State Tracking and Policy Learning in Task-Oriented Dialog Systems

Xin Li[1,2]

[1]NLP Center,
Tencent AI Lab

[2]Department of System Engineering & Engineering Management,
The Chinese University of Hong Kong

October 11, 2018

Overview

1. Task-Oriented Dialogue System
   - System Overview and Task Introduction
2. Neural Belief Tracker: Data-Driven Dialogue State Tracking (ACL 17)
   - Details
3. Global-Locally Self-Attentive Dialogue State Tracker (ACL 18)
   - Details
4. Deep Dyna-Q: Integrating Planning for Task-Completion Dialogue Policy Learning (ACL 18)
   - Details
5. Adversarial Actor-Critic Model For Task-Completion Dialogue Policy Learning (ICASSP 18)
   - Details

Outline

System Pipeline

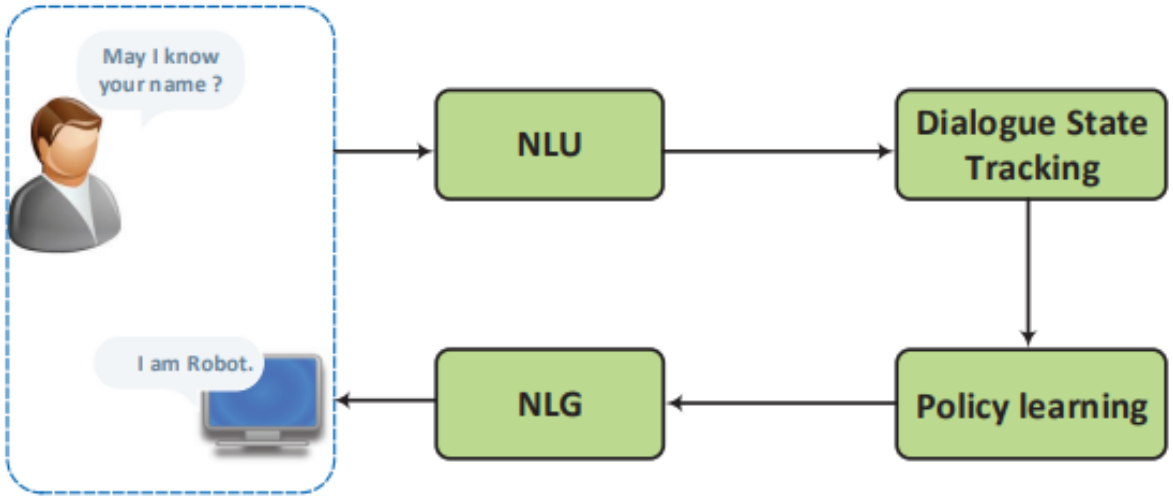


Figure: System Pipeline

Natural Language Understanding (NLU)

- Extract semantic slots and intents from the user utterances.
- Usually involve slot filling, intent prediction and domain classification tasks.

Sentence	show	restaurant	at	New	York	tomorrow
Slots	O	O	O	B-desti	I-desti	B-date
Intent	Find Restaurant					
Domain	Order					

Annotations	
Intent	request, inform, deny, confirm_question, confirm_answer, greeting, closing, not_sure, multiple_choice, thanks, welcome
Slot	city, closing, date, distanceconstraints, greeting, moviename, numberofpeople, price, starttime, state, taskcomplete, theater, theater_chain, ticket, video_format, zip

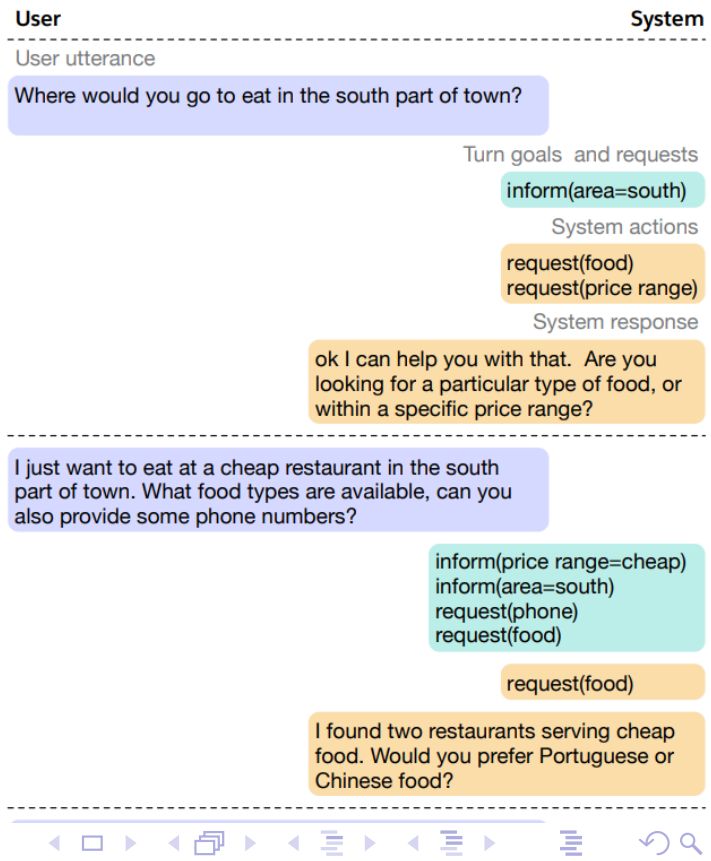
Figure: Example Data and Labels

Dialog State Tracking (DST)

- Detect the **goals** and the **requests** of the user from the extracted slots.
- Detect the **way** the user is trying to interact with the system.

