Deep Dyna-Q: Integrating Planning for Task-Completion Dialogue Policy Learning

Baolin Peng[⋆] **Xiujun Li**[†] **Jianfeng Gao**[†] **Jingjing Liu**[†] **Kam-Fai Wong**[⋆‡] **Shang-Yu Su**[§]
[†]Microsoft Research, Redmond, WA, USA
[⋆]The Chinese University of Hong Kong, Hong Kong
[‡]MoE Key Lab of High Confidence Software Technologies, China
[§]National Taiwan University, Taipei, Taiwan
{xiul,jfgao,jingjl}@microsoft.com
{blpeng,kfwong}@se.cuhk.edu.hk shangyusu.tw@gmail.com

Abstract

Training a task-completion dialogue agent via reinforcement learning (RL) is costly because it requires many interactions with real users. One common alternative is to use a user simulator. However, a user simulator usually lacks the language complexity of human interlocutors and the biases in its design may tend to degrade the agent. To address these issues, we present Deep Dyna-Q, which to our knowledge is the first deep RL framework that integrates planning for task-completion dialogue policy learning. We incorporate into the dialogue agent a model of the environment, referred to as the *world model*, to mimic real user response and generate simulated experience. During dialogue policy learning, the world model is constantly updated with real user experience to approach real user behavior, and in turn, the dialogue agent is optimized using both real experience and simulated experience. The effectiveness of our approach is demonstrated on a movie-ticket booking task in both simulated and human-in-the-loop settings[1].

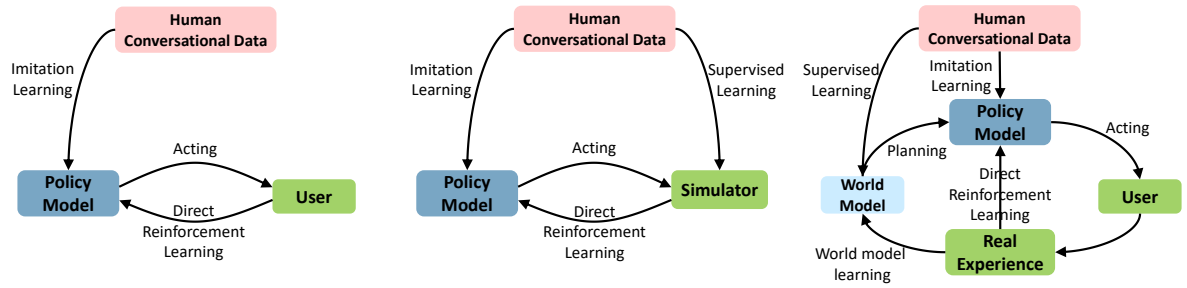
1 Introduction

Learning policies for task-completion dialogue is often formulated as a reinforcement learning (RL) problem (Young et al., 2013; Levin et al., 1997). However, applying RL to real-world dialogue systems can be challenging, due to the constraint that an RL learner needs an environment to operate in. In the dialogue setting, this requires a dialogue agent to interact with real users and adjust its policy in an online fashion, as illustrated in Figure 1(a). Unlike simulation-based games such as Atari games (Mnih et al., 2015) and AlphaGo (Silver et al., 2016a, 2017) where RL has made its greatest strides, task-completion dialogue systems may incur significant real-world cost in case of failure. Thus, except for very simple tasks (Singh et al., 2002; Gašić et al., 2010, 2011; Pietquin et al., 2011; Li et al., 2016a; Su et al., 2016b), RL is too expensive to be applied to real users to train dialogue agents from scratch.

One strategy is to convert human-interacting dialogue to a simulation problem (similar to Atari games), by building a user simulator using human conversational data (Schatzmann et al., 2007; Li et al., 2016b). In this way, the dialogue agent can learn its policy by interacting with the simulator instead of real users (Figure 1(b)). The simulator, in theory, does not incur any real-world cost and can provide unlimited simulated experience for reinforcement learning. The dialogue agent trained with such a user simulator can then be deployed to real users and further enhanced by only a small number of human interactions. Most of recent studies in this area have adopted this strategy (Su et al., 2016a; Lipton et al., 2016; Zhao and Eskenazi, 2016; Williams et al., 2017; Dhingra et al., 2017; Li et al., 2017; Liu and Lane, 2017; Peng et al., 2017b; Budzianowski et al., 2017; Peng et al., 2017a).

However, user simulators usually lack the conversational complexity of human interlocutors, and the trained agent is inevitably affected by biases in the design of the simulator. Dhingra et al. (2017) demonstrated a significant discrepancy in a simulator-trained dialogue agent when evaluated with simulators and with real users. Even more challenging is the fact that there is no universally accepted metric to evaluate a user simulator (Pietquin and Hastie, 2013). Thus, it remains

[1]The source code of this work is available at https://github.com/MiuLab/DDQ



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

Figure 1: Three strategies of learning task-completion dialogue policies via RL.

controversial whether training task-completion dialogue agent via simulated users is a valid approach.

