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

```
borders(texas, kansas)
```

Potential applications

- Natural language interfaces to databases
- Question answering
- Dialogue systems

Some training examples

Input: *What states border Texas?*
Output: `λx.state(x) ∧ borders(x, texas)`

Input: *What is the largest state?*
Output: `argmax(λx.state(x), λx.size(x))`

Input: *What states border the largest state?*
Output:
```
λx.state(x) ∧ borders(x,
  argmax(λy.state(y), λy.size(y)))
```

The approach

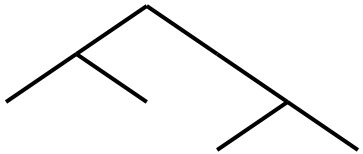
Learn lexical information (syntax & semantics) for words:

Texas := syntax = `NP` : semantics = `texas`

states := syntax = `N` : semantics = `λx.state(x)`

Learn to parse to logical form:

Input: *What states border Texas?*



Output: `λx.state(x) ∧ borders(x, 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

Words	Category Syntax : Semantics
Texas	`NP : texas`
state	`N : λx.state(x)`
borders	`(S\NP)/NP : λx.λy.borders(y, x)`
what	`(S/(S\NP))/N : λf.λg.λx.f(x) ∧ g(x)`

Parsing rules (combinators)

Forward application

X/Y : f	Y : a	➡	X : f(a)
(S\NP)/NP : λx.λy.borders(y, x)	NP : texas	➡	S\NP : λy.borders(y, texas)

Reverse application

Y : a	X\Y : f	➡	X : f(a)
NP : kansas	(S\NP) : λy.borders(y, texas)	➡	S : borders(kansas, texas)

Parsing rules (combinators)

Forward composition

```
X/Y : f        Y/Z : g        ➡ X/Z : λx.f(g(x))

N/N :          N/N :          ➡ N/N :
λf.λx.f(x)       λf.λx.f(x)       λf.λx.f(x)
  ∧ big(x)       ∧ red(x)         ∧ big(x) ∧ red(x)
```

Reverse composition

```
Y\Z : g      X\Y : f      ➡ X\Z : λx.f(g(x))
```

Parsing rules (combinators)

Type-raising

`Y : x` ➡ `X/(X\Y) : λf.f(x)`

`NP : utah` ➡ `S/(S\NP) : λf.f(utah)`

CCG parsing

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

Parsing a question

What	*states*	*border*	*Texas*
(S/(S\NP))/N : λf.λg.λx.f(x) ∧ g(x)	N : λx.state(x)	(S\NP)/NP : λx.λy.borders(y, x)	NP : texas
S/(S\NP) : λg.λx.state(x) ∧ g(x)		S\NP : λy.borders(y, texas)	
S : λx.state(x) ∧ borders(x, texas)			

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 θ 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 Λ

GENLEX step 1: generate n-grams

Input: sentence

Texas borders Kansas

Output: n-grams

Texas
borders
Kansas
Texas borders
borders Kansas
Texas borders Kansas

GENLEX step 2: generate categories

Input: logical form

```
borders(Texas, Kansas)
```

Output: categories

. . .

. . .

. . .

. . .

. . .

. . .

Two GENLEX rules

Input trigger		Output category
a constant `c`	➡	`NP : c`
an arity-two predicate `p`	➡	`(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

Input trigger	Output category
a constant `c`	`NP : c`
arity-one predicate `p`	`N : λx.p(x)`
arity-one predicate `p`	`S\NP : λx.p(x)`
arity-two predicate `p`	`(S\NP)/NP : λx.λy.p(y, x)`
arity-two predicate `p`	`(S\NP)/NP : λx.λy.p(x, y)`
arity-one predicate `p`	`N/N : λg.λx.p(x) ∧ g(x)`
arity-two predicate `p` and constant `c`	`N/N : λg.λx.p(x, c) ∧ g(x)`
	`(N\N)/NP : λx.λg.λy.p(y, x) ∧ g(x)`
arity-two predicate `p`	`NP/N : λg.argmax|min(g(x), λx.f(x))`
arity-one function `f`	`S/NP : λx.f(x)`
arity-one function `f`	

