Optimizing Policy via Deep Reinforcement Learning for Dialogue Management

Guanghao Xu, Hyunjung Lee, Myoung-Wan Koo & Jungyun Seo

Sogang University & Universität Leipzig

January 17, 2018
1 Introduction

2 Theoretical Background
   ■ Deep-Reinforcement Learning

3 Architecture of Dialogue Manager
   ■ Dialogue State
   ■ Dialogue Action
   ■ Q-network

4 Experimental Setup
   ■ Corpora
   ■ SLU error
   ■ Baseline
   ■ Training
   ■ Reward Function

5 Results and Discussion

6 Conclusion and Implications

7 Appendix
Overview
Dialogue Manager

Is there any Thai restaurant?

user

audio signal

Auto Speech Recognition
Spoken Language Understanding
Dialog State Tracking

words

dialog acts

Speech Synthesis
Natural Language Generalization

Dialog Management
Introduction

Dialogue Manager

Our question 1:
How can Dialog System produce appropriate response in the next turn?
Introduction

Dialogue Manager

- **Dialogue Manager (DM)** plays a central role in building a successful Spoken Dialog System (SDS)
  1. by apprehending a state of a dialogue in a current turn
  2. by deciding a proper action to take for a next turn
  3. by implementing a human-like agent which interacts with actual users.
Frameworks so far

Rule-based approach

- easy and undemanding to define a set of rules that the system.
- limited flexibility and high maintenance cost.

Reinforcement Learning (RL) framework

- able to learn and train policy over time with experience
- need interventions from a system developer to represent dialogue state, dialogue actions and a reward function which instructs the system on the right track of dialogues.
Goals of this talk

Deep Reinforcement Learning (Deep-RL)

- to learn in an unsupervised way how to control policies in complex environment.
- The agent equipped with deep RL policy surpasses a human expert in several games.
  e.g. Atari games [1]

Our question 2:
Which insights of deep RL could be drawn to optimize policy in Dialog Manager without hand-crafted features?
Theoretical Background
Q-function

- Given a policy $\pi : S \rightarrow A$, an RL-agent selects ‘best’ actions by maximizing its cumulative discounted reward $R_t$,

$$R_t = r_t + \gamma \cdot r_{t+1} + \gamma^2 \cdot r_{t+2} + \ldots + \gamma^{T-1} \cdot r_T$$

where $\gamma$ is a discount factor and $T$ is a final time step.

- A potential value of actions $a$ in the current state $s$ is estimated by Q-function as

$$Q^*(s, a) = \max_\pi E[R_t|s_t = s, a_t = a, \pi]$$
Deep Reinforcement Learning (henceforth, Deep-RL) adopts a function approximator based on deep neural network which is called Q-network.

- Q-network is to estimate the action-value function

$$Q(s, a; \theta) \approx Q^*(s, a),$$ where $\theta$ is the parameters

- The Q-network could be constructed in any forms e.g. a multi-layer feed forward network, a convolutional neural network, a recurrent neural network.
Deep RL algorithm

- In deep RL algorithm, the learning agent maintains two Q-networks:
  1. Policy Network
  2. Value Network
Q-Network = *Policy + Value Network*

At iteration $i$

$$L_i(\theta_i) = E[(E[r + \gamma \cdot \max_{a'} Q(s', a'; \theta_{i-1})|s, a] - Q(s, a; \theta_i))^2]$$

- **The policy network** is trained toward minimizing loss function $L_i(\theta_i)$ that changes at each iteration.

- **The value network** estimates value of target action.
Architecture of Dialogue Manager
The architecture of our dialogue manager toward policy optimization.
Dialogue State

User Simulator

User’s utterance
request(address)
(What is the address?)

Dialogue State Tracker

Dialog Policy

In-domain DB

name=Darrs Cookhouse
address=Camp Suite 321 Port

System’s utterance

User's utterance

Post processing

offer(name=Darrs Cookhouse)
inform(address=Camp Suite 321 Port)
(The Darrs Cookhouse is in Camp Suite 321 Port.)

Reward

offer(name=$value)
inform(address=$value)
Dialogue State

• **Goal:**
  Information that contains what a user wants the system to do should be tracked during entire dialogues to make appropriate response to the user using the SLU results.

• The dialogue state tracker outputs for each turn distributions for each of the three components as follows:
  1. **Goal**
  2. **Method**
  3. **Requested slots**
    in the form of continuous vector.

