Introduction to Semantic Parsing with CCG

Kilian Evang

Heinrich-Heine-Universität Düsseldorf

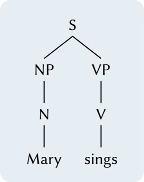
2018-04-24

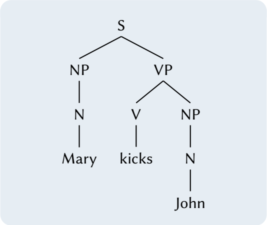
Table of contents

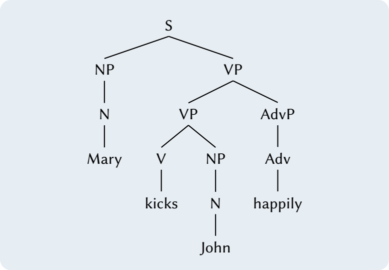
Outline

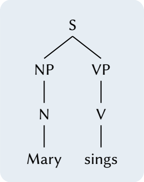
Outline

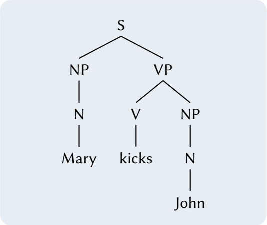
Plain old context-free grammars

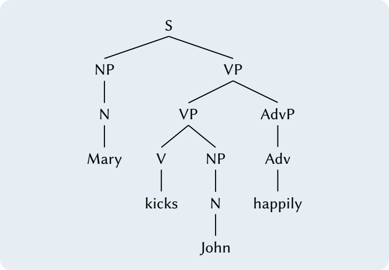












$S \longrightarrow NP \quad VP$

$NP \longrightarrow N$

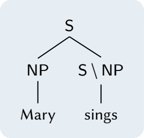
$AdvP \longrightarrow Adv$

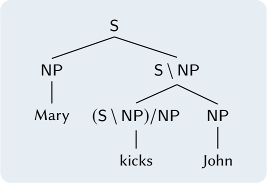
$VP \longrightarrow V$

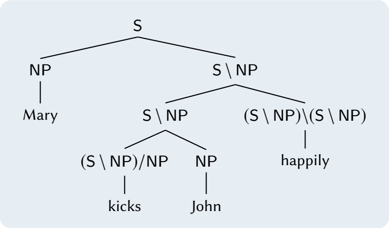
$VP \longrightarrow V \quad NP$

$VP \longrightarrow VP \quad AdvP$

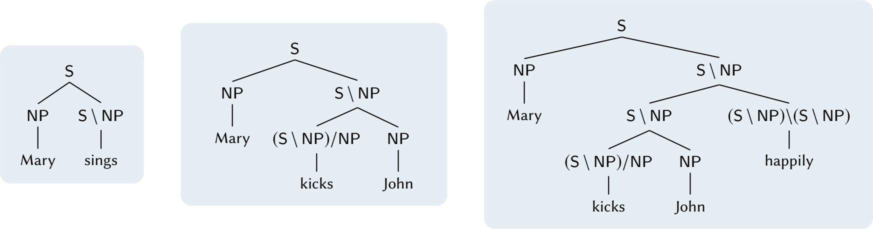
Categorial grammars







Categorial grammars



$$X \longrightarrow X/Y \quad Y$$
$$X \longrightarrow Y \quad X\backslash Y$$

Notation

$$\dfrac{\dfrac{\text{Mary}}{\mathsf{NP}} \quad \dfrac{\text{sings}}{\mathsf{S}\backslash\mathsf{NP}}}{\mathsf{S}}<^0$$

$$\dfrac{\dfrac{\text{Mary}}{\mathsf{NP}} \quad \dfrac{\dfrac{\text{kicks}}{(\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP}} \quad \dfrac{\text{John}}{\mathsf{NP}}}{\mathsf{S}\backslash\mathsf{NP}}>^0}{\mathsf{S}}<^0$$

$$\dfrac{\dfrac{\text{Mary}}{\mathsf{NP}} \quad \dfrac{\dfrac{\dfrac{\text{kicks}}{(\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP}} \quad \dfrac{\text{John}}{\mathsf{NP}}}{\mathsf{S}\backslash\mathsf{NP}}>^0 \quad \dfrac{\text{happily}}{(\mathsf{S}\backslash\mathsf{NP})\backslash(\mathsf{S}\backslash\mathsf{NP})}}{\mathsf{S}\backslash\mathsf{NP}}<^0}{\mathsf{S}}<^0$$