All the GENLEX rules, with examples

Rules		Categories produced from logical form
Input Trigger	Output Category	$\arg\max(\lambda x.state(x) \wedge borders(x, texas), \lambda x.size(x))$
constant c	$NP : c$	$NP : texas$
arity one predicate p_1	$N : \lambda x.p_1(x)$	$N : \lambda x.state(x)$
arity one predicate p_1	$S\backslash NP : \lambda x.p_1(x)$	$S\backslash NP : \lambda x.state(x)$
arity two predicate p_2	$(S\backslash NP)/NP : \lambda x.\lambda y.p_2(y, x)$	$(S\backslash NP)/NP : \lambda x.\lambda y.borders(y, x)$
arity two predicate p_2	$(S\backslash NP)/NP : \lambda x.\lambda y.p_2(x, y)$	$(S\backslash NP)/NP : \lambda x.\lambda y.borders(x, y)$
arity one predicate p_1	$N/N : \lambda g.\lambda x.p_1(x) \wedge g(x)$	$N/N : \lambda g.\lambda x.state(x) \wedge g(x)$
literal with arity two predicate p_2 and constant second argument c	$N/N : \lambda g.\lambda x.p_2(x, c) \wedge g(x)$	$N/N : \lambda g.\lambda x.borders(x, texas) \wedge g(x)$
arity two predicate p_2	$(N\backslash N)/NP : \lambda x.\lambda g.\lambda y.p_2(x, y) \wedge g(x)$	$(N\backslash N)/NP : \lambda g.\lambda x.\lambda y.borders(x, y) \wedge g(x)$
an arg max / min with second argument arity one function f	$NP/N : \lambda g.\arg\max / \min(g, \lambda x.f(x))$	$NP/N : \lambda g.\arg\max(g, \lambda x.size(x))$
an arity one numeric-ranged function f	$S/NP : \lambda x.f(x)$	$S/NP : \lambda x.size(x)$

GENLEX step 3: cross product

Input: training example

Sentence	*Texas borders Kansas*
Logical form	`borders(texas, kansas)`

Output: lexicon entries

Texas
borders
Kansas
Texas borders
borders Kansas
Texas borders Kansas

×

```
NP : texas
NP : kansas
(S\NP)/NP :
  λx.λy.borders(y,
x)
```

GENLEX: output lexicon

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

The initial lexicon

The initial lexicon Λ_0 has two types of entries:

- Domain independent

  what := `(S/(S\NP))/N : λf.λg.λx.f(x) ∧ g(x)`

- Domain dependent

  Texas := `NP : texas`

A simple learning algorithm

Inputs	Initial lexicon Λ_0 Training examples $E = \{(S_i, L_i) : i = 1...n\}$
Initialization	Create lexicon $\Lambda^* = \Lambda_0 \cup \bigcup_{i=1}^{n} \mathrm{GENLEX}(S_i, L_i)$ Create initial parameters θ^0
Computation	Estimate parameters $\theta = \mathrm{SGD}(E, \theta^0, \Lambda^*)$
Output	Return $\mathrm{PCCG}(\Lambda^*, \theta)$

The final learning algorithm

Inputs Λ_0, $E = \{(S_i, L_i) : i = 1...n\}$

Initialization Create θ^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 λ_i to be all lexical items in π
  - 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

Examples from Geo880

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?
which state has the highest peak in the country?
what river runs through illinois?
how many people live in the capital of georgia?
what is largest capital?
how many states are in the usa?
how many big cities are in pennsylvania?
what state contains the highest point in the us?
where is san jose?
how many cities are in montana?
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

	Geo880		Jobs640	
	precision	recall	precision	recall
Z&C05	96.25	79.29	97.36	79.29
COCKTAIL	89.92	79.40	93.25	79.84

Examples of learned lexical entries

states	N	:	λx.state(x)
major	N/N	:	λg.λx.major(x) ∧ g(x)
population	N	:	λx.population(x)
cities	N	:	λx.city(x)
river	N	:	λx.river(x)
run through	(S\NP)/NP	:	λx.λy.traverse(y, x)
the largest	NP/N	:	λg.argmax(g, λx.size(x))
rivers	N	:	λx.river(x)
the highest	NP/N	:	λg.argmax(g, λx.elev(x))
the longest	NP/N	:	λg.argmax(g, λx.len(x))

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

Zettlemoyer & Collins 2009

- 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, λ-calculus with referential expressions
  - E.g., *ones* := N : λx.!⟨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