Online Learning of Relaxed CCG Grammars for Parsing to Logical Form

Luke Zettlemoyer and Michael Collins

MIT Computer Science and Artificial Intelligence Lab

CSAIL

Learn Mappings to Logical Form

Given training examples like:

Input: `List one way flights to Prague.`

Output: $\lambda x.flight(x) \land one\_way(x) \land to(x, PRG)$

Challenging Learning Problem:

- Derivations (or parses) are not annotated

Extending previous approach: [Zettlemoyer & Collins 2005]

- Learn a lexicon and parameters for a weighted Combinatory Categorial Grammar (CCG)

Challenge

Learning CCG grammars works well for complex, grammatical sentences:

Input: `Show me flights from Newark and New York to San Francisco or Oakland that are nonstop.`

Output: $\lambda x.flight(x) \wedge nonstop(x) \wedge (from(x,PRG) \vee from(x,NYC)) \wedge (to(x,SFO) \vee to(x,OAK))$

What about sentences that are common given spontaneous, unedited input?

Input: `Boston to Prague the latest on Friday.`

Output: $argmax(\ \lambda x.from(x,BOS) \wedge to(x,PRG) \wedge day(x,FRI),\ \lambda y.time(y))$

This talk is about an approach that works for both cases.

Outline

- Background
- Relaxed parsing rules
- Online learning algorithm
- Evaluation

Background

- Combinatory Categorial Grammar (CCG)
- Weighted CCGs
- Learning lexical entries: GENLEX

CCG Lexicon

Words	Category
flights	N : $\lambda x.flight(x)$
to	(N\N)/NP : $\lambda x.\lambda f.\lambda y.f(x) \wedge to(y,x)$
Prague	NP : PRG
New York city	NP : NYC
...	...

Parsing Rules (Combinators)

Application

- `X/Y : f      Y : a   =>   X : f(a)`
- `  Y : a    X\Y : f   =>   X : f(a)`

Composition

- `X/Y : f   Y/Z : g   =>  X/Z : λx.f(g(x))`
- `Z\Y : f   X\Y : g   =>  X\Z : λx.f(g(x))`

Additional rules:

- Type Raising
- Crossed Composition

CCG Parsing

Show me	flights	to	Prague
S/N	N	(N\N)/NP	NP
$\lambda f.f$	$\lambda x.flight(x)$	$\lambda y.\lambda f.\lambda x.f(y) \wedge to(x,y)$	PRG

N\N

$\lambda f.\lambda x.f(x) \wedge to(x,PRG)$

N

$\lambda x.flight(x) \wedge to(x,PRG)$

S

$\lambda x.flight(x) \wedge to(x,PRG)$

Weighted CCG

Given a log-linear model with a CCG lexicon Λ, a feature vector f, and weights w.

- The best parse is:

$$y^* = \operatorname*{argmax}_y \; w \cdot f(x, y)$$

Where we consider all possible parses y for the sentence x given the lexicon Λ.

Lexical Generation

Input Training Example

Sentence: Show me flights to Prague.

Logic Form: $\lambda x.flight(x) \wedge to(x,PRG)$

Output Lexicon

Words	Category
Show me	S/N : $\lambda f.f$
flights	N : $\lambda x.flight(x)$
to	(N\N)/NP : $\lambda x.\lambda f.\lambda y.f(x) \wedge to(y,x)$
Prague	NP : PRG
...	...

GENLEX: Substrings cross Categories

Input Training Example

Sentence: `Show me flights to Prague.`

Logic Form: $\lambda x.flight(x) \wedge to(x, PRG)$

Output Lexicon

All possible substrings:

```
Show
me
flights
...
Show me
Show me flights
Show me flights to
...
```

X

Categories created by rules that trigger on the logical form:

NP : PRG

N : $\lambda x.flight(x)$

(S\NP)/NP : $\lambda x.\lambda y.to(y,x)$

(N\N)/NP : $\lambda y.\lambda f.\lambda x.$ …

...

[Zettlemoyer & Collins 2005]

Challenge Revisited

The lexical entries that work for:

Show me	the latest	flight	from Boston	to Prague	on Friday
S/NP	NP/N	N	N\N	N\N	N\N
…	…	…	…	…	…

Will not parse:

Boston	to Prague	the latest	on Friday
NP	N\N	NP/N	N\N
…	…	…	…

Relaxed Parsing Rules

Two changes:

- Add application and composition rules that relax word order
- Add type shifting rules to recover missing words

These rules significantly relax the grammar

- Introduce features to count the number of times each new rule is used in a parse

Review: Application

```
X/Y : f        Y : a    =>   X : f(a)
Y : a        X\Y : f    =>   X : f(a)
```

