Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea Magistrale in Informatica

Generalize Policy on Supporting User Scenario

Relatore:
Chiar.mo Prof.
Mauro Gaspari

Presentata da:
Lorenzo Cazzoli

Sessione II
Anno Accademico 2016/2017

a .... Cazzoli

mio nipote,

che nascerá in questi giorni

Abstract

In this thesis we present a way of combining previously learned robot behavior policies of different users. The main idea is to combine a set of policies, in tabular representation, into a final sub-optimal solution for the problem all users have contributed to. We assume that the features/differences of users are unknown and need to be extracted from the different policies generated from same user. This information is used to weight the importance of a set of actions to sum up two policies.

The proposed approach has been tested on a virtual environment finding out that the combined policy works as a general policy suitable for all users, as it always selects actions that are satisfying the users at the border of the defined sensorial possibilities.

All the assumptions has been finally verified on a real environment finding out all the limitations of the proposed model.

Sommario

La Human-Robot-Interaction é un area di studio che coinvolge diverse discipline. Le differenti categorie di possibili target condizionano il tipo di sfida a cui si é posti di fronte. Per esempio, piattaforme robot pensate per i bambini devono adattarsi alle loro capacitá comunicative e al loro comportamento in continua evoluzione. Al contrario piattaforme per persone anziane devono adeguarsi ad eventali deficit delle capacitá visive ed uditive. I robot hanno bisogno di potersi adattare alle capacitá e deficit delle persone con cui stanno interagendo. Ci si aspetta che nel caso di utenti con problemi alla vista, il robot usi piú suggerimenti vocali, mentre per utenti con problemi uditivi siano usate le mimiche facciali e altri movimenti.

Attraverso l'uso del Reinforcement Learning [Sutton and Barto(2011)] i robot possono facilmente imparare il comportamento corretto per ogni gruppo di utenti indipendentemente dalle loro differenze sensoriali. Il problema é che le informazioni imparate dall'interazione con uno di questi non possono essere usate nel processo di apprendimento di un altro, poiché il secondo utente potrebbe avere diverse capacitá e quindi non garantire che le azioni siano percepite nello stesso modo causando gli stessi risultati. Le policy imparate permettono al robot di comportarsi correttamente solo con l'utente corrente, ma quando il robot é di fronte a un nuovo utente, deve aggiornare la sua policy per adattarsi alla nuova situazione. Se i due utenti non sono troppo diversi, il robot é in grado di generare una nuova policy affidabile, mentre, nel caso contrario, il comportamento generato potrebbe

essere fuorviante a causa dell'imprecisione della policy.

Combinando le policy vogliamo superare questo problema garantendo che il robot si comporti correttamente, con un comportamento sub-ottimale, con tutti gli utenti incontrati e senza il bisogno di imparare una policy da zero per i nuovi. Questo fatto é ancora piú importante quando si impara online interagendo con persone che sono sensibili agli errori e al passare del tempo. In questo caso la policy combinata é usata per inizializzare il processo di apprendimento, accelerandolo e limitando gli errori dovuti all'esplorazione iniziale.

In questa ricerca presentiamo un modo per combinare policy di utenti diversi precedentemente apprese. L'idea é di combinare un insieme di policy, rappresentate in forma tabulare, in una soluzione sub-ottimale al problema a cui tutti gli utenti hanno contribuito. Nel fare ció assumiamo che le caratteristiche degli utenti siano sconosciute e quindi necessitino di essere estratte dalle differenze evidenziate dal comfronto di policy generate interagendo con lo stesso utente. Le informazioni estratte in questo modo sono usate per pesare l'importanza delle azioni scelte per l'utente nel processo di *policy combination*.

Per iniziare sono state raccolte le policy da utenti simulati che dovevano completare il gioco del memory, assistiti da un agente virtuale. Ogni volta che l'utente cercava una coppia sull'area di gioco, l'agente, che era a conoscenza della posizione della carta da trovare, lo aiutava suggerendo la direzione o l'area del tabellone in cui tentare. Per permettere l'adattamento del comportamento dell'agente, si é utilizzato un algoritmo di apprendimento chiamato Q-learning insieme alla tecnica ϵ-greedy per bilanciare *exploration* ed *exploitation* [Watkins and Dayan(1992)]. La comunicazione é stata implementata tramine la simulazione di azioni multimodali composte da sguardo, voce, movimenti della testa ed espressioni facciali.

Un insieme di *personas* [Schulz and Fuglerud(2012)] é stato generato per coprire i diversi tipi di utenti e capire i loro bisogni. Tramite le storie di ogni personas é stato estratto un set di caratteristiche usato per definire il comportamento degli utenti virtuali. Per esempio un utente sordo ignorerá tutti i suggerimenti vocali e seguirá solo le altre parti delle azioni multimodali.

La bontá dei risultati ottenuti nelle simulazione dello scenario, ci ha motivato ad estendere la ricerca con test sullo scenario reale. Questo é stato realizzato utilizzando una piattaforma touch, su cui sono state disposte le carte del memory, posizionata di fronte a FurHat, il robot usato per interagire con l'utente. Un gruppo di volontari é stato usato per la raccolta dei dati, da cui é stata generata la combined policy utilizzata su un gruppo allargato di volontari al fine di validare i risultati ottenuti dalle simulazioni con le personas e quindi l'algoritmo proposto.

La tesi é divisa nei seguenti capitoli: nel capitolo 2 viene fornita una panoramica delle ricerche correlate; il capitolo 3 introduce le basi teoriche richieste per lo studio in questione; il capitolo 4 presenta lo scenario utilizzato nei test, la cui implementazione é trattata nel capitolo 5; nel capitolo 6 e 7 sono riportate le scelte e risultati relativi alle simulazioni effettuate rispettivamente nello scenario virtuale e reale; il capitolo 8 riporta una breve riflessione conclusiva e le possibili direzioni future della ricerca.

Contents

Summary of Notation

Capital letters are used for random variables and major algorithm variables. Lower case letters are used for the values of random variables and for scalar functions. Quantities that are required to be real-valued vectors are written in bold and in lower case (even if random variables).

s	state
a	action
S	set of nonterminal states
S^+	set of all states, including the terminal state
$A(s)$	set of action possible in state s
t	discrete time step
S_t	state at t
A_t	action at t
R_t	reward at t
π	policy, decision making rule
$\pi(s)$	action taken in state s under deterministic policy π
w	set of parameters
ϕ	basis functions
w_π	parameters for the approximated function of π
v_π	expected return under policy π

$v_\pi(s)$	value of state s under policy π
$\hat{Q}(s,a,w)$	approximate value of state-action pair s, a given weight w
α	learning rate parameter
γ	discount rate parameter
ϵ	probability of random action in ϵ-greedy policy
ψ	probability of past policy action in policy reuse algorithm

Chapter 1

Introduction

Human-Robot-Interaction is an increasingly studied field with many disciplines involved. The different categories of targets influence the type of challenges that need to be accomplished. For instance, robot platforms designed for children have to adapt to their maturing behavior and communication abilities. On the other hand, platforms for elderly people have to accommodate to the digressive visual and hearing capacities. The robots need to adapt their behavior to the capabilities and deficits of their interaction partners. It is expected that if the user has some visual lack the robot will use more voice suggestion and in the opposite case, with lacking hearing abilities, more facials expressions or gestures are required.

Using Reinforcement Learning [Sutton and Barto(2011)] robots can easily learn the correct behavior for each group of users independently from their sensorial differences. The problem is that information learned interacting with one user can't be used in a learning process of another user because they presumably have different capabilities, thus it is not guaranteed that two actions are perceived in the same way and cause the same results. The learned policy allows the robot to infer a correct behavior only for the current user. The way in which the user is interacting and interpreting the agent's actions define the problem and the learned behavior. When the robot interacts with a new user it has to update its policy to adapt to the new situation. If the two users are not too different, the robot is able to generate a new reliable behavior policy. Otherwise the generated behavior could be misleading or frightening due to an inaccurate policy. An agent that has to assist multiple and different users has to learn *different behavior* for each user. In addition is required a method to recover already learned behavior for users whose presence recurs more than one time. This is not immediate because it need memory to store the learned policies and an identification system to recognize each user.

Combining policies we can overcome this problem guaranteeing that the robot behaves properly, with a sub-optimal solution, with all encountered users and without the need to learn again a policy from scratch for the current user. This fact is even more important when the system has to interact with humans. In this case the combined policy is used to initialize the learning process, with the target of speed up the learning and limit the initial wrong suggestions due to exploration.

In the past different solutions have been tested to speed up the learning on different tasks performed by the same user. Examples of there are [Taylor and Stone(2009), Sherstov and Stone(2005), Fernández and Veloso(2006), Konidaris and Barto(2006)] where the information learned about a task are used in the resolution of a similar one. They are all interesting approaches, but they are not suitable for the problem that we have in question, since we are working with different users and they are thought to work in different task of the same user.

The policies combination is similar to the combination of the policies performed by an arbitrator on the information learned by the different subagents. This combination techniques are used by [Russell and Zimdars(2003), Rosenblatt(2000)] in order to find new way to select the optimal action in distributed scenarios.

For a successful combination of the policy is important to understand the deficits of the user situated in front of the agent. this is important to make the correct use of policy learned on him. No one would take in consideration the visual suggestions learned during the interaction with a blind user. Understand the users is vital to understand the different way on which the users tackle the problem and this knowledge is used to find a common interpretation that defines the base of a general behavior. In all the situations where different users react to different actions the optimal action need to be selected between the performed actions. If the scenario include multimodal actions every user can influence the final action with parts of his behavior, the parts on which his senses are not lacking in perception.

As in [Griffith et al.(2013)Griffith, Subramanian, Scholz, Isbell, and Thomaz, Thomaz and Breazeal(2008)] human feedback can be used, in the online learning, to improve the quality of the reward and to drive the training of the agent. Used in the correct ways they can also accelerate the learning obtaining good policies even after few rounds.

The goals of this thesis can be summarized in find an implementation of the policy combination algorithm in order to:

1. Use the same policy for multiple users
2. Decrease the time required for online training.

The rest of the thesis is structured as follow: in chapter 2 an overview of recent research organized in three categories is given. They are divided on groups that are concerning the different phases of our research. In the chapter 3 are introduced the concepts and the theoretical background required for the studies. The scenario used for the simulation is presented in the chapter 4, while all the implementations are illustrated in the chapter 5. Chapters 6 and 7 are reporting the choices and results of the simulations performed on the virtual and real world scenarios. The last chapter, the number 8, presents the conclusion and future works.

Chapter 2

Related Work

In the recent past there aren't similar works done in the research communities, that is instead focused on different forms of *knowledge transfer*.

In this chapter we will first cite some of these and after we will see some approaches used to solve different research questions, from which we have taken inspiration. The last section are works that explain how human feedback can be used to improve online training.

