Smart Health (2022)

Contents lists available at ScienceDirect

Smart Health

journal homepage: www.elsevier.com/locate/smhl

Adaptation of a robotic dialog system for medication reminder in elderly care

Zhidong Su[a], Weihua Sheng[a,*], Guanci Yang[b], Alex Bishop[a], Barbara Carlson[c]

[a] *Oklahoma State University, Stillwater, 74078, Oklahoma, USA*
[b] *Guizhou University, Guiyang, 550025, Guizhou, China*
[c] *University of Oklahoma Health Sciences Center, Oklahoma City, 73104, Oklahoma, USA*

ARTICLE INFO

Keywords:
Human-robot interaction
Reinforcement learning
Social robot
Elderly care
Dialog adaptation

ABSTRACT

Social robots can assist older adults in their daily life. Verbal conversation is a natural and convenient way for older adults to interact with social robots. However, most of the existing conversation-based robot services, such as medication reminders, are rule-based systems. These systems require many hand-crafted rules and a significant amount of expert knowledge, therefore they cannot adapt to older adults' characteristics and dialog history. There are many reinforcement learning (RL) based methods for task-oriented dialogues, but they mainly focus on completing the tasks through text-based conversations. Those methods cannot be directly used for elderly care applications involving human-robot interactions (HRI). Considering the above shortcomings, we proposed a dialog system adaptation method (DSAM) for social robots. The DSAM is based on reinforcement learning which considers the characteristics of older adults, the dialog history and user preference to adapt the dialog policy and improve the dialog module. We implemented DSAM in our custom-made ASCCBot social robot. To evaluate DSAM, we firstly tested the dialog agent which was trained by a user simulator with different settings. The results show that the obtained agent achieves a good result with the desired dialog flow compared to the baseline agent. Based on the obtained dialog policy, the adaptation process is evaluated. The results show that with a good success rate, the number of dialog turns is decreased and the NLU module performance is improved by the adaptation process, which proves the effectiveness of DSAM. We also tested DSAM with human subjects. The results show that the average adaptation success rate is 94.7% and the preference distance reaches 0 after 6 rounds of adaptation while creating reminders successfully with a limited amount of user feedback.

1. Introduction

The increasing elderly population is creating a growing burden for the younger generation Zhao & Li (2018). Social robots can offer a solution to this problem. Equipped with cameras, microphones and other advanced sensors, social robots can provide various services to improve the well-being of older adults and reduce the burden on their caregivers. Social robots can carry out tasks like diet management Su et al. (2020), companionship Saint-Aime et al. (2007), negative emotion management Pham et al. (2021), fall detection Liang et al. (2021) and clinical screening interview Manh Do et al. (2021) to improve older adults' well-being through verbal conversation. In our previous work Su et al. (2021), we developed a conversation-based medication management robot for older adults.

Although conversational social robots can assist older adults with their daily life, there are still some drawbacks. Firstly, most of the existing conversational robots are rule-based and require many hand-crafted rules and a significant amount of expert knowledge. Secondly, older adults

*Corresponding author:
e-mail: weihua.sheng@okstate.edu (Weihua Sheng)

http://dx.doi.org/10.1016/j.smhl.xxxx.xx.xxx
Received 9 June 2022; Accepted 12 August 2022; Available online TODO
2352-6483/

tend to have age-related health issues like memory loss Perlmutter et al. (1987) and hearing impairment Lin et al. (2013). Some older adults may not be able to pronounce every word clearly due to tooth loss Bitencourt et al. (2019). These robots failed to adapt to older adults' special needs. For example, when creating medication reminders, if the older adult cannot express the key information clearly, the social robot will not obtain the correct entities to create a reminder. Thirdly, existing social robots are not able to learn from the conversations with older adults or improve their abilities in natural language understanding (NLU) and dialog management.

Assisting older adults to create reminders or carry out clinical screening interviews is regarded as task-oriented conversations. The rule-based methods Cassell et al. (1994), recurrent neural network (RNN) Abro et al. (2019) -based methods and reinforcement learning (RL) Koo et al. (2019)-based methods are commonly used to construct task-oriented conversations. The RL-based methods can adapt the dialog policy gradually according to the conversations with users, which is a desirable feature in human-robot interaction (HRI). Some tasks like booking movie tickets, flight tickets and restaurant reservations have been implemented using RL-based algorithms Lu et al. (2021). However, most of the researches using RL-based methods for task-oriented dialogues mainly focus on finishing the tasks through text-based conversations. Those methods cannot be directly used for many elderly care applications as older adults have the above mentioned cognition impairments and special needs. Therefore we need a new RL-based dialog policy which is capable of adapting to older adults' characteristics.

In this paper, we proposed a dialog system adaptation method (DSAM) to create medication reminders, which considers the characteristics of older adults and the user preference in a reinforcement learning framework. This method can customize the dialog actions of the dialog agent to fit older adults' cognitive capacity and learn from the conversation and user feedback to optimize the dialog module. The main contributions of this paper are three folds. First, this is the first work that applies the RL method to the conversation-based medication reminder for older adults, which considers the real-world conversation situations instead of the pure text-based dialogues. Second, the proposed DSAM is able to optimize the dialog policy considering older adults' characteristics and user preference. Third, we tested the proposed method using both a user simulator and real human subjects. The preliminary results show that: 1) The obtained basic agent can handle realistic medication reminder conversations with a good success rate; 2) For the adaptation process, the number of dialog turns is decreased and the NLU module performance is improved without sacrificing the success rate in the simulation test. In the human subjects test, the adaptation success rate is 94.7% and the preference distance reaches 0 after 6 rounds of conversation while creating reminders successfully.

The rest of this paper is organized as follows. Section II presents the related work. Section III provides the system overview. Section IV details the proposed method. Section V gives the experimental results. Finally, Section VI concludes this paper and discusses the future work.

2. Related Work

Conversational social robots have been developed to help people in their daily life. Manh Do et al. (2021) proposed a clinical screening interview system for older adults. In their system, verbal conversation allowed a social robot to ask older adults various questions in order to detect mental and physical health problems. Lio et al. (2020) developed a Q&A dialog module for older adults using two robots. Their work reduced the side effect of speech recognition failures during the conversation. Akiyoshi et al. (2021) proposed a conversational robot to improve users' mood, in which a column method and a self-schema estimation method were employed to make the users aware of their negative emotions. Su et al. (2020) developed a social robot that can extract the diet information automatically from conversations among family members, which can help doctors develop efficient dietary therapy for the patients. Although the above works can provide meaningful assistance to older adults, they lack adaptation in the sense that the dialog systems cannot learn or be improved over the time.

Reinforcement learning algorithms enable an agent to learn from the interactions with the environment. Qureshi et al. (2018) used the reinforcement learning algorithm to make social robots master human-like social skills through real-world interaction. An event detector and predictor was proposed as the intrinsically motivated reward to improve the agent. Ritschel et al. (2019) adapted a social robot's linguistic style based on an older adult's feedback obtained from buttons. They utilized the multi-armed bandit algorithm Vermorel & Mohri (2005) which is a simplified reinforcement learning algorithm. In order to facilitate children engagement and long-term learning gains using a storytelling robot, Park et al. (2019) utilized the Q-learning method to personalize a policy based on children's verbal and nonverbal emotions. By adapting the policy, the social robot can choose a story with proper lexical and syntactic complexity. To build a task-oriented conversation system, Li et al. (2016) proposed a user simulator to speed up the training and reduce human work load. However, the trained agent cannot handle complex conversations due to the design of the reward function. Shah et al. (2016) added an action-specific feedback to the agent action reward instead of just using the task-level reward which can help speed up the convergence by reducing the reward sparsity and also minimize human involvement.

Robot action adaptation aims to optimize the action to cater for user's characteristics or preference after interacting with users. Adaptation can be achieved implicitly from the interaction history or explicitly from the human feedback. Rudary et al. (2004) built an adaptive reminder system using temporal constraint reasoning with RL which can remind users' daily activities. The simulation results show that it can adapt to the user profile and the short-term and long-term changes. However, it is not conversation based and only considers when and how to remind users properly. How to create medication reminders in a user-friendly way is not included. Ferreira & Lefèvre (2015) proposed a socially inspired reward function for human-robot dialog. The social rewards are defined based on dialog status and social cues like dialog actions. However, when it comes to dialog adaptation, the parameters of the reward functions have to be manually varied to fit the user profiles. Wang et al. (2020) leveraged the policy shaping and reward shaping method to learn from human demonstrations to improve the adaptation efficiency. A potential function based on multi-variate Gaussian is used to learn a reward function from the state-action pairs. Based on the reward function, the reward sparsity issue can be mitigated to some extent and the learning efficiency can be improved, but when it comes to real human-robot interaction, the proposed method only involves the reward from human demonstrations and the task level reward, which is not sufficient to handle the real human-robot environment. How to use reinforcement learning algorithms for medication management and user adaptation is still an open problem.

