Logic-based Semantics (and some IE)

Heavily based on slides by Greg Durett, Jason Eisner, Dan Jurafsky, Alexander Fraser
CS388: Natural Language Processing
Lecture 13: Semantics I

Greg Durrett

Slides adapted from Dan Klein, UC Berkeley
Where are we now?

- Early in the class: sentences are just sequences of words
- Now we can understand them in terms of tree structures as well
- Why is this useful? What does this allow us to do?
- We’re going to see how parsing can be a stepping stone towards more formal representations of language meaning
Today

- First-order logic
- Compositional semantics with first-order logic
- CCG parsing for database queries
- Lambda-DCS for question answering

Some challenges in semantic representation.

What do we do in practice.
Semantics

From Syntax to Meaning!
Programming Language Interpreter

- What is meaning of $3+5\times6$?
- First parse it into $3+(5\times6)$
What is meaning of $3 + 5 \times 6$?

First parse it into $3 + (5 \times 6)$

Now give a meaning to each node in the tree (bottom-up)
Interpreting in an Environment

- How about $3 + 5 \times x$?
- Same thing: the meaning of $x$ is found from the environment (it’s 6)
- Analogies in language?

![Tree diagram](image)
Compiling

- How about $3 + 5 \times x$?
- Don’t know $x$ at compile time
- “Meaning” at a node is a piece of code, not a number

$$5 \times (x+1) - 2$$ is a different expression that produces equivalent code (can be converted to the previous code by optimization)

Analogies in language?
What Counts as Understanding? some notions

- Be able to translate (a compiler is a translator ...)
  - Good definition? Depends on target language.
  - English to English? bah humbug!
  - English to French? reasonable
  - English to Chinese? requires deeper understanding
  - English to logic? deepest - the definition we’ll use!
    - all humans are mortal = ∀x [human(x) ⇒ mortal(x)]

- Assume we have logic-manipulating rules that then tell us how to act, draw conclusions, answer questions ...
What Counts as Understanding? some notions

- We understand if we can respond appropriately
  - ok for commands, questions (these demand response)
  - “Computer, warp speed 5”
  - “throw axe at dwarf”
  - “put all of my blocks in the red box”
  - imperative programming languages
  - database queries and other questions

- We understand a statement if we can determine its truth
  - If you can easily determine whether it’s true, why did anyone bother telling it to you?
  - Comparable notion for understanding NP is to identify what it refers to. Useful, but what if it’s out of sight?
What Counts as Understanding? some notions

- We understand statement if we know *how to determine its truth* (in principle!)
  - Compile it into a procedure for checking truth against the world
    - “All owls in outer space are bachelors”
      
      ```
      for every object
          if x is a owl
              if location(x) ∈ outerspace
                  if x is not a bachelor
                      return false
              return true
      ```

  - What if you don’t have a flying robot? *(Write the code anyway)*
  - How do you identify owls and bachelors? *(Assume library calls)*
  - What if space is infinite, so the procedure doesn’t halt? Same problem for “All prime integers ...” *(You won’t actually run it)*
What Counts as Understanding?

some notions

- We understand a statement if we know *how* one could (in principle) determine its truth
  - Compile it into a procedure that checks truth against the world
  - **Better:** Compile it into a mathematical *formula*
    - $\forall x \ owl(x) \land \text{outerspace}(x) \Rightarrow \text{bachelor}(x)$
  - Now you don’t have to worry about running it
  - Either true or false in the world: a mathematical question!
    - Statement claims that the world is such that this statement is true.
    - Auden (1956): “A sentence uttered makes a world appear
      Where all things happen as it says they do.”
  - But does this help? Can you check *math* against the *real world*?
    - What are the x’s that $\forall x$ ranges over? Which ones make owl(x) true?
    - **Model** the world by an infinite collection of facts and entities
      - Wittgenstein (1921): “The world is all that is the case. The world is
        the totality of facts, not of things.”
What Counts as Understanding?  
some notions

- We understand statement if we know *how one could (in principle) determine its truth*
  - Compile it into a procedure that checks truth against the world
  - Better: Compile it into a mathematical formula
    - $\forall x \text{ owl}(x) \land \text{ outerspace}(x) \Rightarrow \text{ bachelor}(x)$
  - Equivalently, be able to derive all logical consequences
    - What else is true in every world where this statement is true?
      - Necessary conditions – let us draw other conclusions from sentence
    - And what is false in every world where this sentence is false
      - Sufficient conditions – let us conclude the sentence from other facts
    - “Recognizing textual entailment” is an NLP task (∃ competitions!)
      - John ate pizza. Can you conclude that John opened his mouth?
  - Knowing consequences lets you answer questions (in principle):
    - **Easy:** John ate pizza. What was eaten by John?
    - **Hard:** White’s first move is P-Q4. Can Black checkmate?
Logic-based Semantics

- intro to $\lambda$-calculus and logical notation
- Let’s look at some sentences and phrases
  - What logical representations would be reasonable?
Logic: Some Preliminaries

Three major kinds of objects

1. Booleans
   - Roughly, the semantic values of sentences

2. Entities
   - Values of NPs, e.g., objects like this slide
   - Maybe also other types of entities, like times

3. Functions of various types
   - A function returning a boolean is called a “predicate” – e.g., frog(x), green(x)
   - Functions might return other functions!
   - Function might take other functions as arguments!
Logic: Lambda Terms

- Lambda terms:
  - A way of writing “anonymous functions”
    - No function header or function name
    - But defines the key thing: **behavior** of the function
    - Just as we can talk about 3 without naming it “x”
  - Let \( \text{square} = \lambda p \ p\times p \)
  - Equivalent to \( \text{int square}(p) \{ \text{return } p\times p; \} \)
  - But we can talk about \( \lambda p \ p\times p \) without naming it
  - Format of a lambda term: \( \lambda \) variable expression
Logic: Lambda Terms

- Lambda terms:
  - Let \( \text{square} = \lambda p \; p*p \)
  - Then \( \text{square}(3) = (\lambda p \; p*p)(3) = 3*3 \)
  - Note: \( \text{square}(x) \) isn’t a function! It’s just the value \( x*x \).
  - But \( \lambda x \; \text{square}(x) = \lambda x \; x*x = \lambda p \; p*p = \text{square} \)
    (proving that these functions are equal – and indeed they are, as they act the same on all arguments: what is \( (\lambda x \; \text{square}(x))(y) \)?)

