Semantic Representations and Parsing

CSCI 1460: Computational Linguistics
Lecture 18
Announcements

• Final Project!
• Questions?
Topics

• “Semantic” Tasks: Executable Forms and NLI
• Formal Semantics
• Syntax-Semantics Interface and CCG
• Semantics and Deep Learning
Topics

- “Semantic” Tasks: Executable Forms and NLI
- Formal Semantics
- Syntax-Semantics Interface and CCG
- Semantics and Deep Learning
Semantics

• Syntax = the study of form
  • Why is “the cat ran past the dog” a good English sentence but “ran cat dog the the past” not?

• Semantics = the study of meaning
  • Why can’t I say “the cat ran past the dog” in order to mean that the dog ran past the cat?

• General goal is to describe precisely how we get from form to meaning
Semantics

- Generally:
  - Semantics = “sentence meaning” = tied to form/grammar
  - Pragmatics = “speaker meaning” = tied more generally to context
- In linguistics, these are kept separate
- In NLP, we tend to blur the distinction, and focus on specific tasks (e.g., “Alexa, I can’t hear the music” should be received as an instruction)
Tasks

• Arguably, (basically) all NLP tasks require representing “meaning”
  • BOW classifiers, pretrained language models, machine translation
• But there are two types of tasks which we generally focus on when we talk about “semantics”
  • Tasks that require executable (or logical) form
  • Reasoning about entailment
Executable (aka Logical) Form

• Explicit representation of natural language in formal language
• Question Answering over Databases:
  • “Show flights to Denver on Monday” -> \( \text{SELECT * FROM flights WHERE city = "Denver" and day = "Monday"} \)
• Robotics:
  • “Move forward past the sofa” -> \( \forall a \ (\text{move}(a) \land \text{dir}(a, \text{forward}) \land \text{past}(a, \text{Iy.sofa}(y))) \)
• Digital Personal Assistants:
  • “wake me up at 7” -> \( \text{set_alarm}(07:00, \text{GMT-5}) \)
Natural Language Inference (NLI) (aka Recognizing Textual Entailment, or RTE)

- Given a premise $p$ and a hypothesis $h$, does $p$ entail $h$?
- E.g.,
  - Between March and June, scientific observers say, up to 300,000 seals are killed. In Canada, seal-hunting means jobs, but opponents say it is vicious and endangers the species, also threatened by global warming $\rightarrow$ Hunting endangers seal species. (Dagan, 2006)
  - At 8:34, the Boston Center controller received a third transmission from American 11 $\rightarrow$ At 8:34, the Boston Center controller received a third transmission from American 11 (Williams et al, 2018)
Natural Language Inference (NLI) (aka Recognizing Textual Entailment, or RTE)

• Assumed to be a subtask required for many other tasks
  • Question answering, summarization, information retrieval
• Now widely used as a general-purpose evaluation task for systems of “understanding”
• The field has…mixed feelings…about its usefulness as a task
Topics

- “Semantic” Tasks: Executable Forms and NLI
- **Formal Semantics**
- Syntax-Semantics Interface and CCG
- Semantics and Deep Learning
NLP: A brief history

1910
1930
1950
1970
1990
2010
2020

1936: Church-Turing Thesis
1956: Dartmouth Workshop
1962: ACL Founded
1965: Chomsky “Aspects of a Theory of Syntax”

Logic and Computation: Tarski, Church, Turing

Turing Test

Formal Linguistics: Montague, Chomsky

1956: Dartmouth Workshop

1962: ACL Founded

1988: IBM Model 1

Statistical NLP: “Traditional” ML, standardized evals

Early NLP: Mix of rule-based and info-theory methods

End-to-end deep learning

Pretraining and Transfer Learning

Neural networks for cogsci and AI

2012: AlexNet

2018: ELMo
Formal Semantics

The basic aim of semantics is to characterize the notion of a true sentence (under a given interpretation) and of entailment.

(Richard Montague)
There is in my opinion no important theoretical difference between natural languages and the artificial languages of logicians; indeed I consider it possible to comprehend the syntax and semantics of both kinds of languages with a single natural and mathematically precise theory.