Notation

$$\dfrac{\dfrac{\text{Mary}}{\mathsf{NP}} \quad \dfrac{\text{sings}}{\mathsf{S}\backslash\mathsf{NP}}}{\mathsf{S}}{<^0}$$

$$\dfrac{\dfrac{\text{Mary}}{\mathsf{NP}} \quad \dfrac{\dfrac{\text{kicks}}{(\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP}} \quad \dfrac{\text{John}}{\mathsf{NP}}}{\mathsf{S}\backslash\mathsf{NP}}{>^0}}{\mathsf{S}}{<^0}$$

$$\dfrac{\dfrac{\text{Mary}}{\mathsf{NP}} \quad \dfrac{\dfrac{\dfrac{\text{kicks}}{(\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP}} \quad \dfrac{\text{John}}{\mathsf{NP}}}{\mathsf{S}\backslash\mathsf{NP}}{>^0} \quad \dfrac{\text{happily}}{(\mathsf{S}\backslash\mathsf{NP})\backslash(\mathsf{S}\backslash\mathsf{NP})}}{\mathsf{S}\backslash\mathsf{NP}}{<^0}}{\mathsf{S}}{<^0}$$

$$X/Y \quad Y \Rightarrow X \qquad (>^0) \quad \text{(forward application)}$$

$$Y \quad X\backslash Y \Rightarrow X \qquad (<^0) \quad \text{(backward application)}$$

Semantics

$$\frac{\text{Mary}}{\mathsf{NP}} \quad \frac{\text{kicks}}{(\mathsf{S} \setminus \mathsf{NP})/\mathsf{NP}} \quad \frac{\text{John}}{\mathsf{NP}}$$

Semantics

$$\frac{\text{Mary}}{\mathsf{NP} : \textit{mary}} \quad \frac{\text{kicks}}{(\mathsf{S} \backslash \mathsf{NP})/\mathsf{NP}} \quad \frac{\text{John}}{\mathsf{NP}}$$

Semantics

$$\dfrac{\text{Mary}}{\mathsf{NP} : \mathit{mary}} \quad \dfrac{\text{kicks}}{(\mathsf{S} \setminus \mathsf{NP})/\mathsf{NP} : \lambda x.\lambda y.\mathit{kicks}(y, x)} \quad \dfrac{\text{John}}{\mathsf{NP}}$$

Semantics

$$\dfrac{\text{Mary}}{\mathsf{NP} : \mathit{mary}} \quad \dfrac{\text{kicks}}{(\mathsf{S} \backslash \mathsf{NP})/\mathsf{NP} : \lambda x.\lambda y.\mathit{kicks}(y, x)} \quad \dfrac{\text{John}}{\mathsf{NP} : \mathit{john}}$$

Semantics

$$\dfrac{\text{Mary}}{\mathsf{NP} : \mathit{mary}} \quad \dfrac{\text{kicks}}{(\mathsf{S}\setminus\mathsf{NP})/\mathsf{NP} : \lambda x.\lambda y.\mathit{kicks}(y,x)} \quad \dfrac{\text{John}}{\mathsf{NP} : \mathit{john}}$$

$$X/Y : f \quad Y : g \Rightarrow X : f(g) \qquad (>^0) \quad \text{(forward application)}$$

$$Y : g \quad X\backslash Y : f \Rightarrow X : f(g) \qquad (<^0) \quad \text{(backward application)}$$