Slot	User may give as a constraint?
area	Yes, 5 possible values
food	Yes, 91 possible values
name	Yes, 113 possible values
pricerange	Yes, 3 possible values
addr	No
phone	No
postcode	No
signature	No

Figure: Pre-defined slots in DSTC2 (Restaurant Domain)



Dialog State Tracking (DST)

1. A Mechanical Turker transcribes the user recordings in a dialog, giving the following dialog transcription:
   (a) (Hello welcome [...] how can I help?)
   "A cheap restaurant"
   (b) (Did you say expensive?)
   "Um yep okay"
   (c) (Okay, Oak Bistro is a nice restaurant in the expensive pricerange.)
   "What is the phone number?"
2. The Phoenix grammar gives the following results for the user dialog acts:
   (a) inform(pricerange=cheap)
   (b) null()
   (c) request(phone)
3. The manual correction of the dialog acts fixes the second turn from null() to affirm().
4. The deterministic tracker is run on the corrected semantics in conjunction with the system's actions to produce the following dialog state sequence:
   (a) goals: {pricerange=cheap}; requested-slots:{}, method: byconstraints
   (b) goals: {pricerange=expensive}; requested-slots:{}, method: byconstraints
   (c) goals: {pricerange=expensive, name=Oak Bistro}; requested-slots:{phone}, method: byname

Figure: Dialog State Sequence Example

Dialog State Tracking (DST)

- Dialog actions are formed as a list of dictionaries with `act` and `slots` fields.

- `[{"act":"hello", "slots":[]},`
  `{"act":"inform", "slots":[["pricerange","cheap"]]}]`
  "Hello, I want something cheap."
- `[{"act":"negate", "slots":[]},`
  `{"act":"deny", "slots":[["area","north"]]}`
  "No I don't want it to be in the north."
- `[{"act":"thankyou", "slots":[]},`
  `{"act":"bye", "slots":[]}`
  "Thank you good bye."
- `[{"act":"ack", "slots":[]},`
  `{"act":"request", "slots":[["slot", "phone"]]}`
  "Okay, what's the phone number of that place?"
- `[{"act":"inform", "slots":[["pricerange", "dontcare"]]}`
  "In any pricerange."
- `[{"act":"inform", "slots":[["this", "dontcare"]]},`
  `{"act":"inform", "slots":[["food", "japanese"]]}`
  "I don't mind, but it should serve japanese food."
- `[{"act":"confirm", "slots":[["area", "centre"]]},`
  `{"act":"request", "slots":[["slot", "hastv"]]}`
  "Is it in the centre, and does it have television?"
- `[{"act":"reqalts", "slots":[]},`

