CSE842: Natural Language Processing

Lecture 13: Meaning Representation and Semantic Analysis

Meaning

- So far, we have focused on the structure of language – not on what things ***mean***
- We have seen that words have different meaning, depending on the context in which they are used
- Everyday language tasks that require some semantic processing
  - Answering an essay question on an exam
  - Deciding what to order at a restaurant by reading a menu
  - Realizing that you've been insulted
  - …

Meaning

- Now, look at meaning representations -- representations that link linguistic forms to knowledge of the world.
- We are going to cover:
  - What is the meaning of a word
  - How can we represent the meaning
  - What formalisms can be used

First Order Predicate Calculus (First Order Logic)

- FOPC: provides a sound computational basis for the verifiability, inference, and expressiveness requirements
  - Supports the determination of truth
  - Supports compositionality of meaning
  - Supports representation of variables
  - Supports inference

FOPC Syntax

- **Terms:** constants, functions, variables
  - **Constants**: objects in the world, e.g. *Maharani*
  - **Functions:** concepts, e.g. *LocationOf(Maharani)*
  - **Variables:** x, e.g. *LocationOf(x)*
- **Predicates:** symbols that refer to relations that hold among objects in some domain or properties

  Serves(Maharani, VegetarianFood)

  Restaurant(Maharani)

Atomic sentences

Atomic sentence = $predicate(term_1, ..., term_n)$
or $term_1=term_2$

Term = $function\ (term_1, ..., term_n)$
or *constant* or *variable*

e.g., StudentAt(John, MSU)
MajorOf(John) = CSE

Complex sentences

Complex sentences are made from atomic sentences using connectives

$\neg S$, $S_1 \wedge S_2$, $S_1 \vee S_2$, $S_1 \Rightarrow S_2$, $S_1 \Leftrightarrow S_2$

e.g., StudentAt(John, MSU) ∧ (MajorOf(John) = CSE)

FOPC Syntax

- Logical connectives permit compositionality of meaning

I only have five dollars and I don't have a lot of time

Have(Speaker, FiveDoallars) ∧¬Have(Speaker, LotofTime)

FOPC Semantics

- Sentences in FOPC can be assigned truth values, T or F, based on whether the propositions they represent are T or F in the world knowledge
  – Atomic formulae are T or F based on their presence or absence in a DB
  – Composed meanings are inferred from DB and meaning of logical connectives

Variables and Quantifiers

Existential quantification ($\exists$): "There exists",

a restaurant that serves Mexican food near ICSI

$\exists x$ *Restaurant(x)* $\wedge$ *Serves(x, MexicanFood)*
$\wedge$*Near((LocationOf(x), LocationOf(ICSI)*

for this logical formula to be true there must be at least one object such that if we were substitute it for the variable x, the resulting formula is true.

Variables and Quantifiers

Universal quantification ($\forall$): "for all",

All vegetarian restaurants serve vegetarian food

$\forall x$ *VegetarianRestaurant(x) => Serves(x, VegetarianFood)*

for this logical formula to be true the substitution of any object in the knowledge base for the universally quantifier variable should result in a true formula.

Lambda Calculus

- Extension to FOPC (Church 1940)

  Expressions of the form $\lambda x P(x)$

  We can apply a λ-expression to logical terms to yield new FOPC expressions – by binding parameters to specified terms
- Lambda binding
  – Apply lambda-expression to logical expressions to bind lambda-expression's parameters (**lambda reduction**)
  – Simple process: substitute logical expressions for variables in lambda expression

  $\lambda x P(x)(A) \Rightarrow P(A)$, i.e., $\lambda x P(x)(house) \Rightarrow P(house)$

Lamda Notation

- λ-expressions: *λx.P(x)*
- λ-reduction: a process to replace λ variables with the specified FOL terms:

  λx.P(x) (A) => P(A)
- Embedded λ-expressions:

  λx.λy.Near(x,y)

  λx.λy.Near(x,y) (MSU)

  λy.Near(MSU,y) (StateCapital)

  Near(MSU, StateCapital)

Inference

- Modus ponens:

$$\frac{\alpha \quad \alpha\Rightarrow\beta}{\beta}$$

- Forward chaining
  – If α is true and α⇒β, then β is true
- Backward chaining
  – If α⇒β is true, then β is true if α is true. -> Prolog
  – Is different from reasoning backwards from known consequents to unknown antecedents
  – α⇒β and β, then α (abduction, plausible reasoning)

Event and State Representation

Start with an example:

1. I ate.
2. I ate a turkey sandwich.
3. I ate a turkey sandwich at my desk.
4. I ate at my desk.
5. I ate lunch.
6. I ate a turkey sandwich for lunch.
7. I ate a turkey sandwich for lunch at my desk.

