From utterances to logical forms

Bill MacCartney
CS224U
30 April 2014

[with slides adapted from Luke Zettlemoyer]
Zettlemoyer & Collins 2005

• Won Best Paper award at UAI-2005
• Initiated “modern era” of semantic parsing
• First of many semantic parsing papers from Zettlemoyer & collaborators
• Worthy of very close reading: if you really understand this paper, you really understand semantic parsing!
The problem

Learning to map sentences to logical form:

Texas borders Kansas

\[ \text{borders(texas, kansas)} \]
Potential applications

• Natural language interfaces to databases
• Question answering
• Dialogue systems
Some training examples

Input:  What states border Texas?
Output: \( \lambda x. \text{state}(x) \land \text{borders}(x, \text{texas}) \)

Input:  What is the largest state?
Output:  \( \text{argmax}(\lambda x. \text{state}(x), \lambda x. \text{size}(x)) \)

Input:  What states border the largest state?
Output:  \( \lambda x. \text{state}(x) \land \text{borders}(x, \quad \text{argmax}(\lambda y. \text{state}(y), \lambda y. \text{size}(y))) \)
The approach

Learn lexical information (syntax & semantics) for words:

\[
\begin{align*}
\text{Texas} & \quad := \quad \text{syntax} = \text{NP} \quad : \quad \text{semantics} = \text{texas} \\
\text{states} & \quad := \quad \text{syntax} = \text{N} \quad : \quad \text{semantics} = \lambda x. \text{state}(x)
\end{align*}
\]

Learn to parse to logical form:

Input: \quad \textit{What states border Texas?}

Output: \quad \lambda x.\text{state}(x) \land \text{borders}(x, \text{texas})
Background

• Combinatory categorial grammar (CCG)
  ◦ Lexicon
  ◦ Parsing rules (combinators)

• Probabilistic CCG (PCCG)
Combinatory categorial grammar (CCG)

• A grammar formalism introduced by Steedman (1996)

• A rich set of syntactic categories
  ◦ In fact, an *infinite* set — vs. a few dozen in the Penn Treebank
  ◦ Atomic categories + functional categories which combine them

• A small number of ways of combining categories
  ◦ Chiefly function application, plus composition, type-raising, …
  ◦ Vs. hundreds of rules in the Penn Treebank

• Facilitates tight coupling between syntax & semantics
The intuition behind CCG

• Many words & phrases behave like functions
• E.g., *red* is like a function that maps *ball* into *red ball*
• *ball* and *red ball* are both noun-like things
• So, adjectives are like functions from nouns to nouns
• So let the syntactic category for adjectives reflect this
CCG categories

• Atomic categories: S, NP, N
  ◦ S: sentence [semantic type Bool]: *Texas is a state*
  ◦ NP: noun phrase [type Ind]: *Texas*
  ◦ N: noun [type Ind → Bool]: *state*

• Functional categories: X/Y and X\Y
  ◦ X/Y: takes arg of category Y to the right, returns category X
  ◦ X\Y: takes arg of category Y to the left, returns category X
  ◦ N/N: adjective [type ((Ind→Bool)→(Ind→Bool)]: *big*
  ◦ (S\NP)/NP: transitive verb [type (Ind→(Ind→Bool)]: *borders*
## CCG lexicon

<table>
<thead>
<tr>
<th>Words</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Texas</strong></td>
<td>NP : <em>texas</em></td>
</tr>
<tr>
<td><strong>state</strong></td>
<td>N : $\lambda x.\text{state}(x)$</td>
</tr>
<tr>
<td><strong>borders</strong></td>
<td>(S\NP)/NP : $\lambda x.\lambda y.\text{borders}(y, x)$</td>
</tr>
<tr>
<td><strong>what</strong></td>
<td>(S/(S\NP))/N : $\lambda f.\lambda g.\lambda x.f(x) \land g(x)$</td>
</tr>
</tbody>
</table>
Parsing rules (combinators)

Forward application

\[
\begin{align*}
X/Y & : f \\
Y & : a \\
\Rightarrow & X : f(a)
\end{align*}
\]

\[
\begin{align*}
(S/NP)/NP & : \\
NP & : \\
\Rightarrow & S/NP :
\end{align*}
\]

\[
\begin{align*}
\lambda x.\lambda y.\text{borders}(y, x) & : \text{texas} \\
\Rightarrow & \lambda y.\text{borders}(y, \text{texas})
\end{align*}
\]

