Learning Compositional Semantics

CS224U: Natural Language Understanding
Feb. 9, 2012

Percy Liang
Google/Stanford
Review

Last time: Mapping sentences to logical forms (FOL or lambda calculus)

*Alaska borders no states.*

\[ \neg \exists x. \text{state}(x) \land \text{border} (\text{AK}, x) \]
Review

Last time: Mapping sentences to logical forms (FOL or lambda calculus)

Alaska borders no states.

\[ \neg \exists x. \text{state}(x) \land \text{border}(\text{AK}, x) \]

We assumed the following were given:

Lexicon:

- \textit{no} \Rightarrow DT: \lambda P.\lambda Q. \neg \exists x. P(x) \land Q(x)
- \textit{states} \Rightarrow N: \lambda x. \text{state}(x)
Review

Last time: Mapping sentences to logical forms (FOL or lambda calculus)

Alaska borders no states.

\[ \neg \exists x. \text{state}(x) \land \text{border}(\text{AK}, x) \]

We assumed the following were given:

<table>
<thead>
<tr>
<th>Lexicon:</th>
<th>no</th>
<th>DT : \lambda P. \lambda Q. \neg \exists x. P(x) \land Q(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>states</td>
<td>N : \lambda x. \text{state}(x)</td>
</tr>
</tbody>
</table>

| Grammar:    | DT : f           | N : a \Rightarrow NP : f(a)                                |
Review

Last time: Mapping sentences to logical forms (FOL or lambda calculus)

*Alaska borders no states.*

\[ \neg \exists x. \text{state}(x) \land \text{border}(\text{AK}, x) \]

We assumed the following were given:

<table>
<thead>
<tr>
<th>Lexicon:</th>
<th>no</th>
<th>( \Rightarrow ) DT: ( \lambda P. \lambda Q. \neg \exists x. P(x) \land Q(x) )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>states</td>
<td>( \Rightarrow ) N: ( \lambda x. \text{state}(x) )</td>
</tr>
</tbody>
</table>

Grammar:

| DT: f         | N: a             | \( \Rightarrow \) NP: \( f(a) \)                                                                 |

Questions:

But where do they come from?
What if a sentence generates multiple logical forms?
What if a sentences is slightly ungrammatical?
Outline

Today: building real semantic parsers!

sentence → Semantic Parser → logical form
Outline

Today: building real semantic parsers!

Strategy: break up complex mapping into two parts
Outline

Today: building real semantic parsers!

Strategy: break up complex mapping into two parts

Representation (Lexicon/Grammar):

- Should be simple, require minimal human effort
- Generates set of candidate logical forms
  
  Allow overgeneration: \( state \Rightarrow N : \lambda x.\text{river}(x) \)
Outline

Today: building real semantic parsers!

Strategy: break up complex mapping into two parts

Representation (Lexicon/Grammar):

- Should be simple, require minimal human effort
- Generates set of candidate logical forms
  Allow overgeneration: \( state \Rightarrow N : \lambda x.\text{river}(x) \)

Learning:

- Score/rank candidates based on features
- Optimize feature weights discriminatively to minimize training error
Outline

Representation

Learning

Experiments
Semantic Formalisms

sentence → Semantic Parser → logical form
Semantic Formalisms

sentence → Semantic Parser → logical form → Interpretation → denotation
We are free to choose the semantic formalism:

- What kind of logical forms? FOL? lambda calculus?
- What constitutes the lexicon and grammar?
Semantic Formalisms

We are free to choose the semantic formalism:

- What kind of logical forms? FOL? lambda calculus?
- What constitutes the lexicon and grammar?

Desiderata:

Model-theoretic: logical forms must have formal interpretation
(mapping from world to true/false)
Semantic Formalisms

We are free to choose the semantic formalism:

- What kind of logical forms? FOL? lambda calculus?
- What constitutes the lexicon and grammar?

Desiderata:

Model-theoretic: logical forms must have formal interpretation (mapping from world to true/false)

Compositional: meaning (logical form) of phrase computed from combining meaning of sub-phrases
We are free to choose the semantic formalism:

- What kind of logical forms? FOL? lambda calculus?
- What constitutes the lexicon and grammar?

Desiderata:

Model-theoretic: logical forms must have formal interpretation (mapping from world to true/false)
Compositional: meaning (logical form) of phrase computed from combining meaning of sub-phrases

Semantic Formalisms:

- Combinatory Categorial Grammar (CCG)
- Dependency-Based Compositional Semantics (DCS)
Combinatory Categorial Grammar (CCG)

Lexicalized formalism: simple grammar rules, heavy lexicon
Combinatory Categorial Grammar (CCG)

Lexicalized formalism: simple grammar rules, heavy lexicon

Categories (analogous to types in programming languages):

\[
\text{NP} \quad \text{VP} \quad \Rightarrow \quad \text{S}
\]
Combinatory Categorial Grammar (CCG)

Lexicalized formalism: simple grammar rules, heavy lexicon

Categories (analogous to types in programming languages):

\[ \text{NP} \text{ VP} \Rightarrow \text{S} \quad \text{NP} \quad \text{S}\backslash\text{NP} \Rightarrow \text{S} \]
Combinatory Categorial Grammar (CCG)

Lexicalized formalism: simple grammar rules, heavy lexicon

Categories (analogous to types in programming languages):

\[
\begin{align*}
\text{NP} & \quad \text{VP} \quad \Rightarrow \quad \text{S} \\
\text{V} & \quad \text{NP} \quad \Rightarrow \quad \text{VP} \\
\text{NP} & \quad \text{S}\backslash\text{NP} \quad \Rightarrow \quad \text{S}
\end{align*}
\]
Combinatory Categorial Grammar (CCG)

Lexicalized formalism: simple grammar rules, heavy lexicon

Categories (analogous to types in programming languages):

\[
\begin{align*}
\text{NP} & \quad \text{VP} \quad \Rightarrow \quad \text{S} \\
\text{V} & \quad \text{NP} \quad \Rightarrow \quad \text{VP} \\
\text{NP} & \quad \text{S} \backslash \text{NP} \quad \Rightarrow \quad \text{S} \\
\text{(S} \backslash \text{NP}) \backslash \text{NP} & \quad \text{NP} \quad \Rightarrow \quad \text{S} \backslash \text{NP}
\end{align*}
\]
Combinatory Categorial Grammar (CCG)

