Towards Adaptive Robots based on Interaction Traces: A User Study

Abir B. Karami
University of Lyon, CNRS
University of Lyon 1
LIRIS, UMR5205, F-69622, France
Email: abir-beatrice.karami@liris.cnrs.fr

Karim Sehaba
University of Lyon, CNRS
University of Lyon 2
LIRIS, UMR5205, F-69676, France
Email: karim.sehaba@liris.cnrs.fr

Benoît Encelle
University of Lyon, CNRS
University of Lyon 1
LIRIS, UMR5205, F-69622, France
Email: benoit.encelle@liris.cnrs.fr

***Abstract*—We focus on the problem of adaptivity of companion robots to their users. Until recently, propositions on the subject of intelligent service robots were mostly user independent. Our work is part of the FUI-RoboPopuli project, which concentrates on endowing entertainment companion robots with adaptive and social behaviour. We concentrate on the capacity of robots to learn how to adapt and personalize their behaviour according to their users. Markov Decision Processes (MDPs) are largely used for adaptive robots applications. Several approaches were proposed to decrease the sample complexity to learn the MDP model, including the reward function. We proposed in previous work, two learning algorithms to learn the MDP reward function through analysing interaction traces (*i.e.* the interaction history between the robot and their users) including users' feedback. The first algorithm is direct and certain. The second is able to detect the importance of certain information, regarding the users (profiles) and/or the environment, in the adaptation process. We present, in this paper, a user study in addition to simulated experiments. Those experiments prove that our proposed algorithms are able to learn, through interactions with real users, a reward function that leads to an adapted and personalised robot behaviour. We also show the ability of our algorithms to handle exceptions and ambiguities in users feedback during experiments with real users.**

I. Introduction

Recent studies concentrate on user dependent approaches for companion robots applications. Whether in educational applications [1], or social and nursing applications [2], it is expected that robots take into account the fact that their environment includes several kinds of users with different age, preferences, needs, *etc.* Most approaches for adaptive robots are based on classic formal models, like Markov models. In [3], [4], the authors propose a Partially Observable Markov Decision Processes (POMDPs) to generate policies that select the best action corresponding to the current situation and inferred need of the user. However, the robot does not propose personalised behaviour according to the user profile. Another approach, based on rules, concentrates on adapting by analysing the interaction history with each user [5]. The robot adjusts its behaviour in consequence by proposing new and appropriate interactions. A different approach, presented in [6], concerns a rehabilitation robot that learn how to maximise the performance of users by adapting the personality of the robot according to the personality of the user. In these approaches, the robot does not learn the preferences of the user, however, it matches certain behaviour parameters (speed, voice, song to sing . . .) with the user's personality (extroversion, introversion) or with the history of interaction (sing a song after being introduced). In [7], the authors propose a framework where a user can shape the robot directly with good and bad signals in order to learn how to adapt to his preferences (as for a domestic dog). However, the approach is not proposed to learn personal preferences nor to handle environments including several users with different preferences.

In this paper, we focus on designing an adaptive robot that is able to personalise its behaviour according to the situation and its user. We also focus on connecting the most suitable behaviour to a set of user profile and/or environment particularities. Our proposed approach is based on a Markov Decision Process (MDP). We propose and validate two learning algorithms to learn MDP reward function that helps the robot in adapting its behaviour. The learning method is based on analysing the users feedback (as part of the interaction traces between the robot and the user) on the robot actions.

Our work is part of a research project called RoboPopuli, financed by the French ministry of industry. The project is interested in endowing the platform EMOX (EMOtion eXchange) of Awabot Corporation[1] with adaptive and social behaviour. The robot is a turtle bot composed of a camera, laser range-finder, pico projector, micro and speaker. The current working-on version of the robot works under Linux system and uses ROS as software framework. The idea behind the RoboPopuli project is to have a companion robot that proposes activities that respect the user preferences. Activities can be launched manually by the user (through tablet or smart-phone interfaces for example) or by proposition from the robot.

In the following: we present our general architecture and knowledge representation for an adaptive and personalised robot system in Section II. In Section III, we briefly present the approach we use for the robot planning system based on an MDP model. We present, in Section IV, our proposed learning algorithms to learn the reward function that will help the robot planner in its adaptation process. We then present our experimental results in Section V. The first experiment is to evaluate, by simulation, the resulted robot behaviour using the learning algorithms. The second experiment is a user study to prove the capacity of our algorithms in handling ambiguities

[1]www.emox-robot.com

in real users preferences (feedback). Finally, we conclude and present our future work.