- Let \( \text{even} = \lambda p \; (p \mod 2 == 0) \) a predicate: returns true/false
- \( \text{even}(x) \) is true if \( x \) is even
- How about \( \text{even}(\text{square}(x)) \)?
- \( \lambda x \; \text{even}(\text{square}(x)) \) is true of numbers with even squares
  - Just apply rules to get \( \lambda x \; (\text{even}(x*x)) = \lambda x \; (x*x \mod 2 == 0) \)
  - This happens to denote the same predicate as \( \text{even} \) does
Logic: Multiple Arguments

- Lambda terms denote functions of 1 argument
- But how about functions like multiplication?
- We can fake multiple arguments [“currying”]

- Define \textbf{times} as $\lambda x \lambda y \ (x*y)$
- Claim that times(5)(6) is 30
  - times(5) = ($\lambda x \lambda y \ x*y) \ (5) = \lambda y \ 5*y$
    - If this function weren’t anonymous, what would we call it?
  - times(5)(6) = ($\lambda y \ 5*y) \ (6) = 5*6 = 30
Logic: Multiple Arguments

- All lambda terms have one argument
- But we can fake multiple arguments ...

We’ll write “times(5,6)” as syntactic sugar for times(5)(6) or perhaps times(6)(5) Notation varies; doesn’t matter as long as you’re consistent

\[
times(5,6) = \times(5)(6)
= (\lambda x \; \lambda y \; x*y) \; (5)(6)
= (\lambda y \; 5*y)(6)
= 5*6 = 30
\]

- So we can always get away with 1-arg functions ...
  - ... which might return a function to take the next argument. Whoa.

- Remember: square can be written as \( \lambda x \; \text{square}(x) \)
  - And now times can be written as \( \lambda x \; \lambda y \; \times(x,y) \)
So what does \textit{times} actually mean???
\begin{itemize}
  \item \textit{times} was defined in terms of \textit{*}.
  \item But does \textit{*} mean multiplication?
  \item If \textit{*} was defined as another lambda term, then \textit{\texttt{times}(5,6) = *(5,6) = (blah\ blah\ blah)(5)(6)}
    but where do we stop?
\end{itemize}

Similarly, what does \textit{bachelor} mean?
\begin{itemize}
  \item Maybe we defined \textit{bachelor} = \lambda x (\texttt{male}(x) \textbf{and not} \texttt{married}(x))
    but how is \texttt{male} defined?
\end{itemize}

Same problem as in programming languages and dictionaries.
Grounding out

- As in programming languages: something has to be built in.
- Don’t keep doing substitutions forever!
  - Eventually we have to “ground out” in a primitive term
  - Primitive terms are bound to object code
- Maybe \( *(5,6) \) is handled by the hardware
- Maybe \texttt{male(John)} is too
- What code is executed by \texttt{loves(John, Mary)}?
Logic: Interesting Constants

- Thus, have “constants” that name some of the entities and functions (e.g., *):
  - BarackObama – an entity
  - red – a predicate on entities
    - holds of just the red entities: red(x) is true if x is red!
  - loves – a predicate on 2 entities
    - loves(BarackObama, MichelleObama)
    - Question: What does loves(MichelleObama) denote?
- Can define other named objects from the constants
- Can define a meaning for each English word from the named objects
- Meaning of each English word is defined in terms of the constants [maybe indirectly]
Logic:
Connectives & Quantifiers

- $p \lor q \ (= p \lor q) \quad \text{“p or q”}$
- $p \land q \ (= p \land q = p, q) \quad \text{“p and q”}$
- $\neg p \ (= \neg p = \sim p) \quad \text{“not p”}$
- $p \Rightarrow q \quad \text{“if p then q”}$
- $\forall x \quad \text{“for all x”}$
- $\exists x \quad \text{“there exists x”}$

“all pigs are big”
- $\forall x \ pig(x) \Rightarrow big(x) \quad \text{“for all x, if pig(x), then big(x)”}$

“some pig is big”
- $\exists x \ pig(x) \land big(x)$
  
  there exists some x such that pig(x) AND big(x)

“most pigs are big”
Logic:
Connectives & Quantifiers

- **most** – a predicate on 2 predicates on entities
  - \( \text{most}(\text{pig}, \text{big}) = \text{“most pigs are big”} \)
  - Equivalently, \( \text{most}(\lambda x \; \text{pig}(x), \lambda x \; \text{big}(x)) \)
  - returns true if most of the things satisfying the first predicate also satisfy the second predicate

- similarly for other quantifiers
  - \( \text{all}(\text{pig}, \text{big}) \) (equivalent to \( \forall x \; \text{pig}(x) \Rightarrow \text{big}(x) \))
  - \( \text{exists}(\text{pig}, \text{big}) \) (equivalent to \( \exists x \; \text{pig}(x) \; \text{AND} \; \text{big}(x) \))
  - can even build complex quantifiers from English phrases:
    - “between 12 and 75”; “a majority of”; “all but the smallest 2”
Logic in NLP

- Question answering:

  *Who are all the American singers named Amy?*
  
  $\lambda x. \text{nationality}(x, \text{USA}) \land \text{sings}(x) \land \text{firstName}(x, \text{Amy})$

- Function that maps from $x$ to true/false, like `filter`. Execute this on the world to answer the question

- Lambda calculus: powerful system for expressing these functions

- Information extraction: *Lady Gaga and Eminem are both musicians*
  
  $\text{musician}(\text{Lady Gaga}) \land \text{musician}(\text{Eminem})$

- Can now do reasoning. Maybe know:  $\forall x \text{musician}(x) \Rightarrow \text{performer}(x)$
  
  Then: $\text{performer}(\text{Lady Gaga}) \land \text{performer}(\text{Eminem})$
We’ve discussed what semantic representations should look like.

But how do we get them from sentences???