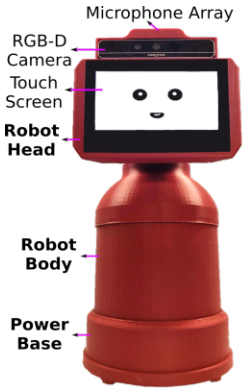


Fig. 1: The ASCCBot social robot platform.

3. System Overview

In this section, we firstly introduce the custom-built social robot. Secondly, the overall architecture of the adaptive robotic dialog system is proposed. Then the formulation of the RL-based medication reminder is developed. The Deep Q-Network used in this paper is introduced at the end.

3.1. The Social Robot

Fig. 1 shows the ASCCBot social robot built our lab Manh Do et al. (2021), based on which we implemented the proposed system. The ASCCBot is able to engage in conversation with humans. It has three parts: head, body and base. The head has two degrees of freedom, which enables it to track human face and turn to the direction of sound based on sound localization. Its face is a touch screen connected to an embedded ARM-based minicomputer. Multiple animated facial expressions are realized on the robot face to better engage the users. The robot features an auditory system which has four microphones to implement sound localization and speech recognition. It also has a vision system which uses a RGB-D camera for face detection. An Intel NUC with a Core i5 processor is the main computer. The robot is able to play news and music, report weather, tell jokes and quotes, play games like rock-paper-scissors. The robot can recognize people's face and take photos. When an emergency situation occurs, such as a fall, the robot will contact family members or caregivers. The caregivers can then operate the robot remotely to assess the situation and take further actions. In addition, various functions such as pain evaluation, mood and loneliness detection, as well as cognitive assessment are implemented in the robot.

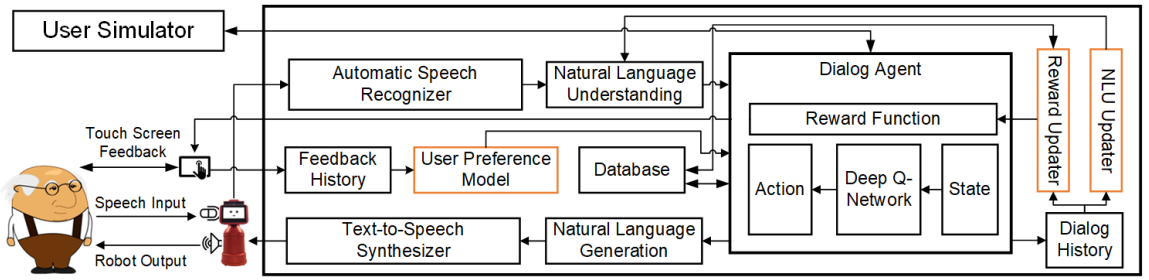


Fig. 2: The architecture of the adaptive robotic dialog system. The three orange rectangles show the three learning modules

3.2. System Architecture

In order to enable the social robot to adapt its behaviors to older adults, an adaptive robotic dialog system architecture is proposed as shown in Fig. 2. This system has two parts: a user simulator that interacts with the dialog agent, and an adaptation module.

1) *User Simulator* : The user simulator has two functions. Firstly, at the beginning, it is used to interact with the dialog agent to train a basic dialog policy using Deep Q-Network (DQN) Mnih et al. (2015). It is unrealistic to ask users to interact with the dialog agent from the beginning since it requires a large number of interaction epochs to get an acceptable performance. We call it the "warm start" stage. Secondly, during the human-robot interaction stage, based on the statistical result of the dialog history and the user feedback, the user simulator is utilized to adapt the dialog policy to the user's characteristics.

2) *Adaptation Module* : The adaptation module consists of basic conversation modules, a dialog agent and three learning modules as shown in the orange rectangle in Fig. 2. The basic conversation modules include the Automatic Speech Recognizer (ASR) module, Natural Language Understanding (NLU) module, Natural Language Generation (NLG) module and Text-to-Speech Synthesizer (TTS) module. These modules can

parse the user's input to the dialog agent and take the agent action as the feedback to the users. The dialog agent generates an action in response to the user's query. Based on the dialog history, the Reward Updater modifies the reward for different actions to control the dialog flow while the NLU Updater expands the NLU module's dictionary to enhance the slot filling ability. Users can directly talk to the robot. The created reminders are stored in the Database. The touch screen can display the created reminders. If the agent action can not cater for the users' preference, the users can select a desired action from a list of suggested actions shown on the touch screen. The feedback is utilized to train a User Preference Model (UPM). After updating the NLU module, the UPM module and the reward function, the user simulator is employed to talk to the agent to train an adapted dialog policy, which is utilized in the next round of human-robot conversation.

3.3. Problem Formulation

The dialog agent decides which action it should take based on user's input and the dialog history, which can be regarded as a Markov Decision Process (MDP) Levin et al. (1997) which can be described as a tuple $\{S, A, T, R, \gamma\}$. S is the state set. A state $s \in S$ includes the latest agent and user actions, the status of slots, the requested and informed slots and other information that can help the agent make decisions. A is the action set. The agent can perform an action $a \in A$ at each turn. T is the state transition probability matrix. R is the reward function to evaluate agent actions. $\gamma \in [0,1]$ is the discount factor. The cumulative reward with the discount factor is $R = \sum_{t=0}^{+\infty} \gamma^t r_t$, where t is time step and $r_t = R(s_t, a_t)$. The goal of MDP is to derive an optimal policy which can maximize the cumulative reward. The policy function π is a probability density function and maps the state to an action $\pi : S \rightarrow A$.

The state and agent actions used by the dialog agent are listed in Table 1, while the abbreviation of the agent or user actions are show in Table 2. The four agent actions REQ, INF, AC and REP combined with the 6 slots result in 24 actions. Therefore, the agent has a total of 29 actions to choose from. For the user actions, the REQ, INF and MO action combined with the 6 slots result in 18 actions. There are a total of 27 actions in the user action space. The agent actions are determined based on the characteristics of human-robot interaction and the previous dialog history. For example, because of the environment noise, speech recognition error, limited NLU ability and user's unclear voice, sometimes the system cannot get the correct key entities. Therefore, the agent will perform the action AC to obtain user's confirmation. The action INF and REQ are to tell the user the obtained information and request new slots. REP is the response to user's AR action. SU informs the user all the slots that the agent has obtained so far. CL stops the conversation.

Table 1: State, actions and slots.

State	**Agent Action**	**User Action**	**Slot**
user action	request	inform	patient
user inform slot	inform	request	medicine
agent act	ask for confirmation	confirm query	dosage
agent request slot	repeat	deny query	start time
agent inform slot	summary	ask to repeat	end time
turn number	close	stop conversation	remind time
filled slots	greeting	long time no reply	
	thanks	greeting	
	ask to modify	deny	
		summary	
		modify	
		thanks	
		other actions	

3.4. Deep Q-Network

We use DQN to learn the dialog policy. DQN is a deep neural network that approximates the optimal action-value function, namely Q-function, $Q(s_t, a_t; \omega) = \max_{\pi}(Q_\pi(s_t, a_t))$, which gives an expected return if an agent picks an action a_t while being in state s_t. Based on the trained Q-function, the agent can select an action a in a greedy way as $a = \arg\max_{a \in A}(Q(s_t, a_t; \omega))$ or in a ϵ-greedy way where the agent chooses a random action a under the probability of ϵ. The temporal difference (TD) algorithms are usually used to train DQN. The idea is to minimize loss function (1) which measures the distance between the predication $Q(s_t, a_t; \omega)$ and TD target y_t (2) at time step t.

$$loss = E[(Q(s_t, a_t; \omega) - y_t)^2] \tag{1}$$

Table 2: Action Notations.

Action	**Notation**	**Action**	**Notation**
ask for confirmation	AC	thanks	TH
inform	INF	greeting	GR
request	REQ	modify	MO
repeat	REP	summary	SU
ask to repeat	AR	close	CL
stop conversation	SC	confirm query	CQ
deny query	DQ	other actions	OA
deny summary	DS	ask to modify	AM
long time no reply	LT		

$$y_t = r_t + \gamma * \max_a(Q_\pi(s_{t+1}, a; \omega)) \tag{2}$$

while r_t is the accrual reward from the environment at time step t, ω represents the network parameter and $\max_a(Q_\pi(s_{t+1}, a; \omega))$ is the predicted reward of action a under state s_{t+1}.

The experience replay strategy Adam et al. (2012) is utilized in this paper to improve the performance of DQN. The tuples (s_t, a_t, r_t, s_{t+1}) are saved in the experience pool. The benefit of using experience replay is that it can make DQN not sensitive to the correlation among the samples, and increase the data efficiency by reusing the samples Lu et al. (2021).

