ChatAdp: ChatGPT-powered Adaptation System for Human-Robot Interaction

Zhidong Su and Weihua Sheng*

***Abstract*— Different people have different preferences when it comes to human-robot interaction. Therefore, it is desirable for the robot to adapt its actions to fit users' preferences. Human feedback is essential to facilitating robot adaptation. However, when the task is complex or the robot action space is large, it requires a large amount of user feedback. ChatGPT is a powerful generative AI tool based on large language models (LLMs), which possesses a significant corpus of information obtained from human society, and exhibits robust proficiency in the comprehension and acquisition of natural language. Therefore, in this paper, we proposed a ChatGPT-powered adaptation system (ChatAdp) for human-robot interaction which requires less user feedback to achieve a good adaptation result. In the proposed ChatAdp, we use ChatGPT as a user simulator to provide feedback. We evaluated ChatAdp in a case study for context-aware conversation adaptation. The results are very promising. Our proposed method can achieve a mean success rate of 92% on the user's natural language-described preferences after receiving 33 rounds of feedback from a user on average, which is only 2% of the number of states covered by the user preferences and outperforms the two baseline methods.**

I. Introduction

Companion robots are equipped with various functions to assist users [1]–[5]. However, many existing robots provide fixed functions without much flexibility, which reduces users' satisfaction with the robot because different users have different preferences which may change over time. Therefore, it is desirable that the robot can gradually learn the preferences of users while adjusting their behaviors accordingly. The reinforcement learning algorithm is widely used in human-robot interaction since it accepts user feedback and optimizes its model gradually according to a reward function [6]. However, the drawback is that it requires a large amount of user feedback to adapt to a particular user, which is not practical for many applications, especially in robot-assisted elderly care.

Using a user simulator to interact with the robot can help reduce the burden on the users. ChatGPT [7] can play the role of a user simulator in conversational robot applications. ChatGPT is a powerful generative AI tool released by OpenAI which accepts prompt text and generates output using their large language model (LLM) [8]. ChatGPT's ability in understanding natural languages can be utilized to reduce the user effort in providing feedback. Namely, users can express their preferences using natural languages and let ChatGPT work as a customized user simulator to adapt the robot model. This offers a more user-friendly and practical way for robot adaptation, especially for older adults.

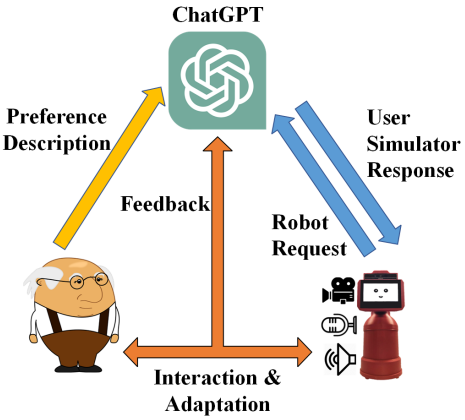

Fig. 1: ChatGPT-powered adaptation system.

Although ChatGPT is powerful, there are still some issues when used as a user simulator in robot adaptation. The output of ChatGPT is not very reliable. For example, the user simulator created according to the user preference description cannot always provide the correct response as expected. Therefore, it requires a mechanism to address this problem. Besides, the same concept may be understood differently by different users. For example, if the preference is "I do not want to be bothered at night", ChatGPT's understanding about "at night" may be between 10 PM and 12 AM. For a particular user, it may mean between 8 PM and 12 AM.

In this paper, we proposed a ChatGPT-powered adaptation system (ChatAdp) for human-robot interaction (HRI) to reduce the effort required from users for adaptation. ChatAdp embraced ChatGPT's LLM ability and tackled its existing drawbacks when directly applied for HRI. As shown in Fig. 1, a user can use simple natural language to describe his/her preferences. ChatGPT works as a user simulator according to the given preferences. The robot can talk to the user simulator to optimize its model using reinforcement learning algorithms before being deployed in the real world. The robot model will be further adapted according to the user feedback during HRI.

The main contributions of this paper are three folds. First, by utilizing ChatGPT's capabilities, the proposed ChatAdp allows a robot to adapt its actions based on users' preferences

This project is supported by the National Science Foundation (NSF) Grants CISE/IIS 1910993, EHR/DUE 1928711, CPS 2212582 and TI 2329852.

