IERG6130 Final Project Report: Policy Learning for Task-oriented Dialogue Systems

Jingyan Zhou, Jinchao Li
SEEM, The Chinese University of Hong Kong
Shatin, Hong Kong
{jyzhou, jcli}@se.cuhk.edu.hk

Abstract

Dialog policy optimization is a crucial part for training task-oriented dialogue systems via reinforcement learning. But the training requirement of interactions with real users are costly, while the interactions with user simulators introduces design bias and lacks language complexity. In this paper, we implemented a policy learning method using Dyna-Q framework[1] , which integrates Q-learning with World model, using both real and simulated interactions. The proposed method learns in a more efficient way, and performs better than the baseline — Deep-Q network.

1 Introduction

1.1 Problem and Motivation

Task-oriented dialogue systems (or Conversational Agents), which can assist users to complete tasks, has been a hot topic for decades. There are several ways to build a task-oriented dialogue system including rule-based, modular-based and end-to-end approaches, and in the research community, the modular-based approach is receiving increasing attention. A typical modular-based system for text-based conversational agents (shown in Figure 1) contains four key parts: 1. Natural Language Understanding (NLU) for extracting users' intends and task-related information; 2. Dialogue State Tracking (DST) that can track the current state of the dialogue; 3. Dialogue Policy (POL) making decision on what action to take next. 4. Natural Language Generation (NLG) generating responses in natural language to the users.

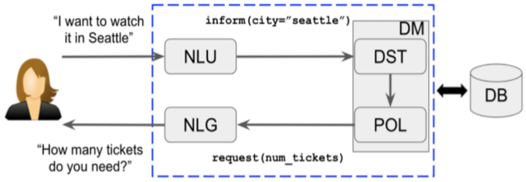

"I want to watch it in Seattle"
inform(city="seattle")
DM
NLU
DST
DB
NLG
POL
"How many tickets do you need?"
request(num_tickets)

Figure 1: Components of a task-oriented dialogue system [1]

To take appropriate dialog-acts and steer the conversation, the dialogue policies are crucial for conversational agents. Recently, reinforcement learning(RL) has great breakthroughs in Atari games and therefore is also being more and more popular in the field of dialog policy optimization as it can learn policies via conversational interactions with users. In this project, we focus on RL-based dialog

[1] Our github repo: https://github.com/para-zhou/RL_DDQ

policy learning. The most ideal setting of RL-based dialog policy learning is to enable the agent to learn directly from the interactions with real users. However, it is costly as the training may require a large number of conversations. Alternatively, user simulators[2] which are usually built by expert knowledge and heuristics or supervised learning (SL) [3] are employed to train the agent. But the simulated user lacks language complexity comparing to the real user. And it may have biases due to unbalanced training data.

To address above problems, an intuitive way is to use the real interactions two-fold: first is to help the policy agent conduct direct reinforcement learning, and second is to update the user simulator. This is the idea of Dyna-Q[4] which integrates planning into policy learning as shown in Figure 2. In this work, we implemented a policy learning method using Dyna-Q framework. We follow the work of Peng et al. in [5] to employ world model[6] to be the user simulator to generate simulated experience and rewards. To capture more information, the world model and Q-function are implemented by multi-layer neural networks, i.e. the deep neural network is integrated to Dyna-Q as Deep Dyna-Q(DDQ) [5].

We present our work in this report as follows: the detailed framework is fully illustrated in Section 2, the experiments and performance evaluation is shown in Section 3, and our work is discussed and concluded in Section 4.

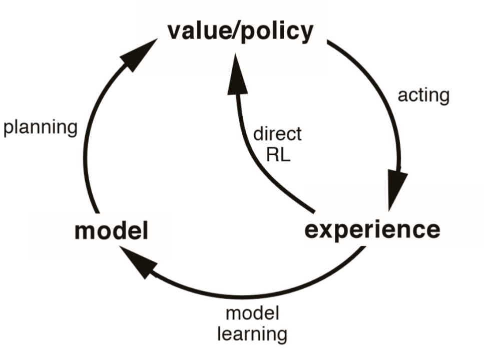


Figure 2: Relationships among planning, learning, and acting [7].

1.2 Related Work

Dialogue policy can be optimized by either supervised learning or reinforcement learning algorithms. Typically, a rule-based agent is employed to warm-start the system, then supervised learning is conducted on the actions generated by the rules [1]. One popular way is to treat this task as a Partially Observed Markov Decision Process(POMDP) [8]. Recently the reinforcement learning which enables the agent to learn from the feedback of real users when interacting with them emerged as a popular approach for dialogue systems, which seems outperformed the random, rule-based, and supervised-based methods [9].

There are two ways to use reinforcement learning for dialogue policy optimization: online learning and batch learning. The online approach requires the learner to interact with users to improve its policy; the batch approach assumes a fixed set of transitions, and optimizes the policy based on the data only, without interacting with users [10]. The online setting often has batch learning as internal step as we done in this work.