4. Methodology

In this section, we firstly introduce the user simulator used to interact with the RL agent. Secondly, the reward function to train the basic dialog policy is proposed. Then the adaptation process is detailed.

4.1. User Simulator

The user simulator mimics human actions to train a RL based dialog agent. We build the user simulator based on the methods in Li et al. (2016) and Li et al. (2017) while meeting the specific needs of our task. The user actions and slots are listed in Table 1. Based on our observation of the human-robot dialog history, we find that users may stop the conversation or not reply to the robot for a long time. Therefore, we define two user actions SC and LT. Because of hearing impairment or the environment noise, the user may not hear robot's utterance clearly which prompts them to ask for repeat. Therefore user action AR is defined. CQ and DQ are the response to agent action AC. Considering the randomness and unpredictability of human speech, we use OA to represent the unrecognized or unexpected user intents. The user simulator can respond DS when there are slot errors in the agent action SU. As a response, the agent can perform AM to request the correct information and the user response is MO when needed.

A goal for the user simulator is randomly generated which includes all the six values of the slots shown in the $Slot$ column in Table 1. The user goal is only visible to the user simulator. For each agent action, the user simulator responds correspondingly. Fig. 3 shows some examples of the user simulator's response to agent actions. For example, when the agent performs REQ, the user simulator chooses action INF with the probability of (1 - P_{s_l} - P_{a_o}). P_{s_l} is the probability of selecting the action SC or LT. Considering the NLU and speech recognition errors, some errors are added to the inform slots with a probability of P_{slot_error}. The user actions SC and LT are selected with a probability of P_{s_l} and AR and OA are be selected with a probability of P_{a_o}.

4.2. Reward Function

The reward is a feedback to agent actions, which informs the agent how valuable it is to select a particular action. By adjusting the reward function, the agent actions can be controlled. In task-oriented dialog systems, important information obtained from users should be confirmed due to the speech recognition errors, limited NLU ability and unclear human voice. Furthermore, when the users ask the agent to repeat the utterance because of their hearing impairment, stopping the conversation or not responding for a long time, the agent should respond as expected. We proposed a reward function in Equation (3) based on human expectation. R_1 is commonly used in the ticket or restaurant booking tasks Lu et al. (2021), which is a task-level based reward. By using R_1, the agent can find a shortest dialog flow to finish the task. When the agent obtains all slots correctly, it is regarded as a success. Otherwise, it is a failed task. The drawback is that this reward makes the agent ignore the expectation under different user actions. Therefore, we propose R_2 and R_3 as the supplemental reward. R_2 controls the agent action AC. In Equation (5), A_u is user action. A_a is agent action. R_3 provides feedback for other agent actions. The idea behind R_2 and R_3 is that if the agent action meets the expectation, the reward is +1; otherwise the reward is -1. When the agent action meets the expectation, the total reward R_b is 0 during the conversation, which

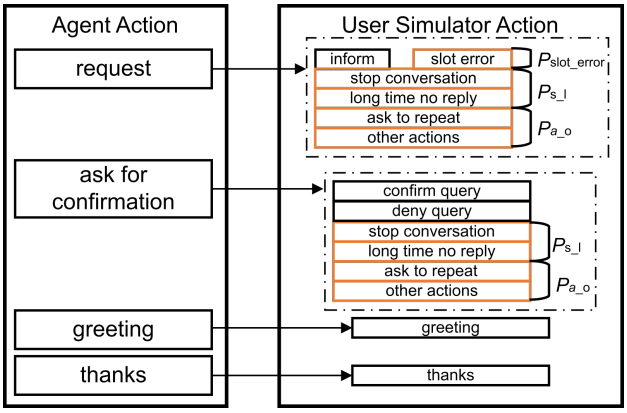


Fig. 3: User simulator response examples.

means there is no penalty on this action. When the users perform action SC or LT and the agent closes the conversation, the reward in R_3 is set to 11 based on our experiment results, which means we regard this action as a positive action.

$$R_b = R_1 + R_2 + R_3 \tag{3}$$

$$R_1 = \begin{cases} -1, & doing\ task \\ 20, & task\ success \\ -5, & task\ fail \end{cases} \tag{4}$$

$$R_2 = \begin{cases} 1, & (A_u\ in\ [INF])\&A_a\ in\ [AC]) \\ -1, & (A_u\ in\ [INF])\&\neg(A_a\ \ in\ [AC]) \\ 0, & other\ situations \end{cases} \tag{5}$$

$$R_3 = \begin{cases} 1 & (A_u\ in\ [AR, OA])\&A_a\ in\ [RE]) \\ & or\ (A_u\ in\ [DS])\&A_a\ in\ [AM]) \\ -1, & (A_u\ in\ [SC, LT])\&\neg(A_a\ \ in\ [CL]) \\ & or\ (A_u\ in\ [AR, OA])\&\neg(A_a\ \ in\ [RE]) \\ & or\ (A_u\ in\ [DS])\&\neg(A_a\ in\ [AM]) \\ 11, & (A_u\ in\ [SC, LT])\&A_a\ in\ [CL]) \\ 0, & other\ situations \end{cases} \tag{6}$$

4.3. Adaptation Method

The purpose of adaptation is to modify the dialog flow to make it fit a particular user's characteristics and preference while improving the accuracy of the NLU module. The adaptation is based on the conversation history between the user and the social robot and the user feedback collected from the touch screen. For the conversation part, we use the method in our previous work Su et al. (2021) to implement the ASR, TTS, NLU and NLG module. This section mainly focuses on the NLU updater module, the reward updater module and the user preference modeling module.

1) *NLU Updater* : The NLU module includes intent recognition and slot filling. For domain-specific tasks, due to the limited annotated corpus, slot filling is usually implemented using rule-based methods combined with a dictionary. The size of dictionary can affect the slot filling performance. Therefore, we propose an NLU Updater as shown in **Algorithm I**. When the users perform CQ to agent action AC (Line 6-12), the dictionary is expanded. If the user denies the query and confirms it after another round of conversation (Line 13-22), the correct pair is recorded to fix speech recognition errors.

2) *Reward Updater* : The reward function R_b enables the agent to obtain a fixed dialog policy, which cannot be adapted. The agent action AC is to ensure the correct slot values, which is important because the NLG model needs to use the collected slots to generate a good output utterance. For example, when the agent asks the *start time* after collecting *medicine* : *flu medicine*, the NLU module can generate the utterance *when will you start taking flu medicine*? If there is an error with the slot *medicine*, the NLU module will generate a wrong utterance which may reduce the users' satisfaction. However, if the agent keeps asking the user to confirm the information that is already clearly expressed in the past, it will also reduce users' satisfaction. Therefore, we believe that after interacting with the users for several rounds, it would be beneficial if the agent can adapt its actions to further improve the dialog policy. We proposed a *Reward Updater* to change the reward function for this

Algorithm I: NLU Updater

Input:
Dialog history $H = \{H_1, H_2, ..., H_i, ..., H_m\}$, m is the number of dialogs in the history. L_i is the number of turns of dialog H_i; NLU dictionary D and accept threshold α;

Output:
Updated NLU dictionary D;

for $H_i \in H$ **do**
  for $j \in L_i$ **do**
    $agent\_utterance = H_{ij}$, $user\_utterance = H_{i(j+1)}$
    $agent\_action = agent\_utterance[action]$
    $user\_uaction = user\_utterance[action]$
    if $agent\_action = AC$ & $user\_uaction = CQ$ **then**
      $slot = agent\_utterance[request\ slot]$
      $value = agent\_utterance[request\ slot][value]$
      if $repeat\_times$(slot,value) $\geq \alpha$ **then**
        $D[slot] \leftarrow D[slot] \cup value$
      end if
    end if
    $ac\_path = [DQ_o, INF_p, CQ_q]$
        or $[AM_o, INF_p, CQ_q]$
    if Judge(ac_path, $user\_utterance$) = True **then**
      $error\_value = H_{i(o-1)}[request\ slot][value]$
      $true\_value = H_{i(q-1)}[request\ slot][value]$
      $pair = [error\_value\ ,\ true\_value]$
      if $repeat\_times(pair) \geq \alpha$ **then**
        $D[correctionpair] \leftarrow D[correctionpair]$
        $\cup$ pair
      end if
    end if
  end for
end for
Return D

purpose, which is shown in **Algorithm II**. R_2 is updated if the conversation follows the *ac_path*. $\neg DQ_o$ means the agent action AC is not denied. $Judge(ac\_path, user\_utterance, S_i)$ decides if the user follows the action path *ac_path* when the agent requests S_i.