Semantics

$$\dfrac{\dfrac{\text{Mary}}{\mathsf{NP} : \textit{mary}} \quad \dfrac{\dfrac{\text{kicks}}{(\mathsf{S} \backslash \mathsf{NP})/\mathsf{NP} : \lambda x.\lambda y.\textit{kicks}(y, x)} \quad \dfrac{\text{John}}{\mathsf{NP} : \textit{john}}}{\mathsf{S} \backslash \mathsf{NP} : \lambda y.\textit{kicks}(y, \textit{john})} >^0}{}$$

$$X/Y : f \quad Y : g \Rightarrow X : f(g) \qquad (>^0) \quad \text{(forward application)}$$

$$Y : g \quad X \backslash Y : f \Rightarrow X : f(g) \qquad (<^0) \quad \text{(backward application)}$$

Semantics

$$
\dfrac{\dfrac{\text{Mary}}{\mathsf{NP} : \mathit{mary}} \quad \dfrac{\dfrac{\text{kicks}}{(\mathsf{S} \backslash \mathsf{NP})/\mathsf{NP} : \lambda x.\lambda y.\mathit{kicks}(y, x)} \quad \dfrac{\text{John}}{\mathsf{NP} : \mathit{john}}}{\mathsf{S} \backslash \mathsf{NP} : \lambda y.\mathit{kicks}(y, \mathit{john})} >^0}{\mathsf{S} : \mathit{kicks}(\mathit{mary}, \mathit{john})} <^0
$$

$$X/Y : f \quad Y : g \Rightarrow X : f(g) \qquad (>^0) \quad \text{(forward application)}$$

$$Y : g \quad X\backslash Y : f \Rightarrow X : f(g) \qquad (<^0) \quad \text{(backward application)}$$

Coordination of Functors

$$
\begin{array}{cccc}
\text{Mary} & \text{sings} & \text{and} & \text{dances} \\
\hline
\mathsf{NP} & \mathsf{S}\backslash\mathsf{NP} & ((\mathsf{S}\backslash\mathsf{NP})\backslash(\mathsf{S}\backslash\mathsf{NP}))/(\mathsf{S}\backslash\mathsf{NP}) & \mathsf{S}\backslash\mathsf{NP} \\
mary & \lambda x.sings(x) & \lambda f.\lambda g.\lambda x.g(x) \wedge f(x) & \lambda x.dances(x)
\end{array}
$$

$$
\frac{((\mathsf{S}\backslash\mathsf{NP})\backslash(\mathsf{S}\backslash\mathsf{NP}))/(\mathsf{S}\backslash\mathsf{NP}) \quad \mathsf{S}\backslash\mathsf{NP}}{(\mathsf{S}\backslash\mathsf{NP})\backslash(\mathsf{S}\backslash\mathsf{NP}) \\ \lambda g.\lambda x.g(x) \wedge dances(x)} >^{0}
$$

$$
\frac{\mathsf{S}\backslash\mathsf{NP} \quad (\mathsf{S}\backslash\mathsf{NP})\backslash(\mathsf{S}\backslash\mathsf{NP})}{\mathsf{S}\backslash\mathsf{NP} \\ \lambda x.sings(x) \wedge dances(x)} <^{0}
$$

$$
\frac{\mathsf{NP} \quad \mathsf{S}\backslash\mathsf{NP}}{\mathsf{S} \\ sings(mary) \wedge dances(mary)} <^{0}
$$

Coordination of Arguments

Mary	and	John	sing
NP	$\mathsf{((S/(S\backslash NP))\backslash(S/(S\backslash NP)))/(S/(S\backslash NP))}$	NP	$\mathsf{S\backslash NP}$
$mary$	$\lambda f.\lambda g.\lambda h.g(h) \wedge f(h)$	$john$	$\lambda x.sings(x)$

$$\frac{\mathsf{NP} : mary}{\mathsf{S/(S\backslash NP)} : \lambda f.f(mary)}\,T^{>} \qquad \frac{\mathsf{NP} : john}{\mathsf{S/(S\backslash NP)} : \lambda f.f(john)}\,T^{>}$$

