Budgeted Policy Learning for Task-Oriented Dialogue Systems

Zhirui Zhang[†] **Xiujun Li**[‡§] **Jianfeng Gao**[‡] **Enhong Chen**[†]
[†]University of Science and Technology of China
[‡]Microsoft Research AI [§]University of Washington
[†]zrustc11@gmail.com [†]cheneh@ustc.edu.cn
[‡]{xiul,jfgao}@microsoft.com

Abstract

This paper presents a new approach that extends Deep Dyna-Q (DDQ) by incorporating a Budget-Conscious Scheduling (BCS) to best utilize a fixed, small amount of user interactions (budget) for learning task-oriented dialogue agents. BCS consists of (1) a Poisson-based global scheduler to allocate budget over different stages of training; (2) a controller to decide at each training step whether the agent is trained using real or simulated experiences; (3) a user goal sampling module to generate the experiences that are most effective for policy learning. Experiments on a movie-ticket booking task with simulated and real users show that our approach leads to significant improvements in success rate over the state-of-the-art baselines given the fixed budget.

1 Introduction

There has been a growing interest in exploiting reinforcement learning (RL) for dialogue policy learning in task-oriented dialogue systems (Levin et al., 1997; Williams, 2008; Young et al., 2013; Fatemi et al., 2016; Zhao and Eskénazi, 2016; Su et al., 2016; Li et al., 2017; Williams et al., 2017; Dhingra et al., 2017; Budzianowski et al., 2017; Chang et al., 2017; Liu and Lane, 2017; Liu et al., 2018; Gao et al., 2019). This is a challenging machine learning task because an RL learner requires real users to interact with a dialogue agent constantly to provide feedback. The process incurs significant real-world cost for complex tasks, such as movie-ticket booking and travel planning, which require exploration in a large state-action space.

In reality, we often need to develop a dialogue agent with some fixed, limited budget due to limited project funding, conversational data, and development time. Specifically, in this study we measure *budget* in terms of the number of real user interactions. That is, we strive to optimize a dialogue agent via a fixed, small number of interactions with real users.

One common strategy is to leverage a user simulator built on human conversational data (Schatzmann et al., 2007; Li et al., 2016). However, due to design bias and the limited amounts of publicly available human conversational data for training the simulator, there always exists discrepancies between the behaviors of real and simulated users, which inevitably leads to a sub-optimal dialogue policy. Another strategy is to integrate planning into dialogue policy learning, as the Deep Dyna-Q (DDQ) framework (Peng et al., 2018), which effectively leverages a small number of real experiences to learn a dialogue policy efficiently. In DDQ, the limited amounts of real user experiences are utilized for: (1) training a *world model* to mimic real user behaviors and generate simulated experiences; and (2) improving the dialogue policy using both real experiences via direct RL and simulated experiences via indirect RL (planning). Recently, some DDQ variants further incorporate discriminators (Su et al., 2018) and active learning (Wu et al., 2019) into planning to obtain high-quality simulated experiences.

DDQ and its variants face two challenges in the fixed-budget setting. First, DDQ lacks any explicit guidance on how to generate highly effective real dialogue experiences. For example, the experiences in the state-action space that has not, or less, been explored by the dialogue agent are usually more desirable. Second, DDQ lacks a mechanism of letting a human (teacher) play the role of the agent to explicitly demonstrate how to drive the dialogue (Barlier et al., 2018). This is useful in the cases where the dialogue agent fails to respond to users in conversations and the sparse negative rewards fail to help the agent improve its dialogue policy. To this end, DDQ needs to be equipped

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 3742–3751
Florence, Italy, July 28 - August 2, 2019.