Reverse application

\[
\begin{align*}
Y & : a \\
X/Y & : f \\
\Rightarrow & X : f(a)
\end{align*}
\]

\[
\begin{align*}
NP & : \\
(S/NP) & : \\
\Rightarrow & S :
\end{align*}
\]

\[
\begin{align*}
kansas & : \\
\lambda y.\text{borders}(y, \text{texas}) & : \\
\Rightarrow & \text{borders}(kansas, \text{texas})
\end{align*}
\]
Parsing rules (combinators)

Forward composition

\[ X/Y : f \quad Y/Z : g \quad \Rightarrow \Box X/Z : \lambda x.f(g(x)) \]

\[ N/N : \quad N/N : \quad \Rightarrow \Box N/N : \]

\[ \lambda f.\lambda x.f(x) \quad \lambda f.\lambda x.f(x) \quad \lambda f.\lambda x.f(x) \]

\[ \land \text{big}(x) \quad \land \text{red}(x) \quad \land \text{big}(x) \land \text{red}(x) \]

Reverse composition

\[ Y\backslash Z : g \quad X\backslash Y : f \quad \Rightarrow \Box X\backslash Z : \lambda x.f(g(x)) \]
Parsing rules (combinators)

Type-raising

\[ Y : x \quad \Rightarrow \quad X/(X\setminus Y) : \lambda f. f(x) \]

\[ NP : utah \quad \Rightarrow \quad S/(S\setminus NP) : \lambda f. f(utah) \]
CCG parsing

<table>
<thead>
<tr>
<th>Texas</th>
<th>borders</th>
<th>Kansas</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP : texas</td>
<td>(S\NP)/NP : (\lambda x.\lambda y.)</td>
<td>NP : kansas</td>
</tr>
<tr>
<td></td>
<td>borders(y, x))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S\NP : (\lambda y.)borders(y, kansas))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S : borders(texas, kansas)</td>
<td></td>
</tr>
</tbody>
</table>
### Parsing a question

<table>
<thead>
<tr>
<th>What</th>
<th>states</th>
<th>border</th>
<th>Texas</th>
</tr>
</thead>
<tbody>
<tr>
<td>( (S/(S\NP))/N ) : ( \lambda f. \lambda g. \lambda x. f(x) \land g(x) )</td>
<td>( N : \lambda x. \text{state}(x) )</td>
<td>( (S\NP)/\NP : \lambda x. \lambda y. \text{borders}(y, x) )</td>
<td>( \text{NP} : \text{texas} )</td>
</tr>
<tr>
<td>( S/(S\NP) : \lambda g. \lambda x. \text{state}(x) \land g(x) )</td>
<td>( S\NP : \lambda y. \text{borders}(y, \text{texas}) )</td>
<td>( S : \lambda x. \text{state}(x) \land \text{borders}(x, \text{texas}) )</td>
<td></td>
</tr>
</tbody>
</table>
Probabilistic CCG (PCCG)

A log-linear model:

• A CCG for parsing
• Lexical features
  \( f_i(L, S, T) \): number of times lexical item \( i \) is used in the parse \( T \) that maps from sentence \( S \) to logical form \( L \)
• A parameter vector \( \theta \) with an entry for each \( f_i \)
PCCG distributions

Log-linear model defines a joint distribution:

\[
P(L, T|S; \theta) = \frac{e^{f(L,T,S) \cdot \theta}}{\sum_{(L,T)} e^{f(L,T,S) \cdot \theta}}
\]

Parses are a hidden variable:

\[
P(L|S; \theta) = \sum_T P(L, T|S; \theta)
\]
PCCG parsing algorithm

• Given sentence $S$, find most probable logical form $L$

$$\text{argmax}_L P(L|S; \theta) = \text{argmax}_L \sum_T P(L, T|S; \theta)$$

• Uses dynamic programming: “chart parsing”
  ◦ Like the CKY algorithm for parsing in a PCFG (covered in CS224N!)
  ◦ Chart contains one cell for every possible token span in sentence
  ◦ Cells are populated bottom-up by combining smaller spans

• Uses a beam to track (& sum over) most likely trees
  ◦ Inference is therefore approximate, but effective in practice
Learning

• Generating lexical items
  (Promiscuously generate any lexicon entries which might help parse.)