First - parse to get a syntax tree.
Second - look up the semantics for each word.
Third - build the semantics for each constituent
  - Work from the bottom up
  - The syntax tree is a “recipe” for how to do it
Parses to Logical Forms

\[ \text{sings(e470)} \]

\[
\text{S} \quad \text{ID} \\
\text{e470} \quad \text{NP} \quad \text{VP} \\
\text{NNP} \quad \text{NP} \quad \text{VBP} \\
\text{Lady} \quad \text{Gaga} \quad \text{sings} \\
\]

\[ \lambda y. \text{sings}(y) \]

function application: apply this to e470

\[ \lambda y. \text{sings}(y) \]

takes one argument (y, the entity) and returns a logical form \text{sings}(y)

- We can use the syntactic parse as a bridge to the lambda-calculus representation, build up a logical form \textit{compositionally}
Parses to Logical Forms

\[ \text{sings(e470)} \land \text{dances(e470)} \]

\[ S \]

\[ e470 \quad \text{NP} \quad \lambda y. \text{sings(y)} \land \text{dances(y)} \]

\[ \text{NNP} \quad \text{NNP} \quad \text{VP} \quad \text{VP} \]

\[ \text{Lady} \quad \text{Gaga} \quad \text{VP} \quad \text{CC} \quad \text{and} \quad \text{VP} \]

\[ \text{VBP} \quad \text{VBP} \]

\[ \text{sings} \quad \text{dances} \]

\[ \lambda y. \text{sings(y)} \quad \lambda y. \text{dances(y)} \]

- General rules:
  - VP: \( \lambda y. a(y) \land b(y) \) \( \rightarrow \) VP: \( \lambda y. a(y) \) CC VP: \( \lambda y. b(y) \)
  - S: \( f(x) \) \( \rightarrow \) NP: \( x \) VP: \( f \)
 Parses to Logical Forms

\[
\lambda x, \lambda y. \text{born}(y, x) \ 3/28/1986
\]

- Function takes two arguments: first x (date), then y (entity)
- How to handle tense: should we indicate that this happened in the past?
QA from Parsing

\( \lambda x. \text{born}(e470,x) \)

Execute this function against a knowledge base to answer the question

- Tricky to parse due to wh-movement...would be easier if we said
  
  Lady Gaga was born when
Semantic Parsing

- For question answering, syntactic parsing doesn’t tell you everything you want to know, but indicates the right structure.

- Solution: *semantic parsing*: many forms of this task depending on semantic formalisms.

- Two today: CCG (looks like what we’ve been doing) and lambda-DGS.
CCG Parsing
Combinatory Categorial Grammar

- Steedman+Szabolcsi 1980s: formalism bridging syntax and semantics
- Parallel derivations of syntactic parse and lambda calculus expression
- Syntactic categories (for this lecture): S, NP, “slash” categories
- S\NP: “if I combine with an NP on my left side, I form a sentence” — verb
- When you apply this, there has to be a parallel instance of function application on the semantics side
Combinatory Categorial Grammar

- Steedman+Szabolcsi 1980s: formalism bridging syntax and semantics
- Syntactic categories (for this lecture): S, NP, “slash” categories
  - $S \backslash NP$: “if I combine with an NP on my left side, I form a sentence” — verb
  - $(S \backslash NP)/NP$: “I need an NP on my right and then on my left” — verb with a direct object

\[
\begin{array}{c}
S \\
sings(e728)
\end{array}
\quad
\begin{array}{c}
S \\
borders(e101,e89)
\end{array}
\]

\[
\begin{array}{c}
NP \\
e728
\end{array}
\quad
\begin{array}{c}
S \backslash NP \\
λy.\,sings(y)
\end{array}
\quad
\begin{array}{c}
NP \\
e101
\end{array}
\quad
\begin{array}{c}
(S \backslash NP)/NP \\
λx.\,λy\,\text{borders}(y,x)
\end{array}
\quad
\begin{array}{c}
NP \\
e89
\end{array}
\]

Eminem

sings

Oklahoma

borders

Texas
CCG Parsing

\[
\frac{(S/(S\setminus NP))/NP}{S/(S\setminus NP)}
\]

\[
\lambda f.\lambda g.\lambda x.f(x) \land g(x)
\]

What

\[
\frac{N}{\lambda x.\text{state}(x)}
\]

states

\[
\frac{(S\setminus NP)/NP}{(S\setminus NP)}
\]

\[
\lambda x.\lambda y.\text{borders}(y, x)
\]

border

\[
\frac{NP}{\lambda y.\text{borders}(y, \text{texas})}
\]

Texas

\[
\frac{NP}{\lambda x.\text{state}(x) \land \text{borders}(x, \text{texas})}
\]

\[
\frac{S}{S}
\]

“What” is a **very** complex type: needs a noun and needs a S\setminus NP to form a sentence. S\setminus NP is basically a verb phrase (border Texas)

Lexicon is highly ambiguous — all the challenge of CCG parsing is in picking the right lexicon entries

Zettlemoyer and Collins (2005)
CCG Parsing

<table>
<thead>
<tr>
<th>Show me</th>
<th>flights</th>
<th>to</th>
<th>Prague</th>
</tr>
</thead>
<tbody>
<tr>
<td>S/N</td>
<td>N</td>
<td>(N/N)/NP</td>
<td>NP</td>
</tr>
<tr>
<td>λf.f</td>
<td>λx. flight(x)</td>
<td>λy. λf. λx. f(y) ∧ to (x, y)</td>
<td>PRG</td>
</tr>
</tbody>
</table>

- “to” needs an NP (destination) and N (parent)

Slide credit: Dan Klein
Building CCG Parsers

- Model: log-linear model over derivations with features on rules:

\[
P(d|x) \propto \exp w^\top \left( \sum_{r \in d} f(r, x) \right)
\]

\[
f\left(\begin{array}{c}
S \\
sings(e728)
\end{array}\right) = \text{Indicator}(S \to NP S\backslash NP)
\]

\[
f\left(\begin{array}{c}
NP \\
e728
\end{array}\right) f\left(\begin{array}{c}
S\backslash NP \\
\lambda y. \text{sings}(y)
\end{array}\right) = \text{Indicator}(S\backslash NP \to \text{sings})
\]

- Can parse with a variant of CKY

Zettlemoyer and Collins (2005)
Applications

- GeoQuery: answering questions about states (~80% accuracy)
- Jobs: answering questions about job postings (~80% accuracy)
- ATIS: flight search
- Can do well on all of these tasks if you handcraft systems and use plenty of training data: these domains aren’t that rich
- What about broader QA?
Complications

- Language is messier than that.
- Real text is longer and more involved.
A reasonable representation?