II. General Architecture and Knowledge Representation

In this section, we detail the general architecture of the robotic system represented in Figure 1. The system input contains observations concerning users including their **Feedback** and the environment including potential users. The system output is the robot's actions. The system includes 4 knowledge bases: Interaction Traces, Activities, Users Profiles and Learned Rewards (detailed in this section) and two main processes: Decision Planning and Learning and Knowledge Extraction (described in Section III and IV respectively).

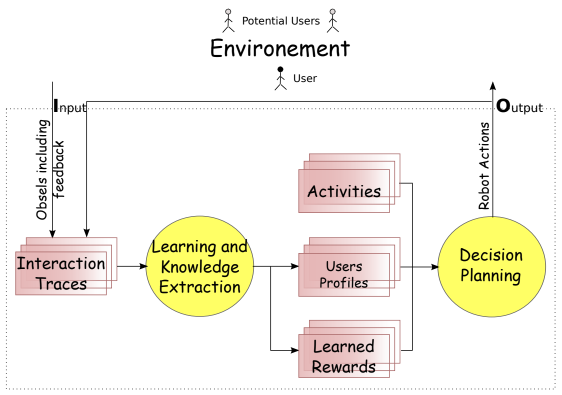


Fig. 1. General Architecture

The **Activities** database holds a description of all possible activities that the robot can share with or propose to its users (project a video to the user, play radio ...). Each activity has environmental attributes that are related to the activity $\mathcal{B}_{ac}$, each attribute $b \in \mathcal{D}_b$ has its domain of values $\mathcal{D}_b$.

Users Profiles hold information about users. Each user is represented by a profile $\mathcal{P}$. This profile contains all information concerning the user that might be useful in the adaptive decision process. Those information are represented as a set of attributes $\mathcal{B}_p$ and their values. Each attribute $x \in \mathcal{B}_p$ has its domain of values $\mathcal{D}_x$.

We based our model of **Interaction Traces** on the model defined by [8]. Generally speaking, a trace is a set of temporally situated **obsels** (OBServed ELementS). Obsels are generated during interactions between the system (*e.g.* the robot) and the environment (including the users). Each obsel has a type, defined by the trace model, and might be connected (in relation) with other obsels from the same trace. Formally, the obsel has a subject (*e.g.* the user or the robot), and a set of attributes/values that characterises the observed event. The trace model defines the obsels types and the relations between them in the trace.

We denote input obsels related to the user $\mathcal{O}_u$, the environment $\mathcal{O}_e$. The robot actions are integrated in the trace as output obsels $\mathcal{O}_r$. We note the set of obsels: $\mathcal{O} = \mathcal{O}_u \cup \mathcal{O}_e \cup \mathcal{O}_r$. The basic part of user related obsels $\mathcal{O}_u$ is his feedback which will be analysed to learn users preferences and how to adapt to them.

In our model, a *Trace* includes the sequence of input/output during *one activity* between the robot and a user. It is represented as described in Definition 1.

Definition 1: A trace $\mathcal{T}$ is represented by a tuple :

$$\langle id_t, id_p, id_{ac}, o_1, \dots o_i, \dots, o_n \rangle \text{ where,}$$

- id_t, id_p and id_{ac}: the trace id, the user id and the id of the activity represented in this trace, respectively.
- $o_1, \dots o_i, \dots, o_n$: the sequence of obsels related to the activity represented in this trace, where o_i: $\langle e, j, \mathcal{B}_o \rangle$:
  - The type of obsel $e \in \mathcal{O}$.
  - The obsel's subject depends on the obsel's origin: the user, the environment and potential users or the robot.
  - The set of obsel related attributes $\mathcal{B}_o$. An attribute $y \in \mathcal{B}_o$ has a value $v^o : y \to \mathcal{D}_y \cup null$.

From each trace, the system can extract several feedback rewards. The *feedback rewards* are gathered and processed later (through learning algorithms) to create the **Learned Rewards** that are integrated in the reward function of the MDP decision model. Both *feedback rewards* and *learned rewards* have the same representation form (Definition 2).