$$\frac{\text{and} \quad \mathsf{S/(S\backslash NP)} : \lambda f.f(john)}{\mathsf{(S/(S\backslash NP))\backslash(S/(S\backslash NP))} : \lambda g.\lambda h.g(h) \wedge h(john)}\,>^{0}$$

$$\frac{\mathsf{S/(S\backslash NP)} : \lambda f.f(mary) \quad \mathsf{(S/(S\backslash NP))\backslash(S/(S\backslash NP))} : \lambda g.\lambda h.g(h) \wedge h(john)}{\mathsf{S/(S\backslash NP)} : \lambda h.h(mary) \wedge h(john)}\,<^{0}$$

$$\frac{\mathsf{S/(S\backslash NP)} : \lambda h.h(mary) \wedge h(john) \quad \mathsf{S\backslash NP} : \lambda x.sings(x)}{\mathsf{S} : sing(mary) \wedge sing(john)}\,>^{0}$$

$$Y : g \Rightarrow T/(T\backslash Y) : \lambda f.f(g) \qquad (T^{>}) \qquad \text{(forward type raising)}$$

$$Y : g \Rightarrow T\backslash(T/Y) : \lambda f.f(g) \qquad (T^{<}) \qquad \text{(backward type raising)}$$

How CGs differ from CFGs

1. informative labels (*categories*)
2. general rules (*combinators*)
3. transparent syntax-semantics interface
   (syntactic dependency $\cong$ function application)

Outline

1. Introduction to CCG
   - Categorial Grammar (CG)
   - Combinatory Categorial Grammar (CCG)
2. Zettlemoyer and Collins (2005)
   - Candidate lexicon generation
   - Features, model, learning
   - Results
3. References

Combinatory Categorial Grammar

- type of CG developed by Mark Steedman [1]
- additional combinators
- elegant treatment of “incomplete” constituents
  - object relative clauses
  - non-canonical coordination
  - ...

Object relative clauses

the	car	that	Sue	bought
NP/N	N	(N \ N)/(S/NP)	NP	(S \ NP)/NP

- Sue: $\mathsf{NP} \Rightarrow \mathsf{S/(S \backslash NP)}$ $(T^{>})$
- Sue bought: $\mathsf{S/(S \backslash NP)} \quad \mathsf{(S \backslash NP)/NP} \Rightarrow \mathsf{S/NP}$ $(>^1)$
- that Sue bought: $\mathsf{(N \backslash N)/(S/NP)} \quad \mathsf{S/NP} \Rightarrow \mathsf{N \backslash N}$ $(>^0)$
- car that Sue bought: $\mathsf{N} \quad \mathsf{N \backslash N} \Rightarrow \mathsf{N}$ $(<^0)$
- the car that Sue bought: $\mathsf{NP/N} \quad \mathsf{N} \Rightarrow \mathsf{NP}$ $(>^0)$

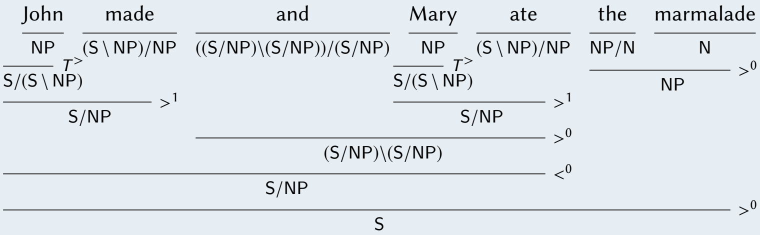
$$X/Y \quad Y/Z \Rightarrow X/Z \quad (>^1) \quad \text{(forward harmonic composition)}$$

$$Y\backslash Z \quad X\backslash Y \Rightarrow X\backslash Z \quad (<^1) \quad \text{(backward harmonic composition)}$$

$$X/Y \quad Y\backslash Z \Rightarrow X\backslash Z \quad (>^1_\times) \quad \text{(forward crossing composition)}$$

$$Y/Z \quad X\backslash Y \Rightarrow X/Z \quad (<^1_\times) \quad \text{(backward crossing composition)}$$

Non-canonical coordination



What makes CCG suitable for semantic parsing?