We propose a new strategy of learning dialogue policy by interacting with real users. Compared to previous works (Singh et al., 2002; Li et al., 2016a; Su et al., 2016b; Papangelis, 2012), our dialogue agent learns in a much more efficient way, using only a small number of real user interactions, which amounts to an affordable cost in many nontrivial dialogue tasks.

Our approach is based on the Dyna-Q framework (Sutton, 1990) where planning is integrated into policy learning for task-completion dialogue. Specifically, we incorporate a model of the environment, referred to as the *world model*, into the dialogue agent, which simulates the environment and generates simulated user experience. During the dialogue policy learning, real user experience plays two pivotal roles: first, it can be used to improve the world model and make it behave more like real users, via supervised learning; second, it can also be used to directly improve the dialogue policy via RL. The former is referred to as *world model learning*, and the latter *direct reinforcement learning*. Dialogue policy can be improved either using real experience directly (i.e., direct reinforcement learning) or via the world model indirectly (referred to as *planning* or *indirect reinforcement learning*). The interaction between world model learning, direct reinforcement learning and planning is illustrated in Figure 1(c), following the Dyna-Q framework (Sutton, 1990).

The original papers on Dyna-Q and most its early extensions used tabular methods for both planning and learning (Singh, 1992; Peng and Williams, 1993; Moore and Atkeson, 1993; Kuvayev and Sutton, 1996). This table-lookup representation limits its application to small problems only. Sutton et al. (2012) extends the Dyna architecture to linear function approximation, making it applicable to larger problems. In the dialogue setting, we are dealing with a much larger action-state space. Inspired by Mnih et al. (2015), we propose Deep Dyna-Q (DDQ) by combining Dyna-Q with deep learning approaches to representing the state-action space by neural networks (NN).

By employing the world model for planning, the DDQ method can be viewed as a model-based RL approach, which has drawn growing interest in the research community. However, most model-based RL methods (Tamar et al., 2016; Silver et al., 2016b; Gu et al., 2016; Racanière et al., 2017) are developed for simulation-based, synthetic problems (e.g., games), but not for human-in-the-loop, real-world problems. To these ends, our main contributions in this work are two-fold:

- We present Deep Dyna-Q, which to the best of our knowledge is the first deep RL framework that incorporates planning for task-completion dialogue policy learning.
- We demonstrate that a task-completion dialogue agent can efficiently adapt its policy on the fly, by interacting with real users via RL. This results in a significant improvement in success rate on a nontrivial task.

2 Dialogue Policy Learning via Deep Dyna-Q (DDQ)

Our DDQ dialogue agent is illustrated in Figure 2, consisting of five modules: (1) an LSTM-based natural language understanding (NLU) module (Hakkani-Tür et al., 2016) for identifying user intents and extracting associated slots; (2) a state tracker (Mrkšić et al., 2016) for tracking the dialogue states; (3) a dialogue policy which selects

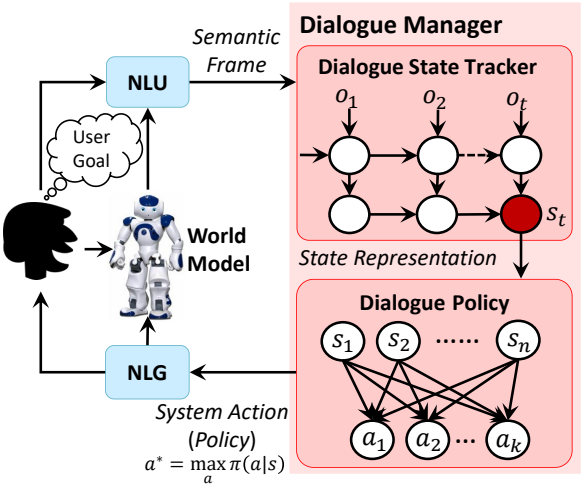


Figure 2: Illustration of the task-completion DDQ dialogue agent.

the next action[2] based on the current state; (4) a model-based natural language generation (NLG) module for converting dialogue actions to natural language response (Wen et al., 2015); and (5) a world model for generating simulated user actions and simulated rewards.

As illustrated in Figure 1(c), starting with an initial dialogue policy and an initial world model (both trained with pre-collected human conversational data), the training of the DDQ agent consists of three processes: (1) *direct reinforcement learning*, where the agent interacts with a real user, collects real experience and improves the dialogue policy; (2) *world model learning*, where the world model is learned and refined using real experience; and (3) *planning*, where the agent improves the dialogue policy using simulated experience.

Although these three processes conceptually can occur simultaneously in the DDQ agent, we implement an iterative training procedure, as shown in Algorithm 1, where we specify the order in which they occur within each iteration. In what follows, we will describe these processes in details.

2.1 Direct Reinforcement Learning

In this process (lines 5-18 in Algorithm 1) we use the DQN method (Mnih et al., 2015) to improve the dialogue policy based on real experience. We consider task-completion dialogue as a Markov Decision Process (MDP), where the agent interacts with a user in a sequence of actions to accomplish a user goal. In each step, the agent observes the dialogue state s, and chooses the action a to execute, using an ϵ-greedy policy that selects a random action with probability ϵ or otherwise follows the greedy policy $a = \text{argmax}_{a'} Q(s, a'; \theta_Q)$. $Q(s, a; \theta_Q)$ which is the approximated value function, implemented as a Multi-Layer Perceptron (MLP) parameterized by θ_Q. The agent then receives reward[3] r, observes next user response a^u, and updates the state to s'. Finally, we store the experience (s, a, r, a^u, s') in the replay buffer D^u. The cycle continues until the dialogue terminates.

