Goals

• Show how you can use SEMPRE for question answering on Freebase.

• Highlight the many applications of SEMPRE.
Outline

Review of semantic parsing

Using SEMPRE for Freebase QA

Other applications
Question answering via semantic parsing

Which states’ capitals are also their largest cities?
Question answering via semantic parsing

Which states’ capitals are also their largest cities?

\[ \mu_x. \text{Type.USState} \sqcap \text{Capital}.\text{argmax}(\text{Type.City} \sqcap \text{ContainedBy}.x, \text{Area}) \]
Question answering via semantic parsing

*Which states’ capitals are also their largest cities?*

\[
\mu x. \text{Type.USState} \sqcap \text{Capital}.\text{argmax}(\text{Type.City} \sqcap \text{ContainedBy}.x,\text{Area})
\]

execute

Arizona, Hawaii, Idaho, Indiana, Iowa, Oklahoma, Utah
Question answering via semantic parsing

Which states’ capitals are also their largest cities?

Semantic parsing

execute

Arizona, Hawaii, Idaho, Indiana, Iowa, Oklahoma, Utah
General framework

[utterance: user input]

semantic parsing

[program]

execute

[behavior: user output]
Semantics?

Percy teaches at Stanford.

\[ \text{teachesAt}(\text{Percy}, \text{Stanford}) \]
Semantics?

Semantic parsing is fundamentally a **translation** task...
Semantics?

Semantic parsing is fundamentally a **translation** task...

How many people live in Seattle?

\[
\text{count} (\text{Type.Person} \sqcap \text{LiveIn.Seattle})
\]
Semantics?

Semantic parsing is fundamentally a translation task...

*How many people live in Seattle?*

R[Population].Seattle
Semantics?

Semantic parsing is fundamentally a **translation** task...

*How many people live in Seattle?*

\[ R[\text{Number}].(R[\text{Population}].\text{Seattle} \sqcap \text{Year.2015}) \]
Semantics?

Semantic parsing is fundamentally a **translation** task...

*How many people live in Seattle?*

\[ \text{R[Number]. arg max}(\text{R[Population].Seattle \land Year. \leq .2015}, \text{Year}) \]

...into a low-level language.
Probabilistic framework

\[ x \] people who have lived in Chicago

parameters

\[ \theta \rightarrow z \] Type.Person ⊓ PlacesLived.Location.Chicago

context

\[ w \rightarrow y \] \{BarackObama,MichelleObama,...\}
Freebase

100M entities (nodes)  1B assertions (edges)

MichelleObama

Gender

Female

PlacesLived

Spouse

1992.10.03

USState

Hawaii

Event21

Location

Type

Event8

StartDate

UnitedStates

ContainedBy

Event3

Location

Type

Chicago

BarackObama

PlaceOfBirth

Honolulu

Person

1961.08.04

Politician

City

[Boellacker, 2008; Google, 2013]
Logical forms: lambda DCS

Type.Person ⊓ PlacesLived.Location.Chicago
Logical forms: lambda DCS

Type.Person ⊓ PlacesLived.Location.Chicago

[Liang, 2013]
Logical forms: lambda DCS

Type.Person \sqcap PlacesLived.Location.Chicago

- **BarackObama**
  - Profession: Politician
  - DateOfBirth: 1961.08.04
  - PlaceOfBirth: Honolulu, USState

- **MichelleObama**
  - Gender: Female
  - Spouse: BarackObama
  - StartDate: 1992.10.03
  - PlaceLived: Chicago

- **Chicago**
  - ContainedBy: City, UnitedStates

- **Honolulu**
  - ContainedBy: City, UnitedStates

[Source: Liang, 2013]
Logical forms: lambda DCS

Type.Person ⊓ PlacesLived.Location.Chicago

- **Person**: (Type, PlacesLived)
- **Chicago**: Location
- **MichelleObama**: (Gender, Female, Type)
- **UnitedStates**: ContainedBy (City, Honolulu)
- **Hawaii**: ContainedBy (USState)
- **Event8**: StartDate (1992.10.03)
- **Event21**: Spouse
- **Event3**: PlaceOfBirth (Honolulu)
- **BarackObama**: (Type, Politician, DateOfBirth (1961.08.04))
Lambda DCS