- few labels
- few rules
- flexible constituency: syntax adapts to semantics
- clear syntax-semantics interface

→ there is not much to learn about syntax, can focus on semantics

Outline

Zettlemoyer and Collins (2005)

- Luke S. Zettlemoyer and Michael Collins (2005): Learning to Map Sentences to Logical Form: Structured Classification with Probabilistic Categorial Grammars [3]
- First paper to apply CCG to semantic parser learning, many followed

Characteristics of ZC05's parser

NLU representation

list of words

MRL

GeoQuery and Jobs (in lambda format)

data

sentences (NLUs) + logical forms (MRs)

evaluation metric

exact match (modulo renaming variables)

lexicon representation

CCG categories with lambda terms

Characteristics of ZC05's parser (cont.)

candidate lexicon generation

using templates (GENLEX)

parsing algorithm

CKY-style (chart)

features

only lexical features

model

log-linear

training algorithm

alternates between GENLEX, pruning, and parameter estimation

Characteristics of ZC05's parser (cont.)

experimental setup

Geo880, Jobs640 datasets, standard splits

Outline

Example parse

$$
\begin{array}{cccc}
\text{What} & \text{states} & \text{border} & \text{Texas} \\
\hline
\mathsf{(S/(S\backslash NP))/N} & \mathsf{N} & \mathsf{(S\backslash NP)/NP} & \mathsf{NP} \\
\lambda f.\lambda g.\lambda x.f(x) \wedge g(x) & \lambda x.\mathit{state}(x) & \lambda x.\lambda y.\mathit{borders}(y, x) & \mathit{texas}
\end{array}
$$

$$
\underline{\lambda f.\lambda g.\lambda x.f(x) \wedge g(x) \quad \lambda x.\mathit{state}(x)}\ >^{0} \qquad \underline{\lambda x.\lambda y.\mathit{borders}(y, x) \quad \mathit{texas}}\ >^{0}
$$

$$
\mathsf{S/(S\backslash NP)} \qquad\qquad \mathsf{S\backslash NP}
$$

$$
\lambda g.\lambda x.\mathit{state}(x) \wedge g(x) \qquad\qquad \lambda y.\mathit{borders}(y, \mathit{texas})
$$

$$
\overline{\hspace{20em}}\ >^{0}
$$

$$
\mathsf{S}
$$

$$
\lambda x.\mathit{state}(x) \wedge \mathit{borders}(x, \mathit{texas})
$$

Learning lexicon entries

Input (a training example)

(1) a. What states border Texas

b. $\lambda x.state(x) \wedge borders(x, texas)$

Desired output

$$\text{What} := (\mathsf{S}/(\mathsf{S} \backslash \mathsf{NP}))/\mathsf{N} : \lambda f.\lambda g.\lambda x.f(x) \wedge g(x)$$

$$\text{states} := \mathsf{N} : \lambda x.state(x)$$

$$\text{border} := (\mathsf{S} \backslash \mathsf{NP})/\mathsf{NP} : \lambda x.\lambda y.borders(y, x)$$

$$\text{Texas} := \mathsf{NP} : texas$$

Hard-coded cross-domain lexicon entries

$$\text{What} := (\mathsf{S}/(\mathsf{S}\setminus\mathsf{NP}))/\mathsf{N} : \lambda f.\lambda g.\lambda x.f(x) \wedge g(x)$$

Lexical entries for names in the domain database

$$\text{Texas} := \mathsf{NP} : \textit{texas}$$
$$\text{Michigan} := \mathsf{NP} : \textit{michigan}$$
$$\text{Seattle} := \mathsf{NP} : \textit{seattle}$$
$$\ldots$$

Candidate lexicon generation with GENLEX

Input (a training example)