• Automatically constructed the dialogue state vector
Dialogue Action

User Simulator → User’s utterance (request(address) (What is the address?)) → Dialogue State Tracker

Reward

In-domain DB: name=Darrs Cookhouse, address=Camp Suite 321 Port

System’s utterance

Post processing

Offer(name=Darrs Cookhouse) inform(address=Camp Suite 321 Port) (The Darrs Cookhouse is in Camp Suite 321 Port.)

Dialogue Policy

Dialogue Action: offer(name=$value) inform(address=$value)
Dialogue Action

- Agent’s responses and user’s utterances are converted into semantic form

\[ \text{Act}(\text{slot}, \text{value}) \]

- Goal:
  : to have better control over the system’s behaviors, rather than directly using raw utterances.

- Due to the sparsity issues, \textit{value} is temporarily left vacant in the level of Q-networks.
- The exact instance of \textit{value} is later added in post-processing step.
Q-network

User Simulator → User’s utterance: request(address) (What is the address?)

Dialogue State Tracker → Dialogue State

Reward

In-domain DB

System’s utterance:
- name=Darrs Cookhouse
- address=Camp Suite 321 Port

Post processing:
- offer(name=Darrs Cookhouse)
- inform(address=Camp Suite 321 Port)
- (The Darrs Cookhouse is in Camp Suite 321 Port.)

Dialogue Action:
- offer(name=$value)
- inform(address=$value)

Dialogue Policy
Our question:

Given the input **Dialog State** $s_t$, how the **Policy** in DM can derive the optimal output, **Dialog Act** $a_t$?
Optimizing Policy

• **Goal:**
  - Q-network should be designed to estimate the **action-value function**

  \[ Q(s, a; \theta) \approx Q^*(s, a) \]

  toward optimizing the dialogue policy automatically.

• The Q-network outputs a probability distributions over all agent’s actions given the current dialogue state vector
Our Q-network is constructed in the multi-layer feed forward network:
Experimental Setup
Corpora: DSTC2 & 3

- The DSTC2 and 3 dialogue corpora were collected using Amazon Mechanical Turk [6, 7].
- The domain of DSTC2 provides restaurant information, whereas DSTC3 extends to tourist information, including bars, cafes and etc.
- Examples of tagged dialogues in DSTC2 is in Appendix IV.
SLU error rates

- To test the SLU error robustness, we mimic three environments with different levels of noise by using the SLU N-best results stated in the corpora.

<table>
<thead>
<tr>
<th>SLU Error Level</th>
<th>Top-1 Error Rate</th>
<th>Top-10 Error Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Low</td>
<td>29.02%</td>
<td>16.69%</td>
</tr>
<tr>
<td>High</td>
<td>36.98%</td>
<td>23.71%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SLU Error Level</th>
<th>Top-1 Error Rate</th>
<th>Top-10 Error Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Low</td>
<td>16.17%</td>
<td>6.78%</td>
</tr>
<tr>
<td>High</td>
<td>31.22%</td>
<td>19.43%</td>
</tr>
</tbody>
</table>
Baseline model: **Rule-based Policy**

- To compare the performance of deep RL-policy, we build a rule-based dialogue policy as a baseline model.

**Table: Algorithm – Rule-based dialogue policy**

1. $G \leftarrow$ the ‘goal’ component of the state tracker output.
2. $R \leftarrow$ the ‘requested slot’ component of the state tracker output.
3. $S \leftarrow$ the DB query result with constraints in $G$.
4. $A_m$: placeholder for output system dialogue acts.
5. **if** $\text{length}(S) = 0$ **then**
6. $A_m = \text{canthelp(slot=value)}$, fill slot=value using $G$.
7. **if** $\text{length}(G) < 2$ **then**
8. $A_m = \text{request(slot)}$, fill slot using slots that not yet included in $G$.
9. **else**:
10. $\text{venue} = \text{random}(S)$
11. $A_m = \text{offer(name=venue.name)}$
12. **for** slot in $R$ **do**
13. $A_m = A_m + \text{inform(venue.slot=venue.value)}$
14. Output system response $A_m$.

- It issues a query and makes a response to user’s utterance using a set of predefined rules.
Exploration Strategy

- During the training of the Q-network, we adopt an $\epsilon$-greedy strategy.
- The probability is initially set to 1.0 and gradually decreased to 0.1 over the first 10k dialogues.
- We set $\epsilon$ to 0 and train the policy for another 10k dialogues.
Reward Function

- During scoring the success rate of a dialogue, a reward function is set as follows:
  - **Reward** +20 for successful dialogues
  - **Penalty** -10 for failed dialogues
  - an additional penalty-1 for each dialogue turn
  - to encourage agent to behaves as fast as possible
Results and Discussion
Results in DSTC2: *deep RL vs rule-based policy*