2.1 Transfer of Knowledge

The reinforcement learning paradigm is a popular way to address problems that have only limited environmental feedback, rather than correctly labeled examples, as is common in other machine learning contexts. While significant progress has been made to improve learning in a single task, the idea of transfer learning has only recently been applied to reinforcement learning tasks. The core idea of transfer is that experience gained in learning to perform one task can help improve learning performance in a related, but different, task. [Taylor and Stone(2009)] present a framework that classifies transfer learning methods in terms of their capabilities and goals, and then use it to survey the existing literature, as well as to suggest future directions for transfer learning work.

In the paper of [Sherstov and Stone(2005)] is presented action transfer, a novel approach to knowledge transfer across tasks in domains with large action sets. They demonstrate the use of knowledge transfer between related tasks to accelerate learning with large action sets. The algorithm rests on the idea that actions relevant to an optimal policy in one task are likely to be relevant in other tasks. Their technique extracts the actions from the optimal solution to the first task and uses them in place of the full action set when learning any subsequent tasks. When optimal actions make up a small fraction of the domain's action set, action transfer can substantially reduce the number of actions and thus the complexity of the problem. However, action transfer between dissimilar tasks can be detrimental.

[Fernández and Veloso(2006)] contribute Policy Reuse as a technique to improve a reinforcement learning agent with guidance from past learned similar policies. Their method relies on using the past policies as a probabilistic bias where the learning agent faces three choices: the exploitation of the ongoing learned policy, the exploration of random unexplored actions, and the exploitation of past policies. Policy Reuse also identifies classes of similar policies revealing a basis of core policies of the domain. They demonstrate that such a basis can be built incrementally, contributing the learning of the structure of a domain.

[Konidaris and Barto(2006)] introduce the use of learned shaping rewards in reinforcement learning tasks, where an agent uses prior experience on a sequence of tasks to learn a portable predictor that estimates intermediate rewards, resulting in accelerated learning in later tasks that are related but distinct. Such agents can be trained on a sequence of relatively easy tasks in order to develop a more informative measure of reward that can be transferred to improve performance on more difficult tasks without requiring a hand coded shaping function.

2.2 Composite Agent

[Russell and Zimdars(2003)] explores a very simple agent design method called Q-decomposition, wherein a complex agent is built from simpler subagents. Each subagent has its own reward function and runs its own reinforcement learning process. It supplies to a central arbitrator the Q-values for each possible action. The arbitrator selects an action maximizing the sum of Q-values from all the subagents. This approach has advantages over designs in which subagents recommend actions. It also has the property that if each subagent runs the Sarsa reinforcement learning algorithm to learn its local Q-function, then a globally optimal policy is achieved.

[Rosenblatt(2000)] introduces a new means of action selection via utility fusion. Distributed asynchronous behaviors indicate the utility of various possible states and their associated uncertainty. A centralized arbiter then combines these utilities and probabilities to determine the optimal action based on the maximization of expected utility. The construction of a utility map allows the system being controlled to be modeled and compensated for.

2.3 Human Feedback

While Reinforcement Learning (RL) is not traditionally designed for interactive supervisory input from a human teacher, several works in both robot and software agents have adapted it for human input by letting a human trainer control the reward signal. [Thomaz and Breazeal(2008)] experimentally examine the assumption underlying these works, namely that the human given reward is compatible with the traditional RL reward signal. This work demonstrates the importance of understanding the human-teacher/robot-learner partnership in order to design algorithms that support how people want to teach and simultaneously improve the robot's learning behavior.

[Griffith et al.(2013)Griffith, Subramanian, Scholz, Isbell, and Thomaz] argue for an alternate, more effective characterization of human feedback: Policy Shaping. They introduce Advise, a Bayesian approach that attempts to maximize the information gained from human feedback by utilizing it as direct policy labels. With these advancements this paper may help to make learning from human feedback an increasingly viable option for intelligent systems.

Chapter 3

Background & Concept

Reinforcement leaning is learning *what to do*, *how to map situation to action* so as to maximize a numerical reward signal. The learner is not told which actions to take but instead must discover which actions yield the most reward by trying them. In the most interesting and challenging cases, actions may affects not only immediate reward but also the next situation and all subsequent rewards. These two characteristics *trial and error* and *delayed reward* are the two most important distinguish features of reinforcement learning.

Reinforcement learning is different form *supervised learning* since it is not learning from examples provided by a knowledgable external supervisor, but it is able to learn from its own experience. The agent is located in an uncharted territory where it must be able to learn independently.

The methods presented in section 3.1.1 and 3.1.2 are today the most widely used reinforcement learning methods. This is probably due to their great simplicity: **a)** they can be applied on-line, with a minimal amount of computations, to experience generated from interactions with an environment **b)** they can be expressed nearly completely by single equations that can be implemented with small computer programs.

In the rest of the chapter we will introduce the algorithms selected for the *policy generation* (sec 3.1) and the proposed solution for the *policy combination* (sec 3.2). In section 3.3 we will see how to use the policy generated with the policy combination algorithm to speed up the learning of a new user.

3.1 Policy Generation

3.1.1 Q-learning

Developed by [Watkins and Dayan(1992)] it is an off-policy TD-control algorithm. Its simplest form, one-step Q-learning, is defined by

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma max_a Q(S_{t+1}, a) - Q(S_t, A_t)]. \qquad (3.1)$$

The learned action-value function, Q, directly approximates q^*, the optimal action-value function, independent of the policy being followed.

The goal of the agent is to maximize its total reward. It does this by learning which action is optimal for each state. The action that is optimal for each state is the action that has the highest long-term reward. This reward is a weighted sum of the expected values of the rewards of all future steps starting from the current state.

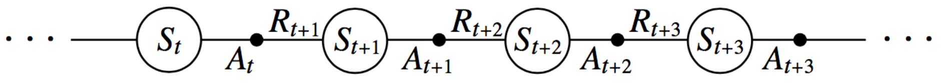
The γ factor is a number between 0 and 1 ($0 \leq \gamma \leq 1$) called *discount factor* and trades off the importance of sooner versus later rewards. A factor of 0 will make the agent considering only the current rewards, while factor near to 1 will make it take advantage of long-term high rewards.

The factor α is also included between 0 and 1 ($0 \leq \alpha \leq 1$). It represent the *learning rate* and determines to what extent the new acquired information will override the old information. A factor equal to 0 will make the agent not learning anything, while a factor of 1 will make the agent consider only the most recent information. In fully deterministic environment a learning rate of 1 is optimal, while in stochastic problem is required to decrease the learning rate to zero.

3.1.2 SARSA

It is a on-policy TD control method that learn action-value function rather than state value function. In an on-policy method we must estimate $q_\pi(s,a)$ for the current behavior policy π, for all states s and action a. This can be done using the TD method [Sutton and Barto(2011), chapter 6, p. 219] for learning v_π.

In Sarsa an episodes consist of alternating sequence of state and state-action pairs:



Sarsa considers transition from state-action pair to state-action pair and learn the value of state-action pairs. The update function:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)] \tag{3.2}$$

is performed after every transition from a non terminal state S_t. If S_{t+1} is terminal then $Q(S_{t+1}, A_{t+1})$ is defined as zero. This rules uses every element of the quintuple $(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$, that make up a transition from a state-action pair to the next.

3.1.3 LSPI

Introduced by [Lagoudakis and Parr(2003)], LSPI is an approach to reinforcement learning for control problems which combines value-function approximation with linear architectures and approximate policy iteration. It learns the state-action value function which allows for action selection without a model and for incremental policy improvement within a policy-iteration framework.

The state-action value function is approximated using a linear architecture:

$$\hat{Q}(s, a, w) = \sum_{i=1}^{k} \phi_i(s, a) w_i = \phi(s, a)^T w. \tag{3.3}$$

Though maximization of the approximate value over all actions in $A(s)$ the greedy policy π can be obtained at any given state s using the formula:

$$\pi(s) = \underset{a \in A(s)}{\arg\max}\, \hat{Q}(s,a) = \underset{a \in A(s)}{\arg\max}\, \phi(s,a)^T w. \tag{3.4}$$

For finite action spaces this is straightforward, but for very large or continuous action spaces, explicit maximization over all actions in $A(s)$ may be impractical.

Any policy π (represented by the basis functions ϕ and a set of parameters w) is fed to LSTDQ [Lagoudakis and Parr(2003), section 6] along with a set of samples for evaluation. LSTDQ performs the maximization above as needed to determine policy π for each s' of each sample (s, a, r, s') in the sample set. LSTDQ outputs the parameters w_π of the approximate value function of policy π and the iteration continues in the same manner.

LSPI would be best characterized as an off-line, off-policy learning algorithm since learning is separated from execution and samples can be collected arbitrarily. On-line and on-policy versions of LSPI are also possible with minor modifications.

Online LSPI

The crucial difference from the offline case is that policy improvements must be performed once every few samples, before an accurate evaluation of the current policy can be completed. In the extreme case, the policy is improved after every transition, and then applied to obtain a new transition sample. Then, another policy improvement takes place, and the cycle repeats. Such a variant of PI[1] is called fully optimistic. In general, online LSPI improves the policy once every several transitions taking the name of partially optimistic.

A second major difference between offline and online LSPI is that, while offline LSPI is supplied with a set of transition samples by the experimenter,

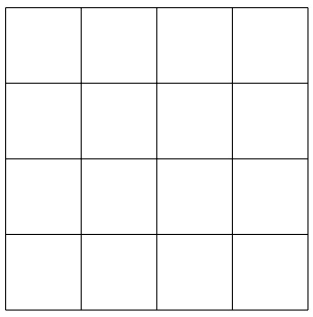
[1]Policy improvement.

online LSPI is responsible for collecting its own samples, by interacting with the controlled process.

Tile Coding

Tile coding [Sutton and Barto(2011), chapter 9, p. 225] is an approximation method that is particularly well suited for efficient on-line learning. In tile coding the receptive fields of the features are grouped into exhaustive partitions of the input space. Each such partition is called a tiling, and each element of the partition is called a tile. Each tile is the receptive field for one binary feature.

The computation of the indices of the present features is particularly easy if grid like tilings are used. If we address a task with two continuous state variables the simplest way to tile the space is with a uniform two-dimensional grid:



Given the x and y coordinates of a point in the space, it is computationally easy to determine the index of the tile it is in. The width and shape of the tiles should be chosen to match the width of generalization that one expects to be appropriate.

The number of tilings should be chosen to influence the density of tiles. The denser the tiling, the finer and more accurately the desired function can be approximated, but the greater the computational costs.

3.2 Policies Combination

In this section we will introduce the idea behind the policies combination algorithm, that is thought to work with multimodal actions and impaired users.

