waffle”: $\text{ate(amy, waffle)}$

- Can’t be right!
  - $\exists x : \text{waffle}(x) \land \text{ate}(bob, x)$
  - What does the translation of “a” have to be?
  - What about “the”?
  - What about “every”?
Grounding

So why does the translation \( \text{likes} : \lambda x. \lambda y. \text{likes}(y,x) \) have anything to do with actual liking?

- It doesn’t (unless the denotation model says so)
- Sometimes that’s enough: wire up \textit{bought} to the appropriate entry in a database

Meaning postulates

- Insist, e.g. \( \forall x,y. \text{likes}(y,x) \rightarrow \text{knows}(y,x) \)
- This gets into lexical semantics issues

Statistical version?
Tense and Events

- So far, verbs / events are represented as predicates
  - “Alice danced” : danced(alice)

- Common to, instead, have event variables e
  - \( \exists e : \text{dance}(e) \land \text{agent}(e, alice) \land (\text{time}(e) < \text{now}) \)

- Event variables let you talk about non-trivial tense / aspect structures
  - “Alice had been dancing when Bob sneezed”
  - \( \exists e, e' : \text{dance}(e) \land \text{agent}(e, alice) \land \text{sneeze}(e') \land \text{agent}(e', bob) \land (\text{start}(e) < \text{start}(e') \land \text{end}(e) = \text{end}(e')) \land (\text{time}(e') < \text{now}) \)
Adverbs

- What about adverbs?
  - “Bob sings terribly”
  - terribly(sings(bob))? 
  - (terribly(sings))(bob)?
  - $\exists e \text{ present}(e) \land \text{type } (e, \text{ singing}) \land \text{agent } (e,bob) \land \text{manner}(e, \text{ terrible})$ ?
  - It’s really not this simple..
Propositional Attitudes

- “Bob thinks that I am a gummi bear”
  - \( \text{thinks(bob, gummi(me))} \) ?
  - \( \text{thinks(bob, “I am a gummi bear”)} \) ?
  - \( \text{thinks(bob, ^gummi(me))} \) ?

- Usual solution involves intensions (\(^X\)) which are, roughly, the set of possible worlds (or conditions) in which \( X \) is true

- Hard to deal with computationally
  - Modeling other agents models, etc
  - Can come up in simple dialog scenarios, e.g., if you want to talk about what your bill claims you bought vs. what you actually bought
Trickier Stuff

- **Non-Intersective Adjectives**
  - green ball: $\lambda x. [\text{green}(x) \land \text{ball}(x)]$
  - fake diamond: $\lambda x. [\text{fake}(x) \land \text{diamond}(x)]$

- **Generalized Quantifiers**
  - the: $\lambda f. [\text{unique-member}(f)]$
  - all: $\lambda f. \lambda g [\forall x. f(x) \rightarrow g(x)]$
  - most?
  - Could do with more general second order predicates, too (worse?)
    - the(cat, meows), all(cat, meows)

- **Generics**
  - “Cats like naps”
  - “The players scored a goal”

- **Pronouns (and bound anaphora)**
  - “If you have a dime, put it in the meter.”

- … the list goes on and on!
Multiple Quantifiers

- **Quantifier scope**
  - Groucho Marx celebrates quantifier order ambiguity:
    “In this country a woman gives birth every 15 min. Our job is to find that woman and stop her.”

- **Deciding between readings**
  - “Bob bought a pumpkin every Halloween”
  - “Bob put a warning in every window”
  - Multiple ways to work this out
    - Make it syntactic (different possible parses)
    - Make it lexical (different meaning for the words)
Add a “sem” feature to each context-free rule

- $S \rightarrow NP$ loves $NP$
- $S\{sem=\text{loves}(x,y)\} \rightarrow NP\{sem=x\}$ loves $NP\{sem=y\}$
- Meaning of $S$ depends on meaning of NPs

TAG version:

```
S
  NP
   X
   V
    loves
    NP
    y
  VP
   X
   V
    kicked
    NP
    the bucket
```

Template filling: $S\{sem=\text{showflights}(x,y)\} \rightarrow$

I want a flight from $NP\{sem=x\}$ to $NP\{sem=y\}$
CCG Parsing

- Combinatory Categorial Grammar
  - Fully lexicalized grammar
  - Categories encode argument sequences
  - Very closely related to the lambda calculus
  - Can have spurious ambiguities (why?)

```
John ⊨ NP : john'
shares ⊨ NP : shares'
buys ⊨ (S\NP)/NP : λx.λy.buys'xy
sleeps ⊨ S\NP : λx.sleeps'x
well ⊨ (S\NP)\(S\NP) : λf.λx.well'(fx)
```