<table>
<thead>
<tr>
<th>SLU Error Level</th>
<th>Policy</th>
<th>Dialogue Success Rate</th>
<th>Average Dialogue Turns</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rule-based</td>
<td>100%</td>
<td>7.42</td>
</tr>
<tr>
<td></td>
<td>Deep RL</td>
<td>99.38%</td>
<td>5.84</td>
</tr>
<tr>
<td>Low</td>
<td>Rule-based</td>
<td>85.57%</td>
<td>7.47</td>
</tr>
<tr>
<td></td>
<td>Deep-RL</td>
<td>90.35%</td>
<td>7.74</td>
</tr>
<tr>
<td>High</td>
<td>Rule-based</td>
<td>77.14%</td>
<td>7.37</td>
</tr>
<tr>
<td></td>
<td>Deep-RL</td>
<td>89.55%</td>
<td>8.16</td>
</tr>
</tbody>
</table>

- The rule-based policy always achieves a 100% dialogue success rate only if there exists no SLU error.
- Under the *Low* SLU error, the deep RL policy outperforms the rule-based policy 4 ~ 5% in terms of dialogue success rate.
- The Deep RL policy has required much shorter turns than the baseline model with rule-based policy.
Results and Discussion

Results in DSTC3: *deep RL vs rule-based policy*

- The advantageous performance results of deep-RL are more noticeable in the extended dialogue domain, DSTC3.

<table>
<thead>
<tr>
<th>SLU 1 Error Level</th>
<th>Policy</th>
<th>Dialogue Success Rate</th>
<th>Average Dialogue Turns</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>Rule-based</td>
<td>100%</td>
<td>8.58</td>
</tr>
<tr>
<td></td>
<td>Deep RL</td>
<td>99.16%</td>
<td>5.84</td>
</tr>
<tr>
<td>Low</td>
<td>Rule-based</td>
<td>91.49%</td>
<td>8.16</td>
</tr>
<tr>
<td></td>
<td>Deep-RL</td>
<td>95.15%</td>
<td>6.86</td>
</tr>
<tr>
<td>High</td>
<td>Rule-based</td>
<td>52.49%</td>
<td>11.53</td>
</tr>
<tr>
<td></td>
<td>Deep-RL</td>
<td>86.85%</td>
<td>8.05</td>
</tr>
</tbody>
</table>
Success Rate under SLU error

- The success rate is converged
  - after 10k dialogues under the *None* SLU error level,
  - after 15k dialogues under the *Low* and *High* case.

Figure: The Success Rate of Dialogues in SLU Error Levels

- The Deep-RL policy needs approximately 90k ~ 700k less than traditional MDP-RL policy.
The overall experimental results suggest

1. Dialogue agent can be trained automatically to successfully complete a dialogue.
2. It can interact with users within much shorter turns by optimizing the policy in deep RL algorithm.
3. Deep-RL policy shows more robustness to SLU error than the rule-based policy.
4. The proposed model requires even smaller size of train data to learn the best action.
Concluding Remarks
Conclusion

- We have proposed the dialogue manager by optimizing the dialogue policy using deep Reinforcement Learning algorithm.

- It shows the deep RL policy is more robust to SLU error and flexible to complex domain of dialogues than the other approaches.

- The deep RL policy interacts with the simulated user more effectively than the rule-based policy.
Implications

Our questions:

• Which insights of deep RL could be drawn to optimize policy in Dialog Manager without hand-crafted features?

• Deep RL offers a **flexible building block** for all steps of Dialogue System without any manually stipulated features.

• It is expected to overcome a challenge by providing promising approaches to manage **diverse domain conversation**.
Thank you!

- **Hyunjung Lee**: hyunjung.lee@uni-leipzig.de
- **Guanghao Xu**: guanghao412@gmail.com
Acknowledgement

This research was supported by the MISP (Ministry of Science, ICT & Future Planning), Korea, under the National Program for Excellence in SW) (2015-0-00910) supervised by the IITP (Institute for Information & communications Technology Promotion).


Appendix I: Reinforcement Learning

• **Goal:**
  to learn its behavior by taking actions in an environment in discrete time steps [2, 3].