(2) a. Utah borders Idaho
    b. *borders*(*utah*, *idaho*)

GENLEX output

$$\text{Utah} := \mathrm{NP} : utah$$

$$\text{Idaho} := \mathrm{NP} : idaho$$

$$\text{borders} := (\mathrm{S} \setminus \mathrm{NP})/\mathrm{NP} : \lambda x.\lambda y.borders(y, x)$$

$$\text{borders} := \mathrm{NP} : idaho$$

$$\text{borders Utah} := (\mathrm{S} \setminus \mathrm{NP})/\mathrm{NP} : \lambda x.\lambda y.borders(y, x)$$

→ spurious items need to be pruned later

GENLEX

$\text{GENLEX}(S, L) = \{x := y | x \in W(S), y \in C(L)\}$
where

- S is a sentence
- L is its logical form
- $W(S)$ is the set of all subsequences of words in S
- C maps L to a set of categories through rules (see next slide)

GENLEX (cont.)

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

Outline

Probabilistic CCG parsing

- mapping a sentence S to (T, L) where T is a parse tree (derivation) and L is a logical form
- even with a perfect lexicon, one S can have multiple (T, L) pairs (lexical ambiguity, attachment ambiguity, spurious ambiguity...)
- solution: define probability distribution $P(L, T|S)$

Probabilistic CCG parsing (cont.)

- here: features = lexicon entries used in a parse
- feature function $\overline{f}$ maps parses (L, T, S) to a feature vector that indicates for each lexicon entry how often it is used in T
- assume a parameter vector $\overline{\theta}$ that scores individual lexical features such that $P(L, T|S; \overline{\theta}) = \frac{e^{\overline{f}(L,T,S)\cdot\overline{\theta}}}{\sum_{(L,T)} e^{\overline{f}(L,T,S)\cdot\overline{\theta}}}$ (log-linear model)
- Given S, return the parse (L, T) with the highest $P(L, T|S; \theta)$
- problem: how to find a good lexicon and a good $\overline{\theta}$?

Bird's-eye view of the learning algorithm

Input: initial lexicon Λ_0, training examples
$E = \{(S_i, L_i) : i = 1 \ldots n\}$

Bird's-eye view of the learning algorithm

Input: initial lexicon Λ_0, training examples
$E = \{(S_i, L_i) : i = 1 \ldots n\}$
Initialization: $\overline{\theta}^0$:= vector with 0.1 for lexicon entries in Λ_0, 0.01 for all other lexicon entries

Bird's-eye view of the learning algorithm

Input: initial lexicon Λ_0, training examples
$E = \{(S_i, L_i) : i = 1 \ldots n\}$
Initialization: $\overline{\theta}^0$:= vector with 0.1 for lexicon entries in Λ_0, 0.01 for all other lexicon entries
for $t = 1 \ldots T$ **do** ▹ epochs

Bird's-eye view of the learning algorithm

Input: initial lexicon Λ_0, training examples
$E = \{(S_i, L_i) : i = 1 \dots n\}$
Initialization: $\overline{\theta}^0$:= vector with 0.1 for lexicon entries in Λ_0, 0.01 for all other lexicon entries

for $t = 1 \dots T$ **do** ▹ epochs
    for $i = 1 \dots n$ **do** ▹ lexical generation

Bird's-eye view of the learning algorithm

Input: initial lexicon Λ_0, training examples
$E = \{(S_i, L_i) : i = 1 \dots n\}$
Initialization: $\overline{\theta}^0$:= vector with 0.1 for lexicon entries in Λ_0, 0.01 for all other lexicon entries

for $t = 1 \dots T$ **do** ▹ epochs
    for $i = 1 \dots n$ **do** ▹ lexical generation
        $\lambda := \Lambda_0 \cup \mathsf{GENLEX}(S_i, L_i)$