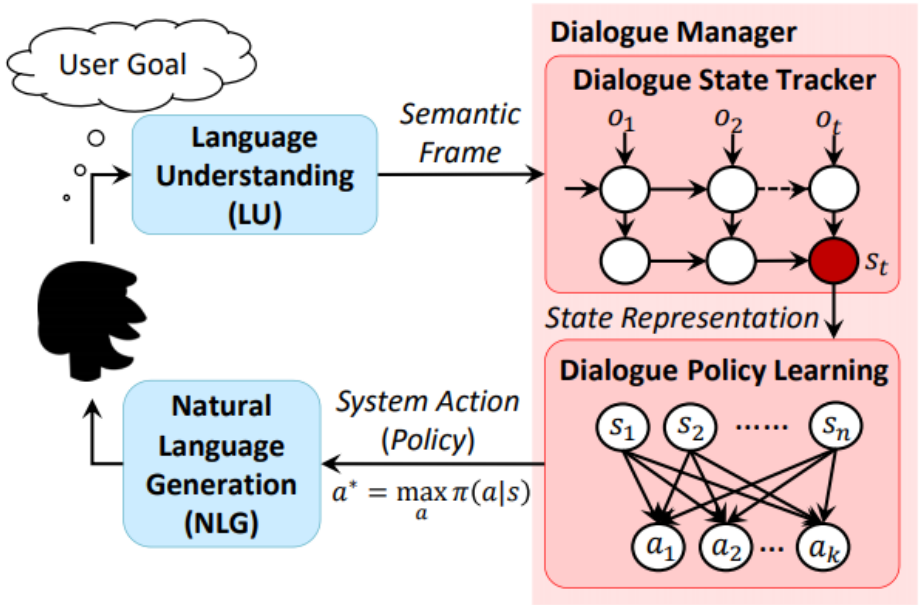
Figure: Example User Actions

- `request(food)`
  (JSON version: `[{"act":"request", "slots":[["slot","food"]]}]`)
  "What kind of food would you like"
- `inform(count=109), impl-conf(food=dontcare), request(pricerange)`
  "There are 109 restaurants if you don't care about the food. What pricerange would you like?"
- `canthelp(food=korean, pricerange=moderate)`
  "I'm sorry but there is no restaurant serving moderate korean food"
- `canthelp(food=spanish, pricerange=moderate), canthelp.exception(name="la tasca")`
  "I am sorry but la tasca is the only spanish restaurant in the moderate price range"
- `offer(name="yippee noodle bar"), inform(pricerange=moderate)`
  "Yippee noodle bar is in the moderate price range"
- `select(pricerange=cheap), select(pricerange=moderate)`
  "Sorry would you like something in the cheap price range or in the moderate price range"
- `offer(name="the lucky star"), inform(phone="01223 244277"), inform(addr="cambridge leisure park clifton way cherry hinton")`
  "The phone number of the lucky star is 01223 244277 and it is on Cambridge Leisure Park Clifton Way Cherry Hinton"
- `repeat()`
  "Sorry I am a bit confused, please tell me again what you are looking for"
- `expl-confirm(area=centre)`
  "Let me confirm. You are looking for a venue in the central area"

Figure: Example System Actions

Dialogue Policy Learning

- Accept dialogue state from DST and generate the next available system action.
  - Often formulated as a reinforcement learning problem.



Natural Language Generation

- The NLG module is to generate natural language responses.
- Template-based NLG: Define a set of rules to map semantic frame (i.e., system/user actions) to natural language.
- Model-based NLG: Generate sentence sketch with slot placeholders via an RNN decoder. Post-processing is needed.
- Hybrid Approach: Template-based + Model-based NLG.

Semantic Frame	Natural Language
confirm()	"Please tell me more about the product your are looking for."
confirm(area=$V)	"Do you want somewhere in the $V?"
confirm(food=$V)	"Do you want a $V restaurant?"
confirm(food=$V,area=$W)	"Do you want a $V restaurant in the $W."

Figure: Template-Based NLG

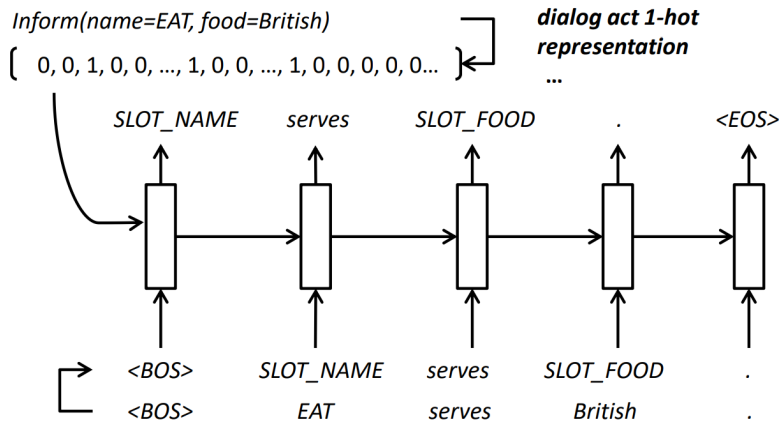


Figure: Model-Based NLG

Outline

Background

- Separate NLU approaches.
  - Fine-grained manual annotation at the word-level is required, hindering scaling to larger, more realistic application domains
- Joint NLU/DST approaches.
  - Those delexicalization-based approaches highly rely on the semantic dictionaries, restricting their applications on the rich variety of user language or to general domains.

FOOD=CHEAP: [affordable, budget, low-cost, low-priced, inexpensive, cheaper, economic, ...]

RATING=HIGH: [best, high-rated, highly rated, top-rated, cool, chic, popular, trendy, ...]

AREA=CENTRE: [center, downtown, central, city centre, midtown, town centre, ...]

Figure: An example semantic dictionary.

The Proposed Model

- The proposed Neural Belief Tracker:
  - Jointly handle the NLU and the DST task to free from the error propagation and the lack of the annotated data.
  - Select the slot-value pairs via semantic matching in the word embedding space rather than delexicalization.

