Interactive Semantic Parsing for If-Then Recipes via Hierarchical Reinforcement Learning

Ziyu Yao

The Ohio State University

In collaboration with: Xiujun Li (MSR, UW), Jianfeng Gao (MSR), Brian Sadler (ARL), Huan Sun (OSU)

Outline

- Background
- Interactive Semantic Parser
  - Why
  - How
- Experiments
- Conclusion

Outline

- Background
- Interactive Semantic Parser
  - Why
  - How
- Experiments
- Conclusion

Semantic Parsing

- General task
  - To map natural language to formal domain-specific meaning representations.

Semantic Parsing

- General task
  - To map natural language to formal domain-specific meaning representations.
- Example
  - Knowledge based question answering
    - NL Question => Logical form in lambda-DCS (or, SPARQL/SQL query)

"Find people who died from lung cancer before 1960 and whose parent died for the same reason"

$$
\begin{aligned}
\lambda x.\exists y.\exists z.&\mathtt{type}(x, \mathtt{DeceasedPerson}) \\
&\wedge \mathtt{type}(y, \mathtt{DeceasedPerson}) \\
&\wedge \mathtt{type}(z, \mathtt{Datetime}) \wedge \mathtt{parents}(x, y) \\
&\wedge \mathtt{causeOfDeath}(x, \mathtt{LungCancer}) \\
&\wedge \mathtt{causeOfDeath}(y, \mathtt{LungCancer}) \\
&\wedge \mathtt{dateOfDeath}(x, z) \wedge < (z, 1960).
\end{aligned}
$$

(Su et al., 2016)

Semantic Parsing

- General task
  - To map natural language to formal domain-specific meaning representations.
- Example
  - General-purpose program synthesis
    - NL question => Python program

"how to sort my_list in descending order in python?"

```
sorted(my_list, reverse=True)
```

Semantic Parsing for If-Then Recipes

- If-Then program: A conditional statement
  - Informally known as "If *this*, then *that*"
  - *Whenever the conditions of the trigger (i.e., "this") are satisfied, the action (i.e., "that") is performed*
  - e.g., "Turn on my lights when I arrive home" (home automation), "tell me if the door opens" (home security), etc.

Semantic Parsing for If-Then Recipes

- If-Then program: A conditional statement
  - Informally known as "If *this*, then *that*"

- Providing services that allow end users to connect and integrate their web applications

Semantic Parsing for If-Then Recipes

- If-Then program: A conditional statement
  - Informally known as "If *this*, then *that*"
- Formally, an If-Then recipe:
  - A natural language description
  - 4 components in the program
    - Trigger channel
    - Trigger function
    - Action channel
    - Action function

Semantic Parsing for If-Then Recipes

- Example
  - NL description

    "*Create a link note on Evernote for my liked tweets*"

  - If-Then program
    - Trigger channel: *Twitter*
    - Trigger function: *New liked tweet by you*
    - Action channel: *Evernote*
    - Action function: *Create a link note*

Outline

- Background
- Interactive Semantic Parser
  - Why
  - How
- Experiments
- Conclusion

Previous Work

- Semantic parsing *in one shot:*
  - User gives an NL description, and system responds with a program
  - (Quirk et al., 2015; Liu et al., 2016; Dong and Lapata, 2016)

Challenges

- Natural language descriptions can be ambiguous, and contain incomplete information
- Example:
  - NL description: "*record to evernote*"
  - Ground truth: [Twitter(*trigger channel*), New liked tweet by you (*trigger function*), Evernote (*action channel*), Create a link note (*action function*)]

Challenges

- Natural language descriptions can be ambiguous, and contain incomplete information
- Example:
  - NL description: "*record to evernote*"
  - Ground truth: [Twitter(*trigger channel*), New liked tweet by you (*trigger function*), Evernote (*action channel*), Create a link note (*action function*)]
  - Other possible interpretations: [Instagram, You like a photo, Evernote, Create a note], …

Challenges

- Natural language descriptions can be ambiguous, and contain incomplete information
- In the widely used dataset (Quirk et al., 2015), 80% of ~4K human evaluated descriptions are considered ambiguous to some degree.

Challenges

- Natural language descriptions can be ambiguous, and contain incomplete information

- Quite difficult for an automated parser to produce a correct program, *if only based on an ambiguous description*.

Interactive Semantic Parsing

- An intelligent agent can ask user questions for clarification to improve parsing accuracy.