Bird's-eye view of the learning algorithm

Input: initial lexicon Λ_0, training examples
$E = \{(S_i, L_i) : i = 1 \ldots n\}$
Initialization: $\overline{\theta}^0$:= vector with 0.1 for lexicon entries in Λ_0, 0.01
for all other lexicon entries
for $t = 1 \ldots T$ **do** ▷ epochs
    for $i = 1 \ldots n$ **do** ▷ lexical generation
        $\lambda := \Lambda_0 \cup \mathsf{GENLEX}(S_i, L_i)$
        $\pi := \mathsf{PARSE}(S_i, L_i, \lambda, \overline{\theta}^{t-1})$

Bird's-eye view of the learning algorithm

Input: initial lexicon Λ_0, training examples
$E = \{(S_i, L_i) : i = 1 \dots n\}$
Initialization: $\overline{\theta}^0$:= vector with 0.1 for lexicon entries in Λ_0, 0.01 for all other lexicon entries

for $t = 1 \dots T$ **do** ▹ epochs
    for $i = 1 \dots n$ **do** ▹ lexical generation
        $\lambda := \Lambda_0 \cup \mathsf{GENLEX}(S_i, L_i)$
        $\pi := \mathsf{PARSE}(S_i, L_i, \lambda, \overline{\theta}^{t-1})$
        λ_i := the set of lexicon entries in π

Bird's-eye view of the learning algorithm

Input: initial lexicon Λ_0, training examples
$E = \{(S_i, L_i) : i = 1 \ldots n\}$
Initialization: $\overline{\theta}^0$:= vector with 0.1 for lexicon entries in Λ_0, 0.01 for all other lexicon entries

for $t = 1 \ldots T$ **do** ▹ epochs
    for $i = 1 \ldots n$ **do** ▹ lexical generation
        $\lambda := \Lambda_0 \cup \mathsf{GENLEX}(S_i, L_i)$
        $\pi := \mathsf{PARSE}(S_i, L_i, \lambda, \overline{\theta}^{t-1})$
        λ_i := the set of lexicon entries in π
    end for

Bird's-eye view of the learning algorithm

Input: initial lexicon Λ_0, training examples $E = \{(S_i, L_i) : i = 1 \ldots n\}$
Initialization: $\overline{\theta}^0$:= vector with 0.1 for lexicon entries in Λ_0, 0.01 for all other lexicon entries

for $t = 1 \ldots T$ **do** ▹ epochs
    for $i = 1 \ldots n$ **do** ▹ lexical generation
        $\lambda := \Lambda_0 \cup \text{GENLEX}(S_i, L_i)$
        $\pi := \text{PARSE}(S_i, L_i, \lambda, \overline{\theta}^{t-1})$
        λ_i := the set of lexicon entries in π
    end for
    $\Lambda_t := \Lambda_0 \cup \bigcup_{i=1}^{n} \lambda_i$ ▹ update the lexicon

Bird's-eye view of the learning algorithm

Input: initial lexicon Λ_0, training examples $E = \{(S_i, L_i) : i = 1 \ldots n\}$

Initialization: $\overline{\theta}^0$:= vector with 0.1 for lexicon entries in Λ_0, 0.01 for all other lexicon entries

for $t = 1 \ldots T$ **do** ▹ epochs

  for $i = 1 \ldots n$ **do** ▹ lexical generation

    $\lambda := \Lambda_0 \cup \mathrm{GENLEX}(S_i, L_i)$

    $\pi := \mathrm{PARSE}(S_i, L_i, \lambda, \overline{\theta}^{t-1})$

    λ_i := the set of lexicon entries in π

  end for

  $\Lambda_t := \Lambda_0 \cup \bigcup_{i=1}^{n} \lambda_i$ ▹ update the lexicon

  $\overline{\theta}^t := \mathrm{ESTIMATE}(\Lambda_t, E, \overline{\theta}^{t-1})$ ▹ parameter estimation

Bird's-eye view of the learning algorithm

Input: initial lexicon Λ_0, training examples $E = \{(S_i, L_i) : i = 1 \ldots n\}$

