Learning to Map Sentences to Logical form with PCCG

Zettlemoyer and Collins 2005
21st Conference on Uncertainty in Artificial Intelligence

Overview

Learning mapping from S to L
from text to logical form

For example
What states border Texas?
λX.state(X) and borders(X,texas)

Issues

- Needs to find predicate names from words
  - relate “border” to “borders(X,Y)”
- Will use log-linear models
  - Like Conditional Random Fields (CRFs)
- Concentrate of “structure learning”
  - In CCG that means “lexical entries”

Domain: GEO880

- GEO880
  - 880 utterances (sentences plus logical forms)
  - US Geography questions
  - Which states border Texas?
  - Which states border Texas that border Utah?
- History:
  - CHAT80 (Pereira)

JOBS640

- JOBS640
  - 640 queries about jobs
  - Which jobs offers more than $60K?
  - Which jobs offers more than $80K and is in Tuscon?

Semantics

- *Constants:*
  - *texas* is of type *e*
  - *state is of type <e,t>*
  - *size maps entities to real numbers <e,r>*
- *Logical conectors*
  - *and, or, not, implies*
- *Quantification*
  - *Forall, Exists*
- *Lambda expressions:*
  - *λX.state(X) and borders(X,texas)*

Semantics

- *Additional quantifiers*
  - *count(λX.state(X) and borders(X,texas)*
  - *argmax(λX.state(X), λX.size(X))*
  - *def(λX.state(X))*
    - *Unique item for which state(X) is true*

CCG

The functional application rules:

a. $A/B \quad B \quad \Rightarrow \quad A$

b. $B \quad A\backslash B \quad \Rightarrow \quad A$

The functional application rules (with semantics):

a. $A/B : f \quad B : g \quad \Rightarrow \quad A : f(g)$

b. $B : g \quad A\backslash B : f \quad \Rightarrow \quad A : f(g)$

CCG Lexicon

Utah	:=	NP
Idaho	:=	NP
borders	:=	$(S\backslash NP)/NP$
the Mississippi	:=	$NP : mississippi\_river$

CCG example

$$
\begin{array}{cccl}
\text{Utah} & \text{borders} & \text{Idaho} & \\
\cline{1-1}\cline{2-2}\cline{3-3}
NP & (S\backslash NP)/NP & NP & \\
utah & \lambda x.\lambda y.borders(y, x) & idaho & \\
\cline{2-3}
 & \multicolumn{2}{c}{(S\backslash NP)} & > \\
 & \multicolumn{2}{c}{\lambda y.borders(y, idaho)} & \\
\cline{1-3}
\multicolumn{3}{c}{S} & < \\
\multicolumn{3}{c}{borders(utah, idaho)} &
\end{array}
$$

CCG example2

What	states	border	Texas
$(S/(S\backslash NP))/N$	N	$(S\backslash NP)/NP$	NP
$\lambda f.\lambda g.\lambda x.f(x) \wedge g(x)$	$\lambda x.state(x)$	$\lambda x.\lambda y.borders(y,x)$	$texas$

$$\frac{\dfrac{\text{What states}}{S/(S\backslash NP)\ \ \lambda g.\lambda x.state(x) \wedge g(x)}\ >\qquad \dfrac{\text{border Texas}}{(S\backslash NP)\ \ \lambda y.borders(y, texas)}\ >}{S\ \ \lambda x.state(x) \wedge borders(x, texas)}\ >$$

CCG (here)

- CCG five rules
  - Forward/backward application
  - Forward/bacward composition
  - Type raising
- Specific lexical entries for special words
  - “what”
- Lexical items can be multiword

PCCG

- Probabilistic CCGs
- <L,T>
  - L is logical form
  - T is sequence of steps to derive L
    - Will be called "parse tree"
- S a sentence
- A PCCG defines conditional distribution
  - P(L,T | S)

Ambiguity

- Ambiguity
  - Multiple <L,T> for S
- Multiple lexical entries
  - New York := NP:new_york_city
  - New York := NP:new_york_state
- Single L by multiple derivations (T)
  - If *same* L for S then “spurious ambiguity”
- Usually there is ambiguity
  - Which is good as that’s what is optimized in learning

PCCG

- Log linear models (like CRFs)
- f(L,T,S) maps to feature vectors R
  - f(L,T,S) ->
    - f1(L,T,S), f2(L,T,S), …
  - Where fn is count of some substructure in (L,T,S).
- Model is parameterized by vector θ

PCCG

- Probability of particular syntax/semantics

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

- Denominator is sum over all valid parses in S

PCCG

- *Features (fn)*
  - *Lexical features only*
  - *Count number times lexical entry used in T*
- *Over simplified but seems to work*

Parsing

- Parsing given a PCCG

$$\arg\max_{L} P(L|S;\bar{\theta}) = \arg\max_{L} \sum_{T} P(L,T|S;\bar{\theta})$$

Parameter Estimation

- Log-likelihood of the training set

$$
\begin{aligned}
O(\bar{\theta}) &= \sum_{i=1}^{n} \log P(L_i|S_i;\bar{\theta}) \\
&= \sum_{i=1}^{n} \log \left(\sum_{T} P(L_i, T|S_i;\bar{\theta}) \right)
\end{aligned}
$$