(Richard Montague)
Formal Semantics
Model Theory 101

“x > 17”
"x > 17"  ✔

"x" = 32
Formal Semantics
Model Theory 101

“x > 17” ✗

“x” = 5

x := 5
“$x > 17$”

“$x$” = 5
“$x$” = 17
“$x$” = 208
“$x$” = -1956
“$x$” = 0.457
Formal Semantics
Model Theory 101

Language "x > 17"

"x" = 5
"x" = 208
"x" = 17
"x" = 0.457
"x" = -1956
Formal Semantics
Model Theory 101

Language "x > 17"

\[ x = 5 \]
\[ x = 208 \]
\[ x = 17 \]
\[ x = -1956 \]

The World = 0.457
Formal Semantics
Model Theory 101

Language “x > 17”
Formal Semantics
Model Theory 101

Language

\[ x > y \]

\[ y > z \]

\[ x > z \]

The World (TBD)
Formal Semantics
Model Theory 101

Language

\[ x > y \]

\[ y > z \]

\[ \frac{\text{Variables (to be grounded)}}{x > z} \]

The World (TBD)
Formal Semantics
Model Theory 101

Language

\[
\begin{align*}
  x & > y \\
  y & > z \\
  \hline
  x & > z
\end{align*}
\]

Relations (defined)

The World (TBD)
A premise (p) entails a hypothesis (h) iff, in every possible world in which p is true, h is also true.

\[ \forall \mathcal{I}( (\mathcal{I} \models p) \Rightarrow (\mathcal{I} \models h) ) \]
Formal Semantics
Model Theory 101

\[
x > y \\
\quad z > w \\
\hline
\quad x > w
\]

\[
x = 10 \quad y = 5 \quad z = 11 \quad w = 8
\]
Formal Semantics
Model Theory 101

\[
\begin{align*}
  x & > y \\
  z & > w \\
\end{align*}
\]

\[
\begin{align*}
  x & > w \\
\end{align*}
\]

\[
\begin{align*}
x & = 10 \quad y = 5 \\
z & = 11 \quad w = 8
\end{align*}
\]
Formal Semantics
Model Theory 101

\[ x > y \]
\[ z > w \]
\[ x > w \]

\[ x = 10 \quad y = 5 \quad z = 11 \quad w = 8 \]
Formal Semantics
Model Theory 101

\[
\begin{align*}
\text{x} & > \text{y} \\
\text{z} & > \text{w} \\
\text{x} & > \text{w}
\end{align*}
\]
Formal Semantics
Model Theory 101

\[
\begin{align*}
  x &> y \quad \textbf{✗} \\
  z &> w \\
  \hline
  x &> w \\
  \hline
  x = 6 \\
  y = 5 \\
  z = 11 \\
  \omega = 8
\end{align*}
\]
the notion of a true sentence (under a given interpretation)

“Broca is a bird”
the notion of a true sentence (under a given interpretation)

“Broca is a bird” ✔
the notion of a true sentence
(under a given interpretation)

“Broca is a bird”
the notion of entailment

“All birds are gray”
“Broca is a bird”

“Broca is gray”
Formal Semantics
Truth Conditions and Truth Values

“Broca is a bird”
Truth conditions specify what the world needs to be like for the sentence to be true.

"Broca is a bird"
Formal Semantics
Truth Conditions and Truth Values

“Broca is a bird”

Truth value says whether or not a sentence is true (give some specific state of the world)
Formal Semantics

Truth Conditions and Truth Values

• Truth Conditional (or “Intentional”) Semantics: the meaning of a sentence is its truth conditions
  • understanding the meaning of “I have a bag of potatoes in my cupboard” does not require knowing whether I have potatoes in my cupboard
• Contrast with Denotational Semantics: the meaning of a sentence is its truth value
  • i.e., “My mom’s name is Karin” and “3 is half of 6” mean the same thing
  • ...?
• Formal semantics uses truth conditional
Formal Semantics
“The Fregean Program”

• Goal is to give an unambiguous account of the mapping for form to meaning
  • Input: A (syntactically parsed) string of words
  • Output: A context-independent logical form (e.g., lambda calculus, first order logic, etc)
• This program is complex and very rich—take CLPS 1342 (and 1341!)
Formal Semantics
Semantic Types
Formal Semantics
Semantic Types