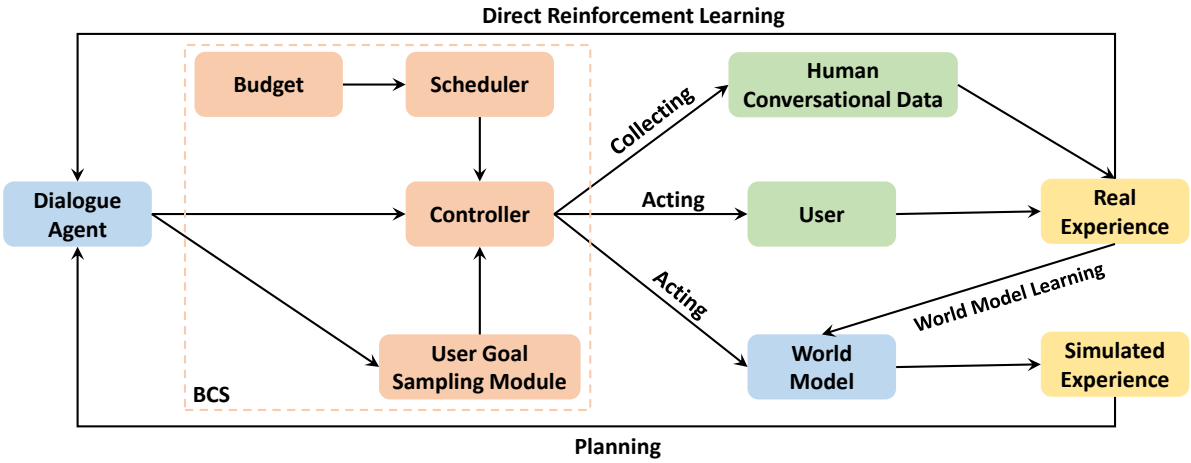

Figure 1: Proposed BCS-DDQ framework for dialogue policy learning. BCS represents the Budget-Conscious Scheduling module, which consists of a scheduler, a controller and a user goal sampling module.

with the ability to decide whether to learn from human demonstrations or from agent-user interactions where the user can be a real user or simulated by the world model.

In this paper, we propose a new framework, called Budget-Conscious Scheduling-based Deep Dyna-Q (BCS-DDQ), to best utilize a fixed, small number of human interactions (budget) for task-oriented dialogue policy learning. Our new framework extends DDQ by incorporating Budget-Conscious Scheduling (BCS), which aims to control the budget and improve DDQ's sample efficiency by leveraging active learning and human teaching to handle the aforementioned issues. As shown in Figure 1, the BCS module consists of (1) a Poisson-based global *scheduler* to allocate budget over the different stages of training; (2) a *user goal sampling module* to select previously failed or unexplored user goals to generate experiences that are effective for dialogue policy learning; (3) a *controller* which decides (based on the pre-allocated budget and the agent's performance on the sampled user goals) whether to collect human-human conversation, or to conduct human-agent interactions to obtain high-quality real experiences, or to generate simulated experiences through interaction with the world model. During dialogue policy learning, real experiences are used to train the world model via supervised learning (world model learning) and directly improve the dialogue policy via direct RL, while simulated experiences are used to enhance the dialogue policy via indirect RL (planning).

Experiments on the movie-ticket booking task with simulated and real users show that our approach leads to significant improvements in success rate over the state-of-the-art baselines given a fixed budget. Our main contributions are two-fold:

- We propose a BCS-DDQ framework, to best utilize a fixed, small amount of user interactions (budget) for task-oriented dialogue policy learning.
- We empirically validate the effectiveness of BCS-DDQ on a movie-ticket booking domain with simulated and real users.

2 Budget-Conscious Scheduling-based Deep Dyna-Q (BCS-DDQ)

As illustrated in Figure 2, the BCS-DDQ dialogue system consists of six 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 (Mrksic et al., 2017) for tracking dialogue state; (3) a dialogue policy that chooses the next action based on the current state and database results; (4) a model-based natural language generation (NLG) module for producing a natural language response (Wen et al., 2015); (5) a world model for generating simulated user actions and simulated rewards; and (6) the BCS module incorporating a global scheduler, a user goal sampling module and a controller, to manage the budget and select the most effective way to generate real or simulated experiences for learning a dialogue policy.

To leverage BCS in dialogue policy learning, we design a new iterative training algorithm, called BCS-DDQ, as summarized in Algorithm 1. It starts with an initial dialogue policy and an ini-

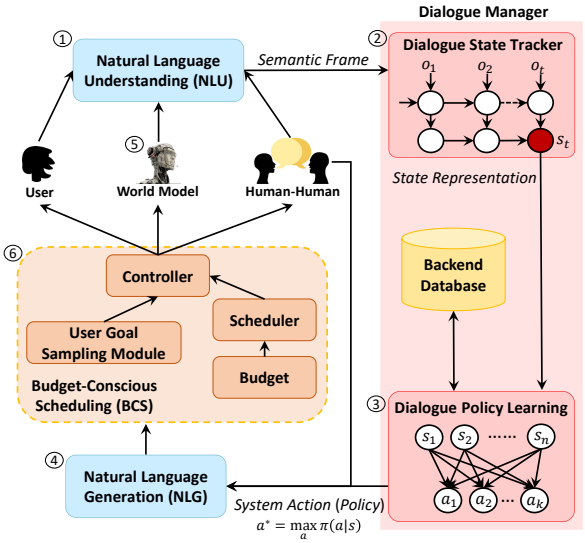

Figure 2: Illustration of the proposed BCS-DDQ dialogue system.

tial world model, both trained with pre-collected human conversational data. Given the total budget b and maximum number of training epochs m, the scheduler allocates the available budget b_k at each training step. Then, the user goal sampling module actively selects a previously failed or unexplored user goal g_u. Based on the agent's performance and the current pre-allocated budget, the controller chooses the most effective way, with cost $c^u = \{0, 1, 2\}$, to generate real or simulated experiences B^r/B^s for this sampled user goal. For convenience, the cost of different dialogue generation methods is defined as the number of people involved:

- cost $c^u = 2$ for collecting human-human demonstrated conversational data.
- cost $c^u = 1$ for conducting the interactions between human and agent.
- cost $c^u = 0$ for performing the interactions between world model and agent.