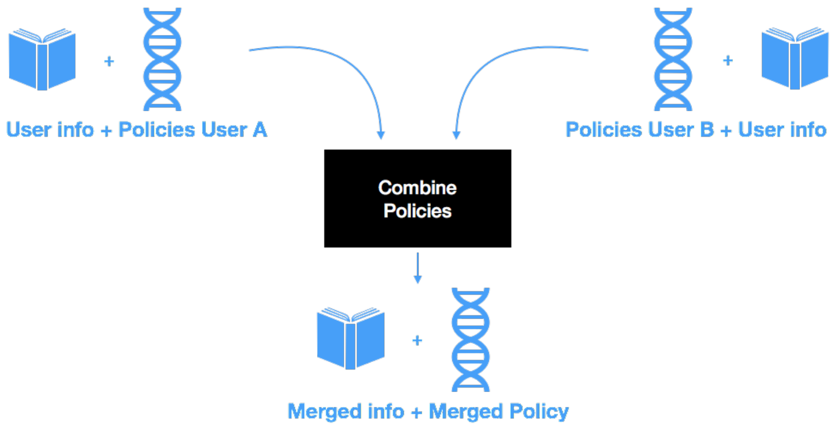

User info + Policies User A
Policies User B + User info
Combine Policies
Merged info + Merged Policy

Figure 3.1: Graphical representation of the policies combination schema.

The main idea represented in the figure 3.1 consists of creating a black box able to generate a *merged policy* from the policies of two different users. To make it possible the system needs the *user information*, an object that contains information about gaps and keys factor of the relative user.

The process can be applied to more than two users, because a third user can be combined with the policy obtained from the combination of the first two and the process can be iterated again on a new user. The final merged policy is expected to be suboptimal for all the users that has contribute for.

3.2.1 User Informations Extraction

In an environment with impaired users, where not all the parts of the composite actions can be perceived, it is possible that in two policies of the same user, the same state is associated to different actions and both are correct.

For examples in the situation where the agent is assisting a blind user to play the memory card games, in the state where the matching card is over the card that the user has a propensity for, the agent can perform two different actions that are proved to be both corrects. In the first case the agent can suggest with an head movement the left direction while the voice is saying “Maybe Up”. In the second case it can point the correct card with the gaze and say again “Maybe Up”.

Since the user can’t percept visual suggestions, the two action, for him, are identical. We can use this defect in our advantage to identify the user’s gap.

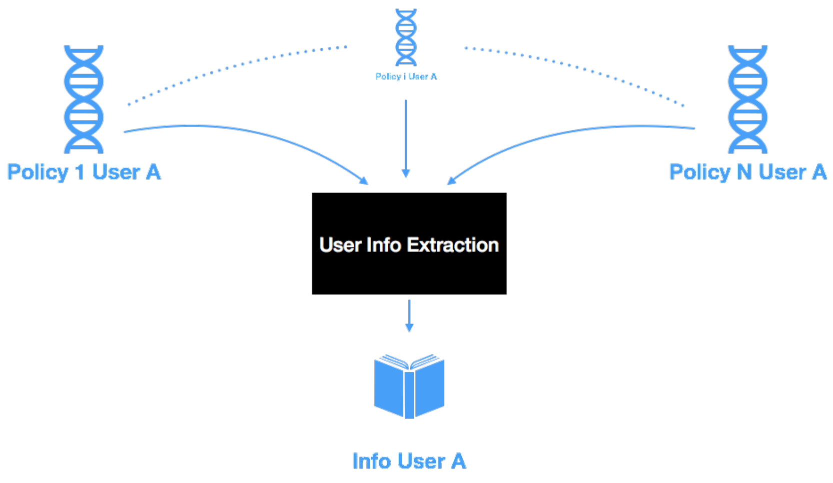


Figure 3.2: Graphical view of the user information extraction schema.

The *user info extraction* algorithm (fig 3.2) compares state by state the actions selected in the different policies and from that comparisons extracts the features of the user.

3.3 Speed Up the Learning

One of the biggest problems of reinforcement learning in online social scenarios is the time required for the learning. The rewards are received directly from the users and a long training phase with many wrong suggestions could cause frustration and lose of attention. Using Policy Reuse [Fernández and

Veloso(2006)], a technique for reinforcement learning guided by past policies, we can learn the current policies following a past policy and avoid the initial exploration that slow down the learning.

Policy Reuse algorithm balances among exploitation of the ongoing learned policy, exploration of random actions, and exploration of the past policies. The exploration versus exploitation tradeoff defines whether to explore unseen parts of the space or to exploit the knowledge already acquired during the learning process.

The main algorithm of Policy Reuse is the PRQ-Learning algorithm, a method to probabilistically reuse a set of past policies that solve different tasks within the same domain. This algorithm is composed by two main parts: **a)** π-reuse exploration strategy **b)** similarity function.

The π-reuse strategy is an exploration strategy able to bias a new learning process with a past policy. Its goal is to balance random exploration, exploitation of the past policy, and exploitation of the new policy, as represented in Equation 3.5.

$$a = \begin{cases} \Pi_{past}(s) & \text{w.p. } \psi \\ \epsilon - greedy(\Pi_{new}(s)) & \text{w.p. } 1 - \psi \end{cases} \tag{3.5}$$

The π-reuse strategy follows the past policy with probability ψ, and it exploits the new policy with probability of $1 - \psi$. As random exploration is always required, it exploits the new policy with an ϵ-greedy strategy.

The similarity function evaluate which are the policies more similar to the current behavior. These policies will be easily preferred in the future knowledge exploration.

Chapter 4

Scenario

The scenario has to be a representation of the environment in which we want to test the efficiency of the policies combination algorithm. For this reason it need to include a user, a social agent, and a simple task on which both user and agent will interact.

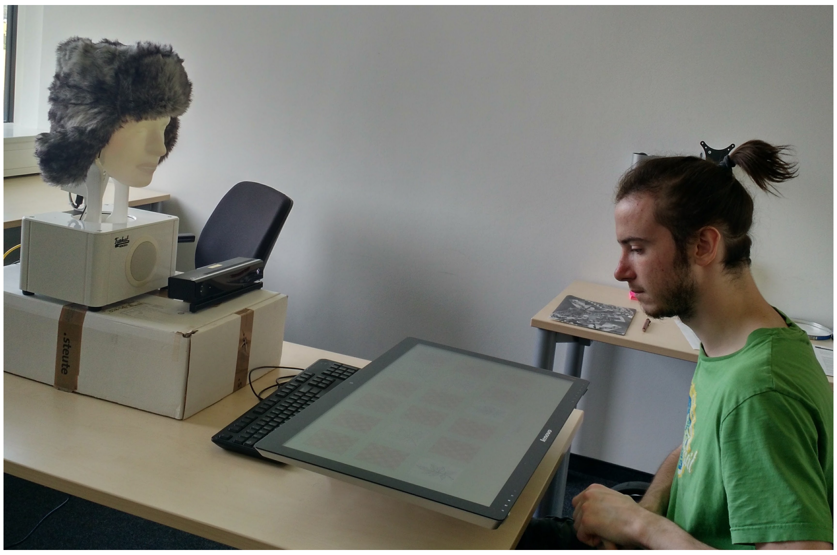


Figure 4.1: Picture of the scenario. It is composed by a user a touch screen with the matching memory game and a social robot.

The task selected for the environment is *Memory* [Wikipedia(2017)], a card game in which all the card are laid face down on a surface and two cards are flipped face up over each turn. The object of the game is to turn over pairs of matching cards. Any deck of playing cards may be used, and in our case it is composed of 9 pairs of similar card as is shown in the figure 4.2. The figure of the card are monochromatic image of leaf. For each kind

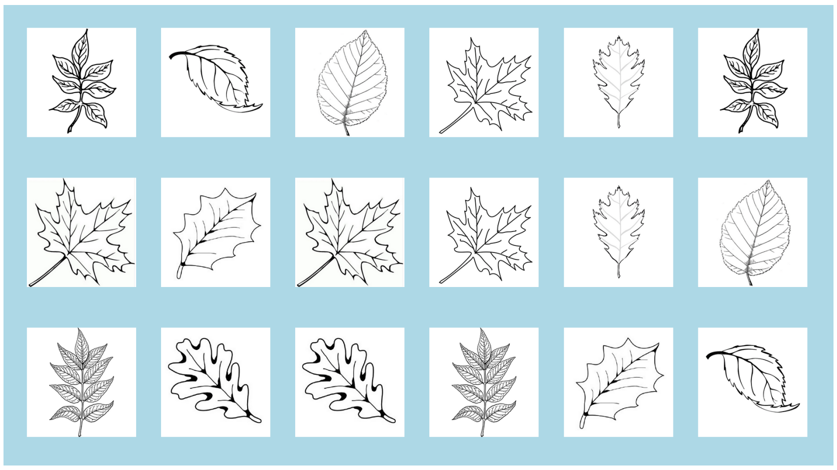


Figure 4.2: Memory deck.

of leaf there is at least a different pair with similar shape or number of sub-leaf. This similarity it is thought to reduce their distinctiveness and make the game harder for a human user. In this way the user, that is trying to complete the game, is more inclined to pay attention to the agent suggestions.

As physical agent, Furhat (Appendix C) fits perfectly the problem. Furhat is a back projected robot head capable of speech and head gestures. The agent is connected with the game in order to know at priori the position of the matching card. This information allows to Furhat to learn which is the action that suggest better the card position without the need of remember and explore the game board.

The last part of our test scenario is the user. The user interact with the game and agent to complete the task.

An import point is the definition of the dynamics of the environment. User and agent interact between each other to complete the task following the routine illustrated in the figure 4.3. The routine starts with the pick, by the user, of the first card. After this action, while the user continue looking for the right card to complete the pair, the game communicates at the agent the position of the matching card. When the user decides which is his second pick the agent detects his gaze and give him a suggestion. The suggestion is based on the position of the observed card and tries to confirm or move the attention of the user on the right card. In the end the user is free to follow or ignore the agent's suggestion before selecting the second card.

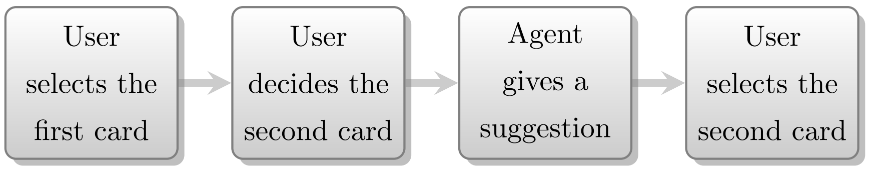

User selects the first card
User decides the second card
Agent gives a suggestion
User selects the second card

Figure 4.3: User agent interactions routine.