- Gilly swallowed a goldfish

- First attempt: `swallowed(Gilly, goldfish)`

- Returns true or false. Analogous to
  - `prime(17)`
  - `equal(4,2+2)`
  - `loves(BarackObama, MichelleObama)`
  - `swallowed(Gilly, Jilly)`

- ... or is it analogous?
A reasonable representation?

- Gilly swallowed a goldfish
  - First attempt: swallowed(Gilly, goldfish)
- But we’re not paying attention to a!
- goldfish isn’t the name of a unique object the way Gilly is

- In particular, don’t want
  Gilly swallowed a goldfish and Milly swallowed a goldfish
to translate as
  swallowed(Gilly, goldfish) AND swallowed(Milly, goldfish)
since probably not the same goldfish ...
Use a Quantifier

- Gilly swallowed a goldfish
  - First attempt: swallowed(Gilly, goldfish)
- Better: $\exists g \text{ goldfish}(g) \text{ AND swallowed}(Gilly, g)$
- Or using one of our quantifier predicates:
  - $\exists g \text{ goldfish}(g), \lambda g \text{ swallowed}(Gilly, g)$
  - Equivalently: $\exists \text{goldfish, swallowed}(Gilly))$
    - “In the set of goldfish there exists one swallowed by Gilly”
- Here goldfish is a predicate on entities
  - This is the same semantic type as red
  - But goldfish is noun and red is adjective .. #@!?
Tense

- Gilly swallowed a goldfish
  - Previous attempt: \( \text{exists(goldfish, } \lambda g \text{ swallowed(Gilly,} g)) \)

- Improve to use tense:
  - Instead of the 2-arg predicate \( \text{swallowed(Gilly,} g) \)
    try a 3-arg version \( \text{swallow(t,Gilly,} g) \) where \( t \) is a time
  - Now we can write:
    \[
    \exists t \text{ past}(t) \text{ AND } \text{exists(goldfish, } \lambda g \text{ swallow}(t,\text{Gilly,} g))
    \]
  - “There was some time in the past such that a goldfish was among the objects swallowed by Gilly at that time”
Gilly swallowed a goldfish

Previous attempt: \(\text{exists(goldfish, swallowed(Gilly))}\)

Improve to use tense:

Instead of the 2-arg predicate \(\text{swallowed(Gilly,g)}\)
try a 3-arg version \(\text{swallow(t,Gilly,g)}\)

Now we can write:

\[\exists t \quad \text{past}(t) \land \text{exists(goldfish, swallow(t,Gilly))}\]

“There was some time in the past such that a goldfish was among the objects swallowed by Gilly at that time”
Event Properties

- Gilly swallowed a goldfish
  - Previous: $\exists t \text{ past}(t) \land \exists \text{goldfish, swallow}(t, \text{Gilly})$

- Why stop at time? An event has other properties:
  - [Gilly] swallowed [a goldfish] [on a dare] [in a telephone booth] [with 30 other freshmen] [after many bottles of vodka had been consumed].
  - Specifies who what why when ...

- Replace time variable $t$ with an event variable $e$
  - $\exists e \text{ past}(e), \text{act}(e, \text{swallowing}), \text{swallower}(e, \text{Gilly}), \exists \text{goldfish, swallowee}(e), \exists \text{booth, location}(e), ...$
    - As with probability notation, a comma represents AND
    - Could define past as $\forall e \exists t \text{ before}(t, \text{now}), \text{ended-at}(e, t)$

“Davidsonian event variable” (after Donald Davidson, 1980)
Quantifier Order

- Gilly swallowed a goldfish in a booth
  - $\exists e \text{ past}(e), \text{ act}(e,\text{swallowing}), \text{ swallower}(e,\text{Gilly}), \exists(g\text{ goldfish}, \text{swallowee}(e)), \exists(\text{booth}, \text{location}(e))$, ...

- Gilly swallowed a goldfish in every booth
  - $\exists e \text{ past}(e), \text{ act}(e,\text{swallowing}), \text{ swallower}(e,\text{Gilly}), \exists(g\text{ goldfish}, \text{swallowee}(e)), \forall(\text{booth}, \text{location}(e))$, ...

- Does this mean what we’d expect??
  - says that there’s only one event with a single goldfish getting swallowed that took place in a lot of booths ...
Quantifier Order

- Groucho Marx celebrates quantifier order ambiguity:
  - In this country a woman gives birth every 15 min. Our job is to find that woman and stop her.
  - $\exists \text{woman} \left( \forall 15\text{min gives-birth-during}(15\text{min}, \text{woman}) \right)$
  - $\forall 15\text{min} \left( \exists \text{woman gives-birth-during}(15\text{min}, \text{woman}) \right)$
  - Surprisingly, both are possible in natural language!
  - Which is the joke meaning (where it’s always the same woman) and why?
Quantifier Order

- Gilly swallowed a goldfish in a booth
  - \( \exists e \) past(e), act(e,swallowing), swallower(e,Gilly), exists(goldfish, swallowee(e)), exists(booth, location(e)), ...
- Gilly swallowed a goldfish in every booth
  - \( \exists e \) past(e), act(e,swallowing), swallower(e,Gilly), exists(goldfish, swallowee(e)), all(booth, location(e)), ...
    \( \exists g \) goldfish(g), swallowee(e,g) \( \forall b \) booth(b) \( \Rightarrow \) location(e,b)

- Does this mean what we’d expect??
  - It’s \( \exists e \) \( \forall b \) which means same event for every booth
  - Probably false unless Gilly can be in every booth during her swallowing of a single goldfish
Quantifier Order

- Gilly swallowed a goldfish in a booth
  - $\exists e$ past(e), act(e, swallowing), swallower(e, Gilly), exists(goldfish, swallowee(e)), exists(booth, location(e)), ...