The generated experiences are used to update the dialogue policy and the world model. This process continues until all pre-allocated budget is exhausted. In the rest of this section, we detail the components of BCS, and describe the learning methods of the dialogue agent and the world model.

2.1 Budget-Conscious Scheduling (BCS)

As illustrated in Figure 2 and Algorithm 1, BSC consists of a budget allocation algorithm for the scheduler, an active sampling strategy for the user goal sampling module, and a selection policy for the controller.

Algorithm 1 BCS-DDQ for Dialogue Policy Learning

Input: The total budget b, the maximum number of training epochs m, the dialogue agent A and the world model W (both pre-trained with pre-collected human conversational data);

procedure TRAINING PROCESS
  while $k < m$ **do**
    $b_k \leftarrow$ Scheduler(b, m, k);
    repeat
      $g^u \leftarrow$ UserGoalSampler(A);
      $B^r, B^s, c^u \leftarrow$ Controller(g^u, b_k, A, W);
      $b_k \leftarrow b_k - c^u$;
    until $b_k \leq 0$
    Train the dialogue agent A on $B^r \cup B^s$
    Train world model W on B^r
  end while
end procedure

2.1.1 Poisson-based Budget Allocation

The global scheduler is designed to allocate budget $\{b_1, \ldots, b_m\}$ (where m is the final training epoch) during training. The budget allocation process can be viewed as a series of random events, where the allocated budget is a random variable. In this manner, the whole allocation process essentially is a discrete stochastic process, which can be modeled as a Poisson process. Specifically, at each training step k, the probability distribution of a random variable b_k equaling n is given by:

$$P\{b_k = n\} = \frac{\lambda_k^n}{n!} e^{-\lambda_k}, \lambda_k = \frac{m+1-k}{m}\lambda \quad (1)$$

The global scheduling in BCS is based on a *Decayed Possion Process*, motivated by two considerations: (1) For simplicity, we assume that all budget allocations are mutually-independent. The Poisson process is suitable for this assumption. (2) As the training process progresses, the dialogue agent tends to produce higher-quality dialogue experiences using the world model due to the improving performance of both the agent and the world model. As a result, the budget demand for the agent decays during the course of training. Thus, we linearly decay the parameter of the Poisson distribution so as to allocate more budget at the beginning of training.

In addition, to ensure that the sum of the allocated budget does not exceed the total budget b, we impose the following constraint:

$$\sum_{k=1}^{m} \mathbb{E}[b_k] = \sum_{k=1}^{m} \frac{m+1-k}{m} \lambda \leq b \qquad (2)$$

Using this formula, we can calculate the range of the parameter value: $\lambda \leq \frac{2b}{m+1}$. In our experiments, we set $\lambda = \frac{2b}{m+1}$ and sample b_k from the probability distribution defined in Equation 1.

2.1.2 Active Sampling Strategy

The active sampling strategy involves the definition of a user goal space and sampling algorithm.

In a typical task-oriented dialogue (Schatzmann et al., 2007), the user begins a conversation with a user goal g^u which consists of multiple constraints. In fact, these constraints correspond to attributes in the knowledge base. For example, in the movie-ticket-booking scenario, the constraints may be the name of the theater (`theater`), the number of tickets to buy (`numberofpeople`) or the name of the movie (`moviename`), and so on. Given the knowledge base, we can generate large amounts of user goals by traversing the combination of all the attributes, and then filtering unreasonable user goals which are not similar to real user goals collected from human-human conversational data. We then group the user goals with the same inform and request slots into a category. Suppose there are altogether l different categories of user goals. We design a Thompson-Sampling-like algorithm (Chapelle and Li, 2011; Eckles and Kaptein, 2014; Russo et al., 2018) to actively select a previously failed or unexplored user goal in two steps.

- Draw a number p_i for each category following $p_i \sim \mathcal{N}(f_i, \sqrt{\frac{l \ln N}{n_i}})$, where $\mathcal{N}$ represents the Gaussian distribution, f_i denotes the failure rate of each category estimated on the validation set, n_i is the number of samples for each category and $N = \sum_i n_i$.
- Select the category with maximum p_i, then randomly sample a user goal g_u in the category.

Using this method, user goals in the categories with higher failure rates or less exploration are more likely to be selected during training, which encourages the real or simulated user to generate dialogue experiences in the state-action space that the agent has not fully explored.

