Semantic parsing

CMSC 723 / LING 723 / INST 725

Hal Daumé III [he/him]

19 Nov 2019

Announcements, logistics

- Exam
  - We'll talk about it next class period (after the makeup exam is finished)
  - We will try to get grades out before class on Thr
- Homework 4/5:
  - HW4-programming will be up tomorrow, due last day of class (05 Dec)
  - HW5-written will be up Thursday (also due 05 Dec, topics through Thanksgiving break)
- P3 due Thursday (today is fine too)
- P4 due two weeks from today (posted now)

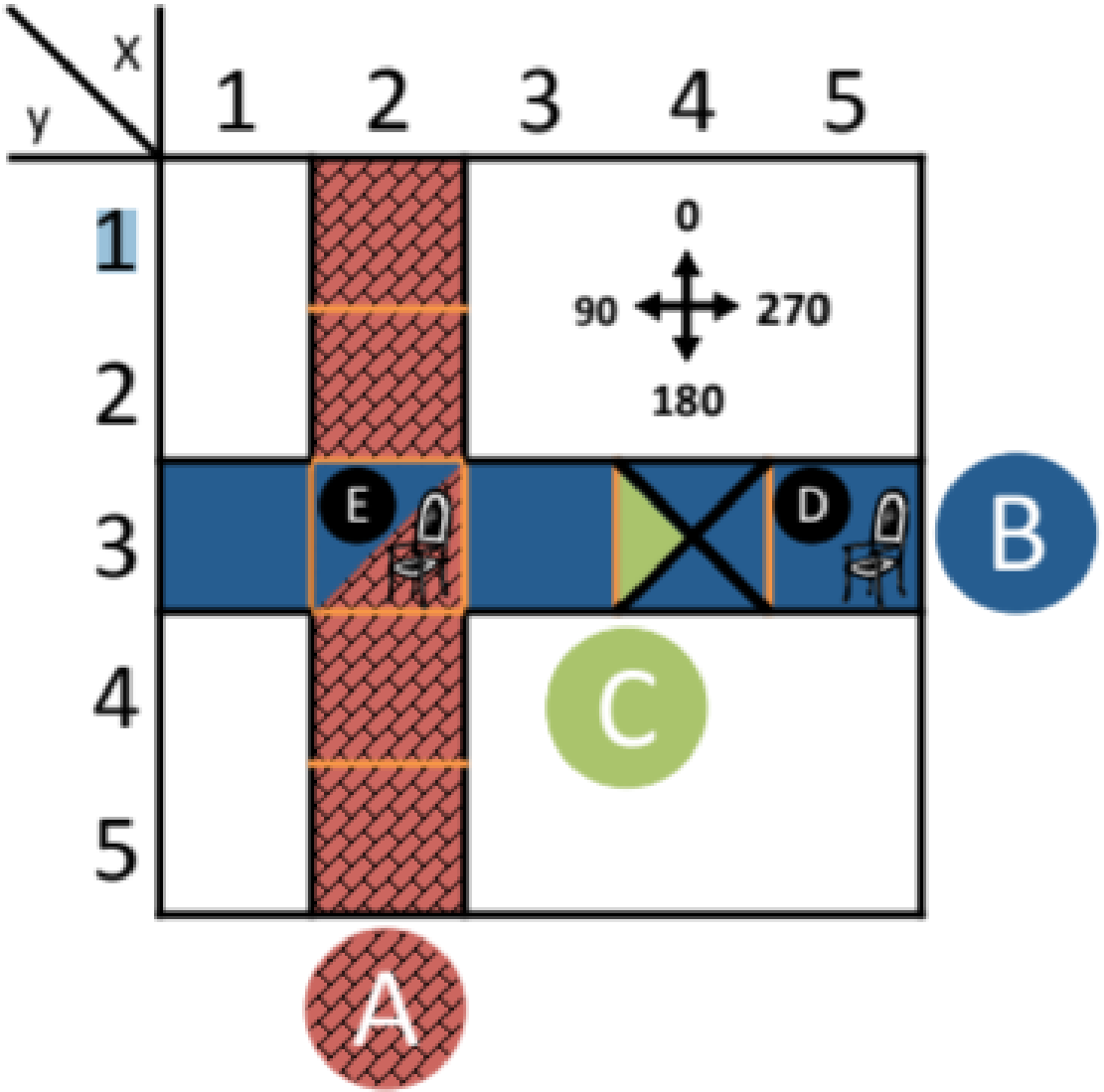
Today

- Semantic parsing
- CCG grammars & lambda calculus
- Semantic parsing with full supervision
- Semantic parsing with denotations alone

Semantic parsing of instructions

Weakly Supervised Learning of Semantic Parsers for Mapping Instructions to Actions