When processing an activity trace, each time an obsel of type feedback is encountered a feedback reward is created. The feedback reward holds in part the complete user profile and activity attributes values. However, as result of learning algorithms these attributes values can be generalised or described as *constraints* (logical expressions) over their possible values.

Definition 2: A feedback/learned reward is represented by a tuple: $\langle \mathcal{C}_p, \mathcal{C}_{ac}, id_{ac}, o_i, o_{i+1}, \mathcal{TR}, v \rangle$, where:

- $\mathcal{C}_p$: the constraints on the user profile attributes $\mathcal{B}_p$.
- $\mathcal{C}_{ac}$: the constraints on the activity attributes $\mathcal{B}_{ac}$.
- id_{ac}: the id of the activity.
- $o_i \in \mathcal{O}_r$: the robot action.
- $o_{i+1} \in \mathcal{O}_u$: the user feedback concerning robot action o_i.
- $\mathcal{TR}$: a backup of traces ids that provoked the creation or a modification of this learned reward.
- $v = \mathcal{V}(o_{i+1})$: the feedback value (the received reward). $\mathcal{V}$ is a predefined value function[2] that assigns a positive or negative weight for each feedback $\mathcal{V} : \mathcal{O}_u \to [-1, 1]$.

III. Robot Planning

A Markov Decision Process (MDP) is represented by a tuple $\langle S, A, T, R \rangle$ where: S is a set of states, A is the set of robot's actions, T is the transition function where $T(s, a, s')$ is the probability of transitioning from state s to state s' after doing action a and R is the reward function, where $r(s, a)$ is the robot's immediate reward for making action a while being in state s. An MDP policy π_{MDP} is

[2]This function follows the assumption that the user's feedback is received directly after the robot action. It is possible to define a function $v = fct(\mathcal{V}(o_1), \dots, \mathcal{V}(o_{i+1}))$ using a heuristic (*e.g.* propagation).

a function $\pi : S \rightarrow A$, which associates an action to each MDP state. There are several algorithms to solve an MDP, classically Value Iteration [9] and Policy Iteration [10]. The complexity of such algorithms is $\mathcal{O}(|S^2||A|)$. If the agent has no knowledge about its environment, the transition and reward functions will be hard to calculate. In this case, the agent can directly learn its policy using reinforcement learning such as for Q-learning [11] that does not require prior knowledge about the transition function. Inverse reinforcement learning algorithms [12] extracts a reward function from an observed optimal behaviour (learn by observation). On the other hand shaping algorithms [7] extracts a reward function from human feedback on agent's behaviour.

We use an MDP for the robot decisions. The desired robot behaviour is adaptive and personalised to the situation and the user. In our problem, the reward function is not known for the robot, however, users' feedback are used to learn it. We propose two learning algorithms to learn the reward function.

IV. Learning From User Feedback

We propose two learning algorithms. The first is a direct and certain, it can be considered as the optimal version for building a certain at all times reward function. The second generalises the learned rewards to be applied on unknown profiles and in unknown situations, this version is risky at its early stages of learning before it merges to a correct generalisation.

Both algorithms learn from received set of Feedback Rewards ($\mathcal{FR}$). Feedback rewards are extracted from the interaction traces. The extraction of a feedback reward $fr = \langle \mathcal{C}_p^{fr}, \mathcal{C}_{ac}^{fr}, id_{ac}^{fr}, o_i^{fr}, o_{i+1}^{fr}, \mathcal{TR}^{fr}, v^{fr} \rangle$ from a trace $\mathcal{T} = \langle id_p, id_{ac}, o_1, \ldots o_i, \ldots, o_n \rangle$ is done simply by filling the profile id_p and the activity id_{ac} attributes values in $\mathcal{C}_p^{fr}, \mathcal{C}_{ac}^{fr}$ respectively. o_i^{fr} is an obsel from $\mathcal{T}$ of type $\mathcal{O}_r$ that precedes another obsel $\mathcal{T}$ of type $\mathcal{O}_u$ (user feedback). The latter is set to be o_{i+1}^{fr}. $\mathcal{TR}$ holds the id of the trace $\mathcal{T}$ and v is the value of the user feedback $v = \mathcal{V}(o_{i+1})$. The output of both algorithms is the set of learned rewards $\mathcal{LR}$ modified after processing every feedback reward fr. $\mathcal{LR}$ is then added as a part of the MDP reward function.