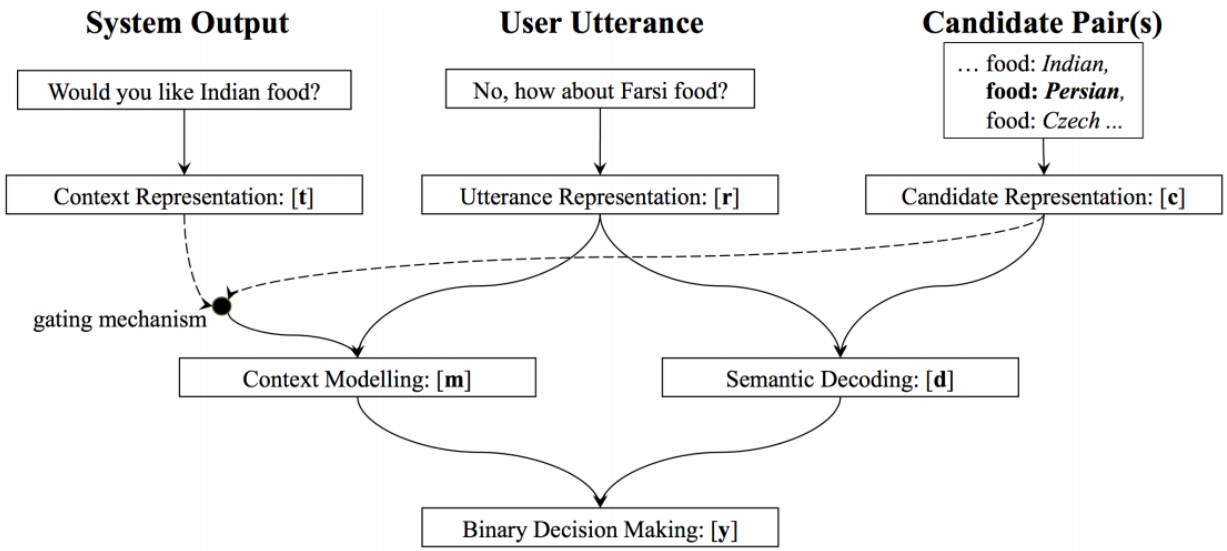


Figure: The architecture of the Neural Belief Tracker

The proposed Model

- Semantic Decoding:

$$\mathbf{c} = \sigma\left(W_c^s(\mathbf{c_s} + \mathbf{c_v}) + b_c^s\right) \tag{7}$$

$$\mathbf{d} = \mathbf{r} \otimes \mathbf{c} \tag{8}$$

- Context Modelling:

$$\mathbf{m_r} = (\mathbf{c_s} \cdot \mathbf{t_q})\mathbf{r} \tag{9}$$

$$\mathbf{m_c} = (\mathbf{c_s} \cdot \mathbf{t_s})(\mathbf{c_v} \cdot \mathbf{t_v})\mathbf{r} \tag{10}$$

Experiment

- Dataset: DSTC-2 and Woz.
- Evaluation Metrics:
  - Joint Goal Accuracy: Proportion of the dialogue turns where the user goals (search constraints) are correctly identified
  - Request Accuracy: Proportion of the dialogue turns where the requested slots are correctly identified.
- Main Results

DST Model	DSTC2		WOZ 2.0	
	Goals	Requests	Goals	Requests
Delexicalisation-Based Model	69.1	95.7	70.8	87.1
Delexicalisation-Based Model + Semantic Dictionary	72.9*	95.7	83.7*	87.6
NEURAL BELIEF TRACKER: NBT-DNN	72.6*	96.4	**84.4***	91.2*
NEURAL BELIEF TRACKER: NBT-CNN	**73.4***	**96.5**	84.2*	**91.6***

Figure: Experimental Results

Outline

Background

- Current approaches do not address the problem that extracting rare slot-value pairs.
  - Most slot-value pairs composing the state rarely occur in the training set.

The proposed approach

- The proposed Global-Locally Self-Attentive Dialogue State Tracker
  - Share parameters between the estimators for each slot.
  - Introduce local modules to learn the slot-specific feature representations.

