Natural Language Processing

Info 159/259
Lecture 18: Semantics (Oct 25, 2018)

David Bamman, UC Berkeley
Graph-based parsing

- For a given sentence $S$, we want to find the highest-scoring tree among all possible trees for that sentence $\mathcal{G}_S$

$$\hat{T}(S) = \arg \max_{t \in \mathcal{G}_S} \text{score}(t, S)$$

- Edge-factored scoring: the total score of a tree is the sum of the scores for all of its edges (arcs):

$$\text{score}(t, S) = \sum_{e \in t} \text{score}(e)$$
Edge-factored features

- Word form of head/dependent
- POS tag of head/dependent
- Distributed representation of h/d
- Distance between h/d
- POS tags between h/d
- Head to left of dependent?

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>headₜ = man</td>
<td>1</td>
</tr>
<tr>
<td>head_pos = NN</td>
<td>1</td>
</tr>
<tr>
<td>distance</td>
<td>4</td>
</tr>
<tr>
<td>child_pos = JJ and head_pos = NN</td>
<td>1</td>
</tr>
<tr>
<td>child_pos = NN and head_pos = JJ</td>
<td>0</td>
</tr>
</tbody>
</table>
\[ \text{score}(e) = \sum_{i=1}^{F} x_i \beta_i \]
\[
\text{score}(e) = \sum_{i=1}^{F} x_i \beta_i
\]

| \text{head}_t = \text{man} | 1 | 3.7 |
| \text{head}_t = \text{man} | 1 | 1.3 |
| distance | 4 | 0.7 |
| child_{pos} = \text{JJ and} | 1 | 0.3 |
| child_{pos} = \text{NN and} | 0 | -2.7 |

score(e) = 8.1
I saw a man today who is tall.
MST Parsing

• We start out with a fully connected graph with a score for each edge

• $N^2$ edges total

(Assume one edge connects each node as dependent and node as head, $N^2$ total)
MST Parsing

- From this graph $G$, we want to find a spanning tree (tree that spans $G$ [includes all the vertices in $G$])

- If the edges have weights, the best parse is the maximal spanning tree (the spanning tree with the highest total weight).
MST Parsing

- To find the MST of any graph, we can use the Chu-Liu-Edmonds algorithm in $O(n^3)$ time.

- More efficient, Gabow et al. find the MST in $O(n^2 + n \log n)$
Learning

\[ \hat{T}(S) = \arg \max_{t \in G_S} \text{score}(t, S) \]

\[ \hat{T}(S) = \arg \max_{t \in G_S} \sum_{e \in E} \phi(e, x)^\top \beta \]

\[ \hat{T}(S) = \arg \max_{t \in G_S} \left[ \sum_{e \in E} \phi(e, x) \right]^\top \beta \]

\( \phi \) is our feature vector scoped over the source dependent, target head and entire sentence \( x \)
Learning

\[ \hat{T}(S) = \arg \max_{t \in G_S} \text{score}(t, S) \]

• Given this formulation, we want to learn weights for \( \beta \) that make the score for the gold tree higher than for all other possible incorrect trees.

• That’s expensive, so let’s just try to make the score for the gold tree higher than the highest scoring tree we predict (if it’s wrong)
Learning

\[
\sum_{e \in E} \phi(e, x) \quad \beta = \Phi_{\text{gold}}(E, x)^\top \beta
\]

score for gold tree in treebank

\[
\sum_{e \in \hat{E}} \phi(e, x) \quad \beta = \hat{\Phi}_{\text{gold}}(\hat{E}, x)^\top \beta
\]

score for argmax tree in our model
Learning

- We can optimize this using SGD by taking the derivative with respect to the difference in scores (which we want to maximize):

\[
\Phi_{gold}(E, x) \top \beta - \Phi_{pred}(\hat{E}, x) \top \beta
\]

\[
= \left[ \Phi_{gold}(E, x) - \Phi_{pred}(\hat{E}, x) \right] \top \beta
\]

\[
\frac{\partial}{\partial \beta} \left[ \Phi_{gold}(E, x) - \Phi_{pred}(\hat{E}, x) \right] \top \beta = \Phi_{gold}(E, x) - \Phi_{pred}(\hat{E}, x)
\]
Perceptron

