Statistical NLP

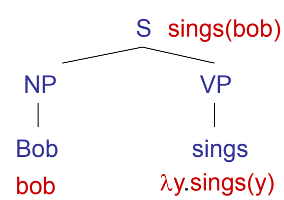
Spring 2010

Lecture 21: Compositional Semantics

Dan Klein – UC Berkeley

Includes slides from Luke Zettlemoyer

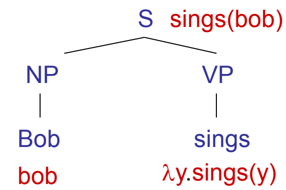
Truth-Conditional Semantics



- 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

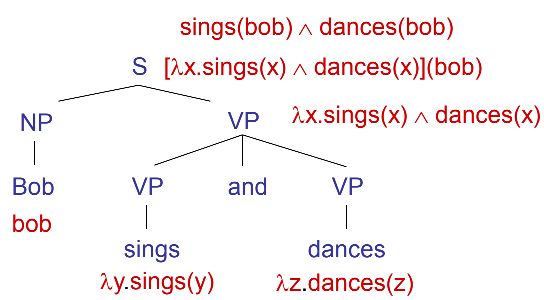
- Proper names:
  - Refer directly to some entity in the world
  - Bob : bob $[[bob]]^W$ → ???



- 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 $e \to t$.
  - 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:
  - $S : \beta(\alpha) \to NP : \alpha \quad VP : \beta$ (function application)
  - $VP : \lambda x . \alpha(x) \wedge \beta(x) \to VP : \alpha \quad and : \varnothing \quad VP : \beta$ (intersection)
- Example:

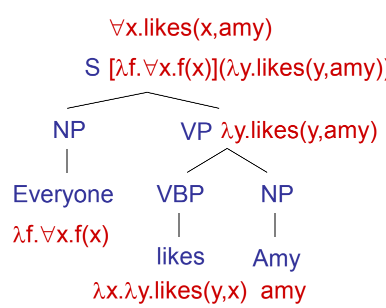


Denotation

- 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” $\rightarrow$ “Who sings?” + “Bob”
  - Chain together facts and use them for comprehension

Other Cases

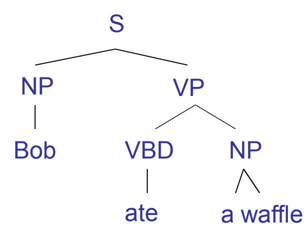
- Transitive verbs:
  - likes : $\lambda x.\lambda y.likes(y,x)$
  - Two-place predicates of type $e \rightarrow (e \rightarrow t)$.
  - likes Amy : $\lambda y.likes(y,Amy)$ is just like a one-place predicate.
- Quantifiers:
  - What does “Everyone” mean here?
  - Everyone : $\lambda f.\forall x.f(x)$
  - 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" : ate(bob,waffle)
  - "Amy ate a waffle" : ate(amy,waffle)

- Can't be right!
  - $\exists$ x : waffle(x) $\wedge$ 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 $\forall$x,y.likes(y,x) $\rightarrow$ knows(y,x)
  - This gets into lexical semantics issues

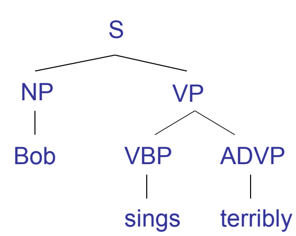
- Statistical version?

Tense and Events

- In general, you don't get far with verbs as predicates
- Better to have event variables e
  - "Alice danced" : danced(alice)
  - $\exists$ e : dance(e) $\wedge$ agent(e,alice) $\wedge$ (time(e) $<$ now)
- Event variables let you talk about non-trivial tense / aspect structures
  - "Alice had been dancing when Bob sneezed"
  - $\exists$ e, e' : dance(e) $\wedge$ agent(e,alice) $\wedge$
    sneeze(e') $\wedge$ agent(e',bob) $\wedge$
    (start(e) $<$ start(e') $\wedge$ end(e) = end(e')) $\wedge$
    (time(e') $<$ now)

Adverbs

- What about adverbs?
  - "Bob sings terribly"
  - terribly(sings(bob))?
  - (terribly(sings))(bob)?
  - $\exists$e present(e) $\wedge$ type(e, singing) $\wedge$ agent(e,bob) $\wedge$ 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 : $\lambda x.[green(x) \wedge ball(x)]$
  - fake diamond : $\lambda x.[fake(x) \wedge diamond(x)]$? $\longrightarrow$ $\lambda x.[fake(diamond(x))$
- Generalized Quantifiers
  - the : $\lambda f.[\textit{unique-member}(f)]$
  - 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 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?)