3) $User\ Preference\ Modeling$: Different people have different preferences on the agent actions, such as the order of the request entities or the generated response utterance. Therefore, it is not a good idea to provide the same dialog policy for different users. Furthermore, from our observation, when it comes to the request slots, the order of the request changes every time the RL model is trained. It can generate some unconventional request orders like requesting *end time* at the beginning and the *start time* in the end. Therefore, it would be better to enable users to explicitly show their preference and allow the agent to adjust its action according to users' feedback. In this paper, we focus on the user preference of request slot orders.

To collect user feedback, we built a User Interface (UI) as shown in Fig. 4. Users can push the buttons on the *Agent Action Suggestion* column to provide a feedback if the current agent action does not fit their preference. Besides, the *Obtained Information* column shows the collected slots and the top of the UI shows the current utterances of the user and the agent which make it easier for users to observe and track the interaction process.

The user feedback events can be divided into three classes: $no\ feedback$ (f_0), $negative\ feedback$ (f_1) and $positive\ feedback$ (f_2). We assign the three events to the preferences $no\ preference$ (p_0), $dislike$ (p_1) and $like$ (p_2). The item Situation ($Situ$) is used to represent the user preference features as can be seen in Table 3. The more feedback and features, the better the user preference can be learned. However, it is impractical to ask the users to provide a large amount of feedback. In order to reduce the requirement of the amount of feedback to model the user preference, only the key actions and slots are selected as features. The user preference feature consists of the last user action, the last user slot, the current agent action and the current agent slot, which is a 42-d vector. In real HRI, users may not provide feedback and sometimes they may provide different feedback for the same situation because of operation mistakes or change of their preference. Therefore, we use the overwrite strategy in which the old feedback will be replaced by the new feedback when it comes to the same situation. When users select the button which shows the same action as the current agent action (f_2), this situation is marked as p_2. When it is different (f_1), there are two pieces of feedback. The current situation is regarded as p_1 and another situation is generated based on the feedback and marked as p_2 as shown in Table 3. When the event is f_0, the situation is regarded as p_0.

Algorithm II: Reward Updater

Input:

Dialog history $H = \{H_1, H_2, ..., H_i, ..., H_m\}$, m is the number of dialogs in the history. L_i is the number of turns of dialog H_i; Accept threshold β, update threshold N, step size K; Slots $Slot = \{S_1, S_2, S_3, S_4, S_5, S_6\}$, S_1 corresponds to slot *patient* in the Slot column in Table 1;

Output:

Updated reward function R_{new};

Initialize $confirm\_vector$=[0, 0, 0, 0, 0, 0]
$ac\_path = [\neg DQ_o, INF_p, CQ_q]$
if $m \geq N$ **then**
  for $H_i \in H_{sub} = \{H_{m-B}, ..., H_m\}$ **do**
    for $S_i \in Slot$ **do**
      if Judge(ac_path, $user\_utterance$, S_i) = True
      or Judge($\neg AC$, $user\_utterance$, S_i) = True
      then
        $confirm\_vector_i = confirm\_vector_i + 1$
      end if
      if Judge(AM, $user\_utterance$, S_i) = True
      then
        $confirm\_vector_i = confirm\_vector_i$ - 1
      end if
    end for
  $confirm\_vector = confirm\_vector / N$
  for $C_i \in confirm\_vector$ **do**
    if $C_i \geq \beta$ **then**

$$R_{2i} = \begin{cases} 1, & (A_u \ in \ [INF]) \\ & \&(A_u[slot] = S_i \\ & \&A_a\neg \ in \ [AC]) \\ -1, & (A_u \ in \ [INF]) \\ & \&(A_u[slot] = S_i \\ & \&A_a \ in \ [AC]) \\ 0, & other \ situations \end{cases}$$

    end if
  end for
  $N = N + K$
$R_{new} = R_1 + R_2 + R_3$
Return R_{new}

The user preference feature is represented by a 42-d vector and classified into 3 classes. Therefore, we model the user preference as a classification task. There are many models like the RNNs or Convolutional Neural Networks (CNNs) Xu & Sarikaya (2013) that can be used for this task. For generalization and simplicity purpose, the Multilayer Perceptron (MLP) algorithm Taud & Mas (2018) is utilized to model the user preference. Based on the user preference model, a user preference reward function R_p is proposed as shown in Equation (7).

$$R_p = \begin{cases} 0, & p_0 == MLP(Situ), \\ -1, & p_1 == MLP(Situ), \\ 1, & p_2 == MLP(Situ). \end{cases} \tag{7}$$

5. Experimental Evaluation

We implemented the DSAM in our ASCCBot social robot. This section presents the experiments and evaluations of this system.

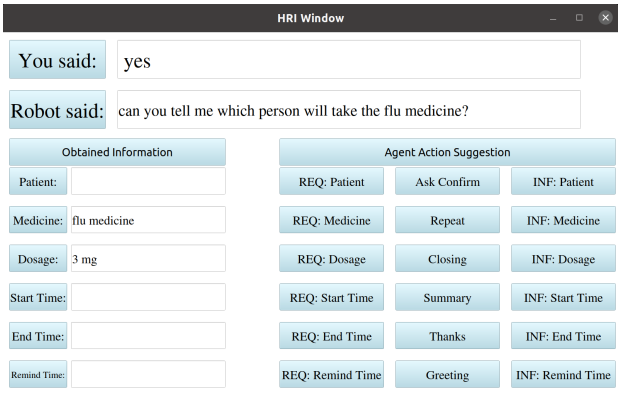


Fig. 4: The UI for user feedback.

Table 3: Preference features.

Original Preference Feature	**Generated Preference Feature**
last user action	last user action
last user slot	last user slot
current agent action	user feedback action
current agent slot	user feedback slot
p_0, p_1, p_2	p_2

5.1. Agent Learning Process Evaluation

In order to evaluate the performance of the dialog agent equipped with the basic reward function R_b, we set different user simulator parameters to test the dialog agent. R_1 is a task-level reward function which is commonly used in the RL-based dialog systems like ticket booking and restaurant booking and is regarded as a baseline reward. We also tested the performance of the dialog agent equipped with R_1 to compare with that of R_b.

1) *Experimental Setup:* We used the user simulator to train the basic dialog policy. The user simulator randomly selects a goal from the user goal set. We generated 500 user goals based on the collected dictionary which includes the medicine, patient name, dosage and time information. Each user goal has one reminder with 6 slots. To simulate the user actions like *stop conversation*, *long time no reply* or *ask to repeat* that appears occasionally, we set P_{s_l} to 0.05 and P_{a_o} to 0.1. The slot error rate P_{slot_error} is set to 0.1, 0.3 and 0.7 to test the system robustness, simulating the expert users, medium users and poor performance users, respectively. The maximum dialog turns are set to 50. The RL policy agent is a single layer neural network with a hidden layer size of 80. During the training, we used ϵ-greedy strategy to choose agent action, where ϵ is set to 0.03. Before training, the user simulator talks with a rule-based dialog agent to generate some initial experience with 100 conversation episodes. This experiment includes 200 interaction epochs. In each epoch, there are 200 conversation episodes. The dialog agent is updated at the end of each epoch.

2) *Results and Analysis:* The experiments were repeated for 5 times for each parameter setting. Fig. 5 (a) and (b) show the success rate and the average turns in each epoch with different slot error rates when using the proposed basic reward function R_b. Fig. 5 (c) and (d) show the success rate and the average turns in each epoch with different slot error rates when using the baseline R_1. The boundaries of the shaded region show the maximum and minimum value of the 5 runs. The middle line shows the average of the 5 runs. The results indicate the following:

- Equipped with the proposed basic reward function R_b, as can be seen from Fig. 5 (a), when the slot error is 0.1 and 0.3, the success rates are similar with both above 0.9. Error rate 0.1 needs less than 50 epochs while error rate 0.3 needs almost 60 epochs to converge, which means the larger the error rate the more time for the agent to find a good action. Fig. 5 (c) shows the success rate using the baseline R_1. We can observe that when the slot error is 0.1, it has a similar result as R_b. When the slot error is 0.3, using R_1 it takes almost 175 epochs to reach the same success rate as using R_b. The shadow area of the success rate using R_1 is larger than that using R_b, which means the dialog policy generated by R_b is more robust than R_1. When the slot error is 0.7, the success rate using R_1 reaches 0.7 and that of using R_b reaches 0.8, which is higher than R_1. It indicates that even when the user expresses the incorrect key information with a probability of 70%, the dialog agent trained with R_b is still able to handle such an extremely poor situation. From Fig. 5(b), we can observe that the average turns are 20, 24 and 28 for slot error 0.1, 0.3 and 0.7, respectively. The results are similar in Fig. 5(d) for R_1. It means the dialog agent generates different dialog strategies for different situations.