• Learning PCCG parameters / weights
  (Learn which lexical items are good by hill-climbing on training data.)
What makes learning hard?

Why is this a harder learning problem than, e.g., learning to parse the Penn Treebank?

• We have to learn the lexicon, not just rule weights
• Training data gives only logical forms, not parse trees
Lexical generation

Input: training example

Sentence: Texas borders Kansas
Logical form: borders(texas, kansas)

lexical generation (GENLEX)

Output: lexicon entries

Texas := NP : texas
borders := (S\NP)/NP : λx.λy.borders(y, x)
Kansas := NP : kansas
GENLEX

Input: a training example \((S_i, L_i)\)

Computation:
1. Create all token sequences (n-grams) in \(S_i\)
2. Create categories from \(L_i\) (using rules, below)
3. Create lexical entries from the cross product (i.e., the Cartesian product) of these two sets

Output: lexicon \(\Lambda\)
GENLEX step 1: generate n-grams

Input: sentence

Output: n-grams

Texas borders Kansas

Texas
borders
Kansas
Texas borders
borders Kansas
Texas borders Kansas
GENLEX step 2: generate categories

Input: logical form

\[ \text{borders(Texas, Kansas)} \]

Output: categories

\[ \ldots \]

\[ \ldots \]

\[ \ldots \]

\[ \ldots \]
Two GENLEX rules

Input trigger
- a constant c
- an arity-two predicate p

Output category
- NP : c
- (S\NP)/NP : λx.λy.p(y, x)

Example input: logical form
borders(Texas, Kansas)

Example output: categories
NP : texas
NP : kansas
(S\NP)/NP : λx.λy.borders(y, x)
All the GENLEX rules

<table>
<thead>
<tr>
<th>Input trigger</th>
<th>Output category</th>
</tr>
</thead>
<tbody>
<tr>
<td>a constant c</td>
<td>NP : c</td>
</tr>
<tr>
<td>arity-one predicate p</td>
<td>N : \lambda x. p(x)</td>
</tr>
<tr>
<td></td>
<td>S/NP : \lambda x. p(x)</td>
</tr>
<tr>
<td>arity-two predicate p</td>
<td>(S/NP)/NP : \lambda x. \lambda y. p(y, x)</td>
</tr>
<tr>
<td>arity-two predicate p</td>
<td>(S/NP)/NP : \lambda x. \lambda y. p(x, y)</td>
</tr>
<tr>
<td>arity-one predicate p</td>
<td>N/N : \lambda g. \lambda x. p(x) \land g(x)</td>
</tr>
<tr>
<td>arity-two predicate p and constant c</td>
<td>N/N : \lambda g. \lambda x. p(x, c) \land g(x)</td>
</tr>
<tr>
<td></td>
<td>(N\N)/NP : \lambda x. \lambda g. \lambda y. p(y, x) \land g(x)</td>
</tr>
<tr>
<td>arity-two predicate p</td>
<td>NP/N : \lambda g. \text{argmax}</td>
</tr>
<tr>
<td>arity-one function f</td>
<td>S/NP : \lambda x. f(x)</td>
</tr>
</tbody>
</table>
All the GENLEX rules, with examples