A. The Certain Direct Learning Algorithm

In this algorithm, the addition of the feedback reward to the learned rewards $\mathcal{LR}$ is done in a simple and direct way:

1) If fr holds the same information as one $lr \in \mathcal{LR}$; meaning that fr and lr have the same activity id id_{ac} and the same robot action o_i and that profile and activity attributes of fr satisfy their corresponding constraints in lr, then:
   a) If both fr and lr have the same feedback value direction (FD) (*i.e.* both v values in fr and lr are positive or negative, then lr stays in $\mathcal{LR}$ after modifying the feedback value with $fct(v^{lr}, v^{fr})$.
   b) If fr and lr have different feedback directions (FDs), then a *contradiction* is detected (the received feedback contradicts with the existing learned knowledge) and both rewards are flagged (added to $\mathcal{EC}$). The $\mathcal{EC}$ set is then sent to be reviewed by an expert to detect the reason of the contradiction. This contradiction might refer to missing attributes in the problem representation[3].
2) If fr holds a new situation to the learned rewards, then fr is added as new learned reward in $\mathcal{LR}$.

B. The Generalised with Risk Learning Algorithm

We aim in this algorithm to learn the reward function faster by minimizing the needed number of experiences. In addition, we aim to be able to generalise the learned function to be applied with unknown profiles and in unknown situations. Differently from the first algorithm, this algorithm tries to detect, for each possible robot action, the important attributes (related to profile or activity) that their values affect the user feedback value direction (FD) (*i.e.* v is positive or negative). Attributes that their values are not important are generalised to any value (*) in all $\mathcal{LR}$ rewards. This algorithm backs up all feedback rewards, so there is no loss of information because of the generalisation. The backup rewards are continuously used in the process of detection of important attributes. In the following, we describe, with examples, (a) how the algorithm uses contradictions between received fr and generalised rewards in $\mathcal{LR}$ to detect important attributes and also (b) how it is possible to detect attributes that were falsely set as important in the first phase.

a) Treating contradictions to extract important attributes: The algorithm search for a contradiction between the feedback values of the received fr one of the lr rewards. If a contradiction is detected, the algorithm tests if for an important attribute concerning the action o_i (shared robot action in fr and lr). First, a not empty set of backup feedback rewards that are related to lr is set to $\mathcal{BR}$. We mean by related that they concern the same robot action, they have the same FD and that the attributes values in br are included in the constraints of lr. Second, the algorithm looks for br with only one attribute value different from fr and with different FD. If found, the concerned attribute is marked as important.

For example, a problem with 2 attributes A and B with possible values a_1, a_2 and b_1, b_2 respectively. If we have $br = \langle a_1, b_1, -1 \rangle$ and $lr = \langle *, *, -1 \rangle$ and the next received $fr = \langle a_2, b_1, +1 \rangle$, then the algorithm will detect a contradiction between fr and lr and the values of the attribute A will be detected as important (the FD change of direction based on the value of A while B has the same value).

b) Checking for false important attributes: It is possible that the algorithm marks some attributes as important when they are actually not. Therefore, we added a function that checks for falsely detected important attribute. For example, if we have fr and lr who have the same FD with only one different attribute value, this attribute is marked as falsely detected important attributes and is removed from the list of important attributes for the concerned action if the following condition is true: there exist in the backup rewards a list of rewards covering all the combinations of *important* attributes with the same FD.

[3]A user might change his preference in a rainy day which informs us the need of adding weather forecast as an attribute.

V. Experiment Results

In this section, we present two experiments to test both learning algorithms. The first uses simulations and evaluates the robots MDP actions that uses the learned reward functions. The second analyses the resulted learned rewards using feedback from real users. In the simulated experiments, the learned reward value update function $fct(v^{lr}, v^{fr})$ returns the average of both values. However, in the real users experiments, it returns a value that represents the percentage of positive and negative users feedback (the ambiguity in users preferences) as in Equation 1.

$$fct(v^{lr}, v^{fr}) = \begin{cases} \frac{max\_number(yes,no)}{Sum\_number(yes,no)} & \text{if} |yes| > |no| \\ -\frac{max\_number(yes,no)}{Sum\_number(yes,no)} & \text{if} |no| > |yes| \end{cases} \quad (1)$$

A. Simulations

We first experimented our learning algorithms through simulation. For this simulation, we used the activity of projecting a video to the user. The MDP state represents, the possible users' profiles (age and gender) and activity related information (noise, daytime, brightness and phase). The MDP set of actions included 21 different actions to select (one of 4 possible rooms to project, one of 2 possible video lengths, one of 8 possible video types, one of 3 possible volume levels, one of three possible brightness levels, and finally project action). A default reward function is given to respect the sequence of phases for realising the activity (moving to the right room, choosing the length of the video, choosing the video gender, setting the sound, setting the brightness and finally projecting).