User: "record to evernote"
HRL agent: "Which event triggers the action?"
User: "If I like a tweet"
HRL agent: "Which event results from the trigger?"
User: "Create a note with link"
Agent Prediction: [tc: `Twitter`, tf: `New liked tweet by you`, ac: `Evernote`, af: `Create a link note`]

Interactive Semantic Parsing

- An intelligent agent can ask user questions for clarification to improve parsing accuracy.

- Goal
  - Improve parsing accuracy, but with as few questions as possible.

Outline

- Background
- Interactive Semantic Parser
  - Why
  - How
- Experiments
- Conclusion

Interactive Semantic Parsing

- Challenges
  - Lack of supervision on when system should ask a question
    - The only feedback is whether a synthesized program is correct or not.

  - How to optimize the parsing accuracy and number of asks at the same time?

Previous Rule-based Agents

- For each component, build a classifier model

- If the prediction of the current component is *lower than a threshold*, ask user a question.

e.g., P(Trigger channel = *Instagram*) = 0.3 < 0.4 (threshold), ask user a question* like “which channel to trigger?”

*Question is formulated using templates. (Chaurasia and Mooney, 2017)

Our Formulation

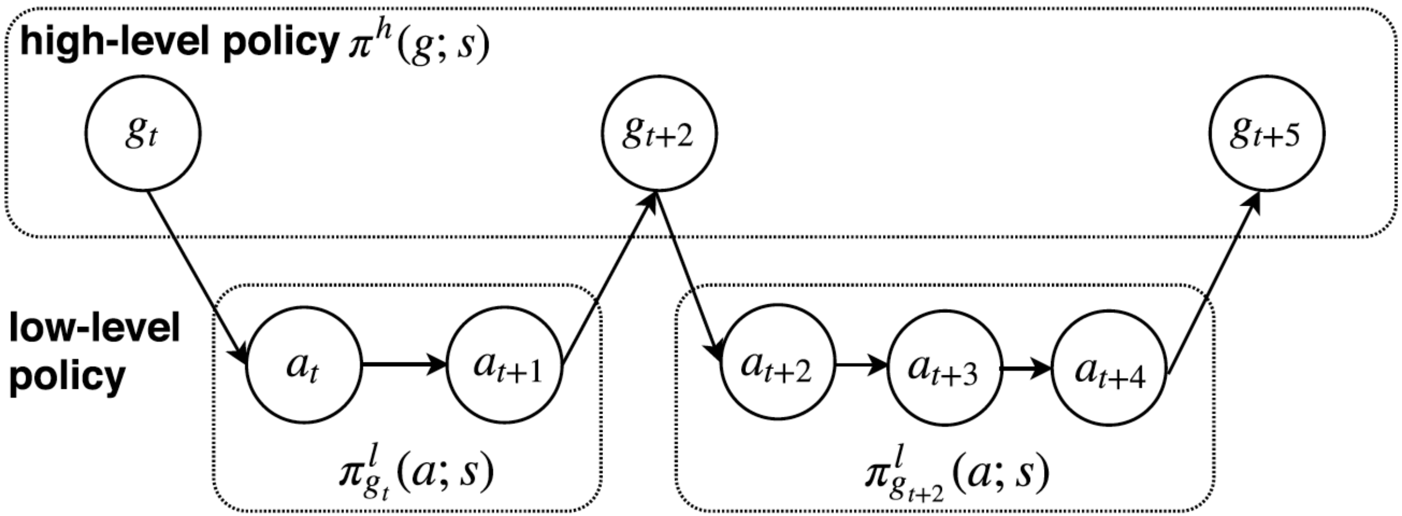
- Treat predicting the 4 components as 4 subtasks

- Hierarchical decision making process
  - At the high level, decide which subtask to work on

  - At the low level, for the current selected subtask, decide whether to make a prediction or to ask user a question, based on the current status

