Learning Structured Natural Language Representations for Semantic Parsing

Jianpeng Cheng, Siva Reddy, Vijay Saraswat and Mirella Lapata

31\textsuperscript{th} July, 2017
School of Informatics, University of Edinburgh
Introduction
Semantic parsing converts natural language utterances to logical forms, which can be executed to yield a task-specific response.

How many daughters does Obama have?

convert

answer(count(relatives.daughter(Obama)))

execute

answer = 2
Semantic parsing applications:

Question answering (Berant and Liang, 2013)
Task-oriented dialog (Artzi and Zettlemoyer, 2011)
Instructing robots (Matuszek et al., 2012)
Goal of Our Work

- **Improve** neural semantic parsing.
Goal of Our Work

- **Improve** neural semantic parsing.
- **Interpret** neural semantic parsing.
Neural sequence to sequence models convert utterances into logical form strings (Dong and Lapata, 2016; Jia and Liang, 2016).
Limitations of Neural Sequence to Sequence Models

- They generate a sequence of tokens (the output may contain extra or missing brackets).
Limitations of Neural Sequence to Sequence Models

- They generate a sequence of tokens (the output may contain extra or missing brackets).
- They are not type-constrained (the output may be meaningless or ungrammatical).
Our Work

- We use a neural transition system to generate tree-structured logical forms with domain-general constrains;

  \[
  \text{university(Obama)} \rightarrow \text{person.education(Obama)}
  \]
Our Work

- We use a neural transition system to generate tree-structured logical forms with domain-general constraints;
  - Outputs are well-formed (enforced by tree-structured decoder).

  e.g., Which university did Obama go to?
  university(Obama) → person.education(Obama)
Our Work

- We use a neural transition system to generate tree-structured logical forms with domain-general constrains;
  - Outputs are well-formed (enforced by tree-structured decoder).
  - Outputs are meaningful and executable (enforced by domain-general constrains).
Our Work

- We use a neural transition system to generate tree-structured logical forms with domain-general constraints;
  - Outputs are well-formed (enforced by tree-structured decoder).
  - Outputs are meaningful and executable (enforced by domain-general constraints).
- We interpret the neural semantic parser (with hard attention)

  e.g., Which university did Obama go to?

  university(Obama) → person.education(Obama)
Our Work

- We use a neural transition system to generate tree-structured logical forms with domain-general constrains;
  - Outputs are well-formed (enforced by tree-structured decoder).
  - Outputs are meaningful and executable (enforced by domain-general constrains).
- We interpret the neural semantic parser (with hard attention)
  - To predict a token in the logical form, we first predict a natural language token or a domain-general token, which is then grounded to the target domain.
Our Work

• We use a neural transition system to generate tree-structured logical forms with domain-general constrains;
  • Outputs are well-formed (enforced by tree-structured decoder).
  • Outputs are meaningful and executable (enforced by domain-general constrains).

• We interpret the neural semantic parser (with hard attention)
  • To predict a token in the logical form, we first predict a natural language token or a domain-general token, which is then grounded to the target domain.
  • e.g., Which university did obama go to?
    university(Obama) $\rightarrow$ person.education(Obama)
Methodology
• We use the Functional Query Language (FunQL, Zelle 1995) as the semantic formalism.
We use the Functional Query Language (FunQL, Zelle 1995) as the semantic formalism.

FunQL is a recursive, tree-structured representation.
Semantic Formalism

- We use the Functional Query Language (FunQL, Zelle 1995) as the semantic formalism.
- FunQL is a recursive, tree-structured representation.
- We then train a neural model to generate trees.
Examples of FunQL

- **Apply**: relation(entity)
  
  Example: Who is the wife of Obama?
  
  relatives.wife(Obama)

- **Count**: count(relation(entity))

  Example: How many daughters does Obama have?
  
  count(relatives.daughter(Obama))

- **Argmax**: argmax(relation1(entity), relation2)

  Example: Who is Obama's eldest daughter?
  
  argmax(relatives.daughter(Obama), person.age)
Examples of FunQL

- **Apply**: relation(entity)
  
