Statistical NLP
Spring 2010

University of California
Berkeley

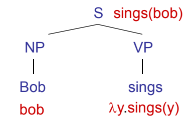
Lecture 21: Compositional Semantics

Dan Klein – UC Berkeley

Includes slides from Luke Zettlemoyer

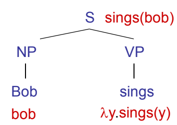
Truth-Conditional Semantics

- Linguistic expressions:
  - "Bob sings"
- Logical translations:
  - sings(bob)
  - Could be p_1218(e_397)
- Denotation:
  - [[bob]] = some specific person (in some context)
  - [[sings(bob)]] = ???
- Types on translations:
  - bob : e (for entity)
  - sings(bob) : t (for truth-value)



Truth-Conditional Semantics

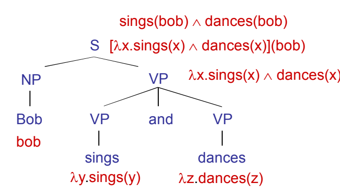
- Proper names:
  - Refer directly to some entity in the world
  - Bob : bob [[bob]]^W → ???
- Sentences:
  - Are either true or false (given how the world actually is)
  - Bob sings : sings(bob)



- So what about verbs (and verb phrases)?
  - sings must combine with bob to produce sings(bob)
  - The λ-calculus is a notation for functions whose arguments are not yet filled.
  - sings : λx.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→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 : β(α) → NP : α VP : β (function application)
  - VP : λx . α(x) ∧ β(x) → VP : α and : ∅ VP : β (intersection)
- Example:

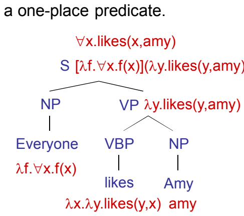


Denotation

- 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

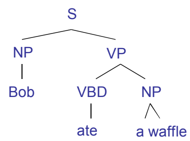
Other Cases

- Transitive verbs:
  - likes : λx.λy.likes(y,x)
  - Two-place predicates of type e→(e→t).
  - likes Amy : λy.likes(y,Amy) is just like a one-place predicate.
- Quantifiers:
  - What does "Everyone" mean here?
  - Everyone : λf.∀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!
  - ∃ x : waffle(x) ∧ ate(bob,x)
  - What does the translation of "a" have to be?
  - What about "the"?
  - What about "every"?



Grounding

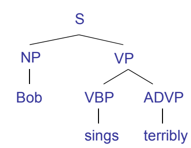
- Grounding
  - So why does the translation likes : λx.λy.likes(y,x) have anything to do with actual liking?
  - It doesn't (unless the denotation model says so)
  - Sometimes that's enough: wire up bought to the appropriate entry in a database
- Meaning postulates
  - Insist, e.g ∀x,y.likes(y,x) → knows(y,x)
  - 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
  - "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)

Adverbs

- What about adverbs?
  - "Bob sings terribly"
  - terribly(sings(bob))?
  - (terribly(sings))(bob)?
  - ∃e present(e) ∧ type(e, singing) ∧ agent(e,bob) ∧ manner(e, terrible) ?
  - Hard to work out correctly!



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 : λx.[green(x) ∧ ball(x)]
  - fake diamond : λx.[fake(x) ∧ diamond(x)] ? → λx.[fake(diamond(x))
- Generalized Quantifiers
  - the : λf.[*unique-member*(f)]
  - all : λf. λg [∀x.f(x) → 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 put a warning in every window"
  - Multiple ways to work this out
    - Make it syntactic (movement)
    - Make it lexical (type-shifting)

Modeling Uncertainty?

- Gaping hole warning!
- 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.
- Not clear humans really extract and process logical statements symbolically anyway.
- Use this to decide the expected utility of calling reinforcements?

- In short, we need probabilistic reasoning, not just probabilistic disambiguation followed by symbolic reasoning!

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?)

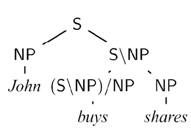
John ⊢ NP : $john'$

shares ⊢ NP : $shares'$

buys ⊢ (S\NP)/NP : $\lambda x.\lambda y.buys' xy$

sleeps ⊢ S\NP : $\lambda x.sleeps' x$

well ⊢ (S\NP)\(S\NP) : $\lambda f.\lambda x.well'(fx)$



Mapping to Logical Form

- Learning to Map Sentences to Logical Form

`Texas borders Kansas`

↓

borders(texas,kansas)

Some Training Examples

Input: `What states border Texas?`
Output: $\lambda x.state(x) \wedge borders(x,texas)$

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

Input: `What states border the largest state?`
Output: $\lambda x.state(x) \wedge borders(x, argmax(\lambda y.state(y), \lambda y.size(y)))$

CCG Lexicon

Words	Category: Syntax : Semantics
Texas	NP : *texas*
borders	(S\NP)/NP : $\lambda x.\lambda y.borders(y,x)$
Kansas	NP : *kansas*
Kansas city	NP : *kansas_city_MO*

Parsing Rules (Combinators)

- Application
  - X/Y : f   Y : a   => X : f(a)

```
(S\NP)/NP                 NP        S\NP
λx.λy.borders(y,x)        texas     λy.borders(y,texas)
```

  - Y : a   X\Y : f   => X : f(a)

```
NP        S\NP                   S
kansas    λy.borders(y,texas)    borders(kansas,texas)
```

- Additional rules
  - Composition
  - Type Raising

CCG Parsing

```
Texas     borders               Kansas
-----     ------------------    ------
NP        (S\NP)/NP             NP
texas     λx.λy.borders(y,x)    kansas
          ----------------------------
                  S\NP
                  λy.borders(y,kansas)
--------------------------------------
        S
        borders(texas,kansas)
```