We improve the value function $Q(s, a; \theta_Q)$ by adjusting θ_Q to minimize the mean-squared loss function, defined as follows:

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

where $\gamma \in [0, 1]$ is a discount factor, and $Q'(.)$ is the target value function that is only periodically updated (line 42 in Algorithm 1). By differentiating the loss function with respect to θ_Q, we arrive at the following gradient:

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

As shown in lines 16-17 in Algorithm 1, in each iteration, we improve $Q(.)$ using minibatch Deep Q-learning.

2.2 Planning

In the planning process (lines 23-41 in Algorithm 1), the world model is employed to generate simulated experience that can be used to improve dialogue policy. K in line 24 is the number of planning steps that the agent performs per step of direct reinforcement learning. If the world model is able to accurately simulate the environment, a big K can be used to speed up the policy learning. In DDQ, we use two replay buffers, D^u for storing real experience and D^s for simulated experience. Learning and planning are accomplished

[2] In the dialogue scenario, actions are dialogue-acts, consisting of a single act and a (possibly empty) collection of $(slot = value)$ pairs (Schatzmann et al., 2007).

[3] In the dialogue scenario, reward is defined to measure the degree of success of a dialogue. In our experiment, for example, success corresponds to a reward of 80, failure to a reward of -40, and the agent receives a reward of -1 at each turn so as to encourage shorter dialogues.

Algorithm 1 Deep Dyna-Q for Dialogue Policy Learning

Require: N, ϵ, K, L, C, Z
Ensure: $Q(s,a;\theta_Q), M(s,a;\theta_M)$
1: initialize $Q(s,a;\theta_Q)$ and $M(s,a;\theta_M)$ via pre-training on human conversational data
2: initialize $Q'(s,a;\theta_{Q'})$ with $\theta_{Q'} = \theta_Q$
3: initialize real experience replay buffer D^u using Reply Buffer Spiking (RBS), and simulated experience replay buffer D^s as empty
4: **for** n=1:N **do**
5: *# Direct Reinforcement Learning* starts
6: *user* starts a dialogue with user action a^u
7: generate an initial dialogue state s
8: **while** s is not a terminal state **do**
9: with probability ϵ select a random action a
10: otherwise select $a = \text{argmax}_{a'} Q(s,a';\theta_Q)$
11: execute a, and observe *user* response a^u and reward r
12: update dialogue state to s'
13: store (s,a,r,a^u,s') to D^u
14: $s = s'$
15: **end while**
16: sample random minibatches of (s,a,r,s') from D^u
17: update θ_Q via Z-step minibatch Q-learning according to Equation (2)
18: *# Direct Reinforcement Learning* ends
19: *# World Model Learning* starts
20: sample random minibatches of training samples (s,a,r,a^u,s') from D^u
21: update θ_M via Z-step minibatch SGD of multi-task learning
22: *# World Model Learning* ends
23: *# Planning* starts
24: **for** k=1:K **do**
25: t = FALSE, $l = 0$
26: sample a user goal G
27: sample user action a^u from G
28: generate an initial dialogue state s
29: **while** t is FALSE $\wedge\ l \leq L$ **do**
30: with probability ϵ select a random action a
31: otherwise select $a = \text{argmax}_{a'} Q(s,a';\theta_Q)$
32: execute a
33: *world model* responds with a^u, r and t
34: update dialogue state to s'
35: store (s,a,r,s') to D^s
36: $l = l+1$, $s = s'$
37: **end while**
38: sample random minibatches of (s,a,r,s') from D^s
39: update θ_Q via Z-step minibatch Q-learning according to Equation (2)
40: **end for**
41: *# Planning* ends
42: every C steps reset $\theta_{Q'} = \theta_Q$
43: **end for**

by the same DQN algorithm, operating on real experience in D^u for learning and on simulated experience in D^s for planning. Thus, here we only describe the way the simulated experience is generated.

Similar to Schatzmann et al. (2007), at the beginning of each dialogue, we uniformly draw a user goal $G = (C, R)$, where C is a set of constraints and R is a set of requests (line 26 in Algorithm 1). For movie-ticket booking dialogues, constraints are typically the name and the date of the movie, the number of tickets to buy, etc. Requests can contain these slots as well as the location of the theater, its start time, etc. Table 3 presents some sampled user goals and dialogues generated by simulated and real users, respectively. The first user action a^u (line 27) can be either a request or an inform dialogue-act. A request, such as `request(theater; moviename=batman)`, consists of a request slot and multiple ($\geqslant$ 1) constraint slots, uniformly sampled from R and C, respectively. An inform contains constraint slots only. The user action can also be converted to natural language via NLG, e.g., `"which theater will show batman?"`

In each dialogue turn, the world model takes as input the current dialogue state s and the last agent action a (represented as an one-hot vector), and generates user response a^u, reward r, and a binary variable t, which indicates whether the dialogue terminates (line 33). The generation is accomplished using the world model $M(s,a;\theta_M)$, a MLP shown in Figure 3, as follows:

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

where (s,a) is the concatenation of s and a, and W and b are parameter matrices and vectors, respectively.