Considering the impractical cost of real user online learning, one popular approach is to employ user simulators to generate user experience and train the agent[2]. However, the performance of the simulators is limited by the way they are constructed (including expert knowledge, supervised learning[11], etc.). Building a reliable user simulator which can model the environment and generate simulated experience is almost as complex as building a policy agent. Recently, there's an increasing trends of training the simulator and the agent simultaneously[3, 5] to improve the performance of the simulator.

As for the direct reinforcement learning, neural network enhanced Q-learning is widely used in RL-based dialog policy learning. The standard deep Q-network(DQN) [12] usually adopt $\epsilon-$

greedy to balance exploration and exploitation. To improve the efficiency of exploration, Lipton et al. proposed Bayes-By-Backprop Q-Network(BBQN) [11] to explore via Thompson Sampling. Another technique which adopts imitation learning, to be specific, Replay Buffer Spikes(RBS) to pre-fill the replay buffer via a rule-based agent is proven to be useful for improving the training efficiency [11].

2 Methodology

2.1 Model Structure

For the policy agent, this workimplemented a Dyna-Q model with rule-based warm start. As illustrated in Figure 3, the Dyna-Q model mainly consists of four parts: (1) RL update with real experience[2], (2) Model[3] learning from real experience, (3) World model search control[4] to simulated experience[5], (4) Planning update with simulated experience.

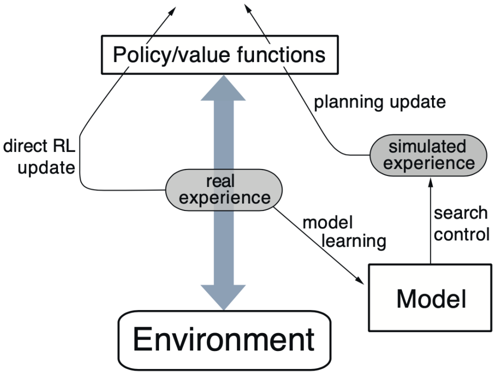


Figure 3: The General Dyna Architecture [7].

Therefore, the policy agent pipeline in our work can be mainly summarized in the following processes:

1. Warm start using rule-based agent
2. Policy update with real experience
3. World model learning with real experience
4. World model search control to simulated experience
5. Planning update with simulated experience
6. Repeat 2-5 until convergence

2.2 DQN, World model and Planning

For simplicity, the policy agent and World model in our work are both implemented by Multi-layer Perceptrons (MLPs), parameterized by θ_Q and θ_M respectively.

The RL policy agent is implemented via Deep Q-network (DQN), which is designed to approximate the Q-function of the RL system, i.e. generates the Q-value of state s, donated as $Q(s, a; \theta_Q)$. In each step, the agent observes the dialogue state s, and chooses the next action using ϵ-greedy policy, i.e. choosing random action with probability ϵ or otherwise choosing $a = \arg\max_{a'} Q(s, a'; \theta_Q)$. With the action a to state s, the agent then receives a reward r[6], observes next user response a^u, as well as the next state s'. The loss function of DQN is:

$$\mathcal{L}(\theta) = \mathbb{E}\{r + \max_{a'} Q'(s', a'; \theta_{Q'}) - Q(s, a; \theta_Q)\}$$

[2] Interactions with environment (real user).

[3] Such as World model, tabular model, etc..

[4] Selecting the starting states and actions for the simulated experiences generated by the model.

[5] Interactions with simulator.

[6] In direct RL process, we set reward of success: 80, failure: -40, each turn: -1.

where $Q'(*)$ is the target Q-network.

Specifically, the input of DQN is an encoding of current dialogue state s. We use a feature vector consisting of (a) dialogue-act and slot(s)[7] of last user/system action, (b) a bag of all appeared slots[8], (c) the current turn count, and (d) the results that subjects to the constraints for information slots. Then the DQN outputs a real-value vector q, containing all possible or the best (dialogue-act, slot) pairs that can be chosen by the system. Base on prior knowledge, some nonsense act-slot pairs will be reduced, such as `request(price)`.

The World model [13] here takes the state s and action a as input, and outputs the simulated user response a^u, reward r[9], and terminate information t, denoted as $M(s, a; \theta_M)$. Specifically, $M(s, a; \theta_M)$ is a multi-task model that combines two classification tasks for approximating a^u and t, and one regression task for simulating r.

In the planning process, the agent performs K steps planning in direct RL policy and interacts with the World model, which is similar to dynamic programming. At the beginning of each step, we randomly draw a user goal with constraints and requests, then the agent interacts and updates with the simulated experience.

2.3 Algorithm

The detailed training procedure is shown in Algorithm 1.