Zhidong Su and Weihua Sheng (Corresponding author) are with the School of Electrical and Computer Engineering, Oklahoma State University, Stillwater, OK, 74078, USA (e-mails: zhidong.su@okstate.edu, weihua.sheng@okstate.edu).

described in natural languages, therefore providing a more user-friendly and efficient adaptation method, which is the first of its kind in HRI research to the best of our knowledge. Second, ChatAdp is able to close the loop by utilizing user feedback to fine-tune ChatGPT, which enables the adaptation of the ChatGPT-powered user simulator. The adapted user simulator can be employed to update the robot model/policy for the next round of adaptation. The adapted model is able to run locally without using ChatGPT, which can save cost and reduce dependence on ChatGPT. Third, we conducted the human subject test and evaluated the performance of the ChatAdp while obtaining promising results.

The rest of this paper is organized as follows. Section II introduces the related work. Section III details the proposed method. Section IV introduces the case study and the experimental results. Section V concludes our work and discusses the future work.

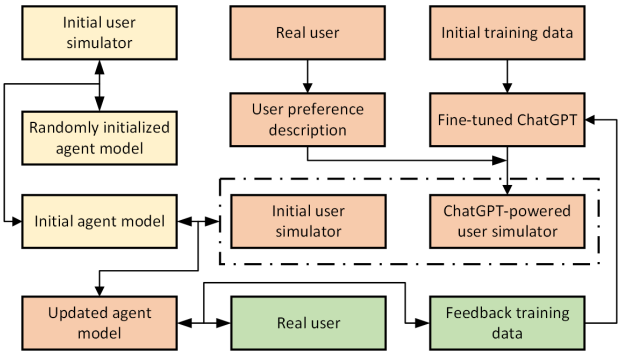

Fig. 2: The software architecture of ChatAdp. (The two Initial user simulator modules with different colors are the same modules. We separate them to make it easier for demonstration. Same for the Real user models. Better view in color.)

II. Related Work

Adaptation in human-robot interaction refers to the ability of a robot to adjust its behavior to better match the human's preferences, needs, and performance to improve human satisfaction, interaction engagement, and adherence, etc. Ritschel et al. [9] utilized the multi-armed bandit algorithm [10] to adapt a companion robot's linguistic style for gameplay and daily chatting based on explicit human feedback through buttons. After 200 rounds of interaction, the robot was able to find a specific linguistic style that fitted the participants' preferences. Gordon et al. [11] utilized the interaction between a social robot tutor and a child to learn a Q-matrix to maximize long-term learning gains for children's second language skills. According to the state, the robot can output utterances with different valences and engagement information. The robot was deployed in the participants' homes for two months to finish the adaptation, which is a very long adaptation process. Qureshi et al. [12] proposed an approach to impart human-like social skills to robots by leveraging reinforcement learning techniques. Specifically, the method involves enabling robots to learn through real-world interactions, where the reward signal is computed as the difference between the output of the event detector at different time steps. They totally collected 13,938 rounds of interaction during the 14 days of experimentation to enable their robot to learn the skills. The above-mentioned works can adapt the robot's actions to fit users. The explicit signal or the robot sensor data is used as feedback to optimize the robot policy. Without knowing how the users want the robot to perform actions, it requires many rounds of interaction to capture the preferences.

ChatGPT is based on a large language model which is trained on an extensive corpus, enabling it to excel in a wide range of applications and tasks. Vemprala et al. [13] proposed various applications using ChatGPT for robotics. They utilized ChatGPT to generate Python codes to control real-world drone flight, aerial robots in a simulator, and a robot arm. Yu et al. [14] utilized an LLM to convert natural language into reward functions. However, no user feedback is used to fine-tune the LLM, making the system unable to keep optimizing the generated reward functions. Wake et al. [15] proposed to use ChatGPT to convert natural language instructions into an action sequence that the robot can execute. They developed prompts that are capable of inferring appropriate actions for multi-stage language instructions in diverse environments. Whenever they need the robot to perform a task, they rely on ChatGPT to provide help. If ChatGPT's service is down, the proposed method cannot work. Therefore, it is desirable if we can take advantage of ChatGPT to train a local model and deploy the local model for HRI, which does not rely on ChatGPT.

In this paper, we propose to use ChatGPT as a user simulator to adapt to users' preferences, which can 1) reduce human effort because of the LLM ability and 2) reduce dependence on ChatGPT in future interactions because once the model is adapted the user simulator is not required for the robot to perform actions.

III. Methodology

In order to enable a companion robot to adapt to users' preferences more naturally and efficiently, we proposed ChatAdp, a ChatGPT-powered adaptation system based on reinforcement learning. Equipped with ChatAdp, users can just use natural languages to describe their preferences while providing very limited feedback. The robot automatically adapts its actions to fit the user preferences. In this section, we first give an overview of the proposed ChatAdp system. Second, we formulate the human-robot interaction using the reinforcement learning framework. We introduce the proposed ChatGPT-powered adaptation algorithm at the end.