4.1 State Representation

The states are the shape in which we want to represent the problem. A first naive representation could be the position of the cards on the board. At first view it seems a good representation, but if we think about its usability this state representation become impossible to use. In a board of 9 pairs of cards the different disposition of the cards are $\frac{18!}{2}$. With the selected representation the number of states will be so huge that the learning will result impossible.

A new idea is to use a spatial representation that allows to store in a state the information about the spatial relation between two cards. For example a state can store the information about the relative position of the observed card and the matching card. In this way, knowing the current state, the agent has an idea about the position of the second card. For this representation 9 states have been designed.

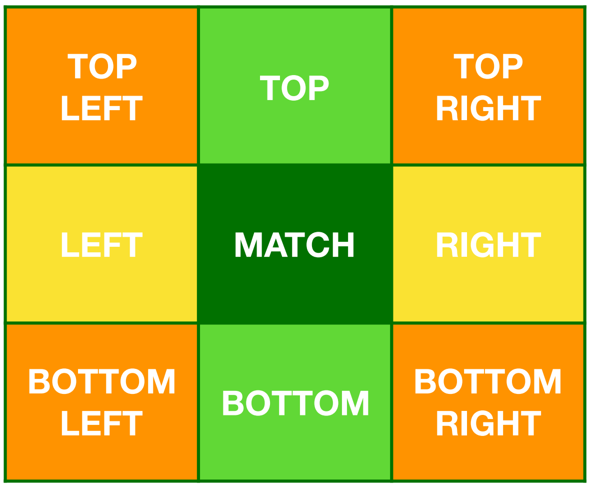
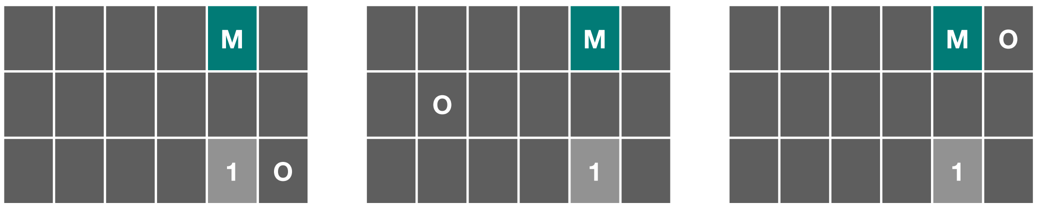


Figure 4.4: Graphical view of the spatial states representation.

The possible states are listed in the figure 4.4. Considering the interactions routine (fig 4.3) we can note that the user decides the second card after the identification of an *observed card*. The spatial state is always evaluated moving the "MATCH" cell of grid of the figure 4.4, and all the grid to follow, over the observed card. The cells of the grid border are extended in the direction of their label, to cover the entire game board. Checking the position of the matching card the current state can be founded reading the label of the grid's cell that is covering it. Three examples of possible states are reported in figure 4.5.

Using that representation and adding a state for all the situations where the number of card turned in the round are different from 1, we can use 10 states to represent every situation of the game.



(a) TOP LEFT (b) TOP RIGHT (c) LEFT

Figure 4.5: Spatial state examples.

This is an acceptable states space size, but it doesn't encode any information about the game progress. Indeed the game is more complicated at the beginning, when all the cards are covered, and becomes more easy in the final phases when the majority of the pairs are already discovered.

Analyzing better the three examples above we can notice that the number of possible cards in the recognized state are different. In the figure 4.5a there are 10 cards with the same relative position respect to the observed card. This, for the selected state representation, is the maximum number of positions on which the matching card can be associated to the observed card with the same resulting state. The figure 4.5c show the opposite case where the state can be associated to only one card, and so will be obvious, with the correct suggestion, to match the second card.

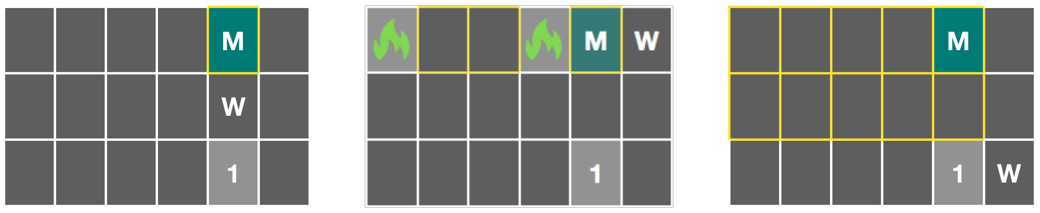
To store also this information we can add a number, to the name of the state, in order to save the number of possible cards in the state area. At the place of storing this number we have choose to convert the number of cards in a level. The level are the following:

Level 0 1 covered missing card

Level 1 2 or 3 covered missing cards

Level 2 more than 3 covered missing cards

The use of the level and not the explicit number avoid problems related to rare states, like for examples "TOP LEFT 10" that can happen only when the observed card is in the corner with the matching card top left and no turned cards in that area. It is easy to understand that these states are difficult to train. Another advantage of the level is that the states space is less than 30, while with the explicit number is around 100.



(a) TOP Level 0 (b) LEFT Level 1 (c) TOP LEFT Level 2

Figure 4.6: Spatial state with level examples.

For implementation choice of the board there are states where not all the levels are possible. That states are “TOP” and “BOTTOM” where the the ”Level 2” is not possible and “MATCH” where only the “Level 0” is possible.

Now that a clear states representation has been defined we can evaluate the real states size.

$$\begin{aligned} \text{size} &= 9 * 3 + 1 - 5 \\ &\qquad\qquad (2 * 2 + 1) \\ &= 23 \end{aligned} \tag{4.1}$$

4.2 Action Space

The use of Furhat as agent provides us different type of actions as speak, move the head, change expression and move the eyes.

In our scenario, these actions can be used to suggest the second card at the user. For example a voice suggestion can say a direction or even the right position of the card while with the head gestures the agent can confirm the idea of the user or move his attention to another area of the board.

For each type of actions we have select a subset of possible suggestions that can be useful in the game and fit the state representation. These subsets

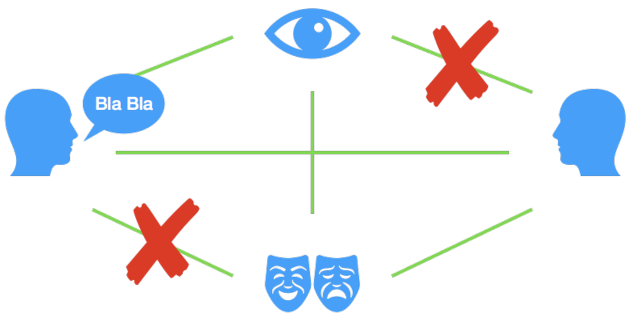


Figure 4.7: Graphical view of the possible combination between the different suggestions families.

are shown in the tables 4.1 4.2 4.3 4.4. In each of these tables is displayed the code that identifies the suggestion (Code), the action performed by the agent (Suggestion) and the interpretation that we give to that action (Meaning).

Code	Suggestion	Meaning
0	Say nothing	Nothing
1	"It is the right card"	The watched card is the matching card
2	"Try at row x and column y"	The matching card is at the row x and column y
3	"Maybe up"	The matching card is up
4	"Maybe down"	The matching card is down
5	"Maybe left"	The matching card is on the left
6	"Maybe right"	The matching card is on the right

Table 4.1: Action: Speech Suggestion

All these suggestions are part of different subsets that can be combined in composed actions. In figure 4.7 the four blue icons represent the different suggestion subsets and the green lines the possible combinations. Not all the combinations are significant and so appropriate. For example the connection between the gaze, on the top, and the head movement, on the right, is deleted because while the head is moving is not possible to detect which card the eyes are observing. In the same way, but for a physical reason, also the

Code	Suggestion	Meaning
0	Do nothing	Nothing
1	Smile	The observed card is the matching card

Table 4.2: Action: Face expression

Code	Suggestion	Meaning
0	Do nothing	Nothing
1	Nod	The observed card is the matching card
2	Turn to left	The matching card is on the left
3	Turn to right	The matching card is on the right
4	Turn up	The matching card is up
5	Turn down	The matching card is down
6	Turn to up-left	The matching card is up left
7	Turn to up-right	The matching card is up right
8	Turn to down-left	The matching card is down left
9	Turn to down-right	The matching card is down right

Table 4.3: Action: Head movement

connection between speech suggestion, on the left, and facial expression, on the bottom, is canceled. Indeed during the execution of the selected facial expression (smile) is not possible to speak.

Combining all the size of the suggestion subsets the action space will have

$$\begin{aligned} \text{size} &= 7 * 2 * 10 * 2 \\ &= 280 \end{aligned} \tag{4.2}$$

but with the use of constrains between the subset family, the final size is equal to 88.

Code	Suggestion	Meaning
0	Do nothing	Nothing
1	Gaze the card at row x and column y	Reveal the position of the matching card

Table 4.4: Action: Gaze

4.3 Reward Function

The reward function is one of the essential ingredients for the success of the learning, because from it, depends the fate of each action. As a matter of fact it is thanks to the reward received that the agent understands which actions are good and which are bad for the current state.

Since the goal is to complete the game in less actions possible we want that the user commits no error. To stimulate that behavior we are giving positive reward for each pair found and not a unique big reward for the completion of the game. The reward selected for the agent are the following:

+1 Pair match

-1 Pair miss

Chapter 5

Implementation

In the following chapter we will first see the choices and adaptations done during the implementation of the different RL algorithms. Since in the problem the agent is interacting with a user and the reward and future state are not know at priori, the implemented algorithms are only online. In the second part the code and the concepts behind the policies combination will be largely explained. The last section illustrates the adaptation of the policy reuse algorithm used to speed up the learning.

5.1 Online RL algorithms

The first two algorithms selected are Q-learning and SARSA. Originally they are offline, but thanks to their simplicity they are really easy to adapt in order to work on an online scenario. The third algorithm selected is the online version of LSPI.

5.1.1 Q-learning same as SARSA

During the implementation of the online version of SARSA we have noticed that an online version of SARSA was not applicable on the problem. This is due to the fact that, as is possible to see from the figure 4.3, after every action performed by the agent the resulting state is a state where the user has to perform an action. In this state the agent never perform any action so the value of

$$Q(s,a) = 0 \quad \forall a \in A(s). \tag{5.1}$$

If we substitute the hypotheses of the equation 5.1 at $Q(S_{t+1}, a)$ in one step of Q-learning (eq 3.1) and SARSA (eq 3.2) the resulting equations are the following:

$$\begin{align} Q(S_t, A_t) &= Q(S_t, A_t) + \alpha[R_{t+1} + \gamma max_a Q(S_{t+1}, a) - Q(S_t, A_t)] \tag{3.1}\\ Q(S_t, A_t) &= Q(S_t, A_t) + \alpha[R_{t+1} + 0 - Q(S_t, A_t)] \notag\\ Q(S_t, A_t) &= Q(S_t, A_t) + \alpha[R_{t+1} - Q(S_t, A_t)] \tag{5.2}\end{align}$$

$$\begin{align} Q(S_t, A_t) &= Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)] \tag{3.2}\\ Q(S_t, A_t) &= Q(S_t, A_t) + \alpha[R_{t+1} + 0 - Q(S_t, A_t)] \notag\\ Q(S_t, A_t) &= Q(S_t, A_t) + \alpha[R_{t+1} - Q(S_t, A_t)] \tag{5.2}\end{align}$$

Both the resulting equations are the same, so has no sense to implement both the algorithms. For this reason, in the test, only Q-learning has been implemented.

