Natural Language Processing

Compositional Semantics

Dan Klein – UC Berkeley
Truth-Conditional Semantics
Truth-Conditional Semantics

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

- **Logical translations:**
  - \( \text{sings(bob)} \)
  - Could be \( p_{1218}(e_{397}) \)

- **Denotation:**
  - \([[[\text{bob}]]] = \text{some specific person (in some context)}\)
  - \([[[\text{sings(bob)}]]] = ???\)

- **Types on translations:**
  - \( \text{bob : e} \) (for entity)
  - \( \text{sings(bob) : t} \) (for truth-value)
Truth-Conditional Semantics

- **Proper names:**
  - Refer directly to some entity in the world
  - Bob : bob $\llbracket \text{bob} \rrbracket^W \rightarrow ???$

- **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 a *predicate* – a function which takes an entity (type $e$) and produces a truth value (type $t$).
    - We can write its type as $e \rightarrow t$.
  - Adjectives?
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:
  - \( S : \beta(\alpha) \rightarrow NP : \alpha \ V P : \beta \) (function application)
  - \( VP : \lambda x . \alpha(x) \land \beta(x) \rightarrow VP : \alpha \ and : \emptyset \ V P : \beta \) (intersection)
- Example:
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, either explicitly or implicitly
  eg prove a consequence from asserted axioms
- 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
Other Cases

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

- **Quantifiers:**
  - What does “Everyone” mean here?
  - $\text{Everyone} : \lambda f. \forall x. f(x)$
  - 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” : ate(bob,waffle)
  - “Amy ate a waffle” : ate(amy,waffle)

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

\[
\begin{array}{c}
\text{S} \\
\text{NP} \quad \text{VP} \\
\text{Bob} \quad \text{VBD} \quad \text{NP} \\
\text{ate} \quad a \quad \text{waffle}
\end{array}
\]
Grounding

- **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?**
In general, you don’t get far with verbs as predicates

Better to have event variables e

“Alice danced” : danced(alice)

∃ e : dance(e) ∧ agent(e, alice) ∧ (time(e) < now)

Event variables let you talk about non-trivial tense / aspect structures

“Alice had been dancing when Bob sneezed”

∃ e, e’ : dance(e) ∧ agent(e, alice) ∧ 
  sneeze(e’) ∧ agent(e’, bob) ∧ 
  (start(e) < start(e’) ∧ end(e) = end(e’)) ∧ 
  (time(e’) < now)
What about adverbs?

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

- “Bob thinks that I am a gummi bear”
  - thinks(bob, gummi(me)) ?
  - thinks(bob, “I am a gummi bear”) ?
  - 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: $\lambda f. \lambda g \left[ \forall x. f(x) \rightarrow g(x) \right]$
  - most?
  - Could do with more general second order predicates, too (why 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!
Scope Ambiguities

- **Quantifier scope**
  - “All majors take a data science class”
  - “Someone took each of the electives”
  - “Everyone didn’t hand in their exam”

- **Deciding between readings**
  - Multiple ways to work this out
    - Make it syntactic (movement)
    - Make it lexical (type-shifting)
Modeling Uncertainty

- Big difference between statistical disambiguation and statistical reasoning.

  The scout saw the enemy soldiers with night goggles.

  - With probabilistic parsers, can say things like “72% belief that the PP attaches to the NP.”
  - That means that probably the enemy has night vision goggles.
  - However, you can’t throw a logical assertion into a theorem prover with 72% confidence.
  - Use this to decide the expected utility of calling reinforcements?

- Do we need probabilistic reasoning, not just probabilistic disambiguation followed by symbolic reasoning?
Logical Form Translation
CCG Parsing

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

\[
\begin{align*}
\text{John} & \vdash \text{NP} : \text{john}' \\
\text{shares} & \vdash \text{NP} : \text{shares}' \\
\text{buys} & \vdash (\text{S}\text{NP})/\text{NP} : \lambda x.\lambda y.\text{buys}'xy \\
\text{sleeps} & \vdash \text{S}\text{NP} : \lambda x.\text{sleeps}'x \\
\text{well} & \vdash (\text{S}\text{NP})/(\text{S}\text{NP}) : \lambda f.\lambda x.\text{well}'(fx)
\end{align*}
\]
The task:

Input: List one way flights to Prague.
Output: $\lambda x. \text{flight}(x) \land \text{one\_way}(x) \land \text{to}(x, \text{PRG})$

Challenging learning problem:

- Derivations (or parses) are not annotated
- Approach: [Zettlemoyer & Collins 2005]
- Learn a lexicon and parameters for a weighted Combinatory Categorial Grammar (CCG)
Background

