CS 4650/7650, Lecture 17: Compositional semantics

Jacob Eisenstein

October 24, 2013

Language has **meaning**, and many of the most exciting potential applications of natural language processing require uncovering that meaning:

- Telling a robot to *go make me a sandwich*
- Information extraction – filling in a database based on text such as *Born in Honolulu, Hawaii, Obama is a graduate of Columbia...*
- Question answering – understanding and responding to questions like *Where was the 44th President of the USA born?*
- Fact-checking an article
- Logic-checking an argument
- ...

The study of meaning is called semantics. Semantics has been a stumbling block for tasks that we've already seen, and other NLP tasks that we will soon see.

- For example, the text:

  I shot an elephant in my pajamas

  has two different interpretations, which correspond to two different parses.
- Lexicalized parsing is a hacky way to express our preference for one meaning over another.
- Semantic ambiguity also poses problems for tasks such as translation.

One might ask whether a computer can ever "really" understand language. One approach to this sort of conundrum is to apply some functional tests, and say that a computer understands when it can pass them:

- **Choose the appropriate action**. You say, *book a non-stop afternoon flight to the capital of Vermont*, and it does the right thing.

- **Answer reading comprehension tests.** You say *She arrived by bicycle*, and then it can answer questions like *How did she get there?* There's an EMNLP 2013 paper about doing this with multiple-choice questions [RBR13].
- **Determine whether a statement is true or false**. Related: determine whether one statement *entails* another, e.g., *She arrived by bicycle* entails *she did not drive*.
- **Translate** statements to some meaning representation.

Over the next few lectures, we'll talk about meaning at various levels of granularity:

- **Compositional** semantics, which treats the meaning of each unit of text as arising from the meaning of its subcomponents. Typically, we're interested in recovering some kind of logical representation.
- **Shallow** semantics, which focuses on identifying shallow predicate-argument structures.
- **Lexical** semantics, which deals with the meaning of individual words, and their relationships.

1 Representation of meaning

Most computational approaches semantics require some scheme for representating the meaning of language. The Jurafsky and Martin text [JM08] identifies five criteria for any such representation.

1.1 Verifiability

Statements can be checked against a database of facts.

- Suppose we have a statement like *Christopher Walken was born in Queens.*
- Is it true or false?
- In a verifiable representation, we can connect the representation of the statement above with a knowledge base of information.

1.2 No ambiguity

Every statement has **one meaning**.

This is necessary to reason about and act upon semantics — just like in programming languages. Natural language obviously fails this test:
I wanna eat someplace that's close to campus.

1.3 Canonical form

Every meaning has **one statement**.

Natural language obviously fails again...

- *Tin Drum serves vegetarian cuisine.*
- *Vegetarian dishes are offered at Tin Drum.*
- *There are vegetarian options at Tin Drum.*

Without a canonical representation,

- we must store each sentence as a separate fact
- or risk being unable to answer simple questions:
  Does Tin Drum serve vegetarian food?

Converting to canonical form requires:

- lexical semantics (*serves* = *offers*)
- structural analysis (*X is served by Y* = *Y serves X*)

1.4 Expressiveness

To be useful, our meaning representation must be able to express a wide range of ideas.

- Obviously we need the right predicates and constants (**non-logical vocabulary**):
  Does Tin Drum serve bi bim bap?
- But there are also more fundamental limits to the expressiveness of various representations, relating to statements like:
  - *All restaurants serve bi bim bap.*
  - *Most restaurants serve bi bim bap.*
  - *I think that Tin Dum serves bi bim bap.*
  - *Tin Drum used to serve bi bim bap.*

1.5 Inference

We need a way of making **inferences** from language.

- Some inferences are purely linguistic.

$$\frac{\begin{array}{c}\textit{some wugs are wags}\\ \textit{all wags are foo}\end{array}}{\textit{some wugs are foo}}$$

- Other inferences require world knowledge.

Christopher Walken was born in Queens

Christopher Walken was not born in Atlanta

Purely linguistic inference requires:

- a mapping from words to logical symbols
- compositional rules that allow us to build logical statements through the sentence
- variables (for some inferences)
- inference rules

World knowledge inference also requires:

- A way to ground the semantic representation in a **model** of the world.

We can perform linguistic inference through first-order logic and lambda calculus; there are other proposals, but this is the most ready for computational implementation. We can apply world knowledge through model-theoretic semantics, which ties logical symbols to a model of the world.

2 First-order logic

- **Terms** are the basic building blocks.
  - constants, e.g., Moe's, Tin-Drum
  - functions of other terms, e.g., LocationOf(Tin-Drum)
  - variables, e.g. x, y
- **Predicates** describe relations between terms: Serves(Moe's,burritos)
- **Formulae** are combinations of predicates, using
  - **connectives**, e.g., Serves(Moe's,burritos)$\wedge\neg$Expensive(Moe's)
  - **quantifiers**, e.g., $\exists x$Serves(Moe's, x)

2.1 Model-theoretic semantics

Model-theoretic semantics is the bridge between a representation and a "model" of the world.