By observing the dialogues between the user simulator and agent using R_b, we noticed that in the slot collecting stage, the dialog flow follows the sequence: $REQ(agent) \rightarrow INF(user) \rightarrow AC(agent) \rightarrow CQ/DQ(user)$ when there is no other random user action. When using R_1, the

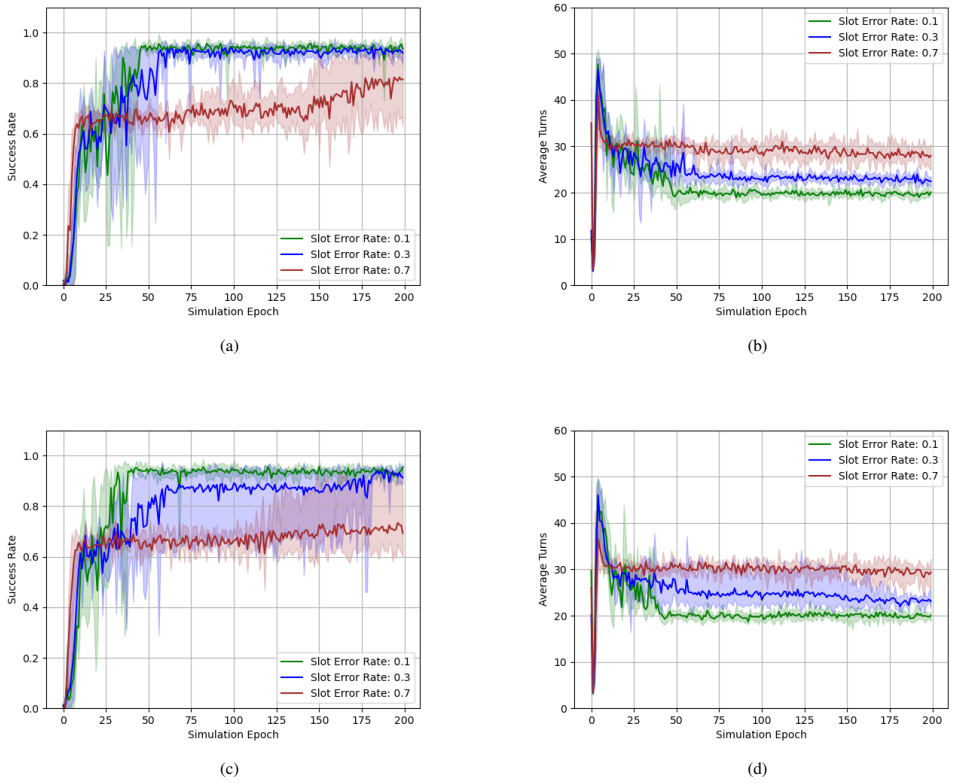


Fig. 5: Success rate and average turns of different rewards: (a) – Success rate of R_b; (b) - Average turns of R_b; (c) – Success rate of R_1; (d) – Average turns of R_1.

dialog flow follows the sequence: $REQ(agent) \rightarrow INF(user) \rightarrow REQ(agent) \rightarrow INF(user)$. For example, the user informs his name *John* but the speech recognition result is *Joan*. By performing the *AC* action, the agent can correct this slot immediately which the NLG module can use to generate the correct utterance for the next turn of conversation like "*Hi Jonn, can you tell me the medicine*?". However, with R_1's sequence, the agent can not correct this slot immediately and generates the wrong utterance like "*Hi Joan, can you tell me the medicine*?", which reduces user's satisfaction and trust on the service provided by the robot.

For comparison purpose, we constructed a rule-based agent to talk with the user simulator. However, the rule-based agent requires expert knowledge and 45 hand-crafted rules to achieve the similar results and dialog flow with our proposed method using the basic reward function R_b. With R_b and the exploration and exploitation property of the RL algorithm, the human effort of constructing rules can be significantly reduced.

5.2. Adaptation Evaluation on Simulated Users

1) *Experimental Setup* : We utilized the dialog agent trained with the base reward function R_b as the basic model to be adapted, where P_{slot_error} of user simulator US_1 is set to 0.2. Another user simulator US_2 is employed to talk with the agent, where P_{slot_error} is also set to 0.2. US_2 uses the NLG module to generate text response instead of directly sending the slots and values to the agent. The NLU module is then utilized to extract entities from the user utterance. The slots *patient*, *medicine* and *dosage* mainly use the dictionaries to extract entities. The slot filling success rate is largely based on the dictionary. Therefore, we focus on the expansion of the three slots' dictionaries and the corresponding actions. Their dictionaries are set to be empty before the adaptation. US_2 talks with the agent for 50 episodes. The *Reward Updater* is performed after the 50 episodes and the *NLU Updater* is performed after each episode. After the 50 episodes, the agent interacts with US_1 for 50 epochs with 50 episodes in each epoch, where the updated reward function R_{new} is utilized. The updated agent is used to interact with US_2 for another round.

2) *Results and Analysis* : We repeated this experiment for 5 times and got the following results. Fig. 6(a) shows the number of turns in different rounds. Fig. 6(b) shows the number of times that the system's NLU module failed to extract the three slots in each round. Fig. 6(c) shows the average success rate in each round. The results indicate the following:

- From Fig. 6(a), we can observe that Round 1 has the most turns because the first round is based on the basic reward function and without any adaptation. After the first round, the agent is adapted. Therefore, from Round 2 to Round 5, we can observe that the number of turns in each episode decreases gradually. At Round 5, the numbers of turns are mostly below 20. Furthermore, from Fig. 6(c), we can observe that the success rates in Round 4 and Round 5 reach 1.0, which indicates that even with the reduction of the interaction turns, the agent can still maintain a good performance.

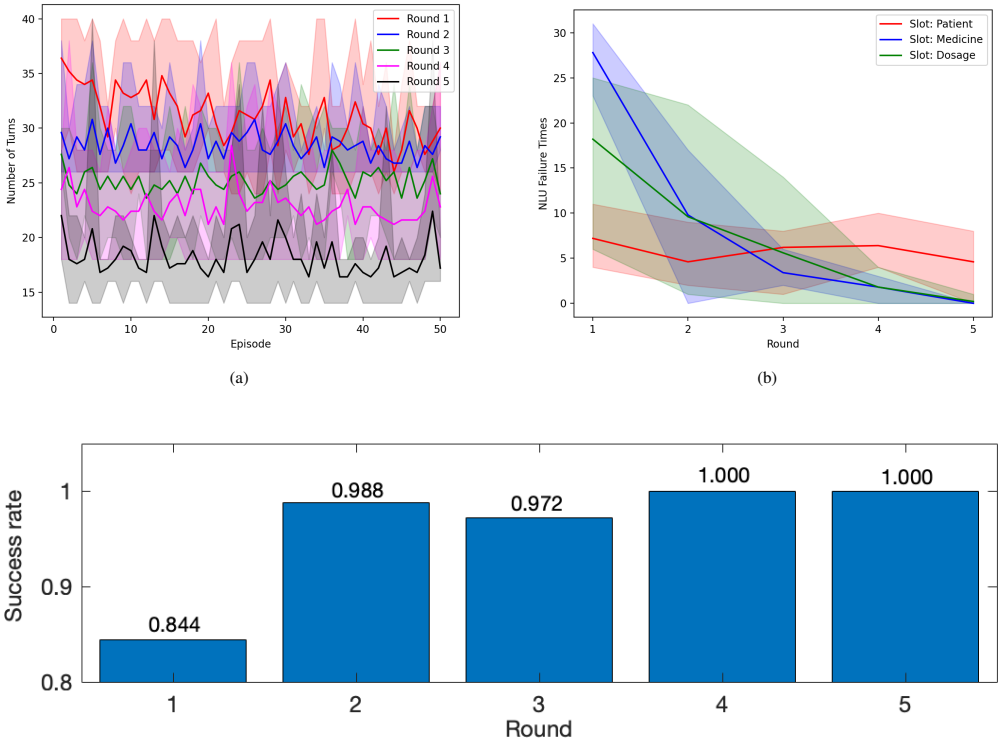


Fig. 6: Adaptation performance in different rounds evaluated on simulated users: (a) – Number of turns; (b) - NLU failure times; (c) - Average success rate.

- Before the adaptation, we set the dictionaries of the slots *patient*, *medicine* and *dosage* to be empty. Therefore, as can be seen in Fig. 6(b), the NLU module did not work well for slot *medicine* and *dosage* at the first round. The person name, which is a well-studied slot in the named entity recognition domain, is extracted using dictionary and NLTK's Loper & Bird (2002) part-of-speech tool. Therefore, its failure time is less than *medicine* and *dosage* at the beginning. With the adaptation, the failure time decreases quickly. The improvement on the NLU module in turn contributes to the reduction of interaction turns and enhancement of the success rate as can be seen in Fig. 6(a) and Fig. 6(c).