2.1.3 Controller

Given a sampled user goal g^u, based on the agent's performance on g^u and pre-allocated budget b_k, the controller decides whether to collect human-human dialogues, human-agent dialogues, or simulated dialogues between the agent and the world model. We design a heuristic selection policy of Equation 3 where dialogue experiences B are collected as follow: we first generate a set of simulated dialogues B^s given g^u, and record the success rate S_{g^u}. If S_{g^u} is higher than a threshold λ_1 (i.e. $\lambda_1 = 2/3$) or there is no budget left, we use B^s for training. If S_{g^u} is lower than a threshold λ_2 (i.e. $\lambda_2 = 1/3$) and there is still budget, we resort to human agents and real users to generate real experiences B^r_{hh}. Otherwise, we collect real experiences generated by human users and the dialogue agent B^r_{ha}.

$$(B, c^u) = \begin{cases} (B^s\ ,\ 0) & \text{if } S_{g^u} \geq \lambda_1 \text{ or } b_k = 0 \\ (B^r_{hh},\ 2) & \text{if } S_{g^u} \leq \lambda_2 \text{ and } b_k \geq 2 \\ (B^r_{ha},\ 1) & \text{otherwise} \end{cases} \qquad (3)$$

Combined with the active sampling strategy, this selection policy makes fuller use of the budget to generate experiences that are most effective for dialogue policy learning.

2.2 Direct Reinforcement Learning and Planning

Policy learning in task-oriented dialogue using RL can be cast as a Markov Decision Process which consists of a sequence of <state, action, reward> tuples. We can use the same Q-learning algorithm to train the dialogue agent using either real or simulated experiences. Here we employ the Deep Q-network (DQN) (Mnih et al., 2015).

Specifically, at each step, the agent observes the dialogue state s, then chooses an action a using an ϵ-greedy policy that selects a random action with probability ϵ, and otherwise follows the greedy policy $a = \arg\max_{a'} Q(s, a'; \theta_Q)$. $Q(s, a; \theta_Q)$ approximates the state-action value function with a Multi-Layer Perceptron (MLP) parameterized by θ_Q. Afterwards, the agent receives reward r, observes the next user or simulator response, and updates the state to s'. The experience (s, a, r, a^u, s') is then stored in a real experience buffer B^r[1] or simulated experience buffer B^s depending on the source. Given these experiences, we optimize the

[1] $B^r = \{B^r_{hh}, B^r_{ha}\}$

value function $Q(s, a; \theta_Q)$ through mean-squared loss:

$$\mathcal{L}(\theta_Q) = \mathbb{E}_{(s,a,r,s') \sim B^r \cup B^s}[(y - Q(s, a; \theta_Q))^2]$$
$$y = r + \gamma \max_{a'} Q'(s', a'; \theta_{Q'}) \tag{4}$$

where $\gamma \in [0, 1]$ is a discount factor, and $Q'(\cdot)$ is the target value function that is updated only periodically (i.e., fixed-target). The updating of $Q(\cdot)$ thus is conducted through differentiating this objective function via mini-batch gradient descent.

2.3 World Model Learning

We utilize the same design of the world model in Peng et al. (2018), which is implemented as a multi-task deep neural network. At each turn of a dialogue, the world model takes the current dialogue state s and the last system action a from the agent as input, and generates the corresponding user response a^u, reward r, and a binary termination signal t. The computation for each term can be shown as below:

$$\begin{aligned} h &= \tanh(W_h[s, a] + b_h) \\ r &= W_r h + b_r \\ a^u &= \text{softmax}(W_a h + b_a) \\ t &= \text{sigmoid}(W_t h + b_t) \end{aligned} \tag{5}$$

where all W and b are weight matrices and bias vectors respectively.

3 Experiments

We evaluate BCS-DDQ on a movie-ticket booking task in three settings: simulation, human evaluation and human-in-the-loop training. All the experiments are conducted on the text level.

3.1 Setup

Dataset. The dialogue dataset used in this study is a subset of the movie-ticket booking dialogue dataset released in Microsoft Dialogue Challenge (Li et al., 2018). Our dataset consists of 280 dialogues, which have been manually labeled based on the schema defined by domain experts, as in Table 1. The average length of these dialogues is 11 turns.

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 1: The dialogue annotation schema

Dialogue Agents. We benchmark the BCS-DDQ agent with several baseline agents:

- The **SL** agent is learned by a variant of imitation learning (Lipton et al., 2018). At the beginning of training, the entire budget is used to collect human-human dialogues, based on which the dialogue agent is trained.
- The **DQN** agent is learned by standard DQN At each epoch of training, the budget is spent on human-agent interactions, and the agent is trained by direct RL.
- The **DDQ** agent is learned by the DDQ method (Peng et al., 2018). The training process is similar to that of the DQN agent, differing in that DDQ integrates a jointly-trained world model to generate simulated experience which can further improve the dialogue policy. At each epoch of training, the budget is spent on human-agent interactions.
- The **BCS-DDQ** agent is learned by the proposed BCS-DDQ approach. For a fair comparison, we use the same number of training epochs m used for the DQN and DDQ agents.