• Want language to be a (formal) abstraction over “the world”, so only two primitive types:
  • e is the semantic type of entities
  • t is the semantic type of truth values
• Other types are defined recursively:
  • If a is a semantic type and b is a semantic type, then <a, b> is a semantic type
The Fregean Program
Semantic Types

• Referring expressions are type e: “Broca”, “Eddy”, “Ty”, “the cat on the mat”* …

• Propositional Sentences are type t: “Eddy is a cat”

• 1-place Predicates (adjectives, common nouns) are type <e, t>:

• $\llbracket \text{cat} \rrbracket =$
The Fregean Program
Semantic Types

• Referring expressions are type e: “Broca”, “Eddy”, “Ty”, “the cat on the mat”* ... 

• Propositional Sentences are type t: “Eddy is a cat”

• 1-place Predicates (adjectives, common nouns) are type <e, t>:

  \[
  \llbracket \text{cat} \rrbracket = \begin{bmatrix}
  \text{Broca} & \rightarrow & 0 \\
  \text{Eddy} & \rightarrow & 1 \\
  \text{Ty} & \rightarrow & 1
  \end{bmatrix}
  \]
The Fregean Program
Semantic Types

• Referring expressions are type e: “Broca”, “Eddy”, “Ty”, “the cat on the mat”* ...

• Propositional Sentences are type t: “Eddy is a cat”

• 1-place Predicates (adjectives, common nouns) are type <e, t>:

  \[
  \begin{array}{c}
  \text{cat} \\
  \end{array}
  \]

  
  \[
  \begin{array}{c}
  \text{Broca} \rightarrow 0 \\
  \text{Eddy} \rightarrow 1 \\
  \text{Ty} \rightarrow 1 \\
  \end{array}
  \]

  (Often thought of as sets, i.e. the “characteristic function of a set”)
The Fregean Program
Semantic Types

• 2-place Predicates (transitive verbs) are ???
The Fregean Program
Semantic Types

- 2-place Predicates (transitive verbs) are type $\langle e, \langle e, t \rangle \rangle$:
  - Takes entity as input, returns function (/set)
The Fregean Program
Semantic Types

• 2-place Predicates (transitive verbs) are type <e,<e,t>>:

\[
\text{[likes]} =
\begin{array}{c}
\text{Broca} \\
\text{Ty} \\
\text{Eddy}
\end{array}
\begin{array}{c}
\begin{array}{c}
\text{Broca} \rightarrow \text{1} \\
\text{Eddy} \rightarrow \text{0} \\
\text{Ty} \rightarrow \text{0}
\end{array} \\
\begin{array}{c}
\text{Broca} \rightarrow \text{1} \\
\text{Eddy} \rightarrow \text{1} \\
\text{Ty} \rightarrow \text{1}
\end{array} \\
\begin{array}{c}
\text{Broca} \rightarrow \text{1} \\
\text{Eddy} \rightarrow \text{1} \\
\text{Ty} \rightarrow \text{0}
\end{array}
\end{array}
\]

takes entity as input, returns function (/set)
The Fregean Program
Semantic Types

• Quantifiers (e.g. all, every) are ???
The Fregean Program

Semantic Types

- Quantifiers (e.g. all, every) are type $<e, t>, t>$

  takes a function $(\text{set})$ as input, returns a truth value.
  i.e., sets of sets
The Fregean Program
Semantic Types

• Quantifiers (e.g. all, every) are type <\langle e, t \rangle, t >
• every cat sleeps: \[1 \text{ if } \forall e \ (\text{cat}(e) \rightarrow \text{sleeps}(e)); \text{ else } 0\]
The Fregean Program
Semantic Types

- Quantifiers (e.g. all, every) are type \(<<e,t>, t>\)
- \(\left[ \text{every} \right](x)(y) = \ y \rightarrow 1 \text{ if } \forall e \ (x(e) \rightarrow y(e)); \text{ else } 0\)
The Fregean Program
Semantic Types

- Quantifiers (e.g. all, every) are type \(<<e, t>, t>\)
- \([\text{every}] = \lambda f \lambda g. \forall x f(x) \rightarrow g(x)\)
The Fregean Program