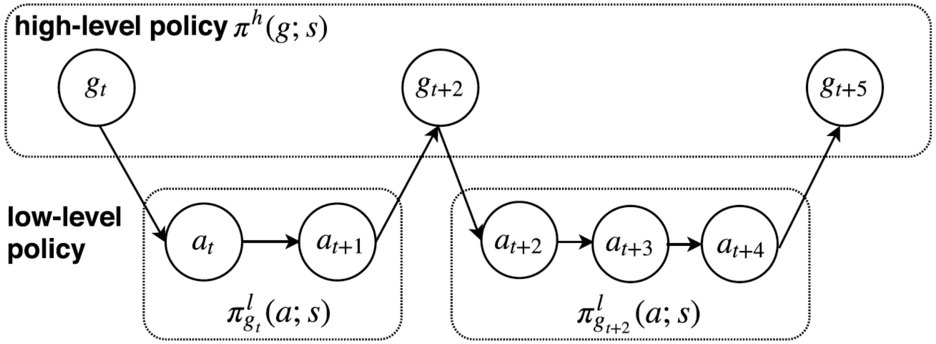
Hierarchical Decision Making

- In a Hierarchical Reinforcement Learning framework (Sutton et al., 1999)



s: state information.
g_t: the subtask to work on from time step t
a_t: the low-level action at time step t when working on subtask g_t.

Actions



- High-level action space
  - 4 subtasks
  - Each representing predicting one component, e.g., trigger channel
- Low-level action space
  - For each component selected at the high level, e.g., trigger channel,

    {all possible trigger channels} U {AskUser}

States

- A state s consists of 9 items:
  - The initial recipe description I
  - The boolean indicator b_i $(i = 1\sim4)$ for each subtask, showing whether each subtask has been predicted or not
  - The received user answer d_i $(i = 1\sim4)$ for each subtask

Rewards

- Low level

$$r^l_{g_t}(s_t, a_t) = \begin{cases} 1 & \text{if } a_t = \ell_{g_t} \\ -\beta & \text{if } a_t = \text{AskUser} \\ -1 & \text{otherwise} \end{cases}$$

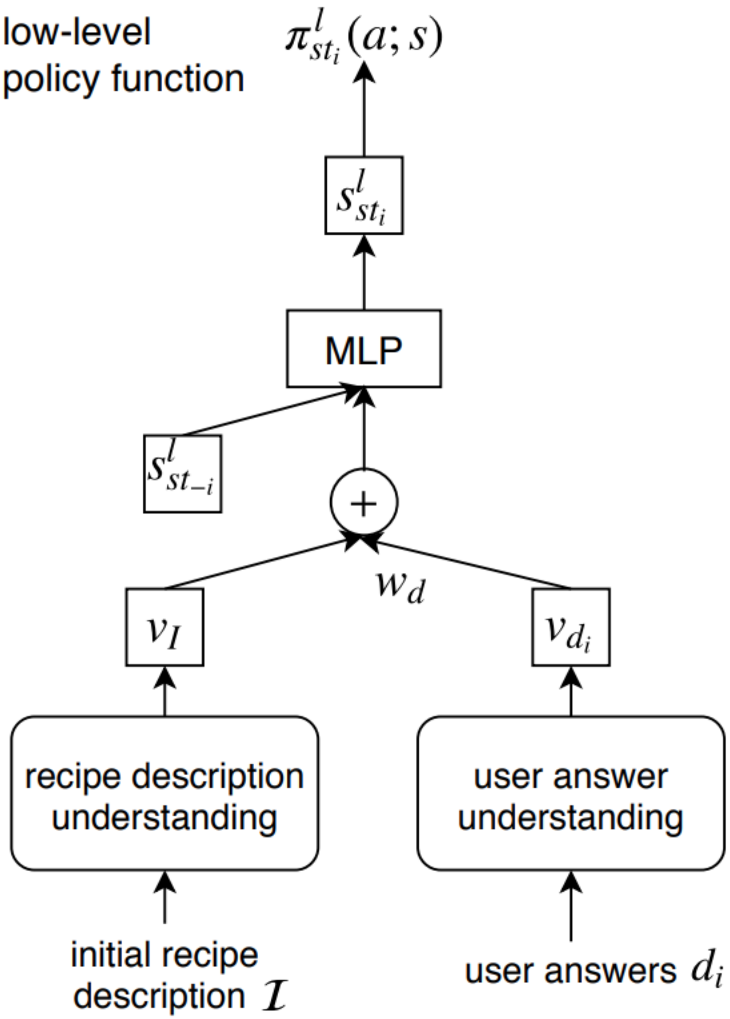
* ℓ_{g_t}:: ground-truth label for subtask
$\beta \epsilon [0,1)$: penalty for asking the user

- High level

$$r^h(s_t, g_t) = \begin{cases} \sum_{k=t}^{t+N} r^l_{g_t}(s_k, a_k) & \text{for eligible } t \\ 0 & \text{otherwise} \end{cases}$$

*eligible t: at the beginning of a subtask or when a subtask terminates

Low-level Policy Function Design



The low-level policy function **for subtask st_i**:

- $v_i = (1 - w_d)v_I + w_d v_{d_i}$ represents the information integrated from both the recipe description and the user answer, traded off by weight w_d.