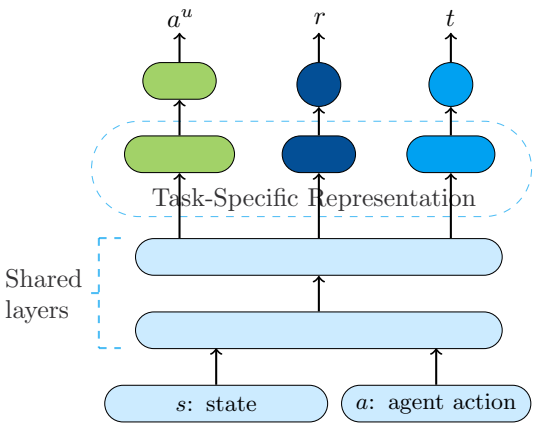


Figure 3: The world model architecture.

2.3 World Model Learning

In this process (lines 19-22 in Algorithm 1), $M(s, a; \theta_M)$ is refined via minibatch SGD using real experience in the replay buffer D^u. As shown in Figure 3, $M(s, a; \theta_M)$ is a multi-task neural network (Liu et al., 2015) that combines two classification tasks of simulating a^u and t, respectively, and one regression task of simulating r. The lower layers are shared across all tasks, while the top layers are task-specific.

3 Experiments and Results

We evaluate the DDQ method on a movie-ticket booking task in both simulation and human-in-the-loop settings.

3.1 Dataset

Raw conversational data in the movie-ticket booking scenario was collected via Amazon Mechanical Turk. The dataset has been manually labeled based on a schema defined by domain experts, as shown in Table 4, which consists of 11 dialogue acts and 16 slots. In total, the dataset contains 280 annotated dialogues, the average length of which is approximately 11 turns.

3.2 Dialogue Agents for Comparison

To benchmark the performance of DDQ, we have developed different versions of task-completion dialogue agents, using variations of Algorithm 1.

- A **DQN** agent is learned by standard DQN, implemented with direct reinforcement learning only (lines 5-18 in Algorithm 1) in each epoch.
- The **DDQ(K)** agents are learned by DDQ of Algorithm 1, with an initial world model pre-trained on human conversational data, as described in Section 3.1. K is the number of planning steps. We trained different versions of DDQ(K) with different K's.
- The **DDQ(K, rand-init θ_M)** agents are learned by the DDQ method with a randomly initialized world model.
- The **DDQ(K, fixed θ_M)** agents are learned by DDQ with an initial world model pre-trained on human conversational data. But the world model is not updated afterwards. That is, the *world model learning* part in Algorithm 1 (lines 19-22) is removed. The DDQ(K, fixed θ_M) agents are evaluated in the simulation setting only.
- The **DQN(K)** agents are learned by DQN, but with K times more real experiences than the DQN agent. DQN(K) is evaluated in the simulation setting only. Its performance can be viewed as the upper bound of its DDQ(K) counterpart, assuming that the world model in DDQ(K) perfectly matches real users.

Implementation Details All the models in these agents ($Q(s, a; \theta_Q)$, $M(s, a; \theta_M)$) are MLPs with `tanh` activations. Each policy network $Q(.)$ has one hidden layer with 80 hidden nodes. As shown in Figure 3, the world model $M(.)$ contains two shared hidden layers and three task-specific hidden layers, with 80 nodes in each. All the agents are trained by Algorithm 1 with the same set of hyper-parameters. ϵ-greedy is always applied for exploration. We set the discount factor γ = 0.95. The buffer sizes of both D^u and D^s are set to 5000. The target value function is updated at the end of each epoch. In each epoch, $Q(.)$ and $M(.)$ are refined using one-step ($Z = 1$) 16-tuple-minibatch update. [4] In planning, the maximum length of a simulated dialogue is 40 ($L = 40$). In addition, to make the dialogue training efficient, we also applied a variant of imitation learning, called Reply Buffer Spiking (RBS) (Lipton et al., 2016). We built a naive but occasionally successful rule-based agent based on human conversational dataset (line 1 in Algorithm 1), and pre-filled the real experience replay buffer D^u with 100 dialogues of experience (line 2) before training for all the variants of agents.

3.3 Simulated User Evaluation

In this setting the dialogue agents are optimized by interacting with user simulators, instead of real users. Thus, the world model is learned to mimic user simulators. Although the simulator-trained agents are sub-optimal when applied to real users due to the discrepancy between simulators and real users, the simulation setting allows us to perform a detailed analysis of DDQ without much cost and to reproduce the experimental results easily.