- Gilly swallowed a goldfish in every booth
  - $\exists e$ past(e), act(e, swallowing), swallower(e, Gilly), exists(goldfish, swallowee(e)), all(booth, $\lambda b$ location(e, b))

- Other reading ($\forall b \exists e$) involves quantifier raising:
  - all(booth, $\lambda b$ [$\exists e$ past(e), act(e, swallowing), swallower(e, Gilly), exists(goldfish, swallowee(e)), location(e, b)])
  - “for all booths b, there was such an event in b”
Intensional Arguments

- **Willy wants a unicorn**
  - $\exists e \, \text{act}(e, \text{wanting}), \text{wanter}(e, \text{Willy}), \exists \text{unicorn}, \lambda u \, \text{wantee}(e, u)$
    - "there is a particular unicorn $u$ that Willy wants"
    - In this reading, the wantee is an individual entity
  - $\exists e \, \text{act}(e, \text{wanting}), \text{wanter}(e, \text{Willy}), \text{wantee}(e, \lambda u \, \text{unicorn}(u))$
    - "Willy wants any entity $u$ that satisfies the unicorn predicate"
    - In this reading, the wantee is a type of entity
    - Sentence doesn’t claim that such an entity exists

- **Willy wants Lilly to get married**
  - $\exists e \, \text{present}(e), \text{act}(e, \text{wanting}), \text{wanter}(e, \text{Willy}), \text{wantee}(e, \lambda e' \, [\text{act}(e', \text{marriage}), \text{marrier}(e', \text{Lilly})])$
    - "Willy wants any event $e'$ in which Lilly gets married"
    - Here the wantee is a type of event
    - Sentence doesn’t claim that such an event exists

- **Intensional verbs besides want:** hope, doubt, believe, …
Intensional Arguments

- Willy wants a unicorn
  - $\exists e \ act(e, \text{wanting}), \ \text{wanter}(e, \text{Willy}), \ \text{wantee}(e, \ \lambda u \ \text{unicorn}(u))$
  - “Willy wants anything that satisfies the unicorn predicate”
  - here the wantee is a type of entity

- Problem:
  - $\lambda g \ \text{unicorn}(g)$ is defined by the actual set of unicorns (“extension”)
  - But this set is empty: $\lambda g \ \text{unicorn}(g) = \lambda g \ \text{FALSE} = \lambda g \ \text{pegasus}(g)$
  - Then $\text{wants a unicorn} = \text{wants a pegasus}$. Oops!
  - So really the wantee should be criteria for unicornness (“intension”)

- Traditional solution involves “possible-world semantics”
  - Can imagine other worlds where set of unicorns $\neq$ set of pegasi
Possible Worlds

- Traditional solution involves "possible-world semantics"
  - Wittgenstein (1921): "The world is all that is the case. The world is the totality of facts, not of things."
  - Can imagine other worlds where set of unicorns ≠ set of pegasi
  - Most facts can vary according to which world w you’re in:
    - loves(Barack, Michelle)
    - loves(w, Barack, Michelle)
    - most(\(\lambda x \text{pig}(x), \lambda x \text{big}(x)\))
    - most(w, \(\lambda x \text{pig}(w, x), \lambda x \text{big}(w, x)\))
    - most(\text{pig}, \text{big})
    - most(w, \text{pig}(w), \text{big}(w))
    - wants(Willy, unicorn)
    - wants(w, Willy, unicorn)
    - wants(Willy, \(\lambda u \text{unicorn}(u)\))
    - wants(w, Willy, \(\lambda w' \lambda u \text{unicorn}(w',u)\))

  "intension" of unicorn, not tied to current world w
  Function checks in any world w' whether something is a unicorn
  These criteria are the same in every world:
  unicorn \(\equiv \lambda w' \lambda u \text{(has_horn}(w',u), \text{horselike}(w',u), \text{magical}(w',u), \ldots)\)
Possible Worlds: More uses

- **Modals (woulda coulda shoulda)**
  - **Deontic \( \forall \) modal**
    - **You must pay the rent**
    - **In all possible worlds that are “like” this world, and in which you fulfill your obligations: you do pay the rent**
  - **Deontic \( \exists \) modal**
    - **You may pay the rent**
    - **In some possible world that is “like” this world, and in which you fulfill your obligations: you do pay the rent**
  - **Epistemic \( \forall \) modal**
    - **You must have paid the rent**
    - **In all possible worlds that are “like” this world, and which are consistent with my observations: you paid the rent**
  - **Bouletic \( \exists \) modal**
    - **You can pay the rent**
    - **In some possible world that is “like” this world, and in which you have no additional powers: you do pay the rent**

... and more ...

(varies by language, but always quantifies over some set of “accessible” worlds)
Possible Worlds: More uses

- **Modals (woulda coulda shoulda)**
  - You *must pay* the rent
  - In *all* possible worlds that are “like” this world, and in which you fulfill your obligations: you pay the rent

- **Counterfactuals**
  - If you hadn’t, you’d be homeless
  - In *all* possible worlds that are “like” this world, except that you didn’t pay the rent: you are now homeless

- What are the “worlds that are ‘like’ this world”? (“accessible” worlds)
  - You don’t pay rent, but otherwise change “as little as possible.” (Same apartment, same eviction laws, no miracles to save you from the gutter, …)
  - But rather slippery how to figure out what those “minimum changes” are!
  - Let’s watch instant replays on the Subjunc-TV (Hofstadter, 1979):
    - “Here’s what *would’ve* happened … if Palindromi hadn’t stepped out of bounds”  
    - “… if only it hadn’t been raining” “… if only they’d been playing against Chicago”  
    - “… if only they’d been playing baseball” “… if only 13 weren’t prime”
Possible Worlds: More uses

- Modals (woulda coulda shoulda)
  - You must pay the rent
  - In all possible worlds that are “like” this world, and in which you fulfill your obligations, you pay the rent

- Counterfactuals
  - If you hadn’t, you’d be homeless
  - In all possible worlds that are “like” this world, except that you didn’t pay the rent, you are now homeless
  - \( p(\text{homeless} \mid \text{didn’t pay rent}) > 0.5 \)

Traditional view is that some worlds are “accessible” and others aren’t. But reasoning about what would tend to happen if you didn’t pay the rent seems to require probabilistic reasoning.

So maybe you have something like a probability distribution over worlds?