Entity
Chicago
Lambda DCS

Entity
Chicago

Join
PlaceOfBirth.Chicago
Lambda DCS

Entity
Chicago

Join
PlaceOfBirth.Chicago

Intersect
Type.Person
PlaceOfBirth.Chicago
Lambda DCS

Entity
Chicago

Join
PlaceOfBirth.Chicago

Intersect
Type.Person ▇ PlaceOfBirth.Chicago

Aggregation
count(Type.Person ▇ PlaceOfBirth.Chicago)
Lambda DCS

Entity
Chicago

Join
PlaceOfBirth.Chicago

Intersect
Type.Person ⊓ PlaceOfBirth.Chicago

Aggregation
\text{count}(\text{Type.Person} ∩ \text{PlaceOfBirth.Chicago})

Superlative
\text{argmin}(\text{Type.Person} ∩ \text{PlaceOfBirth.Chicago}, \text{DateOfBirth})
Lambda DCS

Entity
Chicago

Join
PlaceOfBirth.Chicago

Intersect
Type.Person ⊓ PlaceOfBirth.Chicago

Aggregation
\textit{count}(\textit{Type.Person} ⊓ \textit{PlaceOfBirth.Chicago})

Superlative
\textit{argmin}(\textit{Type.Person} ⊓ \textit{PlaceOfBirth.Chicago}, \textit{DateOfBirth})

Anaphora
\(\mu x. \text{Type.Person} ⊓ \text{Children.Influence}.x\)
Lambda DCS

Entity
Chicago

Join
PlaceOfBirth.Chicago

Intersect
Type.Person △ PlaceOfBirth.Chicago

Aggregation
count(Type.Person △ PlaceOfBirth.Chicago)

Superlative
argmin(Type.Person △ PlaceOfBirth.Chicago, DateOfBirth)

Anaphora
μx. Type.Person △ Children.Influence.x

Variable
argmax(Type.Person, R[λx.count(Parent.Parent.x)])
Comparison to lambda calculus

Lambda calculus

\[ \lambda x. \text{Type}(x, \text{Person}) \land \exists e. \text{PlacesLived}(x, e) \land \text{Location}(e, \text{Chicago}) \]
Comparison to lambda calculus

Lambda calculus

\[ \lambda x. \text{Type}(x, \text{Person}) \land \exists e. \text{PlacesLived}(x, e) \land \text{Location}(e, \text{Chicago}) \]

Lambda dependency-based compositional semantics (DCS)

\[ \text{Type.Person} \sqcap \text{PlacesLived.Location.Chicago} \]
Probabilistic framework

\( x \)  people who have lived in Chicago

parameters

\( \theta \)  Type.Person ∩ PlacesLived.Location.Chicago

context

\( w \)  \{BarackObama,MichelleObama,…\}

(Over)-generating derivations

utterance ➔ Grammar ➔ derivation 1

 derivation 2
...


(Over)-generating derivations

A Real Dumb Grammar

(lexicon) *Chicago* ⇒ E : Chicago
(lexicon) *people* ⇒ E : Type.Person
(lexicon) *live* ⇒ E × E : PlacesLived

... 

(join) E × E : b E : u ⇒ E : b.u
(intersect) E : u E : v ⇒ E : u ⊓ v
Derivations

Derivation: construction of logical form given utterance

Type.Person ⊓ PlaceLived.Location.Chicago

Type.Person who
people

PlaceLived.Location.Chicago
have
PlaceLived.Location
lived
in Chicago
Chicago
Derivations

Derivation: construction of logical form given utterance

Type.Person ⊓ PlaceLived.Location.Chicago

Type.Person

people

who

PlaceLived.Location.Chicago

have

PlaceLived.Location

in

Chicago

lexicon

lived

Chicago
Derivations

Derivation: construction of logical form given utterance

Type.Person ⊓ PlaceLived.Location.Chicago

Type.Person people

PlaceLived.Location.Chicago

join

lexicon lexicon lexicon

have lived in Chicago

lexicon lexicon
Derivations

Derivation: construction of logical form given utterance
Floating parsers

Type.Person \(\cap\) PlaceLived.Location.Chicago

\(\text{intersect}\)

Type.Person \(\mathbf{who}\) PlaceLived.Location.Chicago

\(\text{join}\)