5.3. Adaptation Evaluation on Real Users

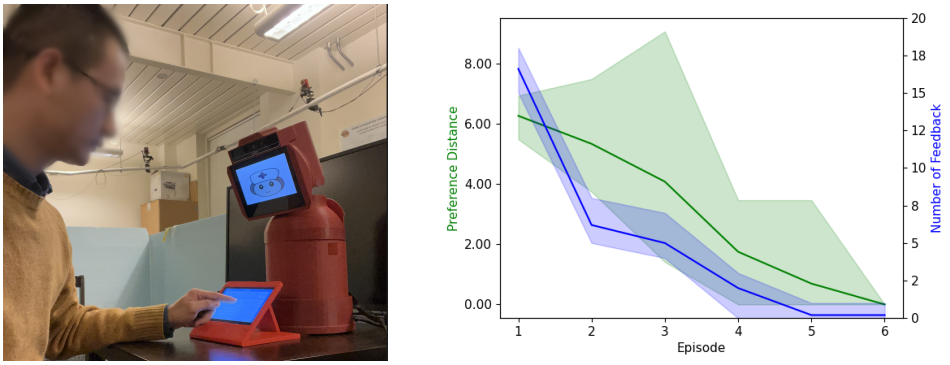
1) *Experimental Setup* : We recruited five users (5 males, between the ages of 25 and 35.) to test the adaptation process. They were asked to create 10 reminders as shown in Table 4. They also provided feedback through the touch screen by pushing the buttons if the current agent action do not fit their preference as shown in Fig. 7(a). The user preference model is updated after creating each reminder. For the NLU Updater, the acceptance threshold α is set to 1. For the Reward Updater, the acceptance threshold β is set to 0.5, update threshold N is 2 and the step size K is 2.

We defined a metric named Preference Distance (PD) to measure the difference between the preferred slot order p and current slot order c. PD is defined in Equation (8), where n is the number of slots. s_i is the ith slot. $index_p(s_i)$ is the index of the s_i in the slot order p. $index_c(s_i)$ is the index of the s_i in the slot order c. PD is 0 if the two orders are the same.

$$PD = \sqrt[2]{\sum_{i=1}^{n}[index_p(s_i) - index_c(s_i)]^2} \tag{8}$$

2) *Results and Analysis* : The users interacted with our social robot through verbal conversation to test the system and provided feedback through a touch screen. They tested the system based on two settings.

The first setting is for purely modelling the user preference. The users are asked to write down their preferred slot order before the interaction and create reminders selected from Table 4. It takes each user around 18 minutes to finish the test. Fig. 7(b) shows the PD metric and the number of feedback provided in the user preference adaptation process and Table 5 shows the corresponding statistical results. The number of feedback includes f_0, f_1 and f_2 mentioned in Part C, Section IV. The results indicate the following:



(a) (b)

Fig. 7: Adaptation on real users: (a) – A user is providing feedback to the dialog system; (b) - User preference adaptation in different rounds.

Table 4: Reminder information.

Reminder index	**Patient**	**Medicine**	**Dosage**	**Start date**	**End date**	**Reminder time**
1	Steven	Flu medicine	3 pills	Today	Next Monday	8 am and 3 pm
2	Jack	Penicillin	1.5 milligrams	Tomorrow	December 30th	1 pm
3	Steven	Flu medicine	3 pills	Today	Next Monday	8 am and 3 pm
4	Jack	Penicillin	1.5 milligrams	Tomorrow	December 30th	1 pm
5	Jobs	Ibuprofen	2.25 milliliters	December 23rd	Next Friday	At noon
6	Sherry	Aspirin	3.5 mg	Right now	tomorrow	8:30 am
7	Jack	Amoxicillin	4.15 ml	Today	Next Monday	3 pm
8	Steven	Cold medicine	3 spoons	March 1st	The day after tomorrow	9 am and 6 pm
9	Sherry	Penicillin	1 mg	Today	Next Monday	8 am and 3 pm
10	Steven	Cold medicine	3 pills	Today	Next Tuesday	In the morning

- From Fig. 7(b), we can observe that the amount of feedback and the preference distance decrease with the increase of the number of interactions. It indicates that the agent can adapt its actions to the user preference with the obtained feedback. From Table 5 we can observe that the amount of feedback at Episode 5 and 6 is nearly 0, which means for 4 out of 5 users, they do not need to provide any feedback during the last two episodes because the user preference model has already modeled their preference. At Episode 6, the mean of the preference distance is 0, which means that after the adaptation process, the agent can improve its actions to fit all users' preference.

Table 5: Statistical results of the 6-episode adaptation process.

Metrics	Preference Distance		No. of Feedback	
	Mean	Std. Dev.	Mean	Std. Dev.
Episode 1	6.26	0.74	16.60	1.34
Episode 2	5.33	1.41	6.20	1.30
Episode 3	4.08	2.96	5.00	1.22
Episode 4	1.75	1.64	2.00	1.22
Episode 5	0.69	1.55	0.20	0.45
Episode 6	0.00	0.00	0.20	0.45

The second setting is to combine the NLU Updater and Reward Updater with the User Preference Modeling to evaluate the adaptation performance. The NLU Updater updates the NLU module after creating each reminder (each episode). The User Preference Modeling is updated as well. The Reward Updater generates the renewed reward function R_{new} after creating every two reminders since each round has two episodes. The users were asked to create the 10 reminders shown in Table 4. It took each user around 30 minutes to finish the test.

Fig. 8 shows the adaptation process of five users. Fig. 8(a) to Fig. 8(e) show the adaptation process of action *AC*. Fig. 8(f) shows the preference distance. If there is a bar of *EXP Patient*, it means the agent is expected to perform *AC* on the slot *patient*. If there is a bar of

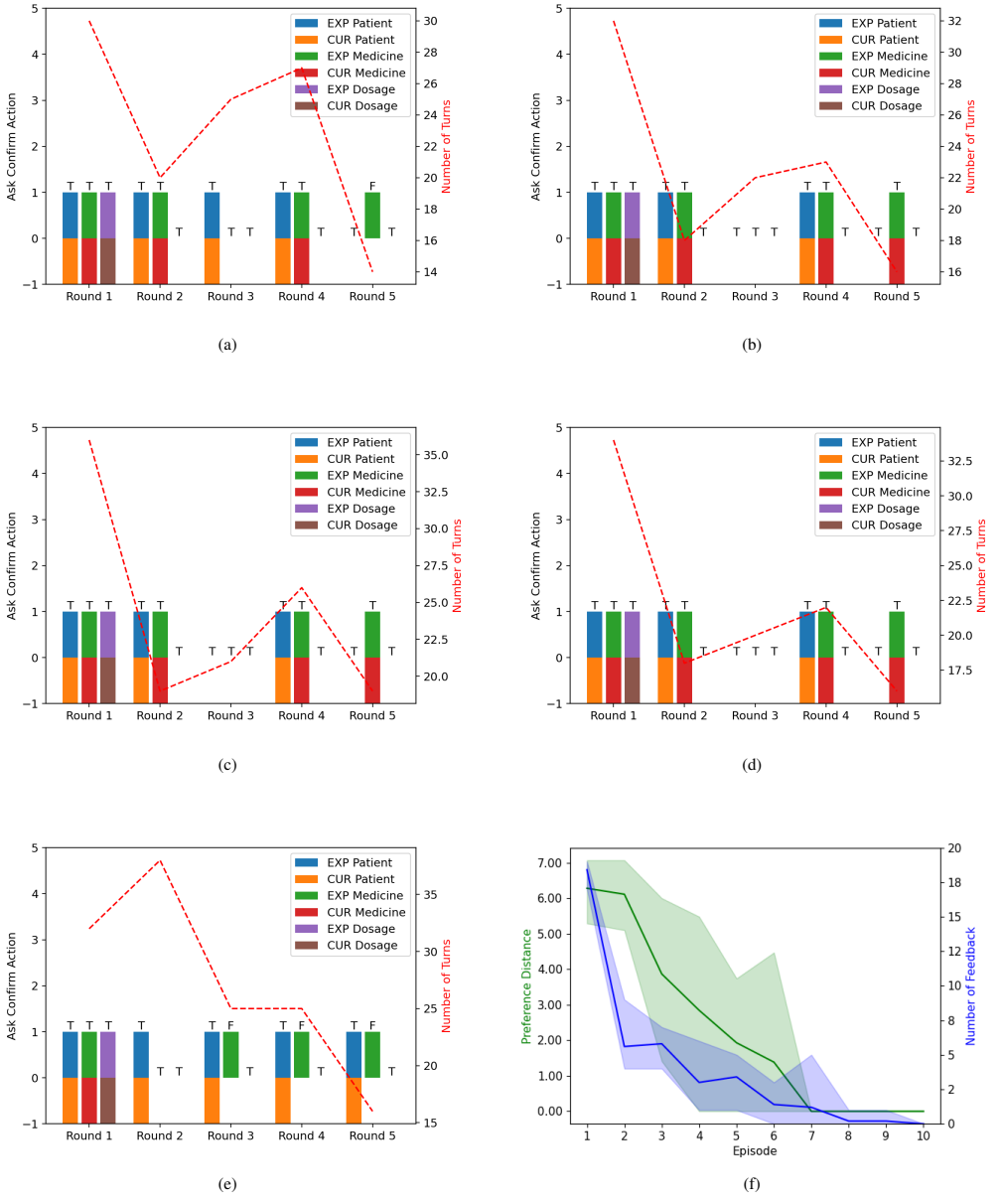


Fig. 8: Adaptation process of 5 rounds: (a) – User 1: Ask confirm action; (b) - User 2: Ask confirm action; (c) – User 3: Ask confirm action; (d) – User 4: Ask confirm action; (e) – User 5: Ask confirm action; (f) – Slot orders.