The procedure of evaluation is a loop of the following: (1) Re/Calculate the MDP policy. (2) Generate n traces. (3) Evaluate robots actions in the n traces. (4) Extract feedback rewards from traces, then learn and update the MDP reward function. (5) Repeat from step 1 until convergence.

The generation of the traces is made by simulation using the MDP policy for generating the robot action and some *predefined rules of preferences* to generate users feedback. We had predefined rules for each action of each phase which is connected to certain attributes of the activity or the profile (*e.g.* the room selection depends on the age of the user and the day time, and the video gender depends on the age and the user gender).

We evaluated both algorithms to learn the reward function on the same randomly generated situations (user profile and activity related information). We followed the procedure described earlier with $n = 100$ traces. After each n traces we calculated the number of robot's actions that were followed with a negative user feedback (negative actions). We also compared the number of complete positive traces (*i.e.* positive experiences), where the robot succeeded to achieve a complete activity with the user without receiving any negative feedback.

Results shown in Figure 2 present the convergence of both learning algorithms to an optimal adaptive and personalised behaviour with no negative actions. Both algorithms were able to learn, at 100%, the rules of preference. It shows that the second algorithm with generalisation creates 10% of negative actions during the first 100 traces and converges to 0% negative actions after 500 with exception. In the first 100 traces, there was 66 completely positive experiences resulting from the algorithm with generalisation. We confirm that this algorithm was able to learn the dependencies between the profile and activity attributes with each robot action. Important attributes that were predefined for the simulation were completely learned at the end of the experiment. On the other hand, the direct algorithm started with 37% of negative actions during the first 100 traces and converged after 1000 traces. The first 100 traces included 42 positive experiences only.

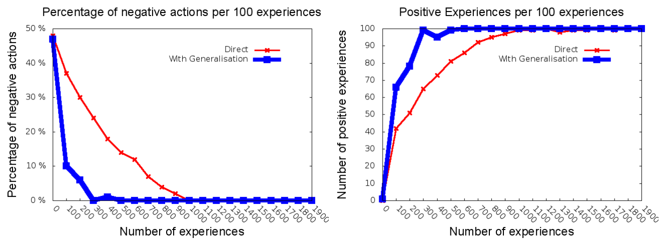


Fig. 2. Comparison between the direct algorithm and the algorithm with generalisation: the number of negative actions and the number of positive experiences (with zero negative feedback) presented per 100 experience

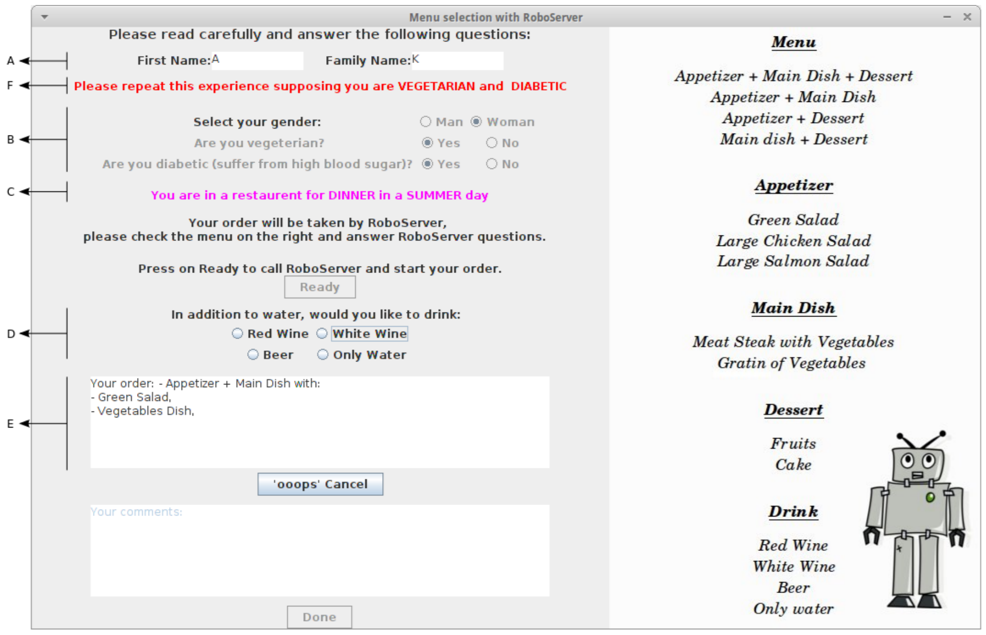
In the evaluated simulations, the predefined rules of preference were followed at all times. This means there was no ambiguity in the rules of preference. However, we know that this is far from true in real applications. For example, most but definitely not all male adults like to watch sport related programs. For this reason, we present, in this paper, a second experiment with real users in order to study the ability of our proposed algorithms to handle ambiguities while holding probabilities on rules of preferences.