Disharmonic Application

- Reverse the direction of the principal category:

$$\texttt{X\textbackslash Y : f} \quad \texttt{Y : a} \quad \Rightarrow \quad \texttt{X : f(a)}$$

$$\texttt{Y : a} \quad \texttt{X/Y : f} \quad \Rightarrow \quad \texttt{X : f(a)}$$

$$\dfrac{\dfrac{\textbf{flights}}{\begin{array}{c}\mathbf{N}\\ \lambda x.flight(x)\end{array}} \qquad \dfrac{\textbf{one way}}{\begin{array}{c}\mathbf{N/N}\\ \lambda f.\lambda x.f(x) \wedge one\_way(x)\end{array}}}{\begin{array}{c}\mathbf{N}\\ \lambda x.flight(x) \wedge one\_way(x)\end{array}}$$

Review: Composition

```
X/Y : f     Y/Z : g    =>   X/Z : λx.f(g(x))
Y\Z : g     X\Y : f    =>   X\Z : λx.f(g(x))
```

Disharmonic Composition

- Reverse the direction of the principal category:

$$X\backslash Y : f \quad Y/Z : g \quad \Rightarrow \quad X/Z : \lambda x.f(g(x))$$

$$Y\backslash Z : g \quad X/Y : f \quad \Rightarrow \quad X\backslash Z : \lambda x.f(g(x))$$

$$\dfrac{\dfrac{\dfrac{\text{to Prague}}{\begin{array}{c}\mathbf{N\backslash N}\\ \lambda f.\lambda x.f(x)\wedge to(x,PRG)\end{array}} \quad \dfrac{\text{the latest}}{\begin{array}{c}\mathbf{NP/N}\\ \lambda f.argmax(\lambda x.f(x),\lambda x.time(x))\end{array}}}{\begin{array}{c}\mathbf{NP\backslash N}\\ \lambda f.argmax(\lambda x.f(x)\wedge to(x,PRG),\ \lambda x.time(x))\end{array}} \quad \dfrac{\text{flight}}{\begin{array}{c}\mathbf{N}\\ \lambda x.flight(x)\end{array}}}{\begin{array}{c}\mathbf{N}\\ argmax(\lambda x.flight(x)\wedge to(x,PRG),\ \lambda x.time(x))\end{array}}$$

Missing content words

Insert missing semantic content

- `NP : c  =>  N\N : λf.λx.f(x) ∧ p(x,c)`

$$
\cfrac{
  \cfrac{\textbf{flights}}{\mathbf{N} \\ \lambda x.\mathit{flight}(x)}
  \quad
  \cfrac{\cfrac{\textbf{Boston}}{\mathbf{NP} \\ \mathit{BOS}}}{\mathbf{N\backslash N} \\ \lambda f.\lambda x.f(x) \wedge \mathit{from}(x, \mathrm{BOS})}
}{\mathbf{N} \\ \lambda x.\mathit{flight}(x) \wedge \mathit{from}(x, \mathrm{BOS})}
\quad
\cfrac{\textbf{to Prague}}{\mathbf{N\backslash N} \\ \lambda f.\lambda x.f(x) \wedge \mathit{to}(x, \mathrm{PRG})}
$$

$$
\cfrac{}{\mathbf{N} \\ \lambda x.\mathit{flight}(x) \wedge \mathit{from}(x, \mathrm{BOS}) \wedge \mathit{to}(x, \mathrm{PRG})}
$$

Missing content-free words

Bypass missing nouns

- `N\N : f => N : f(λx.true)`

Northwest Air	**to Prague**
N/N $\lambda f.\lambda x.f(x) \wedge airline(x,\mathrm{NWA})$	**N\N** $\lambda f.\lambda x.f(x) \wedge to(x,\mathrm{PRG})$
	N $\lambda x.to(x,\mathrm{PRG})$

N

$\lambda x.airline(x,\mathrm{NWA}) \wedge to(x,\mathrm{PRG})$

A Complete Parse

Boston	to Prague	the latest	on Friday
NP BOS	N\N $\lambda f.\lambda x.f(x) \wedge to(x, \text{PRG})$	NP/N $\lambda f.argmax(\lambda x.f(x), \lambda x.time(x))$	N\N $\lambda f.\lambda x.f(x) \wedge day(x, \text{FRI})$

N\N
$\lambda f.\lambda x.f(x) \wedge from(x, \text{BOS})$

N
$\lambda x.day(x, \text{FRI})$

N\N
$\lambda f.\lambda x.f(x) \wedge from(x, \text{BOS}) \wedge to(x, \text{PRG})$

NP\N
$\lambda f.argmax(\lambda x.f(x) \wedge from(x, \text{BOS}) \wedge to(x, \text{PRG}), \lambda x.time(x))$

N
$argmax(\lambda x.from(x, \text{BOS}) \wedge to(x, \text{PRG}) \wedge day(x, \text{FRI}), \lambda x.time(x))$

A Learning Algorithm

The approach is:

- Online: processes data set one example at a time
- Able to Learn Structure: selects a subset of the lexical entries from GENLEX
- Error Driven: uses perceptron-style parameter updates
- Relaxed: learns how much to penalize the use of the relaxed parsing rules

Inputs: Training set $\{(x_i, z_i) \mid i=1\ldots n\}$ of sentences and logical forms. Initial lexicon Λ. Initial parameters w. Number of iterations T.