CUR Patient, it means the agent performed *AC* on the slot *patient* currently. If the expectation is performed in the current action (Co-occurrence or co-disappearance of the bar), it is marked as T. We can observe the following from the results:

- In the first two rounds (Reminder 1, 2 and 3, 4), the users need to create the same reminders. Therefore, we can observe that at Round 3, the results of user 2, user 3 and user 4 are as expected. For user 1 and user 5, because of the user accent and speech recognition error, the robot did not obtain the correct information even for the same reminder and the users had to ask the robot to modify the information, *AC* was performed at Round 3 for some slots. At Round 3 (Reminder 5 and 6), there are two new reminders. Therefore, at Round 4, the two bars appear as can be seen from user 1 to 5 and the number of turns also increases. At Round 4 (Reminder 7 and 8), the slot *patient* is not new, so action *AC* was not performed for user 1 to 4. The action *AC* success rates for user 1 to 5 are 93.3%, 100.0%, 100.0%, 100.0% and 80.0%, respectively. The average of the adaptation success rate is 94.7% and the number of turns drops below 20 at the end.

- Similar to Fig. 7(b), from Fig. 8(f) we can observe that the amount of feedback and preference distance decrease with the increase of the number of interactions. The action *AC* adaptation is included during the slot order adaptation process, therefore the zero-preference distance appears one episode later than what is shown in Fig. 7(b). Even in the complex real world situations, because of the proposed basic reward function R_b, the robot could still finish the tasks for all users to create all reminders successfully based on our observation of the dialog logs.

5.4. Summary

We conducted 3 experiments to test the proposed system. In the first experiment, we evaluated the performance of the dialog agent with the basic reward function R_b. The results show that the obtained dialog agent is robust to the slot errors and the dialog flow works as expected. The second experiment evaluates the performance of the $NLU\ Updater$ and the $Reward\ Updater$ using a user simulator US_2, the results indicate that while obtaining a good performance, the number of interaction turns and NLU failure times decrease along with the adaptation process. In the last experiment, we recruited human subjects to test our system with three adaptation processes. The results show that the average adaptation success rate is 94.7% and the preference distance reaches zero after 6 rounds of conversation while creating reminders successfully. Both the simulated and human subject tests show good performance.

6. Conclusion and Future Work

In this paper, we proposed an adaptive robotic dialog system for medication reminder using reinforcement learning. A user simulator is developed based on previous dialog history considering real interaction situations. In this system, the dialog policy and the NLU module can be improved based on human-robot conversations and user feedback through a touch screen. We utilized both the user simulator and human subjects to interact with the dialog adaptation system and the results show that our adaptive robotic dialog system achieves a good performance. Although the proposed adaptation method is utilized in our medication reminder application, this method can be used in many other task-oriented dialog systems and real-world human-robot interactions. In the future, we will improve the efficiency of the reinforcement learning algorithm to reduce the training time. We will also explore multi-modal feedback data to further reduce human's involvement in the adaptation process.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This project is supported by the National Science Foundation (NSF) Grants CISE/IIS 1910993, EHR/DUE 1928711 and CPS 2212582.

References

Abro, W. A., Qi, G., Gao, H., Khan, M. A., & Ali, Z. (2019). Multi-turn intent determination for goal-oriented dialogue systems. In *2019 International Joint Conference on Neural Networks (IJCNN)* (pp. 1–8). doi:10.1109/IJCNN.2019.8852246.

Adam, S., Busoniu, L., & Babuska, R. (2012). Experience replay for real-time reinforcement learning control. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *42*, 201–212.

Akiyoshi, T., Nakanishi, J., Ishiguro, H., Sumioka, H., & Shiomi, M. (2021). A robot that encourages self-disclosure to reduce anger mood. *IEEE Robotics and Automation Letters*, *6*, 7925–7932.

Bitencourt, F. V., Corrêa, H. W., & Toassi, R. F. C. (2019). Tooth loss experiences in adult and elderly users of primary health care. *Ciencia and saude coletiva*, *24*, 169—180.

Cassell, J., Pelachaud, C., Badler, N., Steedman, M., Achorn, B., Becket, T., Douville, B., Prevost, S., & Stone, M. (1994). Animated conversation: rule-based generation of facial expression, gesture spoken intonation for multiple conversational agents. In *SIGGRAPH '94 Proceedings of the 21st annual conference on Computer graphics and interactive techniques* (pp. 413–420). ACM.

Ferreira, E., & Lefèvre, F. (2015). Reinforcement-learning based dialogue system for human-robot interactions with socially-inspired rewards. *Comput. Speech Lang.*, *34*, 256–274.

Koo, S., Yu, H., & Lee, G. G. (2019). Adversarial approach to domain adaptation for reinforcement learning on dialog systems. *Pattern Recognition Letters*, *128*, 467–473.

Levin, E., Pieraccini, R., & Eckert, W. (1997). Learning dialogue strategies within the markov decision process framework. In *1997 IEEE Workshop on Automatic Speech Recognition and Understanding Proceedings* (pp. 72–79). doi:10.1109/ASRU.1997.658989.

Li, X., Chen, Y., Li, L., & Gao, J. (2017). End-to-end task-completion neural dialogue systems. *CoRR*, *abs/1703.01008*. URL: http://arxiv.org/abs/1703.01008. arXiv:1703.01008.

Li, X., Lipton, Z. C., Dhingra, B., Li, L., Gao, J., & Chen, Y. (2016). A user simulator for task-completion dialogues. *CoRR*, *abs/1612.05688*. arXiv:1612.05688.

Liang, F., Hernandez, R., Lu, J., Ong, B., Moore, M. J., Sheng, W., & Zhang, S. (2021). Collaborative fall detection using a wearable device and a companion robot. In *2021 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 3684–3690). doi:10.1109/ICRA48506.2021.9561323.

Lin, F. R., Yaffe, K., Xia, J., Xue, Q.-L., Harris, T. B., Purchase-Helzner, E., Satterfield, S., Ayonayon, H. N., Ferrucci, L., Simonsick, E. M., & Health ABC Study Group, f. t. (2013). Hearing Loss and Cognitive Decline in Older Adults. *JAMA Internal Medicine*, *173*, 293–299.

Lio, T., Yoshikawa, Y., Chiba, M., Asami, T., Isoda, Y., & Ishiguro, H. (2020). Twin-robot dialogue system with robustness against speech recognition failure in human-robot dialogue with elderly people. *Applied Sciences*, *10*, 1522.

Loper, E., & Bird, S. (2002). Nltk:the natural language toolkit. *CoRR*, *cs.CL/0205028*. URL: https://arxiv.org/abs/cs/0205028.

Lu, K., Cao, Y., Chen, X., & Zhang, S. (2021). Efficient dialog policy learning with hindsight, user modeling, and adaptation. *IEEE Transactions on Cognitive and Developmental Systems*, (pp. 1–1).

Manh Do, H., Sheng, W., Harrington, E. E., & Bishop, A. J. (2021). Clinical screening interview using a social robot for geriatric care. *IEEE Transactions on Automation Science and Engineering*, *18*, 1229–1242.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*, 529–533. doi:10.1038/nature14236.

Park, H. W., Grover, I., Spaulding, S., Gomez, L., & Breazeal, C. (2019). A model-free affective reinforcement learning approach to personalization of an autonomous social robot companion for early literacy education. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*, 687–694.

Perlmutter, M., Adams, C., Berry, J., Kaplan, M., Person, D., & Verdonik, F. (1987). Aging and memory. *Annual review of gerontology and geriatrics*, *7*, 57—92. URL: http://europepmc.org/abstract/MED/3120755.