Algorithm 1 DDQ for Dialogue Policy Learning

```
Require: $N$, $\epsilon$, $K$, $L$, $C$
Ensure: $Q(s, a; \theta_Q)$, $M(s, a; \theta_M)$
initialize $Q(s, a; \theta_Q)$ and $M(s, a; \theta_M)$ via pre-training on real experience
initialize $Q'(s, a; \theta_{Q'})$ with $\theta_{Q'} = \theta_Q$
initialize real experience replay buffer $\mathbb{R}$ and simulated experience replay buffer $\mathbb{S}$ as empty
for $n$=1:$N$ do
    # Direct Reinforcement Learning
    user starts with action $a^u$, generate an initial state $s$
    while not $t$ do
        with probability $\epsilon$ select a random action $a$
        o/w select $a = \arg\max_{a'} Q(s, a'; \theta_Q)$
        observe user response $a^u$ and reward $r$, update next state $s'$
        store $(s, a, r, a^u, s')$ to $\mathbb{R}$
        $s = s'$
    end while
    sample randomly from $\mathbb{R}$, update $\theta_Q$ with $\mathbb{R}$
    # World Model Learning
    sample randomly from $\mathbb{R}$, update $\theta_M$
    # Planning
    for $k$=1:$K$ do
        $t$ = FALSE, $l = 0$
        sample a user action $a^u$, generate an initial state $s$
        while not $t \wedge l \leq L$ do
            with probability $\epsilon$ select a random action $a$
            o/w select $a = \arg\max_{a'} Q(s, a'; \theta_Q)$
            world model responds with $a^u$, $r$ and $t$, update $s'$
            store $(s, a, r, s')$ to $\mathbb{S}$
            $l = l + 1$, $s = s'$
        end while
        sample randomly from $\mathbb{S}$, update $\theta_Q$
    end for
    every $C$ steps reset $\theta_{Q'} = \theta_Q$
end for
```

[7] There are some necessary predefined acts and slots in a specific domain. For example, in the movie-booking task, dialog-acts can be `suggest`, `inform`, `request`, etc.; and slots can be `movie_name`, `theater_name`, `time`, `date`, `ticket_price`, `num_tickets`, etc.

[8] For the replay buffer.

[9] In planning process, reward is generated by the World model.

3 Experiments and Results

3.1 Setup and Data

1. Task: The task in our work is movie-ticket booking, which contains 11 acts and 16 slots listed in Table 1 below.

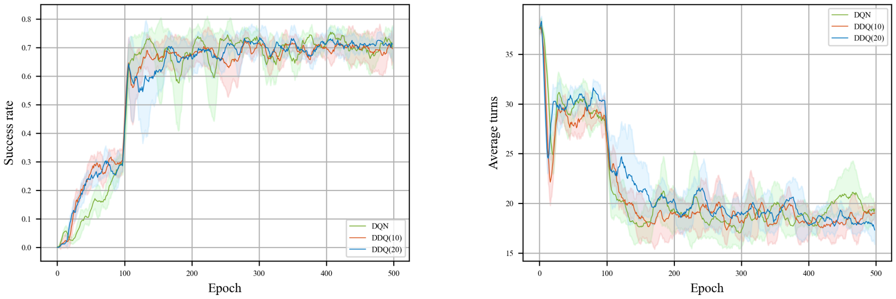
Acts	Slots
greeting, request, inform, deny, confirm_question, confirm_answer, not_sure, multiple_choice, thanks, welcome, closing	city, closing, date, distanceconstraints, greeting, moviename, numberofpeople, price, starttime, state, ticket, theater, zip, theater_chain, taskcomplete, video_format

Table 1: Acts and slots in movie-ticket booking task domain.

2. Dataset: The seed data used here is from MiuLab[10], where the raw conversational data comes from Amazon Mechanical Turk. Every utterance in the corpora is labelled with dialog action and slots which indicate the dialog state.
3. Training: Referring to BBQ-Net [11], the rule-based agent, NLU, DST, NLG modules are adopted from other works [14–17]. And then we use the proposed DDQ and baseline DQN as POL module. At the beginning, we pre-filled a small number of successful experience RBS by rule-based agent. Then we started the policy learning from random state, with same epochs and update frequency $N = 500, C = 1$, but different steps K. Considering that the ϵ-greedy policy tends to fail for the large action space, we set $\epsilon = 0.3$ first then changed it to 0 after 100-th epoch.
4. Evaluation Metrics: (1) Success rate, (2) Average turns. The average reward is directly related with the success rate and average turns, so we didn't demonstrate it for conciseness.

3.2 Experiments and Results

To demonstrate the efficiency of our model, we compare DDQ against ϵ-greedy in a standard Deep Q-Network. And to evaluate the performance of these models, we report the results on success rate and average turns, as shown in Figure 4.