Lexicalized formalism: simple grammar rules, heavy lexicon

Categories (analogous to types in programming languages):

\[
\begin{align*}
\text{NP} & \rightarrow \text{VP} & \text{NP} & \rightarrow \text{S} \\
\text{V} & \rightarrow \text{NP} & \text{(S\NP)} / \text{NP} & \rightarrow \text{S\NP}
\end{align*}
\]

In general:

Base categories: S, NP, N

Derived categories: if X, Y are categories, then X/Y and X\Y are too
Combinatory Categorial Grammar (CCG)

Lexicon:

Alice \( \text{NP} : \text{alice} \)
Bob \( \text{NP} : \text{bob} \)
saw \( (S\backslash \text{NP})/\text{NP} : \lambda y.\lambda x.\text{saw}(x,y) \)
Combinatory Categorial Grammar (CCG)

Lexicon:

- Alice np : alice
- Bob np : bob
- saw (S\NP)/NP : λy.λx.saw(x,y)

Grammar (template):

Forward application (>) Y/X : f X : a ⇒ Y : f(a)

Backward application (<) X : a Y \X : f ⇒ Y : f(a)
Combinatory Categorial Grammar (CCG)

Lexicon:

- **Alice**: NP : alice
- **Bob**: NP : bob
- **saw**: (S\NP)/NP : λy.λx.saw(x,y)

Grammar (template):

- Forward application (>) : Y/X : f X : a \(\Rightarrow\) Y : f(a)
- Backward application (<) : X : a Y\X : f \(\Rightarrow\) Y : f(a)

Derivation:

```
S : saw(alice, bob)

NP : alice

Alice

(S\NP)/NP : λy.λx.saw(x, y)

saw

Bob

NP : bob
```
Combinatory Categorial Grammar (CCG)

More grammar rule templates:

Forward composition \((B \succ)\) \[ Y/X : f \quad X/Z : a \quad \Rightarrow \quad Y/Z : \lambda z. f(a(z)) \]
Combinatory Categorial Grammar (CCG)

More grammar rule templates:

Forward composition ($B >$) \[ Y/X : f \quad X/Z : a \quad \Rightarrow \quad Y/Z : \lambda z.f(a(z)) \]

Type raising ($T >$) \[ X : a \quad \Rightarrow \quad Y/(Y/X) : \lambda f.f(a) \]
Combinatory Categorial Grammar (CCG)

More grammar rule templates:

Forward composition (B >) \[ \frac{Y/X : f \quad X/Z : a}{Y/Z : \lambda z. f(a(z))} \]

Type raising (T >) \[ \frac{X : a}{Y/(Y\backslash X) : \lambda f. f(a)} \]

\[ S : \text{saw(alice, bob)} \]
\[ S/\text{NP} : \lambda y. \text{saw(alice, y)} \]
\[ S/(S\backslash \text{NP}) : \lambda f. f(\text{alice}) \]
\[ (S\backslash \text{NP})/\text{NP} : \lambda y. \lambda x. \text{saw}(x, y) \]

\[ \text{Alice} \]
\[ \text{Bob} \]
Combinatory Categorial Grammar (CCG)

More grammar rule templates:

Forward composition \((B >)\)

\[ Y/X : f \quad X/Z : a \quad \Rightarrow \quad Y/Z : \lambda z.f(a(z)) \]

Type raising \((T >)\)

\[ X : a \quad \Rightarrow \quad Y/(Y \backslash X) : \lambda f.f(a) \]

Composition creates non-traditional bracketing useful for right-node raising:

\[ S : saw(alice, bob) \wedge heard(carol, bob) \]

[[Alice saw] and [Carol heard]] Bob.
CCG Meets Real Data

Non-contentful words:

$$\lambda x. \text{flight}(x) \land \text{to}(x, \text{boston})$$

*Show me flights to Boston*
CCG Meets Real Data

Non-contentful words:

\[ \lambda x. \text{flight}(x) \land \text{to}(x, \text{boston}) \]

*Show me flights to Boston*

Solution: identity functions: *show me* \[\Rightarrow \] \[\frac{N}{N} : \lambda f. f\]
CCG Meets Real Data

Non-contentful words:

\[ \lambda x. \text{flight}(x) \land \text{to}(x, \text{boston}) \]

*Show me flights to Boston*

Solution: identity functions: *show me* \[\Rightarrow\] \[N/N : \lambda f.f\]

Missing content:

\[ \lambda x. \text{flight}(x) \land \text{to}(x, \text{boston}) \]

*Boston flights*
CCG Meets Real Data

Non-contentful words:
\[ \lambda x. \text{flight}(x) \land \text{to}(x, \text{boston}) \]

*Show me flights to Boston*

Solution: identity functions: *show me* \[ \Rightarrow \quad \frac{N}{N} : \lambda f. f \]

Missing content:
\[ \lambda x. \text{flight}(x) \land \text{to}(x, \text{boston}) \]

*Boston flights*

Solution: type-raising: \[ \frac{NP}{x} \quad \Rightarrow \quad \frac{NP}{N} : \lambda f. \lambda a. f(a) \land \text{to}(a, x) \]
CCG Meets Real Data

Non-contentful words:

\[ \lambda x. \text{flight}(x) \land \text{to}(x, \text{boston}) \]

*Show me flights to Boston*

Solution: identity functions: 

\[ \Rightarrow \quad N/N : \lambda f. f \]

Missing content:

\[ \lambda x. \text{flight}(x) \land \text{to}(x, \text{boston}) \]

*Boston flights*

Solution: type-raising: 

\[ \Rightarrow \quad \text{NP}/N : \lambda f. \lambda a. f(a) \land \text{to}(a, x) \]

Non-standard ordering:

\[ \lambda x. \text{flight}(x) \land \text{oneway}(x) \]

*flights one-way*
CCG Meets Real Data

Non-contentful words:
\[ \lambda x. \text{flight}(x) \land \text{to}(x, \text{boston}) \]
*Show me flights to Boston*

Solution: identity functions: *show me* \[ \Rightarrow \quad N/N : \lambda f.f \]

Missing content:
\[ \lambda x. \text{flight}(x) \land \text{to}(x, \text{boston}) \]
*Boston flights*

Solution: type-raising: \( \text{NP} : x \) \[ \Rightarrow \quad \text{NP}/N : \lambda a. \lambda f. f(a) \land \text{to}(a, x) \]

Non-standard ordering:
\[ \lambda x. \text{flight}(x) \land \text{oneway}(x) \]
*flights one-way*

Solution: disharmonic combinators: \( X : a \quad Y/X : f \) \[ \Rightarrow \quad Y : f(a) \]
Dependency-Based Compositional Semantics (DCS)

What is the most populous city in California?
What is the most populous city in California?

city

population

argmax

loc

CA
How to interpret the logical form?