<table>
<thead>
<tr>
<th>Input Trigger</th>
<th>Output Category</th>
<th>Categories produced from logical form</th>
</tr>
</thead>
<tbody>
<tr>
<td>constant (c)</td>
<td>(NP : c)</td>
<td>(\text{arg max}(\lambda x.\text{state}(x)) \land \text{borders}(x, \text{texas}), \lambda x.\text{size}(x)))</td>
</tr>
<tr>
<td>arity one predicate (p_1)</td>
<td>(N : \lambda x.p_1(x))</td>
<td>(NP : \text{texas})</td>
</tr>
<tr>
<td>arity one predicate (p_1)</td>
<td>(S\backslash NP : \lambda x.p_1(x))</td>
<td>(N : \lambda x.\text{state}(x))</td>
</tr>
<tr>
<td>arity two predicate (p_2)</td>
<td>((S\backslash NP)\backslash NP : \lambda x.\lambda y.p_2(y, x))</td>
<td>(S\backslash NP : \lambda x.\lambda y.\text{borders}(y, x))</td>
</tr>
<tr>
<td>arity two predicate (p_2)</td>
<td>((S\backslash NP)\backslash NP : \lambda x.\lambda y.p_2(x, y))</td>
<td>((S\backslash NP)\backslash NP : \lambda x.\lambda y.\text{borders}(x, y))</td>
</tr>
<tr>
<td>arity one predicate (p_1)</td>
<td>(N/N : \lambda g.\lambda x.p_1(x) \land g(x))</td>
<td>(N/N : \lambda g.\lambda x.\text{state}(x) \land g(x))</td>
</tr>
<tr>
<td>literal with arity two predicate (p_2) and constant second argument (c)</td>
<td>((N/N)\backslash NP : \lambda g.\lambda x.p_2(x, c) \land g(x))</td>
<td>((N/N)\backslash NP : \lambda g.\lambda x.\text{borders}(x, \text{texas}) \land g(x))</td>
</tr>
<tr>
<td>arity two predicate (p_2)</td>
<td>((N/N)\backslash NP : \lambda x.\lambda g.\lambda y.p_2(x, y) \land g(x))</td>
<td>((N/N)\backslash NP : \lambda g.\lambda x.\lambda y.\text{borders}(x, y) \land g(x))</td>
</tr>
<tr>
<td>an arg max / min with second argument arity one function (f)</td>
<td>(NP/N : \lambda g.\text{arg max} / \text{min}(g, \lambda x.f(x)))</td>
<td>(NP/N : \lambda g.\text{arg max}(g, \lambda x.\text{size}(x)))</td>
</tr>
<tr>
<td>an arity one numeric-ranged function (f)</td>
<td>(S/NP : \lambda x.f(x))</td>
<td>(S/NP : \lambda x.\text{size}(x))</td>
</tr>
</tbody>
</table>
## GENLEX step 3: cross product

### Input: training example

<table>
<thead>
<tr>
<th>Sentence</th>
<th>Logical form</th>
</tr>
</thead>
<tbody>
<tr>
<td>Texas borders Kansas</td>
<td>borders(texas, kansas)</td>
</tr>
</tbody>
</table>

### Output: lexicon entries

- Texas
- borders
- Kansas
- Texas borders
- borders Kansas
- Texas borders Kansas

NP : texas
NP : kansas
(S\NP)/NP :
\[ \lambda x.\lambda y.\text{borders}(y, x) \]
## GENLEX: output lexicon

<table>
<thead>
<tr>
<th>Expression</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Texas</td>
<td>:=</td>
<td>NP : texas</td>
</tr>
<tr>
<td>Texas</td>
<td>:=</td>
<td>NP : kansas</td>
</tr>
<tr>
<td>( \lambda x. \lambda y. \text{borders}(y, x) )</td>
<td>:=</td>
<td>( \lambda x. \lambda y. \text{borders}(y, x) )</td>
</tr>
<tr>
<td>borders</td>
<td>:=</td>
<td>NP : texas</td>
</tr>
<tr>
<td>borders</td>
<td>:=</td>
<td>NP : kansas</td>
</tr>
<tr>
<td>( \lambda x. \lambda y. \text{borders}(y, x) )</td>
<td>:=</td>
<td>( \lambda x. \lambda y. \text{borders}(y, x) )</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>\text{Texas borders Kansas}</td>
<td>:=</td>
<td>NP : texas</td>
</tr>
<tr>
<td>\text{Texas borders Kansas}</td>
<td>:=</td>
<td>NP : kansas</td>
</tr>
<tr>
<td>\text{Texas borders Kansas}</td>
<td>:=</td>
<td>(S\NP) / NP</td>
</tr>
<tr>
<td>\text{Texas borders Kansas}</td>
<td>:=</td>
<td>( \lambda x. \lambda y. \text{borders}(y, x) )</td>
</tr>
</tbody>
</table>
The initial lexicon

The initial lexicon $\Lambda_0$ has two types of entries:

- **Domain independent**
  
  \[ \text{what} := (S/(S\setminus NP))/N : \lambda f. \lambda g. \lambda x. f(x) \land g(x) \]