How to represent "eat"? How to represent the variable number of arguments?

Event and State Representation

One possible solution:

Eating1(Speaker).

Eating2(Speaker, TurkeySandwich)

Eating3(Speaker, TurkeySandwich, Desk)

Eating4(Speaker, Desk)

Eating5(Speaker, Lunch)

Eating6(Speaker, TurkeySandwich, Lunch)

Eating7(Speaker, TurkeySandwich, Lunch Desk)

What are the problems with this representation?

Reification and neo-Davidsonian

Reification: elevate events to entities that can be quantified over: add event variable.

neo-Davidsonian event representation:

- Represent event using a single argument that stands for the event itself.
- Everything else (e.g., arguments of the event) is captured by additional predication.

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

Representing Time

- How do we represent time and temporal relationships between events?

  Last year Martha Stewart was happy but soon she will be sad.
- Where do we get temporal information?
  - Verb tense
  - Temporal expressions
  - Sequence of presentation

Temporal Variables

I arrived in New York

$$\exists e, i, n\ Arriving(e) \wedge Arriver(e, Speaker) \wedge Destination(e, NY) \wedge IntervalOf(e, i) \wedge EndPoint(e, n) \wedge Precedes(e, Now)$$

I am arriving in New York

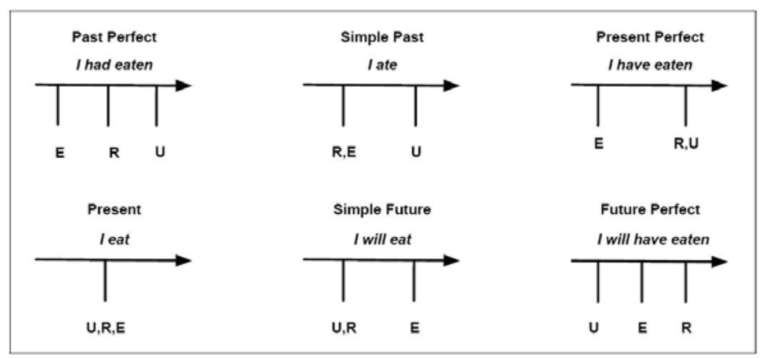
$$\exists e, i, n\ Arriving(e) \wedge Arriver(e, Speaker) \wedge Destination(e, NY) \wedge IntervalOf(e, i) \wedge MemberOf(e, Now)$$

I will arrive in New York

$$\exists e, i, n\ Arriving(e) \wedge Arriver(e, Speaker) \wedge Destination(e, NY) \wedge IntervalOf(e, i) \wedge EndPoint(e, n) \wedge Precedes(Now, e)$$

Linear representations (Reichenbach '47)

Reference point is separated from utterance time and event time



Verbs and Event Types: Aspect

- **Statives**: states or properties of objects at a particular point in time

  Mary needs sleep.

  **Mary is needing sleep. *Need sleep. *Mary needs sleep in a week.*

- **Activities**: events with no clear endpoint

  Harry drives a Porsche.

  **Harry drives a Porsche in a week.*

Verbs and Event Types: Aspect

- **Accomplishments**: events with durations and endpoints that result in some change of state

  He booked me a reservation; United flew to NY.

  Marlon filled out the form.

  Marlon stopped filling out the form (Marlon did not fill out the form) vs. Harry stopped driving a Porsche (Harry still drove a Porsche …for a while)

- **Achievements**: events that change state but have no particular duration

  I found the key; Larry reached NY.

  **Larry stopped reaching NY.*

  **Larry reached NY for a few minutes.*

Beliefs

In addition to the "real world", there are hypothetical world. Words such as: *believe, want, imagine, know* create speaker's hypothetical world.

I believe that Mary ate British food.

FOPC: $\exists u, v$ *believing(u)* $\wedge$ *eating(v)* $\wedge$*believer(u, speaker)* $\wedge$ *believed_prop(u, v)* $\wedge$ *eater(v, Mary)* $\wedge$ *eaten(v, british_food)*

Any problem?

Beliefs

In addition to the "real world", there are hypothetical world. Words such as: *believe, want, imagine, know* create speaker's hypothetical world.

I believe that Mary ate British food.

FOPC: $\exists u, v$ *believing(u)* $\wedge$ *eating(v)* $\wedge$*believer(u, speaker)* $\wedge$ *believed_prop(u, v)* $\wedge$ *eater(v, Mary)* $\wedge$ *eaten(v, british_food)*

There is a problem: every part of the above has to be true in order for the whole formula to be true

But we don't know if this is true or not.