People \(\text{have}\) PlaceLived.Location \(\text{in}\) Chicago

\(\text{lexicon}\)

Lived \(\text{Chicago}\)
Floating parsers

Key idea: detach logical form from sentence
Floating parsers

Type.Person \( \cap \) PlaceLived.Location.Chicago

intersect

Type.Person

PlaceLived.Location.Chicago

join

PlaceLived.Location

Chicago

lexicon

people who have lived in Chicago

Key idea: detach logical form from sentence

Pruning: use world knowledge / pragmatics — avoid empty sets, type errors, redundant operations
Many possible derivations!

\[ x = \text{people who have lived in Chicago} \]
Many possible derivations!

\[ x = \text{people who have lived in Chicago} \]

set of candidate derivations \( \mathcal{D}(x) \)
Many possible derivations!

\[ x = \text{people who have lived in Chicago} \]

set of candidate derivations \( \mathcal{D}(x) \)
Many possible derivations!

\[ x = \text{people who have lived in Chicago} \]

set of candidate derivations \( \mathcal{D}(x) \)
$x$: utterance

$d$: derivation

Feature vector $\phi(x, d) \in \mathbb{R}^F$: 
$x$: utterance

$d$: derivation

Feature vector $\phi(x, d) \in \mathbb{R}^F$:

- apply join: 1
- apply intersect: 1
- apply lexicon: 3
- skipped IN: 1
- skipped NN: 0

$lived$ maps to $PlacesLived.Location$: 1

Alignment score: 1.52

Denotation size: 1

...
Scoring derivations

Feature vector: $\phi(x, d) = [1.3, 2, 0, 1, 0, 0, \ldots] \in \mathbb{R}^F$
Scoring derivations

Feature vector: \( \phi(x, d) = [1.3, 2, 0, 1, 0, 0, \ldots] \in \mathbb{R}^F \)

Parameter vector: \( \theta = [1.2, -2.7, 3.4, \ldots] \in \mathbb{R}^F \)
Scoring derivations

Feature vector: \( \phi(x, d) = [1.3, 2, 0, 1, 0, 0, \ldots] \in \mathbb{R}^F \)

Parameter vector: \( \theta = [1.2, -2.7, 3.4, \ldots] \in \mathbb{R}^F \)

Scoring function:

\[
\text{Score}_\theta(x, d) = \phi(x, d) \cdot \theta = \sum_{j=1}^{F} \theta_j \phi_j(x, d)
\]
Log-linear model

Candidate derivations (defined by grammar): $D(x)$
Log-linear model

Candidate derivations (defined by grammar): \( \mathcal{D}(x) \)

Model: distribution over derivations \( d \) given utterance \( x \)

\[
p(d \mid x, \theta) = \frac{\exp(\text{Score}_\theta(x,d))}{\sum_{d' \in \mathcal{D}(x)} \exp(\text{Score}_\theta(x,d'))}
\]
Probabilistic framework

\[ x \quad \text{people who have lived in Chicago} \]

\[ \theta \rightarrow z \quad \text{Type.Person} \sqcap \text{PlacesLived.Location.Chicago} \]

\[ w \rightarrow y \quad \{\text{BarackObama, MichelleObama, ...}\} \]
Learning

Training data:

What’s Bulgaria’s capital?
Sofia
What movies has Tom Cruise been in?
TopGun,VanillaSky,...
...
+grammar, +features
Learning

Training data:

What’s Bulgaria’s capital?
Sofia

What movies has Tom Cruise been in?
TopGun, VanillaSky,...
...
+grammar, +features

Objective: Maximum likelihood

$$\arg \max_{\theta} \sum_{i=1}^{n} \log p_{\theta}(y^{(i)} \mid x^{(i)})$$
Learning

Training data:

- What’s Bulgaria’s capital?
  - Sofia
- What movies has Tom Cruise been in?
  - TopGun, VanillaSky, ...
  - ...

+ grammar, + features

Objective: Maximum likelihood

\[
\arg \max_{\theta} \sum_{i=1}^{n} \log p_{\theta}(y^{(i)} \mid x^{(i)})
\]

Algorithm:

AdaGrad (stochastic gradient with per-feature step size)
Training intuition

Where did Mozart tupress?