**Data:** training data $x \in \mathbb{R}^F$, $y \in \{-1, +1\}$, $i = 1 \ldots N$; initialize $\beta_0 = 0^F$;

$k = 0$;

**while** not converged **do**

$k = k + 1$;

$i = k \mod N$;

**if** $\hat{y}_i \neq y_i$ **then**

$\beta_{t+1} = \beta_t + y_i x_i$

**else**

do nothing;

**end**

**end**

Perceptron update for binary classification = adding the feature values to the current estimate of $\beta$
Algorithm 1 Structured perceptron

1: function PERCEPTRONUPDATE(x, E, β)
2: \( \Phi_{gold}(E, x) \leftarrow \text{createFeatures}(x, E) \)
3: \( \hat{\Phi}_{pred}(\hat{E}, x) \leftarrow \text{createFeatures}(x, \hat{E}) \)
4: \( \hat{E} \leftarrow \text{CLU}(x, \beta) \)
5: \( \beta \leftarrow \beta + \Phi_{gold}(E, x) - \hat{\Phi}_{pred}(\hat{E}, x) \)
6: end function

Create feature vector from true tree
Use CLU to find best tree given scores from current \( \beta \)
Update \( \beta \) with the different between the feature vectors
Natural Language Processing

Info 159/259
Lecture 18: Semantics (Oct 25, 2018)

David Bamman, UC Berkeley
Why is syntax important?

- Foundation for semantic analysis (on many levels of representation: semantic roles, compositional semantics, frame semantics)

I bought a car from the salesperson
Why is syntax insufficient?

- Syntax encodes the structure of language but doesn’t directly address meaning.

- Syntax alone doesn’t ground “grab” in an action to take in the world.
Lexical semantics

- Vector representation that encodes information about the *distribution* of contexts a word appears in.

- Words that appear in similar contexts have similar representations (and similar meanings, by the *distributional hypothesis*).

- We can represent what individual words “mean” as a function of what other words they’re related to (but that’s still not grounding).
Lexical semantics

<table>
<thead>
<tr>
<th>Word</th>
<th>Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>grab</td>
<td>1</td>
</tr>
<tr>
<td>throw</td>
<td>0.824</td>
</tr>
<tr>
<td>pull</td>
<td>0.818</td>
</tr>
<tr>
<td>knock</td>
<td>0.799</td>
</tr>
<tr>
<td>grabbing</td>
<td>0.789</td>
</tr>
<tr>
<td>steal</td>
<td>0.787</td>
</tr>
<tr>
<td>pulling</td>
<td>0.764</td>
</tr>
<tr>
<td>grabs</td>
<td>0.756</td>
</tr>
<tr>
<td>away</td>
<td>0.746</td>
</tr>
<tr>
<td>catch</td>
<td>0.74</td>
</tr>
</tbody>
</table>
• “Grab” = execute GrabbingFunction()

• “the cup” = object ID 9AF1948A81CD22
Why is syntax insufficient?

- Even if we have a reference model for each word in a sentence, syntax doesn’t tell us how those referents changes as a function of their compositionally.

- What is the meaning of a VP?
Semantics

Lexical semantics is concerned with representing the meaning of words (and their relations)

Logical semantics is concerned with representing the meaning of sentences.
Meaning representation

• A representation of the meaning of a sentence needs to bridge linguistic aspects of the sentence with non-linguistic knowledge about the world.
Following directions

<table>
<thead>
<tr>
<th>Linguistic</th>
<th>Non-linguistic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verbs like <em>grab, release</em></td>
<td>Actions a robot can execute</td>
</tr>
<tr>
<td>Nouns like <em>cup, block</em></td>
<td>Specific entities in the world</td>
</tr>
<tr>
<td>transitive VP (V NP)</td>
<td>An action executed upon an object</td>
</tr>
</tbody>
</table>
Barack Obama was the 44th President of the United States. Obama was born on August 4 in Honolulu, Hawaii. In late August 1961, Obama's mother moved with him to the University of Washington in Seattle for a year...

Was Barack Obama born in the United States?

