Computational Semantics

Deep Processing for NLP
Ling 571
February 8, 2016

Roadmap

- Motivation: Dialog Systems
- Key challenges
- Meaning representation
  - Representational requirements
  - First-order logic
    - Syntax & Semantics
  - Representing compositional meaning

Dialogue Systems

- User: What do I have on Thursday?
- Parse:
  - (S
  - (Q-WH-Obj
  - (Whwd What)
  - (Aux do)
  - (NP (Pron I))
  - (VP/NP (V have)
  - (NP/NP *t*)
  - (PP (Prep on)
  - (NP (N Thursday))))))

Dialogue Systems

- Parser:
  - Yes, it's grammatical!
  - Here's the structure!
- System: Great, but what am I supposed to DO?!
- Need to associate meaning with structure

Dialogue Systems

- (S
- (Q-WH-Obj Action: check; cal: USER; Date:Thursday
- (Whwd What)
- (Aux do)
- (NP (Pron I)) Cal: USER
- (VP/NP (V have)
- (NP/NP *t*)
- (PP (Prep on)
- (NP (N Thursday)))))) Date: Thursday

Natural Language

- Syntax: Determine the structure of natural language input
- Semantics: Determine the meaning of natural language input

Tasks for Semantics

- Semantic interpretation required for many tasks
  - Answering questions
  - Following instructions in a software manual
  - Following a recipe

- Requires more than phonology, morphology, syntax
- Must link linguistic elements to world knowledge

Semantics is Complex

- Sentences have many entailments, presuppositions
- *Instead, the protests turned bloody, as anti-government crowds were confronted by what appeared to be a coordinated group of Mubarak supporters*.
  - The protests became bloody.
  - The protests had been peaceful.
  - Crowds oppose the government.
  - Some support Mubarak.
  - There was a confrontation between two groups.
  - Anti-government crowds are not Mubarak supporters.
  - Etc..

Challenges in Semantics

- Semantic representation:
  - What is the appropriate formal language to express propositions in linguistic input?
    - E.g. predicate calculus
      - $\exists x.(dog(x) \land disappear(x))$

- Entailment:
  - What are all the valid conclusions that can be drawn from an utterance?
    - ‘Lincoln was assassinated’ entails ‘Lincoln is dead.’

Challenges in Semantics

- Reference: How do linguistic expressions link to objects/concepts in the real world?
  - ‘the dog’ , ‘the President’, ‘the Superbowl’

- Compositionality: How can we derive the meaning of a unit from its parts?
  - How do syntactic structure and semantic composition relate?
    - ‘rubber duck’ vs ‘rubber chicken’
    - ‘kick the bucket’

Tasks in Computational Semantics

- Computational semantics aims to extract, interpret, and reason about the meaning of NL utterances, and includes:
  - Defining a **meaning representation**
  - Developing techniques for **semantic analysis,** to convert NL strings to meaning representations
  - Developing methods for reasoning about these representations and performing inference from them

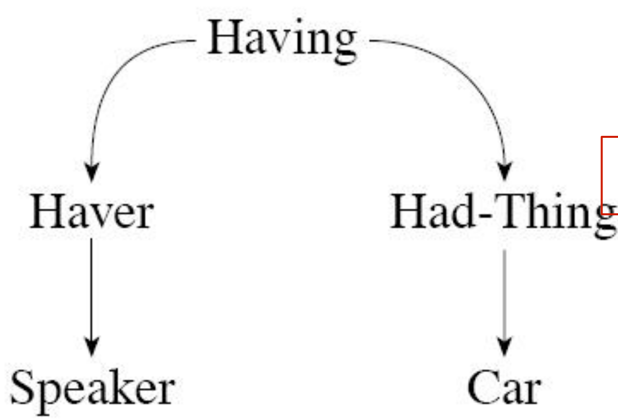
Complexity of Computational Semantics

- Requires:
  - Knowledge of language: words, syntax, relationships b/t structure and meaning, composition procedures
  - Knowledge of the world: what are the objects that we refer to, how do they relate, what are their properties?
  - Reasoning: Given a representation and a world, what new conclusions – bits of meaning – can we infer?
- Effectively AI-complete
  - Need representation, reasoning, world model, etc

Representing Meaning

$$\exists e, y\ Having(e) \wedge Haver(e, Speaker) \wedge HadThing(e, y) \wedge Car(y)$$