Parsing a Question

```
What                     states          border                Texas
--------------------     ------------    ------------------    -----
S/(S\NP)/N               N               (S\NP)/NP             NP
λf.λg.λx.f(x)∧g(x)       λx.state(x)     λx.λy.borders(y,x)    texas
--------------------------------------   --------------------------
        S/(S\NP)                                 S\NP
        λg.λx.state(x)∧g(x)                      λy.borders(y,texas)
--------------------------------------------------------------------
                    S
                    λx.state(x) ∧ borders(x,texas)
```

Lexical Generation

Input Training Example

Sentence: Texas borders Kansas

Logic Form: borders(texas,kansas)

Output Lexicon

Words	Category
Texas	NP : texas
borders	(S\NP)/NP : λx.λy.borders(y,x)
Kansas	NP : kansas
...	...

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 Λ

GENLEX Cross Product

Input Training Example

Sentence: Texas borders Kansas

Logic Form: borders(texas,kansas)

Output Lexicon

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

X

- Output Categories:
  - NP : texas
  - NP : kansas
  - (S\NP)/NP : λx.λy.borders(y,x)

GENLEX is the cross product in these two output sets

GENLEX: Output Lexicon

Words	Category
Texas	NP : *texas*
Texas	NP : *kansas*
Texas	(S\NP)/NP : *λx.λy.borders(y,x)*
borders	NP : *texas*
borders	NP : *kansas*
borders	(S\NP)/NP : *λx.λy.borders(y,x)*
...	...
Texas borders Kansas	NP : *texas*
Texas borders Kansas	NP : *kansas*
Texas borders Kansas	(S\NP)/NP : *λx.λy.borders(y,x)*

Weighted CCG

Given a log-linear model with a CCG lexicon Λ, 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 Λ.

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

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

Step 1: Check Correctness

- Let $y^* = \underset{y}{\operatorname{argmax}}\ w \cdot f(x_i, y)$
- 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 $\hat{y} = \arg \max_{y \text{ s.t. } L(y)=z_i} w \cdot f(x_i, y)$
- Define λ_i to be the lexical entries in $\hat{y}$
- Set lexicon to $\Lambda = \Lambda \cup \lambda_i$

Step 3: Update Parameters

- Let $y' = \underset{y}{\operatorname{argmax}}\ w \cdot f(x_i, y)$
- If $L(y') \neq z_i$
  - Set $w = w + f(x_i, \hat{y}) - f(x_i, y')$

Output: Lexicon Λ and parameters w.

Example Learned Lexical Entries

Words	Category
states	N : *λx.state(x)*
major	N/N : *λg.λx.major(x)∧g(x)*
population	N : *λx.population(x)*
cities	N : *λx.city(x)*
river	N : *λx.river(x)*
run through	(S\NP)/NP : *λx.λy.traverse(y,x)*
the largest	NP/N : *λg.argmax(g,λx.size(x))*
rivers	N : *λx.river(x)*
the highest	NP/N : *λg.argmax(g,λx.elev(x))*
the longest	NP/N : *λg.argmax(g,λx.len(x))*
...	...

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
  ...        ...        ...        ...          ...          ...
```

Will not parse:

```
Boston   to Prague   the latest   on Friday
  NP        N\N         NP/N         N\N
  ...       ...          ...         ...
```

Disharmonic Application

Reverse the direction of the principal category:

```
X\Y : f      Y : a     =>   X : f(a)
Y : a      X/Y : f     =>   X : f(a)
```

```
       flights                  one way
 -----------------    -----------------------------
         N                        N/N
   λx.flight(x)         λf.λx.f(x)∧one_way(x)
 ---------------------------------------------------
                          N
              λx.flight(x)∧one_way(x)
```

Missing content words

Insert missing semantic content

NP : c => N\N : λf.λx.f(x) ∧ p(x,c)

flights | Boston | to Prague

N : λx.flight(x)

NP : BOS

N\N : λf.λx.f(x)∧to(x,PRG)

N\N : λf.λx.f(x)∧from(x,BOS)

N : λx.flight(x)∧from(x,BOS)

N : λx.flight(x)∧from(x,BOS)∧to(x,PRG)

Missing content-free words

Bypass missing nouns

N\N : f => N : f(λx.true)

Northwest Air | to Prague

N/N : λf.λx.f(x)∧airline(x,NWA)

N\N : λf.λx.f(x)∧to(x,PRG)

N : λx.to(x,PRG)

N : λx.airline(x,NWA) ∧ to(x,PRG)

A Complete Parse

Boston | to Prague | the latest | on Friday

NP : BOS

N\N : λf.λx.f(x)∧to(x,PRG)

NP/N : λf.argmax(λx.f(x),λx.time(x))

N\N : λf.λx.f(x)∧day(x,FRI)

N\N : λf.λx.f(x)∧from(x,BOS)

N : λx.day(x,FRI)

N\N : λf.λx.f(x)∧from(x,BOS) ∧to(x,PRG)

NP\N : λf.argmax(λx.f(x)∧from(x,BOS)∧to(x,PRG), λx.time(x))

N : argmax(λx.from(x,BOS)∧to(x,PRG)∧day(x,FRI), λx.time(x))

Geo880 Test Set

Exact Match Accuracy:	Precision	Recall	F1
Zettlemoyer & Collins 2007	95.49	83.20	88.93
Zettlemoyer & Collins 2005	96.25	79.29	86.95
Wong & Money 2007	93.72	80.00	86.31