Tree representation:
```
  S
   \ /
  NP S\NP
     \ /
    NP buys shares
```
Mapping to Logical Form

- Learning to Map Sentences to Logical Form

Texas borders Kansas

\[ \text{borders(\text{texas, kansas})} \]
Some Training Examples

Input: What states border Texas?
Output: $\lambda x. \text{state}(x) \land \text{borders}(x, \text{texas})$

Input: What is the largest state?
Output: $\text{argmax} (\lambda x. \text{state}(x), \lambda x. \text{size}(x))$

Input: What states border the largest state?
Output: $\lambda x. \text{state}(x) \land \text{borders}(x, \text{argmax}(\lambda y. \text{state}(y), \lambda y. \text{size}(y)))$
## CCG Lexicon

<table>
<thead>
<tr>
<th>Words</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Texas</td>
<td>NP : texas</td>
</tr>
<tr>
<td>borders</td>
<td>(S\NP)/NP : λx.λy.borders(y,x)</td>
</tr>
<tr>
<td>Kansas</td>
<td>NP : kansas</td>
</tr>
<tr>
<td>Kansas city</td>
<td>NP : kansas_city_MO</td>
</tr>
</tbody>
</table>
Parsing Rules (Combinators)

- **Application**
  - \( X/Y : f \quad Y : a \quad \Rightarrow \quad X : f(a) \)

- **Additional rules**
  - **Composition**
  - **Type Raising**
CCG Parsing

\[
\begin{array}{c}
\text{Texas} \\
\text{NP} \\
texas \\
\end{array} \\
\begin{array}{c}
\text{borders} \\
(S\backslash\text{NP})/\text{NP} \\
\lambda x.\lambda y.\text{borders}(y,x) \\
\end{array} \\
\begin{array}{c}
\text{Kansas} \\
\text{NP} \\
kansas \\
\end{array} \\
\begin{array}{c}
\text{S}\backslash\text{NP} \\
\lambda y.\text{borders}(y,\text{kansas}) \\
\text{S} \\
\text{borders}(\text{texas, kansas})
\end{array}
\]
## Parsing a Question

<table>
<thead>
<tr>
<th>What</th>
<th>states</th>
<th>border</th>
<th>Texas</th>
</tr>
</thead>
<tbody>
<tr>
<td>S/(S/\NP)/N</td>
<td>N</td>
<td>(S/\NP)/NP</td>
<td>NP</td>
</tr>
<tr>
<td>(\lambda f. \lambda g. \lambda x. f(x) \land g(x))</td>
<td>(\lambda x. \text{state}(x))</td>
<td>(\lambda x. \lambda y. \text{borders}(y,x))</td>
<td>(\text{texas})</td>
</tr>
<tr>
<td>S/(S/\NP)</td>
<td></td>
<td>S/\NP</td>
<td></td>
</tr>
<tr>
<td>(\lambda g. \lambda x. \text{state}(x) \land g(x))</td>
<td></td>
<td>(\lambda y. \text{borders}(y,\text{texas}))</td>
<td></td>
</tr>
<tr>
<td>S</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(\lambda x. \text{state}(x) \land \text{borders}(x,\text{texas}))</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Lexical Generation

Input Training Example

Sentence: Texas borders Kansas
Logic Form: $\textit{borders}(\textit{texas},\textit{kansas})$

Output Lexicon

<table>
<thead>
<tr>
<th>Words</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Texas</td>
<td>NP : $\textit{texas}$</td>
</tr>
<tr>
<td>borders</td>
<td>$\textit{(S\backslash NP)}/\textit{NP} : \lambda x.\lambda y.\textit{borders}(y,x)$</td>
</tr>
<tr>
<td>Kansas</td>
<td>NP : $\textit{kansas}$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
GENLEX

• Input:
  ■ a training example \((S_i, L_i)\)

• Computation:
  1. Create all substrings of words in \(S_i\)
  2. Create categories from \(L_i\)
  3. Create lexical entries that are the cross product of these two sets

• Output:
  ■ Lexicon \(\Lambda\)
GENLEX Cross Product

Input Training Example

<table>
<thead>
<tr>
<th>Sentence:</th>
<th>Texas borders Kansas</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic Form:</td>
<td>borders(texas,kansas)</td>
</tr>
</tbody>
</table>