[4] We found in our experiments that setting $Z > 1$ improves the performance of all agents, but does not change the conclusion of this study: DDQ consistently outperforms DQN by a statistically significant margin. Conceptually, the optimal value of Z used in planning is different from that in direct reinforcement learning, and should vary according to the quality of the world model. The better the world model is, the more aggressive update (thus bigger Z) is being used in planning. We leave it to future work to investigate how to optimize Z for planning in DDQ.

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*

Table 1: Results of different agents at training epoch = {100, 200, 300}. Each number is averaged over 5 runs, each run tested on 2000 dialogues. Excluding DQN(5) and DQN(10) which serve as the upper bounds, any two groups of success rate (except three groups: at epoch 100, DDQ(5, rand-init θ_M) and DDQ(10, fixed θ_M), at epoch 200, DDQ(5, rand-init θ_M) and DDQ(10, rand-init θ_M), at epoch 300, DQN and DDQ(10, fixed θ_M)) evaluated at the same epoch is statistically significant in mean with $p < 0.01$. (Success: success rate)

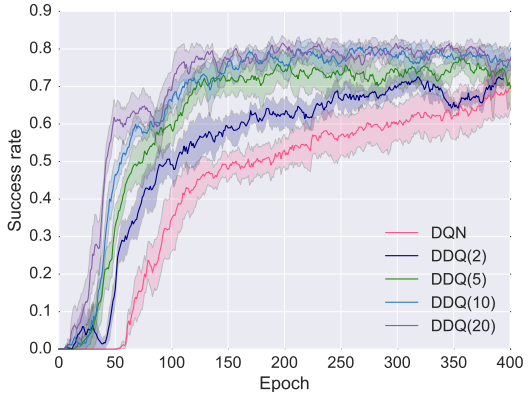


Figure 4: Learning curves of the DDQ(K) agents with $K = 2, 5, 10, 20$. The DQN agent is identical to a DDQ(K) agent with $K = 0$.

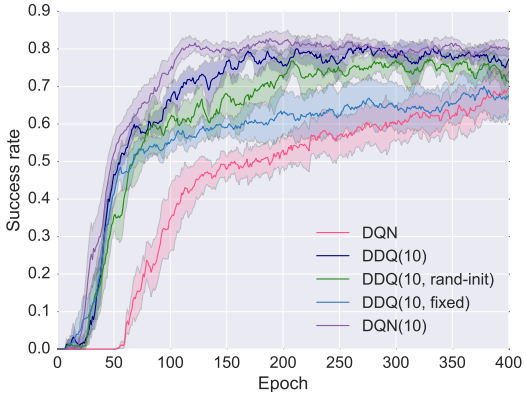


Figure 5: Learning curves of DQN, DDQ(10), DDQ(10, rand-init θ_M), DDQ(10, fixed θ_M), and DQN(10).

User Simulator We adapted a publicly available user simulator (Li et al., 2016b) to the task-completion dialogue setting. During training, the simulator provides the agent with a simulated user response in each dialogue turn and a reward signal at the end of the dialogue. A dialogue is considered successful only when a movie ticket is booked successfully and when the information provided by the agent satisfies all the user's constraints. At the end of each dialogue, the agent receives a positive reward of $2 * L$ for success, or a negative reward of $-L$ for failure, where L is the maximum number of turns in each dialogue, and is set to 40 in our experiments. Furthermore, in each turn, the agent receives a reward of -1, so that shorter dialogues are encouraged. Readers can refer to Appendix B for details on the user simulator.

Results The main simulation results are reported in Table 1 and Figures 4 and 5. For each agent, we report its results in terms of success rate, average reward, and average number of turns (averaged over 5 repetitions of the experiments). Results show that the DDQ agents consistently outperform DQN with a statistically significant margin. Figure 4 shows the learning curves of different DDQ agents trained using different planning steps. Since the training of all RL agents started with RBS using the same rule-based agent, their performance in the first few epochs is very close. After that, performance improved for all values of K, but much more rapidly for larger values. Recall that the DDQ(K) agent with K=0 is identical to the DQN agent, which does no planning but relies on direct reinforcement learning only. Without planning, the DQN agent took about 180 epochs (real dialogues) to reach the success rate of 50%,

Agent	Epoch = 100			Epoch = 150			Epoch = 200		
	Success	Reward	Turns	Success	Reward	Turns	Success	Reward	Turns
DQN	.0000	-58.69	39.38	.4080	-5.730	30.38	.4545	0.350	30.38
DDQ(5)	.4620	00.78	31.33	.5637	15.05	26.17	.6000	19.84	26.32
DDQ(5, rand-init θ_M)	.3600	-11.67	31.74	.5500	13.71	26.58	.5752	16.84	26.37
DDQ(10)	**.5555**	14.69	25.92	**.6416**	25.85	24.28	**.7332**	38.88	20.21
DDQ(10, rand-init θ_M)	.5010	6.27	29.70	.6055	22.11	23.11	.7023	36.90	21.20

Table 2: The performance of different agents at training epoch = {100, 150, 200} in the human-in-the-loop experiments. The difference between the results of all agent pairs evaluated at the same epoch is statistically significant ($p < 0.01$). (Success: success rate)

and DDQ(10) took only 50 epochs.

