Weighted User Goal Sampling for Dialog Policy Learning

Kai Yin*, Zhenyu Wang
School of Software Engineering, South China University of Technology, Guangzhou, 510006

*sekaiyin@mail.scut.edu.cn

Abstract. A deep reinforcement learning algorithm has been widely used for learning dialog policy in a task-oriented dialog system. Dialog agent collects training data and improves its policy by interacting with users. Interact with real users is time-consuming and not realistic. Consequently, we usually build a user simulator instead of a real user. At the beginning of each dialogue, the simulator will sample a user goal extract from training data. Then the simulator will communicate with the dialog agent to accomplish this user goal. The existing user simulator usually samples this user goal randomly. This will cause the dialog agent to waste a lot of time to learn what it already learned. To solve this problem, we propose two user goal weighting methods which give relatively large weight to the user goal the current dialog agent can’t accomplish. This lets the dialog agent pay more attention to those user goals. Experiment result in a movie-ticket booking task shows that the proposed weighted user goal sampling method can effectively accelerate the policy learning progress compares to the random sampling method.

Keywords: reinforcement learning, dialog policy, user simulator, the user goal sampling, sliding window.

1. Introduction

Task-oriented dialog systems focus on help users complete tasks under a specific domain through nature language. Those systems typically include four components: nature language understanding (NLU) for identifying user intent and extract slot value pairs associated with this task; dialog state tracking (DST) for keep record and update the dialog state; dialog policy (DP) for choosing the next system action based on the dialog state from DST; nature language generation (NLG) for transforming the system action into nature language [1,2].

Dialog policy can be viewed as a sequential decision-making problem and solved with reinforcement learning methods [3]. The dialog agent learning its policy through communication with a user simulator [4, 5, 6, 7, 8]. The simulator communicates with the dialog agent in order to accomplish a specific user goal. The user's goal is to extract from training data and is used to indicate the user’s demands for the task. The simulator will sample a user goal before the dialogue. The existing user simulator usually uses a random sampling strategy, so the dialog agent will spend a lot of time to learn the goal that it already mastered.

To alleviate this problem, we propose two user goal weighting method to give each user goal a different weight. Then the simulator will sample user goals based on the goal weight. The contribution of this work is as follows: 1) We propose two user goal weighting methods which can automatically ad-
just the weight of each user goal based on the ability of the dialog agent. The goal weight is further used to encourage the dialog agent to explore and learn what it has not mastered. 2) We evaluate the effectiveness of our method on a movie-ticket booking domain.

2. Related work
Dialog policy is usually modeled as a markov decision process (MDP) [9, 10], which can be represented as a tuple of \((S, A, P, R, \gamma)\). \(S\) is state set, in a dialog system, it represents the current dialog state including last user action, last system action, information about slots, result from knowledge base and so on. \(A\) is an action set to represent all the actions that the dialog agent can execute. \(P\) is the transition probability between states. \(R\) is a reward function that gives a reward signal after the dialog agent executes an action. \(\gamma\) is the discount factor for feature reward.

Deep Q-Network (DQN) [11] is generally used to solve reinforcement learning problem. It applies a deep neural network to approximate the Q value of each action \(a\) in state \(s\) as \(Q(a; s, \theta)\), where \(\theta\) is the network’s parameters. The policy of DQN can define as \(\pi_Q(s) = \arg \max_{a \in A} Q(s, a; \theta)\) that choose the action with the biggest Q value. During policy learning, we can also apply a \(\epsilon -\) greedy policy. It selects a random action with probability \(\epsilon\) otherwise selects action according to \(\pi_Q(s)\). The objective of DQN is to minimize the TD-error shown in formula (1):

\[
L = E_{s,a,s',r} [(Q(a, s; \theta) - y)^2]
\]  

where \(y = r + \gamma \max_{a' \in A} Q(s', a'; \theta)\).

Training DQN agents need to collect data by interact with the environment. To better reuse the collected data a technique called experience replay [12] is proposed. The dialogue data will store in the replay buffer in the form of \((s, a, s', r)\). When update DQN’s parameters, a mini-batch of data is sampled from the replay buffer to calculate the gradient.