A. System Overview

Fig. 2 presents the software architecture of the proposed ChatAdp. It has three parts. The first part is shown in the light yellow blocks. We call it $Initialization$. This part is responsible for training an initial agent model, which enables the robot to perform actions that fit average users' needs. The $Initial\ user\ simulator$ module works as an average user

that can provide feedback to the actions generated by the $Randomly\ initialized\ agent\ model$ module. Optimized by a reinforcement learning algorithm, the $Initial\ agent\ model$ is obtained, which can be deployed for HRI.

The second part is shown in the light orange blocks, which is called *Pre-adaptation*. The *Initial training data* is used to fine-tune ChatGPT to obtain a *Fine-tuned ChatGPT*, which enables ChatGPT to understand the task it needs to do. Before interacting with the robot, the *Real user* needs to provide *User preference description* through natural language to tell the robot about how he/she prefers the robot to perform tasks. With the *User preference description*, the *Fine-tuned ChatGPT* is utilized as a *ChatGPT-powered user simulator*. It works with the *Initial user simulator* to interact with the *Initial agent model*, which will be updated to the *Updated agent model* using the reinforcement learning method.

The third part is shown in the light green blocks. We call it *Post-adaptation*. The *Real user* interacts with the robot equipped with the *Updated agent model*. The feedback is utilized to construct personalized training data to fine-tune ChatGPT for the next round of adaptation.

B. Human-Robot Interaction Formulation

The robot is capable of performing actions based on the observations of its sensors and interaction history. It is desirable for a robot to be able to adapt its actions according to the preferences of users, which can be reflected through their feedback. The reinforcement learning algorithm uses the reward from the environment to optimize its policy, which is suitable for HRI. The policy determines the appropriate action to choose, and the reward can be derived from the user's positive, negative, or other feedback.

The human-robot interaction process is modeled as a Markov Decision Process (MDP) [16]. This process can be represented by a tuple comprising of five elements: the state set, denoted by S, which includes the sensory observations and interaction history of the robot and is used as input to the robot model or action policy; the action set, denoted by A, which represents the output of the robot model; the state transition probability matrix, denoted by T, which is commonly used in model-based reinforcement learning methods; the reward function, denoted by R, and the discount factor, denoted by $\gamma \in [0, 1]$. The cumulative reward with discount factor can be expressed as $R = \sum_{t=0}^{+\infty} \gamma^t r_t$, where t is the time step and $r_t = R(s_t, a_t)$. $\pi : S \rightarrow A$ represents the policy function, which maps a state to an action. The primary objective of the MDP is to derive an optimal policy that can maximize the cumulative reward.

In this study, the deep-Q-network (DQN) is employed to approximate the optimal action policy for a robot. Unlike model-based methods that necessitate the state transition probability matrix T, DQN is a model-free deep neural network [17] that eliminates the need to model the intricacies of the real world. The Q-function represents the optimal robot action policy, given by $Q(s_t, a_t; \omega) = \max_{\pi}(Q_\pi(s_t, a_t))$, which offers an expected return based on the optimal policy π when the robot takes action a_t in state s_t. The DQN algorithm is optimized using the temporal difference (TD) algorithm, and the loss function (Eq.(1)) is used to minimize the difference between the predicted return $Q(s_t, a_t; \omega)$ and the TD target y_t (Eq.(2)), which includes the actual reward at step t and the maximum expected return at step $t+1$. The network parameter ω is optimized using backpropagation. Ultimately, the optimal ω can be utilized by the robot to select an action a_t using Eq. (3).

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

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

$$a_t = \arg\max_{a_t \in A}(Q(s_t, a_t; \omega)) \tag{3}$$

C. ChatGPT-powered Adaptation Algorithm

Table I shows the ChatGPT-powered Adaptation Algorithm. There are three steps, which correspond to the three sections in Fig. 2. In Step 1, the initial policy implemented using DQN is obtained. The user simulator $user_s$ can generate a state using a function named $generate\_state$, which uses the current time t, conversation history h and user preference description $user_{desc}$ for state generation. The $generate\_response$ function provides a response based on the initial user preference, which accepts two parameters, the current state s_t and agent action a_t. In Step 2, the fine-tuned ChatGPT $ChatGPT_{ft}$ is obtained using the *fine-tune* function with $data_t$. The "Fine-tuneModel" function utilizes the fine-tuned ChatGPT $ChatGPT_{ft}$ to adapt the agent policy. In Step 3, the user feedback $feedback$ is used to fine-tune ChatGPT, which is further utilized to adapt the agent policy. The policy with the updated network parameter ω is equipped for the next round of HRI.