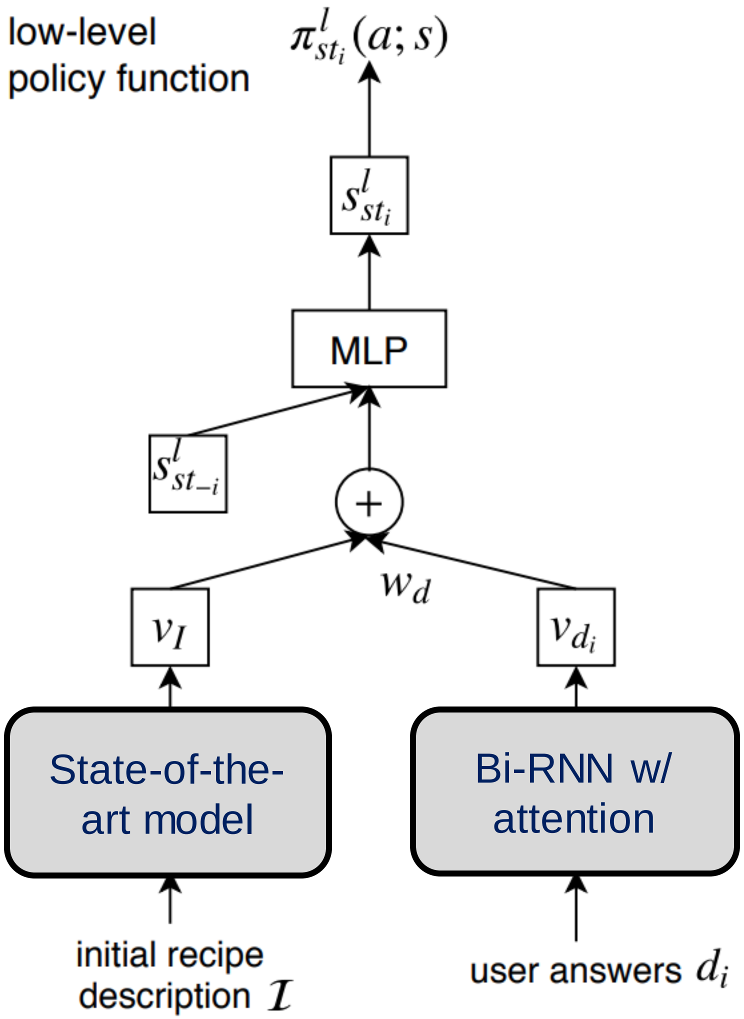
- $s^l_{st_i}$: the *low-level* state vector of subtask st_i, i.e.,

$$s^l_{st_i} = \tanh(W_{c_i}[s^l_{st_1}; \ldots; s^l_{st_{i-1}}; v_i; s^l_{st_{i+1}}; \ldots; s^l_{st_4}])$$

- Low-level policy value (probability distribution over action space):

$$\pi^l_{st_i}(a;s) = softmax(W^l_{st_i} s^l_{st_i})$$

Low-level Policy Function Design



The low-level policy function **for subtask st_i**:

- $v_i = (1 - w_d)v_I + w_d v_{d_i}$ represents the information integrated from both the recipe description and the user answer, traded off by weight w_d.

- *Low-level* state vector of subtask st_i:

$$s^l_{st_i} = \tanh(W_{c_i}[s^l_{st_1}; \ldots; s^l_{st_{i-1}}; v_i; s^l_{st_{i+1}}; \ldots; s^l_{st_4}])$$