Online Q-learning

The Q-learning algorithm is shown in procedural form in algorithm 1. The procedure starts by setting all the values of the Q matrix to 0 and goes trough each step of each episode to discover which actions are better in each state.

An ϵ-greedy exploration is used to test new actions. The action selected in the line 5 is with a probabilities $1-\epsilon$ from the current policy or random in the other case. In the line after (line 6), the procedure performs the action and since it is online it waits until the user selects the second card to collect the reward r and the information about the new state s'. The value of ϵ

Algorithm 1: Online Q-learning

```
1  Initialize $Q(s,a) \forall s \in S, a \in A(s)$,arbitrarily, and
     $Q(terminal - state, \cdot) = 0$
2  for each episode do
3      $s \leftarrow$ initial state
4      for each step of episode to s is terminal do
5          $a \leftarrow \begin{cases} \arg\max_a Q(s,a) & \text{w.p. } 1-\epsilon \\ \text{a uniform random action} & \text{w.p. } \epsilon \end{cases}$
6          Take action $a$, observe $r, s'$
7          $Q(s,a) = Q(s,a) + \alpha[r - Q(s,a)]$
8          $s \leftarrow s'$
9      end
10 end
```

decrease during the simulation in order to promote an initial exploration and a late exploitation.

We can note, at line 7, the use of the one step equation derived in the previous section (eq 5.2), different from the standard version of Q-learning where the rule 3.1 is commonly used.

5.1.2 Online LSPI

Remembering that the crucial differences from the offline case is that policy improvements must be performed once every few samples and that online LSPI is responsible for collecting its own samples, we are going to illustrate the final procedure showed in algorithm 2.

The ϵ-greedy exploration is used, in line 6, to select, at every step k:

- a uniform random exploratory action with probability $\epsilon_k \in [0,1]$

- the greedy (maximizing) action with probability $1 - \epsilon_k$.

Typically, ϵ_k decreases over time, as k increases, so that the algorithm increasingly exploits the current policy, as this policy (expectedly) approaches the optimal one.

Algorithm 2: Online LSPI with ϵ-greedy exploration

```
   Input: $\phi, \gamma, K_\theta, \{\epsilon_k\}_{k\geq 0}, \delta$
   /* $\phi$ Basis functions                                          */
   /* $K_\theta$ Transition between consecutive policy improvement */
   /* $\{\epsilon_k\}_{k\geq 0}$ Exploration schedule                      */
1  $\ell \leftarrow 0$
2  Initialize policy $\pi_0$
3  $\Gamma_0 \leftarrow \delta I_{n\times n}$; $\Lambda_0 \leftarrow 0_{n\times n}$; $\mathbf{z}_n \leftarrow 0_n$
4  Measure initial state $s_0$
5  for each time step $k \geq 0$ do
6      $a_k \leftarrow \begin{cases} \pi_\ell(s_k) & \text{w.p. } 1-\epsilon_k \\ \text{a uniform random action} & \text{w.p. } \epsilon_k \end{cases}$
7      Apply action $a_k$; Measure next state $s_{k+1}$ and reward $r_{k+1}$
8      $\Gamma_{k+1} \leftarrow \Gamma_k + \phi(s_k, a_k)\phi^\mathsf{T}(s_k, a_k)$
9      $\Lambda_{k+1} \leftarrow \Lambda_k + \phi(s_k, a_k)\phi^\mathsf{T}(s_k, \pi_\ell(s_{k+1}))$
10     $\mathbf{z}_{k+1} \leftarrow \mathbf{z}_k + \phi(s_k, a_k) r_{k+1}$
11     if $k = (\ell+1)K_\theta$ then
12         solve $\frac{1}{k+1}\Gamma_{k+1}\theta_\ell = \gamma\frac{1}{k+1}\Lambda_{k+1}\theta_\ell + \frac{1}{k+1}\mathbf{z}_{k+1}$
13         $h_{\ell+1}(s) \leftarrow \arg\max_a \phi^\mathsf{T}(s,a)\theta_\ell \quad \forall x$
14         $\ell \leftarrow \ell + 1$
15     end
16 end
```

When $K_\theta = 0$ the policy is updated after every sample and online LSPI is fully optimistic. When $K_\theta \geq 0$ the algorithm is partially optimistic. The number K_θ should not be chosen too large, and a significant amount of exploration is recommended. For this reason ϵ_k should not approach 0 too fast.

Offline LSPI rebuilds Γ, Λ and $\mathbf{z}$ from scratch before every policy improvement. Online LSPI cannot do this, because the few samples that arrive before the next policy improvement are not sufficient to construct informative new estimates of Γ, Λ and $\mathbf{z}$. Instead, these estimates are continuously updated.

As basis functions, tile coding (sec 3.1.3) is used without the needs of extra implementation, because the code is available online thanks to [Sutton(2016)]. We have used a 4×4 Rectangular grid representation. This will create the possibility of generalization between any dimension of the states in which the components (spatial state and level) are within 0.25 of each other. Despite this fairly broad generalization, we would like the ability to make fine distinctions if required. For this we need many tilings, say 32[1]. This will give us an ultimate resolution of $0.25/32 = 0.0078$, or better than 1%. The length of the basic functions array will be the same as the number of tiles.

5.2 Policy Combination

In this section we are going to explain how the "Combine Policies" and "User Info Extraction" black boxes are implemented. We will start with the user information extraction because the data produced by this algorithm are used as input in the other.

5.2.1 User Informations Extraction

The *user info extraction* algorithm (alg 3) compares state by state the actions selected in the different policies. The evaluation between the actions

[1]A power of two is good here.

is made at suggestions level, so that for speech and head suggestions are counted the incompatibilities and for facial and gaze if they are different to “Do nothing” at least one time.

Algorithm 3: User Information Extraction

```
   Input: $P$
   /* $P$ Set of user policies                                  */
1  for each state $s$ in $P_1$ do
2      $A \leftarrow \{P_1(s), ..., P_n(s)\}$ ;  /* Set of current state actions */
3      if speech suggestion in $A$ are incompatibles then
4          $\text{info}_{speech} \leftarrow \text{info}_{speech} + 1$
5      end
6      if head movement in $A$ are incompatibles then
7          $\text{info}_{head} \leftarrow \text{info}_{head} + 1$
8      end
9      if $smile \in A$ then
10         $\text{info}_{facial} \leftarrow \text{True}$
11     end
12     if $gaze \in A$ then
13         $\text{info}_{gaze} \leftarrow \text{True}$
14     end
15 end
16 Convert the error in percentage
17 return info
```

For incompatibilities we intend suggestions that are different. An exception is done for the pairs formed by a random suggestion and the suggestion that points the correct card or simply do nothing. For example the speech suggestions “Maybe left” and “Try the card at row x and column y” are accepted. The different analysis of the facial and gaze differences is due to the fact that a user that is using these suggestions is for sure able to see.

The procedure return the user information as shown in the figure 5.1.

```
info = {
    "speech" : 100%,
    "head" : 0%,
    "facial" : True,
    "gaze" : True
}
```

(a) Deaf user

```
info = {
    "speech" : 0%,
    "head" : 0%,
    "facial" : True,
    "gaze" : True
}
```

(b) Normal user

```
info = {
    "speech" : 0%,
    "head" : 0%,
    "facial" : True,
    "gaze" : False
}
```

(c) Astigmatic or Myopic user

```
info = {
    "speech" : 0%,
    "head" : 0%,
    "facial" : False,
    "gaze" : False
}
```

(d) Partially-sighted user

```
info = {
    "speech" : 0%,
    "head" : 100%,
    "facial" : False,
    "gaze" : False
}
```

(e) Blind user

```
info = {
    "speech" : 0%,
    "head" : 100%,
    "facial" : False,
    "gaze" : True
}
```

(f) Blind user

Figure 5.1: Example of user info.

The case 5.1f show that the gaze could be present in the actions selected by a blind user even if the user can't perceive it. This could happen only for the gaze and not for the facial suggestions because the execution of the gaze force the head suggestion to "Do nothing" while the execution of the "Smile" force the speech suggestion to "Say nothing". A blind user ignores both head and gaze, but consider the speech that will hardly be null at the end of the training, excluding all the actions in which the "Smile" suggestion is included.

For a good estimation of the user info is better to compare more policies possible, where the policies has to be optimal to avoid incompatibilities due to an incomplete training.

5.2.2 Combination Algorithm

With the algorithm described above and the policy generated with the use of the reinforcement learning algorithms, we now have all the elements required for the *policies combination* algorithm.

This procedure simply takes a policy, in tabular representation, for both the users and combines the actions using an algebra defined to work at suggestion level, for the selected actions space. The resulting procedure is illustrated in algorithm 4.

Basically when the two users information differ in the head/speech error levels, the head/speech suggestions of the user with lower number of errors are taken. On the other side, when the head/speech error levels are equal, the actions of every state are generated summing the suggestions of the users.

The merged info associated to the resulting policy are evaluated as follows:

$$
\begin{aligned}
&\texttt{info} = \{\\
&\quad \texttt{``speech''} : min(\texttt{info1}_{speech}, \texttt{info2}_{speech}),\\
&\quad \texttt{``head''} : min(\texttt{info1}_{head}, \texttt{info2}_{head}),\\
&\quad \texttt{``facial''} : \texttt{info1}_{facial} \wedge \texttt{info2}_{facial},\\
&\quad \texttt{``gaze''} : \texttt{info1}_{gaze} \wedge \texttt{info2}_{gaze}\\
&\}
\end{aligned}
$$

This information, plus the merged policy, can be used in the combination with successive users.