Vienna
Training intuition

*Where did Mozart *tupress?*

PlaceOfBirth.WolfgangMozart
PlaceOfDeath.WolfgangMozart
PlaceOfMarriage.WolfgangMozart

**Vienna**
Training intuition

*Where did Mozart tupress?*

PlaceOfBirth.WolfgangMozart ⇒ Salzburg

PlaceOfDeath.WolfgangMozart ⇒ Vienna

PlaceOfMarriage.WolfgangMozart ⇒ Vienna

Vienna
Training intuition

*Where did Mozart tupress?*

PlaceOfBirth.WolfgangMozart ⇒ Salzburg

PlaceOfDeath.WolfgangMozart ⇒ Vienna

PlaceOfMarriage.WolfgangMozart ⇒ Vienna

Vienna
Training intuition

Where did Mozart tüzpress?
PlaceOfBirth.WolfgangMozart ⇒ Salzburg
PlaceOfDeath.WolfgangMozart ⇒ Vienna
PlaceOfMarriage.WolfgangMozart ⇒ Vienna

Vienna

Where did Hogarth tüzpress?
Training intuition

Where did Mozart tupress?

PlaceOfBirth.WolfgangMozart ⇒ Salzburg
PlaceOfDeath.WolfgangMozart ⇒ Vienna
PlaceOfMarriage.WolfgangMozart ⇒ Vienna

Vienna

Where did Hogarth tupress?

PlaceOfBirth.WilliamHogarth
PlaceOfDeath.WilliamHogarth
PlaceOfMarriage.WilliamHogarth

London
Training intuition

*Where did Mozart typepress?*

\[
\text{PlaceOfBirth}.\text{WolfgangMozart} \Rightarrow \text{Salzburg} \\
\text{PlaceOfDeath}.\text{WolfgangMozart} \Rightarrow \text{Vienna} \\
\text{PlaceOfMarriage}.\text{WolfgangMozart} \Rightarrow \text{Vienna}
\]

*Vienna*

*Where did Hogarth typepress?*

\[
\text{PlaceOfBirth}.\text{WilliamHogarth} \Rightarrow \text{London} \\
\text{PlaceOfDeath}.\text{WilliamHogarth} \Rightarrow \text{London} \\
\text{PlaceOfMarriage}.\text{WilliamHogarth} \Rightarrow \text{Paddington}
\]

*London*
Training intuition

Where did Mozart tupress?

PlaceOfBirth.WolfgangMozart \Rightarrow Salzburg
PlaceOfDeath.WolfgangMozart \Rightarrow Vienna
PlaceOfMarriage.WolfgangMozart \Rightarrow Vienna

Vienna

Where did Hogarth tupress?

PlaceOfBirth.WilliamHogarth \Rightarrow London
PlaceOfDeath.WilliamHogarth \Rightarrow London
PlaceOfMarriage.WilliamHogarth \Rightarrow Paddington

London
Training intuition

Where did Mozart typepress?

PlaceOfBirth.WolfgangMozart ⇒ Salzburg
PlaceOfDeath.WolfgangMozart ⇒ Vienna
PlaceOfMarriage.WolfgangMozart ⇒ Vienna

Vienna

Where did Hogarth typepress?

PlaceOfBirth.WilliamHogarth ⇒ London
PlaceOfDeath.WilliamHogarth ⇒ London
PlaceOfMarriage.WilliamHogarth ⇒ Paddington

London
Two types of errors

Correct

Ranking
Errors
(features)

Coverage
Errors
(grammar)

23
Outline

Review of semantic parsing

Using SEMPRE for Freebase QA

Other applications
Setting up SEMPRE

```
git clone https://github.com/percyliang/sempre

cd sempre

./pull-dependencies core corenlp freebase

make module-classes freebase
```
Freebase players

Entities:

fb:en.barack_obama

Types:

fb:people.person

Properties:

fb:people.person.place_of_birth
Freebase players

Entities:

    fb:en.barack_obama

    : (union fb:people.person fb:biology.animal_owner ...)