Computation: For $t = 1\ldots T,\ i = 1\ldots n$:

Step 1: Check Correctness

- Let $y^* = \arg\max_y w \cdot f(x_i, y)$
- If $L(y^*) = z_i$, go to the next example

Step 2: Lexical Generation

- Set $\lambda = \Lambda \cup \text{GENLEX}(x_i, z_i)$
- Let $\hat{y} = \arg\max_{y\ s.t.\ L(y)=z_i} w \cdot f(x_i, y)$
- Define λ_i to be the lexical entries in y^*
- Set lexicon to $\Lambda = \Lambda \cup \lambda_i$

Step 3: Update Parameters

- Let $y' = \arg\max_y w \cdot f(x_i, y)$
- If $L(y') \neq z_i$
    - Set $w = w + f(x_i, \hat{y}) - f(x_i, y')$

Output: Lexicon Λ and parameters w.

Related Work

Semantic parsing with:

- Inductive Logic Prog. [Zelle, Mooney 1996; Thompson, Mooney 2002]
- Machine Translation [Papineni et al. 1997; Wong, Mooney 2006, 2007]
- Probabilistic CFG Parsing [Miller et. al, 1996; Ge, Mooney 2006]
- Support Vector Mach. [Kate, Mooney 2006; Nguyen et al. 2006]

CCG: [Steedman 1996, 2000]

- Log-linear models [Clark, Curran 2003]
- Multi-modal CCG [Baldridge 2002]
- Wide coverage semantics [Bos et al. 2004]
- CCG Bank [Hockenmaier 2003]

Related Work for Evaluation

Hidden Vector State Model: He and Young 2006

- Learns a probabilistic push-down automaton with EM
- Is integrated with speech recognition

λ-WASP: Wong & Mooney 2007

- Builds a synchronous CFG with statistical machine translation techniques
- Easily applied to different languages

Zettlemoyer and Collins 2005

- Uses GENLEX with maximum likelihood batch training and stricter grammar

Two Natural Language Interfaces

ATIS (travel planning)

- Manually-transcribed speech queries
- 4500 training examples
- 500 example development set
- 500 test examples

Geo880 (geography)

- Edited sentences
- 600 training examples
- 280 test examples

Evaluation Metrics

Precision, Recall, and F-measure for:

- Completely correct logical forms
- Attribute / value partial credit

$$\lambda x.flight(x) \wedge from(x,BOS) \wedge to(x,PRG)$$

is represented as:

$$\{from = BOS,\ to = PRG\}$$

Two-Pass Parsing

Simple method to improve recall:

- For each test sentence that can not be parsed:
  - Reparse with word skipping
  - Every skipped word adds a constant penalty
  - Output the highest scoring new parse

We report results with and without this two-pass parsing strategy

ATIS Test Set

Exact Match Accuracy:

	Precision	Recall	F1
Single-Pass	90.61	81.92	**86.05**
Two-Pass	85.75	84.60	85.16

ATIS Test Set

Partial Credit Accuracy:

	Precision	Recall	F1
Single-Pass	96.76	86.89	91.56
Two-Pass	95.11	96.71	**95.9**
He & Young 2006	---	---	90.3

Geo880 Test Set

Exact Match Accuracy:

	Precision	Recall	F1
Single-Pass	95.49	83.20	**88.93**
Two-Pass	91.63	86.07	88.76
Zettlemoyer & Collins 2005	96.25	79.29	86.95
Wong & Money 2007	93.72	80.00	86.31

ATIS Development Set

Exact Match Accuracy:

	Precision	Recall	F1
Full online method	87.26	74.44	**80.35**
Without features for new rules	70.33	42.45	52.95
Without relaxed word order rules	82.81	63.98	72.19
Without missing word rules	77.31	56.94	65.58

Summary

We presented an algorithm that:

- Learns the lexicon and parameters for a weighted CCG
  - Introduces operators to parse relaxed word order and recover missing words
  - Uses online, error-driven updates
- Improves parsing accuracy for spontaneous, unedited inputs
- Maintains the advantages of using a detailed grammatical formalism

The End

Thanks