Hyper-parameter Settings. We use an MLP to parameterize function $Q(\cdot)$ in all the dialogue agents (SL, DQN, DDQ and BCS-DDQ), with hidden layer size set to 80. The ϵ-greedy policy is adopted for exploration. We set discount factor $\gamma = 0.9$. The target value function $Q'(\cdot)$ is updated at the end of each epoch. The world model contains one shared hidden layer and three task-specific hidden layers, all of size 80. The number of planning steps is set to 5 for using the world model to improve the agent's policy in DDQ and BCS-DDQ frameworks. Each dialogue is allowed a maximum of 40 turns, and dialogues exceeding this maximum are considered failures. Other parameters used in BCS-DDQ are set as $l = 128, d = 10$.

Training Details. The parameters of all neural networks are initialized using a normal distribution with a mean of 0 and a variance of $\sqrt{6/(d_{row} + d_{col})}$, where d_{row} and d_{col} are the number of rows and columns in the structure (Glorot and Bengio, 2010). All models are optimized by RMSProp (Tieleman and Hinton, 2012). The mini-batch size is set to 16 and the initial learn-

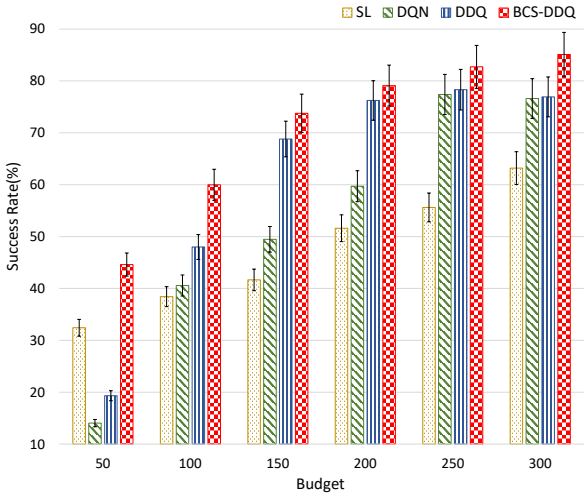

Figure 3: The success rates of different agents (SL, DQN, DDQ, BCS-DDQ) given a fixed budget ($b = \{50, 100, 150, 200, 250, 300\}$). Each number is averaged over 5 runs, each run tested on 50 dialogues.

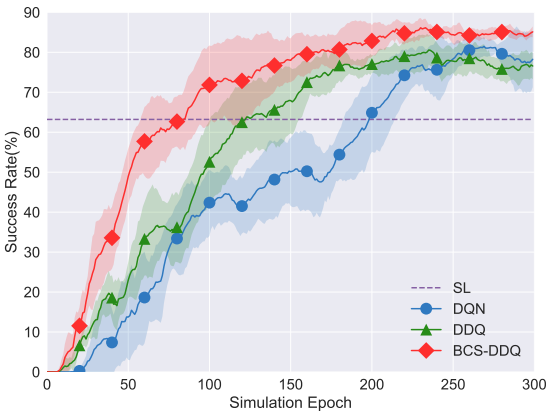

Figure 4: The learning curves of different agents (DQN, DDQ and BCS-DDQ) with budget $b = 300$.

ing rate is 5e-4. The buffer sizes of B^r and B^s are set to 3000. In order to train the agents more efficiently, we utilized a variant of imitation learning, Reply Buffer Spiking (Lipton et al., 2018), to pre-train all agent variants at the starting stage.

3.2 Simulation Evaluation

In this setting, the dialogue agents are trained and evaluated by interacting with the user simulators (Li et al., 2016) instead of real users. In spite of the discrepancy between simulated and real users, this setting enables us to perform a detailed analysis of all agents without any real-world cost. During training, the simulator provides a simulated user response on each turn and a reward signal at the end of the dialogue. The dialogue is considered successful if and only if a movie ticket is booked successfully and the information provided by the agent satisfies all the user's constraints (user goal). When the dialogue is completed, 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 allowed (40). To encourage shorter dialogues, the agent receives a reward of -1 on each turn.

In addition to the user simulator, the training of SL and BCS-DDQ agents requires a high-performance dialogue agent to play the role of the human agent in collecting human-human conversational data. In the simulation setting, we leverage a well-trained DQN agent as the human agent.

Main Results. We evaluate the performance of all agents (SL, DQN, DDQ, BCS-DDQ) given a fixed budget ($b = \{50, 100, 150, 200, 250, 300\}$). As shown in Figure 3, the BCS-DDQ agent consistently outperforms other baseline agents by a statistically significant margin. Specifically, when the budget is small ($b = 50$), SL does better than DQN and DDQ that haven't been trained long enough to obtain significant positive reward. BCS-DDQ leverages human demonstrations to explicitly guide the agent's learning when the agent's performance is very bad. In this way, BCS-DDQ not only takes advantage of imitation learning, but also further improves the performance via exploration and RL. As the budget increases, DDQ can leverage real experiences to learn a good policy. Our method achieves better performance than DDQ, demonstrating that the BCS module can better utilize the budget by directing exploration to parts of the state-action space that have been less explored.