• An agent in RL selects ‘best’ actions by **maximizing its cumulative discounted reward** $R_t$,

$$R_t = r_t + \gamma \cdot r_{t+1} + \gamma^2 \cdot r_{t+2} + \ldots + \gamma^{T-1} \cdot r_T$$

where $\gamma$ is a discount factor and $T$ is a final time step [2].
Appendix I: Reinforcement Learning

- At each time $t$, the agent
  1. receives a representation of state $s_t \in S$, where $S$ is a state space
  2. selects an action $a_t \in A$, where $A$ is a set of possible actions that the agent can take.
  3. receives a reward $r_t$
  4. transits to a new state $s_{t+1}$.
Appendix I: Reinforcement Learning

- Given that the agent follows a policy $\pi : S \rightarrow A$, an potential value of actions $a$ in the current state $s$ is estimated by the $Q$-function as

$$Q^*(s, a) = \max_{\pi} E[R_t | s_t = s, a_t = a, \pi]$$

- The more accurate the $Q$-function is, the better policy the agent learns.
- However, they are quite inefficient, especially when the state space becomes large or even infinite.
Appendix I: Reinforcement Learning

- To ensure adequate exploration of state space, the $\epsilon$-greedy strategy is applied.

- The agent greedily chooses an action based on the value of agent’s action calculated by the policy network,

$$a = \max_a Q(s, a; \theta), \text{ with probability } 1 - \epsilon$$

and selects a random action with probability $\epsilon$
Appendix II: Q-network

Example of an Input layer of Q-network

<table>
<thead>
<tr>
<th>Components</th>
<th>Output of Dialogue State Tracker</th>
<th>SLU N-best results of user’s utterance</th>
<th>Results of DB query</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of dimension</td>
<td>Goals</td>
<td>Methods</td>
<td>Requested</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>5</td>
<td>9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>food</th>
<th>pricerange</th>
<th>name</th>
<th>area</th>
<th>this</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.9458</td>
<td>0.6613</td>
<td>0.0</td>
<td>0.0613</td>
<td>0.0</td>
</tr>
</tbody>
</table>
Appendix III: User Simulator

User Simulator

User's utterance
request(address)
(What is the address?)

Dialogue State Tracker

Dialogue State

In-domain DB

name=Darrs Cookhouse
address=Camp Suite 321 Port

Post processing

System's utterance

Offer(name=Darrs Cookhouse)
inform(address=Camp Suite 321 Port)
(The Darrs Cookhouse is in Camp Suite 321 Port.)

Dialogue Action

Offer(name=$value)
inform(address=$value)
Appendix III: User Simulator

• Deep RL agent learns over times by experiences.
• The dialogue manager needs a lot of dialogues to be trained, which is impractical to train with real users [4].

• Goal:
  to train Deep RL agent toward optimizing policy automatically by interacting with user-simulator based on agenda-based [5].
Appendix III: User Simulator

- The process of how user simulator operates

1. Initialize the simulator with a certain agenda which consists of
   - **CONSTRAINTS**
     - e.g. *food=korean, price=cheap, area=east*...
   - **REQUESTS**
     - e.g. *address, phone, signature*...

2. During the dialogue, the simulator interacts with the dialog **agent** based on its agenda

3. Evaluate the success rate of dialogues.
### Table: Example Dialogues in DSTC2 Domain

<table>
<thead>
<tr>
<th>Turn</th>
<th>Speaker</th>
<th>Dialog Act</th>
<th>Real Utterance</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>System</td>
<td>Welcomemsg()</td>
<td>How can I help you?</td>
</tr>
<tr>
<td>1</td>
<td>User</td>
<td>inform(area=centre)</td>
<td>Is there any restaurant in the centre area?</td>
</tr>
<tr>
<td>1</td>
<td>System</td>
<td>request(pricerange)</td>
<td>What price range do you want?</td>
</tr>
<tr>
<td>2</td>
<td>User</td>
<td>inform(pricerange=moderate)</td>
<td>Moderate.</td>
</tr>
<tr>
<td>2</td>
<td>System</td>
<td>offer(name=Venue), inform(area=centre)</td>
<td>&quot;Venue&quot; is a restaurant in the centre area.</td>
</tr>
<tr>
<td>3</td>
<td>User</td>
<td>request(food)</td>
<td>Which kind of food do they serve?</td>
</tr>
<tr>
<td>3</td>
<td>System</td>
<td>offer(name=Venue), inform(food=Thai)</td>
<td>&quot;Venue&quot; is mainly serving Thai food.</td>
</tr>
<tr>
<td>7</td>
<td>User</td>
<td>request(address)</td>
<td>What is the address?</td>
</tr>
<tr>
<td>7</td>
<td>System</td>
<td>inform(address=9558 ...)</td>
<td>The address is 9558 Ramirez Village Apt.</td>
</tr>
<tr>
<td>8</td>
<td>User</td>
<td>thank you()</td>
<td>Thanks!</td>
</tr>
<tr>
<td>8</td>
<td>System</td>
<td>bye()</td>
<td>Bye</td>
</tr>
</tbody>
</table>