Statistical NLP

Spring 2007

Lecture 20: More Semantics

Dan Klein – UC Berkeley

Includes examples from Johnson, Jurafsky and Gildea, Luo, Palmer

Meaning

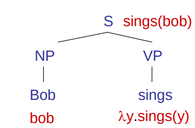
- "Meaning"
  - What is meaning?
    - "The computer in the corner."
    - "Bob likes Alice."
    - "I think I am a gummi bear."
  - Knowing whether a statement is true?
  - Knowing the conditions under which it's true?
  - Being able to react appropriately to it?
    - "Who does Bob like?"
    - "Close the door."
- A distinction:
  - Linguistic (semantic) meaning
    - "The door is open."
  - Speaker (pragmatic) meaning
- Today: assembling the semantic meaning of sentence from its parts

Entailment and Presupposition

- Some notions worth knowing:
  - Entailment:
    - A entails B if A being true necessarily implies B is true
    - ? "Twitchy is a big mouse" → "Twitchy is a mouse"
    - ? "Twitchy is a big mouse" → "Twitchy is big"
    - ? "Twitchy is a big mouse" → "Twitchy is furry"
  - Presupposition:
    - A presupposes B if A is only well-defined if B is true
    - "The computer in the corner is broken" presupposes that there is a (salient) computer in the corner

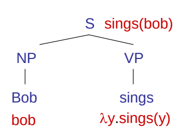
Truth-Conditional Semantics

- 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

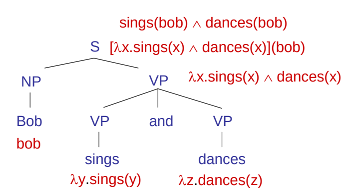
- Proper names:
  - Refer directly to some entity in the world
  - Bob : bob [[bob]]^W → ???
- Sentences:
  - Are either true or false (given how the world actually is)
  - Bob sings : sings(bob)



- 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 $e \rightarrow t$.
  - 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 : α VP : β (function application)
  - VP : $\lambda x . \alpha(x) \wedge \beta(x) \rightarrow$ VP : α and : $\varnothing$ VP : β (intersection)
- Example:

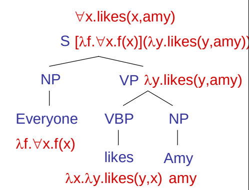


Denotation

- 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

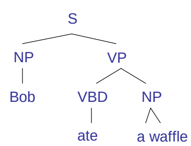
Other Cases

- 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

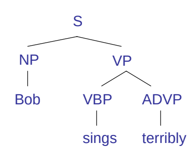
- Grounding
  - So why does the translation likes : λx.λy.likes(y,x) have anything to do with actual liking?
  - It doesn’t (unless the denotation model says so)
  - Sometimes that’s enough: wire up bought to the appropriate entry in a database
- Meaning postulates
  - Insist, e.g ∀x,y.likes(y,x) → knows(y,x)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

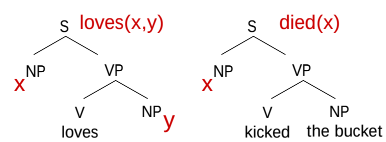
- Non-Intersective Adjectives
  - green ball : λx.[green(x) ∧ ball(x)]
  - fake diamond : λx.[fake(x) ∧ diamond(x)] ? ⟶ λx.[fake(diamond(x))
- Generalized Quantifiers
  - the : λf.[*unique-member*(f)]
  - all : λf. λg [∀x.f(x) → g(x)]
  - most?
  - Could do with more general second order predicates, 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 pumpkin in every window"
  - Multiple ways to work this out
    - Make it syntactic (movement)
    - Make it lexical (type-shifting)

Implementation, TAG, Idioms

- Add a "sem" feature to each context-free rule
  - S → NP loves NP
  - S[sem=loves(x,y)] → NP[sem=x] loves NP[sem=y]
  - Meaning of S depends on meaning of NPs
- TAG version:



- Template filling: S[sem=showflights(x,y)] →
  I want a flight from NP[sem=x] to NP[sem=y]

Modeling Uncertainty

- Gaping hole warning!
- Big difference between the syntax and semantics models presented here.

  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 symbol 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?)

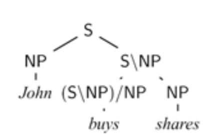
$John \vdash \mathrm{NP} : john'$

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

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

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

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



Reference Resolution

- Noun phrases refer to entities in the world, many pairs of noun phrases co-refer:

John Smith, CFO of Prime Corp. since 1986,

saw his pay jump 20% to $1.3 million

as the 57-year-old also became

the financial services co.'s president.