First-order Logic



Semantic Network

Conceptual Dependency

Car
$\Uparrow$ POSS-BY
Speaker

```
Having
 Haver:   Speaker
 HadThing:   Car
```

Frame-Based

Meaning Representations

- All consist of structures from set of symbols
  - Representational vocabulary
- Symbol structures correspond to:
  - Objects
  - Properties of objects
  - Relations among objects
- Can be viewed as:
  - Representation of meaning of linguistic input
  - Representation of state of world
- Here we focus on **literal** meaning

Representational Requirements

- Verifiability
  - Can compare representation of sentence to KB model
- Unambiguous representations
  - Semantic representation itself is unambiguous
- Canonical Form
  - Alternate expressions of same meaning map to same rep
- Inference and Variables
  - Way to draw valid conclusions from semantics and KB
- Expressiveness
  - Represent any natural language utterance

Meaning Structure of Language

- Human languages
  - Display basic predicate-argument structure
  - Employ variables
  - Employ quantifiers
  - Exhibit a (partially) compositional semantics

Predicate-Argument Structure

- Represent concepts and relationships
- Words behave like predicates:
  - Verbs, Adj, Adv:
    - Eat(John,VegetarianFood); Red(Ball)
- Some words behave like arguments:
  - Nouns: Eat(John,VegetarianFood); Red(Ball)
- Subcategorization frames indicate:
  - Number, Syntactic category, order of args

First-Order Logic

- Meaning representation:
  - Provides sound computational basis for verifiability, inference, expressiveness
- Supports determination of propositional truth
- Supports compositionality of meaning
- Supports inference
- Supports generalization through variables

First-Order Logic

- FOL **terms:**
  - **Constants:** specific objects in world*;*
    - *A, B, Maharani*
    - Refer to exactly one object; objects referred to by many
  - **Functions:** concepts refer to objects, e.g. Frasca's loc
    - *LocationOf(Frasca)*
    - Refer to objects, avoid using constants
  - **Variables:**
    - *x, e*

FOL Representation

- **Predicates:**
  - Relations among objects
    - *Maharani serves vegetarian food.* ➔
    - *Serves(Maharani, VegetarianFood)*
    - *Maharani is a restaurant.* ➔
    - *Restaurant(Maharani)*
- **Logical connectives:**
  - Allow compositionality of meaning
    - *Maharani serves vegetarian food and is cheap.*
    - *Serves(Maharani,VegetarianFood) ∧ Cheap(Maharani)*

Variables & Quantifiers

- Variables refer to:
  - Anonymous objects
  - All objects in some collection
- Quantifiers:
  - $\exists$: existential quantifier: “there exists”
    - Indefinite NP, one such object for truth
    - A cheap restaurant that serves vegetarian food

      $\exists x \, Restaurant(x) \wedge Serves(x, VegetarianFood) \wedge Cheap(x)$
  - $\forall$: universal quantifier: “for all”
    - All vegetarian restaurants serve vegetarian food.

$\forall x \, VegetarianRestaurant(x) \Rightarrow Serves(x, VegetarianFood)$

FOL Syntax Summary

$$
\begin{array}{rcl}
\textit{Formula} & \rightarrow & \textit{AtomicFormula} \\
& | & \textit{Formula Connective Formula} \\
& | & \textit{Quantifier Variable}, \ldots \textit{ Formula} \\
& | & \neg\, \textit{Formula} \\
& | & (\textit{Formula}) \\
\textit{AtomicFormula} & \rightarrow & \textit{Predicate}(\textit{Term}, \ldots) \\
\textit{Term} & \rightarrow & \textit{Function}(\textit{Term}, \ldots) \\
& | & \textit{Constant} \\
& | & \textit{Variable} \\
\textit{Connective} & \rightarrow & \wedge \mid \vee \mid \Rightarrow \\
\textit{Quantifier} & \rightarrow & \forall \mid \exists \\
\textit{Constant} & \rightarrow & A \mid \textit{VegetarianFood} \mid \textit{Maharani} \cdots \\
\textit{Variable} & \rightarrow & x \mid y \mid \cdots \\
\textit{Predicate} & \rightarrow & \textit{Serves} \mid \textit{Near} \mid \cdots \\
\textit{Function} & \rightarrow & \textit{LocationOf} \mid \textit{CuisineOf} \mid \cdots
\end{array}
$$

Compositionality

- **Compositionality**: The meaning of a complex expression is a function of the meaning of its parts and the rules for their combination.
  - Formal languages are compositional.
  - Natural language meaning is largely, though not fully, compositional, but much more complex.
    - How can we derive things like loves(John, Mary) from John, loves(x,y), and Mary?