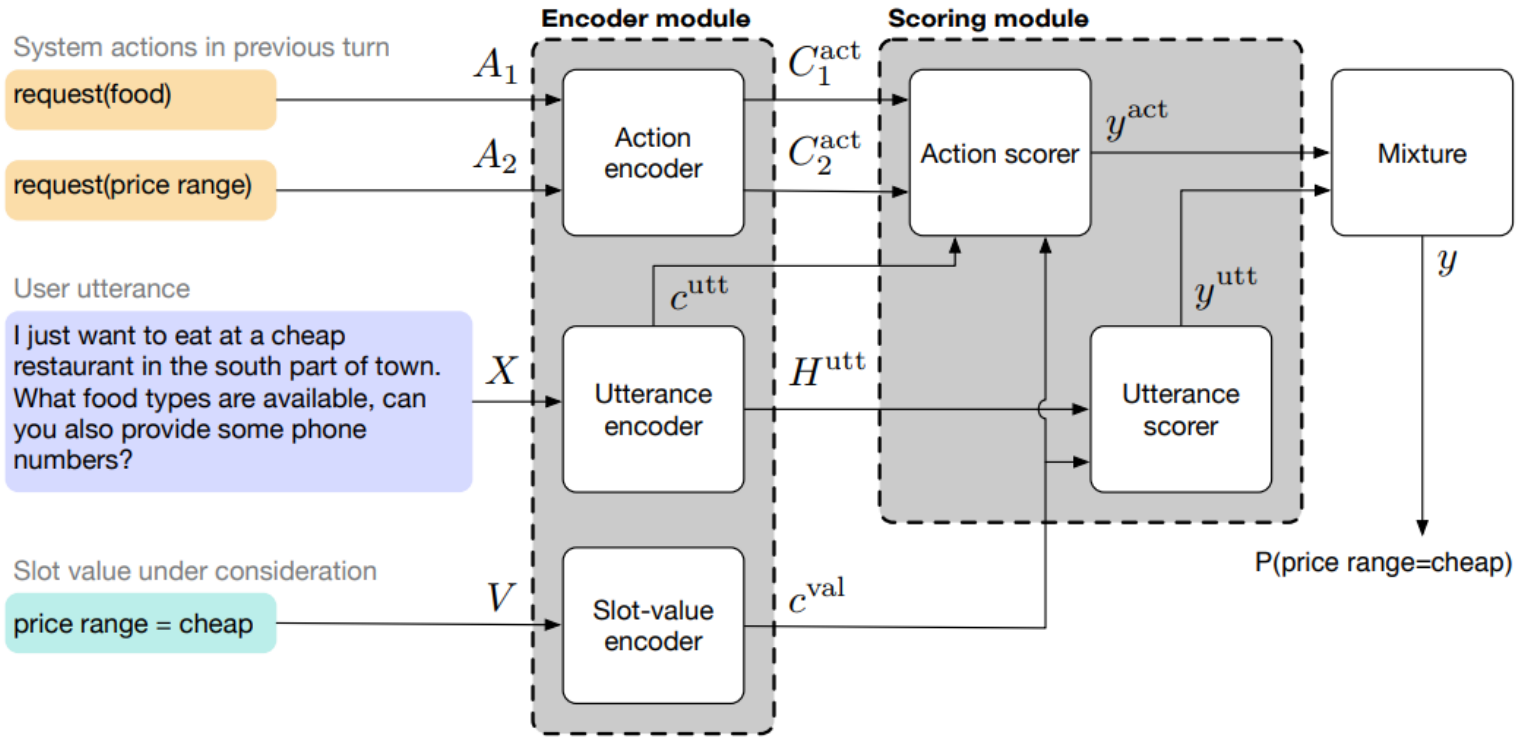
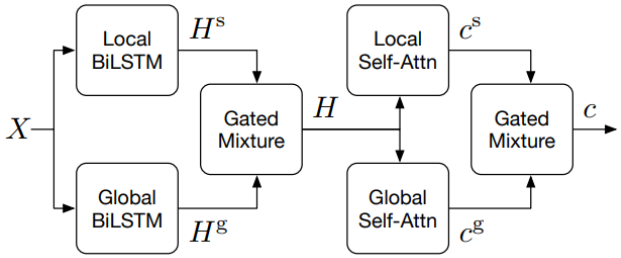


Figure: The architecture of the GLAD.

The proposed approach

- Global-Locally self-Attentive Encoder:



$$H^{\text{utt}}, c^{\text{utt}} = \text{encode}(U) \quad (12)$$
$$H_j^{\text{act}}, C_j^{\text{act}} = \text{encode}(A_j) \quad (13)$$
$$H^{\text{val}}, c^{\text{val}} = \text{encode}(V) \quad (14)$$

- Scoring Module:

$$a_i^{\text{utt}} = \left(H_i^{\text{utt}}\right)^{\intercal} c^{\text{val}} \in \mathbb{R} \quad (15)$$
$$p^{\text{utt}} = \text{softmax}\left(a^{\text{utt}}\right) \in \mathbb{R}^m \quad (16)$$
$$q^{\text{utt}} = \sum_i p_i^{\text{utt}} H_i^{\text{utt}} \in \mathbb{R}^{d_{\text{rnn}}} \quad (17)$$
$$y^{\text{utt}} = Wq^{\text{utt}} + b \in \mathbb{R} \quad (18)$$

$$a_j^{\text{act}} = \left(C_j^{\text{act}}\right)^{\intercal} c^{\text{utt}} \in \mathbb{R} \quad (19)$$
$$p^{\text{act}} = \text{softmax}\left(a^{\text{act}}\right) \in \mathbb{R}^{l+1} \quad (20)$$
$$q^{\text{act}} = \sum_j p_j^{\text{act}} C_j^{\text{act}} \in \mathbb{R}^{d_{\text{rnn}}} \quad (21)$$
$$y^{\text{act}} = \left(q^{\text{act}}\right)^{\intercal} c^{\text{val}} \in \mathbb{R} \quad (22)$$

Experimental Results

- Main Results

Model	DSTC2		WoZ	
	Joint goal	Turn request	Joint goal	Turn request
Delexicalisation-Based Model	69.1%	95.7%	70.8%	87.1%
Delex. Model + Semantic Dictionary	72.9%	95.7%	83.7%	87.6%
Neural Belief Tracker (NBT) - DNN	72.6%	96.4%	84.4%	91.2%
Neural Belief Tracker (NBT) - CNN	73.4%	96.5%	84.2%	91.6%
GLAD	**74.5± 0.2%**	**97.5± 0.1%**	**88.1± 0.4%**	**97.1± 0.2%**