- Low-level policy value (probability distribution over action space):

$$\pi^l_{st_i}(a;s) = softmax(W^l_{st_i} s^l_{st_i})$$

High-level Policy Function Design

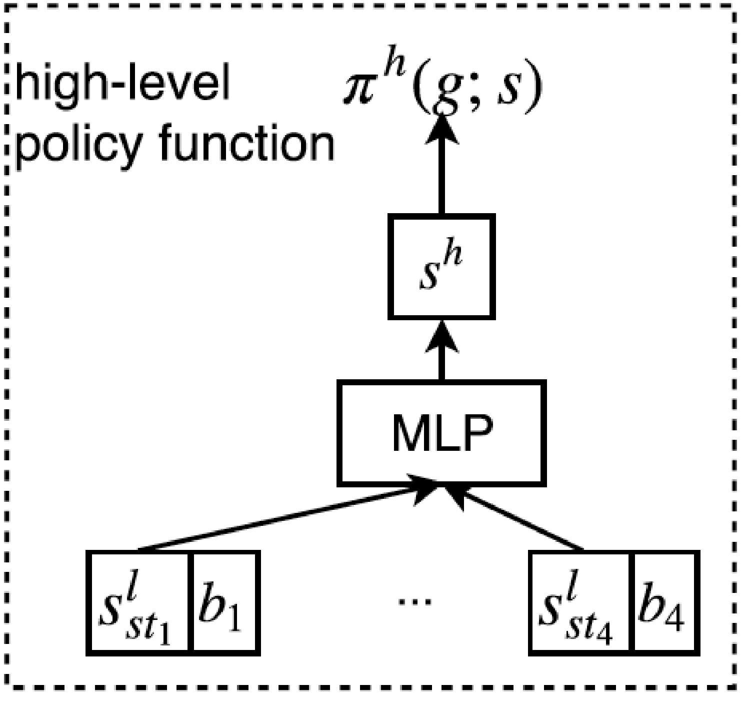
High-level policy decides which subtask to work on:

- $s^l_{st_i}$: the state vector for subtask st_i $(i = 1\sim4)$
- b_i: a boolean value indicating whether subtask st_i is completed
- *High-level* state vector:

$$s^h = \tanh(W_c[s^l_{st_1}; b_1; \dots; s^l_{st_4}; b_4])$$

- Policy value (probability distribution over 4 subtasks):

$$\pi^h(g; s) = softmax(W^h s^h)$$



Hierarchical Policy Learning

- Learned by the REINFORCE algorithm (Williams, 1992)
- For each policy, perform gradient ascent to maximize the future rewards

User Simulator

- Why user simulator is needed?
  - Save real human efforts in training

- Simulating user answers when the agent asks clarification questions about channels and functions.

User Simulator

- Simulating user answers for channels by *channel names*, e.g., “Gmail”.
- Simulating user answers for functions by:
  - Revised function name / definition from IFTTT.com and their paraphrases
    - e.g., “This Trigger fires <u>every time you like a tweet</u>”
  - Extractions from user data
    - e.g., extracting X from recipe description “If X then Y” as a user answer when asked about the corresponding trigger function

Outline

- Background
- Interactive Semantic Parser
  - Why
  - How
- Experiments
- Conclusion

Experiments: Dataset

- Training and validation
  - 291,285 pairs of <NL description, If-Then program> (Ur et al., 2016)
- Testing
  - 3,870 pairs (Quirk et al., 2015)
  - Each description manually annotated by 5 AMTurkers

Test Data	CI	VI		Total
		VI-1/2	VI-3/4	
Size (%)	727 (18.79)	1,271 (32.84)	1,872 (48.37)	3,870 (100)

CI: "clear description"
VI: "vague description"

Experiments: Methods to Compare

- LAM: Latent Attention Model (Liu et al., 2016); one of the state-of-the-art If-Then parsing models in one shot
  - One classifier for each of 4 components
- LAM-rule
- LAM-sup
- HRL (our model)
- HRL-fixedOrder (fixing the high-level subtask order)

Experiments: Methods to Compare

- LAM: Latent Attention Model (Liu et al., 2016); one of the state-of-the-art If-Then parsing models in one shot
  - One classifier for each of 4 components

- LAM-rule
  - By running the trained LAM.
  - Prediction probability < threshold (0.85) ⇨ Ask.
  - Concatenating the received user answer with the recipe description as input for the next time step.

Experiments: Methods to Compare

- LAM: Latent Attention Model (Liu et al., 2016); one of the state-of-the-art If-Then parsing models in one shot
  - One classifier for each of 4 components

- LAM-rule
- LAM-sup
  - LAM with "user answer understanding" module
    - Input: recipe description, user answer (if any).
    - Output: *predict* "AskUser" for asking questions, or predict the channel/function value.