IV. Case Study

In this section, we present a case study of context-aware conversation adaptation, which utilizes ChatAdp for adaptation. We also evaluate ChatAdp's performance in this case study.

A. Introduction

In this case study, the robot uses its microphone and camera to observe the user's activity and proactively start a conversation if needed. Users can accept, reject or ignore the robot initiation, or initiate a conversation if they need any help from the robot based on their preferences. The goal is to maximize users' acceptance rate of the robot's proactive initiation. We introduce the state, agent actions, and reward function of this case study.

The actions performed by the agents/robots and their corresponding state components and dimensions are presented in Table II. To represent the state components, a one-hot vector is employed and concatenated to form the state. The dimension of each component is shown in the Dim column.

In relation to the sub-state S_1, there are a set of 33 home environment sound events. These sound events include: 1) coughing, 2) crying, 3) clearing of throat, 4) sneezing, 5)

TABLE I: ChatGPT-powered Adaptation Algorithm.

Input: Initial user simulator $user_s$, ChatGPT model $ChatGPT$, user preference description $user_{desc}$, training data $data_t$, train_days, epochs, adaptation threshold N.
Output: adapted agent model ω.

#Step 1: Initialization;
Randomly initialize ω;
For day **in** train_days:
  $h = \varnothing$; buffer $b = \varnothing$; $t = 0$; s_t=$user_s$(generate_state(t,h));
  For t **in** epochs:
    a_t=arg $\max_{a \in A}(Q_\pi(s_t, a_t; \omega))$; #Eq.(3)
    ur=$user_s$(generate_response(s_t, a_t));
    $h = h \cup (s_t, a_t)$; s_{t+1}=$user_s$(generate_state(t,h));
    $b = b \cup (s_t, a_t, ur, s_{t+1})$;
    $\omega \leftarrow \omega$ - lr* $\frac{\partial L(\omega)(b)}{\partial \omega}$; #Eq.(1);

#Step 2: Pre-adaptation;
Function Response($user_s$, $ChatGPT_{ft}$, s_t, a_t, $user_{desc}$):
  If fit(s_t, $user_{desc}$), **Then**
    ur = $ChatGPT_{ft}$(s_t, a_t,$user_{desc}$);
  Else ur=$user_s$(generate_response(s_t, a_t));
  Return ur;
Function Fine-tuneModel($ChatGPT_{ft}$):
  For day **in** train_days:
    $h = \varnothing$; buffer $b = \varnothing$; $t = 0$;
    s_t=$user_s$(generate_state(t,h,$user_{desc}$));
    For t **in** epochs:
      a_t=arg $\max_{a \in A}(Q_\pi(s_t, a_t; \omega))$; #Eq.(3)
      ur=Response($user_s$, $ChatGPT_{ft}$, s_t, a_t, $user_{desc}$);
      $h = h \cup (s_t, a_t)$; s_{t+1}=$user_s$(generate_state(t,h,$user_{desc}$));
      $b = b \cup (s_t, a_t, ur, s_{t+1})$;
      $\omega \leftarrow \omega$ - lr* $\frac{\partial L(\omega)(b)}{\partial \omega}$; #Eq.(1);
  Return ω;
$ChatGPT_{ft}$ = fine-tune($ChatGPT$, $data_t$);
ω = Fine-tuneModel($ChatGPT_{ft}$);

#Step 3: Post-adaptation;
$feedback = \varnothing$;
While True
  $feedback$ = $feedback \cup$ HRI(real user, $Q_\pi(\omega)$);
  If length($feedback$)%N = 0, **Then**
    $ChatGPT_{ft}$ = fine-tune($ChatGPT_{ft}$, $feedback$);
    ω = Fine-tuneModel($ChatGPT_{ft}$);

sniffing, 6) falling down, 7) burping, 8) yawning, 9) snoring, 10) drinking water, 11) drinking milk, 12) drinking soup, 13) eating an apple, 14) eating noodles, 15) eating chips, 16) washing hands, 17) washing clothes, 18) washing dishes, 19) using scissors, 20) using a blender, 21) using a stove, 22) using a hair dryer, 23) using a microwave, 24) using a fan, 25) watching television, 26) typing on a keyboard, 27) cutting food, 28) frying food, 29) pouring water into a glass, 30) shaving, 31) brushing teeth, 32) flushing a toilet, and 33) doing nothing.