- Combinatory Categorial Grammar (CCG)
- Weighted CCGs
- Learning lexical entries: GENLEX
<table>
<thead>
<tr>
<th>Words</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>flights</td>
<td>$N : \lambda x.\text{flight}(x)$</td>
</tr>
<tr>
<td>to</td>
<td>$(N\setminus N)/NP : \lambda x.\lambda f.\lambda y.f(x) \land \text{to}(y,x)$</td>
</tr>
<tr>
<td>Prague</td>
<td>$NP : PRG$</td>
</tr>
<tr>
<td>New York city</td>
<td>$NP : NYC$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Parsing Rules (Combinators)

**Application**
- \( \frac{X}{Y} : f \quad Y : a \Rightarrow X : f(a) \)
- \( \frac{Y : a}{X} \quad X \setminus Y : f \Rightarrow X : f(a) \)

**Composition**
- \( \frac{X}{Y} : f \quad Y/Z : g \Rightarrow X/Z : \lambda x. f(g(x)) \)
- \( \frac{Y \setminus Z : f}{X} \quad X \setminus Y : g \Rightarrow X \setminus Z : \lambda x. f(g(x)) \)

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

<table>
<thead>
<tr>
<th>Show me</th>
<th>flights</th>
<th>to Prague</th>
</tr>
</thead>
<tbody>
<tr>
<td>S/N</td>
<td>N</td>
<td>(N\N)/NP</td>
</tr>
<tr>
<td>\f f</td>
<td>\x flight(x)</td>
<td>\y \f \x f(y) \land to(x,y)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N\N</td>
</tr>
<tr>
<td></td>
<td></td>
<td>\f \x f(x) \land to(x,PRG)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N</td>
</tr>
<tr>
<td></td>
<td></td>
<td>\x flight(x) \land to(x,PRG)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>S</td>
</tr>
<tr>
<td></td>
<td></td>
<td>\x flight(x) \land to(x,PRG)</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:

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

Where we consider all possible parses $y$ for the sentence $x$ given the lexicon $\Lambda$. 
Lexical Generation

Input Training Example

Sentence: Show me flights to Prague.
Logic Form: \( \lambda x.\text{flight}(x) \land \text{to}(x, \text{PRG}) \)

Output Lexicon

<table>
<thead>
<tr>
<th>Words</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show me</td>
<td>S/N : ( \lambda f.f )</td>
</tr>
<tr>
<td>flights</td>
<td>N : ( \lambda x.\text{flight}(x) )</td>
</tr>
<tr>
<td>to</td>
<td>(N/N)/NP : ( \lambda x.\lambda y.f(x) \land \text{to}(y,x) )</td>
</tr>
<tr>
<td>Prague</td>
<td>NP : PRG</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
GENLEX: Substrings X Categories

Input Training Example

<table>
<thead>
<tr>
<th>Sentence</th>
<th>Show me flights to Prague.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic Form</td>
<td>$\lambda x. \text{flight}(x) \land \text{to}(x, \text{PRG})$</td>
</tr>
</tbody>
</table>

Output Lexicon

<table>
<thead>
<tr>
<th>All possible substrings:</th>
<th>Categories created by rules that trigger on the logical form:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show me flights</td>
<td>NP : PRG</td>
</tr>
<tr>
<td>...</td>
<td>N : $\lambda x. \text{flight}(x)$</td>
</tr>
<tr>
<td>Show me flights</td>
<td>(S\NP)/NP : $\lambda x. \lambda y. \text{to}(y, x)$</td>
</tr>
<tr>
<td>Show me flights to</td>
<td>(N\N)/NP : $\lambda y. \lambda f. \lambda x. ...$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

[Zettlemoyer & Collins 2005]
Robustness

The lexical entries that work for:

<table>
<thead>
<tr>
<th>Show me</th>
<th>the latest</th>
<th>flight</th>
<th>from Boston</th>
<th>to Prague</th>
<th>on Friday</th>
</tr>
</thead>
<tbody>
<tr>
<td>S/NP</td>
<td>NP/N</td>
<td>N</td>
<td>N\N</td>
<td>N\N</td>
<td>N\N</td>
</tr>
</tbody>
</table>

Will not parse:

<table>
<thead>
<tr>
<th>Boston to Prague</th>
<th>the latest</th>
<th>on Friday</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP</td>
<td>N\N</td>
<td>N\N</td>
</tr>
</tbody>
</table>

...
Relaxed Parsing Rules

Two changes

- Add application and composition rules that relax word order
- Add type shifting rules to recover missing words

These rules significantly relax the grammar

- Introduce features to count the number of times each new rule is used in a parse
Review: Application

\[ \frac{X}{Y} : f \quad Y : a \quad \Rightarrow \quad X : f(a) \]
\[ Y : a \quad \frac{X \setminus Y}{f} \quad \Rightarrow \quad X : f(a) \]
Disharmonic Application