What is the most populous city in California?
How to interpret the logical form?

What is the most populous city in California?

Los Angeles
How to interpret the logical form?

What is the most populous city in California?

World/Database \(\xrightarrow{\text{Los Angeles}}\)
### World/Database

<table>
<thead>
<tr>
<th>city</th>
<th>state</th>
</tr>
</thead>
<tbody>
<tr>
<td>San Francisco</td>
<td>Alabama</td>
</tr>
<tr>
<td>Chicago</td>
<td>Alaska</td>
</tr>
<tr>
<td>Boston</td>
<td>Arizona</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>loc</th>
<th>border</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mount Shasta</td>
<td>Washington</td>
</tr>
<tr>
<td>San Francisco</td>
<td>Oregon</td>
</tr>
<tr>
<td>Boston</td>
<td>Idaho</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

...
Basic DCS Trees

DCS tree

Database

city
1
1
1
loc
2
1
CA
13
A DCS tree encodes a **constraint satisfaction problem** (CSP)
A DCS tree encodes a constraint satisfaction problem (CSP)
A DCS tree encodes a constraint satisfaction problem (CSP)
A DCS tree encodes a constraint satisfaction problem (CSP)
A DCS tree encodes a constraint satisfaction problem (CSP)
A DCS tree encodes a constraint satisfaction problem (CSP)
A DCS tree encodes a constraint satisfaction problem (CSP)
A DCS tree encodes a **constraint satisfaction problem (CSP)**
Basic DCS Trees

A DCS tree encodes a constraint satisfaction problem (CSP)
A DCS tree encodes a **constraint satisfaction problem (CSP)**

**Computation**: dynamic programming $\Rightarrow$ time $= O(\# \text{ nodes})$

---

**Basic DCS Trees**

**DCS tree**

- `city`
- `loc`
- `CA`

**Constraints**

- $c \in \text{city}$
- $c_1 = \ell_1$
- $\ell \in \text{loc}$
- $\ell_2 = s_1$
- $s \in \text{CA}$

**Database**

- **city**
  - San Francisco
  - Chicago
  - Boston
  - ...

- **loc**
  - Mount Shasta California
  - San Francisco California
  - Boston Massachusetts
  - ...
  - ...

- **CA**
  - California
Properties of DCS Trees

![Diagram of a DCS Tree]

- city
  - loc
    - state
      - border
        - CA
  - traverse
    - major
    - traverse
      - AZ
Properties of DCS Trees

```
  city
   1 1
   / \   \
loc 2   traverse
   1 1
      / \
state 1   river
     1 1
        / \
border 1   major
      1 1
         / \
  CA 1   traverse
       2 1
          / \
      AZ
```

Trees
Properties of DCS Trees

Linguistics

syntactic locality

Trees
Divergence between Syntactic and Semantic Scope

*most populous city in California*
Divergence between Syntactic and Semantic Scope

*most populous city in California*

**Syntax**

- `city`
- `most`
- `populous`
- `in`
- `California`
Divergence between Syntactic and Semantic Scope

*most populous city in California*

**Syntax**

```
(\text{most}) \quad \text{in} \quad \text{California}
```

**Semantics**

```
\text{argmax}(\lambda x. \text{city}(x) \land \text{loc}(x, \text{CA}), \lambda x. \text{population}(x))
```
Divergence between Syntactic and Semantic Scope

*most populous city in California*

**Syntax**

```
city
```

**Semantics**

```
\text{argmax}(\lambda x.\text{city}(x) \land \text{loc}(x, \text{CA}), \lambda x.\text{population}(x))
```
Divergence between Syntactic and Semantic Scope

most populous city in California

Syntax

Semantics

\[
\text{argmax}(\lambda x. \text{city}(x) \land \text{loc}(x, \text{CA}), \lambda x. \text{population}(x))
\]

Problem: syntactic scope is lower than semantic scope
Divergence between Syntactic and Semantic Scope

*most populous city in California*

**Syntax**

```
 city
/    
|    |
/    |
/
 most  in
```

**Semantics**

```
argmax(\lambda x. \text{city}(x) \land \text{loc}(x, \text{CA}), \lambda x. \text{population}(x))
```

**Problem:** syntactic scope is lower than semantic scope

If DCS trees look like syntax, how do we get correct semantics?
Solution: Mark-Execute

most populous city in California

Superlatives

\[
\begin{array}{c}
\text{argmax} \\
\text{population} \\
\text{city} \\
X_1 \\
\end{array}
\]

\[
\begin{array}{c}
\text{loc} \\
\text{C} \\
\text{CA} \\
\end{array}
\]
most populous city in California

Mark at syntactic scope

Superlatives

Mark at syntactic scope
Solution: Mark-Execute

*most populous city in California*

**Execute** at semantic scope

**Mark** at syntactic scope
Solution: Mark-Execute

*Alaska borders no states.*

**Execute** at semantic scope

**Mark** at syntactic scope
Solution: Mark-Execute

*Some river traverses every city.*

**Execute** at semantic scope

**Mark** at syntactic scope
Solution: Mark-Execute

Some river traverses every city.

Execute at semantic scope

Mark at syntactic scope

Quantification (wide)
Some river traverses every city.

**Execute** at semantic scope

**Mark** at syntactic scope

Analogy: Montague’s quantifying in, Carpenter’s scoping constructor
From Sentences to DCS Trees

Lexicon (very simple/crude)

no $\Rightarrow$ no

state $\Rightarrow$ state
From Sentences to DCS Trees

Lexicon (very simple/crude)

\[ \text{no} \Rightarrow \text{no} \]
\[ \text{state} \Rightarrow \text{state} \]

Grammar (very simple/crude)

\[ ab \Rightarrow \]

\[ a \quad \downarrow \quad b \]

\[ a \quad \downarrow \quad b \]

\[ a \quad \downarrow \quad b \]

\[ a \quad \downarrow \quad b \]
From Sentences to DCS Trees

Lexicon (very simple/crude)

\[ no \Rightarrow no \]
\[ state \Rightarrow state \]

Grammar (very simple/crude)

\[ a \ b \Rightarrow i \ j \ k \ l \ b \ c \ a \]
\[ a \ b \Rightarrow i \ j \ k \ l \ a \ c \ b \]
Words to Predicates (Lexical Semantics)

What is the most populous city in CA?
Words to Predicates (Lexical Semantics)

What is the most populous city in CA?