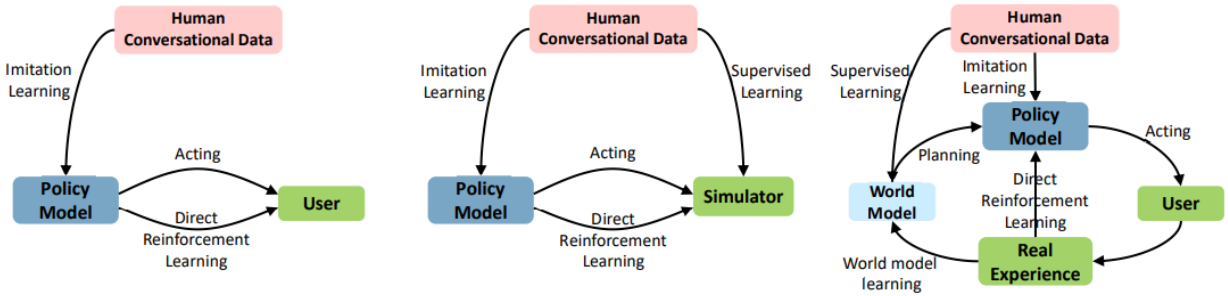
- Case Study

System actions in previous turn	User utterance	Predicted turn belief state
N/A	I would like Polynesian food in the South part of town. Please send me phone number and address.	request(phone) request(address) inform(food=polynesian) inform(area=south)
request(address) request(phone) There is a moderately priced italian place called Pizza hut at cherry hilton. would you like the address and phone number?	Yes please.	request(phone) request(address)
request(food) request(price range) ok I can help you with that. Are you looking for a particular type of food, or within a specific price range?	I just want to eat at a cheap restaurant in the south part of town. What food types are available, can you also provide some phone numbers?	request(phone) inform(price range=cheap) inform(area=south) -inform(food=dontcare) +request(food)

Outline

Background

- Policy learning strategies:

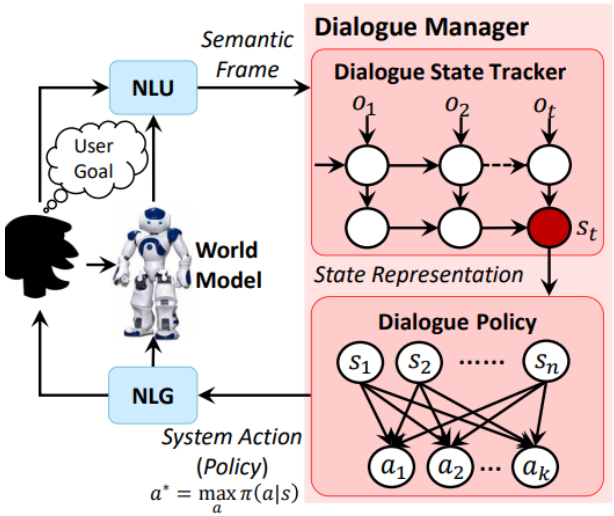


(a) Learning with real users (b) Learning with user simulators (c) Learning with real users via DDQ

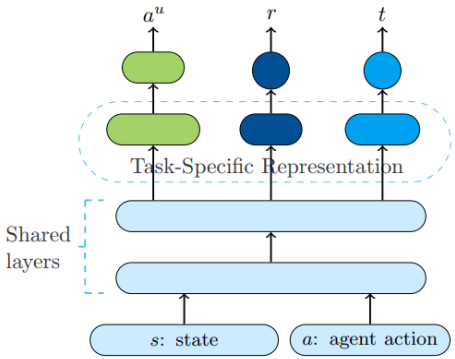
- Traditional Approaches:
  - Interact with real users. (Too expensive!)
  - Interact with user-simulator. (Lack the conversational complexity)
- The proposed Deep Dyna-Q Model:
  - Introduce a *world model* to simulate the environment ((dialog state, system action)⇒user action).
  - Policy is improved by direct RL (DQN) and indirect RL (planning).

Deep Dyna-Q model

- Architecture



- World Model



- Q-Networks: MLP with tanh activation.

$$\begin{aligned} \mathcal{L}(\theta_Q) &= \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}^u}[(y_i - Q(s,a;\theta_Q))^2] \\ y_i &= r + \gamma \max_{a'} Q'(s',a';\theta_{Q'}) \end{aligned} \quad (1)$$

- System action is selected via ϵ-greedy strategy.
- World Model:

$$\begin{aligned} h &= \tanh(W_h(s,a) + b_h) \\ r &= W_r h + b_r \\ a^u &= \text{softmax}(W_a h + b_a) \\ t &= \text{sigmoid}(W_t h + b_t) \end{aligned}$$

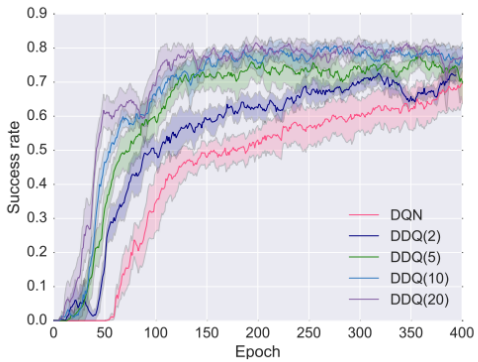
- Work flow: 1-step direct RL, 1-step word-model learning and K-step planning.

