Natural Language Processing

Compositional Semantics
Dan Klein – UC Berkeley

Truth-Conditional Semantics

Linguistic expressions:
- “Bob sings”

Logical translations:
- \( \text{sing}(\text{bob}) \)
- \( \text{Cloud} \text{下雨}(\text{李四}, \text{小四}) \)

Denotation:
- \( \llbracket \text{bob} \rrbracket = \) some specific person (in some context)
- \( \llbracket \text{sing}(\text{bob}) \rrbracket = ?? \)

Types on translations:
- \( \text{bob} : e \) (for entity)
- \( \text{sing}(\text{bob}) : t \) (for truth value)

Adjectives?

Proper names:
- Refer directly to some entity in the world
  - Bob: bob \( \llbracket \text{bob} \rrbracket = ?? \)

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

So what about verbs (and verb phrases)?
- \( \text{sing} \) must combine with Bob to produce \( \text{sing}(\text{bob}) \)
- \( \text{L-calculus} \) is a notation for functions whose arguments are not yet filled.
- \( \text{sing} : l \times \text{sing}(\text{bob}) \)
- 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:
  - $S \rightarrow b \,(a)$
  - $NP \rightarrow a$
  - $VP \rightarrow b \,(\text{function application})$
  - $VP \rightarrow l \,x \cdot a \,(x) \land b \,(x) \rightarrow VP$
  - $\text{and} \rightarrow \&$
  - $VP \rightarrow b \,(\text{intersection})$

Example:

$S \rightarrow NP \rightarrow VP$
$\text{Bob}$
$VP \rightarrow \text{and}$
$sings$
$dances$

$NP \rightarrow Bob$
$VP \rightarrow l \,x \cdot a \,(x) \land b \,(x) \rightarrow VP$
$sings(y)$
$dances(z)$

$\text{and} \rightarrow \&$

$S \rightarrow NP \rightarrow VP$
$\text{Bob}$
$VP \rightarrow \text{and}$
$sings$
$dances$

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, 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:
  - $\text{likes} : l \,x \cdot l \,y \cdot \text{likes}(y,x)$
  - Two-place predicates of type $\epsilon \rightarrow \epsilon$:
  - $\text{likes Amy} : l \,y \cdot \text{likes}(y,\text{Amy})$ is just like a one-place predicate.
- Quantifiers:
  - What does "Everyone" mean here?
  - Everyone : $\exists \,x \cdot (l \,y \cdot \text{likes}(y,x))$
  - Mostly works, but some problems
  - Want to change our N/NP rule
  - Won't work for "Amy likes everyone -
  - "Everyone likes someone."
  - This gets sticky quickly!

Indefinites

- First try
  - "Bob ate a waffle" : $\text{ate}(\text{Bob}, \text{waffle})$
  - "Amy ate a waffle" : $\text{ate}(\text{Amy}, \text{waffle})$
- Can't be right!
  - $\exists \,x : \text{waffle}(x) \land \text{ate}(\text{Bob}, x)$
  - What does the translation of "the" have to be?
  - What about "that"?
  - What about "every"?
Grounding

- So why does the translation like:x,l:likes(x,y) have anything to do with actual liking?
  - It doesn’t (unless the denotation model says so)
- Sometimes that’s enough: wire up bought to the appropriate entry in a database

Meaning postulates

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

Statistical version?

Tense and Events

- In general, you don’t get far with verbs as predicates
- Better to have event variables e
  - Nice-dance(e) \text{ [dancer\_alias] }
  - \exists{e} : \text{dance(e)} \land \text{agent(e,alice)} \land \text{time(e) < 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) < start(e')} \land \text{end(e) = end(e')} \land \text{time(e') < now}

Adverbs

- What about adverbs?
  - "Bob sings terribly"
  - terribly(sings)bobby?
  - Is 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 gumi bear"
  - thinks(bob, gummi(me)) ?
  - thinks(bob, "I am a gumi 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: \( l(x). [\text{green}(x) \land \text{ball}(x)] \)
  - non-intersective: \( l(x). [\text{green}(x) \lor \text{diamond}(x)] \)  

- Generalized Quantifiers:
  - for any: \( l(x). [\text{fake}(x) \land \text{diamond}(x)] \)  
  - for all: \( l(f). l(g) [\forall x. f(x) \land g(x)] \)  
  - could do with many general second order predicates, too (why worry?)