- **Domain dependent**
  
  \[ \text{Texas} := \text{NP} : \text{texas} \]
A simple learning algorithm

**Inputs**
- Initial lexicon $\Lambda_0$
- Training examples $E = \{(S_i, L_i) : i = 1...n\}$

**Initialization**
- Create lexicon $\Lambda^* = \Lambda_0 \cup \bigcup_{i=1}^{n} \text{GENLEX}(S_i, L_i)$
- Create initial parameters $\theta^0$

**Computation**
- Estimate parameters $\theta = \text{SGD}(E, \theta^0, \Lambda^*)$

**Output**
- Return $\text{PCCG}(\Lambda^*, \theta)$
The final learning algorithm

Inputs
\[ \Lambda_0, E = \{(S_i, L_i) : i = 1...n\} \]

Initialization
Create \( \theta^0 \)

Computation
For \( t = 1...T \)
  Prune lexicon:
    For each \((S_i, L_i) \in E:\)
      Set \( \lambda = \Lambda_0 \cup \text{GENLEX}(S_i, L_i) \)
      Calculate \( \pi = \text{PARSE}(S_i, L_i, \lambda, \theta^{t-1}) \)
      Define \( \lambda_i \) to be all lexical items in \( \pi \)
    Set \( \Lambda^t = \Lambda_0 \cup \bigcup_{i=1}^{n} \lambda_i \)
  Estimate parameters: \( \theta^t = \text{SGD}(E, \theta^{t-1}, \Lambda^t) \)

Output
Return \( \text{PCCG}(\Lambda^T, \theta^T) \)
Experiments

Two database domains:

• Geo880
  ◦ 600 training examples
  ◦ 280 test examples

• Jobs640
  ◦ 500 training examples
  ◦ 140 test examples
what cities in texas have the highest number of citizens?
what is the area of the state with the smallest population density?
what state is des moines located in?
what are the major cities in states through which the mississippi runs?
what are the major cities in the smallest state in the us?
what is the capital of ohio?
what is the population of denver?
what is the biggest city in nebraska?
what are the major cities in new mexico?
what is the capital of california?
what is the capital of utah?
what states border states that border states that border states that border texas?
what are the population of mississippi?
where is mount whitney?
what is the population of the state with the largest area?
what is the capital of iowa?
what is the most populous state through which the mississippi runs?
how many states border on the state whose capital is boston?
which states does the longest river cross?
what is the capital of new york?
what is the smallest city in arkansas?
how many people live in mississippi?
where is san jose?
what states border michigan?
name the rivers in arkansas.
what rivers are in nevada?
could you tell me what is the highest point in the state of oregon?
what state which the mississippi runs through has the largest population?
what state borders new york?
which states border hawaii?
what is the population of atlanta ga?
which state is the smallest?
what is the largest city in missouri?
how much population does texas have?
give me the number of rivers in california?
how many states does iowa border?
what states border states that the ohio runs through?
which states border texas?
what is the population of dallas?
what is the state with the lowest point?
what is the capital of maryland?
what is the longest river in the states that border nebraska?
Evaluation

Test for completely correct semantics

• Precision:
  # correct / total # parsed

• Recall:
  # correct / total # sentences
## Results

<table>
<thead>
<tr>
<th></th>
<th>Geo880</th>
<th>Jobs640</th>
</tr>
</thead>
<tbody>
<tr>
<td>precision</td>
<td>recall</td>
<td>precision</td>
</tr>
<tr>
<td>Z&amp;C05</td>
<td>96.25</td>
<td>79.29</td>
</tr>
<tr>
<td>COCKTAIL</td>
<td>89.92</td>
<td>79.40</td>
</tr>
</tbody>
</table>
### Examples of learned lexical entries