Yoav Artzi and **Luke Zettlemoyer**



(a) chair
$\lambda x.chair(x)$

(b) hall
$\lambda x.hall(x)$

(c) the chair
$\iota x.chair(x)$

(d) you
you

(e) blue hall
$\lambda x.hall(x) \wedge blue(x)$

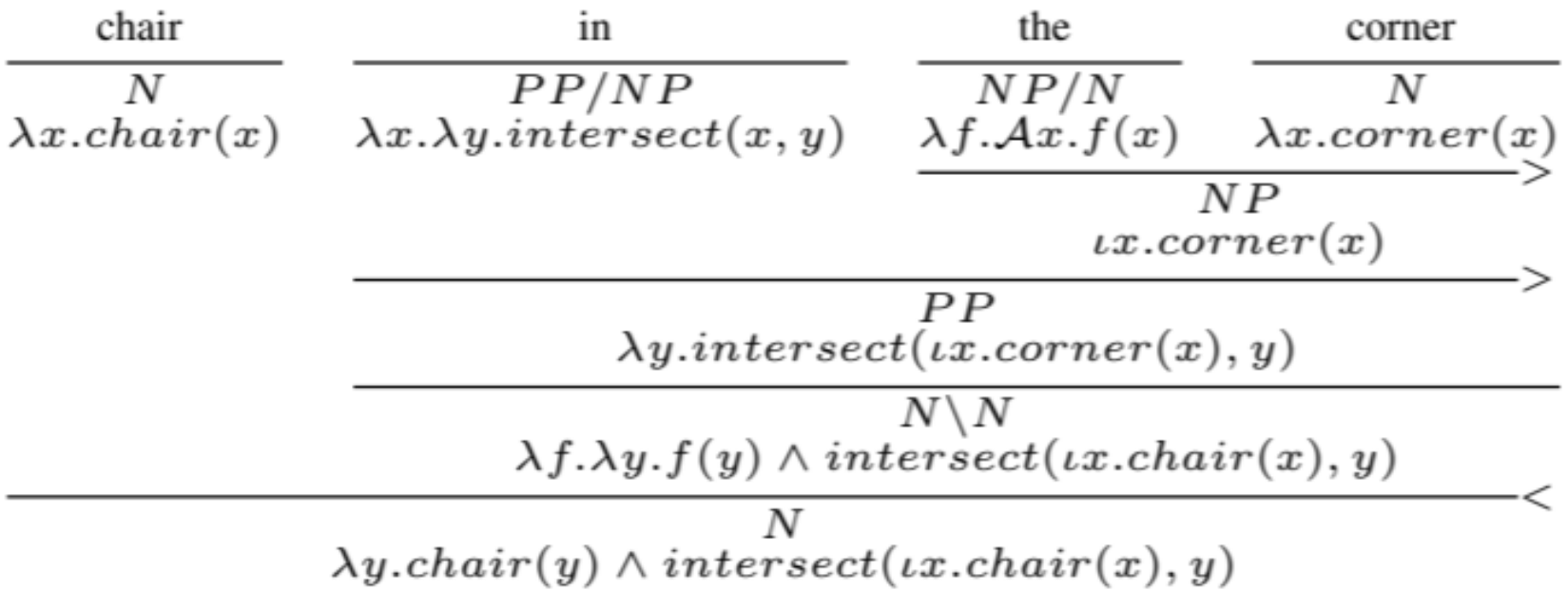
(f) chair in the intersection
$\lambda x.chair(x) \wedge$
$intersect(\iota y.junction(y), x)$

(g) in front of you
$\lambda x.in\_front\_of(you, x)$

Goal

Imp.: move from the sofa to the chair

LF: $\lambda a.move(a) \wedge to(a, \iota x.chair(x)) \wedge from(a, \iota y.sofa(y))$