For the sub-state S_2, we consider 6 categories, namely, 1) bedroom, 2) living room, 3) kitchen, 4) bathroom, 5) dining room, and 6) other. Utilizing the integrated microphone and camera, the robot obtains sound-based daily activities and image-based home scenes. We trained a sound-based daily activity recognition model and an image-based home scene recognition model based on a convolutional neural network, which is not introduced because it is not the focus of this paper.

The sub-state denoted by S_3 represents the temporal dimension at which an activity is observed. To enable efficient analysis, we partition the 24-hour day into twelve non-overlapping time intervals, each consisting of 2 hours. Thus, we can express S_3 as follows: $S_3 = [0, 2), [2, 4), [4, 6), [6, 8), [8, 10), ..., [20, 22), [22, 24)$. S_4 denotes the time elapsed between the current activity and its preceding activity, with the exception of the 'do nothing' activity. To define S_4, we consider a set of time intervals, denominated by $S_4 = \{0, 1, 2, 5, 10, 15, 20, 30, 60, 90, 120, 200\}$, with minute granularity.

TABLE II: State and agent actions.

State	Dim	Agent Actions
S_1: sound-based daily activity	33	do nothing
S_2: image-based home scene	6	show compassion
S_3: current time	12	chitchat
S_4: activity repeat time interval	12	remind to take medicine
S_5: current task	14	ask to offer help
S_6: agent action	14	call someone for help
S_7: user action	3	set a timer
Total dimension	94	pain check
		confirm sound
		record sound
		remind to turn off
		remind to be careful
		remind to put back
		remind to unplug

The sub-state S_5 indicates the task that the robot is performing, which is analogous to the agent's action. The agent's action is dependent on the robot's present observation. To facilitate the robot's execution of daily activities and exhibit care towards users, we developed 14 distinct agent actions, as presented in the $Agent\ Actions$ column of Table II. Additionally, the sub-state, $user\ action$, is defined as $S_7 = \{\text{"}accept\text{"}, \text{"}reject\text{"}, \text{"}others\text{"}\}$. Overall, the state space encompasses 94 dimensions, while the agent action space comprises 14 dimensions, resulting in a total of 16,765,056 states.

The reward function is a vital component in the dialog agent's behavior optimization process, as it serves as a metric to assess the agent's performance. In this study, if the user feedback intent is "accept", reward R = 1. If the user feedback intent is "reject", R = -1. Otherwise, R = 0.

B. Evaluation

1) Experimental Setup: Before the human subject test, we built a rule-based initial user simulator which is used to train an initial agent policy and for adaptation usage in Steps 2 and 3 (Table I), a rule-based target user simulator built according to the user preferences for evaluation purposes, and initial training data to fine-tune ChatGPT. The initial training data was generated by a Python script and manually screened by a human expert. There are 5 preference descriptions and a total of 4176 training samples. Table III shows 3 examples of the training data, which are in the format required by OpenAI to fine-tune the model named *ada*. For each sample, there are two elements, namely, "prompt" and "completion". The

TABLE III: Data samples of ChatGPT fine-tuning

1: {“prompt”:“preference: the agent needs to set a timer when I use a stove.\ncontext: {‘sound-based daily activity’: ‘use stove’, ‘image-based home scene’:‘bathroom’, ‘current time’: [2, 4], ‘activity repeat time interval’: 60, ‘current task’: ‘do_nothing’, ‘agent action’: ‘set a timer’} \nmy response:!\nSupported:”,“completion”:“ accept”}
2: {“prompt”:“preference: the agent needs to remind me to take medicine during my 3 mealtimes if I cough.\ncontext: {‘sound-based daily activity’:‘cough’, ‘image-based home scene’:‘living room’, ‘current time’: [20, 22], ‘activity repeat time interval’: 90, ‘current task’: ‘do_nothing’, ‘agent action’: ‘record sound’}\nmy response:!\nSupported:”,“completion”:“ reject”}
3: {“prompt”:“preference: please do not bother me when I am typing keyboard.\ncontext: {‘sound-based daily activity’: ‘type keyborad’, ‘image-based home scene’:‘study room’, ‘current time’: [10, 12], ‘activity repeat time interval’: 5, ‘current task’: ‘do_nothing’, ‘agent action’: ‘do_nothing’}\nmy response:!\nSupported:”,“completion”:“ accept”}