<table>
<thead>
<tr>
<th></th>
<th>N</th>
<th>(S\NP)/NP</th>
<th>NP/N</th>
</tr>
</thead>
<tbody>
<tr>
<td>states</td>
<td>$\lambda x.\text{state}(x)$</td>
<td>$\lambda g.\lambda x.\text{major}(x) \land g(x)$</td>
<td>$\lambda g.\text{argmax}(g, \lambda x.\text{size}(x))$</td>
</tr>
<tr>
<td>major</td>
<td>$\lambda x.\text{population}(x)$</td>
<td>$\lambda x.\text{city}(x)$</td>
<td>$\lambda x.\text{river}(x)$</td>
</tr>
<tr>
<td>population</td>
<td>$\lambda x.\text{river}(x)$</td>
<td>$\lambda x.\text{river}(x)$</td>
<td>$\lambda g.\text{argmax}(g, \lambda x.\text{elev}(x))$</td>
</tr>
<tr>
<td>cities</td>
<td>$\lambda x.\text{river}(x)$</td>
<td>$\lambda x.\text{river}(x)$</td>
<td>$\lambda g.\text{argmax}(g, \lambda x.\text{len}(x))$</td>
</tr>
<tr>
<td>river</td>
<td>$\lambda x.\text{river}(x)$</td>
<td>$\lambda x.\text{river}(x)$</td>
<td>$\lambda g.\text{argmax}(g, \lambda x.\text{elev}(x))$</td>
</tr>
<tr>
<td>run through</td>
<td>$\lambda x.\lambda y.\text{traverse}(y, x)$</td>
<td>$\lambda x.\lambda y.\text{traverse}(y, x)$</td>
<td>$\lambda g.\text{argmax}(g, \lambda x.\text{elev}(x))$</td>
</tr>
<tr>
<td>the largest</td>
<td>$\lambda g.\text{argmax}(g, \lambda x.\text{size}(x))$</td>
<td>$\lambda g.\text{argmax}(g, \lambda x.\text{elev}(x))$</td>
<td>$\lambda g.\text{argmax}(g, \lambda x.\text{len}(x))$</td>
</tr>
<tr>
<td>rivers</td>
<td>$\lambda x.\text{river}(x)$</td>
<td>$\lambda x.\text{river}(x)$</td>
<td>$\lambda g.\text{argmax}(g, \lambda x.\text{elev}(x))$</td>
</tr>
<tr>
<td>the highest</td>
<td>$\lambda g.\text{argmax}(g, \lambda x.\text{size}(x))$</td>
<td>$\lambda g.\text{argmax}(g, \lambda x.\text{elev}(x))$</td>
<td>$\lambda g.\text{argmax}(g, \lambda x.\text{len}(x))$</td>
</tr>
<tr>
<td>the longest</td>
<td>$\lambda g.\text{argmax}(g, \lambda x.\text{size}(x))$</td>
<td>$\lambda g.\text{argmax}(g, \lambda x.\text{elev}(x))$</td>
<td>$\lambda g.\text{argmax}(g, \lambda x.\text{len}(x))$</td>
</tr>
</tbody>
</table>
Error analysis

Low recall: GENLEX is not general enough
  • Fails to parse 10% of training examples

Some unparsed examples include:
  • Through which states does the Mississippi run?
  • If I moved to California and learned SQL on Oracle could I find anything for 3000 on Unix?
Zettlemoyer & Collins 2007

• Shifted from Geo880 to ATIS (~5,000 examples)
  
  on may four atlanta to denver delta flight 257  
  show me information on delta from fort worth to philadelphia  
  april 22nd dallas to miami the latest nighttime departure one way

• Non-standard CCG combinators with learned costs
  ◦ e.g., relaxed function application, to allow flexible word order
  ◦ e.g., role-hypothesizing type-shifting, to insert missing words

• New online learning algorithm: structured perceptron
• Tackles (discourse-)context-dependent interpretation

• Uses interaction sequences from ATIS dataset
  
  show me the flights from boston to philly
  show me the ones that leave in the morning
  what kind of plane is used on these flights

• Extends CCG, $\lambda$-calculus with referential expressions
  
  E.g., $ones := N : \lambda x.!(e, t)(x)$, semantics to be recovered from context

• Derivations (features, scoring) now include context
Kwiatkowski et al. 2010

- Luke’s first paper as a UW prof
- Introduces UBL: unification-based learning
- Initial lexicon pairs whole sentences with logical forms
- UBL iteratively splits lexical items into smaller parts
- Experiments in four languages, two different MRs
And the hits just keep on coming ...

Artzi & Zettlemoyer 2011, *Bootstrapping Semantic Parsers from Conversations*

Kwiatkowski et al. 2011, *Lexical Generalization in CCG Grammar Induction for Semantic Parsing*

Matuszek et al. 2012, *Learning to Parse Natural Language Commands to a Robot Control System*

Artzi & Zettlemoyer 2013, *Weakly supervised learning of semantic parsers for mapping instructions to actions*

Kwiatkowski et al. 2013, *Scaling Semantic Parsers with On-the-fly Ontology Matching*