CCG: lexicon

John := NP

Mary := NP

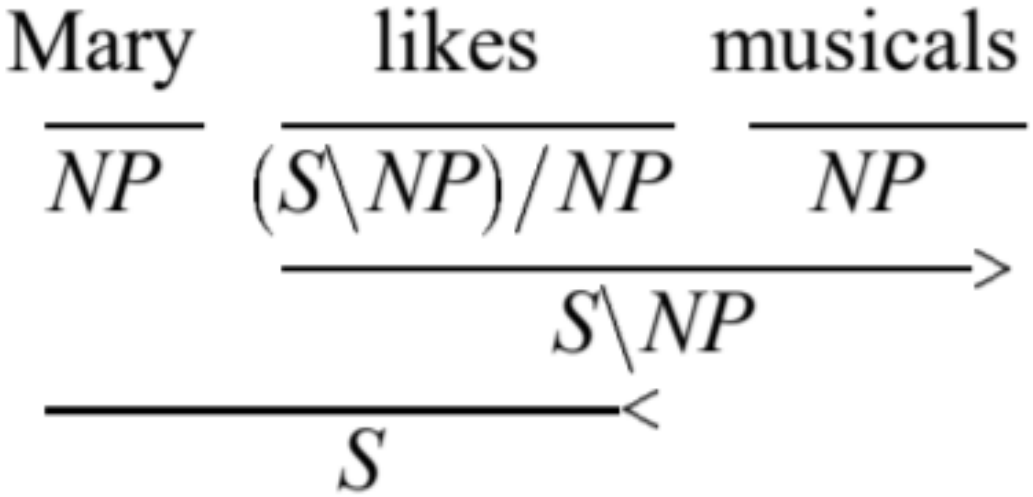
likes := (S\NP)/NP

CCG: application

John := NP

Mary := NP

likes := (S\NP)/NP



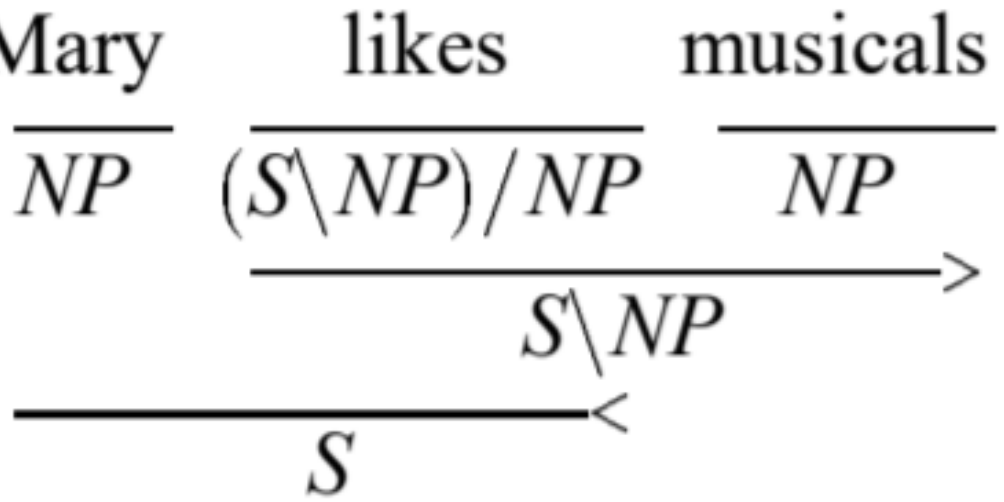
Forward App (>) X/Y + Y → X

Backward App (<) Y + X\Y → X

Parse “most birds fly”

Forward App (>) X/Y + Y → X

Backward App (<) Y + X\Y → X



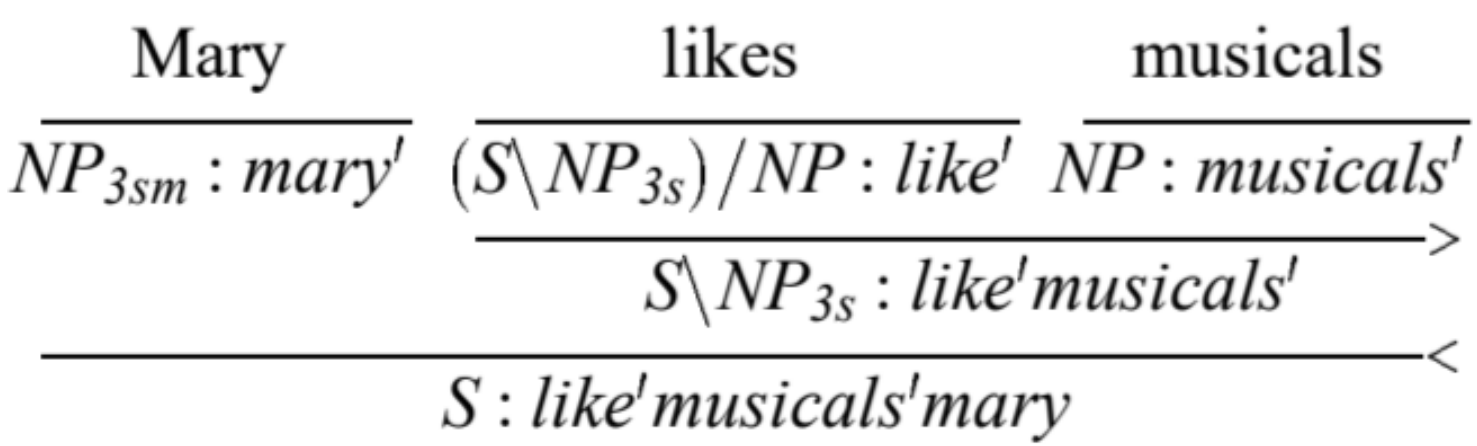
Word	Categories
fly	$S\backslash(S/(S\backslash NP))$ $S\backslash NP$
flies	$S\backslash NP(obj)$
swim	$S\backslash(S/(S\backslash NP))$ $S\backslash NP$
swims	$S\backslash NP(obj)$
most	$(S/(S\backslash NP))/NP$
do *do not*	$(S/(S\backslash NP))\backslash NP$ $(S/(S\backslash NP))\backslash NP$
is	$(S/NP)\backslash NP$ $(S/N)\backslash NP(obj)$
are *are not*	$(S/NP)\backslash NP$ $(S/NP)\backslash NP$
birds *penguins* *a penguin* *bats* *tweety, tim* *fictional* *a fictional*	N, NP N, NP $N, NP(obj)$ N, NP $NP(obj)$ NP/N NP/N