Lexical Triggers:
1. String match

CA $\Rightarrow$ CA
Words to Predicates (Lexical Semantics)

argmax

What is the most populous city in CA?

Lexical Triggers:

1. String match

   CA ⇒ CA

2. Function words (20 words)

   most ⇒ argmax
Words to Predicates (Lexical Semantics)

What is the most populous city in CA?

Lexical Triggers:

1. String match

   CA \Rightarrow CA

2. Function words (20 words)

   most \Rightarrow \text{argmax}

3. Nouns/adjectives

   city \Rightarrow \text{city state river population}
Predicates to DCS Trees (Compositional Semantics)

\[ C_{i,j} = \text{set of DCS trees for span } [i, j] \]
Predicates to DCS Trees (Compositional Semantics)

\[ C_{i,j} = \text{set of DCS trees for span } [i, j] \]
Predicates to DCS Trees (Compositional Semantics)

\[ C_{i,j} = \text{set of DCS trees for span } [i, j] \]
Predicates to DCS Trees (Compositional Semantics)

\[ C_{i,j} = \text{set of DCS trees for span } [i, j] \]
Predicates to DCS Trees (Compositional Semantics)

\( C_{i,j} = \text{set of DCS trees for span } [i, j] \)

city

most populous city in California

\( C_{i,k} \)

\( C_{k,j} \)
Predicates to DCS Trees (Compositional Semantics)

\[ C_{i,j} = \text{set of DCS trees for span } [i, j] \]

Diagram:

- \( C_{i,k} \) represents the set of DCS trees for the span \( [i, k] \) and is associated with the predicate `most populous`.
- \( C_{k,j} \) represents the set of DCS trees for the span \( [k, j] \) and is associated with the predicate `city in California`.

Nodes:
- `population`
- `argmax`
- `city`
- `loc`
- `CA`

Edges:
- Arrows indicate the direction of the predicate application.
- The nodes are connected to represent the syntactic structure of the predicate expressions.
$C_{i,j} = \text{set of DCS trees for span } [i, j]$
Predicates to DCS Trees (Compositional Semantics)

\[ C_{i,j} = \text{set of DCS trees for span } [i, j] \]
Predicates to DCS Trees (Compositional Semantics)

\[ C_{i,j} = \text{set of DCS trees for span } [i, j] \]

most populous city in California

\[ \text{argmax}_C \text{ population} \]

city

\[ \text{loc}_{\text{CA}} \]

[\text{city}]

\[ \text{border} \]

\[ \text{population} \]

\[ \text{argmax}_C \text{ population} \]

\[ \text{city} \]

\[ \text{loc}_{\text{CA}} \]

\[ C_{i,k} \]

\[ C_{k,j} \]
Predicates to DCS Trees (Compositional Semantics)

\[ C_{i,j} = \text{set of DCS trees for span } [i, j] \]

most populous city in California

\[ C_{i,k} \]

\[ C_{k,j} \]

\[ \text{argmax} \]

\[ \text{population} \]

\[ \text{city} \]

\[ \text{loc} \]

\[ \text{CA} \]

\[ \text{most populous} \]

\[ \text{city in California} \]
<table>
<thead>
<tr>
<th></th>
<th>CCG</th>
<th>DCS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Comparison

<table>
<thead>
<tr>
<th>CCG</th>
<th>DCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical form</td>
<td>DCS trees</td>
</tr>
<tr>
<td>lambda calculus formulae</td>
<td></td>
</tr>
<tr>
<td>( \lambda x.\text{city}(x) \land \text{loc}(x,\text{CA}) )</td>
<td></td>
</tr>
</tbody>
</table>

\[ \text{city} \quad 1 \quad \text{loc} \quad 2 \quad \text{CA} \]
## Comparison

<table>
<thead>
<tr>
<th></th>
<th>CCG</th>
<th>DCS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Logical form</strong></td>
<td>lambda calculus formulae</td>
<td>DCS trees</td>
</tr>
<tr>
<td></td>
<td>$\lambda x.\text{city}(x) \land \text{loc}(x, \text{CA})$</td>
<td>$\text{city} \quad 1 \quad 1 \quad \text{loc} \quad 2 \quad 1 \quad \text{CA}$</td>
</tr>
<tr>
<td><strong>Lexicon</strong></td>
<td>categories + lambda calculus</td>
<td>predicates</td>
</tr>
<tr>
<td><strong>major</strong></td>
<td>$\mathbb{N}/\mathbb{N} : \lambda f.\lambda x.f(x) \land \text{major}(x)$</td>
<td>major</td>
</tr>
</tbody>
</table>
Comparison

**CCG**

**Logical form**

Lambda calculus formulae

\[ \lambda x. \text{city}(x) \land \text{loc}(x, \text{CA}) \]

**Lexicon**

Categories + lambda calculus predicates

**major**

\[ N/N : \lambda f. \lambda x. f(x) \land \text{major}(x) \]

**Grammar**

Combinator rules

\[ Y/X : a \quad X : b \quad \Rightarrow \quad Y : a(b) \]

**DCS**

DCS trees

\[ \text{city} - 1 \quad \text{loc} - 2 \quad \text{CA} \]

**Lexicon**

Predicates

**major**

\[ \text{major} \]

**Grammar**

Approximately dependency parsing

\[ \text{a} - i \quad j \quad \text{b} \]
## Comparison

<table>
<thead>
<tr>
<th></th>
<th>CCG</th>
<th>DCS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Logical form</strong></td>
<td>lambda calculus formulae</td>
<td>DCS trees</td>
</tr>
<tr>
<td></td>
<td>$\lambda x.\text{city}(x) \land \text{loc}(x, \text{CA})$</td>
<td>city 1 (\rightarrow) loc 2 (\rightarrow) CA</td>
</tr>
<tr>
<td><strong>Lexicon</strong></td>
<td>categories + lambda calculus</td>
<td>predicates</td>
</tr>
<tr>
<td><strong>major</strong></td>
<td>$N/N : \lambda f.\lambda x.f(x) \land \text{major}(x)$</td>
<td>major</td>
</tr>
<tr>
<td><strong>Grammar</strong></td>
<td>combinator rules</td>
<td>$\approx$ dependency parsing</td>
</tr>
<tr>
<td></td>
<td>$Y/X : a \quad X : b \quad \Rightarrow \quad Y : a(b)$</td>
<td>a (\rightarrow_i) j (\rightarrow_b)</td>
</tr>
<tr>
<td><strong>Nature</strong></td>
<td>tighter control</td>
<td>simple/permissive</td>
</tr>
</tbody>
</table>
## Comparison