Beliefs

How about a representation like this:

believing(Speaker, Eating(Mary, Britishfood)

Beliefs

How about a representation like this:

believing(Speaker, Eating(Mary, Britishfood)

This is not FOPC, second argument is a formula not a term

Predicates in FOPC hold between the objects in the domain being modeled, not between the relations that hold among the objects in the domain. Therefore, FOPC lacks a meaningful way to assert relations about full propositions.

Solution

Introduce an operator, called *believes*, that takes two FOPC formulas as arguments, a formula for *believer*, and a formula for *believed proposition*.

$Believes(speaker, \exists v\ isa(v, eating) \wedge eater(v, Mary) \wedge eaten(v, british\_food))$

Modal operators: *believe, want, imagine, know*

Modal Logic: is a logic augmented with modal operators

Semantic Analysis

Review

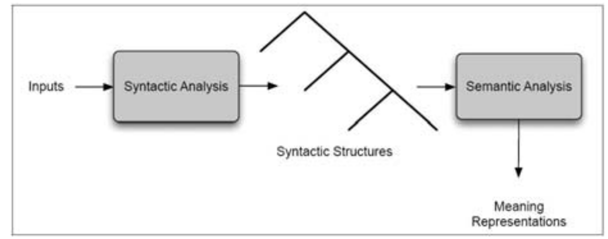
- Some representations of meaning:
  - First order logic
  - Frames
  - etc.
- Some linguistically relevant categories we want to represent
  - Predicates, arguments, variables, quantifiers
  - events, time, aspect
- Today: How can we derive these meaning representations?

Semantic Analysis

Assigning meaning representation to linguistic inputs

Several approaches:

- Syntax-driven semantic analysis
- Semantic grammars
- Information extraction -> separate lecture!



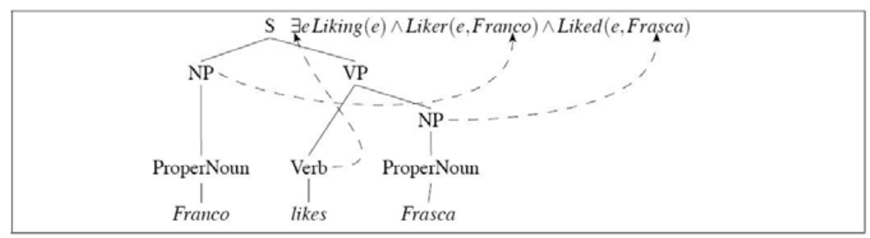
Syntax-Driven Semantic Analysis

- Key ideas: Principle of Compositionality (Frege)
  - The meaning of a sentence is composed by the meaning of its parts

  Which parts? Words?
- How to make this principle useful?
  - Account for the meaning not solely for the words, but also
    - Ordering
    - Grouping
    - Relations among the words

  } Syntactic Components and Relations

Example



Specific vs. General-Purpose Rules

- We don't want to have to specify for every possible parse tree what semantic representation it maps to
- We want to identify general mappings from parse trees to semantic representations:
  - Again (as with feature structures) we will augment the lexicon and the grammar
  - **Rule-to-rule hypothesis:** a mapping exists between rules of the grammar and rules of semantic representation

Solution: Semantic Augmentations to Context-Free Grammars

- Augmenting context-free grammar rules with semantic attachments
- What are attachments?
  - Instructions that specify how to compute the meaning representation of a construction from the meanings of its consistent parts
- Abstract:

$A \rightarrow \alpha_1 \ldots \alpha_n$ syntactic constituent

$\{f(\alpha_j sem, \ldots, \alpha_k sem)\}$ semantic attachment

A.sem can be computed by running the function f on some subset of the semantic attachments of A's constituents.

Semantic Attachments

- Extend each grammar rule with instructions on how to map the components of the rule to a semantic representation (grammars are getting complex)

  S → NP VP {VP.sem(NP.sem)}
- Each semantic function is defined in terms of the semantic representation of choice
- Problem: how to define these functions and how to specify their composition so we always get the meaning representation we want from our grammar?

Example

Maharani closed: closed(maharani)

- Associating constants with constituents
  - ProperNoun → Maharani {Maharani}
- Defining functions to produce these from input
  - NP → ProperNoun {ProperNoun.sem}
  - Assumption: meaning reps of children are passed up to parents for non-branching constituents
- Verbs here are where the action is
  - VP → Verb {Verb.sem}
  - Verb → closed {λx.Closed(x)}

How do we combine these pieces?

– S→ NP VP {VP.sem(NP.sem)}

λx.Closed(x) (Maharani)

Closed(Maharani)

Lambda Notation for Semantic analysis

- Semantic attachment to grammar rules consist primarily of λ-reduction
  - One element of an attachment serves as a functor and the rest serve as arguments to it
- Lexicon introduces meaning representation