3. User goal weighting methods
The main idea of goal weighting is to increase the sampling probability to the user goal the current system can’t accomplish. And decrease the sampling probability to the user goal the current system can accomplish. This gives dialog agents more opportunity to explore and learn what it hasn’t mastered. Fig 1 shows the framework of our proposed method.

![Fig. 1 Illustration of the proposed weighted user goal sampling framework](image)

3.1. Global weight
In the global weight method, we use a global array of size \(N\) to record the weight of each user goal during the whole training process, where \(N\) is the number of user goals. The weight of each user goal
is initially initialized to 1. During the training process, the dialog agent tries to accomplish the user goal sampled by the user simulator. At the end of each dialogue if the dialog agent successfully accomplishes this goal we minus 0.1 of its weight otherwise we plus 0.1 to its weight. We can use formula (2) to calculate the sampling probability of each goal, where $w_i$ is the weight of user goal $i$ and $P_i$ is the sample probability of goal $i$.

$$P_i = \frac{w_i}{\sum_{j=1}^{N} w_j}$$

(2)

To avoid the weight goes too big or too small, we set 0.1 as the minimum value of weight and 2.0 as the maximum value of weight.

### 3.2. Local sliding window weight

The global weight method records the mastery of the dialog agent to each user goal globally, which could not accurately represent the ability of the current dialog agent. To solve this problem, we incorporate a sliding window to record the user goal weight in the nearest $K$ training epoch. $K$ is the size of the sliding window. The main idea of this method is the weight information closer to the current training epoch can represent the agent's mastery of each user goal more accurately.

In this method, an $N \times K$ two-dimensional array (sliding window) is initialized and the initial weight is set to 1. During the $i$-th training epoch, formula (3) is used to calculate the current window index. Then the goal weight in this window to 1 is reset to 1. After that, this window will be used to record the goal weight of the current training epoch. We use the average weight of $K$ windows to guide the goal sample as shown in formula (4), where $w_{i,k}$ is the goal weight in window $k$.

$$\text{index} = i \% K$$

(3)

$$P_i = \frac{\sum_{k=1}^{K} w_{i,k}}{\sum_{j=1}^{N} \sum_{k=1}^{K} w_{j,k}}$$

(4)

### 4. Experiment

We implement our method in an open-source user simulator [13] which design for a movie-ticket booking task. The dialog agent aims to help users booking a movie ticket that meets the requirements of users. The ontology of this task is the list in tab 1.

| Annotation       | Intent                  | Slot                  |
|------------------|-------------------------|-----------------------|
| request, inform, deny, confirm question, Intent confirm answer, greeting, closing, not sure, multiple choice, thanks, welcome | city, closing, date, distance constraints, greeting, movie name, number of people, price, start time, state, task complete, theater, theater chain, ticket, video format, zip |

**Tab.1 The ontology of the task**

Deep Q Network is selected as the dialog policy learning algorithm, which is a multi-layer perceptron (MLP) with a single hidden layer. tanh is used as the activation function, and the hidden layer size is set to 80. During policy learning we apply $\epsilon$-greedy strategy for exploration, $\epsilon$ is set to 0.15 and decay to 0.05 finally. The discount factor $\gamma$ for feature reward is set to 0.95. The size of the
replay buffer is set to 20000. For two goal weighting methods we set the maximum weight to 2.0 and set the minimum weight to 0.1.

The input of DQN is the current encoding of dialog state, which mainly contains: 1) one-hot encoding of the last action from the system and mentioned slots; 2) on the hot encoding of the last action from the user and mentioned slots; 3) a scalar representation indicating the number of results from knowledge base; 4) a scalar representation of current dialogue turns.

We compare three goal sample strategy which are random sample (RS), global weight sample (GWS), and local sliding window weight sample (LSWS).