B. Evaluation by Real Users

The Scenario: We chose for this experiment the activity of selecting a menu in a restaurant with the help of RoboWaiter. A user profile includes information about his/her gender (male or female), vegetarianism (vegetarian or not) and diabetes (diabetic or not). Activity related information are the meal time (noon or evening) and the season (winter or summer). The menu is described in Figure 3 (on the right). The experiment is done using a user interface (another experiment will be done with a real robot in future work). When the user finishes selecting his/her profile information, the system asks the user about his preferences about the menu type (Appetizer + Main Dish + Dessert, Appetizer + Main Dish, Main Dish + Dessert, Appetizer + Dessert) and with respect to his/her choice, continues questioning for each of the corresponding entries (Appetizer, Main Dish, Dessert and Drink). Questions about the menu type and the drink are asked in all experiences. We call experience, a complete sequence of interactions from the moment of starting the menu selection until the order is completely filled. Each experience is represented in one *Trace*.

What we wish to experiment in this study is the capability of our algorithms to:

- Handle **ambiguity** (*i.e.* different users with similar profiles that are in the same situation and have different



Please read carefully and answer the following questions:

First Name: A Family Name: K

Please repeat this experience supposing you are VEGETARIAN and DIABETIC

Select your gender: Man [✓ Woman]

Are you vegeterian? [✓ Yes] No

Are you diabetic (suffer from high blood sugar)? [✓ Yes] No

You are in a restaurent for DINNER in a SUMMER day

Your order will be taken by RoboServer,
please check the menu on the right and answer RoboServer questions.

Press on Ready to call RoboServer and start your order.

Ready

In addition to water, would you like to drink:
Red Wine White Wine
Beer Only Water

Your order: - Appetizer + Main Dish with:
- Green Salad,
- Vegetables Dish,

'ooops' Cancel

Your comments:

Done

Menu

Appetizer + Main Dish + Dessert
Appetizer + Main Dish
Appetizer + Dessert
Main dish + Dessert

Appetizer

Green Salad
Large Chicken Salad
Large Salmon Salad

Main Dish

Meat Steak with Vegetables
Gratin of Vegetables

Dessert

Fruits
Cake

Drink

Red Wine
White Wine
Beer
Only water

Fig. 3. User interface for the menu selection activity.

reactions/feedbacks) in the representation of the learned reward function.

- Prove the capability of the algorithm with generalisation to detect relations between some users' choices and a set of attributes' values despite the potential ambiguities. Mainly: the importance of the vegetarianism attribute value in the selection of "Meat Dish, Chicken Salad and Salmon Salad", the importance of the diabetes attribute value in the selection of "Cake", and possibly other relations like drink preferences depending on season or menu type depending on meal time.

The interface is shown in Figure 3. It begins with the fields of identification (Mark A) and then profile selection (Mark B). The activity related situation is described after selecting the profile (Mark C). After giving some guidance information to the user and when the user feels ready to start his order, questions to obtain his/her choice are given sequentially (Mark D). The user choices are added to his/her order (Mark E) to be verified and confirmed.