Algorithm 4: Combine policies algorithm.

```
   input : $\pi_1, \pi_2, \text{info1}_1, \text{info2}$
1  for each state $s$ in $\pi_1$ do
2      if $\pi_1(s)$ != $\pi_2(s)$ then
3          if $info1_{speech} == info2_{speech}$ then
4              $\pi_{combined} \leftarrow$ The sum of the speech and facial suggestions
                 using the algebra of table 5.1
5          else if $info1_{speech} < info2_{speech}$ then
6              $\pi_{combined} \leftarrow$ speech and facial suggestions of the user$_1$
7          else
8              $\pi_{combined} \leftarrow$ speech and facial suggestions of the user$_2$
9          end
10         if $info1_{head} == info2_{head}$ then
11             $\pi_{combined} \leftarrow$ The sum of the gaze and head suggestions using
                 the algebra of table 5.2
12         else if $info1_{head} < info2_{head}$ then
13             $\pi_{combined} \leftarrow$ gaze and head suggestions of the user$_1$
14         else
15             $\pi_{combined} \leftarrow$ gaze and head suggestions of the user$_2$
16         end
17     end
18 end
```

Action algebra

The algebra divides the action in two parts: **a)** hearing suggestions and **b)** visual suggestions. The suggestions involved in the combination are summed using the rules listed in the tables 5.1 and 5.2.

This algebra gives the priority to the actions that are considered more specific. This behavior generates policies where suggestions like *Gaze/Say the correct card* are prevalent among the final suggestions. These solutions are correct but they hide alternative solutions where specific actions, for

Speech and Facial Expressions		
Suggestion 1	**Suggestion 2**	**Results**
Action i	Action i	Action i
Do nothing	Action i	Action i
“Say exact position”	Action i	“Say exact position”
Other Combination		Error

Table 5.1: Algebra of the hearing suggestions.

Gaze and Head Movement		
Suggestion 1	**Suggestion 2**	**Results**
Action i	Action i	Action i
Do nothing	Action i	Action i
Gaze exact position	Action i	Action i[1]
Gaze exact position	Action i	Gaze[2]
Other Combination		Error

Table 5.2: Algebra of the visual suggestions.

particular states, have the same expressiveness. Examples of these are all the directional suggestions in states with level 0. These actions, for that states, ensure a precision of the 100% but they can be replaced during the combination of the policies.

5.3 Policy Reuse Adaptation

In our version of the algorithm, we don’t have a set of policies but only a policy that we call π_{bias} generated from the combination of policies of different users. The bias policy is inserted in the balance equation at the place of the

[1]Only one user is able to perceive the Gaze.

[2]Both the users are able to perceive the Gaze.

set of past policy achieving a new revisited version of the exploration:

$$a = \begin{cases} \pi_{bias}(s) & \text{w.p. } \psi \\ \epsilon - greedy(\pi_{new}(s)) & \text{w.p. } 1-\psi \end{cases} \tag{5.3}$$

The adapted π-reuse strategy follows the bias policy with probability ψ, and it exploits the new policy with probability of $1-\psi$. As random exploration is always required, it exploits the new policy with an ϵ-greedy strategy.

The similarity function is not needed anymore because we are following a single past policy and not a set of policies. In the end of the training the π_{new}, trained on the current user, is returned to the system.

Algorithm 5: Π-reuse method

```
   Input: $\pi_{bias}, \phi, v$
   /* $\psi$ Probability to follows the past policy    */
   /* $v$ Decay of $\psi$                               */
1  Initialize $Q^{\pi_{new}}(s,a) = 0 \quad \forall s \in S, a \in A(s)$
2  for each episode $k$ do
3      $s \leftarrow$ initial state
4      $\psi_1 \leftarrow \psi$
5      for each step $h$ in the episode $k$ do
6          $a = \begin{cases} \pi_{bias}(s) & \text{w.p. } \psi \\ \epsilon - greedy(\pi_{new}(s)) & \text{w.p. } 1-\psi \end{cases}$
7          Apply $a$; Measure next state $s'$ and reward $r_{k,h}$
8          $Q^{\pi_{new}}(s,a) \leftarrow (1-\alpha)Q(s,a)^{\pi_{new}} + \alpha[r + \gamma argmax_{a'} Q^{\pi_{new}(s',a')}]$
9          $\psi_{h+1} \leftarrow \psi_h v$
10         $s \leftarrow s'$
11     end
12 end
```

Chapter 6

Virtual Simulation

The concepts introduced in the chapters before are first tested and validated on an *easier* scenario. The scenario in merit is a virtual reconstruction of the scenario defined in the chapter 4. The word “easier” doesn’t mean that the problem is simplified, but that the test modalities are simpler.

The advantage of a virtual scenario are countless, in the first phase of the development of a new technique. A first example is the possibilities of repeat the test how many times we want, thing impossible in presence of physical user.

For the virtualization of the scenario some adaptation is required and are explained in the section 6.1. In the other three sections the procedure of the simulation (sec 6.2) and the results (sec 6.3) are illustrated. We will conclude the chapter with a discussion (sec 6.4) of the obtained results.

6.1 Adaptation of Implementation

The virtual scenario has to simulate everything of the original problem, starting from the environment, passing to the user, and arriving to the action performed from the agent.

6.1.1 Virtual Environment

The solution adopted for the virtual environment is to virtualize the cards board using a textual representation. The game, now, can be played on the command line inserting the index of the card that the player wants to pick. A graphical view is proposed in the figure 6.1. In that, we can see how the cover cards are represented with a “X” and the image of the cards are replaced by number e.g. “2”.

X	X	X	X	7	X
X	2	X	X	X	X
X	7	X	2	X	X

Figure 6.1: Command line version of the Memory game.

The program that is running behind the virtual board, is responsible to check the consistence of the board after each pick. This is possible verifying the following rules before displaying the selected card:

- If the selected card is not visible and there aren’t more then two cards turned over in the current turn, the selected card will be turned to show its image;
- If there are two cards turned over, the card will be checked. If both cards are equal, they stay visible on the board, otherwise both the cards are turned over again. In both the case a positive or negative reward is communicated to the agent.

The memory program is also responsible for the communication of the matching card position to the agent.

6.1.2 Virtual Agent

The virtual agent is not so different from the real one. The real agent is composed by the reinforcement learning algorithm, that perform the learning task, and Furhat, the physical body of the agent, that execs the actions in the environment.

The virtual agent is free from the physical body and directly communicate the selected action to the user. The simulated user will interpret the action and will react to it.

The communication are done with the exchanges of JSON packages like the one in the figure 6.2

```
action = {"speech" : 4, "facial" : 0, "head" : 5, "gaze" : 0}
```

Figure 6.2: Example of virtual action.

6.1.3 Fictional User

To combine the policies we need policies generated from different users. For this purpose, using the techniques of the Personas [Schulz and Fuglerud(2012)], we have generated a set of virtual users that differ for the value of the following features: **a)** Focus **b)** Memory **c)** Hearing and **d)** Sight.

The fictional users, defined in that way, are composed by two modules. The first is a simple intelligence that manage the selection of the cards and the second has the role of understand the suggestions provided by the agent.

The personas generated for the simulations are listed in Appendix A.

User Behavior

The virtual user follows some really simple dynamics. His behavior can differ on the base of the card that he has to pick. When he has to select the first card, he checks whether he remembers the positions of two cards that make up a pair. If this happens the user will select one of the two cards, otherwise he will take a random card from the board excluding the cards that he already knows.

The selection of the second card is a bit more complicated because we need to simulate a reaction at the suggestions that is as realistic as possible. First of all the user checks for the presence of the matching card in his memory. If he doesn't remember it, he will select a card following the suggestions of the agent in the limit of its features.

The suggestions are interpreted by the user in order to reduce the number of possible cards. Depending on the features of the user the action can be completely or partially ignored.

User Skill Translation

For each persona the features represented in the radar chart, with value included in the range 1 to 6, are converted to value understandable from the system.

The simulated user considers the actions performed by the agent with a probability of p, where

$$p = (\text{focus} - 1) * 0.2. \tag{6.1}$$

User with low level of *focus* have a propensity to ignore the suggestions of the robot.

The *memory* feature is used to indicate the number of cards that the user can remember. These are equal to

$$m = \text{memory} + 1 \tag{6.2}$$

where the minimal possible result is 2 equivalent to a pair, the last seen.

The other two features are used to encode the impairment of the user. A user hears the speech suggestions with a probability of q, where

$$q = (\text{hearing} - 1) * 0.2. \tag{6.3}$$

Users with a value equal to 1 are considered deaf, while users with a *hearing* value of 4 have just the 60% of probabilities to hear the suggestions.

The mapping of the *sight* is a bit different. It is performed following the information in the table 6.1. Each level encode the suggestion family that the user is able to perceive.

Value	User classification	Perceived suggestions
6	Normal viewer	Head movement, Facial expression, Gaze
5	Normal viewer	Head movement, Facial expression, Gaze
4	Astigmatic	Head movement, Facial expression
3	Myopic	Head movement, Facial expression
2	Partially-sighted	Head movement
1	Blind	Nothing

Table 6.1: Sight mapping.

User Memory Management

The user can remember only a limited number of cards that, in the worst case, is equal to 2. The discovered cards are added in the memory only after the end of the turn. This choice yields to a situation where the user, during a turn, knows all the cards that are showed on the board plus the cards in his memory. When the turn is over, and a pair is not discover, the picked cards are added to the memory. The memory can't include pairs already discovered, because this will lock the card selection algorithm of the fictional user, so in the case that a pair already discovered is in the memory, it will be removed. When the memory is full and there is no space to store a new card, the oldest card in the memory will leave space for the new one.

Interpreter

This module is responsible for the interpretation of the actions performed by the agent. It takes the performed action and in the respect of the user features, return a set of cards where the user will randomly pick the next.

First we have to introduce the *no informations set* (NIS), a set composed of all the covered cards on the board less the cards remembered by the user.

The function starts checking p to understand if the action has to be considered or ignored by the virtual user. In the case that the action is ignored the next card will be chosen in the NIS and no reward is given to the agent in this turn. In the opposite case, when the fictional user doesn't remember the card and decides to follow the suggestion, the intersection of the sets of cards derived from each suggestion that compose the action is returned to the main module of the user.

How are the sets of cards of each suggestion family made? To start a NIS is associated to each of them. The interpreter function checks the action received from the agent in each of its parts. These contain codes that are associated to virtual actions (fig 6.3) and a relative interpretation that will reduce the sets size of each suggestion set. If the final set is empty, because of inconsistent suggestions in the same action, a NIS is returned and a punishment of -3 is given to the agent.

The interpreter also gives an additional reward, equal to $\frac{1}{\#(\text{final set})}$, to incentivize the use of specific suggestions and speed up the learning.

```
action = {"speech" : 4, "facial" : 0, "head" : 4, "gaze" : 0}
```

Figure 6.3: Example of inconsistent action.

In the figure 6.3 we can see an example of inconsistent action due to the incompatibility of the speech and head suggestions. The voice is saying "Maybe down" while the head is pointing up. The resulting set of possible cards will be empty.

The interpretation of each suggestion are listed and explained in the Appendix B.