Non-linguistic

Hawaii is a place

Hawaii is a place within the United States

“born in X” entails “from x”
<table>
<thead>
<tr>
<th>Domain</th>
<th>Question</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geo</td>
<td>which state has the most rivers running through it?</td>
<td>(argmax $0 (state:t $0) (count $1 (and (river:t $1) (loc:t $1 $0))))</td>
</tr>
<tr>
<td>ATIS</td>
<td>all flights from dallas before 10am</td>
<td>(lambda $0 e (and (flight $0) (from $0 dallas:ci) (&lt; (departure time $0) 1000:ti)))</td>
</tr>
<tr>
<td>SQL</td>
<td>What record company did conductor Mikhail Snitko record for after 1996?</td>
<td>SELECT Record Company WHERE (Year of Recording &gt; 1996) AND (Conductor = Mikhail Snitko)</td>
</tr>
<tr>
<td>Django</td>
<td>if length of bits is lesser than integer 3 or second element of bits is not equal to string 'as'</td>
<td>if len(bits) &lt; 3 or bits[1] != 'as':</td>
</tr>
</tbody>
</table>

Meaning representation

A meaning representation should be *unambiguous*; each statement in a meaning representation should have one meaning.
Meaning representation

Syntax resolves some ambiguity
“Once every hour, someone is involved in an Internet scam”

“That person is Michael Scott”
Same structure for “someone” meaning:

- Each person for each scam
- One person in the same scam (Michael Scott)
First-order logic (FOL)

- We want to represent every sentence as an unambiguous proposition in FOL

<table>
<thead>
<tr>
<th>Sentence</th>
<th>Luke was fighting with Darth Vader</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOL</td>
<td>FIGHT(LUKE, VADER)</td>
</tr>
</tbody>
</table>
**FIGHT(LUKE, VADER)**

- This is a *relation*; we define what it means.
- These are *constants*; we know who they uniquely identify.
Star Wars Episode IV: A New Hope (Q17738)

**director**
- George Lucas
  - 11 references

**screenwriter**
- George Lucas
  - 1 reference

**cast member**
- Harrison Ford
  - character role
  - Han Solo
  - 12 references

- Alec Guinness
  - character role
  - Obi-Wan Kenobi
  - 8 references

- Carrie Fisher
  - character role
  - Princess Leia
  - edit

+ add value
How we map a natural language sentence to FOL is the task of semantic parsing; but we define the FOL relations and entities to be sensitive to what matters in our model.

<table>
<thead>
<tr>
<th>Sentence</th>
<th>FOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Luke fought with Vader</td>
<td></td>
</tr>
<tr>
<td>Skywalker dueled with Darth Vader</td>
<td></td>
</tr>
<tr>
<td>Luke was fighting with Darth Vader</td>
<td></td>
</tr>
</tbody>
</table>

First-order logic (FOL)
First-order logic (FOL)

- How we map a natural language sentence to FOL is the task of semantic parsing; but we define the FOL relations and entities to be sensitive to what matters in our world model.

<table>
<thead>
<tr>
<th>Sentence</th>
<th>FOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Luke battled Vader</td>
<td><code>FIGHT(LUKE, VADER)</code></td>
</tr>
<tr>
<td>Luke fought with Vader</td>
<td><code>FIGHT(LUKE, VADER)</code></td>
</tr>
<tr>
<td>Luke was fighting with Darth Vader</td>
<td><code>FIGHT(LUKE, VADER)</code></td>
</tr>
<tr>
<td>Skywalker dueled with Darth Vader</td>
<td><code>DUEL(LUKE, VADER)</code></td>
</tr>
</tbody>
</table>

Maybe in our star wars model we want to preserve the difference between fighting and dueling.
First-order logic (FOL)

- How we map a natural language sentence to FOL is the task of semantic parsing; but we define the FOL relations and entities to be sensitive to what matters in our world model.

For a robot model, there is only one possible “grabbing” kind of action, so subtle differences don’t matter.