  **Example**: Who is the wife of Obama?
  
  relatives.wife(Obama)

- **Count**: count(relation(entity))
  
  **Example**: How many daughters does Obama have?
  
  count(relatives.daughter(Obama))

- **Argmax**: argmax(relation1(entity), relation2)
  
  **Example**: Who is Obama's eldest daughter?
  
  argmax(relatives.daughter(Obama), person.age)
Examples of FunQL

- **Apply**: relation(entity)
  
  **Example**: Who is the wife of Obama?
  relatives.wife(Obama)

- **Count**: count(relation(entity))
  
  **Example**: How many daughters does Obama have?
  count(relatives.daughter(Obama))

- **Argmax**: argmax(relation1(entity), relation2)
  
  **Example**: Who is Obama’s eldest daughter?
  argmax(relatives.daughter(Obama), person.age)
Examples of FunQL

- **Apply**: `relation(entity)`
  
  **Example**: Who is the wife of Obama?
  `relatives.wife(Obama)`

- **Count**: `count(relation(entity))`
  
  **Example**: How many daughters does Obama have?
  `count(relatives.daughter(Obama))`

- **Argmax**: `argmax(relation1(entity), relation2)`
  
  **Example**: Who is Obama’s eldest daughter?
  `argmax(relatives.daughter(Obama), person.age)`

- **And**: `and(relation1(entity1), relation2(entity2))`
  
  **Example**: Which of Obama’s daughter studied in Harvard?
  `and(daughter(Obama), person.education(Harvard))`
To generate a tree-structured representation, we specify:

- a canonical generation order;
To generate a tree-structured representation, we specify:

- a canonical generation order;
- a model to encode context and generation history.
A Canonical Generation Order for Trees

answer

and

relatives.daughter

person.education

Obama

Harvard
A Canonical Generation Order for Trees

Top-down (Dyer et al. 2016):
A Canonical Generation Order for Trees

Top-down (Dyer et al. 2016):

answer

and

relatives.daughter

Obama

person.education

Harvard
A Canonical Generation Order for Trees

Top-down (Dyer et al. 2016):

answer and relatives.daughter and Obama Harvard

answer and relatives.daughter Obama Harvard

Harvard
A Canonical Generation Order for Trees

Top-down (Dyer et al. 2016):

answer

and

relatives.daughter

Obama

and

relatives.daughter

Obama

person.education

Harvard

Harvard
**A Canonical Generation Order for Trees**

**Top-down** (Dyer et al. 2016):

```
answer

and

relatives.daughter

Obama

person.education

Harvard
```

---

*answer* and *relatives.daughter* and *Obama* and *person.education* and *Harvard*
A Canonical Generation Order for Trees

Top-down (Dyer et al. 2016):
answer and relatives.daughter Obama

and

person.education Harvard
Tree-generation actions:

1. Generate non-terminal node (**NT**);
2. Generate terminal node (**TER**);
3. Complete subtree (**REDUCE**).
Tree-generation actions:

1. Generate non-terminal node (NT);
2. Generate terminal node (TER);
3. Complete subtree (REDUCE).

Combined with functional query language:

- **NT** further includes: NT(count), NT(argmax), NT(argmin), NT(and), · · · , NT(relation)
- **TER** further includes: TER(relation), TER(entity)
We use *explicit* constraints to restrict the space of transition actions at each time step:

- tree-structure constraints ensure logical forms are well-formed;
Constraints

We use \textit{explicit} constraints to restrict the space of transition actions at each time step:

- tree-structure constraints ensure logical forms are well-formed;
- domain-general constraints ensure logical forms follow the grammar of functional query language.
Transition System for Tree Generation

Oracle action sequence

NT(answer)
Oracle action sequence

NT(answer), NT(and)
Transition System for Tree Generation

Oracle action sequence

\text{NT(}\text{answer}\text{)}, \text{NT(}\text{and}\text{)}, \text{NT(}\text{relation}\text{)}
Transition System for Tree Generation