Kinds of Reference

- Referring expressions
  - *John Smith*
  - *President Smith*
  - *the president*
  - *the company's new executive*

More common in newswire, generally harder in practice

- Free variables
  - Smith saw *his pay* increase
- Bound variables
  - Every company trademarks its name.

More interesting grammatical constraints, more linguistic theory, easier in practice

Grammatical Constraints

- Gender / number
  - Jack gave Mary a gift. She was excited.
  - Mary gave her mother a gift. She was excited.
- Position (cf. binding theory)
  - The company's board polices itself / it.
  - Bob thinks Jack sends email to himself / him.
- Direction (anaphora vs. cataphora)
  - She bought a coat for Amy.
  - In her closet, Amy found her lost coat.

Discourse Constraints

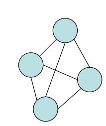
- Recency
- Salience
- Focus
- Centering Theory [Grosz et al. 86]

Other Constraints

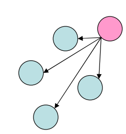
- Style / Usage Patterns
  - *Peter Watters* was named CEO. *Watters'* promotion came six weeks after his brother, *Eric Watters*, stepped down.
- Semantic Compatibility
  - Smith had bought *a used car* that morning. *The used car dealership* assured him it was in good condition.

Two Kinds of Models

- Mention Pair models
  - Treat coreference chains as a collection of pairwise links
  - Make independent pairwise decisions and reconcile them in some way (e.g. clustering or greedy partitioning)



- Entity-Mention models
  - A cleaner, but less studied, approach
  - Posit single underlying entities
  - Each mention links to a discourse entity [Pasula et al. 03], [Luo et al. 04]



Mention Pair Models

- Most common machine learning approach
- Build classifiers over pairs of NPs
  - For each NP, pick a preceding NP or NEW
  - Or, for each NP, choose link or no-link
- Clean up non-transitivity with clustering or graph partitioning algorithms
  - E.g.: [Soon et al. 01], [Ng and Cardie 02]
  - Some work has done the classification and clustering jointly [McCallum and Wellner 03]
- Kind of a hack, results in the 50's to 60's on all NPs
  - Better number on proper names and pronouns
  - Better numbers if tested on gold entities
- Failures are mostly because of insufficient knowledge or features for hard common noun cases

Pairwise Features

Category	Features	Remark
Lexical	exact_strm	1 if two mentions have the same spelling; 0 otherwise
	left_subsm	1 if one mention is a left substring of the other; 0 otherwise
	right_subsm	1 if one mention is a right substring of the other; 0 otherwise
	acronym	1 if one mention is an acronym of the other; 0 otherwise
	edit_dist	quantized editing distance between two mention strings
	spell	pair of actual mention strings
	ncd	number of different capitalized words in two mentions
Distance	token_dist	how many tokens two mentions are apart (quantized)
	sent_dist	how many sentences two mentions are apart (quantized)
	gap_dist	how many mentions in between the two mentions in question (quantized)
Syntax	POS_pair	POS-pair of two mention heads
	apposition	1 if two mentions are appositive; 0 otherwise
Count	count	pair of (quantized) numbers, each counting how many times a mention string is seen
Pronoun	gender	pair of attributes of {female, male, neutral, unknown }
	number	pair of attributes of {singular, plural, unknown}
	possessive	1 if a pronoun is possessive; 0 otherwise
	reflexive	1 if a pronoun is reflexive; 0 otherwise

	Devtest		Feb02		Sep02	
Model	ACE-val(%)	ECM-F(%)	ACE-val(%)	ECM-F(%)	ACE-val(%)	ECM-F(%)
MP	89.8	73.2 (±2.9)	90.0	73.1 (±4.0)	88.0	73.1 (±6.8)
EM	89.9	71.7 (±2.4)	88.2	70.8 (±3.9)	87.6	72.4 (±6.2)

[Luo et al. 04]

An Entity Mention Model

- Example: [Luo et al. 04]
- Bell Tree (link vs. start decision list)
- Entity centroids, or not?
  - Not for [Luo et al. 04], see [Pasula et al. 03]
  - Some features work on nearest mention (e.g. recency and distance)
  - Others work on "canonical" mention (e.g. spelling match)
  - Lots of pruning, model highly approximate
  - (Actually ends up being like a greedy-link system in the end)