CCG: adding semantics

John := NP

Mary := NP

likes := (S\NP)/NP



Forward App (>) X/Y:f + Y:a → X:f(a)

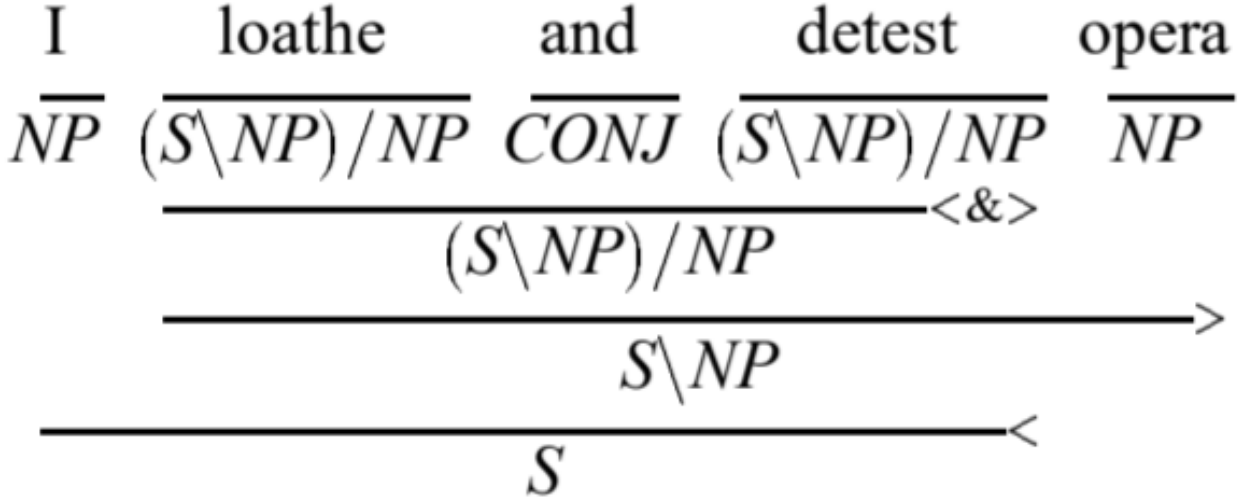
Backward App (<) Y:a + X\Y:f → X:f(a)

CCG: adding semantics

John := NP

Mary := NP

likes := (S\NP)/NP



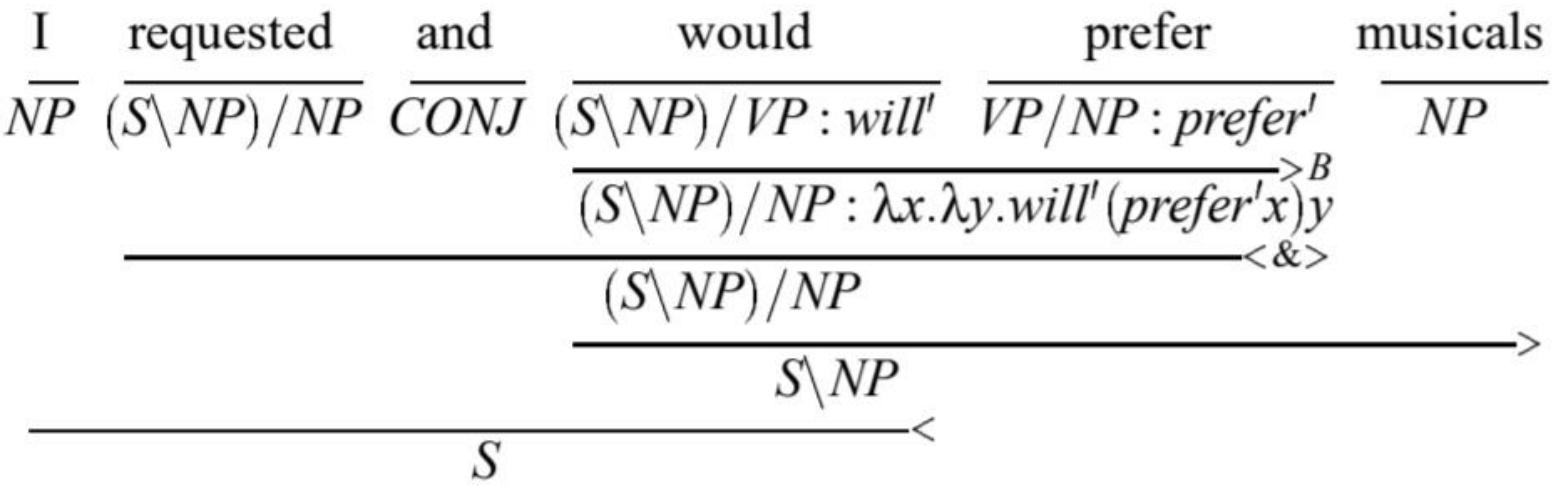
Forward App (>)	X/Y:f + Y:a → X:f(a)
Backward App (<)	Y:a + X\Y:f → X:f(a)
Coordination (<&>)	X:a + r + X:b → X:r(a,b)