The Protocol: The experiment was done with 25 different adult users (14 men and 11 women). Given the lack of access to diabetic and vegetarian users, each of our 25 users repeated the experience 4 times. The first experience using his/her real profile and 3 other experiences with fake profiles covering the other 3 possibilities from: vegetarian/diabetic, non-vegetarian/non-diabetic, vegetarian/non-diabetic and non-vegetarian/diabetic (See Mark F in Figure 3). The activity related situations (lunch/dinner and summer/winter) were set randomly and in an equal number of times for each possible user gender. The activity related situations were the same during the 4 experiences of one user. The duration of experiment (4 experiences) with each user took between 3 to 6 minutes.

The Results: We present in Table I the percentages of users positive answers (Yes) for each question they had. The table categorises the results depending on each attribute value during the experience (the columns). For example, considering all experiences where the user was male and was asked his preference about the Meat Dish; in 30% of these experiences the user answered Yes (70% answered No). In contrary, for the question related to Vegetables Dish, 70% of male users answered Yes (30% answered No), knowing that only users who chose menus types including Main Dish were asked those 2 questions. Percentages in this table are a categorised form of the reward function learned by the direct algorithm where reward values are between [-1,1] (Yes percentages that are greater than 50% represent the positive rewards and those that are less than 50% represent the negative rewards). An example of a reward in the reward function: $R(s = \langle male, vegetarian, diabetic, dinner, winter, Appetizer + Main\_Dish\rangle, a = only\_water?) = 0.0$, which means that the gained reward/penalty when doing action $a = only\_water?$ in state s is 0.0. This reward was calculated using Equation 1 (knowing the number of yes and no answers of users of same profile and in the same situation). In this example the robot received one time yes and one time no as answers, therefore, the reward is 0.0. Another example, with $s = \langle male, non\_vegetarian, diabetic, dinner, summer, Appetizer + Main\_Dish\rangle$ and $a = only\_water?$, the robot received 2 times yes and one time no as answers. Therefore, the calculated reward is $2/(2+1) = 0.66$.

In Table II, we show the detected important attributes by the algorithm with generalisation. We notice the detection of mainly important attributes related to vegetarianism and diabetes. For example, we notice that vegetarianism was detected as important attribute for the Green Salad choice (which was chosen 98% of times by vegetarian users I). Also vegetarianism was detected as important for the Meat Dish (which was chosen 61% of times by non-vegetarian users). Diabetes was detected important for Appetizer + Main Dish menu type, where diabetic users (preventing desserts) took this choice 82% of times. Meal time was detected important for the choice of

		Male/Female	Vegetarian/Non-Vegetarian	Diabetic/Non-Diabetic	Lunch/Dinner	Summer/Winter
Appetizer + Main Dish	Yes %	52/43	46/50	82/14	52/44	52/44
Appetizer + Dessert	Yes %	0/9	6/2	2/6	0/8	6/2
Main Dish + Dessert	Yes %	11/20	16/14	4/26	15/15	19/12
Appetizer + Main Dish + Dessert	Yes %	38/27	32/34	12/54	33/33	23/42
Green Salad	Yes %	60/63	98/26	60/62	59/64	62/61
Salmon Salad	Yes %	26/23	2/47	27/22	20/30	23/26
Chicken Salad	Yes %	14/14	0/28	12/16	22/7	15/13
Meat Dish	Yes %	30/32	0/61	27/36	33/29	31/31
Vegetables Dish	Yes %	70/68	100/39	73/64	67/71	69/69
Fruits	Yes %	59/68	63/64	100/56	78/52	65/62
Cake	Yes %	41/32	37/36	0/44	22/48	35/38
Red Wine	Yes %	18/9	6/22	4/24	2/25	17/12
White Wine	Yes %	9/5	12/2	2/12	4/10	8/6
Beer	Yes %	7/5	6/6	0/12	8/4	4/8
Only Water	Yes %	66/82	76/70	94/52	85/62	71/75

TABLE I. RESULTS OF THE DIRECT ALGORITHM: USERS ANSWERS CATEGORISED BY ATTRIBUTE VALUE, PERCENTAGE OF POSITIVE AND NEGATIVE ANSWERS.

fruits as dessert, where 78% of users chose Fruits as dessert for Lunch. vegetarianism was also detected important for the choice of White Wine, where 12% of vegetarian users versus 2% of non-vegetarian users had this choice.