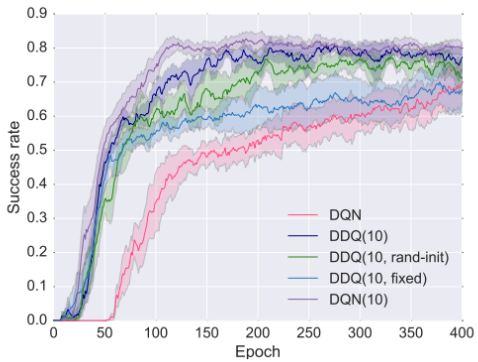
Experiment

- Main Results:

Agent	Epoch = 100			Epoch = 200			Epoch = 300		
	Success	Reward	Turns	Success	Reward	Turns	Success	Reward	Turns
DQN	.4260	-3.84	31.93	.5308	10.78	22.72	.6480	27.66	22.21
DDQ(5)	.6056	20.35	26.65	.7128	36.76	19.55	.7372	39.97	18.99
DDQ(5, rand-init θ_M)	.5904	18.75	26.21	.6888	33.47	20.36	.7032	36.06	18.64
DDQ(5, fixed θ_M)	.5540	14.54	25.89	.6660	29.72	22.39	.6860	33.58	19.49
DQN(5)	*.6560*	*29.38*	*21.76*	*.7344*	*41.09*	*16.07*	*.7576*	*43.97*	*15.88*
DDQ(10)	**.6624**	28.18	24.62	**.7664**	42.46	21.01	**.7840**	45.11	19.94
DDQ(10, rand-init θ_M)	.6132	21.50	26.16	.6864	32.43	21.86	.7628	42.37	20.32
DDQ(10, fixed θ_M)	.5884	18.41	26.41	.6196	24.17	22.36	.6412	26.70	22.49
DQN(10)	*.7944*	*48.61*	*15.43*	*.8296*	*54.00*	*13.09*	*.8356*	*54.89*	*12.77*

- Impact of K and *world model*:





Example Output

	DQN agent	
1	*usr*: Hi, what date can I book 2 tickets for deadpool?	*usr*: Hi, which theater will show batman? I want 1 ticket.
2	*agt*: Which city would you like?	*agt*: Which city would you like?
3	*usr*: I want to watch at seattle.	*usr*: Boston.
4	*agt*: What time would you like to see it?	*agt*: What time would you like to see it?
5	*usr*: What start time is available?	*usr*: What start time is available?
6	*agt*: Which theater would you like?	*agt*: What date would you like to see it?
7	*usr*: Which theater is available?	*usr*: This weekend.
8	*agt*: What time would you like to see it?	*agt*:Which theater would you like?
9	*usr*: What start time is available?	*usr*: Which theater is available?
10	*agt*: Which theater would you like?	*agt*: Which theater would you like?
11	*usr*: Which theater is available?	*usr*: Which theater is available?
12	*agt*: How many tickets do you need?	*agt*: What time would you like to see it?
13	*usr*: I want 2 tickets please!	*usr*: What start time is available?
14	*agt*: Which theater would you like?	*agt*: Which theater would you like?
15	*usr*: Which theater is available?	*usr*: Thanks, I do not need the ticket.
16	*agt*: What time would you like to see it?	*agt*: Thank you.
	⋮	
39	*agt*: Which theater would you like?	
40	*usr*: Which theater is available?	
Failure:	Agent does not book the right ticket within the max turns.	**Failure:** User terminates the dialogue before reaching the max turns because the agent repeatly asks the same question.
	DDQ(10) agent	
1	*usr*: Hi, when is deadpool playing at seattle?	*usr*: Hi, which theater will show batman? I want 1 ticket.
2	*agt*: Tonight.	*agt*: Century Eastport 16.
3	*usr*: What is the start time?	*usr*: What is the start time?
4	*agt*: Which theater would you like?	*agt*: 9PM.
5	*usr*: I want to watch it at amc pacific place 11.	*usr*: Can I buy a ticket here?
6	*agt*: 10pm is available.	*agt*: This weekend?
7	*usr*: I want 2 tickets please!	*usr*: Yes, one ticket please!
8	*agt*: Great, here are 2 deadpool tickets for you at amc pacific place 11 in seattle tonight 10 pm.	*agt*: Sure, here it is - batman at Century Eastport 16, 9PM this weekend.
9	*usr*: Thank you.	*usr*: Thank you.
10	*agt*: Thank you.	*agt*: Thank you.
	Success	**Success**

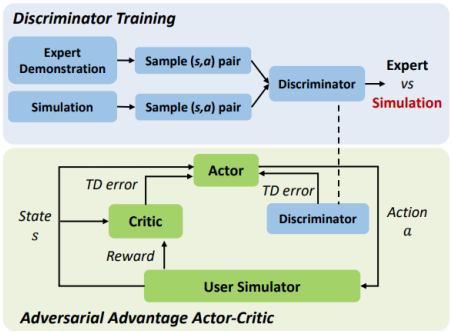
Outline

Motivation

- Handle the issue of *reward sparsity*–the majority of the experiences (i.e., the tuples of state, action and reward) have small impact on the agent.
- Solutions:
  - Leveraging the prior knowledge learned from human-human dialogues.
  - Introducing additional intrinsic rewards.