Types:

    fb:people.person

    : fb:type.type

Properties:

    fb:people.person.place_of_birth

    : (-¡ fb:location.location fb:people.person)
Running SEMPRE

Browse Freebase:

freebase/scripts/fbshell.rb

Interactive prompt:

./run @mode=simple-freebase -Grammar.inPaths cs224u.grammar
Grammar rules

three plus four hundred

(rule \$Number (\$PHRASE) (NumberFn))

(rule \$Number ($Number plus $Number)
   (lambda x (lambda y (call + (var x) (var y))))))

How a rule works:

- Match RHS to produce input derivations
- Call **semantic function** (SemanticFn) on input derivations to produce zero or more output derivations
SEMPRE components

- **Formula**: logical form (Java program or lambda DCS)
- **Value**: denotation (Java object)
SEMPRE components

- **Formula**: logical form (Java program or lambda DCS)
- **Value**: denotation (Java object)
- **Executor**: maps logical forms to denotations (JavaExecutor or SparqlExecutor)
- **Parser**: maps utterances to logical forms (BeamParser or FloatingParser)
- **Learner**: maps examples to parameters
Creating a simple grammar

[demo]
SEMPRE highlights

• Integrates rule-based and statistical methods

• Agnostic to grammar (CFG, CCG, loose or tight)

• Agnostic to logical form (lambda DCS, lambda calculus, Java, AMR)

• Agnostic to answer (any Java object)

• Grammar: SemanticFn, built on CoreNLP

• Learning: online feature-rich discriminative training with embedded execution
Pointers

Issues/questions:

https://github.com/percyliang/sempre/issues
Pointers

Issues/questions:

https://github.com/percyliang/sempre/issues

Internal repository on NLP machines (ask Percy for permissions):

git clone jamie:/user/psl/git/semparse.git

Internal mailing list:

stanford-sempre@googlegroups.com
Outline

Review of semantic parsing

Using SEMPRE for Freebase QA

Other applications
There is a room with a chair and a computer.
Solving LSAT logic puzzles

Exactly six of seven jugglers—G, H, K, L, N, P, and Q—are each assigned to exactly one of three positions—front, middle, and rear—on one of two teams—team 1 and team 2.

For each team, exactly one juggler must be assigned to each position according to the following conditions:

• If either G or H or both are assigned to teams, they are assigned to front positions.

• …
Solving LSAT logic puzzles

Exactly six of seven jugglers—G, H, K, L, N, P, and Q—are each assigned to exactly one of three positions—front, middle, and rear—on one of two teams—team 1 and team 2.

For each team, exactly one juggler must be assigned to each position according to the following conditions:

- If either G or H or both are assigned to teams, they are assigned to front positions.
- ...

Which one of the following is an acceptable list of assignments of jugglers to team 2?

- front: Q; middle: K; rear: N
- ...

[with Robin Jia (Lev/MacCartney/Manning/Levy's dataset)]
In what city did Piotr’s last 1st place finish occur?
### Compositionality on web tables

How many times has this competitor placed 5th or better in competition?

[with Ice Pasupat; ACL 2015]

<table>
<thead>
<tr>
<th>Year</th>
<th>Competition</th>
<th>Venue</th>
<th>Position</th>
<th>Event</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2001</td>
<td>World Youth Championships</td>
<td>Debrecen, Hungary</td>
<td>2nd</td>
<td>400 m</td>
<td>47.12</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1st</td>
<td>Medley relay</td>
<td>1:50.46</td>
</tr>
<tr>
<td>2003</td>
<td>European Junior Championships</td>
<td>Tampere, Finland</td>
<td>3rd</td>
<td>4x400 m relay</td>
<td>3:06.12</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2nd</td>
<td>4x400 m relay</td>
<td>3:08.62</td>
</tr>
<tr>
<td>2005</td>
<td>European U23 Championships</td>
<td>Erfurt, Germany</td>
<td>11th (sf)</td>
<td>400 m</td>
<td>46.62</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1st</td>
<td>4x400 m relay</td>
<td>3:04.41</td>
</tr>
<tr>
<td></td>
<td>Universiade</td>
<td>Izmir, Turkey</td>
<td>7th</td>
<td>400 m</td>
<td>46.89</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1st</td>
<td>4x400 m relay</td>
<td>3:02.57</td>
</tr>
<tr>
<td>2006</td>
<td>World Indoor Championships</td>
<td>Moscow, Russia</td>
<td>2nd (h)</td>
<td>4x400 m relay</td>
<td>3:06.10</td>
</tr>
<tr>
<td></td>
<td>European Championships</td>
<td>Gothenburg, Sweden</td>
<td>3rd</td>
<td>4x400 m relay</td>
<td>3:01.73</td>
</tr>
<tr>
<td>2007</td>
<td>European Indoor Championships</td>
<td>Birmingham, UK</td>
<td>3rd</td>
<td>4x400 m relay</td>
<td>3:08.14</td>
</tr>
<tr>
<td></td>
<td>Universiade</td>
<td>Bangkok, Thailand</td>
<td>7th</td>
<td>400 m</td>
<td>46.85</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1st</td>
<td>4x400 m relay</td>
<td>3:02.05</td>
</tr>
<tr>
<td>2008</td>
<td>World Indoor Championships</td>
<td>Valencia, Spain</td>
<td>4th</td>
<td>4x400 m relay</td>
<td>3:08.76</td>
</tr>
<tr>
<td></td>
<td>Olympic Games</td>
<td>Beijing, China</td>
<td>7th</td>
<td>4x400 m relay</td>
<td>3:00.32</td>
</tr>
<tr>
<td>2009</td>
<td>Universiade</td>
<td>Belgrade, Serbia</td>
<td>2nd</td>
<td>4x400 m relay</td>
<td>3:05.69</td>
</tr>
</tbody>
</table>