- The *domain* of the model is a set of objects: *a,b,c,d,e,...*
  Each non-logical symbol has a *denotation* as an object: $[\![\text{Tin-Drum}]\!] = a$, $[\![\text{noodes}]\!] = e$

- *Relations* are sets of tuples: $\textsc{Serves} = \{\langle a,e\rangle, \langle b,e\rangle, \langle b,f\rangle, \ldots\}$
- *Properties* are relations between *objects* and *booleans*.
  Equivalently, properties are just sets of objects. $\textsc{cheap} = \{a, b\}$, $\textsc{noisy} = \{a, c\}$
- Many different logical statements have the same denotation:

$$\begin{aligned}&[\![\textsc{brother-of}(\textsc{lisa})]\!] = [\![\textsc{son-of}(\textsc{homer})]\!] \\ =&[\![\textsc{son-of}(\textsc{marge})]\!] = [\![\textsc{bart}]\!]\end{aligned}$$

2.2 Quantifiers and variables

Universal quantification

- *All restaurants that serve burritos are healthy.*
- $\forall x \textsc{Serves}(x, \textsc{burritos}) \Rightarrow \textsc{Healthy}(x)$

Existential quantification

- *There is a restaurant in Tech Square that serves burritos.*
- $\exists x \textsc{Serves}(x, \textsc{burrito}) \wedge \textsc{Location}(x, \textsc{TechSquare})$

Quantifier scope is funny! (sort of)

- *A woman gives birth every fifteen minutes.*
  We must find this woman and stop her.
- $\exists woman\ \forall 15min\ \textsc{gives-birth-during}(woman, 15min)$
- $\forall 15min\ \exists woman\ \textsc{gives-birth-during}(woman, 15min)$

2.3 Lambda notation

Lambda expressions are a way of describing "anonymous" functions.

- You may know them from functional programming, e.g.

$$\textsc{square} = \lambda x.x * x$$

- Lambda notation gives us a convenient way to talk about function composition, which will be critical for building meaning from syntactic structures.
  - *What vegetarian items are served at Moe's?*
  - Things that are vegetarian: $\lambda x.\textsc{Vegetarian}(x)$
  - Things that Moe's serves: $\lambda x.\textsc{Serves}(\textsc{Moes}, x)$
  - Vegetarian things that Moe's serves: $\lambda x.\textsc{Serves}(\textsc{Moes}, x) \wedge \textsc{Vegetarian}(x)$

3 Computational semantics

What was the point of doing parsing again? Arguably, to get semantics.

- Ideally, the meaning of a sentence decomposes into functions of the constituents, an idea attributed to Frege.
- **Montague** showed how to link syntactic parsing and semantic analysis of sentences.

3.1 An example

For example, suppose we have the sentence *Neal reads Proust*.

- The two nouns *Neal* and *Proust* have associated logical constants NEAL and PROUST.
- The verb *reads* corresponds to a logical function $\lambda x \lambda y \textsc{reads}(x, y)$
- We would like the meaning of the sentence to be $\textsc{reads}(\textsc{Neal}, \textsc{Proust})$.
- But how do we derive this, and not $\textsc{reads}(\textsc{Proust}, \textsc{Neal})$?

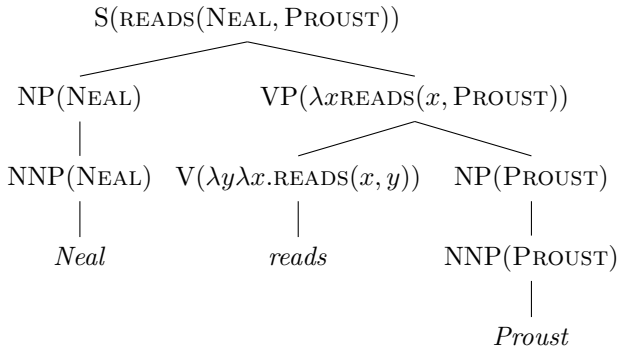
The answer is that we associate logical formulae not only with terminals (words), but also with the productions in the grammar!

NNP $\rightarrow$ *Neal*	$\{\textsc{Neal}\}$
NNP $\rightarrow$ *Proust*	$\{\textsc{Proust}\}$
V $\rightarrow$ *reads*	$\{\lambda y, \lambda x.\textsc{reads}(x, y)\}$
NP $\rightarrow$ NNP	$\{NNP.sem\}$
VP $\rightarrow$ V NP	$\{V.sem(NP.sem)\}$
S $\rightarrow$ NP VP	$\{VP.sem(NP.sem)\}$

- The first three lines define a **lexicon**. We'll need one line for every sense of every word.
- The nouns are obvious, but the transitive verb *reads* is more interesting. It takes two arguments through its lambda expressions, and applies READS to them.
- Next we define the meaning of productions. The simplest production is that of a proper noun from a noun phrase. The denotation of the NP should be identical to the denotation of the proper noun. Noun phrases with determiners and adjectives are more interesting.
- The verb phrase production applies the denotation of the verb to the denotation of the NP.