The Proposed Approach

- Architecture:



- Algorithm:

Algorithm 1 Adversarial Advantage Actor-Critic Model

Input: Expert demonstrations $Demo$, initialize actor π, discriminator D and two value functions $V^{\pi_\theta}(s)$, $V^{\pi_\theta}_{\text{GAN}}(s)$
for i=1:N **do**
  Restart the dialogue simulator, get state representation s, initialize transition tuple *buffer* = []
  while s is not a terminal state **do**
    Perform the action a_t according to the actor $\pi(a_t \mid s; \theta)$
    Receive the reward r_t and switch to a new state s'
    Store (s, a_t, r_t, s') to the transition tuple *buffer*
    $s := s'$
  end while
  Train the actor π with gradients (6)
  Train value function $V^{\pi_\theta}(s)$ by minimizing the TD error (5)
  Sample state action pairs (s, a) from expert demonstration $Demo$
  Train the actor π with gradients (9)
  Update the reward with $-\log(1 - D(s, a))$ and train value function $V^{\pi_\theta}_{\text{GAN}}(s)$ by minimizing the TD error (10)
  Update the discriminator parameters (8)
end for

- Actor: generate actions (I-layer FC with pre-training).
- Critic: give score on the selected action. (Extrinsic Rewards)
- Discriminator: distinguish the generated and the real experiences. (Intrinsic Rewards)

$$\delta^{\pi_\theta} = r + \gamma V^{\pi_\theta}(s') - V^{\pi_\theta}(s), \tag{5}$$

$$\nabla_\theta J(\theta) = \mathbb{E}[\nabla_\theta \log \pi_\theta(a \mid s) \delta^{\pi_\theta}]. \tag{6}$$

$$\nabla_\theta J(\theta) = \mathbb{E}[\nabla_\theta \log \pi_\theta(a \mid s) A^{\pi_\theta}_{\text{GAN}}(s, a)]. \tag{9}$$

$$\delta^{\pi_\theta}_{\text{GAN}} = r_{\text{GAN}} + \gamma V^{\pi_\theta}_{\text{GAN}}(s') - V^{\pi_\theta}_{\text{GAN}}(s). \tag{10}$$

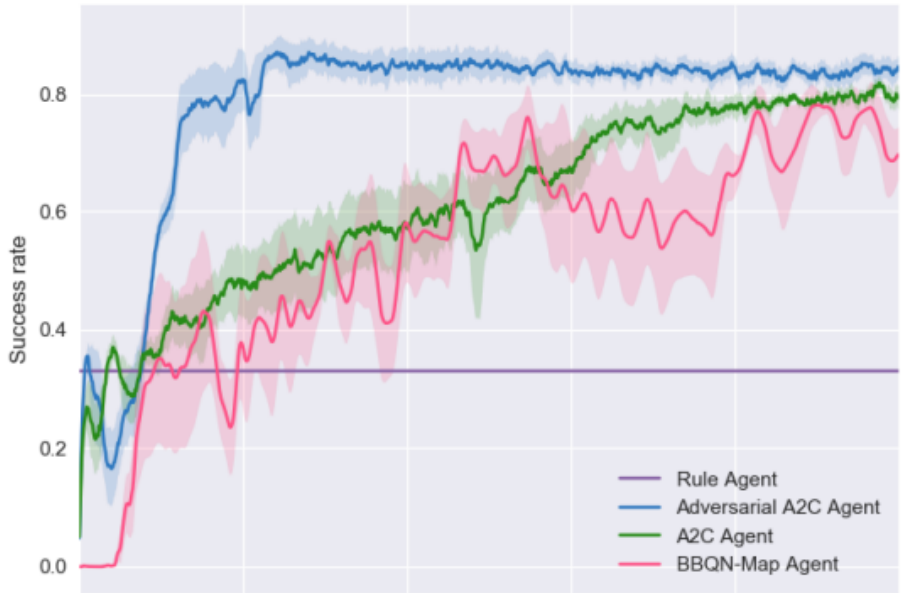
$$\min_{\theta_D} \mathcal{L}_D = -\mathbb{E}_{(s,a)\sim Simu} \log D(s, a; \theta_D) - \mathbb{E}_{(s,a)\sim Demo} \log(1 - D(s, a; \theta_D)) \tag{8}$$

Experiment

- Main Results:

Agents	Success Rate	Reward	Turn
Rule	41.34	0.26	16.00
A2C	81.24	5.08	15.43
BBQN-Map	81.56	5.00	18.75
Adversarial A2C	**87.52**	**5.93**	**13.52**

- Learning Curves:



The End