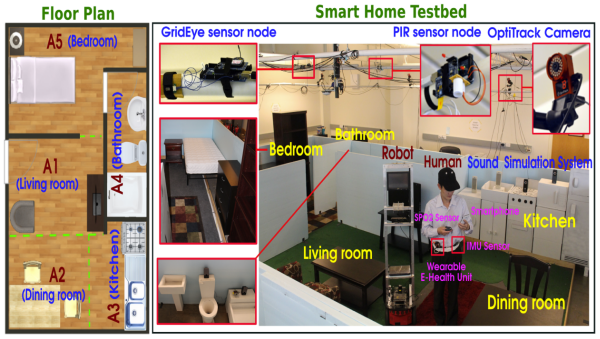

Fig. 3: The ASCC Smart Home testbed.

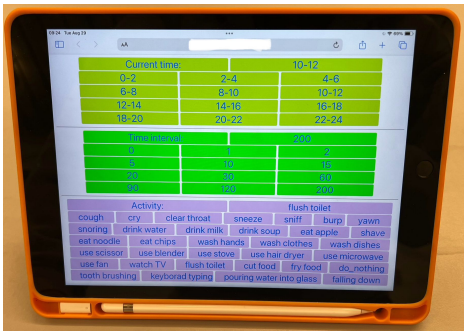

Fig. 4: The web user interface for context setting.

“prompt” element has a natural language description of a user preference and a context description in natural language instead of vectors. The “completion” element includes a response of a user who has a preference mentioned in the “prompt” element in such a context.

We recruited 5 male human subjects aged between 25 and 35 to test ChatAdp. The human subject test was approved by the Oklahoma State University IRB office under application No. IRB-22-252. We introduced the whole system to the participants and demonstrated the system. After that, we asked each user to describe 5 preferences that they like the robot to behave to assist their daily life, which are not seen in the initial training data.

The adaptation process was conducted in our ASCC Smart Home testbed [18] as shown in Fig. 3, where the human subjects interact with the robot. The testbed mimics an apartment, which includes a bedroom, a living room, a kitchen, a bathroom and a dining room. The human subjects choose a room and the robot stays with the user. They can also use different rooms for different conversations. In order to conduct the experiment in an efficient manner, as can be seen in Fig. 4, we developed a web page as a user interface for setting the context. Users can use a tablet’s browser to select the sub-states S_3 (current time), S_4 (activity repeat time interval), and S_1 (sound-based daily activity) that they want to perform. The sound of the activity is played by the speaker to mimic the real sound activity. The rest of the sub-states are obtained automatically by the robot.

We use the robot camera to obtain home scene images and the microphone to record the sounds generated from daily activities. We trained two convolutional neural network (CNN) based models for home scene recognition and sound activity recognition.

After receiving the user preferences, the ChatGPT fine-tuned by the initial training data is used to update the agent model, which is called “day 0” training in this paper. After training, the participants are asked to perform their daily life routines following their preferences and interact with the robot around 15 times per day for 2 days. The interaction data is used to fine-tune ChatGPT and the fine-tuned ChatGPT is used to update the agent model/policy. It takes each participant around 20 minutes to finish the test.

We also implemented two baseline methods for comparison purpose: 1) the original DQN reinforcement learning method (ORL) which does not use the ChatGPT-powered user simulator for adaptation, and 2) a supervised learning neural network-based (NN) behavior cloning method. This method belongs to imitation learning which directly imitates the human-robot interaction data to train a policy. We use the interaction data each day to adapt the agent model using the three methods. The ORL-based policy has the same dimensions of input and output as ChatAdp’s policy and uses all user feedback data. The NN-based policy also has the same dimensions of input and output as ChatAdp’s policy but only the “accept” user feedback data are used because the “reject” user feedback cannot be used as a label. The updated models were evaluated by simulation data generated based on the described preferences. The evaluation was repeated 4 times.

2) Results and Analysis: Table IV shows some preference description examples provided by the participants. Table V shows the number of states that are covered by 5 preference descriptions of each participant and the total amount of feedback in the two-day test. $P1$ means participant 1. $Total\ State\ NUM$ means the total number of states. Fig. 5

