Natural Language Processing

Berkeley

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 \quad \text{(for entity)}
  - \text{sings(bob)} : t \quad \text{(for truth-value)}
Truth-Conditional Semantics

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

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

- **So what about verbs (and verb phrases)?**
  - \( \text{sings} \) must combine with \( \text{bob} \) to produce \( \text{sings}(\text{bob}) \)
  - 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 \( \text{e} \rightarrow \text{t} \).
  - Adjectives?
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)
- $VP : \lambda x . \alpha(x) \land \beta(x) \rightarrow VP : \alpha \quad \text{and} : \emptyset \quad VP : \beta$  (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
- 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!

\[
\frac{
\begin{array}{c}
\forall x. \text{likes}(x,\text{amy}) \\
S \left[ \lambda f. \forall x. f(x) \right] (\lambda y. \text{likes}(y,\text{amy}))
\end{array}
}{
\begin{array}{c}
\text{NP} \\
\text{VP} \quad \lambda y. \text{likes}(y,\text{amy})
\end{array}
}
\]

\[
\frac{
\begin{array}{c}
\text{Everyone} \\
\lambda f. \forall x. f(x)
\end{array}
}{
\begin{array}{c}
\text{VP} \quad \text{likes} \\
\text{NP} \\
\text{Amy} \quad \lambda x. \lambda y. \text{likes}(y,x) \quad \text{amy}
\end{array}
}
\]
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”?
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” : $\text{danced(alice)}$
- $\exists e: \text{dance(e)} \land \text{agent(e,alice)} \land (\text{time(e)} < \text{now})$

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

- “Alice had been dancing when Bob sneezed”
- $\exists e, e': \text{dance(e)} \land \text{agent(e,alice)} \land$
  $\text{sneeze(e')} \land \text{agent(e',bob)} \land$
  $(\text{start(e)} < \text{start(e')} \land \text{end(e)} = \text{end(e')}) \land$
  $(\text{time(e')} < \text{now})$
What about adverbs?

- “Bob sings terribly”
- terribly(sings(bob))?
- (terribly(sings))(bob)?
- ∃e present(e) \land type(e, singing) \land agent(e,bob) \land manner(e, 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: $\lambda f. \lambda g [\forall x. f(x) \rightarrow g(x)]$
  - 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!
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 uses a phone as an alarm each morning”
  - 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?

- In short, we need probabilistic reasoning, not just probabilistic disambiguation followed by symbolic reasoning.
Logical Form Translation
Combinatory Categorial Grammar

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

\[ John \vdash NP : john' \]
\[ shares \vdash NP : shares' \]
\[ buys \vdash (S\backslash NP) / NP : \lambda x.\lambda y.\text{buys}'xy \]
\[ sleeps \vdash S\backslash NP : \lambda x.\text{sleeps}'x \]
\[ well \vdash (S\backslash NP) \backslash (S\backslash NP) : \lambda f.\lambda x.\text{well}'(fx) \]
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>$(N\backslash N)/NP : \lambda x.\lambda f.\lambda y. f(x) \land to(y,x)$</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

- $X/Y : f \quad Y : a \Rightarrow X : f(a)$
- $Y : a \quad X \backslash Y : f \Rightarrow X : f(a)$

Composition

- $X/Y : f \quad Y/Z : g \Rightarrow X/Z : \lambda x.f(g(x))$
- $Y \backslash Z : f \quad X \backslash Y : g \Rightarrow X \backslash Z : \lambda x.f(g(x))$

Additional rules:

- Type Raising
- Crossed Composition
### CCG Parsing

<table>
<thead>
<tr>
<th><strong>Show me</strong></th>
<th><strong>flights</strong></th>
<th><strong>to</strong></th>
<th><strong>Prague</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>S/N</td>
<td>N</td>
<td>(N\N) /NP</td>
<td>NP PRG</td>
</tr>
<tr>
<td>λf.f</td>
<td>λx.flight(x)</td>
<td>λy.λf.λx.f(y)∧to(x, y)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>N\N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>λf.λx.f(x)∧to(x, PRG)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>λx.flight(x)∧to(x, PRG)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S</td>
<td>λx.flight(x)∧to(x, PRG)</td>
<td></td>
</tr>
</tbody>
</table>
Given a log-linear model with a CCG lexicon $\Lambda$, a feature vector $f$, and weights $w$. The best parse is:

$$y^* = \underset{y}{\operatorname{argmax}} \ w \cdot f(x, y)$$

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 : $\lambda x.\lambda f.\lambda y.f(x) \land \text{to}(y,x)$</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

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

Output Lexicon

All possible substrings:
Show me flights ...
Show me flights to ...

Categories created by rules that trigger on the logical form:

NP : PRG
N : $\lambda x.\text{flight}(x)$
(S\NP)/NP : $\lambda x.\lambda y.\text{to}(y, x)$
(N\N)/NP : $\lambda y.\lambda f.\lambda x. \ldots$

[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>NP/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

\[
\begin{align*}
X/Y : f & \quad Y : a & \Rightarrow & \quad X : f(a) \\
Y : a & \quad X\backslash Y : f & \Rightarrow & \quad X : f(a)
\end{align*}
\]
Reverse the direction of the principal category:

\[
\begin{align*}
X \setminus Y : f & \quad Y : a \quad \Rightarrow \quad X : f(a) \\
Y : a & \quad X \setminus Y : f \quad \Rightarrow \quad X : f(a)
\end{align*}
\]

<table>
<thead>
<tr>
<th>flights</th>
<th>one way</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda x.\text{flight}(x)$</td>
<td>$\lambda f.\lambda x. f(x) ^\wedge \text{one_way}(x)$</td>
</tr>
<tr>
<td>$\mathbb{N}$</td>
<td>$\mathbb{N}/\mathbb{N}$</td>
</tr>
</tbody>
</table>

$\forall x\in\mathbb{N} \cdot \text{flight}(x) ^\wedge \text{one\_way}(x)$
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 BOS</td>
<td>N\N</td>
</tr>
<tr>
<td>(\lambda x.\text{flight}(x))</td>
<td></td>
<td>(\lambda f.\lambda x.f(x) \land \text{to}(x,\text{PRG}))</td>
</tr>
<tr>
<td>N\N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(\lambda f.\lambda x.f(x) \land \text{from}(x,\text{BOS}))</td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>(\lambda x.\text{flight}(x) \land \text{from}(x,\text{BOS}))</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(\lambda x.\text{flight}(x) \land \text{from}(x,\text{BOS}) \land \text{to}(x,\text{PRG}))</td>
</tr>
</tbody>
</table>
Bypass missing nouns

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

<table>
<thead>
<tr>
<th>Northwest Air</th>
<th>to Prague</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda f. \lambda x. f(x) \land \text{airline}(x, \text{NWA})$</td>
<td>$\lambda f. \lambda x. f(x) \land \text{to}(x, \text{PRG})$</td>
</tr>
<tr>
<td>$\lambda x. \text{to}(x, \text{PRG})$</td>
<td>$\lambda x. \text{airline}(x, \text{NWA}) \land \text{to}(x, \text{PRG})$</td>
</tr>
</tbody>
</table>
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 \( T \).

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

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 \( \hat{\lambda} = \arg \max_y \max \{ w \cdot f(x_i, y) \mid y \text{ s.t. } L(y) = z_i \} \)
- 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 \( w = w + f(x_i, \hat{\lambda}) - f(x_i, y') \)

Output: Lexicon \( \Lambda \) and parameters \( w \).
Hidden Vector State Model: He and Young 2006
  ▪ Learns a probabilistic push-down automaton with EM
  ▪ Is integrated with speech recognition

λ-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
Evaluation Metrics

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 cannot 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>