CCG: adding semantics

John := NP

Mary := NP

likes := (S\NP)/NP



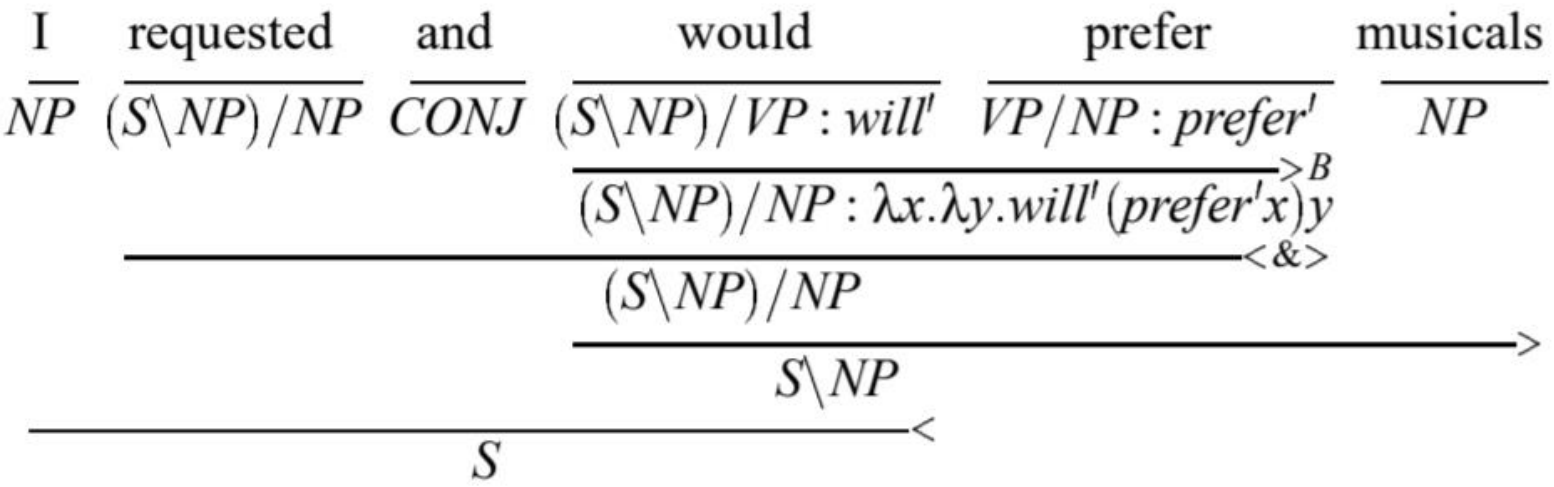
Forward App (>)	X/Y:f + Y:a → X:f(a)
Backward App (<)	Y:a + X\Y:f → X:f(a)
Coordination (<&>)	X:a + r + X:b → X:r(a,b)
Forward Comp (>B)	X/Y:f + Y/Z:g → X/Z: x -> f(g(x))

CCG: adding semantics

John := NP

Mary := NP

likes := (S\NP)/NP



Forward App (>)	X/Y:f + Y:a → X:f(a)
Backward App (<)	Y:a + X\Y:f → X:f(a)
Coordination (<&>)	X:a + r + X:b → X:r(a,b)
Forward Comp (>B)	X/Y:f + Y/Z:g → X/Z: x -> f(g(x))
Type raising	(see paper)

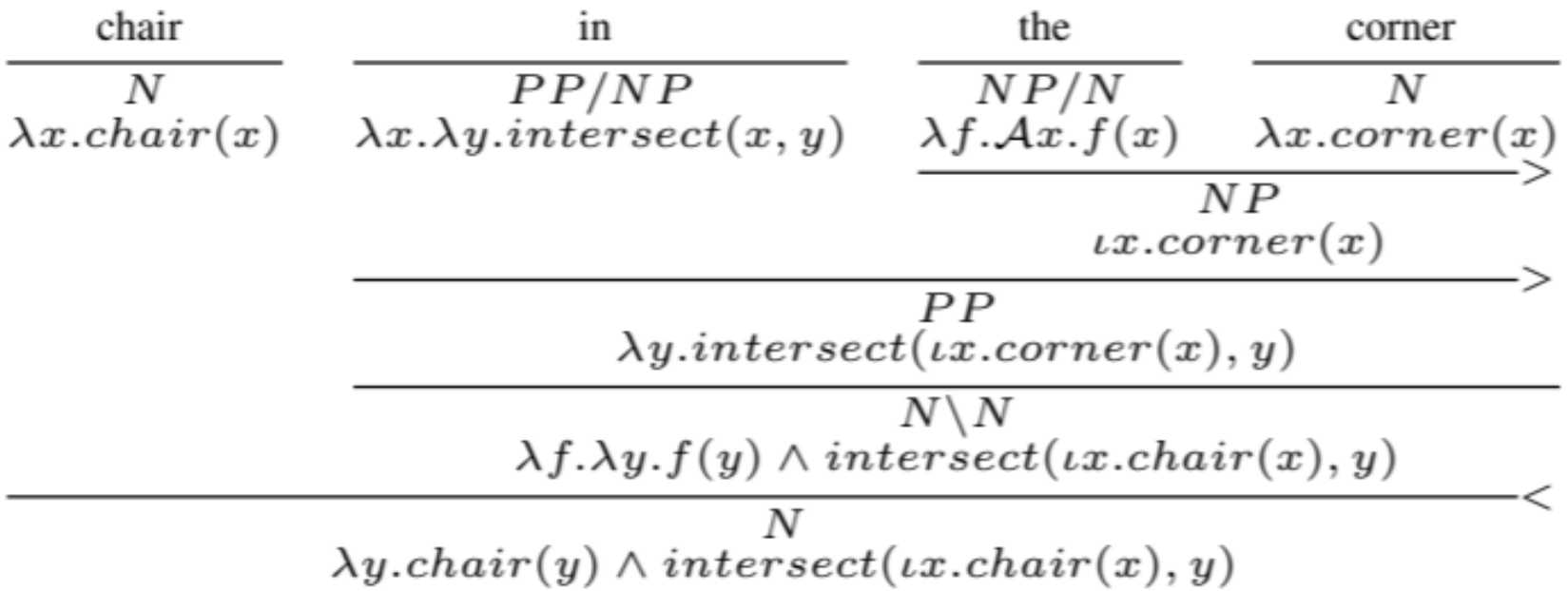
Traditional supervised training of CCG parsers