**How many times has this competitor placed 5th or better in competition?**
Context-dependent semantic parsing

abc ijk xyz

*add an ”s” to the end of the first group*

abcs ijk xyz

*add another to the end of the second*

abcs ijks xyz

*and the third*

abcs ijks xyzs
Interpreting high-level instructions

**Text:** “get the cup, fill it with water and then microwave the cup”

- `moveto(cup_3)`
- `grasp(cup_3)`
- `moveto(sink_1)`
- `keep(cup_3, on, sink_1)`
- `turn(sink_knob_1)`
- `wait(5min)`
- `turn(sink_knob_1)`

- `moveto(microwave_1)`
- `open(microwave_door_1)`
- `keep(cup_3, in, microwave_1)`
- `close(microwave_door_1)`
- `press(microwave_button_1)`

Unseen verb “fill” is grounded at test time using environment.

“get me a bowl of chips”

“microwave a cup of ramen”

Lexicon Λ from training
Agenda-based semantic parsing

what city was abraham lincoln born in

AbeLincoln
LincolnTown

PlaceOfBirthOf.AbeLincoln
Type.Loc
ContainedBy.LincolnTown

with Jonathan Berant, in submission
Agenda-based semantic parsing

what city was abraham lincoln born in

Type.City △ PlaceOfBirthOf.AbeLincoln
Type.Loc △ ContainedBy.LincolnTown

AbrahamProphet AbeLincoln

PlaceOfBirthOf

PlacesLived

ContainedBy StarredIn

Type.City ⊓ PlaceOfBirthOf.AbeLincoln
Type.Loc ⊓ ContainedBy.LincolnTown

with Jonathan Berant, in submission
Agenda-based semantic parsing

what city was abraham lincoln born in

20
AbeLincoln
LincolnTown
. . ... ⊓ ContainedBy.LincolnTown
. . .
s1 s2
a1
s3
a2 . . .
a3
sT+1
aT
[with Jonathan Berant, in submission]
Agenda-based semantic parsing

what city was abraham lincoln born in

Type.City ⊓ PlaceOfBirthOf.AbeLincoln
Type.Loc ⊓ ContainedBy.LincolnTown

Type.City

Type.Loc

AbeLincoln

LincolnTown

PlaceOfBirthOf

PlacesLived

StarredIn

>1M

362

20

391

508

s1 s2

a1

s3

a2

a3

aT

sT+1

[with Jonathan Berant, in submission]
Agenda-based semantic parsing

Learn which derivations to try first ⇒ 8x speedup
Overnight semantic parsing
 Overnight semantic parsing

Domain

Seed lexicon
article → TypeNP[article]
publication date → RelNP[publicationDate]
cites → VP/NP[cites]
...

(1) by builder (∼30 minutes)
Overnight semantic parsing

Domain

Seed lexicon

article → TypeNP[article]
publication date → RelNP[publicationDate]
cites → VP/NP[cites]
...

Logical forms and canonical utterances

article with the largest publication date
arg max(type.article, publicationDate)

person that is author of the most number of article
arg max(type.person, R[\lambda x.\text{Count}(type.article \cap author.x)])
...