Lambda Expressions

- Lambda (λ) notation: (Church, 1940)
  - Just like lambda in Python, Scheme, etc
  - Allows abstraction over FOL formulas
    - Supports compositionality
  - Form: λ + variable + FOL expression
    - E.g. $\lambda x.P(x)$ "Function taking x to P(x)"
    - $\lambda x.P(x)\ (A) \rightarrow P(A)$

λ-Reduction

- λ-reduction: Apply λ-expression to logical term
  - Binds formal parameter to term

$$\lambda x.P(x)$$

$$\lambda x.P(x)(A)$$

$$P(A)$$

- Equivalent to function application

Nested λ-Reduction

- Lambda expression as body of another

$$\lambda x.\lambda y.Near(x,y)$$

$$\lambda x.\lambda y.Near(x,y)(Bacaro)$$

$$\lambda y.Near(Bacaro,y)$$

$$\lambda y.Near(Bacaro,y)(Centro)$$

$$Near(Bacaro,Centro)$$

Lambda Expressions

- Currying;
  - Converting multi-argument predicates to sequence of single argument predicates

  - Why?
    - Incrementally accumulates multiple arguments spread over different parts of parse tree

Semantics of Meaning Rep.

- Model-theoretic approach:
  - FOL terms (objects): denote elements in a domain
  - Atomic formulas are:
    - If properties, sets of domain elements
    - If relations, sets of tuples of elements
- Formulas based on logical operators:

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$
False	*False*	*True*	*False*	*False*	*True*
False	*True*	*True*	*False*	*True*	*True*
True	*False*	*False*	*False*	*True*	*False*
True	*True*	*False*	*True*	*True*	*True*

- Compositionality provided by lambda expressions

Inference

- Standard AI-type logical inference procedures
  - Modus Ponens
  - Forward-chaining, Backward Chaining
  - Abduction
  - Resolution
  - Etc,..
- We'll assume we have a prover

Representing Events

- Initially, single predicate with some arguments
  - Serves(Maharani,IndianFood)
  - Assume # ags = # elements in subcategorization frame
- Example:
  - I ate.
  - I ate a turkey sandwich.
  - I ate a turkey sandwich at my desk.
  - I ate at my desk.
  - I ate lunch.
  - I ate a turkey sandwich for lunch.
  - I ate a turkey sandwich for lunch at my desk.

Events

- Issues?
  - Arity – how can we deal with different #s of arguments?

Neo-Davidsonian Events

- Neo-Davidsonian representation:
  - Distill event to single argument for event itself
  - Everything else is additional predication

$$\exists e Eating(e) \wedge Eater(e, Speaker) \wedge Eaten(e, TS) \wedge Meal(e, Lunch) \wedge Location(e, Desk)$$

- Pros:
  - No fixed argument structure
    - Dynamically add predicates as necessary
  - No extra roles
  - Logical connections can be derived

Meaning Representation for Computational Semantics

- Requirements:
  - Verifiability, Unambiguous representation, Canonical Form, Inference, Variables, Expressiveness
- Solution:
  - First-Order Logic
    - Structure
    - Semantics
    - Event Representation
- Next: Semantic Analysis
  - Deriving a meaning representation for an input

Summary

- First-order logic can be used as a meaning representation language for natural language
- Principle of compositionality: the meaning of a complex expression is a function of the meaning of its parts
- λ-expressions can be used to compute meaning representations from syntactic trees based on the principle of compositionality
- In the next section, we will look at a syntax-driven approach to semantic analysis in more detail