<table>
<thead>
<tr>
<th></th>
<th><strong>CCG</strong></th>
<th><strong>DCS</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Logical form</strong></td>
<td>lambda calculus formulae</td>
<td>DCS trees</td>
</tr>
<tr>
<td></td>
<td>$\lambda x.\text{city}(x) \land \text{loc}(x,\text{CA})$</td>
<td>$\text{city} \quad 1 \quad 1 \quad \text{loc} \quad 2 \quad 1 \quad \text{CA}$</td>
</tr>
<tr>
<td><strong>Lexicon</strong></td>
<td>categories + lambda calculus</td>
<td>predicates</td>
</tr>
<tr>
<td><strong>major</strong></td>
<td>$N/N : \lambda f.\lambda x.f(x) \land \text{major}(x)$</td>
<td>major</td>
</tr>
<tr>
<td><strong>Grammar</strong></td>
<td>combinator rules</td>
<td>$\simeq$ dependency parsing</td>
</tr>
<tr>
<td></td>
<td>$Y/X : a \quad X : b \quad \Rightarrow \quad Y : a(b)$</td>
<td>$a \quad i \quad j \quad b$</td>
</tr>
<tr>
<td><strong>Nature</strong></td>
<td>tighter control</td>
<td>simple/permissive</td>
</tr>
<tr>
<td><strong>Origin</strong></td>
<td>linguistics</td>
<td>NLP</td>
</tr>
</tbody>
</table>
Outline

Representation

Learning

Experiments
What is the largest city in California?

\[ \text{argmax}\left(\{c : \text{city}(c) \land \text{loc}(c, \text{CA})\}, \text{population}\right) \]
Supervision

**Detailed Supervision**

What is the largest city in California?

expert

\[
\text{argmax}\left(\{c : \text{city}(c) \land \text{loc}(c, \text{CA})\}, \text{population}\right)
\]
Supervision

Detailed Supervision
- doesn’t scale up

What is the largest city in California?

expert

argmax({c : city(c) ∧ loc(c, CA)}, population)
Supervision

**Detailed Supervision**
- doesn’t scale up

**What is the largest city in California?**

expert

\[
\text{argmax}\left\{c : \text{city}(c) \wedge \text{loc}(c, \text{CA})\right\}, \text{population}
\]

**Natural Supervision**

**What is the largest city in California?**

*Los Angeles*
Supervision

**Detailed Supervision**
- doesn’t scale up

*What is the largest city in California?*

\[
\text{expert} \\ \text{argmax}\{c : \text{city}(c) \land \text{loc}(c, \text{CA})\}, \text{population}
\]

**Natural Supervision**

*What is the largest city in California?*

\[
\text{non-expert} \\ \text{Los Angeles}
\]
Supervision

Detailed Supervision
- doesn’t scale up

What is the largest city in California?

expert
argmax(\{c : \text{city}(c) \land \text{loc}(c, \text{CA})\}, \text{population})

Natural Supervision
- scales up

What is the largest city in California?

non-expert
Los Angeles
**Supervision**

**Detailed Supervision**
- doesn’t scale up
- representation-dependent

**What is the largest city in California?**

\[ \text{argmax}\{c : \text{city}(c) \land \text{loc}(c, \text{CA})\}, \text{population}\]  

**Natural Supervision**
- scales up

**What is the largest city in California?**

\[ \text{Los Angeles}\]
Supervision

**Detailed Supervision**
- doesn’t scale up
- representation-dependent

**Natural Supervision**
- scales up
- representation-independent

What is the largest city in California?

expert

\[ \text{argmax}\{c : \text{city}(c) \land \text{loc}(c, \text{CA})\}, \text{population} \]
Considerations

*Computational*: how to efficiently search exponential space?
Considerations

Computational: how to efficiently search exponential space?

What is the most populous city in California?

Los Angeles
Considerations

**Computational:** how to efficiently search exponential space?

*What is the most populous city in California?*

$\lambda x. \text{state}(x)$

*Los Angeles*
Considerations

**Computational:** how to efficiently search exponential space?

*What is the most populous city in California?*

\( \lambda x. \text{city}(x) \)

*Los Angeles*
Considerations

Computational: how to efficiently search exponential space?

What is the most populous city in California?

\[ \lambda x. \text{city}(x) \land \text{loc}(x, \text{CA}) \]

Los Angeles
Considerations

**Computational:** how to efficiently search exponential space?

*What is the most populous city in California?*

\[
\lambda x. \text{state}(x) \land \text{border}(x, \text{CA})
\]

*Los Angeles*
Considerations

**Computational**: how to efficiently search exponential space?

*What is the most populous city in California?*

*population(CA)*

*Los Angeles*
Considerations

Computational: how to efficiently search exponential space?

What is the most populous city in California?

\[
\text{argmax}(\lambda x. \text{city}(x) \land \text{loc}(x, \text{CA}), \lambda x. \text{population}(x))
\]

Los Angeles
Considerations

Computational: how to efficiently search exponential space?

What is the most populous city in California?

Los Angeles
Considerations

**Computational**: how to efficiently search exponential space?

What is the most populous city in California?

\[
\arg\max(\lambda x.\text{city}(x) \land \text{loc}(x, \text{CA}), \lambda x.\text{population}(x))
\]

Los Angeles

**Statistical**: how to parametrize mapping from sentence to logical form?

What is the most populous city in California?

\[
\arg\max(\lambda x.\text{city}(x) \land \text{loc}(x, \text{CA}), \lambda x.\text{population}(x))
\]
Graphical Model

world

\[ \text{world} \]

\[ z \]

\[ \text{capital} \]

\[ \text{CA} \]
Graphical Model

world

z
1
2
1
1
CA
capital

world

w

y
Sacramento
Interpretation: $p(y \mid z, w)$ (deterministic)
**Graphical Model**

Interpretation: $p(y \mid z, w)$ (deterministic)
**Graphical Model**

- **Capital of California?**
- **Parameters**: $\theta$
- **World**: $w$
- **World Capital**: $y$ (Sacramento)
- **Interpretation**: $p(y \mid z, w)$ (deterministic)
Graphical Model