Semantic Types

• Quantifiers (e.g. all, every) are type $\langle\langle e, t \rangle, t \rangle$

$\boxed{\text{every}} = \lambda f \lambda g. \forall x f(x) \rightarrow g(x)$

bound variable
The Fregean Program
Compositionality

“Principle of Compositionality”: The meaning of the whole is a function of the meaning of the parts and the way in which they are combined.
The Fregean Program
Compositionality

Lexical Semantics

“Principle of Compositionality”: The meaning of the whole is a function of the meaning of the parts and the way in which they are combined.
The Fregean Program
Compositionality

**Lexical Semantics**

“Principle of Compositionality”: The meaning of the whole is a function of the meaning of the parts and the way in which they are combined.

**Syntax**
The Fregean Program
Compositionality

“Eddy likes Broca”
“Eddy likes Broca”

likes

Eddy  Broca
The Fregean Program
Compositionality

“Eddy likes Broca”

\[\langle \text{likes} \rangle_{<e,<e,t>} \rightarrow \text{Ty} \rightarrow \text{Eddy} \rightarrow \text{Broca} \rightarrow 1\]

\[\langle \text{Eddy} \rangle_{e} \rightarrow \text{Eddy} \rightarrow \text{Broca} \rightarrow 1\]

\[\langle \text{Broca} \rangle_{e} \rightarrow \text{Broca} \rightarrow \text{Eddy} \rightarrow \text{Ty} \rightarrow 0\]
The Fregean Program
Compositionality

“Eddy likes Broca”

\[
\text{likes} \langle e, <e, t> \rangle \\
\text{Eddy}_e \\
\text{Eddy} \\
\text{Broca}_e \\
\text{Broca} \\
\text{Broca} \rightarrow 1 \\
\text{Eddy} \rightarrow 0 \\
\text{Ty} \rightarrow 0 \\
\text{Broca} \rightarrow 1 \\
\text{Eddy} \rightarrow 1 \\
\text{Ty} \rightarrow 1 \\
\text{Broca} \rightarrow 1 \\
\text{Eddy} \rightarrow 1 \\
\text{Ty} \rightarrow 0
\]
The Fregean Program
Compositionality

“Eddy likes Broca”

⟦likes⟧<e,<e,t>>

Eddy

[Broca]e

Broca

Ty

Eddy
The Fregean Program
Compositionality

“Eddy likes Broca”

\[
\text{⟦\text{likes}\⟧}^{<e,<e,t>}(\text{Eddy})
\]

\[
\text{⟦\text{Broca}\⟧}^{e}
\]

\[
\text{Broca}
\]

\[
\text{Ty}
\]

\[
\text{Eddy}
\]
"Eddy likes Broca"

\[
\text{⟦Eddy likes⟧}_{e,t} \\ [Eddy likes]_e \\
\text{⟦Broca⟧}_e \\
\text{Broca}
\]
The Fregean Program
Compositionality

“Eddy likes Broca”

\[\langle e, t \rangle \]

\[\text{Broca} \rightarrow 1\]
\[\text{Eddy} \rightarrow 1\]
\[\text{Ty} \rightarrow 0\]
The Fregean Program
Compositionality

"Eddy likes Broca"

\[ [\text{Eddy likes}]_{<e,t>} \rightarrow \begin{cases} \text{Broca} & \rightarrow 1 \\ \text{Eddy} & \rightarrow 1 \\ \text{Ty} & \rightarrow 0 \end{cases} \]
The Fregean Program
Compositionality

“Eddy likes Broca”

\[ [Eddy \text{ likes}]_{<e,t>}(\text{Broca}) \rightarrow \begin{cases} \text{Broca} & \rightarrow 1 \\ \text{Eddy} & \rightarrow 1 \\ \text{Ty} & \rightarrow 0 \end{cases} \]
The Fregean Program
Compositionality

“Eddy likes Broca”

⟦Eddy likes Broca⟧ₜ  ---->  1
Topics

• “Semantic” Tasks: Executable Forms and NLI
• Formal Semantics
• Syntax-Semantics Interface and CCG
• Semantics and Deep Learning
Combinatory Categorial Grammar (CCG)

S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → Verb NP PP
Det → that | this | a
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-Noun → Houston | NWA
Aux → does
Preposition → from | to | on | near | through