(a) Success Rate

(b) Average Turns

Figure 4: The performance of DDQ(K) and DQN.

As illustrated in Figure 4, the success rate rises faster for DDQ at the beginning, and turns higher and smoother until convergence than the DQN; for the average turns, the DDQ agents use fewer turns until convergence. Moreover, under the premise of correct world model, the agents with suitably larger steps of planning perform better than the ones with small planning steps. All in all, these results depict that DDQ agents consistently outperform the DQN agent in success, effectiveness and robustness.

[10] `https://github.com/MiuLab/TC-Bot/tree/master/src/deep_dialog/data`

Table 2 shows two successful and failed conversational samples between trained DDQ agents and users. The task failed when the agent not satisfied the user's goals. For the successful conversation, we can find that the agent completes the task soon after it has collected the sufficient information.

Successful	Failed
0 *usr*: Can I buy tickets for hail caesar at seattle?	0 *usr*: What start time can I book 4 tickets for deadpool?
1 *sys*: Which theater would you like?	1 *sys*: 6:10pm is available.
2 *usr*: I want to watch at regal meridian 16.	2 *usr*: I want to set it up tomorrow.
3 *sys*: 8:45 pm is available.	3 *sys*: City is not available!
4 *usr*: I want to set it up tomorrow.	4 *usr*: I want to watch at philadelphia.
5 *sys*: Yes, please.	5 *sys*: Starttime is not available!
6 *usr*: I want 2 tickets please!	6 *usr*: Which theater is available?
7 *sys*: Okay - I was able to book 2 tickets for you	7 *sys*: Oh sorry, there is no ticket available.
to see hail caesar at regal meridian 16	· · ·
in seattle at 8:45 pm tomorrow.	37 *sys*: State is not available!
8 *usr*: Thank you!	38 *usr*: Thank you!
9 *sys*: Thank you!	39 *sys*: Thank you!

Table 2: Acts and slots in movie-ticket booking task domain.

4 Discussion and Conclusion

In this work, we implemented a Deep Dyna-Q model for policy learning in task-oriented dialogue systems. The DDQ model effectively makes use of the real and simulated experience by learning and planning. The experimental results show that the DDQ agents consistently outperform the DQN agent (which doesn't contain planning) in success, effectiveness and robustness.

The agents with suitably larger steps of planning perform better than the ones with small planning steps. But it not means the larger K the better, because the max turns is limited, and the performance is dependent on the quality of World model and simulated experience.

There are some challenging but interesting future works for improving the dialogue policy learning. One is the effective exploration method, because ϵ-greedy policy tends to fail in the large action spaces. The trade-off between exploration and exploitation deserves more research effort. Another one is the domain extension problem for task-orient agents, which means extending the agent in one specific domain to other domains.

References

[1] D. Jurafsky. *Speech & language processing*. Pearson Education India, 2000.

[2] J. Schatzmann, K. Weilhammer, M. Stuttle, and S. Young. A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *The knowledge engineering review*, 21(2):97–126, 2006.

[3] B. Liu and I. Lane. Iterative policy learning in end-to-end trainable task-oriented neural dialog models. *CoRR*, abs/1709.06136, 2017.

[4] R. S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine learning proceedings 1990*, pages 216–224. Elsevier, 1990.

[5] B. Peng, X. Li, J. Gao, et al. Deep dyna-q: Integrating planning for task-completion dialogue policy learning. *arXiv preprint arXiv:1801.06176*, 2018.

[6] D. Ha and J. Schmidhuber. World models. *CoRR*, abs/1803.10122, 2018.

[7] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[8] S. Young, M. Gašić, B. Thomson, et al. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179, 2013.

[9] H. Chen, X. Liu, D. Yin, et al. A survey on dialogue systems: Recent advances and new frontiers. *Acm Sigkdd Explorations Newsletter*, 19(2):25–35, 2017.

[10] J. Gao, M. Galley, L. Li, et al. Neural approaches to conversational ai. *Foundations and Trends® in Information Retrieval*, 13(2-3):127–298, 2019.

[11] Z. Lipton, X. Li, J. Gao, et al. Bbq-networks: Efficient exploration in deep reinforcement learning for task-oriented dialogue systems. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[13] X. Liu, J. Gao, X. He, et al. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *Association for Computational Linguistics*, 2015.

[14] X. Li, Z. C. Lipton, B. Dhingra, et al. A user simulator for task-completion dialogues, 2016.

[15] D. Hakkani-Tür, G. Tür, A. Celikyilmaz, et al. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. In *Interspeech*, pages 715–719, 2016.

[16] N. Mrkšić, D. O. Séaghdha, T.-H. Wen, et al. Neural belief tracker: Data-driven dialogue state tracking. *arXiv preprint arXiv:1606.03777*, 2016.

[17] T.-H. Wen, M. Gasic, N. Mrksic, et al. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745*, 2015.