Intuitively, the optimal value of K needs to be determined by seeking the best trade-off between the quality of the world model and the amount of simulated experience that is useful for improving the dialogue agent. This is a non-trivial optimization problem because both the dialogue agent and the world model are updated constantly during training and the optimal K needs to be adjusted accordingly. For example, we find in our experiments that at the early stages of training, it is fine to perform planning aggressively by using large amounts of simulated experience even though they are of low quality, but in the late stages of training where the dialogue agent has been significantly improved, low-quality simulated experience is likely to hurt the performance. Thus, in our implementation of Algorithm 1, we use a heuristic[5] to reduce the value of K in the late stages of training (e.g., after 150 epochs in Figure 4) to mitigate the negative impact of low-qualify simulated experience. We leave it to future work how to optimize the planning step size during DDQ training in a principled way.

Figure 5 shows that the quality of the world model has a significant impact on the agent's performance. The learning curve of DQN(10) indicates the best performance we can expect with a *perfect* world model. With a pre-trained world model, the performance of the DDQ agent improves more rapidly, although eventually, the DDQ and DDQ(rand-init θ_M) agents reach the same success rate after many epochs. The world model learning process is crucial to both the efficiency of dialogue policy learning and the final performance of the agent. For example, in the early stages (before 60 epochs), the performances of DDQ and DDQ(fixed θ_M) remain very close to each other, but DDQ reaches a success rate almost

[5]The heuristic is not presented in Algorithm 1. Readers can refer to the released source code for details.

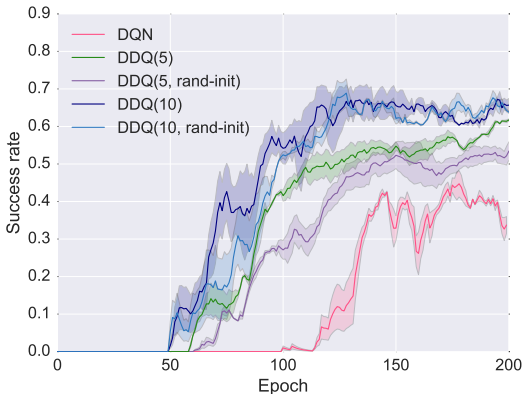


Figure 6: Human-in-the-loop dialogue policy learning curves in four different agents.

10% better than DDQ(fixed θ_M) after 400 epochs.

3.4 Human-in-the-Loop Evaluation

In this setting, five dialogue agents (i.e., DQN, DDQ(10), DDQ(10, rand-init θ_M), DDQ(5), and DDQ(5, rand-init θ_M)) are trained via RL by interacting with real human users. In each dialogue session, one of the agents was randomly picked to converse with a user. The user was presented with a user goal sampled from the corpus, and was instructed to converse with the agent to complete the task. The user had the choice of abandoning the task and ending the dialogue at any time, if she or he believed that the dialogue was unlikely to succeed or simply because the dialogue dragged on for too many turns. In such cases, the dialogue session is considered failed. At the end of each session, the user was asked to give explicit feedback whether the dialogue succeeded (i.e., whether the movie tickets were booked with all the user constraints satisfied). Each learning curve is trained with two runs, with each run generating 150 dialogues (and $K*150$ additional simulated dialogues when planning is applied). In total, we collected 1500 dialogue sessions for training all five agents.

The main results are presented in Table 2 and

Simulation Sample		Real User Sample	

movie-ticket booking user goal:
{
"request_slots": {
"ticket": "?"
"theater": "?"
"starttime": "?"
"date": "?"
}
}
	"constraint_slots": {
	"numberofpeople":"2"
	"moviename": "deadpool"
	"city": "seattle"
	}
		movie-ticket booking user goal:
		{
		"request_slots": {
		"ticket": "?"
		"theater": "?"
		"starttime": "?"
		}
		}
			"constraint_slots": {
			"date":"this weekend"
			"numberofpeople": "1"
			"moviename": "batman"
			}

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**	

Table 3: Two sample dialogue sessions by DQN and DDQ(10) agents trained at epoch 100: Left: simulated user experiments; Right: human-in-the-loop experiments. (*agt*: agent, *usr*: user)

Figure 6, with each agent averaged over two independent runs. The results confirm what we observed in the simulation experiments. The conclusions are summarized as below:

- The DDQ agent significantly outperforms DQN, as demonstrated by the comparison between DDQ(10) and DQN. Table 3 presents four example dialogues produced by two dialogue agents interacting with simulated and human users, respectively. The DQN agent, after being trained with 100 dialogues, still behaved like a naive rule-based agent that requested information bit by bit in a fixed order. When the user did not answer the request explicitly (e.g., `usr: which theater is available?`), the agent failed to respond properly. On the other hand, with planning, the DDQ agent trained with 100 real dialogues is much more robust and can complete 50% of user tasks successfully.
- A larger K leads to more aggressive planning and better results, as shown by DDQ(10) vs. DDQ(5).
- Pre-training world model with human con-

versational data improves the learning efficiency and the agent's performance, as shown by DDQ(5) vs. DDQ(5, rand-init θ_M), and DDQ(10) vs. DDQ(10, rand-init θ_M).