Semantic Parsing: $p(z \mid x, \theta)$
(probablistic)

Interpretation: $p(y \mid z, w)$
(deterministic)
Semantic Parsing Log-linear Model

$z$: city, loc, CA

$x$: city in California
Semantic Parsing Log-linear Model

\[
\text{features}(x, z) = \begin{pmatrix}
1 \\
1 \\
2 \\
1
\end{pmatrix} \in \mathbb{R}^d
\]
Semantic Parsing Log-linear Model

\[
\text{features}(x, z) = \begin{pmatrix}
    \text{in} & \ldots \ldots & \text{loc} & : 1 \\
\end{pmatrix} \in \mathbb{R}^d
\]
Semantic Parsing Log-linear Model

$z$: city, loc, CA

$x$: city in California

features($x, z$) =
\[
\begin{pmatrix}
\text{in} & \ldots & \text{loc} & : 1 \\
\text{city} & 1 & 1 & \text{loc} & : 1
\end{pmatrix}
\in \mathbb{R}^d$
Semantic Parsing Log-linear Model

\[
\begin{aligned}
\text{features}(x, z) &= \begin{pmatrix}
\text{in} & \cdots & \text{loc} & : 1 \\
\text{city} & \cdots & \text{loc} & : 1 \\
\text{...} & & & \\
\end{pmatrix} \\
&\in \mathbb{R}^d
\end{aligned}
\]
Semantic Parsing Log-linear Model

$$
\begin{align*}
z &: \text{city} \quad \text{loc} \quad \text{CA} \\
x &: \text{city} \quad \text{in} \quad \text{California}
\end{align*}
$$

$$
\text{features}(x, z) = \begin{pmatrix}
\text{in} & \ldots & \text{loc} & : 1 \\
\text{city} & -1 & \text{loc} & : 1 \\
\ldots & & & \\
\end{pmatrix} \in \mathbb{R}^d
$$

$$
\text{score}(x, z) = \text{features}(x, z) \cdot \theta
$$
Semantic Parsing Log-linear Model

\[ z: \text{city} \quad \text{loc} \quad \text{CA} \]
\[ x: \text{city in California} \]

\[ \text{features}(x, z) = \left( \begin{array}{c} \text{in} \quad \text{loc} : 1 \\ \text{city} \quad \text{loc} : 1 \\ \cdots \end{array} \right) \in \mathbb{R}^d \]

\[ \text{score}(x, z) = \text{features}(x, z) \cdot \theta \]

\[ p(z \mid x, \theta) = \frac{e^{\text{score}(x, z)}}{\sum_{z' \in \mathcal{Z}(x)} e^{\text{score}(x, z')}} \]
Learning

Objective Function:

\[ p(y \mid z, w) \ p(z \mid x, \theta) \]

Interpretation: Semantic parsing
Learning

Objective Function:

$$\max_\theta \quad p(y \mid z, w) \ p(z \mid x, \theta)$$

Interpretation: Semantic parsing

26
Learning

Objective Function:

\[
\max_\theta \sum_z p(y \mid z, w) p(z \mid x, \theta)
\]

Interpretation: Semantic parsing
Learning

Objective Function:

$$\max_{\theta} \sum_z p(y \mid z, w) p(z \mid x, \theta)$$

Interpretation Semantic parsing

EM-like Algorithm:

parameters $\theta$

$(0, 0, \ldots, 0)$
Learning

Objective Function:

\[
\max_{\theta} \sum_z p(y \mid z, w) p(z \mid x, \theta)
\]

Interpretation

Semantic parsing

EM-like Algorithm:

parameters \( \theta \)

(0, 0, ..., 0)

enumerate/score DCS trees
Learning

Objective Function:

$$\max_{\theta} \sum_z p(y \mid z, w) p(z \mid x, \theta)$$

Interpretation  Semantic parsing

EM-like Algorithm:

parameters $\theta$ $k$-best list

(0, 0, ..., 0)

enumerate/score DCS trees

tree1

tree2

tree3 ✓
tree4

tree5
Learning

Objective Function:

\[ \max_{\theta} \sum_z p(y \mid z, w) p(z \mid x, \theta) \]

Interpretation Semantic parsing

EM-like Algorithm:

parameters \( \theta \)

\((0.2, -1.3, \ldots, 0.7)\)

enumerate/score DCS trees

numerical optimization (L-BFGS)

\(k\)-best list

tree1 ✗
tree2 ✗
tree3 ✓
tree4 ✗
tree5 ✗
Learning

Objective Function:

$$\max_{\theta} \sum_{z} p(y | z, w) p(z | x, \theta)$$

Interpretation
Semantic parsing

EM-like Algorithm:

parameters $\theta$

$$(0.2, -1.3, \ldots, 0.7)$$

enumerate/score DCS trees

numerical optimization (L-BFGS)

$k$-best list

- tree3 ✓
- tree8 ✓
- tree6 ✗
- tree2 ✗
- tree4 ✗
Learning

Objective Function:

\[
\max_\theta \sum_z p(y | z, w) p(z | x, \theta)
\]

Interpretation  Semantic parsing

EM-like Algorithm:

parameters \( \theta \)

\( (0.3, -1.4, \ldots, 0.6) \)

count

\( k \)-best list

tree3  

tree8  

tree6  

tree2  

tree4  

enumerate/score DCS trees

numerical optimization (L-BFGS)
Learning

Objective Function:
\[ \max_{\theta} \sum_{z} p(y \mid z, w) p(z \mid x, \theta) \]

Interpretation Semantic parsing

EM-like Algorithm:

Parameters \( \theta \)

(0.3, −1.4, …, 0.6)

Enumerate/score DCS trees

Numerical optimization (L-BFGS)

K-best list

- tree3
- tree8
- tree2
- tree4
- tree9
Outline

Representation

Learning

Experiments
US Geography Benchmark

Standard semantic parsing benchmark since 1990s
600 training examples, 280 test examples
US Geography Benchmark

Standard semantic parsing benchmark since 1990s
600 training examples, 280 test examples

What is the highest point in Florida?

How many states have a city called Rochester?

What is the longest river that runs through a state that borders Tennessee?

Of the states washed by the Mississippi river which has the lowest point?