Output Lexicon

- **Output Substrings:**
  - Texas
  - borders
  - Kansas
  - Texas borders
  - borders Kansas
  - Texas borders Kansas

- **Output Categories:**
  - NP : texas
  - NP : kansas
  - (S\NP)/NP :
    - \( \lambda x.\lambda y.\text{borders}(y,x) \)

GENLEX is the cross product in these two output sets
## GENLEX: Output Lexicon

<table>
<thead>
<tr>
<th>Words</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Texas</td>
<td>NP : texas</td>
</tr>
<tr>
<td>Texas</td>
<td>NP : kansas</td>
</tr>
<tr>
<td>Texas</td>
<td>$(S\backslash NP)/NP : \lambda x. \lambda y. borders(y,x)$</td>
</tr>
<tr>
<td>borders</td>
<td>NP : texas</td>
</tr>
<tr>
<td>borders</td>
<td>NP : kansas</td>
</tr>
<tr>
<td>borders</td>
<td>$(S\backslash NP)/NP : \lambda x. \lambda y. borders(y,x)$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Texas borders Kansas</td>
<td>NP : texas</td>
</tr>
<tr>
<td>Texas borders Kansas</td>
<td>NP : kansas</td>
</tr>
<tr>
<td>Texas borders Kansas</td>
<td>$(S\backslash NP)/NP : \lambda x. \lambda y. borders(y,x)$</td>
</tr>
</tbody>
</table>
Weighted CCG

Given a log-linear model with a CCG lexicon $\Lambda$, a feature vector $f$, and weights $w$. The best parse is:

$$y^* = \arg\max_y w \times f(x, y)$$

Where we consider all possible parses $y$ for the sentence $x$ given the lexicon $\Lambda$. 
Inputs: Training set \{(x_i, z_i) \mid i=1\ldots n\} of sentences and logical forms. Initial lexicon \Lambda. Initial parameters \(w\). Number of iterations \(T\).

Computation: For \(t = 1\ldots T, i =1\ldots n:\)

Step 1: Check Correctness
- Let \(y^* = \arg\max_y w \times f(x_i, y)\)
- If \(L(y^*) = z_i\), go to the next example

Step 2: Lexical Generation
- Set \(\lambda = \Lambda \cup \text{GENLEX}(x_i, z_i)\)
- Let \(\hat{y} = \arg\max_{y \text{ s.t. } L(y) = z_i} w \times f(x_i, y)\)
- Define \(\lambda_i\) to be the lexical entries in \(\hat{y}\)
- Set lexicon to \(\Lambda = \Lambda \cup \lambda_i\)

Step 3: Update Parameters
- Let \(y' = \arg\max_y w \times f(x_i, y)\)
- If \(L(y') \neq z_i\)
  - Set \(w = w + f(x_i, \hat{y}) - f(x_i, y')\)

Output: Lexicon \(\Lambda\) and parameters \(w\).
<table>
<thead>
<tr>
<th>Words</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>states</td>
<td>N : $\lambda x.\text{state}(x)$</td>
</tr>
<tr>
<td>major</td>
<td>N/N : $\lambda g.\lambda x.\text{major}(x) \land g(x)$</td>
</tr>
<tr>
<td>population</td>
<td>N : $\lambda x.\text{population}(x)$</td>
</tr>
<tr>
<td>cities</td>
<td>N : $\lambda x.\text{city}(x)$</td>
</tr>
<tr>
<td>river</td>
<td>N : $\lambda x.\text{river}(x)$</td>
</tr>
<tr>
<td>run through</td>
<td>(S\NP)/NP : $\lambda x.\lambda y.\text{traverse}(y,x)$</td>
</tr>
<tr>
<td>the largest</td>
<td>NP/N : $\lambda g.\text{argmax}(g,\lambda x.\text{size}(x))$</td>
</tr>
<tr>
<td>rivers</td>
<td>N : $\lambda x.\text{river}(x)$</td>
</tr>
<tr>
<td>the highest</td>
<td>NP/N : $\lambda g.\text{argmax}(g,\lambda x.\text{elev}(x))$</td>
</tr>
<tr>
<td>the longest</td>
<td>NP/N : $\lambda g.\text{argmax}(g,\lambda x.\text{len}(x))$</td>
</tr>
</tbody>
</table>