- Collect corpus of (sentence, CCG parse)
- Extract lexicon
- Learn weights on grammar rules
- Figure out ways to deal with “unparsable” sentences

Goal: parse from denotations alone

Imp.: move from the sofa to the chair

LF: $\lambda a.move(a) \wedge to(a, \iota x.chair(x)) \wedge from(a, \iota y.sofa(y))$



Following instructions

- Task: s = (x,y,o), a={Left,Right,Move,Null}
- Model:
  - CCG parser to map instruction *x* to meaning *z*
  - Logical form *z* mapped to sequence of actions *a* deterministically
- Learning:
  - Examples (x, s, V), V is validation function
    - VD = sequence of actions matches demonstration
    - VS = sequence of actions ends up in a correct state
  - No training data for *z*s
  - Both validation functions highly ambiguous
  - Small seed lexicon and all logical constants

CCG model

gories. For example, the lexical entry chair $\vdash N$: $\lambda x.chair(x)$ for the word "chair" in the parse in Figure 4 pairs it with a category that has syntactic type N and meaning $\lambda x.chair(x)$. Fig

2001; Collins, 2004; Taskar et al., 2004). Let x be a sentence, y be a CCG parse, and $\text{GEN}(x; \Lambda)$ be the set of all possible CCG parses for x given the lexicon Λ. Define $\phi(x, y) \in \mathbb{R}^d$ to be a d-dimensional *feature–vector* representation and $\theta \in \mathbb{R}^d$ to be a parameter vector. The optimal parse for sentence x is

$$y^*(x) = \arg \max_{y \in \text{GEN}(x;\Lambda)} \theta \cdot \phi(x, y)$$

and the final output logical form z is the λ-calculus expression at the root of $y^*(x)$. Section 7.2 de-

Challenges

- Where does the CCG lexicon come from?

  “GENLEX”

- How do you learn a good parser when you don’t have true parse trees?

  “Guess and check”

GENLEX

- Given: instruction *x*, state *s*, validation function *V*, current lexicon and params
- Overgenerate list of possible lexical entries, then prune

them. The full set is generated by taking the cross product of a set of templates, computed by factoring out all templates in the seed lexicon Λ_0, and all logical constants. For example, if Λ_0 has a lexical

item with the category $AP/NP : \lambda x.\lambda a.to(a, x)$ we would create entries $w \vdash AP/NP : \lambda x.\lambda a.p(a, x)$ for every phrase w in x and all constants p with the same type as to.[5]

- Prune (roughly) by:
  - Parsing *x* with the new proposed lexical item(s)
  - Resulting parse trees has score better (by a margin) than without
  - Parse tree actually uses new lexical item(s)

Parameter update

- Given: instruction *x*, state *s*, validation function *V*, current lexicon and params

- Generate k-best list of possible parses of *x* (using beam search)
- Let *G* be those that validate, *B* be those that do not
- Update parameters:
  - Toward *g*s that are scored lower than some *b*
  - Away from *b*s that are scored higher than some *g*