- Generics:
  - fire: \( l(f). [\text{unique-member}(f)] \)  
  - all: \( l(f). l(g) [\forall x. f(x) \land g(x)] \)  

- 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

- Logical form translation steps:

  - 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?
CCG Parsing

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

\[ \text{John (S(NP)/NP)} \rightarrow \text{S'NP} \]
\[ \text{John (S(NP)/NP)} \rightarrow \text{NP} \]
\[ \text{John (S(NP)/NP)} \rightarrow \text{NP} \]
\[ \text{John (S(NP)/NP)} \rightarrow \text{NP} \]

Mapping to LF: Zettlemoyer & Collins 05/07

The task:

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

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

CCG Lexicon

<table>
<thead>
<tr>
<th>Words</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>flights</td>
<td>N : Ax.flight(x)</td>
</tr>
<tr>
<td>to</td>
<td>(M(X))/NP : Ax.Af, A(e)(x) &amp; (o_{x}, 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
- $X/Y : f$  $Y : a$ $\Rightarrow$  $X : f(a)$
- $Y : a$  $X/Y : f$  $\Rightarrow$  $X : f(a)$

Composition
- $X/Y : f$  $X/Z : g$  $\Rightarrow$  $X/Z : l_x.f(g(x))$
- $Y : f$  $X/Y : g$  $\Rightarrow$  $X : l_x.f(g(x))$

Additional rules:
- Type Raising
- Crossed Composition

CCG Parsing

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:  $\exists x.\, \text{Flight}(x) \land \text{to}(x,\text{PRG})$

Output Lexicon

<table>
<thead>
<tr>
<th>Words</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show me</td>
<td>S/N : Art.f</td>
</tr>
<tr>
<td>flights</td>
<td>S/N : Art.Flight(a)</td>
</tr>
<tr>
<td>to</td>
<td>S/N : Art.to(x,PRG)</td>
</tr>
<tr>
<td>Prague</td>
<td>Art : PRO</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
GENLEX: Substrings X Categories

Input Training Example

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

Output Lexicon

All possible substrings: Show me flights --- Categories created by rules that trigger on the logical form:

Show me flights $\lambda x.\text{flight}(x)$ ---

Show me flights $\lambda x.\text{flight}(x)$ ---

Show me flights $\lambda x.\text{flight}(x)$ ---

Show me flights $\lambda x.\text{flight}(x)$ ---

Robustness

The lexical entries that work for:

Show me the latest flight from Boston to Prague on Friday

Will not parse:

Boston to Prague the latest on Friday

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

$X/Y : f$ $Y : a$ $\Rightarrow X : f(a)$

$Y : a$ $X/V : f$ $\Rightarrow X : f(a)$
Disharmonic Application

- Reverse the direction of the principal category:
  \[ Y : a \quad \text{and} \quad X : f(a) \]

Missing content words

Insert missing semantic content

- NF : e \[\Rightarrow\] WU : Af, a, f(a) \land p(a, e)

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

Training: For \( t = 1, \ldots, T \):
  1. Check Correctness
     - If \( \Lambda(y^*) = z_i \), go to the next example
  2. Lexical Generation
     - Set \( y = \text{argmax} \ y \times f(x, y) \)
     - Define \( l \) to be the lexical entries in \( y^* \)
     - Set lexicon to \( \Lambda \leftarrow \Lambda \cup \{l\} \)
  3. Update Parameters
     - Let \( w' = w + f(x, y) - f(x, y^*) \)

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

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

J/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

\[ \text{Ask: Flight(x)} \land \text{From(x, BOS)} \land \text{to(x, PRG)} \]

is represented as:
\[ \{ \text{From = BOS, to = PRG} \} \]

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>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-Pass</td>
<td>90.61</td>
<td>81.92</td>
</tr>
<tr>
<td>Two-Pass</td>
<td>85.75</td>
<td>84.60</td>
</tr>
</tbody>
</table>

### Geo880 Test Set

**Exact Match Accuracy:**

<table>
<thead>
<tr>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-Pass</td>
<td>95.49</td>
<td>81.20</td>
</tr>
<tr>
<td>Two-Pass</td>
<td>91.63</td>
<td>86.07</td>
</tr>
</tbody>
</table>

Zettlemoyer & Collins 2005: 96.25, 79.29, 86.95
Wong & Mooney 2007: 93.72, 80.00, 86.31