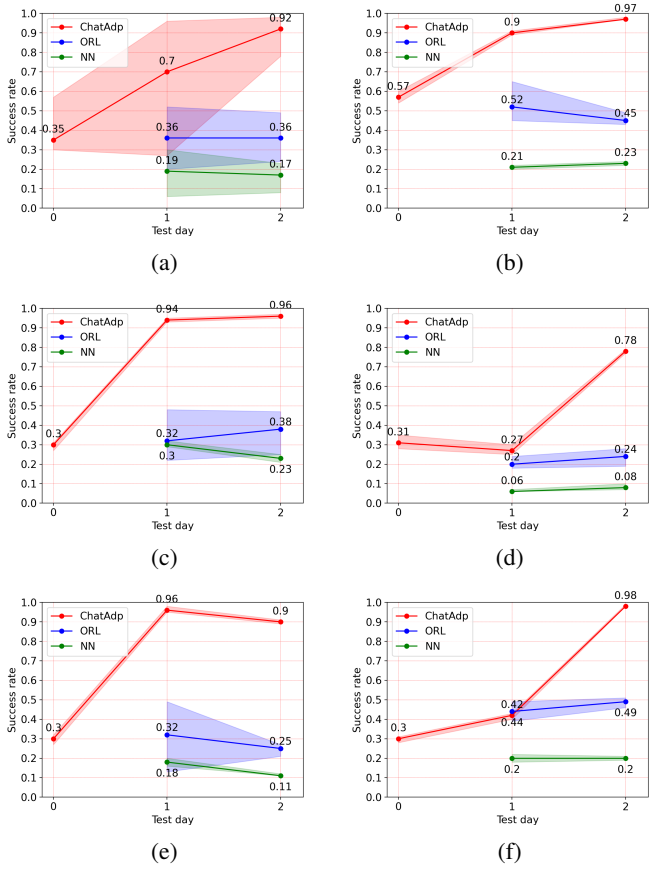

Fig. 5: Adaptation results: (a) – Average success rate of all participants; (b)-(f) Participant 1-5. (The ORL and NN methods show no results on day 0 as no feedback is provided.)

shows the adaptation results including the average results and individual results. The success rate is calculated by dividing the number of accepted agent actions by the total number of agent actions in a simulation day.

From Fig. 5(a), we can observe that the average adaptation success rate is 92% after 2-day adaptation with averagely 33 rounds of user feedback, which is only 2% of the average number of states covered by the user preferences (1666, Table V). We notice that a user preference has a piece of implicit location information "kitchen" expressed by "in the place where I cook meals". Utilizing ChatGPT's LLM ability of common sense, the proposed ChatAdp is able to adapt the agent model to fit this preference without any user feedback. ChatGPT sometimes has different understandings of the concepts like time interval/frequency or daytime for different people. Therefore, on day 0, the success rate is 35%. After utilizing user feedback, the adapted ChatGPT is able to provide user-preferred output and the success rate can reach 92% after 2 days.

Overall, the adaptation success rate of the proposed ChatAdp is 56% and 75% higher than the ORL- and NN-based methods, respectively. The ChatGPT-powered user simulator is able to provide more correct training data by mimicking user preferences, which achieves better performance. Based on the above analysis, we can draw a conclusion that ChatAdp has a good adaptation ability.

TABLE IV: Participants' preference description examples.

Preference Description
1: The agent needs to remind me to take medicine if I drink water at night.
2: The agent needs to record the sound event if I flush the toilet during the daytime.
3: Robot, could you chitchat with me if I yawn in the study room?
4: Robot, do not forget to remind put them back when I use scissors in the living room.
5: If I use a hair dryer, the agent needs to remind me to unplug it. But do not remind me so frequently.
6: The agent needs to set a timer if I brush tooth in the place where I cook meals.

TABLE V: Preference state numbers and feedback amount.

	P1	P2	P3	P4	P5	
Preference 1	864	576	108	72	576	
Preference 2	360	72	108	216	360	
Preference 3	864	32	432	48	792	
Preference 4	216	32	72	60	96	
Preference 5	864	360	72	432	648	**Mean**
Total State NUM	3168	1072	792	828	2472	1666
Feedback NUM	33	29	33	35	35	33

Whilst the proposed ChatAdp can achieve satisfactory results with minimal user feedback, there remain certain limitations inherent in the current methodology. First, the fine-tuning of ChatGPT is based on OpenAI's service. Therefore, the fine-tuning time varies. According to our observations, the duration of the fine-tuning varied considerably, ranging from one minute to as many as 30 minutes. Additionally, it should be noted that each inquiry directed towards obtaining a response from ChatGPT and fine-tuning comes at a financial expense. The expenditure required to complete the experiment is approximately 10 US dollars for the use of the ChatGPT service.

