Natural Language Processing

Lecture 23—11/12/2015
Jim Martin

Today

- More Semantics
  - Review/Finish up compositional semantics

Semantic Analysis

- Semantic analysis is the process of taking in some linguistic input and assigning a meaning representation to it.
  - There a lot of different ways to do this that make more or less (or no) use of syntax
  - We're going to start with the idea that syntax does matter
    - The compositional rule-to-rule approach

Augmented Rules

- We'll accomplish this by attaching semantic formation rules to our syntactic CFG rules
  - One semantic rule for each syntactic rule.
- Abstractly

$$A \rightarrow \alpha_1 ... \alpha_n \quad \{f(\alpha_1.sem, ... \alpha_n.sem)\}$$

- This should be read as the semantics we attach to A can be computed from some function applied to the semantics of A's parts.

Example

- Easy parts…
  - NP -> PropNoun
  - PropNoun -> Frasca
  - PropNoun -> Franco
- Attachments
  {PropNoun.sem}
  {Frasca}
  {Franco}

Example

- S -> NP VP
- VP -> Verb NP
- Verb -> likes
- {VP.sem(NP.sem)}
- {Verb.sem(NP.sem)
- ???

$$\lambda x \lambda y \exists e Liking(e) \wedge Liker(e, y) \wedge Liked(e, x)$$

Lambda Forms

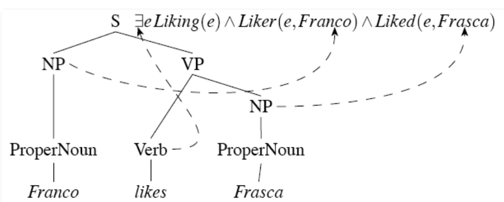
- A simple addition to FOL
  - Take a FOL sentence with variables in it that are to be bound.
  - Allow those variables to be bound by treating the lambda form as a function with formal arguments

$$\lambda x P(x)$$

$$\lambda x P(x)(Sally)$$

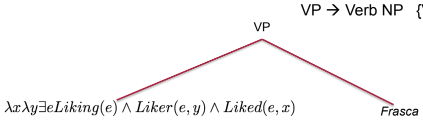
$$P(Sally)$$

Compositional Semantics by Lambda Application



Lambda Applications and Reductions

VP → Verb NP {Verb.sem(NP.Sem)}

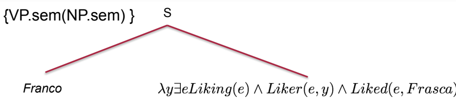


$$\lambda x \lambda y \exists e Liking(e) \wedge Liker(e, y) \wedge Liked(e, x)(Frasca)$$

$$\lambda y \exists e Liking(e) \wedge Liker(e, y) \wedge Liked(e, Frasca)$$

Lambda Applications and Reductions

S → NP VP {VP.sem(NP.sem) }



$$\lambda y \exists e Liking(e) \wedge Liker(e,y) \wedge Liked(e, Frasca)(Franco)$$

$$\exists e Liking(e) \wedge Liker(e, Franco) \wedge Liked(e, Frasca)$$

Complications

- You really ought to be suspicious that all those examples involve proper nouns that map to constants in the representation.
- That's the simplest possible case. Making it work for harder cases is more involved...
  - Mismatches between the syntax and semantics
    - Complex NPs with quantifiers

Complex NPs

- Things get quite a bit more complicated when we start looking at more complicated NPs
  - Such as...
    - *A menu*
    - *Every restaurant*
    - *Not every waiter*
    - *Most restaurants*
    - *All the morning non-stop flights to Houston*

Quantifiers

- Contrast...
  - *Frasca closed*

$\exists e\ Closed(e) \wedge ClosedThing(e, Frasca)$

- With
  - *Every restaurant closed*

$\forall x\ Restaurant(x) \Rightarrow \exists e\ Closed(e) \wedge ClosedThing(e, x)$

Quantifiers

Roughly, "every" in an NP like this is used to *stipulate something* about every member of *some class*. The NP is specifies the class. And somebody else is specifies the thing stipulated.... So the NP is a template-like thing

$$\forall x\ Restaurant(x) \Rightarrow Q(x)$$

The trick is going to be getting the Q to be right thing

Quantifiers

- Wrap a lambda around it...

$$\lambda Q.\forall x\ Restaurant(x) \Rightarrow Q(x)$$

- This requires a change to the kind (type) of things that we'll allow lambda variables to range over... Now it's both FOL predicates and terms.

Rules

$$NP \rightarrow Det\ Nominal \qquad \{Det.Sem(Nominal.Sem)\}$$

$$Det \rightarrow every \qquad \{\lambda P.\lambda Q.\forall x\, P(x) \Rightarrow Q(x)\}$$

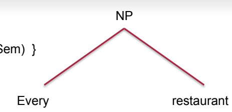
$$Nominal \rightarrow Noun \qquad \{Noun.sem\}$$

$$Noun \rightarrow restaurant \qquad \{\lambda x.Restaurant(x)\}$$

11/12/15 Speech and Language Processing - Jurafsky and Martin 16

Example

NP → Det Nominal { Det.Sem(Nominal.Sem) }



$$\lambda P.\lambda Q.\forall x\, P(x) \Rightarrow Q(x)(\lambda x.Restaurant(x))$$