<table>
<thead>
<tr>
<th>Sentence</th>
<th>Grab the cup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sentence</td>
<td>Snatch the cup!</td>
</tr>
<tr>
<td>Sentence</td>
<td>Take the cup</td>
</tr>
<tr>
<td>FOL</td>
<td>\texttt{GRAB(Robot, Cup)}</td>
</tr>
</tbody>
</table>
First-order logic (FOL)

- Constants
- Relations (including properties)
- Variables
- Quantifiers
- Functions
- Logical connectives
Constants

- Constants refer to specific entities in the world being described.
- Refer to exactly one object in that world.
- Dependent on the model; people, things, places, events, etc.
Constants

- DarthVader
- DarthMaul
- Emperor
- Luke
- HanSolo
- Leia
- Obi-Wan
- Chewbacca
- R2-D2
- C-3PO
- Yoda
- Luke_Lightsaber_1
- Luke_Lightsaber_2
- Vader_Lightsaber_1
- Tatooine
- Hoth
- Endor

the domain simply enumerates the objects; we know nothing else about them yet
Relations

• N-ary relations hold among FOL terms (constants, variables, functions).

<table>
<thead>
<tr>
<th>Unary (property)</th>
<th>Human(Luke), Robot(C-3PO)</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>binary relation</em></td>
<td>Fights(Luke, Vader)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Properties (unary relations)

- Informally, properties describe attributes of entities in the model (e.g. Luke being human).
- Formally, properties denote \textit{sets of elements} in the domain.
Properties

- \([[[\text{HUMAN}] [Darth Vader, Emperor, Luke, Han Solo, Leia, Obi-Wan}]
- \([[[\text{ROBOT}] [R2-D2, C-3PO}]

- \([[[\text{LIGHTSABER}] [\text{Luke_Lightsaber}_1, \text{Luke_Lightsaber}_2, \text{Vader_Lightsaber}_1}]

- \([[[\text{RED}] [\text{Vader_Lightsaber}_1}]
- \([[[\text{BLUE}] [\text{Luke_Lightsaber}_1}]
- \([[[\text{GREEN}] [\text{Luke_Lightsaber}_2}]
- \([[[\text{PLANET}] [\text{Tattooine, Hoth, Endor}]

Relations

• Relations denote sets of tuples in the domain.
Relations

- \([[\text{FIGHT}]] = \{<\text{Darth Vader, Luke}>, <\text{Darth Vader, Obi-Wan}>, <\text{Luke, Emperor}>\}
- \([[\text{LIVES}]] = \{<\text{Luke, Tatooine}>\}
- \([[\text{FATHER}]] = \{<\text{Darth Vader, Luke}>, <\text{Darth Vader, Leia}>\}
- \([[\text{GIVES}]] = \{<\text{Obi-Wan, Luke, Luke_Lightsaber_1}>, <\text{Obi-Wan, Leia, Leia_Lightsaber_1}>\}
Extensional definition

- The denotation of **RED** is the set of objects that are lightsabers.
- The denotation of **FIGHT** is the set of pairs of entities that fight.
Logical connectives

- Formal means for composing expressions in a meaning representation language (symbols, quantifiers)

- Boolean operators on connectives yield known truth values

<table>
<thead>
<tr>
<th>Negation</th>
<th>$\neg \phi$</th>
<th>True if $\phi$ is false</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conjunction</td>
<td>$\phi \land \psi$</td>
<td>True if $\phi$ and $\psi$ are both true</td>
</tr>
<tr>
<td>Disjunction</td>
<td>$\phi \lor \psi$</td>
<td>True if $\phi$ or $\psi$ is true</td>
</tr>
<tr>
<td>Implication</td>
<td>$\phi \Rightarrow \psi$</td>
<td>True unless $\phi$ is true and $\psi$ is false</td>
</tr>
<tr>
<td>Equivalence</td>
<td>$\phi \Leftrightarrow \psi$</td>
<td>True if $\phi$ and $\psi$ are both true or both false</td>
</tr>
</tbody>
</table>
\[
M_{\text{star\_wars}}
\]

- Darth Vader
- Emperor
- Luke
- Han Solo
- Leia
- Obi-Wan
- Chewbacca
- R2-D2
- C-3PO