Pham, M., Do, H. M., Su, Z., Bishop, A., & Sheng, W. (2021). Negative emotion management using a smart shirt and a robot assistant. *IEEE Robotics and Automation Letters*, *6*, 4040–4047. doi:10.1109/LRA.2021.3067867.

Qureshi, A. H., Nakamura, Y., Yoshikawa, Y., & Ishiguro, H. (2018). Intrinsically motivated reinforcement learning for human–robot interaction in the real-world. *Neural Networks*, *107*, 23–33. Special issue on deep reinforcement learning.

Ritschel, H., Seiderer, A., Janowski, K., Wagner, S., & André, E. (2019). Adaptive linguistic style for an assistive robotic health companion based on explicit human feedback. In *Proceedings of the 12th ACM International Conference on PErvasive Technologies Related to Assistive Environments* PETRA '19 (p. 247–255). New York, NY, USA: Association for Computing Machinery. doi:10.1145/3316782.3316791.

Rudary, M., Singh, S., & Pollack, M. E. (2004). Adaptive cognitive orthotics: Combining reinforcement learning and constraint-based temporal reasoning. In *Proceedings of the Twenty-First International Conference on Machine Learning* ICML '04 (p. 91). New York, NY, USA: Association for Computing Machinery. URL: https://doi.org/10.1145/1015330.1015411. doi:10.1145/1015330.1015411.

Saint-Aime, S., Le-Pevedic, B., Duhaut, D., & Shibata, T. (2007). Emotirob: Companion robot project. In *RO-MAN 2007 - The 16th IEEE International Symposium on Robot and Human Interactive Communication* (pp. 919–924).

Shah, P., Hakkani-Tur, D., & Heck, L. (2016). Interactive reinforcement learning for task-oriented dialogue management. In *NIPS 2016 Deep Learning for Action and Interaction Workshop*.

Su, Z., Li, Y., & Yang, G. (2020). Dietary composition perception algorithm using social robot audition for mandarin chinese. *IEEE Access*, *8*, 8768–8782. doi:10.1109/ACCESS.2019.2963560.

Su, Z., Liang, F., Do, H. M., Bishop, A., Carlson, B., & Sheng, W. (2021). Conversation-based medication management system for older adults using a companion robot and cloud. *IEEE Robotics and Automation Letters*, *6*, 2698–2705. doi:10.1109/LRA.2021.3061996.

Taud, H., & Mas, J. (2018). Multilayer perceptron (mlp). In *Geomatic Approaches for Modeling Land Change Scenarios* (pp. 451–455). Springer.

Vermorel, J., & Mohri, M. (2005). Multi-armed bandit algorithms and empirical evaluation. In J. Gama, R. Camacho, P. B. Brazdil, A. M. Jorge, & L. Torgo (Eds.), *Machine Learning: ECML 2005* (pp. 437–448). Berlin, Heidelberg: Springer Berlin Heidelberg.

Wang, H., Peng, B., & Wong, K.-F. (2020). Learning efficient dialogue policy from demonstrations through shaping. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 6355–6365). Online: Association for Computational Linguistics. URL: https://aclanthology.org/2020.acl-main.566. doi:10.18653/v1/2020.acl-main.566.

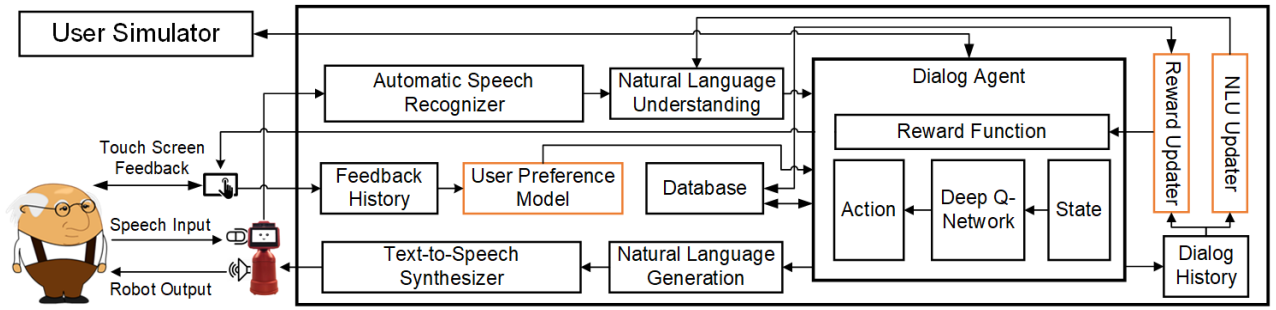
Xu, P., & Sarikaya, R. (2013). Convolutional neural network based triangular crf for joint intent detection and slot filling. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding* (pp. 78–83).

Zhao, J., & Li, X. (2018). The status quo of and development strategies for healthcare towns against the background of aging population. *Journal of Landscape Research*, *10*, 41–44.

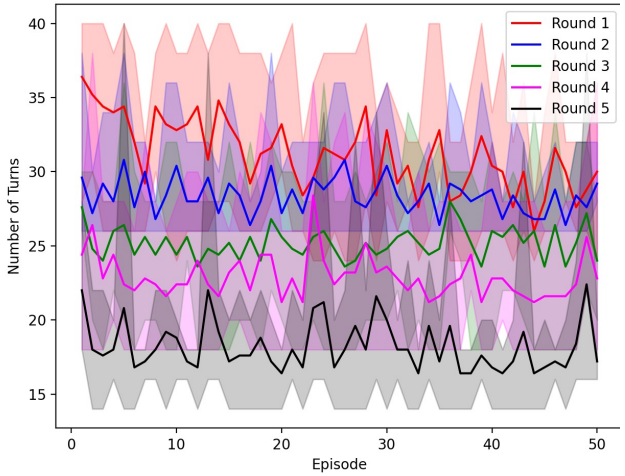
Adaptation of a robotic dialog system for medication reminder in elderly care

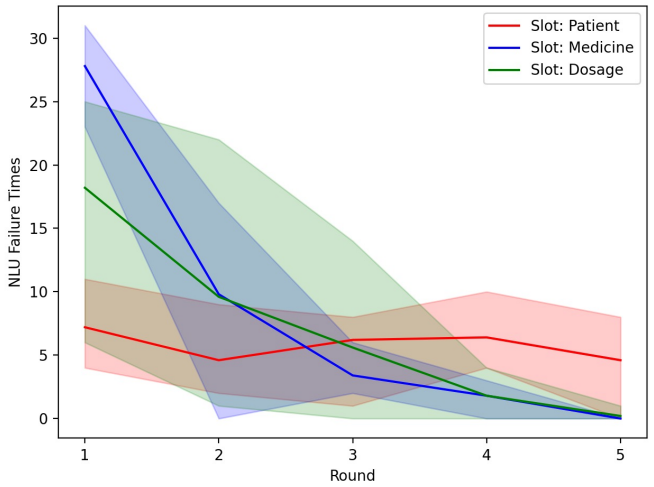
A dialog system adaptation method: adapt to older adults' characteristics and preference.

The system architecture

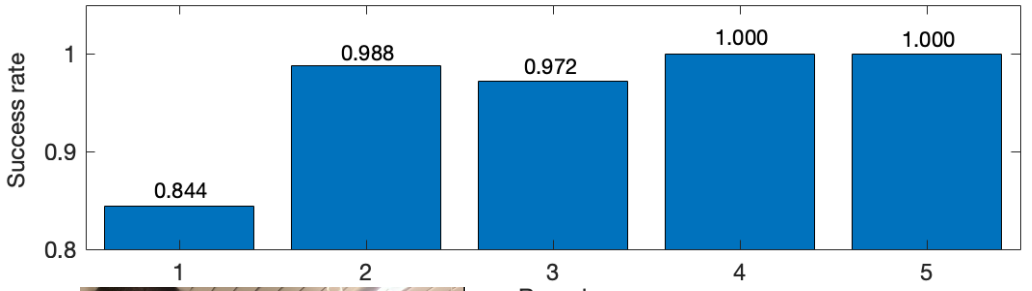


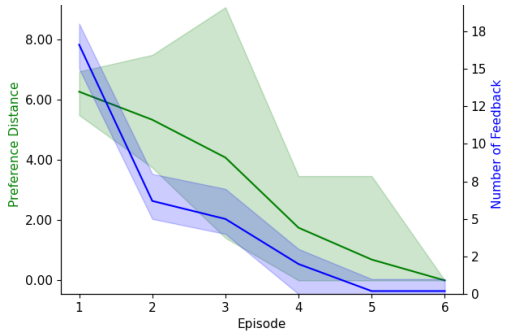
Results





Results





The average adaptation success rate is 94.7% and the preference distance reaches 0 after 6 rounds of adaptation while creating reminders successfully with a limited amount of user feedback.

Zhidong Su, Weihua Sheng, Guanci Yang, Alex Bishop, Barbara Carlson