Step 1: (Lexical generation)

a. Set $\lambda_G \leftarrow GENLEX(x_i, s_i, \mathcal{V}_i; \Lambda, \theta), \lambda \leftarrow \Lambda \cup \lambda_G$

b. Let E be the k highest scoring executions from $GEN(x_i, s_i; \lambda)$ which use at most one entry from λ_G

c. Select lexical entries from the highest scoring valid parses: $\lambda_i \leftarrow \bigcup_{e \in MAXV_i(E;\theta)} LEX(e^y)$

d. Update lexicon: $\Lambda \leftarrow \Lambda \cup \lambda_i$

Step 2: (Update parameters)

a. Set $G_i \leftarrow MAXV_i(GEN(x_i, s_i; \Lambda); \theta)$ and $B_i \leftarrow \{e | e \in GEN(x_i, s_i; \Lambda) \wedge \mathcal{V}_i(e^{\vec{a}}) \neq 1\}$

b. Construct sets of margin violating good and bad parses:

$$R_i \leftarrow \{g | g \in G_i \wedge \exists b \in B_i \; s.t. \; \langle \theta, \Phi_i(g) - \Phi_i(b) \rangle < \gamma \Delta_i(g, b)\}$$

$$E_i \leftarrow \{b | b \in B_i \wedge \exists g \in G_i \; s.t. \; \langle \theta, \Phi_i(g) - \Phi_i(b) \rangle < \gamma \Delta_i(g, b)\}$$

c. Apply the additive update:

$$\theta \leftarrow \theta + \frac{1}{|R_i|} \sum_{r \in R_i} \Phi_i(r) - \frac{1}{|E_i|} \sum_{e \in E_i} \Phi_i(e)$$

Data & some results

	Single Sentence	Sequence
Final state validation		
Complete system	81.98 (2.33)	59.32 (6.66)
No implicit actions	77.7 (3.7)	38.46 (1.12)
No joint execution	73.27 (3.98)	31.51 (6.66)
Trace validation		
Complete system	82.74 (2.53)	58.95 (6.88)
No implicit actions	77.64 (3.46)	38.34 (6.23)
No joint execution	72.85 (4.73)	30.89 (6.08)

	Oracle	SAIL
# of instruction sequences	501	706
# of instruction sequences with implicit actions	431	
Total # of sentences	2679	3233
Avg. sentences per sequence	5.35	4.61
Avg. tokens per sentence	7.5	7.94
Vocabulary size	373	522

Table 1: Corpora statistics (lower-cased data).

Key ideas

- Only observing input and denotation, without parse tree
- k-best list generation plus validation for optimization

- CCG and inferring new lexical items

Closely related papers

Goldwasser, D., & Roth, D. (2011). Learning from Natural Instructions. In *Proceedings of the International Joint Conference on Artificial Intelligence.*

Liang, P., Jordan, M., & Klein, D. (2009). Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the Association for Computational Linguistics the International Joint Conference on Natural Language Processing.*

Zettlemoyer, L., & Collins, M. (2005). Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence.*

Natural language interfaces to databases

Ann Copestake [a1] and Karen Sparck Jones [a1]

DOI: https://doi.org/10.1017/S0269888900005476 Published online by Cambridge University Press: 07 July 2009

Abstract

This paper reviews the current state of the art in natural language access to databases. This has been a long-standing area of work in natural language processing. But though some commercial systems are now available, providing front ends has proved much harder than was expected, and the necessary limitations on front ends have to be recognized. The paper discusses the issues, both general to language and task-specific, involved in front end design, and the way these have been addressed, concentrating on the work of the last decade. The focus is on the central process of translating a natural language question into a database query, but other supporting functions are also covered. The points are illustrated by the use of a single example application. The paper concludes with an evaluation of the current state, indicating that future progress will depend on the one hand on general advances in natural language processing, and on the other on expanding the capabilities of traditional databases.

NLIDBS

```
/*********************************************
Following is a description of the geobase predicates:

state(name, abbreviation, capital, population, area, stat

city(state, state_abbreviation, name, population)

river(name, length, [states through which it flows])

border(state, state_abbreviation, [states that border it]

highlow(state, state_abbreviation, highest_point, highest

mountain(state, state_abbreviation, name, height)

road(number, [states it passes through])

lake(name, area, [states it is in])

*********************************************/
```

[give,me,the,cities,in,virginia,'.'],
answer(A,(city(A),loc(A,B),const(B,stateid(virginia))))

[what,are,the,high,points,of,states,surrounding,mississippi,?],
answer(A,(high_point(B,A),loc(A,B),state(B),next_to(B,C),
const(C,stateid(mississippi))))

[name,the,rivers,in,arkansas,'.'],
answer(A,(river(A),loc(A,B),const(B,stateid(arkansas))))

```
state('alabama','al','montgomery',3894.0e+3,51.7e+3,22,'birmingham','mobile','montgomery','huntsville').
state('alaska','ak','juneau',401.8e+3,591.0e+3,49,'anchorage','fairbanks','juneau','sitka').
state('arizona','az','phoenix',2718.0e+3,114.0e+3,48,'phoenix','tucson','mesa','tempe').
state('arkansas','ar','little rock',2286.0e+3,53.2e+3,25,'little rock','fort smith','north little rock','pine bluff').
state('california','ca','sacramento',23.67e+6,158.0e+3,31,'los angeles','san diego','san francisco','san jose').
state('colorado','co','denver',2889.0e+3,104.0e+3,38,'denver','colorado springs','aurora','lakewood').
state('connecticut','ct','hartford',3107.0e+3,5.02e+3,5,'bridgeport','hartford','new haven','waterbury').
state('delaware','de','dover',594.0e+3,2044,1,'wilmington','newark','dover','brookside').
state('district of columbia', 'dc','washington',638.0e+3,1100,0,'tenleytown','washington','georgetown','duval circle').
```