- Luke_Lightsaber_1
- Luke_Lightsaber_2
- Vader_Lightsaber

- Tatooine
- Hoth
- Endor

- \([\text{[HUMAN]}]\) = \{\text{Darth Vader, Emperor, Luke, Han Solo, Leia, Obi-Wan}\}
- \([\text{[ROBOT]}]\) = \{\text{R2-D2, C-3PO}\}
- \([\text{[LIGHTSABER]}]\) = \{\text{Luke\_Lightsaber\_1, Luke\_Lightsaber\_2, Vader\_Lightsaber\_1}\}
- \([\text{[RED]}]\) = \{\text{Vader\_Lightsaber\_1}\}
- \([\text{[BLUE]}]\) = \{\text{Luke\_Lightsaber\_1}\}
- \([\text{[GREEN]}]\) = \{\text{Luke\_Lightsaber\_2}\}
- \([\text{[PLANET]}]\) = \{\text{Tatooine, Hoth, Endor}\}

- \([\text{[FIGHT]}]\) = \{<\text{Darth Vader, Luke}>, <\text{Darth Vader, Obi-Wan}>, <\text{Luke, Emperor}>\}
- \([\text{[LIVES]}]\) = \{<\text{Luke, Tatooine}>\}
- \([\text{[FATHER]}]\) = \{<\text{Darth Vader, Luke}>, <\text{Darth Vader, Leia}>\}
Meaning

- Model-theoretic semantics; the truth of a proposition is determined with respect to some model $\mathcal{M}$ of the world.

- Separate from verification (does not need to be true of the real world); what would the world need to be like for the sentence to be true?
We’ll use [[*]] to describe the denotation of a term in a specific model $\mathcal{M}$. 

\[[[\text{LUKE}]]_{\mathcal{M}}\]

\[[[\text{FIGHTS}]]_{\mathcal{M}}\]
Truth

• The truth of a proposition under a model depends on whether the denotation of the constants is in the denotation of the relation.

$\text{Luke lives on Tatoine} \ (\text{[[Luke]}_M, \text{[[Tattoine]}}_M) \in \text{[[LIVES]]}_M$

$\text{FALSE}$

$\text{Luke fights Han} \ (\text{[[Luke]}_M, \text{[[Han]}}_M) \in \text{[[FIGHTS]]}_M$

$\text{TRUE}$
Luke lives on Tatooine and fights Darth Vader

$$((\text{[Luke]}), \text{[Tatooine]})) \in \text{[LIVES]}$$

$$\land$$

$$((\text{[Luke]}), \text{[Darth Vader]})) \in \text{[FIGHTS]}$$

Luke lives on Tatooine and fights Han

$$((\text{[Luke]}), \text{[Tatooine]})) \in \text{[LIVES]}$$

$$\land$$

$$((\text{[Luke]}), \text{[Han]})) \in \text{[FIGHTS]}$$
First-order logic

• This machinery lets us evaluate the truth conditions about specific entities and relations in $\mathcal{M}$. 
First-order logic

Does C-3PO fight anyone?

- Does C-3PO fight Luke?
- Does C-3PO fight Han?
- Does C-3PO fight Darth Vader?
- Does C-3PO fight Leia?
- Does C-3PO fight R2D2?
- Does C-3PO fight the Emperor?
First-order logic

C-3PO fights someone

\[ \exists x \text{ fights}(\text{C-3PO}, x) \]

- Variable: \( x \) is a free variable with no committed assignment
- Quantifier: \( \exists x \) denotes that \( x \) is now bound and has a truth value in \( M \)
First-order logic

\[ \exists x. \text{fights}(C-3PO, x) \]

\[ \forall x. \text{fights}(C-3PO, x) \]
Semantic parsing

- How do we get from a natural language statement to a proposition whose truth content we can evaluate in a model?

“Luke fights Han”

\(((\text{[[Luke]]}_M, \text{[[Han]]}_M) \in \text{[[FIGHTS]]}_M)\)
Compositionality

• Principle of compositionally: the meaning of a complex expression is function of the meaning of its constituent parts (Frege).