4 Conclusion

We propose a new strategy for a task-completion dialogue agent to learn its policy by interacting with real users. Compared to previous work, our agent learns in a much more efficient way, using only a small number of real user interactions, which amounts to an affordable cost in many non-trivial domains. Our strategy is based on the Deep Dyna-Q (DDQ) framework where planning is integrated into dialogue policy learning. The effectiveness of DDQ is validated by human-in-the-loop experiments, demonstrating that a dialogue agent can efficiently adapt its policy on the fly by interacting with real users via deep RL.

One interesting topic for future research is *exploration in planning*. We need to deal with the challenge of adapting the world model in a changing environment, as exemplified by the domain extension problem (Lipton et al., 2016). As pointed out by Sutton and Barto (1998), the general problem here is a particular manifestation of the conflict between exploration and exploitation. In a planning context, exploration means trying actions that may improve the world model, whereas exploitation means trying to behave in the optimal way given the current model. To this end, we want the agent to explore in the environment, but not so much that the performance would be greatly degraded.

Acknowledgments

We would like to thank Chris Brockett, Yun-Nung Chen, Michel Galley and Lihong Li for their insightful comments on the paper. We would like to acknowledge the volunteers from Microsoft Research for helping us with the human-in-the-loop experiments. This work was done when Baolin Peng and Shang-Yu Su were visiting Microsoft. Baolin Peng is in part supported by Innovation and Technology Fund (6904333), and General Research Fund of Hong Kong (12183516).

References

Pawel Budzianowski, Stefan Ultes, Pei-Hao Su, Nikola Mrksic, Tsung-Hsien Wen, Inigo Casanueva, Lina Rojas-Barahona, and Milica Gasic. 2017. Sub-domain modelling for dialogue management with hierarchical reinforcement learning. *arXiv preprint arXiv:1706.06210* .

Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2017. Towards end-to-end reinforcement learning of dialogue agents for information access. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 484–495.

Milica Gašić, Filip Jurčíček, Simon Keizer, François Mairesse, Blaise Thomson, Kai Yu, and Steve Young. 2010. Gaussian processes for fast policy optimisation of pomdp-based dialogue managers. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Association for Computational Linguistics, pages 201–204.

Milica Gašić, Filip Jurčíček, Blaise Thomson, Kai Yu, and Steve Young. 2011. On-line policy optimisation of spoken dialogue systems via live interaction with human subjects. In *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*. IEEE, pages 312–317.

Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. 2016. Continuous deep q-learning with model-based acceleration. In *International Conference on Machine Learning*. pages 2829–2838.

Dilek Hakkani-Tür, Gokhan Tur, Asli Celikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. 2016. Multi-domain joint semantic frame parsing using bi-directional RNN-LSTM. In *Proceedings of The 17th Annual Meeting of the International Speech Communication Association*.

Leonid Kuvayev and Richard S Sutton. 1996. Model-based reinforcement learning with an approximate, learned model. In *in Proceedings of the Ninth Yale Workshop on Adaptive and Learning Systems*. Citeseer.

Esther Levin, Roberto Pieraccini, and Wieland Eckert. 1997. Learning dialogue strategies within the markov decision process framework. In *Automatic Speech Recognition and Understanding, 1997. Proceedings., 1997 IEEE Workshop on*. IEEE, pages 72–79.

Jiwei Li, Alexander H Miller, Sumit Chopra, Marc'Aurelio Ranzato, and Jason Weston. 2016a. Dialogue learning with human-in-the-loop. *arXiv preprint arXiv:1611.09823* .

Xiujun Li, Zachary C Lipton, Bhuwan Dhingra, Lihong Li, Jianfeng Gao, and Yun-Nung Chen. 2016b. A user simulator for task-completion dialogues. *arXiv preprint arXiv:1612.05688* .

Xuijun Li, Yun-Nung Chen, Lihong Li, Jianfeng Gao, and Asli Celikyilmaz. 2017. End-to-end task-completion neural dialogue systems. In *Proceedings of the The 8th International Joint Conference on Natural Language Processing*. pages 733–743.

Zachary C Lipton, Jianfeng Gao, Lihong Li, Xiujun Li, Faisal Ahmed, and Li Deng. 2016. Efficient exploration for dialogue policy learning with bbq networks & replay buffer spiking. *arXiv preprint arXiv:1608.05081* .

Bing Liu and Ian Lane. 2017. Iterative policy learning in end-to-end trainable task-oriented neural dialog models. In *Proceedings of 2017 IEEE Workshop on Automatic Speech Recognition and Understanding*.

Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. 2015. Representation learning using multi-task deep neural networks for semantic classification and information retrieval .

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529–533.

Andrew W Moore and Christopher G Atkeson. 1993. Prioritized sweeping: Reinforcement learning with less data and less time. *Machine learning* 13(1):103–130.

Nikola Mrkšić, Diarmuid O Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2016. Neural belief tracker: Data-driven dialogue state tracking. *arXiv preprint arXiv:1606.03777* .