Estimate distribution from observing the world’s facts and rules, but smoothed somehow? So my distribution will allocate a little probability to worlds where you didn’t pay the rent and became homeless, or didn’t pay the rent but moved in with your parents, etc. ... even though I’m sure none of these worlds actually happened.
Control

- Willy wants Lilly to get married
  - $\exists e \text{ present}(e), \text{ act}(e, \text{ wanting}), \text{ wanter}(e, \text{ Willy}), \text{ wantee}(e, \lambda f [\text{ act}(f, \text{ marriage}), \text{ marrier}(f, \text{ Lilly})])$

- Willy wants to get married
  - Same as Willy wants Willy to get married
  - Just as easy to represent as Willy wants Lilly ...
  - The only trick is to construct the representation from the syntax. The empty subject position of “to get married” is said to be controlled by the subject of “wants.”
Nouns and Their Modifiers

- Nouns and adjectives both restrict an entity’s properties:
  - expert: $\lambda g \text{ expert}(g)$
  - big fat expert: $\lambda g \text{ big}(g), \text{ fat}(g), \text{ expert}(g)$
  - Baltimore expert (i.e., expert from Baltimore):
    $\lambda g \text{ Related}(\text{Baltimore, g}), \text{ expert}(g)$

- But they sometimes first combine into compound concepts:
  - Adj+N: bogus expert (i.e., someone who has bogus_expertise):
    $\lambda g (\text{bogus}(\text{expert}))(g)$ [not $\lambda g \text{ bogus}(g), \text{ expert}(g)$ since they’re not an expert!]
  - N+N: Baltimore expert (i.e., expert on Baltimore – different stress):
    $\lambda g (\text{Modified-by}(\text{Baltimore, expert}))(g)$
  - (N+V)+ending: dog catcher:
    $\lambda g \exists e \text{ act}(e, \text{catching}), \text{catcher}(e, g), \exists \text{exists}(\text{dog,catchee}(e))$
  - garbage collection:
    $\lambda e (\text{act}(e, \text{collecting}), \exists \text{exists}(\text{garbage,collectee}(e)))$

- If we didn’t make a compound concept first, things would go awry
  law expert and dog catcher
  $= \lambda g \text{ Related}(\text{law,g}), \text{ expert}(g), \text{ Related}(\text{dog, g}), \text{ catcher}(g)$ **wrong**
  $= \text{dog expert and law catcher}$
Nouns and Their Modifiers

We can argue about the details of the compound representations, e.g., how much of the semantics is explicit in the lambda-term, how much is in the semantics of individual words like \textit{bogus}, and how much is shoved under the carpet into primitives like Modified-by, which are assumed to piece together a reasonable meaning using world knowledge and context.

- \( \lambda g \ (\text{bogus}(\text{expert}))(g) \) \dots \textit{bogus} can construct a new concept
  or \( \lambda g \ (\text{Modified-by}(\text{bogus,expert}))(g) \)  \\
- \( \lambda g \ (\text{Modified-by}(\text{Baltimore, expert}))(g) \)  \\
  or \( \lambda g \ (\text{Baltimore}(\text{expert}))(g) \)  \\
  or \( \lambda g \ (\text{expert}(\text{Baltimore}))(g) \)
Nouns and Their Modifiers

- the goldfish that Gilly swallowed
- every goldfish that Gilly swallowed
- three goldfish that Gilly swallowed

\[ \lambda g \[ \text{goldfish}(g), \text{swallowed}(\text{Gilly}, g) \] \]

like an adjective!

- three swallowed-by-Gilly goldfish

Or for real: \[ \lambda g \[ \text{goldfish}(g), \exists e \[ \text{past}(e), \text{act}(e,\text{swallowing}), \text{swallower}(e,\text{Gilly}), \text{swallowee}(e, g) \] \] \]
Adverbs

- Lili passionately wants Billy
  - Wrong?: \( \text{passionately(want}(Lili,Billy)) = \text{passionately(true)} \)
  - Better: \( (\text{passionately(want)})(Lili,Billy) \)
  - Best: \( \exists e \ \text{present}(e), \ \text{act}(e,\text{wanting}), \ \text{wanter}(e,Lili), \ \text{wantee}(e, Billy), \ \text{manner}(e, \text{passionate}) \)

- Lili often stalks Billy
  - \( (\text{often(stalk)})(Lili,Billy) \)
  - \( \text{many(day, } \lambda d \ \exists e \ \text{present}(e), \ \text{act}(e,\text{staking}), \ \text{stalker}(e,Lili), \ \text{stalkee}(e, Billy), \ \text{during}(e,d)) \)

- Lili obviously likes Billy
  - \( (\text{obviously(like)})(Lili,Billy) \) – one reading
  - \( \text{obvious(like}(Lili, Billy)) \) – another reading
Speech Acts

• What is the meaning of a full sentence?
  • Depends on the punctuation mark at the end. 😊
  • Billy likes Lili. → assert(like(B,L))
  • Billy likes Lili? → ask(like(B,L))
    ▪ or more formally, “Does Billy like Lili?”
  • Billy, like Lili! → command(like(B,L))
    ▪ or more accurately, “Let Billy like Lili!”

• Let’s try to do this a little more precisely, using event variables.

• (We’ll leave out the world variables.)
Speech Acts

- What did Gilly swallow?
  - \textbf{ask}(\lambda x \exists e \text{ past}(e), \text{ act}(e,\text{swallowing}), \text{ swallower}(e,\text{Gilly}), \text{ swallowee}(e,x))
  - Argument is identical to the modifier “that Gilly swallowed”
  - Is there any common syntax?

- Eat your fish!
  - \textbf{command}(\lambda f \text{ act}(f,\text{eating}), \text{ eater}(f,\text{Hearer}), \text{ eatee}(...))

- I ate my fish.
  - \textbf{assert}(\exists e \text{ past}(e), \text{ act}(e,\text{eating}), \text{ eater}(f,\text{Speaker}), \text{ eatee}(...))
Compositional Semantics

\[
\text{assert}(\forall \text{nation}, \exists e \text{ present(e)}, \text{act}(e, \text{wanting}), \text{wanter}(e, x), \text{wantee}(e, \lambda e' \text{ act}(e', \text{loving}), \text{lover}(e', B), \text{lovee}(e', M)))
\]
Barack Obama signed the Affordable Care act on Tuesday. He gave a speech later that afternoon on how the act would help the American people. Several prominent Republicans were quick to denounce the new law.