• Constituent parts here = syntactic constituents

S.meaning = function(NP.meaning, VP.meaning)
This is the meaning we want in the end

The meaning of the entities is their denotation in $\mathcal{M}$
How do we represent the meaning of this partial structure?
Lambda calculus

- Lambda expressions: descriptions of anonymous functions.

\[ \lambda x. P(x) \]

- sequence of variables
- FOL expression
Lambda calculus

- Lambda expressions: descriptions of anonymous functions.
  
  \[ \lambda x. P(x) \]

- A lambda expression is a function that can be applied to FOL terms as arguments to yield new FOL expressions where the variables are bound to the argument terms.
Lambda calculus

• Lambda expressions: descriptions of anonymous functions.

\[ \lambda x. P(x)(A) \]

Yields:

\[ P(A) \]
Lambda calculus

- Lambda expression can take multiple arguments (where subsequent reduction yields new lambda expressions)

\[ \lambda x.\lambda y.P(x, y) \]
Lambda calculus

\( \lambda y.\lambda x.P(x, y)(A) \)

\[ \text{Yields:} \]
\[ \lambda x.P(x, A) \]

\( \lambda x.P(x, A)(B) \)

\[ \text{Yields:} \]
\[ P(B, A) \]
Lambda calculus

The state of one thing being near something else

\( \lambda y.\lambda x.\text{NEAR}(x, y) \)

The state of one thing being near Cafe Strada

\( \lambda y.\lambda x.\text{NEAR}(x, y)(\text{CAFESTRADA}) \)

\( \lambda x.\text{NEAR}(x, \text{CAFESTRADA}) \)

\( \lambda x.\text{NEAR}(x, \text{CAFESTRADA})(\text{Boalt}) \)

Fully specified FOL formula: Boalt is near Cafe Strada

\( \text{NEAR}(\text{Boalt}, \text{CAFESTRADA}) \)
Lambda calculus

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

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

Order matters!
Currying

- Converting a predicate with multiple arguments into a sequence of single-argument predicates

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

\[ \lambda z.\lambda y.\lambda x.\text{GIVE}(x, y, z)(A) \]

\[ \lambda y.\lambda x.\text{GIVE}(x, y, A)(B) \]

\[ \lambda x.\text{GIVE}(x, B, A)(C) \]

\[ \text{GIVE}(C, B, A) \]
S:fights(Luke, Han)

NP:Luke

VP:???

V:???

NP:Han

Luke

fights

Han
Lambda calculus

• With lambda calculus, pieces of logical forms correspond to pieces of linguistic structure.

• We can solve our problem by pairing each terminal with a entity (Han) or a lambda expression

• Fights is a verb that expects subject argument (the fighter) and an object argument (the thing fought).

\[ \text{fights} \quad \lambda y.\lambda x.\text{FIGHTS}(x, y) \]
Here we can let naturally the NP to the right be the argument to this lambda expression to yield a new lambda expression

\[
\lambda y.\lambda x.\text{FIGHTS}(x, y)(\text{HAN}) \rightarrow \lambda x.\text{FIGHTS}(x, \text{HAN})
\]
Here we can state the NP to the left should be the argument to this lambda expression to yield a new lambda expression:

$$\lambda x.\text{FIGHTS}(x,HAN)(LUKE) \rightarrow \text{FIGHTS}(LUKE,HAN)$$
S: FIGHTS(Luke, Han)

NP: Luke
  | Luke

VP: \( \lambda x. \text{FIGHTS}(x, \text{Han}) \)

V: \( \lambda y. \lambda x. \text{FIGHTS}(x, y) \)
  | fights

NP: Han
  | Han
<table>
<thead>
<tr>
<th><strong>S</strong> → <strong>NP VP</strong></th>
<th><strong><a href="mailto:VP.sem@NP.sem">VP.sem@NP.sem</a></strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>VP → V NP</strong></td>
<td><strong><a href="mailto:V.sem@NP.sem">V.sem@NP.sem</a></strong></td>
</tr>
<tr>
<td><strong>V → fights</strong></td>
<td>$\lambda y.\lambda x.\text{fights}(x,y)$</td>
</tr>
<tr>
<td><strong>V → battles</strong></td>
<td>$\lambda y.\lambda x.\text{fights}(x,y)$</td>
</tr>
<tr>
<td><strong>NP → Han</strong></td>
<td>Han</td>
</tr>
<tr>
<td><strong>NP → Luke</strong></td>
<td>Luke</td>
</tr>
</tbody>
</table>

In general, we can let the non-terminals determine how the semantics of their constituents are combined.
Syntax

• We could represent the relationship between syntax and semantics in a CFG.

• But what we want is fine-grained control over the mapping between words and semantic primitives.
Tuesday

- Semantic parsing