Initialization: $\overline{\theta}^0$:= vector with 0.1 for lexicon entries in Λ_0, 0.01 for all other lexicon entries

for $t = 1 \ldots T$ **do** ▹ epochs

  for $i = 1 \ldots n$ **do** ▹ lexical generation

    $\lambda := \Lambda_0 \cup \mathsf{GENLEX}(S_i, L_i)$

    $\pi := \mathsf{PARSE}(S_i, L_i, \lambda, \overline{\theta}^{t-1})$

    λ_i := the set of lexicon entries in π

  end for

  $\Lambda_t := \Lambda_0 \cup \bigcup_{i=1}^{n} \lambda_i$ ▹ update the lexicon

  $\overline{\theta}^t := \mathsf{ESTIMATE}(\Lambda_t, E, \overline{\theta}^{t-1})$ ▹ parameter estimation

end for

Bird's-eye view of the learning algorithm

Input: initial lexicon Λ_0, training examples
$E = \{(S_i, L_i) : i = 1 \ldots n\}$
Initialization: $\overline{\theta}^0$:= vector with 0.1 for lexicon entries in Λ_0, 0.01 for all other lexicon entries

for $t = 1 \ldots T$ **do** ▹ epochs
  for $i = 1 \ldots n$ **do** ▹ lexical generation
    $\lambda := \Lambda_0 \cup \mathsf{GENLEX}(S_i, L_i)$
    $\pi := \mathsf{PARSE}(S_i, L_i, \lambda, \overline{\theta}^{t-1})$
    λ_i := the set of lexicon entries in π
  end for
  $\Lambda_t := \Lambda_0 \cup \bigcup_{i=1}^{n} \lambda_i$ ▹ update the lexicon
  $\overline{\theta}^t := \mathsf{ESTIMATE}(\Lambda_t, E, \overline{\theta}^{t-1})$ ▹ parameter estimation
end for

Output: lexicon Λ_T, parameters $\overline{\theta}^T$

Outline

Results

	Geo880		Jobs640	
	Precision	Recall	Precision	Recall
Zettlemoyer & Collins [3]	96.25	79.29	97.36	79.29
Tang & Mooney [2]	89.92	79.40	93.25	79.84

Some learned lexicon entries

$$
\begin{aligned}
\text{states} &:= \mathsf{N} : \lambda x.\mathit{state}(x) \\
\text{major} &:= \mathsf{N/N} : \lambda f.\lambda x.\mathit{major}(x) \wedge f(x) \\
\text{population} &:= \mathsf{N} : \lambda x.\mathit{population}(x) \\
\text{cities} &:= \mathsf{N} : \lambda x.\mathit{river}(x) \\
\text{rivers} &:= \mathsf{N} : \lambda x.\mathit{river}(x) \\
\text{run through} &:= \mathsf{(S \setminus NP)/NP} : \lambda x.\lambda y.\mathit{traverse}(y, x) \\
\text{the largest} &:= \mathsf{NP/N} : \lambda f.\arg\max(f, \lambda x.\mathit{size}(x)) \\
\text{river} &:= \mathsf{N} : \lambda x.\mathit{river}(x) \\
\text{the highest} &:= \mathsf{NP/N} : \lambda f.\arg\max(f, \lambda x.\mathit{elev}(x)) \\
\text{the longest} &:= \mathsf{NP/N} : \lambda f.\arg\max(f, \lambda x.\mathit{len}(x))
\end{aligned}
$$

Outline

[1] Steedman, Mark. 2001. *The syntactic process*. The MIT Press.
[2] Tang, Lappoon R. & Raymond J. Mooney. 2001. Using multiple clause constructors in inductive logic programming for semantic parsing. In *Proceedings of the 12th european conference on machine learning*, 466–477.
[3] Zettlemoyer, Luke & Michael Collins. 2005. Learning to map sentences to logical form: structured classification with probabilistic categorial grammars. In *Proceedings of the twenty-first conference annual conference on uncertainty in artificial intelligence (UAI-05)*, 658–666. AUAI Press.