The dialogue success rate is used to evaluate the performance of three goal sampling methods. If and only if the system helps the user booking a movie ticket that meets the constraints in the user goal and answers the corresponding requests in the user goal, we think the dialogue is successful. All of the results below are the average of five runs. As shown in Tab.2, at epoch 100 the LSWS-10 method (10 is the sliding window size) achieve a success rate of 0.665 while the RS method took more than 300 epochs and GWS took about 200 epochs to reach a similar success rate. Fig 2 shows the learning curve of three methods, as we can see GWS and LSWS method can effectively accelerate the dialog policy learning compare to the RS method.

| Method    | Epoch=100 | Epoch=200 | Epoch=300 | Epoch=400 |
|-----------|-----------|-----------|-----------|-----------|
| RS        | 0.447     | 0.574     | 0.653     | 0.677     |
| GWS       | 0.449     | 0.660     | **0.722** | 0.769     |
| LSWS-10   | **0.665** | 0.731     | 0.703     | **0.795** |

Tab.2 Result of success rate for the different method at epoch 100-400

![Learning Curve](image)

Fig.2 The learning curve of RS, GWS and LSWS-10
We further test the influence of different window sizes. We choose window size 5, 10, and 20, the learning curve is shown in fig 3. We can see LSWS-10 outperform LSWS-5 and LSWS-20 between about 100 and 200 epochs, this may be due to the instability of the policy at this time. Too small or too big window size can’t represent the ability of current policy accurately. But after that when the policy goes stable, the influence of window size is not so important.

![Learning Curve](image)

**Fig.3** The learning curve of LSWS with different window size 5, 10 and 20

5. Conclusion
In this paper, we propose two user goal weighting methods to give different weights to each user goal based on the ability of the current dialog agent. The main idea of the two user goal weighting methods is to encourage the dialog agent to explore and learn the goal it has not mastered. We implement the methods and evaluate their effectiveness on a movie-ticket booking task. The result shows the sampling method based on user goal weight can effectively accelerate policy learning.

References

[1] Hongshen C, Xiaorui L, Dawei Y and Jiliang T 2017 *J.SIGKDD Explor* A survey on dialogue systems: recent advances and new frontiers 19(2) pp25-35.
[2] Zhang Z, Takanobu R, Zhu Q, Minlie H and Xiaoyan Z 2020 *J.Science China Technological Sciences* Recent advances and challenges in task-oriented dialog systems pp1-17.
[3] Singh S, Litman D, Kearns M and Walker M 2002 *J.Journal of Artificial Intelligence Research* Optimizing dialogue management with reinforcement learning: experiments with the njfun system 16 pp105-133.
[4] Schatzmann J, Thomson B, Weilhammer K, Hui Y and Young S 2007 *Human Language Technology Conf. of the North American Chapter of the Association of Computational Linguistics* Agenda-based user simulation for bootstrapping a POMDP dialogue system pp149-152.
[5] Chandramohan S, Geist M, Lefevre F and Pietquin O 2011 *Interspeech 2011,12th Annual Conf. of the Int. Speech Communication Association* User simulation in dialogue systems using inverse reinforcement learning pp1025-1028.
[6] Gašić M, Jurčiček F, Thomson B, Kai Y and Young S 2011 *IEEE Workshop on Automatic Speech Recognition & Understanding* On-line policy optimisation of spoken dialogue systems via live interaction with human subjects pp312-317
[7] Baolin P, Xiujun L, Jianfeng G, Jingjing L Kam-Fai and Shang-Yu S 2018 Proc. of the 56th Annual Meeting of the Association for Computational Linguistics Deep dyna-q Integrating planning for task-completion dialogue policy learning pp2182–2192

[8] Scheffler K and Young S 2002 Proc. of the second Int. Conf. on Human Language Technology Research Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning pp12-19

[9] Young S, Gašić M, Thomson B, and Jason D 2013 J.Proc. of the IEEE Pomdp-based statistical spoken dialog systems: a review 101(5) pp1160-1179

[10] Bellman R 1957 J.Journal of mathematics and mechanics A Markovian decision process 6 pp679-684

[11] Mnih V, Kavukcuoglu K, Silver D and et al 2015 Nat. Human-level control through deep reinforcement learning 518(7540) pp529-533

[12] Long-Ji L 1992 J.Mach. Learn. Self-improving reactive agents based on reinforcement learning planning and teaching 8 pp293–321

[13] Xiujun L, Lipton Z C, Dhingra B, Lihong L, Jianfeng G and Yun-Nung C 2016 CoRR A user simulator for task-completion dialogues abs/1612.05688