- Reverse the direction of the principal category:

\[
\begin{align*}
\text{flights} & & \text{one way} \\
N & \lambda x. \text{flight}(x) & N/N & \lambda f. \lambda x. f(x) \land \text{one way}(x) \\
N & \lambda x. \text{flight}(x) \land \text{one way}(x) \\
\end{align*}
\]
Insert missing semantic content

\[
\text{NP} : c \implies \text{N\N} : \lambda f. \lambda x. f(x) \land p(x, c)
\]

<table>
<thead>
<tr>
<th>flights</th>
<th>Boston</th>
<th>to Prague</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\lambda x. \text{flight}(x))</td>
<td>(\text{NP BOS})</td>
<td>(\text{N\N}\lambda f. \lambda x. f(x) \land \text{to}(x, \text{PRG}))</td>
</tr>
<tr>
<td></td>
<td>(\text{N\N})</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(\lambda f. \lambda x. f(x) \land \text{from}(x, \text{BOS}))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(\text{N})</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(\lambda x. \text{flight}(x) \land \text{from}(x, \text{BOS}))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(\text{N})</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(\lambda x. \text{flight}(x) \land \text{from}(x, \text{BOS}) \land \text{to}(x, \text{PRG}))</td>
<td></td>
</tr>
</tbody>
</table>
Bypass missing nouns

- $N\setminus N : f \Rightarrow N : f(\lambda x.\text{true})$

```
N\setminus N
\lambda f. \lambda x.f(x) \land \text{airline}(x, \text{NWA})
```

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

Northwest Air

to Prague
Inputs: Training set \(\{(x_i, z_i) \mid i=1...n\}\) of sentences and logical forms. Initial lexicon \(\Lambda\). Initial parameters \(w\). Number of iterations \(T\).

Training: For \(t = 1...T, i = 1...n\):

Step 1: Check Correctness
- Let \(y^* = \arg\max_y w \cdot f(x_i, y)\)
- 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 \(\lambda = \arg\max_{y, L(y) = z_i} w \cdot f(x_i, y)\)
- Define \(\lambda_i\) to be the lexical entries in \(y^\lambda\)
- Set lexicon to \(\Lambda = \Lambda \cup \lambda_i\)

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

Output: Lexicon \(\Lambda\) and parameters \(w\).
Related Work for Evaluation

Hidden Vector State Model: He and Young 2006
- Learns a probabilistic push-down automaton with EM
- Is integrated with speech recognition

\(\lambda\text{-WASP}:\) Wong & Mooney 2007
- Builds a synchronous CFG with statistical machine translation techniques
- Easily applied to different languages

Zettlemoyer and Collins 2005
- Uses GENLEX with maximum likelihood batch training and stricter grammar
Two Natural Language Interfaces

ATIS (travel planning)
- Manually-transcribed speech queries
- 4500 training examples
- 500 example development set
- 500 test examples

Geo880 (geography)
- Edited sentences
- 600 training examples
- 280 test examples
Precision, Recall, and F-measure for:

- Completely correct logical forms
- Attribute / value partial credit

\[
\lambda x. \text{flight}(x) \land \text{from}(x, \text{BOS}) \land \text{to}(x, \text{PRG})
\]

is represented as:

\[
\{ \text{from} = \text{BOS}, \text{to} = \text{PRG} \}
\]
Two-Pass Parsing

Simple method to improve recall:

- For each test sentence that can not be parsed:
  - Reparse with word skipping
  - Every skipped word adds a constant penalty
  - Output the highest scoring new parse
### ATIS Test Set [Z+C 2007]

**Exact Match Accuracy:**

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-Pass</td>
<td>90.61</td>
<td>81.92</td>
<td><strong>86.05</strong></td>
</tr>
<tr>
<td>Two-Pass</td>
<td>85.75</td>
<td>84.60</td>
<td>85.16</td>
</tr>
</tbody>
</table>
Geo880 Test Set

Exact Match Accuracy:

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-Pass</td>
<td>95.49</td>
<td>83.20</td>
<td><strong>88.93</strong></td>
</tr>
<tr>
<td>Two-Pass</td>
<td>91.63</td>
<td>86.07</td>
<td><strong>88.76</strong></td>
</tr>
<tr>
<td>Zettlemoyer &amp; Collins 2005</td>
<td>96.25</td>
<td>79.29</td>
<td><strong>86.95</strong></td>
</tr>
<tr>
<td>Wong &amp; Mooney 2007</td>
<td>93.72</td>
<td>80.00</td>
<td><strong>86.31</strong></td>
</tr>
</tbody>
</table>