…
US Geography Benchmark

Standard semantic parsing benchmark since 1990s
600 training examples, 280 test examples

What is the highest point in Florida?
⇒ answer(A, highest(A, (place(A), loc(A, B), const(B, stateid(florida)))))

How many states have a city called Rochester?
⇒ answer(A, count(B, (state(B), loc(C, B), const(C, cityid(rochester, _))), A))

What is the longest river that runs through a state that borders Tennessee?
⇒ answer(A, longest(A, (river(A), traverse(A, B), state(B), next_to(B, C), const(C, stateid(tennessee)))))

Of the states washed by the Mississippi river which has the lowest point?
⇒ answer(A, lowest(B, (state(A), traverse(C, A), const(C, riverid(mississippi)), loc(B, A), place(B))))

...
US Geography Benchmark

Standard semantic parsing benchmark since 1990s
600 training examples, 280 test examples

What is the highest point in Florida?
⇒ Walton County

How many states have a city called Rochester?
⇒ 2

What is the longest river that runs through a state that borders Tennessee?
⇒ Missouri

Of the states washed by the Mississippi river which has the lowest point?
⇒ Louisiana

... 

Supervision in past work: question + program
Supervision in this work: question + answer
### Input to Learning Algorithm

**Training data** (600 examples)

<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>What is the highest point in Florida?</td>
<td>Walton County</td>
</tr>
<tr>
<td>How many states have a city called Rochester?</td>
<td>2</td>
</tr>
<tr>
<td>What is the longest river that runs through a state that borders Tennessee?</td>
<td>Missouri</td>
</tr>
<tr>
<td>Of the states washed by the Mississippi river which has the lowest point?</td>
<td>Louisiana</td>
</tr>
</tbody>
</table>

...
## Input to Learning Algorithm

**Training data** (600 examples)

<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>What is the highest point in Florida?</td>
<td>Walton County</td>
</tr>
<tr>
<td>How many states have a city called Rochester?</td>
<td>2</td>
</tr>
<tr>
<td>What is the longest river that runs through a state that borders Tennessee?</td>
<td>Missouri</td>
</tr>
<tr>
<td>Of the states washed by the Mississippi river which has the lowest point?</td>
<td>Louisiana</td>
</tr>
</tbody>
</table>

... ...

## Lexicon (20 general, 22 specific)

<table>
<thead>
<tr>
<th>Term</th>
<th>Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>argmax</td>
<td>most</td>
</tr>
<tr>
<td>city</td>
<td>city</td>
</tr>
<tr>
<td>state</td>
<td>state</td>
</tr>
<tr>
<td>mountain</td>
<td>mountain</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Input to Learning Algorithm

Training data (600 examples)

What is the highest point in Florida?
How many states have a city called Rochester?
What is the longest river that runs through a state that borders Tennessee?
Of the states washed by the Mississippi river which has the lowest point?

World/Database

Lexicon (20 general, 22 specific)

<table>
<thead>
<tr>
<th>term</th>
<th>synset</th>
</tr>
</thead>
<tbody>
<tr>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>argmax</td>
<td>most</td>
</tr>
<tr>
<td>city</td>
<td>city</td>
</tr>
<tr>
<td>state</td>
<td>state</td>
</tr>
<tr>
<td>mountain</td>
<td>mountain</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>term</th>
<th>synset</th>
</tr>
</thead>
<tbody>
<tr>
<td>San Francisco</td>
<td>California</td>
</tr>
<tr>
<td>Chicago</td>
<td></td>
</tr>
<tr>
<td>Boston</td>
<td></td>
</tr>
<tr>
<td>Mount Shasta</td>
<td>California</td>
</tr>
<tr>
<td>Boston</td>
<td>Massachusetts</td>
</tr>
<tr>
<td>Washington</td>
<td>Oregon</td>
</tr>
<tr>
<td>Washington</td>
<td>Idaho</td>
</tr>
<tr>
<td>Oregon</td>
<td>Washington</td>
</tr>
</tbody>
</table>
Experiment 1

On $\text{GEO}$, 250 training examples, 250 test examples

![Graph showing test accuracy from 75 to 100 on the y-axis. Numbers 75, 80, 85, 90, 95, and 100 are marked on the y-axis.](image)
### Experiment 1

On Geo, 250 training examples, 250 test examples

<table>
<thead>
<tr>
<th>System</th>
<th>Description</th>
<th>Lexicon (gen./spec.)</th>
<th>Logical forms</th>
</tr>
</thead>
<tbody>
<tr>
<td>CGCR10</td>
<td>FunQL [Clarke et al., 2010]</td>
<td>✓ ✓</td>
<td>✗</td>
</tr>
</tbody>
</table>

#### Test Accuracy

![Graph showing test accuracy of CGCR10 at 73.2%]
Experiment 1

On GEO, 250 training examples, 250 test examples

<table>
<thead>
<tr>
<th>System</th>
<th>Description</th>
<th>Lexicon (gen./spec.)</th>
<th>Logical forms</th>
</tr>
</thead>
<tbody>
<tr>
<td>CGCR10</td>
<td>FunQL [Clarke et al., 2010]</td>
<td>✔ ✔</td>
<td>✗</td>
</tr>
<tr>
<td>LJK11</td>
<td>DCS [Liang et al., 2011]</td>
<td>✔ ✗</td>
<td>✗</td>
</tr>
</tbody>
</table>

![Test accuracy diagram]

- CGCR10: 73.2%
- DCS: 78.9%
Experiment 1

On GEO, 250 training examples, 250 test examples

<table>
<thead>
<tr>
<th>System</th>
<th>Description</th>
<th>Lexicon (gen./spec.)</th>
<th>Logical forms</th>
</tr>
</thead>
<tbody>
<tr>
<td>CGCR10</td>
<td>FunQL [Clarke et al., 2010]</td>
<td>✓ ✓</td>
<td>✗</td>
</tr>
<tr>
<td>LJK11</td>
<td>DCS [Liang et al., 2011]</td>
<td>✓ ✗</td>
<td>✗</td>
</tr>
<tr>
<td>LJK11⁺</td>
<td>DCS [Liang et al., 2011]</td>
<td>✓ ✓</td>
<td>✗</td>
</tr>
</tbody>
</table>

![Test Accuracy Chart](chart.png)

- CGCR10: 73.2%
- DCS: 78.9%
- DCS⁺: 87.2%
Experiment 2

On Geo, 600 training examples, 280 test examples
Experiment 2

On Geo, 600 training examples, 280 test examples

System Description

Lexicon Logical forms

Test accuracy
Experiment 2

On Geo, 600 training examples, 280 test examples

<table>
<thead>
<tr>
<th>System</th>
<th>Description</th>
<th>Lexicon</th>
<th>Logical forms</th>
</tr>
</thead>
<tbody>
<tr>
<td>zc05</td>
<td>CCG [Zettlemoyer &amp; Collins, 2005]</td>
<td>✗ ✗</td>
<td>✓</td>
</tr>
</tbody>
</table>