Learning Curves. We also investigate the training process of different agents. Figure 4 shows the learning curves of different agents with a fixed budget ($b = 300$). At the beginning of training, similar to a very small budget situation, the performance of the BCS-DDQ agent improves faster thanks to its combination of imitation learning and reinforcement learning. After that, BCS-DDQ consistently outperforms DQN and DDQ as training progresses. This proves that the BCS module can generate higher quality dialogue experiences for training dialogue policy.

Agent	Epoch=100			Epoch=150			Epoch=200		
	Success	Reward	Turns	Success	Reward	Turns	Success	Reward	Turns
DQN	0.3032	-18.77	32.31	0.4675	2.07	30.07	0.5401	18.94	26.59
DDQ	0.4204	-2.24	27.34	0.5467	15.46	22.26	0.6694	32.00	18.66
BCS-DDQ	**0.7542**	43.80	15.42	**0.7870**	47.38	16.13	**0.7629**	44.45	16.20

Table 2: The performance of different agents at training epoch = {100, 150, 200} in the human-in-the-loop experiments. The differences between the results of all agent pairs evaluated at the same epoch are statistically significant ($p < 0.05$). (Success: success rate)

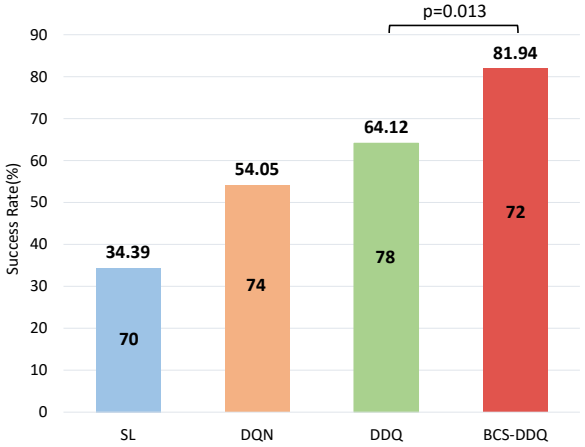

Figure 5: The human evaluation results for SL, DQN, DDQ and BCS-DDQ agents, the number of test dialogues indicated on each bar, and the p-values from a two-sided permutation test. The differences between the results of all agent pairs are statistically significant ($p < 0.05$).

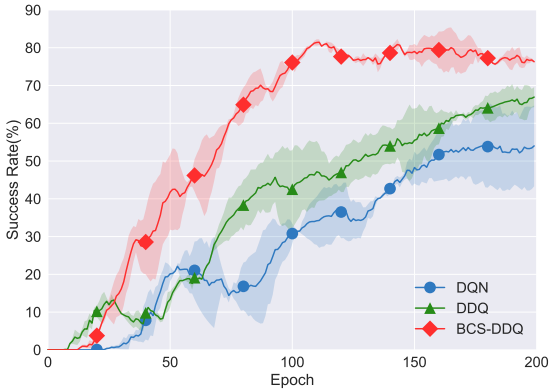

Figure 6: Human-in-the-Loop learning curves of different agents with budget $b = 200$.

3.3 Human Evaluation

For human evaluation, real users interact with different agents without knowing which agent is behind the system. At the beginning of each dialogue session, we randomly pick one agent to converse with the user. The user is provided with a randomly-sampled user goal, and the dialogue session can be terminated at any time, if the user believes that the dialogue is unlikely to succeed, or if it lasts too long. In either case, the dialogue is considered as failure. At the end of each dialogue, the user is asked to give explicit feedback about whether the conversation is successful.

Four agents (SL, DQN, DDQ and BCS-DDQ) trained in simulation (with $b = 300$) are selected for human evaluation. As illustrated in Figure 5, the results are consistent with those in the simulation evaluation (the rightmost group with budget=300 in Figure 3). In addition, due to the discrepancy between simulated users and real users, the success rates of all agents drop compared to the simulation evaluation, but the performance degradation of BCS-DDQ is minimal. This indicates that our approach is more robust and effective than the others.

3.4 Human-in-the-Loop Training

We further verify the effectiveness of our method in human-in-the-loop training experiments. In this experiment, we replace the user simulator with real users during training. Similar to the human evaluation, based on a randomly-sampled user goal, the user converses with a randomly-selected agent and gives feedback as to whether the conversation is successful. In order to collect human-human conversations during the training of the BCS-DDQ agent, human agents are interacting directly with real users through the dialogue system. In a dialogue session, the human agent has access to the conversation history, as well as the current search results from the knowledge base, before selecting each dialogue action in response to the real user. Each learning curve is trained with two runs, with each run assigning a budget of 200 human interactions.