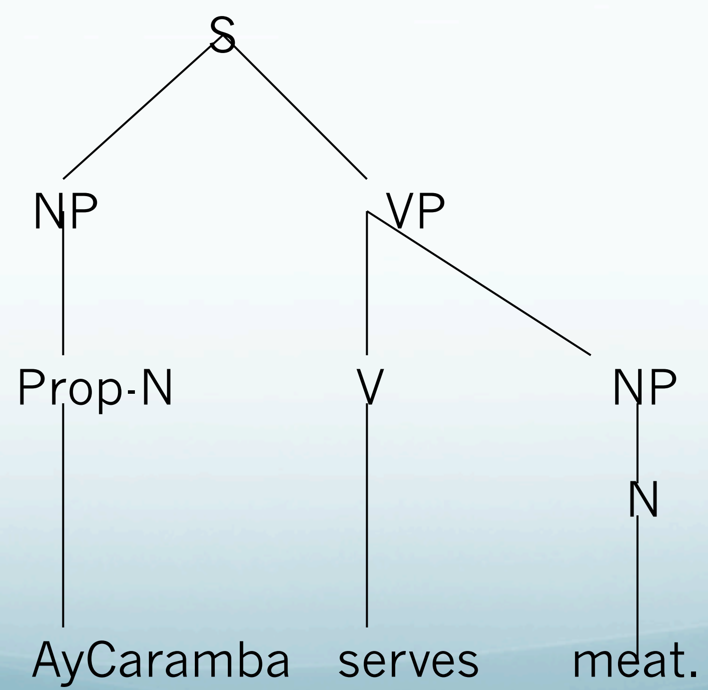
Syntax-driven Semantic Analysis

- Key: Principle of Compositionality
  - Meaning of sentence from meanings of parts
    - E.g. groupings and relations from syntax
- Question: Integration?
- Solution 1: Pipeline
  - Feed parse tree and sentence to semantic unit
  - Sub-Q: Ambiguity:
    - Approach: Keep all analyses, later stages will select

Simple Example

- AyCaramba serves meat.

$$\exists e\ Serving(e) \wedge Server(e, AyCaramba) \wedge Served(e, Meat)$$



Rule-to-Rule

- Issue:
  - How do we know which pieces of the semantics link to what part of the analysis?
  - Need detailed information about sentence, parse tree
    - Infinitely many sentences & parse trees
    - Semantic mapping function per parse tree → intractable
- Solution:
  - Tie semantics to finite components of grammar
    - E.g. rules & lexicon
  - Augment grammar rules with semantic info
    - Aka "attachments"
      - Specify how RHS elements compose to LHS

Semantic Attachments

- Basic structure:
  - A→ $a_1 \ldots a_n$ {f(a_j.sem,...a_k.sem)}
  - A.sem
- Language for semantic attachments
  - Arbitrary programming language fragments?
    - Arbitrary power but hard to map to logical form
    - No obvious relation between syntactic, semantic elements
  - Lambda calculus
    - Extends First Order Predicate Calculus (FOPC) with function application
  - Feature-based model + unification
- Focus on lambda calculus approach

Semantic Analysis Approach

- Semantic attachments:
  - Each CFG production gets semantic attachment
- Phrase semantics is function of SA of children
  - Complex functions parametrized
    - E.g. Verb → closed
      - Need unary predicate
        - One arg: subject, not yet available

Semantic Analysis Example

- Basic model:
  - Neo-Davidsonian event-style model
  - Complex quantification
- Example:
  - Every restaurant closed.
  - (S (NP (Det every) (Nom (Noun restaurant)))
  - (VP (V closed)))
  - Target representation:

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

Defining Representation

- Idea: Every restaurant = $\forall x \mathrm{Re}\, staurant(x)$
  - Good enough?
    - No: roughly 'everything is a restaurant'
    - Saying something about all restaurants – nuclear scope
- Solution: Dummy predicate
  $\forall x \mathrm{Re}\, staurant(x) \Rightarrow Q(x)$
  - Good enough?
    - No: no way to get Q(x) from elsewhere in sentence
- Solution: Lambda

$$\lambda Q.\forall x \mathrm{Re}\, staurant(x) \Rightarrow Q(x)$$

Creating Attachments

- Noun → restaurant $\{\lambda x.Restaurant(x)\}$
- Nom → Noun { Noun.sem }
- Det → Every $\{\lambda P.\lambda Q.\forall x P(x) \Rightarrow Q(x)\}$
- NP → Det Nom { Det.sem(Nom.sem) }

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

$$\lambda P.\lambda Q.\forall x P(x) \Rightarrow Q(x)(\lambda y.\mathrm{Re}\,staurant(y))$$

$$\lambda Q.\forall x \lambda y.\mathrm{Re}\,staurant(y)(x) \Rightarrow Q(x)$$

$$\lambda Q.\forall x\,\mathrm{Re}\,staurant(x) \Rightarrow Q(x)$$

Full Representation

- Verb → close $\{\lambda x.\exists e Closed(e) \wedge ClosedThing(e,x)\}$
- VP → Verb { Verb.sem }
- S → NP VP { NP.sem(VP.sem) }

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

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

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