CFG Phrase Structure Grammar
Combinatory Categorial Grammar (CCG)

S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → Verb NP PP

Det → that | this | a
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-Noun → Houston | NWA
Aux → does
Preposition → from | to | on | near | through

CFG Phrase Structure Grammar
Combinatory Categorial Grammar (CCG)

“Eddy” := NP : entity
Combinatory Categorial Grammar (CCG)

“Eddy” := NP : entity
Combinatory Categorial Grammar (CCG)

“Eddy” := NP : entity

Semantics
Combinatory Categorial Grammar (CCG)

“likes” := (S\NP)/NP: \( \lambda x.\lambda y.\text{likes}(y,x) \)
Combinatory Categorial Grammar (CCG)

“likes” := (S\NP)/NP: λx.λy.\text{likes}(y,x)

If I get an NP on my right

likes := (S\NP)/NP: \lambda x.\lambda y.\\text{likes}(y,x)

If I get an NP on my right
Combinatory Categorial Grammar (CCG)

“likes” := (S\NP)/NP: \lambda x.\lambda y.\text{likes}(y,x)

If I get an NP on my right and another one on my left
Combinatory Categorial Grammar (CCG)

“likes” := $(S \backslash NP) / NP$: $\lambda x. \lambda y. \text{likes}(y, x)$

If I get an NP on my right and another one on my left, I’ll make a sentence.
Combinatory Categorial Grammar (CCG)

“likes” := (S\NP)/NP: λx.λy.likes(y,x)

If I get an NP on my right and another one on my left, I’ll make a sentence.

If I get an argument x and another argument y, I’ll return some meaningful value.
Combinatory Categorial Grammar (CCG)

Principle of Combinatory Transparency

“likes” := (S\NP)/NP: λx.λy.likes(y,x)

If I get an NP on my right and another one on my left, I’ll make a sentence.

If I get an argument x and another argument y, I’ll return some meaningful value.
Eddy $\rightarrow$ NP: Eddy

Broca $\rightarrow$ NP: Broca

likes $\rightarrow$ (S\NP)/NP : $\lambda x.\lambda y.\text{likes}(y,x)$
Eddy → NP: Eddy
Broca → NP: Broca
likes → (S\NP)/NP : λx.λy.likes(y,x)
Combinatory Categorial Grammar (CCG)

Eddy → NP: Eddy
Broca → NP: Broca
likes → (S\NP)/NP : λx.λy.likes(y, x)
<table>
<thead>
<tr>
<th>Eddy</th>
<th>likes</th>
<th>Broca</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP: Eddy</td>
<td>(S\NP)/NP : \lambda x.\lambda y.\text{likes}(y,x)</td>
<td>NP: Broca</td>
</tr>
</tbody>
</table>

Eddy → NP: Eddy
Broca → NP: Broca
likes → (S\NP)/NP : \lambda x.\lambda y.\text{likes}(y,x)
Eddy $\rightarrow$ NP: Eddy
Broca $\rightarrow$ NP: Broca
likes $\rightarrow$ (S\NP)/NP : $\lambda x.\lambda y.\text{likes}(y,x)$
Combinatory Categorial Grammar (CCG)

Eddy → NP: Eddy
Broca → NP: Broca
likes → (S\NP)/NP : λx.λy.likes(y,x)
Topics

- “Semantic” Tasks: Executable Forms and NLI
- Formal Semantics
- Syntax-Semantics Interface and CCG
- Semantics and Deep Learning
Semantics and Deep Learning

• Broadly, two types of work happening
  1. Use deep learning to make better semantic parsers
  2. Skip semantic parsing entirely, assume it will be captured "latently" by the network
Semantics and Deep Learning
DL for better parsers

- E.g., treat English->Logical Form as a machine translation problem
- You can do this for your final project!
Semantics and Deep Learning
Treating Semantics as “Latent”

• Can we just train a model end-to-end to e.g., retrieve from a database or train a robot to navigate in an environment?
• Raw tokens/perception in -> correct behavior out?
• Lots of debate! (E.g., https://compositionalintelligence.github.io/)
• Are NN’s “compositional”? Do they need to be to capture semantics?
• More next lecture!