The main results are presented in Table 2 and Figure 6. We can see that the BCS-DDQ agent consistently outperforms DQN and DDQ during the course of training, confirming the conclusion

Real User Dialogue Sample	
movie-ticket booking user goal:	
{	
"constraint_slots": {	"request_slots": {
"numberofpeople": "four"	"ticket":"?"
"moviename": "creed"	"theater": "?"
"city": "regency"	}
"date": "tomorrow"	
"starttime": "around noon"	
}	
}	
DDQ agent	
1	*usr*: I want to watch creed at around noon.
2	*agt*: around noon is available.
3	*usr*: I want four tickets please!
4	*agt*: Which city would you like?
5	*usr*: I want to watch at regency.
6	*agt*: Which theater would you like?
7	*usr*: Which theater is available?
8	*agt*: What date would you like to watch it?
9	*usr*: I want to set it up tomorrow.
10	*agt*: Which theater would you like?
11	*usr*: Which theater is available?
12	*agt*: Which city would you like?
13	*usr*: I want to watch at regency.
14	*agt*: Which theater would you like?
15	*usr*: Which theater is available?
16	*agt*: What date would you like to watch it?
⋮	
Failure:	Agent does not book the right ticket within the max turns.
BCS-DDQ agent	
1	*usr*: I want to watch creed at around noon.
2	*agt*: Which city would you like?
3	*usr*: I want to watch at regency.
4	*agt*: What date would you like to watch it?
5	*usr*: I want to set it up tomorrow.
6	*agt*: Which theater would you like?
7	*usr*: Which theater is available?
8	*agt*: **century eastport 16 is available.**
9	*usr*: Could you help me to book the tickets?
8	*agt*: How many tickets do you need?
9	*usr*: I want four tickets please!
10	*agt*: Great - I was able to purchase 4 tickets for you to see creed tomorrow at century eastport 16 theater in regency at around noon.
11	*usr*: Thank you.
12	*agt*: Thank you.
Success	

Table 3: Sample dialogue sessions by DDQ and BCS-DDQ agents trained at epoch 200 (with total budget $b = 200$) in the human-in-the-loop experiments: (*agt*: agent, *usr*: user)

drawn from the simulation evaluation. Besides, Table 3 shows example dialogues produced by two dialogue agents (DDQ and BCS-DDQ) interacting with human users respectively. We can see that DDQ agent fails to respond to the user question "which theater is available?", which lead to the repeated inquiry of theater information. By introducing human demonstrations for agent training,

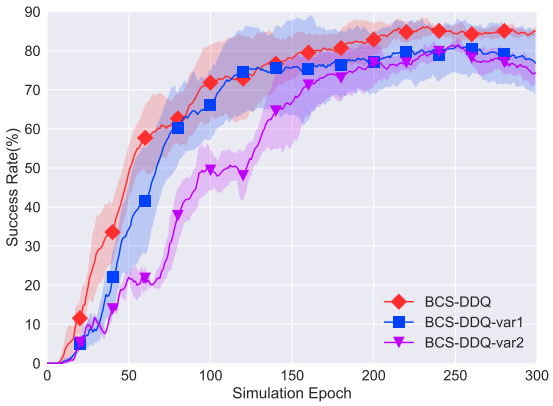

Figure 7: The learning curves of BCS-DDQ and its variants agents with budget $b = 300$.

BCS-DDQ agent can successfully respond to the available theater information.

3.5 Ablation Study

We investigate the relative contribution of the budget allocation algorithm and the active sampling strategy in BCS-DDQ by implementing two variant BCS-DDQ agents:

- The **BCS-DDQ-var1** agent: Replacing the decayed Poisson process with a regular Poisson process in the budget allocation algorithm, which means that the parameter λ is set to $\frac{b}{m}$ during training.
- The **BCS-DDQ-var2** agent: Further replacing the active sampling strategy with random sampling, based on the BCS-DDQ-var1 agent.

The results in Figure 7 shows that the budget allocation algorithm and active sampling strategy are helpful for improving a dialogue policy in the limited budget setting. The active sampling strategy is more important, without which the performance drops significantly.

4 Conclusion

We presented a new framework BCS-DDQ for task-oriented dialogue policy learning. Compared to previous work, our approach can better utilize the limited real user interactions in a more efficient way in the fixed budget setting, and its effectiveness was demonstrated in the simulation evaluation, human evaluation, including human-in-the-loop experiments.

In future, we plan to investigate the effectiveness of our method on more complex task-oriented dialogue datasets. Another interesting direction

is to design a trainable budget scheduler. In this paper, the budget scheduler was created independently of the dialogue policy training algorithm, so a trainable budget scheduler may incur additional cost. One possible solution is transfer learning, in which we train the budget scheduler on some well-defined dialogue tasks, then leverage this scheduler to guide the policy learning on other complex dialogue tasks.