...
Challenge Revisited

The lexical entries that work for:

```
Show me the latest flight from Boston to Prague on Friday
```

```
S/NP    NP/N     N    N\N     N\N    N\N    N\N
...     ...     ...    ...     ...    ...    ...
```

Will not parse:

```
Boston to Prague the latest on Friday
```

```
NP   N\N    NP/N   N\N
...  ...    ...    ...
```
Reverse the direction of the principal category:

\[
X \backslash Y : f \\
Y : a \\
\Rightarrow \\
X : f(a)
\]

\[
Y : a \\
X/Y : f \\
\Rightarrow \\
X : f(a)
\]
Missing content words

Insert missing semantic content

NP : c => N\N : \lambda f. \lambda x. f(x) \land p(x, c)

<table>
<thead>
<tr>
<th>flights</th>
<th>Boston</th>
<th>to Prague</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>NP</td>
<td>N\N</td>
</tr>
<tr>
<td>\lambda x. flight(x)</td>
<td>BOS</td>
<td>\lambda f. \lambda x. f(x) \land to(x, PRG)</td>
</tr>
<tr>
<td></td>
<td>N\N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>\lambda f. \lambda x. f(x) \land from(x, BOS)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>\lambda x. flight(x) \land from(x, BOS)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>\lambda x. flight(x) \land from(x, BOS) \land to(x, PRG)</td>
<td></td>
</tr>
</tbody>
</table>
Missing content-free words

Bypass missing nouns

\[ N \setminus N : f \rightarrow N : f(\lambda x. \text{true}) \]

Northwest Air

\[
\lambda f. \lambda x.f(x) \land \text{airline}(x, \text{NWA})
\]

to Prague

\[
\lambda f. \lambda x.f(x) \land \text{to}(x, \text{PRG})
\]

\[
\lambda x. \text{to}(x, \text{PRG})
\]

\[
\lambda x. \text{airline}(x, \text{NWA}) \land \text{to}(x, \text{PRG})
\]
A Complete Parse

<table>
<thead>
<tr>
<th>Boston</th>
<th>to Prague</th>
<th>the latest</th>
<th>on Friday</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP</td>
<td>N\N</td>
<td>NP/N</td>
<td>N\N</td>
</tr>
<tr>
<td>BOS</td>
<td>(\lambda f. \lambda x. f(x) \land to(x, \text{PRG}))</td>
<td>(\lambda f. \text{argmax}(\lambda x. f(x), \lambda x. \text{time}(x)))</td>
<td>(\lambda f. \lambda x. f(x) \land \text{day}(x, \text{FRI}))</td>
</tr>
<tr>
<td></td>
<td>(\lambda f. \lambda x. f(x) \land \text{from}(x, \text{BOS}))</td>
<td></td>
<td>(\lambda x. \text{day}(x, \text{FRI}))</td>
</tr>
<tr>
<td></td>
<td>(\lambda f. \lambda x. f(x) \land \text{from}(x, \text{BOS}) \land to(x, \text{PRG}))</td>
<td>(\lambda f. \text{argmax}(\lambda x. f(x) \land \text{from}(x, \text{BOS}) \land to(x, \text{PRG}), \lambda x. \text{time}(x)))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(\lambda f. \text{argmax}(\lambda x. f(x) \land \text{from}(x, \text{BOS}) \land to(x, \text{PRG}), \lambda x. \text{time}(x)))</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(\text{argmax}(\lambda x. \text{from}(x, \text{BOS}) \land to(x, \text{PRG}) \land \text{day}(x, \text{FRI}), \lambda x. \text{time}(x)))</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Precision</td>
<td>Recall</td>
<td>F1</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------</td>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td><strong>GeoQuery, Exact match:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zettlemoyer &amp; Collins 2007</td>
<td>95.49</td>
<td>83.20</td>
<td>88.93</td>
</tr>
<tr>
<td>Zettlemoyer &amp; Collins 2005</td>
<td>96.25</td>
<td>79.29</td>
<td>86.95</td>
</tr>
<tr>
<td>Wong &amp; Money 2007</td>
<td>93.72</td>
<td>80.00</td>
<td>86.31</td>
</tr>
<tr>
<td><strong>ATIS, Partial credit:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zettlemoyer &amp; Collins 2007</td>
<td>95.11</td>
<td>96.71</td>
<td>95.9</td>
</tr>
<tr>
<td>He &amp; Young 2006</td>
<td>---</td>
<td>---</td>
<td>90.3</td>
</tr>
</tbody>
</table>