6.2 Procedure

The simulation has been performed on all the generated fictional users. All of them are used to train a set of policies using both the learning algorithm[1]. For each algorithm the user has played 10 *games* composed of 500 *rounds*. Each game generates a policy that improve after every turn[2] of the rounds, where a round is composed by all the turns required to find all the pairs.

The training of each policy starts with the ϵ factor, responsible for the exploration, set to 0.9 that make the initial selection of the actions totally random ignoring the actions in the policy. Every 50 rounds, a tenth of game, the ϵ is decreased of 0.1 until it reaches the minimum value of 0.1. We are not interested in the initial value of γ, because the estimate of the optimal future value is always equal to 0. In Q-learning is also initialized the learning factor α. It start from a value of 0.5 and is decrease every 50 rounds of 0.05.

When the policies of each user has been generated and collected, the policies combination algorithm can be tested. The policies of each user are stored in separate folder. Knowing the path of the users policies folders, the algorithm automatically extract the users information from the comparison of the policies and randomly selects a policy from each user, that is addressed to generate the final merged policy.

Different merged policies has been generated for different subset of users. Their goodness has been tested via multiple executions of a round in which a user, of the users set in cause, is assisted from a rule based agent that follow the merged policy.

6.3 Results

We will now see the results of the different step of the simulation. Not all the results are reported, but only the most significant for the understanding of the problem.

The first important results is that all the simulations where we have

[1]Online Q-learning and online LSPI

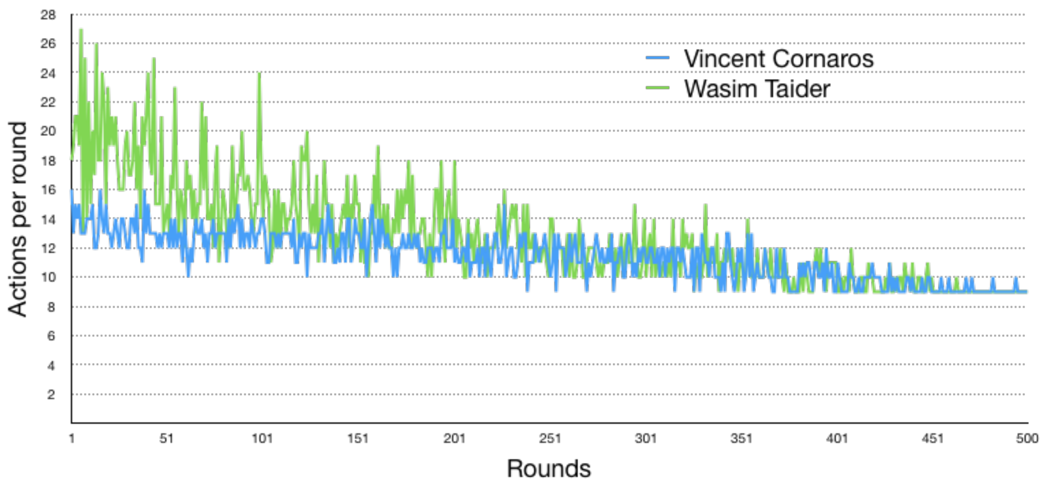
[2]In the case of online LSPI the policy improvement is performed every 5 turns.

generated the policies using online Q-learning have converged to an optimal solution in the selected number of rounds. This result has not been reached by the simulation that were running online LSPI.

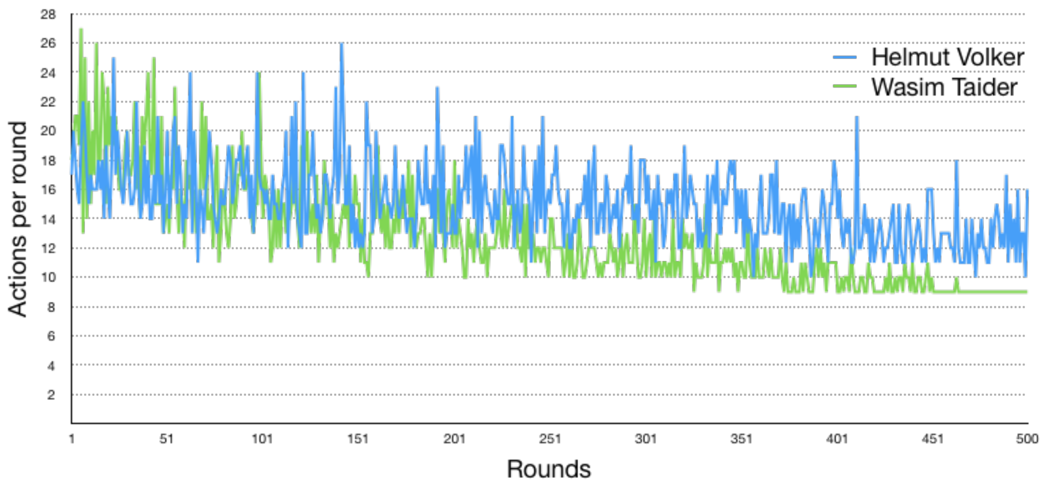
An important result is the distribution of the number of picks used by the user to complete the single round. This number is expected to decrease during the simulation, but sometimes we can have different evolution due to the effcts of the features. In the figure 6.4 are shown the learning progress of different users. The x axis reports the number of the round, while the y axis represents the number of picks per round. Each graph highlights the effect of a feature in the learning process.

With the policies obtained from the Q-learning algorithm is possible to extract the users information. The second results of the experiment are a comparison of the information extracted and the original features of the users and are reported in the figure 6.5. The figure is composed of two columns. On the left side we have the information extracted from the policies, while on the right are shown the graphical representations of the features.

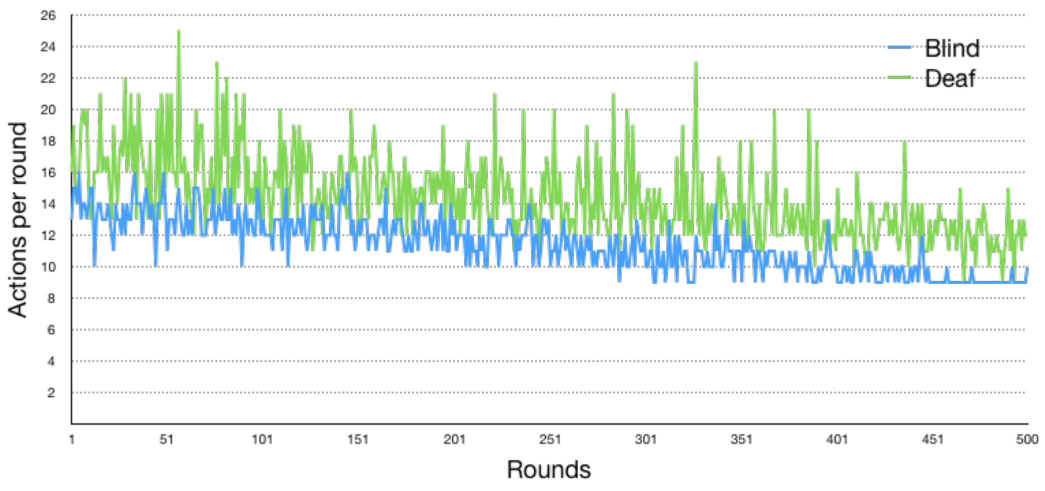
Extracted the user information, the next step is the combination of the policies. The figure 6.6 gives a visual representation of the users policies. Each graph contains three policies. The two, without label, are the policies used for the combination while the one with the blue cross is the merged policy. On the axis are represented the code that identifies the states and the code of the composite actions.



(a) Memory



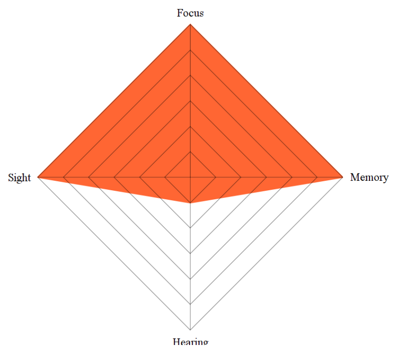
(b) Focus



(c) Sight vs Hearing

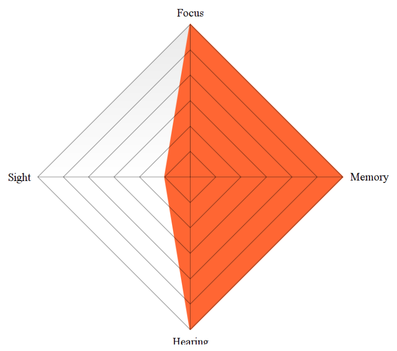
Figure 6.4: Learning process features analysis.

```
info = {
    "speech" : 100%,
    "head" : 9%,
    "facial" : True,
    "gaze" : True
}
```



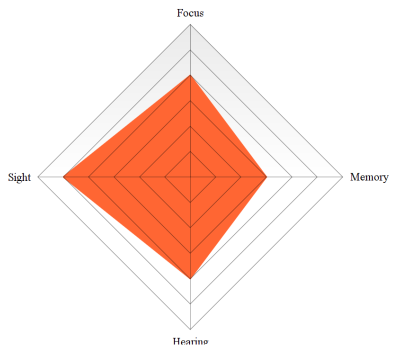
(a) Deaf

```
info = {
    "speech" : 4.5%,
    "head" : 100%,
    "facial" : True,
    "gaze" : True
}
```



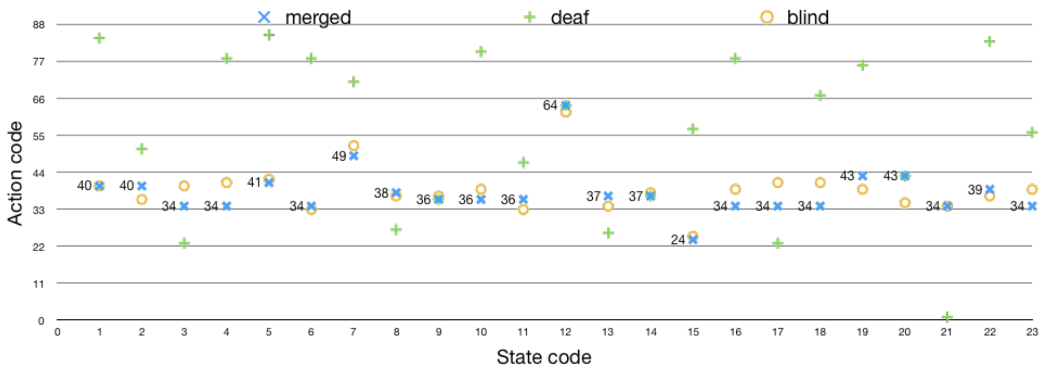
(b) Blind

```
info = {
    "speech" : 4.5%,
    "head" : 31%,
    "facial" : True,
    "gaze" : True
}
```