V. Conclusion and Future Work

In this paper, we proposed a ChatGPT-powered adaptation system, ChatAdp, to reduce the effort required from users in robot adaptation. In ChatAdp, we utilized ChatGPT to work as a user simulator that provides feedback to the robot instead of using real users. We tested ChatAdp in a case study of context-aware conversation adaptation. The evaluation results show that the proposed ChatAdp exhibits satisfactory performance while using fewer user interactions, therefore making the robots more acceptable to the users when they are deployed in real homes. In the future, we will evaluate the proposed method in more cases and with more users. We will also investigate the strategies for handling user preferences that are implicitly inferred rather than explicitly stated.

REFERENCES

[1] Z. Su, F. Liang, H. M. Do, A. Bishop, B. Carlson, and W. Sheng, "Conversation-based medication management system for older adults using a companion robot and cloud," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2698–2705, 2021.

[2] Y. Li, G. Yang, Z. Su, S. Li, and Y. Wang, "Human activity recognition based on multienvironment sensor data," *Information Fusion*, vol. 91, pp. 47–63, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1566253522001841

[3] Z. Su, Y. Li, and G. Yang, "Dietary composition perception algorithm using social robot audition for mandarin chinese," *IEEE Access*, vol. 8, pp. 8768–8782, 2020.

[4] M. Pham, H. M. Do, Z. Su, A. Bishop, and W. Sheng, "Negative emotion management using a smart shirt and a robot assistant," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 4040–4047, 2021.

[5] Z. Su, W. Sheng, G. Yang, A. Bishop, and B. Carlson, "Adaptation of a robotic dialog system for medication reminder in elderly care," *Smart Health*, vol. 26, p. 100346, 2022.

[6] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013. [Online]. Available: https://doi.org/10.1177/0278364913495721

[7] ChatGPT, https://openai.com/blog/chatgpt accessed in April, 2023.

[8] J. Huang, S. S. Gu, L. Hou, Y. Wu, X. Wang, H. Yu, and J. Han, "Large language models can self-improve," 2022.

[9] H. Ritschel, A. Seiderer, K. Janowski, S. Wagner, and E. André, "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*, ser. PETRA '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 247–255. [Online]. Available: https://doi.org/10.1145/3316782.3316791

[10] A. Slivkins, "Introduction to multi-armed bandits," *CoRR*, vol. abs/1904.07272, 2019. [Online]. Available: http://arxiv.org/abs/1904.07272

[11] G. Gordon, S. Spaulding, J. Kory Westlund, J. J. Lee, L. Plummer, M. Martinez, M. Das, and C. Breazeal, "Affective personalization of a social robot tutor for children's second language skills," vol. 30, Mar. 2016. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/9914

[12] A. H. Qureshi, Y. Nakamura, Y. Yoshikawa, and H. Ishiguro, "Intrinsically motivated reinforcement learning for human–robot interaction in the real-world," *Neural Networks*, vol. 107, pp. 23–33, 2018, special issue on deep reinforcement learning. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608018301072

[13] S. Vemprala, R. Bonatti, A. Bucker, and A. Kapoor, "Chatgpt for robotics: Design principles and model abilities," Microsoft, Tech. Rep. MSR-TR-2023-8, February 2023. [Online]. Available: https://www.microsoft.com/en-us/research/publication/chatgpt-for-robotics-design-principles-and-model-abilities/

[14] W. Yu, N. Gileadi, C. Fu, S. Kirmani, K.-H. Lee, M. G. Arenas, H.-T. L. Chiang, T. Erez, L. Hasenclever, J. Humplik, B. Ichter, T. Xiao, P. Xu, A. Zeng, T. Zhang, N. Heess, D. Sadigh, J. Tan, Y. Tassa, and F. Xia, "Language to rewards for robotic skill synthesis," 2023.

[15] N. Wake, A. Kanehira, K. Sasabuchi, J. Takamatsu, and K. Ikeuchi, "Chatgpt empowered long-step robot control in various environments: A case application," 2023.

[16] E. Levin, R. Pieraccini, and W. Eckert, "Learning dialogue strategies within the markov decision process framework," in *1997 IEEE Workshop on Automatic Speech Recognition and Understanding Proceedings*, 1997, pp. 72–79.

[17] M. Sewak, *Deep Q Network (DQN), Double DQN, and Dueling DQN*. Singapore: Springer Singapore, 2019, pp. 95–108. [Online]. Available: https://doi.org/10.1007/978-981-13-8285-7_8

[18] H. M. Do, M. Pham, W. Sheng, D. Yang, and M. Liu, "Rish: A robot-integrated smart home for elderly care," *Robotics and Autonomous Systems*, vol. 101, pp. 74 – 92, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889017300477