5 Acknowledgments

We appreciate Sungjin Lee, Jinchao Li, Jingjing Liu, Xiaodong Liu, and Ricky Loynd for the fruitful discussions. We would like to thank the volunteers from Microsoft Research for helping us with the human evaluation and the human-in-the-loop experiments. We also thank the anonymous reviewers for their careful reading of our paper and insightful comments. This work was done when Zhirui Zhang was an intern at Microsoft Research.

References

Merwan Barlier, Romain Laroche, and Olivier Pietquin. 2018. Training dialogue systems with human advice. In *AAMAS*.

Pawel Budzianowski, Stefan Ultes, Pei hao Su, Nikola Mrksic, Tsung-Hsien Wen, Iñigo Casanueva, Lina Maria Rojas-Barahona, and Milica Gasic. 2017. Sub-domain modelling for dialogue management with hierarchical reinforcement learning. In *SIGDIAL*.

Cheng Chang, Runzhe Yang, Lu Chen, Xiang Zhou, and Kai Yu. 2017. Affordable on-line dialogue policy learning. In *EMNLP*.

Olivier Chapelle and Lihong Li. 2011. An empirical evaluation of thompson sampling. In *NIPS*.

Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2017. End-to-end reinforcement learning of dialogue agents for information access. In *ACL*.

Dean Eckles and Maurits Kaptein. 2014. Thompson sampling with the online bootstrap. *CoRR*, abs/1410.4009.

Mehdi Fatemi, Layla El Asri, Hannes Schulz, Jing He, and Kaheer Suleman. 2016. Policy networks with two-stage training for dialogue systems. In *SIGDIAL Conference*.

Jianfeng Gao, Michel Galley, and Lihong Li. 2019. Neural approaches to conversational ai. *Foundations and Trends® in Information Retrieval*, 13(2-3):127–298.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*.

Dilek Z. Hakkani-Tür, Gökhan Tür, Asli elikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. 2016. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. In *INTERSPEECH*.

Esther Levin, Roberto Pieraccini, and Wieland Eckert. 1997. Learning dialogue strategies within the markov decision process framework. In *ASRU 1997*, pages 72–79.

Xiujun Li, Yun-Nung Chen, Lihong Li, and Jianfeng Gao. 2017. End-to-end task-completion neural dialogue systems. In *IJCNLP*.

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

Xiujun Li, Sarah Panda, Jingjing Liu, and Jianfeng Gao. 2018. Microsoft dialogue challenge: Building end-to-end task-completion dialogue systems. *arXiv preprint arXiv:1807.11125*.

Zachary C Lipton, Jianfeng Gao, Lihong Li, Xiujun Li, Faisal Ahmed, and Li Deng. 2018. Efficient exploration for dialogue policy learning with bbq networks & replay buffer spiking. *AAAI*.

Bing Liu and Ian Lane. 2017. Iterative policy learning in end-to-end trainable task-oriented neural dialog models. *ASRU*, pages 482–489.

Bing Liu, Gökhan Tür, Dilek Z. Hakkani-Tür, Pararth Shah, and Larry P. Heck. 2018. Dialogue learning with human teaching and feedback in end-to-end trainable task-oriented dialogue systems. In *NAACL-HLT*.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature*, 518:529–533.

Nikola Mrksic, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve J. Young. 2017. Neural belief tracker: Data-driven dialogue state tracking. In *ACL*.

Baolin Peng, Xiujun Li, Jianfeng Gao, Jingjing Liu, and Kam-Fai Wong. 2018. Integrating planning for task-completion dialogue policy learning. In *ACL*.

Daniel J Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, Zheng Wen, et al. 2018. A tutorial on thompson sampling. *Foundations and Trends® in Machine Learning*, 11(1):1–96.

Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve J. Young. 2007. Agenda-based user simulation for bootstrapping a pomdp dialogue system. In *HLT-NAACL*.

Peihao Su, Milica Gasic, Nikola Mrksic, Lina Maria Rojas-Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve J. Young. 2016. Continuously learning neural dialogue management. *CoRR*, abs/1606.02689.

Shang-Yu Su, Xiujun Li, Jianfeng Gao, Jingjing Liu, and Yun-Nung Chen. 2018. Discriminative deep dyna-q: Robust planning for dialogue policy learning. In *EMNLP*.

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei hao Su, David Vandyke, and Steve J. Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *EMNLP*.

Jason D Williams. 2008. The best of both worlds: Unifying conventional dialog systems and pomdps. In *Ninth Annual Conference of the International Speech Communication Association*.

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 *ACL*.

Yuexin Wu, Xiujun Li, Jianfeng Gao, Jingjing Liu, and Yiming Yang. 2019. Switch-based active deep dyna-q: Efficient adaptive planning for task-completion dialogue policy learning. In *AAAI*.

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

Tiancheng Zhao and Maxine Eskénazi. 2016. Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. In *SIGDIAL Conference*.