- Differentiate to get:

$$
\begin{aligned}
\frac{\partial O}{\partial \theta_j} &= \sum_{i=1}^{n} \sum_{T} f_j(L_i, T, S_i) P(T|S_i, L_i;\bar{\theta}) \\
&\quad - \sum_{i=1}^{n} \sum_{L,T} f_j(L, T, S_i) P(L, T|S_i;\bar{\theta})
\end{aligned}
$$

Parameter Estimation

- Estimate using dynamic programming
  - Use inside outside algorithm
- Maximize likelihood using
  - (EM) or
  - Stochastic gradient ascent

Set $\bar{\theta}$ to some initial value
for $k = 0 \ldots N-1$
  for $i = 1 \ldots n$
    $\bar{\theta} = \bar{\theta} + \frac{\alpha_0}{(1+ct)} \frac{\partial \log P(L_i|S_i;\bar{\theta})}{\partial \bar{\theta}}$

Learning

- *Learning*
  - *A Induction of a lexicon (structure learning)*
  - *Parameter estimation*

Lexical Entry Generation

- GENLEX
  - Generate entries give S and L that allow sentence to be parsed as L
  - Good examples
    - Utah := NP:utah
    - Idaho := NP:idaho
    - Borders := (S\NP)/NP:λX.λY.borders(X,Y)
  - Bad examples
    - Utah := NP:idaho
    - Idaho := (S\NP)/NP:λX.λY.borders(X,Y)

GENLEXLtriggers

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	$S/NP : \lambda x.f(x)$	$S/NP : \lambda x.size(x)$

PCCG:training

Inputs:

- Training examples $E = \{(S_i, L_i) : i = 1 \ldots n\}$ where each S_i is a sentence, each L_i is a logical form.
- An initial lexicon Λ_0

PCCG:procedures

Procedures:

- PARSE($S, L, \Lambda, \bar{\theta}$): takes as input a sentence S, a logical form L, a lexicon Λ, and a parameter vector $\bar{\theta}$. Returns the highest probability parse for S with logical form L, when S is parsed by a PCCG with lexicon Λ and parameters $\bar{\theta}$. If there is more than one parse with the same highest probability, the entire set of highest probability parses is returned. Dynamic programming methods are used when implementing PARSE, see section 2.4 of this paper.

- ESTIMATE($\Lambda, E, \bar{\theta}$): takes as input a lexicon Λ, a training set E, and a parameter vector $\bar{\theta}$. Returns parameter values $\bar{\theta}$ that are the output of stochastic gradient descent on the training set E under the grammar defined by Λ. The input $\bar{\theta}$ is the initial setting for the parameters in the stochastic gradient descent algorithm. Dynamic programming methods are used when implementing ESTIMATE, see section 2.4.

- GENLEX(S, L): takes as input a sentence S and a logical form L. Returns a set of lexical items. See section 3.1 for a description of GENLEX.

PCCG: algorithm

- Step 1
  - Search small number of sufficient lexical entries to parse S
  - Find highest scoring parse
  - Identify the entries that were used in best parse
- Step 2
  - Re-estimate the parameters
  - Using stocastic gradient ascent

PCCG: algorithm

Initialization: Define $\bar{\theta}$ to be a real-valued vector of arity $|\Lambda^*|$, where $\Lambda^* = \Lambda_0 \cup \bigcup_{i=1}^{n} \text{GENLEX}(S_i, L_i)$. $\bar{\theta}$ stores a parameter value for each potential lexical item. The initial parameters $\bar{\theta}^0$ are taken to be 0.1 for any member of Λ_0, and 0.01 for all other lexical items.

Algorithm:

- For $t = 1 \ldots T$

Step 1: (Lexical generation)

- For $i = 1 \ldots n$:
  - Set $\lambda = \Lambda_0 \cup \text{GENLEX}(S_i, L_i)$.
  - Calculate $\pi = \text{PARSE}(S_i, L_i, \lambda, \bar{\theta}^{t-1})$.
  - Define λ_i to be the set of lexical entries in π.
- Set $\Lambda_t = \Lambda_0 \cup \bigcup_{i=1}^{n} \lambda_i$

Step 2: (Parameter Estimation)

- Set $\bar{\theta}^t = \text{ESTIMATE}(\Lambda_t, E, \bar{\theta}^{t-1})$

Output: Lexicon Λ_T together with parameters $\bar{\theta}^T$.

Related Work

- Older NL interfaces to Databases
- X and Mooney work
  - Cocktail
  - Had explicit lexical entries, not using CCG
- ATIS (flight information)
  - BBN work (non-CCG)

Experiments

	Geo880		Jobs640	
	P	R	P	R
Our Method	96.25	79.29	97.36	79.29
COCKTAIL	89.92	79.40	93.25	79.84

Best Learned Entries

states	:=	$N : \lambda x.state(x)$
major	:=	$N/N : \lambda f.\lambda x.major(x) \wedge f(x)$
population	:=	$N : \lambda x.population(x)$
cities	:=	$N : \lambda x.city(x)$
rivers	:=	$N : \lambda x.river(x)$
run through	:=	$(S\backslash NP)/NP : \lambda x.\lambda y.traverse(y, x)$
the largest	:=	$NP/N : \lambda f.\arg\max(f, \lambda x.size(x))$
river	:=	$N : \lambda x.river(x)$
the highest	:=	$NP/N : \lambda f.\arg\max(f, \lambda x.elev(x))$
the longest	:=	$NP/N : \lambda f.\arg\max(f, \lambda x.len(x))$