Answer

And

Relatives.daughter

Person.education

Obama

Harvard

Oracle action sequence

NT(answer), NT(and), NT(relation), TER(entity)
Transition System for Tree Generation

Oracle action sequence

- NT(answer)
- NT(and)
- NT(relation)
- TER(entity)
- REDUCE
Transition System for Tree Generation

Oracle action sequence

<table>
<thead>
<tr>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>NT(answer), NT(and), NT(relation), TER(entity), REDUCE, <strong>NT(relation)</strong></td>
</tr>
</tbody>
</table>
Transition System for Tree Generation

Oracle action sequence

NT(answer), NT(and), NT(relation), TER(entity), REDUCE, NT(relation), TER(entity)
Transition System for Tree Generation

Oracle action sequence

NT(answer), NT(and), NT(relation), TER(entity), REDUCE, NT(relation), TER(entity), REDUCE
Transition System for Tree Generation

answer

and

relatives.daughter

Obama

person.education

Harvard

Oracle action sequence

NT(answer), NT(and), NT(relation), TER(entity), REDUCE, NT(relation), TER(entity), REDUCE, REDUCE
Transition System for Tree Generation

Oracle action sequence

\[ \text{NT(answer), NT(and), NT(relation), TER(entity), REDUCE, NT(relation), TER(entity), REDUCE, REDUCE, REDUCE} \]
A Model to Encode the Context

Encode sentential context with bidirectional LSTMs and generation history with stack-LSTM

- The next action is NT(relation)
- The next logical form token is person.education
To predict the next transition action, we use **soft attention** to combine the generation history and sentential context.

**soft attention:**
1. \( rep\_buffer = \text{attention}(stack[-1], buffer[1:-1]) \)
2. \( rep\_stack = stack[-1] \)
3. \( rep\_system = \text{MLP}(rep\_buffer, rep\_stack) \)
4. \( output\_action = \text{softmax}(rep\_system) \)
Predicting the Next Logical Form Token

When NT(relation), TER(relation) or TER(entity) are triggered, a token needs to be predicted. We use the interpretable **hard attention**: predict a natural language token and then map it to the target domain.

**hard attention:**
1. **copy** a natural language predicate from the sentence: e.g., copy the predicate *studied*
2. **map** the natural language predicate into the domain-specific predicate: e.g., map *studied* to *person.education*
Training
Example: Which of Obama’s daughter studied in Harvard?
Output: answer(and(daughter(Obama),
person.education(Harvard)))

We can maximize the likelihood of the gold-standard logical forms and do back-propagation.
Example: Which of Obama’s daughter studied in Harvard?
Output: Malia Obama

- Infer surrogate logical forms (i.e., which give correct denotation) from the denotation and use those to back-propagate.
Example: Which of Obama’s daughter studied in Harvard?

Output: Malia Obama

- Infer surrogate logical forms (i.e., which give correct denotation) from the denotation and use those to back-propagate.
- This is achieved with a rule-based system and beam-search.
Example: Which of Obama’s daughter studied in Harvard?
Output: Malia Obama

- Infer surrogate logical forms (i.e., which give correct denotation) from the denotation and use those to back-propagate.
- This is achieved with a rule-based system and beam-search.
- In this set up, we integrate the neural semantic parser with a discriminative ranker.
• The neural model generates a list of candidate logical forms by beam-search.
Discriminative Ranking

- The neural model generates a list of candidate logical forms by beam-search.
- We train a log-linear model (Berant et al., 2013) to rank the candidate logical forms.