NLIDBS general algorithm

- Given only (sentence, answer) pairs

- For each sentence
  - Generate lots of parse trees
  - If any of them gives the correct answer on the database:
    - assume it’s correct and update

Variants

Compositional Semantic Parsing on Semi-Structured Tables

Panupong Pasupat
Computer Science Department
Stanford University
ppasupat@cs.stanford.edu

Percy Liang
Computer Science Department
Stanford University
pliang@cs.stanford.edu

- From geoquery → flight booking → freebase or Wikipedia tables

Year	City	Country	Nations
1896	Athens	Greece	14
1900	Paris	France	24
1904	St. Louis	USA	12
...	...	...	...
2004	Athens	Greece	201
2008	Beijing	China	204
2012	London	UK	204

x_1: *"Greece held its last Summer Olympics in which year?"*
y_1: {2004}

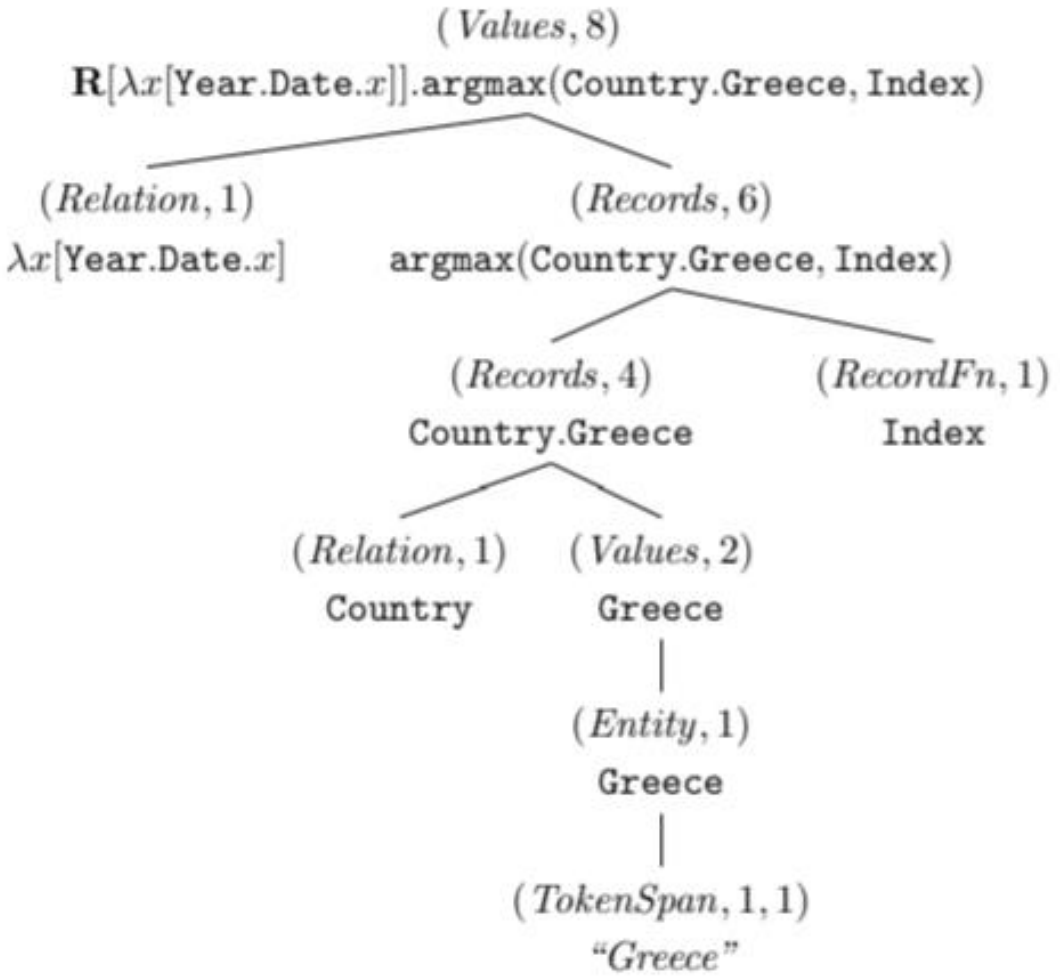
x_2: *"In which city's the first time with at least 20 nations?"*
y_2: {Paris}

x_3: *"Which years have the most participating countries?"*
y_3: {2008, 2012}

x_4: *"How many events were in Athens, Greece?"*
y_4: {2}

x_5: *"How many more participants were there in 1900 than in the first year?"*
y_5: {10}

Figure 1: Our task is to answer a highly compositional question from an HTML table. We learn a semantic parser from question-table-answer triples $\{(x_i, t_i, y_i)\}$.



Today

- Semantic parsing
- CCG grammars & lambda calculus
- Semantic parsing with full supervision
- Semantic parsing with denotations alone
  - Key idea: over-generate and check
- Other applications