(1) by builder (~30 minutes)

(2) via domain-general grammar
Overnight semantic parsing

Domain

Seed lexicon
article \rightarrow \text{TypeNP}[article]
publication date \rightarrow \text{RelNP}[publicationDate]
cites \rightarrow \text{VP/NP}[cites]
...

Logical forms and canonical utterances

article with the largest publication date
arg \max(\text{type.article}, \text{publicationDate})

person that is author of the most number of article
arg \max(\text{type.person}, R[\lambda x. \text{Count}(\text{type.article} \sqcap \text{author}.x)])
...

Paraphrases

what is the newest published article?
who has published the most articles?
...
Overnight semantic parsing

Domain

Seed lexicon

- article → TypeNP[article]
- publication date → RelNP[publicationDate]
- cites → VP/NP[cites]

Logical forms and canonical utterances

- article with the largest publication date
  \[ \text{arg max}(\text{type.article, publicationDate}) \]
- person that is author of the most number of articles
  \[ \text{arg max}(\text{type.person, R[\lambda x. Count(type.article \sqcap \text{author}.x)])} \]

Paraphrases

- what is the newest published article?
- who has published the most articles?

Semantic parser

(1) by builder (∼30 minutes)

(2) via domain-general grammar

(3) via crowdsourcing (∼5 hours)

(4) via domain-general paraphrasing model
Answering macro questions

Which country has the highest CO2 emissions?

Which had the highest increase since last year?

What fraction is from the five countries with highest GDP?

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>World</td>
<td>World</td>
<td>30,455.23</td>
<td>50,358.42</td>
<td>4.49</td>
<td>-0.3</td>
</tr>
<tr>
<td>179</td>
<td>Asia &amp; Oceania</td>
<td>China</td>
<td>12,338.41</td>
<td>13,264.09</td>
<td>3.53</td>
<td>7.5</td>
</tr>
<tr>
<td>188</td>
<td>China</td>
<td>China</td>
<td>6,803.92</td>
<td>7,716.80</td>
<td>5.83</td>
<td>13.3</td>
</tr>
<tr>
<td>1</td>
<td>North America</td>
<td>United States</td>
<td>6,885.07</td>
<td>6,410.54</td>
<td>14.10</td>
<td>-6.9</td>
</tr>
<tr>
<td>7</td>
<td>United States</td>
<td>United States</td>
<td>5,833.13</td>
<td>5,424.53</td>
<td>17.87</td>
<td>-7</td>
</tr>
<tr>
<td>54</td>
<td>Europe</td>
<td>Europe</td>
<td>4,828.98</td>
<td>4,310.30</td>
<td>7.14</td>
<td>-6.9</td>
</tr>
<tr>
<td>91</td>
<td>Eurasia</td>
<td>Eurasia</td>
<td>2,560.86</td>
<td>2,566.03</td>
<td>8.22</td>
<td>-9.2</td>
</tr>
<tr>
<td>107</td>
<td>Middle East</td>
<td>Middle East</td>
<td>1,508.55</td>
<td>1,714.09</td>
<td>7.22</td>
<td>3.3</td>
</tr>
<tr>
<td>194</td>
<td>India</td>
<td>India</td>
<td>1,473.73</td>
<td>1,802.12</td>
<td>1.38</td>
<td>8.7</td>
</tr>
<tr>
<td>102</td>
<td>Russia</td>
<td>Russia</td>
<td>1,568.38</td>
<td>1,572.07</td>
<td>11.23</td>
<td>-7.4</td>
</tr>
<tr>
<td>8</td>
<td>Central &amp; South</td>
<td>Central &amp; South</td>
<td>1,226.85</td>
<td>1,219.78</td>
<td>2.57</td>
<td>0.7</td>
</tr>
</tbody>
</table>
Which restaurants have my friends been to in the last week?

Which restaurants will still be open Sunday at 10pm?

On Friday night, leave the front light on.
Code and data

http://www-nlp.stanford.edu/software/sempre/
http://www.codalab.org

Collaborators

Jonathan Berant
Andrew Chou
Roy Frostig
Ice Pasupat
Yushi Wang
Robin Jia
Reggy Long

Funding

Google
Microsoft
DARPA

Thank you!