$\exists e \text{ sign}(e, \text{Barack Obama}) \land \text{patient}(e, \text{ACA}) \land \text{time}(e, \text{Tuesday})$

- Need to impute missing information, resolve coreference, etc.
- Still unclear how to represent some things precisely or how that information could be leveraged (several prominent Republicans)
(At least) Three Solutions

- Crafted annotations to capture some subset of phenomena: predicate-argument structures (semantic role labeling), time (temporal relations), ...

- Slot filling: specific ontology, populate information in a predefined way
  
  (Earthquake: magnitude=8.0, epicenter=central Italy, ...)

- Entity-relation-entity triples: focus on entities and their relations (note that prominent events can still be entities)
  
  (Lady Gaga, singerOf, Bad Romance)
Slot Filling
Slot Filling

- Most conservative, narrow form of IE

Indian Express — A massive earthquake of magnitude 7.3 struck Iraq on Sunday, 103 kms (64 miles) southeast of the city of As-Sulaymaniyah, the US Geological Survey said, reports Reuters. US Geological Survey initially said the quake was of a magnitude 7.2, before revising it to 7.3.

Speaker: Alan Clark

“Gender Roles in the Holy Roman Empire”

Old work: HMMs, later CRFs trained per role

Freitag and McCallum (2000)
Slot Filling: MUC

(a) Template

<table>
<thead>
<tr>
<th>SELLER</th>
<th>BUSINESS</th>
<th>ACQUIRED</th>
<th>PURCHASER</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSR Limited</td>
<td>Oil and Gas</td>
<td>Delhi Fund</td>
<td>Esso Inc.</td>
</tr>
</tbody>
</table>

(b) Document

[S CSR] has said that [S it] has sold [S its] [B oil interests] held in [A Delhi Fund]. [P Esso Inc.] did not disclose how much [P they] paid for [A Delhi].

- Key aspect: need to combine information across multiple mentions of an entity using coreference

Haghighi and Klein (2010)
Slot Filling: Forums

- Extract product occurrences in cybercrime forums, but not everything that looks like a product is a product

<table>
<thead>
<tr>
<th>TITLE: [ buy ] Backconnect bot</th>
</tr>
</thead>
<tbody>
<tr>
<td>BODY: Looking for a solid backconnect bot.</td>
</tr>
<tr>
<td>If you know of anyone who codes them please let me know</td>
</tr>
</tbody>
</table>

(a) File 0-initiator4856

<table>
<thead>
<tr>
<th>TITLE: Exploit cleaning?</th>
</tr>
</thead>
<tbody>
<tr>
<td>BODY: Have some Exploits, i need fud.</td>
</tr>
</tbody>
</table>

(b) File 0-initiator10815

Not a product in this context

Portnoff et al. (2017), Durrett et al. (2017)
Relation Extraction
Relation Extraction

- Extract entity-relation-entity triples from a fixed inventory

During the war in **Iraq**, **American journalists** were sometimes caught in the line of fire

- Use NER-like system to identify entity spans, classify relations between entity pairs with a classifier

- Systems can be feature-based or neural, look at surface words, syntactic features (dependency paths), semantic roles

- Problem: limited data for scaling to big ontologies

ACE (2003-2005)
Extracting relations from text

- Company report:
  “International Business Machines Corporation (IBM or the company) was incorporated in the State of New York on June 16, 1911, as the Computing-Tabulating-Recording Co. (C-T-R)...”

- Extracted Complex Relation
  **Company-Founding**
  | Company            | IBM               |
  | Location           | New York          |
  | Date               | June 16, 1911     |
  | Original-Name      | Computing-Tabulating-Recording Co. |

- But we will focus on the simpler task of extracting relation **triples**
  - Founding-year(IBM, 1911)
  - Founding-location(IBM, New York)
Automated Content Extraction (ACE)

17 relations from 2008 “Relation Extraction Task”

Slide from D. Jurafsky
Automated Content Extraction (ACE)

- Physical-Located \( \text{PER-GPE} \)
  He was in Tennessee
- Part-Whole-Subsidiary \( \text{ORG-ORG} \)
  XYZ, the parent company of ABC
- Person-Social-Family \( \text{PER-PER} \)
  John’s wife Yoko
- Org-AFF-Founder \( \text{PER-ORG} \)
  Steve Jobs, co-founder of Apple...
UMLS: Unified Medical Language System

- 134 entity types, 54 relations

<table>
<thead>
<tr>
<th>Injury</th>
<th>disrupts</th>
<th>Physiological Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bodily Location</td>
<td>location-of</td>
<td>Biologic Function</td>
</tr>
<tr>
<td>Anatomical Structure</td>
<td>part-of</td>
<td>Organism</td>
</tr>
<tr>
<td>Pharmacologic Substance</td>
<td>causes</td>
<td>Pathological Function</td>
</tr>
<tr>
<td>Pharmacologic Substance</td>
<td>treats</td>
<td>Pathologic Function</td>
</tr>
</tbody>
</table>
Doppler echocardiography can be used to diagnose left anterior descending artery stenosis in patients with type 2 diabetes

Echocardiography, Doppler
DIAGNOSES
Acquired stenosis
Ontological relations

Examples from the WordNet Thesaurus

- **IS-A (hypernym):** subsumption between classes
  - Giraffe IS-A ruminant IS-A ungulate IS-A mammal IS-A vertebrate IS-A animal...

- **Instance-of:** relation between individual and class
  - San Francisco instance-of city
Patterns for Relation Extraction

- Hand-written rules for relation extraction were used in MUC (such as the Fastus system)
- Recently there has been a renewed wide interest in learning rules for relation extraction focused on precision
  - The presumption is that interesting information occurs many times on the web, with different contexts
    - e.g., how many times does "Barack Obama is the 44th President of the United States" occur on the web?
  - Focusing on high precision is reasonable because the high redundancy will allow us to deal with recall
How to build relation extractors

1. Hand-written patterns
2. Supervised machine learning
3. Semi-supervised and unsupervised
   - Bootstrapping (using seeds)
   - Distant supervision
   - Unsupervised learning from the web
Rules for extracting IS-A relation

Early intuition from Hearst (1992)

- “Agar is a substance prepared from a mixture of red algae, such as Gelidium, for laboratory or industrial use”

- What does Gelidium mean?
- How do you know?  