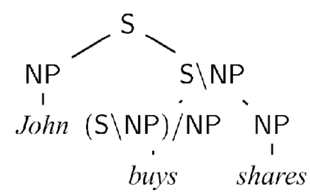
$John \vdash \text{NP} : john'$

$shares \vdash \text{NP} : shares'$

$buys \vdash (\text{S}\backslash\text{NP})/\text{NP} : \lambda x.\lambda y.buys' xy$

$sleeps \vdash \text{S}\backslash\text{NP} : \lambda x.sleeps' x$

$well \vdash (\text{S}\backslash\text{NP})\backslash(\text{S}\backslash\text{NP}) : \lambda f.\lambda x.well'(fx)$



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

S\NP
λy.borders(y,kansas)

S
borders(texas,kansas)

Parsing a Question

What	states	border	Texas
S/(S\NP)/N $\lambda f.\lambda g.\lambda x.f(x) \wedge g(x)$	N $\lambda x.state(x)$	(S\NP)/NP $\lambda x.\lambda y.borders(y,x)$	NP $texas$

S/(S\NP)
$\lambda g.\lambda x.state(x) \wedge g(x)$

S\NP
$\lambda y.borders(y,texas)$

S
$\lambda x.state(x) \wedge borders(x,texas)$

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 : $\lambda x.\lambda y.borders(y,x)$
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 : $\lambda x.\lambda y.borders(y,x)$
borders	NP : *texas*
borders	NP : *kansas*
borders	(S\NP)/NP : $\lambda x.\lambda y.borders(y,x)$
...	...
Texas borders Kansas	NP : *texas*
Texas borders Kansas	NP : *kansas*
Texas borders Kansas	(S\NP)/NP : $\lambda x.\lambda y.borders(y,x)$

Weighted CCG

Given a log-linear model with a CCG lexicon Λ, a feature vector f, and weights w.

The best parse is:

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

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

Step 3: Update Parameters

- Let $y' = \arg\max_y w \cdot f(x_i, y)$
- 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 : $\lambda x.state(x)$
major	N/N : $\lambda g.\lambda x.major(x) \wedge g(x)$
population	N : $\lambda x.population(x)$
cities	N : $\lambda x.city(x)$
river	N : $\lambda x.river(x)$
run through	(S\NP)/NP : $\lambda x.\lambda y.traverse(y,x)$
the largest	NP/N : $\lambda g.argmax(g,\lambda x.size(x))$
rivers	N : $\lambda x.river(x)$
the highest	NP/N : $\lambda g.argmax(g,\lambda x.elev(x))$
the longest	NP/N : $\lambda g.argmax(g,\lambda x.len(x))$
...	...

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**
$\lambda x.flight(x)$	$\lambda f.\lambda x.f(x) \wedge one\_way(x)$

N

$\lambda x.flight(x) \wedge one\_way(x)$

Missing content words

Insert missing semantic content

NP : c => N\N : λf.λx.f(x) ∧ p(x,c)

```
flights          Boston                      to Prague
-------          ------                      ---------
N                NP                          N\N
λx.flight(x)     BOS                         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                                N\N
λf.λx.f(x)∧airline(x,NWA)          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 $\lambda f.\lambda x.f(x)\wedge to(x,\mathrm{PRG})$	NP/N $\lambda f.argmax(\lambda x.f(x),\lambda x.time(x))$	N\N $\lambda f.\lambda x.f(x)\wedge day(x,\mathrm{FRI})$

N\N
$\lambda f.\lambda x.f(x)\wedge from(x,\mathrm{BOS})$

N\N
$\lambda f.\lambda x.f(x)\wedge from(x,\mathrm{BOS})\wedge to(x,\mathrm{PRG})$

N
$\lambda x.day(x,\mathrm{FRI})$

NP\N
$\lambda f.argmax(\lambda x.f(x)\wedge from(x,\mathrm{BOS})\wedge to(x,\mathrm{PRG}),\ \lambda x.time(x))$

N
$argmax(\lambda x.from(x,\mathrm{BOS})\wedge to(x,\mathrm{PRG})\wedge day(x,\mathrm{FRI}),\ \lambda x.time(x))$

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