Alexandros Papangelis. 2012. A comparative study of reinforcement learning techniques on dialogue management. In *Proceedings of the Student Research Workshop at the 13th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 22–31.

Baolin Peng, Xiujun Li, Jianfeng Gao, Jingjing Liu, Yun-Nung Chen, and Kam-Fai Wong. 2017a. Adversarial advantage actor-critic model for task-completion dialogue policy learning. *arXiv preprint arXiv:1710.11277* .

Baolin Peng, Xiujun Li, Lihong Li, Jianfeng Gao, Asli Celikyilmaz, Sungjin Lee, and Kam-Fai Wong. 2017b. Composite task-completion dialogue policy learning via hierarchical deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 2221–2230.

Jing Peng and Ronald J Williams. 1993. Efficient learning and planning within the dyna framework. *Adaptive Behavior* 1(4):437–454.

Olivier Pietquin, Matthieu Geist, Senthilkumar Chandramohan, et al. 2011. Sample efficient on-line learning of optimal dialogue policies with kalman temporal differences. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*. volume 22, page 1878.

Olivier Pietquin and Helen Hastie. 2013. A survey on metrics for the evaluation of user simulations. *The knowledge engineering review* .

Sébastien Racanière, Théophane Weber, David Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adrià Puigdomènech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, et al. 2017. Imagination-augmented agents for deep reinforcement learning. In *Advances in Neural Information Processing Systems*. pages 5694–5705.

Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young. 2007. Agenda-based user simulation for bootstrapping a pomdp dialogue system. In *NAACL 2007; Companion Volume, Short Papers*. Association for Computational Linguistics, pages 149–152.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016a. Mastering the game of go with deep neural networks and tree search. *Nature* 529(7587):484–489.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. 2017. Mastering the game of go without human knowledge. *Nature* 550(7676):354.

David Silver, Hado van Hasselt, Matteo Hessel, Tom Schaul, Arthur Guez, Tim Harley, Gabriel Dulac-Arnold, David Reichert, Neil Rabinowitz, Andre Barreto, et al. 2016b. The predictron: End-to-end learning and planning. *arXiv preprint arXiv:1612.08810* .

Satinder Singh, Diane Litman, Michael Kearns, and Marilyn Walker. 2002. Optimizing dialogue management with reinforcement learning: Experiments with the njfun system. *Journal of Artificial Intelligence Research* 16:105–133.

Satinder P Singh. 1992. Reinforcement learning with a hierarchy of abstract models. In *Proceedings of the National Conference on Artificial Intelligence*. JOHN WILEY & SONS LTD, 10, page 202.

Pei-Hao Su, Milica Gasic, Nikola Mrksic, Lina Rojas-Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016a. Continuously learning neural dialogue management. *arXiv preprint arXiv:1606.02689* .

Pei-Hao Su, Milica Gasic, Nikola Mrksic, Lina Rojas-Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016b. On-line active

reward learning for policy optimisation in spoken dialogue systems. *arXiv preprint arXiv:1605.07669* .

Richard S Sutton. 1990. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the seventh international conference on machine learning*. pages 216–224.

Richard S Sutton and Andrew G Barto. 1998. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge.

Richard S Sutton, Csaba Szepesvári, Alborz Geramifard, and Michael P Bowling. 2012. Dyna-style planning with linear function approximation and prioritized sweeping. *arXiv preprint arXiv:1206.3285* .

Aviv Tamar, Yi Wu, Garrett Thomas, Sergey Levine, and Pieter Abbeel. 2016. Value iteration networks. In *Advances in Neural Information Processing Systems*. pages 2154–2162.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745* .

Jason D Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid code networks: Practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE* 101(5):1160–1179.

Tiancheng Zhao and Maxine Eskenazi. 2016. Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. *arXiv preprint arXiv:1606.02560* .

A Dataset Annotation Schema

Table 4 lists all annotated dialogue acts and slots in details.

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

Table 4: The data annotation schema

B User Simulator

In the task-completion dialogue setting, the entire conversation is around a user goal implicitly, but the agent knows nothing about the user goal explicitly and its objective is to help the user to accomplish this goal. Generally, the definition of user goal contains two parts:

- *inform_slots* contain a number of slot-value pairs which serve as constraints from the user.
- *request_slots* contain a set of slots that user has no information about the values, but wants to get the values from the agent during the conversation. ticket is a default slot which always appears in the *request_slots* part of user goal.

To make the user goal more realistic, we add some constraints in the user goal: slots are split into two groups. Some of slots must appear in the user goal, we called these elements as *Required slots*. In the movie-booking scenario, it includes moviename, theater, starttime, date, numberofpeople; the rest slots are *Optional slots*, for example, theater_chain, video_format etc.

We generated the user goals from the labeled dataset mentioned in Section 3.1, using two mechanisms. One mechanism is to extract all the slots (known and unknown) from the first user turns (excluding the greeting user turn) in the data, since usually the first turn contains some or all the required information from user. The other mechanism is to extract all the slots (known and unknown) that first appear in all the user turns, and then aggregate them into one user goal. We dump these user goals into a file as the user-goal database. Every time when running a dialogue, we randomly sample one user goal from this user goal database.