Slide from D. Jurafsky
Rules for extracting IS-A relation

Early intuition from Hearst (1992)

- “Agar is a substance prepared from a mixture of red algae, such as Gelidium, for laboratory or industrial use”

- What does Gelidium mean?
- How do you know?
Hearst’s Patterns for extracting IS-A relations

(Hearst, 1992): Automatic Acquisition of Hyponyms

“Y such as X ((, X)* (, and|or) X)”
“such Y as X”
“X or other Y”
“X and other Y”
“Y including X”
“Y, especially X”
## Hearst’s Patterns for extracting IS-A relations

<table>
<thead>
<tr>
<th>Hearst pattern</th>
<th>Example occurrences</th>
</tr>
</thead>
<tbody>
<tr>
<td>X and other Y</td>
<td>...temples, treasuries, and other important civic buildings.</td>
</tr>
<tr>
<td>X or other Y</td>
<td>Bruises, wounds, broken bones or other injuries...</td>
</tr>
<tr>
<td>Y such as X</td>
<td>The bow lute, such as the Bambara ndang...</td>
</tr>
<tr>
<td>Such Y as X</td>
<td>...such authors as Herrick, Goldsmith, and Shakespeare.</td>
</tr>
<tr>
<td>Y including X</td>
<td>...common-law countries, including Canada and England...</td>
</tr>
<tr>
<td>Y, especially X</td>
<td>European countries, especially France, England, and Spain...</td>
</tr>
</tbody>
</table>
Intuition: relations often hold between specific entities

- **located-in** (ORGANIZATION, LOCATION)
- **founded** (PERSON, ORGANIZATION)
- **cures** (DRUG, DISEASE)

Start with Named Entity tags to help extract relation!
Which relations hold between 2 entities?

Drug

Cure?

Prevent?

Cause?

Disease

Slide from D. Jurafsky
What relations hold between 2 entities?

PERSON

- Founder?
- Investor?
- Member?
- Employee?
- President?

ORGANIZATION

Slide from D. Jurafsky
Extracting Richer Relations Using Rules and Named Entities

Who holds what office in what organization?

PERSON, POSITION of ORG
- George Marshall, Secretary of State of the United States

PERSON (named | appointed | chose | etc.) PERSON Prep? POSITION
- Truman appointed Marshall Secretary of State

PERSON [be]? (named | appointed | etc.) Prep? ORG POSITION
- George Marshall was named US Secretary of State
Beyond sequences of words

- Patterns over parse trees can work much better (why?)
- Is there anything better than parse trees?
- What other elements are we still missing?
Hand-built patterns for relations

- **Plus:**
  - Human patterns tend to be high-precision
  - Can be tailored to specific domains

- **Minus**
  - Human patterns are often low-recall
  - A lot of work to think of all possible patterns!
  - Don’t want to have to do this for every relation!

- **The big Question** (for me/Yoav)
  - Can we improve the human rule-writing process?
    - How?
Supervised Methods

- For named entity tagging, parsing, coreference, statistical/neural systems are the state of the art
- However, for relation extraction, this is not necessarily true
  - Still many hand-crafted rule-based systems out there that work well
  - But hand-crafting such systems takes a lot of work, so classification approaches are interesting (and they are improving with time)
Supervised machine learning for relations

- Choose a set of relations we’d like to extract
- Choose a set of relevant named entities
- Find and label data
  - Choose a representative corpus
  - Label the named entities in the corpus
  - Hand-label the relations between these entities
  - Break into training, development, and test
- Train a classifier on the training set
How to do classification in supervised relation extraction

1. Find all pairs of named entities (usually in same sentence)

2. Decide if 2 entities are related

3. If yes, classify the relation

   - Why the extra step?
     - Faster classification training by eliminating most pairs
     - Can use distinct feature-sets appropriate for each task.
Automated Content Extraction (ACE)

17 sub-relations of 6 relations from 2008 “Relation Extraction Task”

Slide from D. Jurafsky
American Airlines, a unit of AMR, immediately matched the move, spokesman Tim Wagner said.
Classifiers for supervised methods

- Now you can use any classifier you like
  - Decision Tree
  - MaxEnt
  - Naïve Bayes
  - SVM
  - ...
- Train it on the training set, tune on the dev set, test on the test set

Slide modified from D. Jurafsky
Distant Supervision

- Lots of relations in our knowledge base already (e.g., 23,000 film-director relations); use these to bootstrap more training data

- If two entities in a relation appear in the same sentence, assume the sentence expresses the relation

Director

[Steven Spielberg]’s film [Saving Private Ryan] is loosely based on the brothers’ story

Allison co-produced the Academy Award-winning [Saving Private Ryan], directed by [Steven Spielberg]

Mintz et al. (2009)
Open IE
Open Information Extraction

- "Open"ness — want to be able to extract all kinds of information from open-domain text

- Acquire commonsense knowledge just from "reading" about it, but need to process lots of text ("machine reading")

- Typically no fixed relation inventory
TextRunner

- Extract positive examples of (e, r, e) triples via parsing and heuristics
- Train a Naive Bayes classifier to filter triples from raw text: uses features on POS tags, lexical features, stopwords, etc.

Barack Obama, 44th president of the United States, was born on August 4, 1961 in Honolulu

=> Barack Obama, was born in, Honolulu

- 80x faster than running a parser (which was slow in 2007...)
- Use multiple instances of extractions to assign probability to a relation

Banko et al. (2007)
ReVerb

- More constraints: open relations have to begin with verb, end with preposition, be contiguous (e.g., *was born on*)

- Extract more meaningful relations, particularly with light verbs

| is | is an album by, is the author of, is a city in |
| has | has a population of, has a Ph.D. in, has a cameo in |
| made | made a deal with, made a promise to |
| took | took place in, took control over, took advantage of |
| gave | gave birth to, gave a talk at, gave new meaning to |
| got | got tickets to, got a deal on, got funding from |

Fader et al. (2011)
What have we seen today?

- We’ve discussed what semantic representations should look like.
  - Logical forms
  - Lambda functions
  - Neo-davidsonian Semantics
- Compositional semantics
  - How to build sentence representations hierarchically?
  - CCG
- Complications
  - Interesting language phenomena
  - + Possible worlds
- “Real” text
  - (drastically) simplify to “Information Extraction”