- The sentence production then applies the denotation of the verb phrase, this time to the denotation of the sentence.

We can now analyze the sentence, using the parse tree



Notice how the derivation left us with this interesting denotation for the verb phrase, $\lambda x.\textsc{reads}(x, \textsc{Proust}$. We could have lambda expressions in the analysis of a noun phrase. For example, *the students who read Proust* might mean something like

$$\lambda x.\textsc{student}(x) \wedge \textsc{read}(x, \textsc{Proust}) \tag{1}$$

3.2 A simpler example

Now let's think about what it would take to do a simpler example, like *Neal reads*. I argue:

- We need a production rule for intransitive verbs,

$$\text{VP} \rightarrow \text{V} \qquad \{V.sem\} \tag{2}$$

- We need a lexical entry for *reads* as an intransitive verb,

$$\text{V} \rightarrow \textit{reads} \qquad \lambda x.\textsc{reads}(x) \tag{3}$$

3.3 Noun phrases and quantifiers

What about an indefinite NP, like *a book*? The reading treats nouns as **properties**, and we can treat the indefinite article *a* as an *existential quantifier* (this is a little simpler than how determiners are handled in the reading). So we have to add the following lines to our fragment:

$$\begin{aligned} \text{DT} \rightarrow a \qquad & \{\lambda P.\exists x P(x)\} \\ \text{NP} \rightarrow \text{DT NP} \qquad & \{DT.sem(NP.sem)\} \end{aligned}$$

- Note that the meaning for *a* is a lambda expression that takes a predicate as an argument. If you've been trained in the more enlightened styles of functional programming, you do this all the time.

- We could easily introduce a lexical entry for *every*, with the translation $\lambda P.\forall x P(x)$.
- As you saw in the reading, *the* is more complicated; we define the meaning of *the* such that *the X* translates to "the unique item having property *X*."

4 Semantics in practice

There has been some success in rule-based semantic parsing, generally in closed domains.

- SHRDLU (Winograd 1971): classic AI system in blocksworld domain
- Chat-80 (Warren and Pereira 1982): question-answering about geographical facts

But the usual difficulties with rule-based NLP apply:

- It's hard to foresee and cover all the cases.
- Rules to resolve ambiguity are sometimes contradictory.
- Efforts to resolve disagreements between rules leads to complex software that is hard to maintain.

4.1 Learning Semantic Parsers

Zettlemoyer and Collins [ZC05] show that it is possible to learn semantic parsing from labeled examples. This work was first applied to parsing geographical queries:

- *what states border Texas*,
- $\lambda x.\textsc{state}(x) \wedge \textsc{borders}(x, \textsc{Texas})$
- *what states border the state that borders the most states?*
- *what is the largest state*

There are three components:

- Logical form, L
- Sentence, S
- Steps to parse, T

We can compute features $f(L, T, S)$ and build a log-linear model:

$$P(L, T|S) = \frac{\exp \boldsymbol{\theta}^\mathsf{T} \boldsymbol{f}(L, T, S)}{\sum_{L', T'} \exp \boldsymbol{\theta}^\mathsf{T} \boldsymbol{f}(L', T', S)} \tag{4}$$

But only L and S are observed in training data. We have to *marginalize* over derivations:

$$\arg\max_L P(L|S; \theta) = \arg\max_L \sum_T P(L, T|S; \theta)$$

This probability distribution was defined in terms of a log-linear model, with lexical features. We can find a maximum likelihood-solution by computing expected feature gradients using a dynamic programming algorithm similar to inside-outside.

This work is applied to a different syntactic formalism, called Combinatory Categorial Grammar (CCG). We had some reading on this earlier, but didn't have time to discuss it.

- In CCG, words have complex types that specify their syntactic behavior, and syntactic analysis applies a small number of combinatory operations on these types, such as forward application and composition.
- Augmenting word categories to include semantic information is fairly straightforward, and looks similar to our approach for augmenting CFGs with semantics.
- However, CCG is mildly-context sensitive, and therefore is more powerful than CFGs. It can successfully analyze sentences like *John baked and Mary ate the delicious pie.*

4.2 Semantics as a latent variable

Liang et al. [LJK13] showed that it is possible to learn even without knowing the logical form L! Instead, they have question-answer pairs, and marginalize over logical forms that would support the correct answer. Recent work at ACL 2013 and EMNLP 2013 has extended this from narrow domains like GeoQuery to open domain settings using Freebase.

References

[JM08] Daniel Jurafsky and James H. Martin. *Speech and Language Processing (2nd Edition)*. Pearson Prentice Hall, 2 edition, May 2008.

[LJK13] Percy Liang, Michael I Jordan, and Dan Klein. Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2):389–446, 2013.

[RBR13] Matthew Richardson, Christopher J.C. Burges, and Erin Renshaw. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 193–203, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

[ZC05] Luke S Zettlemoyer and Michael Collins. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. *Proc. of UAI*, 5, 2005.