<table>
<thead>
<tr>
<th>input query</th>
<th>Which of Obama’s daughter studied in Havard</th>
</tr>
</thead>
<tbody>
<tr>
<td>candidate logical forms</td>
<td></td>
</tr>
<tr>
<td>answer(person.education(Havard))</td>
<td></td>
</tr>
<tr>
<td>answer(daughter(Obama))</td>
<td></td>
</tr>
<tr>
<td>answer(and(daughter(Obama), person.education(Havard)))</td>
<td></td>
</tr>
<tr>
<td>answer(person.employment(Havard))</td>
<td></td>
</tr>
<tr>
<td>answer(count(daughter(Obama)))</td>
<td></td>
</tr>
<tr>
<td>......</td>
<td></td>
</tr>
</tbody>
</table>

gives correct denotation
Experiments
Experimental Set-up

- Training with **utterance-logical form pairs**
  GEOQUERY (Zelle and Mooney, 1996) of **880** examples

- Training with **utterance-denotation pairs**
  WEBQUESTIONS (Berant et al., 2013) of **5,810** examples,
  GRAPHQUESTIONS (Su et al., 2016) of **5,166** examples

- Training with **distant supervision**
  SPADES (Bisk et al., 2016) of **93,319** examples
Experiments on GeoQuery (Zelle and Mooney, 1996)

- PCCG induction (Kwiatkowski et al., 2013): 88.0
- LambdaDCS (Liang et al., 2011): 91.1
- Neural seq2seq (Dong and Lapata, 2016): 84.6
- Neural seq2seq (Jia and Liang, 2016): 85.0
- This work: 86.7
Experiments on WebQuestions (Berant et al., 2013)

- Sempre (Berant et al., 2014) 35.7
- IE (Yao and Van Durme, 2014) 33.0
- Imitation learning (Berant and Liang, 2015) 49.7
- Dep2lambda (Reddy et al., 2016) 50.3
- Seq2seq (our implementation) 48.3
- WikiQA (Xu et al., 2016) 53.3
- This work 49.4
Experiments on GraphQuestions (Su et al., 2016)

- Sempre (Berant et al., 2013) 10.8
- Paraphrase (Berant and Liang, 2014) 12.79
- Jacanna (Yao and Van Durme, 2014) 5.08
- Seq2seq (our implementation) 16.24
- This work 17.02

1 Sempre (Berant et al., 2013)
2 Paraphrase (Berant and Liang, 2014)
3 Jacanna (Yao and Van Durme, 2014)
• We conduct additional experiments on SPADES (Bisk et al., 2016) dataset.
• We inspect the predicate-argument structures induced by hard attention.
• The induced structures start rivaling linguists structures with increase in training data.
Task description: given a declarative sentence and an entity masked by blank. Find the most informative predicate that is useful to predicate that entity.

e.g., Boeing was founded in 1916 and is headquartered in _blank_.

Linguists vs Machines
Agreed cases

Boeing was founded in 1916 and is headquartered in _blank_.

_blank_ has confirmed to play captain Haddock.

Mathematica is a product of _blank_.

_blank_ is a corporation that is owned by the Edmonton city
Agreed cases

Boeing was founded in 1916 and is headquartered in _blank_.

_blank_ has confirmed to play captain Haddock.

Mathematica is a product of _blank_.

_blank_ is a corporation that is owned by the Edmonton city
Disagreed cases

Eddie dumped Debbie to marry _blank_.

Wilhelm Maybach and his son _blank_ started Maybach.

Romney is the worst governor that has _blank_ ever had.
Linguists vs Machines

Disagreed cases

Eddie dumped Debbie to marry __blank__.
Disagreed cases

Eddie dumped Debbie to marry _blank_.

Wilhelm Maybach and his son _blank_ started Maybach.
Disagreed cases

Eddie dumped Debbie to marry _blank_.

Wilhelm Maybach and his son _blank_ started Maybach.
Disagreed cases

Eddie dumped Debbie to marry _blank_.

Wilhelm Maybach and his son _blank_ started Maybach.

Romney is the worst governor that has _blank_ ever had.
Disagreed cases

Eddie dumped Debbie to marry _blank_.

Wilhelm Maybach and his son _blank_ started Maybach.

Romney is the worst governor that has _blank_ ever had.
Conclusion
Summary of This Work

- We improve neural semantic parsers with a constrained neural transition system to generate well-formed and meaningful logical forms.
- We inspect natural language structures discovered by neural semantic parsers and find they differ from linguistic-based ones.
- Our semantic parser is available at https://github.com/cheng6076/scanner.