Appetizer + Main Dish	diabetes, daytime
Appetizer + Dessert	vegetarianism
Main Dish + Dessert	daytime, diabetes
Appetizer + Main Dish + Dessert	sex, season
Green Salad	vegetarianism, diabetes
Salmon Salad	-
Chicken Salad	diabetes
Meat Dish	vegetarianism, season
Vegetables Dish	-
Fruits	vegetarianism, daytime
Cake	-
Red Wine	sex
White Wine	vegetarianism, diabetes, daytime, season
Beer	diabetes, season
Only Water	-

TABLE II. DETECTED IMPORTANT ATTRIBUTES BY THE ALGORITHM WITH GENERALISATION.

On the other hand, diabetes was not detected as important attribute for Fruits choice, even with a probability of 100% of diabetic users choosing Fruits as dessert [4]. Also, Only Water was more chosen than other type of drinks by diabetic users during all meal times and all types of seasons, however, non of these attributes were detected as important for the Only Water choice.

In conclusion, optimally detecting the important attributes while dealing with real users and ambiguities is not simple. However, the question is, can the actual detected attributes lead to an acceptable adaptive behaviour? In future work, we would like to test the learned rewards of both algorithms with the same users in order to evaluate the quality of the resulted MDP policy.

VI. CONCLUSION AND FUTURE WORK

In this paper, we presented an architecture and learning methods for an adaptive robot that learns users' preferences using users' feedback. Simulated experiments showed the ability of the robot learn, adapt and personalise its behaviour to its users. We also focused on experiments with real users to prove the capability of our proposed algorithms to handle ambiguities in users' feedback. Results show that ambiguities were well handled using probabilities in representing the reward function and that most important attributes were detected using the algorithm with generalisation.

In future work, we will assess the learned reward function in another user study and we will also experiment our results with a real robot. We aim to evaluate the general architecture with several activities and the adaptation on the high level (selection of activity) and low level (during an activity).

[4]This percentage covers users who chose a menu type with dessert.

REFERENCES

[1] M. Saerbeck and T. Schut, "Expressive robots in education: varying the degree of social supportive behavior of a robotic tutor," *In CHI '10: Proceedings of the 28th international conference on Human factors in computing systems*, pp. 1613–1622, 2010.

[2] C. Breazeal, J. Gray, and M. Berlin, "An Embodied Cognition Approach to Mindreading Skills for Socially Intelligent Robots," *The International Journal of Robotics Research*, vol. 28, no. 5, pp. 656–680, May 2009.

[3] A. B. Karami and A.-i. Mouaddib, "A Decision Model of Adaptive Interaction Selection for a Robot Companion," *proceedings of the 5th European Conference on Mobile Robots, ECMR'11*, pp. 83–88, 2011.

[4] M. Pollack, L. Brown, and D. Colbry, "Pearl: A mobile robotic assistant for the elderly," *Workshop on Automation as Caregiver: the Role of Intelligent Technology in Elder Care (AAAI)*, 2002.

[5] T. Kanda and H. Ishiguro, "Communication robots for elementary schools," *Proceedings of AISB'05 Symposium Robot Companions: Hard Problems and Open Challenges in Robot-Human Interaction (Hatfield Hertfordshire)*, pp. 54–63, 2005.

[6] A. Tapus, C. Tapus, and M. J. Matarić, "User-robot personality matching and assistive robot behavior adaptation for post-stroke rehabilitation therapy," *Internationl Journal on Intelligent Service Robotics*, vol. 1, no. 2, pp. 169–183, Feb. 2008.

[7] W. B. Knox and P. Stone, "Interactively shaping agents via human reinforcement: the tamer framework," in *Proceedings of the fifth international conference on Knowledge capture*, ser. K-CAP '09. New York, NY, USA: ACM, 2009, pp. 9–16.

[8] D. Clauzel, K. Sehaba, and Y. Prié, "Enhancing synchronous collaboration by using interactive visualisation of modelled traces," *Simulation Modelling Practice and Theory*, vol. 19, no. 1, pp. 84–97, jan 2011.

[9] R. Bellman, "A Markovian Decision Process," *Indiana University Math. J.*, vol. 6, pp. 679–684, 1957.

[10] R. A. Howard, *Dynamic Programming and Markov Processes*. Cambridge, MA: MIT Press, 1960.

[11] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. MIT Press/Bradford Books, 1998.

[12] A. Y. Ng and S. Russell, "Algorithms for inverse reinforcement learning," in *in Proc. 17th International Conf. on Machine Learning*. Morgan Kaufmann, 2000, pp. 663–670.