Experiments: Methods to Compare

- LAM: Latent Attention Model (Liu et al., 2016); one of the state-of-the-art If-Then parsing models in one shot
  - One classifier for each of 4 components

- LAM-rule
- LAM-sup
  - Synthesized training data based on the performance of LAM-rule
    - e.g., completing *with* asking users:
      - <recipe description, Ø> "AskUser"
      - <recipe description, *received user answer*> true label

Experiments: Methods to Compare

- LAM: Latent Attention Model (Liu et al., 2016); one of the state-of-the-art If-Then parsing models in one shot
  - One classifier for each of 4 components

- LAM-rule
- LAM-sup

- HRL (our model)
- HRL-fixedOrder (fixing the high-level subtask order)

Simulation Evaluation on Test Set

Model	CI		VI-1/2		VI-3/4	
	C+F Acc	#Asks	C+F Acc	#Asks	C+F Acc	#Asks
LAM	0.801	0	0.436	0	0.166	0
LAM-rule	0.897	1.433	0.743	2.826	0.721	5.568
LAM-sup	0.894	**0.684**	0.803	**1.482**	0.780	2.921
HRL-fixedOrder	**0.950**	1.522	0.855	1.958	0.871	2.777
HRL	0.949	1.226*	**0.888***	1.748*	**0.878***	**2.615***

- Simulation Evaluation: User answers are sampled from the *simulated* answer pool.
- C+F Accuracy: when all the 4 subtasks get correct predictions.
- #Asks: averaged number of questions for completing the entire task.
- * denotes significant different in mean between HRL vs. HRL-fixedOrder.

Simulation Evaluation on Test Set

Model	CI		VI-1/2		VI-3/4	
	C+F Acc	#Asks	C+F Acc	#Asks	C+F Acc	#Asks
LAM	0.801	0	0.436	0	0.166	0
LAM-rule	0.897	1.433	0.743	2.826	0.721	5.568
LAM-sup	0.894	**0.684**	0.803	**1.482**	0.780	2.921
HRL-fixedOrder	**0.950**	1.522	0.855	1.958	0.871	2.777
HRL	0.949	1.226*	**0.888***	1.748*	**0.878***	**2.615***

1. All interactive agents perform better than the non-interactive LAM.
2. LAM-rule simply asks redundant questions.
3. HRL-based agents outperform other agents by:
   - 5% on CI, 8%~15% on VI (taking up *80%* of the dataset).
   - Reasonable/minimal number of questions.
4. HRL demands significantly less questions to humans.

Human Evaluation on VI-3/4

- The most challenging VI-3/4 dataset
- Two volunteer students familiar with IFTTT
- Each session:
  - One If-Then recipe sampled from VI-3/4
    - with official descriptions of each component
  - One agent sampled from {LAM-rule, LAM-sup, HRL, HRL-fixedOrder}
    - Unknown to the participant
- The participant is encouraged to answer in their own words when being asked
  - For better user experience: Each agent is limited to ask at most 1 question for each component

Human Evaluation on VI-3/4

- In total, collected 496 conversations
- Note:
  - LAM's result is based on the 496 recipes
  - * denote significant in mean between HRL-based agents and {LAM-rule, LAM-sup}

Model	C+F Acc	#Asks
LAM	0.206	0
LAM-rule	0.518	2.781
LAM-sup	0.433	2.614
HRL-fixedOrder	0.581	2.306*
HRL	**0.634***	**2.221***

Human Evaluation on VI-3/4

1. All agents' performance is not as good as in Simulation Evaluation
   - Mainly due to the high language complexity in real user answers
2. The two HRL-based agents outperform LAM-rule/sup by 6%~20% Acc, with *fewer* questions
3. HRL vs. HRL-fixedOrder: better Acc and fewer #Asks

Model	C+F Acc	#Asks
LAM	0.206	0
LAM-rule	0.518	2.781
LAM-sup	0.433	2.614
HRL-fixedOrder	0.581	2.306*
HRL	**0.634***	**2.221***

Outline

- Background
- Interactive Semantic Parser
  - Why
  - How
- Experiments
- Conclusion

Conclusion

- Formulated interactive semantic parsing for If-Then recipes with HRL
- Improved parsing accuracy without asking user many questions
- Generalizable to other semantic parsing tasks (beyond If-Then recipes) with human-machine interaction/collaboration

Acknowledgement

Thanks! Questions?