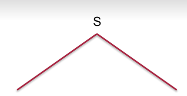
$$\lambda Q.\forall x\ \lambda x.Restaurant(x)(x) \Rightarrow Q(x)$$

$$\lambda Q.\forall x\ Restaurant(x) \Rightarrow Q(x)$$

11/12/15 Speech and Language Processing - Jurafsky and Martin 17

Every Restaurant Closed

S → NP VP { NP.Sem(VP.Sem) }



$$\forall x\ Restaurant(x) \Rightarrow \exists e\ Closed(e) \wedge ClosedThing(e,x)$$

$$\lambda Q.\forall x\ Restaurant(x) \Rightarrow Q(x) \qquad \lambda x.\exists e\ Closed(e) \wedge ClosedThing(e,x)$$

closed

$$\lambda P.\lambda Q.\forall x P(x) \Rightarrow Q(x) \qquad \lambda x.Restaurant(x)$$

Every *restaurant*

11/12/15 Speech and Language Processing - Jurafsky and Martin 18

Simple NP fix

- The semantics of proper nouns used to just be things that amounted to constants... *Franco*. Now they need to be a little more complex. This works
  - \lambda x Franco(x)

Revised

- Now all these examples should work
  - *Every restaurant closed.*
  - *Sunflower closed.*
- What about? $\exists x,e.Restaurant(x) \wedge Closing(e) \wedge Closed(e,x)$
  - *A restaurant closed.*
- This rule stays the same
  - NP --> Det Nominal
- Just need the semantic attachment for
  - Det --> *a*

Revised

- So if the template for "every" is

$$\{\lambda P.\lambda Q.\forall x\, P(x) \Rightarrow Q(x)\}$$

- Then the template for "a" should be what?

$$Det \rightarrow a \qquad \{\lambda P.\lambda Q.\exists x\, P(x) \wedge Q(x)\}$$

So Far So Good

- We can make effective use of lambdas to overcome
  - Mismatches between the syntax and semantics
  - While still preserving strict compositionality
- The style of the grammar is such that
  - Lexical items provide the bulk of the "content" of the representations
  - Grammar rules provide the instructions for how to put things together
    - Mainly in terms of which elements should be treated as functions and which are arguments.

Problem: Quantifier Ambiguity

- Contrast
  - *Every American has a governor.*

$$\forall x American(x) \implies \exists y Governor(y) \wedge GovernorOf(x,y)$$

  - *Every Coloradan has a governor.*

$$\exists x Governor(x) \wedge (\forall y Coloradan(y) \implies (GovernorOf(y,x))$$

- Given our current scheme which one do we get?

Problem

- Clearly these sentences have the same syntax. The only difference is in the words American and Coloradan. Words that have the same part of speech (lexical class) and very similar meanings.
- The fact that both interpretations are possible is an idiosyncratic fact about our political system. Not something in the language.

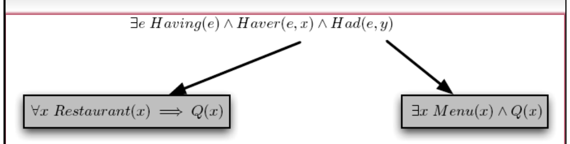
Problem

- Every restaurant has a menu.

$$\forall x\, Restaurant(x) \Rightarrow \exists y\, (Menu(y) \wedge \exists e\, (Having(e) \wedge Haver(e,x) \wedge Had(e,y)))$$

$$\exists y\, Menu(y) \wedge \forall x\, (Restaurant(x) \Rightarrow \exists e\, (Having(e) \wedge Haver(e,x) \wedge Had(e,y)))$$

What We Really Want



Captures the predicate argument structure and the relevant possibilities for the quantifiers. But underspecifies the final representation.

Store and Retrieve

- Now, given a representation like that we can get all the meanings out that we want by
  - Retrieving the quantifiers one at a time and placing them in front (again, using lambdas)
  - The order determines the scoping (the meaning).

Store

- The Store..

$$\exists e\, Having(e) \wedge Haver(e,s_1) \wedge Had(e,s_2)$$
$$(\lambda Q.\forall x\, Restaurant(x) \Rightarrow Q(x), 1),$$
$$(\lambda Q.\exists x\, Menu(x) \wedge Q(x), 2)$$

Retrieve

- Use lambda reduction to retrieve from the store and incorporate the arguments in the right way.
  - Retrieve element from the store and apply it to the core representation
  - With the variable corresponding to the retrieved element as a lambda variable
  - Huh?

Retrieve

- Example, pull out 2 first (that's s2) and apply it to the predicate representation.

$$\lambda Q.\exists x\,(Menu(x) \wedge Q(x))$$
$$(\lambda s_2.\exists e\, Having(e) \wedge Haver(e,s_1) \wedge Had(e,s_2))$$

$$\exists x\,(Menu(x) \wedge \exists e\, Having(e) \wedge Haver(e,s_1) \wedge Had(e,x))$$

Example

Then pull out S1 and apply it to the previous result.

$$\lambda Q.\forall x\,(Restaurant(x) \Rightarrow Q(x))$$
$$(\lambda.s_1 \exists y\,(Menu(y) \wedge \exists e\, Having(e) \wedge Haver(e,s_1) \wedge Had(e,x))$$

$$\forall x\, Restaurant(x) \Rightarrow \exists y\, Menu(y) \wedge \exists e\, Having(e) \wedge Haver(e,x) \wedge Had(e,y))$$

Ordering Determines Outcome

- Now if we had done it in the other order (first S1, and then S2) we could have gotten the other meaning (other quantifier scoping.