![Bar chart showing test accuracy](chart.png)

**Accuracy:** 79.3%
Experiment 2

On Geo, 600 training examples, 280 test examples

<table>
<thead>
<tr>
<th>System</th>
<th>Description</th>
<th>Lexicon</th>
<th>Logical forms</th>
</tr>
</thead>
<tbody>
<tr>
<td>zc05</td>
<td>CCG [Zettlemoyer &amp; Collins, 2005]</td>
<td>✗✗</td>
<td>✓</td>
</tr>
<tr>
<td>zc07</td>
<td>relaxed CCG [Zettlemoyer &amp; Collins, 2007]</td>
<td>✗✗</td>
<td>✓</td>
</tr>
</tbody>
</table>

![Bar chart showing test accuracy](chart.png)

- zc05: 79.3%
- zc07: 86.1%
## Experiment 2

On Geo, 600 training examples, 280 test examples

<table>
<thead>
<tr>
<th>System</th>
<th>Description</th>
<th>Lexicon</th>
<th>Logical forms</th>
</tr>
</thead>
<tbody>
<tr>
<td>zc05</td>
<td>CCG [Zettlemoyer &amp; Collins, 2005]</td>
<td>X X</td>
<td>✓</td>
</tr>
<tr>
<td>zc07</td>
<td>relaxed CCG [Zettlemoyer &amp; Collins, 2007]</td>
<td>X X</td>
<td>✓</td>
</tr>
<tr>
<td>KZGS10</td>
<td>CCG w/unification [Kwiatkowski et al., 2010]</td>
<td>X X</td>
<td>✓</td>
</tr>
</tbody>
</table>

![Bar chart showing test accuracy](chart.png)

- zc05: 79.3%
- zc07: 86.1%
- KZGS10: 88.9%
Experiment 2

On Geo, 600 training examples, 280 test examples

<table>
<thead>
<tr>
<th>System</th>
<th>Description</th>
<th>Lexicon</th>
<th>Logical forms</th>
</tr>
</thead>
<tbody>
<tr>
<td>zc05</td>
<td>CCG [Zettlemoyer &amp; Collins, 2005]</td>
<td>✗ ✗</td>
<td>✓</td>
</tr>
<tr>
<td>zc07</td>
<td>relaxed CCG [Zettlemoyer &amp; Collins, 2007]</td>
<td>✗ ✗</td>
<td>✓</td>
</tr>
<tr>
<td>KZGS10</td>
<td>CCG w/unification [Kwiatkowski et al., 2010]</td>
<td>✗ ✗</td>
<td>✓</td>
</tr>
<tr>
<td>LJK11</td>
<td>DCS [Liang et al., 2011]</td>
<td>✓ ✗</td>
<td>✗</td>
</tr>
</tbody>
</table>

![Test accuracy chart](chart.png)

- zc05: 79.3%
- zc07: 86.1%
- KZGS10: 88.9%
- DCS: 88.6%

31
## Experiment 2

On Geo, 600 training examples, 280 test examples

<table>
<thead>
<tr>
<th>System</th>
<th>Description</th>
<th>Lexicon</th>
<th>Logical forms</th>
</tr>
</thead>
<tbody>
<tr>
<td>zc05</td>
<td>CCG [Zettlemoyer &amp; Collins, 2005]</td>
<td></td>
<td>X X</td>
</tr>
<tr>
<td>zc07</td>
<td>relaxed CCG [Zettlemoyer &amp; Collins, 2007]</td>
<td>X X</td>
<td></td>
</tr>
<tr>
<td>KZGS10</td>
<td>CCG w/unification [Kwiatkowski et al., 2010]</td>
<td>X X</td>
<td></td>
</tr>
<tr>
<td>LJK11</td>
<td>DCS [Liang et al., 2011]</td>
<td></td>
<td>X X</td>
</tr>
<tr>
<td>LJK11+</td>
<td>DCS [Liang et al., 2011]</td>
<td>✓ ✓ ✓</td>
<td>X</td>
</tr>
</tbody>
</table>

![Accuracy Chart](https://via.placeholder.com/150)

- **zc05**: 79.3%
- **zc07**: 86.1%
- **KZGS10**: 88.9%
- **DCS**: 88.6%
- **DCS+**: 91.1%
Some Intuition on Learning
Some Intuition on Learning

parameters $\theta$ → (1) search DCS trees (hard!) → $k$-best lists

(2) numerical optimization
Some Intuition on Learning

(1) search DCS trees (hard!)

parameters $\theta$ \quad $\rightarrow$ \quad $k$-best lists

(2) numerical optimization

If no DCS tree on $k$-best list is correct, skip example in (2)
Some Intuition on Learning

parameters $\theta$

(1) search DCS trees (hard!)

(2) numerical optimization

$k$-best lists

If no DCS tree on $k$-best list is correct, skip example in (2)
Some Intuition on Learning

parameters $\theta$

(1) search DCS trees (hard!)

(2) numerical optimization

$k$-best lists

If no DCS tree on $k$-best list is correct, skip example in (2)

Effect: automatic curriculum learning, learning improves search
Current Limitations

Unknown facts: \textit{How far is Los Angeles from Boston?}

Database has no distance information
Current Limitations

Unknown facts: *How far is Los Angeles from Boston?*
   Database has no distance information

Unknown concepts: *What states are landlocked?*
   Need to induce database view for \( \text{landlocked}(x) = \neg \text{border}(x, \text{ocean}) \)
Current Limitations

Unknown facts: *How far is Los Angeles from Boston?*
   Database has no distance information

Unknown concepts: *What states are landlocked?*
   Need to induce database view for \( \text{landlocked}(x) = \neg \text{border}(x, \text{ocean}) \)

Unknown words: *What is the largest settlement in California?*
   Training examples do not contain the word *settlement*
Summary

sentence → Semantic Parser → logical form → Interpretation → denotation
Learning from Weak Supervision

- Model logical form as latent variable
- Semantic formalisms: CCG, DCS
Summary

Learning from Weak Supervision

- Model logical form as latent variable
- Semantic formalisms: CCG, DCS

Strategy:
- Lexicon/grammar generates set of candidate logical forms
- Learned feature weights capture linguistic generalizations