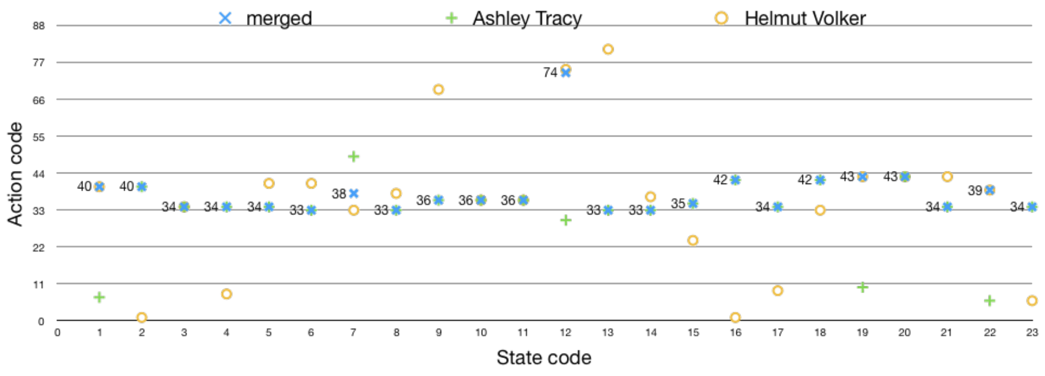


(c) Helmut Volker

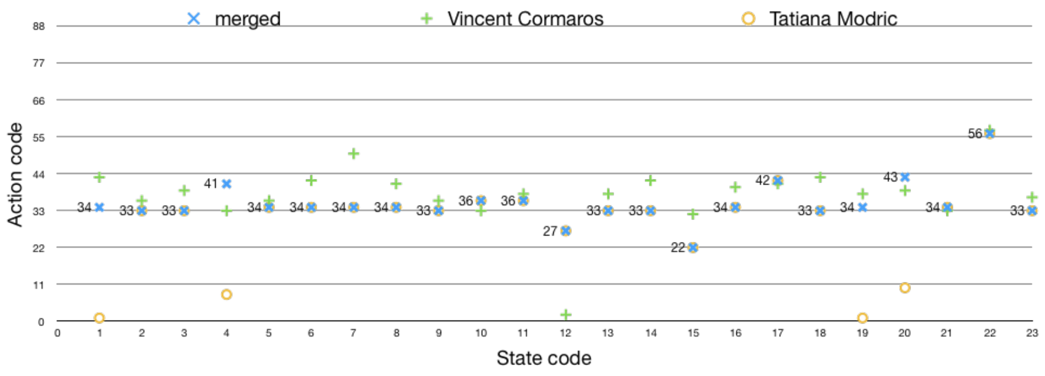
Figure 6.5: Features and user information comparisons.



(a) Blind & Deaf combination.



(b) Ashley Tracy & Helmut Volker combination.



(c) Vincent Cormaros & Tatiana Modric combination.

Figure 6.6: Graphical representation of the policies.

6.4 Discussion

Each user has found a solution in the 500 rounds, but if we consider the total number of picked cards the results are very different. It is really interesting understand how the features have influenced the performances of the users in the different phase of the learning. Looking the graph in figure 6.4a we can notice how the learning differs in the first half of the training. The cause of that difference is due to the memory feature. The user named as "Vincent Cornaros" has a better memory than the other user and he can reach good results also in the first phase, when the agent is performing an high number of random actions. Indeed a user with a large memory after the firsts mistakes, at the beginning of the round, has the memory full of card and start to match the pairs using the memory and ignoring the suggestion of the agent. This behavior is not possible with a low memory and it is for this reason that the two player start to have similar results only after half of the rounds, when the ϵ factor is lower than 0.5 and the majority of the suggested actions are selected from the policy. In the end, independently from the memory level, the two users achieve the same performances.

The second graph (fig 6.4b) compare two users with different level of focus. This is evident in the second part of the learning where "Helmut Volker", a old man that doesn't trust the technology, tend to ignore the agent suggestions. Even if at the beginning it performs better than "Wasim Taider" is learning is slower and the results barely improve. In the implementation of the interpreter (sec 6.1.3) we have said that when the suggestion is ignored no reward are given to the agent. Helmut has a focus value equal to 4, in oder words he follows only the 60% of the actions, that means that in 40% of the cases the agent doesn't receive any reward because we don't know if the action performed is good or not. Checking in the data log we have verified that the bad performance of the last rounds are due to wrong pick performed after the ignoring of a suggestion, proving that despite the slow down of the learning the agent was correctly trained and the non optimal performance are effect of random picks.

The last graph (fig 6.4c) compares two users with different impairment. We can see how the performance of the deaf are always worst than the blind user. Another detail that is visible in the graph is how more stable is the distribution of the blind user, with less variations and peaks. This two results are explainable by the fact that probably the speech suggestions are more precise that the visual suggestions. This is not completely true because half of the suggestions are equivalent, but if we consider the top suggestions of each suggestions family, that are "Say the exact position" and "Gaze the right card", they have a different precision. The top speech suggestion is more precise than every visual suggestion and guarantees a precision of 100% in each state. The impossibility for a blind user to use this suggestion make more complicated the matching of the pairs in the first picks, where the number of cover cards is higher.

All the previous results are about the policy learned with online Q-learning, because the situation with online LSPI is completely different. We don't have any results. The procedure was extremely slow and was not converging even after day of simulation and more rounds. The problem of the slow evaluation of the policy could be provoked by the size of the matrix that has to be solve in the linear system (line 12 algorithm 2). This matrix has size of $n \times n$ where n is equal to the number of basis functions multiplied for the number of actions. In our case that is equal to $32 * 88 = 2816$ a number too big to be evaluated online. In a second experiment the basis functions were reduced to 3 reaching the final size of $n = 264$. With this new size the time was considerably reduced but the matrix of the approximated Q function was not converging, probably because a wrong approximation. This wrong approximation could depend on the number of tiles or on the selected grid representation.

In figure 6.5 are reported the comparisons between the features of the users and the information extracted, for them, by the user information extraction algorithm. As expected the blind and deaf user has respectively 100% of error in the visual and hearing suggestions, but the unexpected datas are that they have not 0% in the other component. An error of 4.5% is equivalent to one incompatibilities, meaning that the blind and deaf users could have 1 and 2 wrong actions in their policies.

After a better inspection we have discover that every user contains an incompatibility in the state “Right Level 0”. From the simulation appears that this state is never visited by all the users. It looks like that the distribution of the pairs on the board and the style of play of the users, make this state impossible to visit. Similarly other states with incompatibilities, has been discover. They are the states with level 0, hardly visited and as consequence easily non corrects. For this reason higher is the number of the considered policies easier is to find one of this state untrained and as consequence an incompatibility. This is also the explanation to the fact that for the information extracted from the policies the blind user is able to perceive the gaze and the facial suggestions. For sure one of the untrained states contains a composed action that include the gaze, making the blind user able to perceive these suggestions.

The last topic that we have to discuss are the results of the policies combination algorithm. First we have to say that all the policies generated has given correct suggestion to the user involved in the generation. Even if sometime a equivalent action can be used, the results in terms of number of picks required to complete the round are always optimal or near the optimal solution.

A second results come out from the visual representation of the combined policies. We can see in each graph of figure 6.6 a visual representation of the users policies. From the graphical visualization is easy to note how the merged policies are always in the range of actions between the code 33 and 43. This range include the actions that are using the speech suggestion “Say the exact position” and the remaining parts of that action are the visual suggestions to the card. This results depend from the created algebra, that give advantage to the actions considered more powerful. Recovering the untrained states problem of the paragraph before the actions corresponding to the state 12 are always out of the range. This state is the state “Right Level 0” that is untrained in each policy and as consequence are random actions. The other actions that appear out of the range for the merged policy are associated to states with level 0 and are using different multimodal actions to suggest the right card.

In the future optic of the test in the real scenario, we get some important results to take in consideration. The first is that a good training is essential for the goodness of the final results. The second important result is the evidence of a similar output from all the policies combination test. This introduces the possibilities to use the merged policies as base policy in the training on future users. As consequence a merged policy generated on an users set that include enough diversity, could be use in off-policy algorithm to drive the learning avoiding problems in the initial exploration and obtaining an optimal policy for the current user.

Chapter 7

Human World

In the second part of the research we want to supply an additional proof at the results gained in the virtual simulation and test the usability of the policies combination algorithm in a process of off-policy learning in the optics of speed up the online learning.

We will first introduce, in section 7.1, the adaptation required to implement the scenario and to follow the explanation of the procedure (sec 7.2) used to collect the data. In conclusion we will list and discuss the results in the sections 7.3 and 7.4.

7.1 Adaptation of implementation

The figure 7.1 illustrates the elements required for the implementation of the task scenario. From the picture is possible to extract information about the four components and the interactions between the parts. Starting from the left we have the *real user* which unique task is to play memory on the *touch screen* and to do it, he can use the help provided by *Furhat*. The touch screen is used to display the cards to the user and communicates the selections to the central logic situated in the *server*. The central logic controls everything less the user. It communicates to Furhat which actions he has to perform, it runs the reinforcement learning algorithm and checks

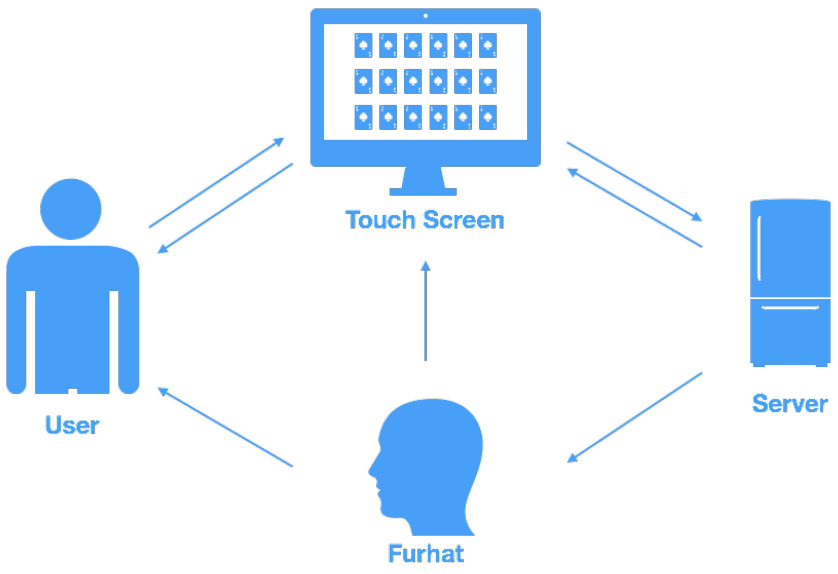


Figure 7.1: Elements of the real scenario.

that all the rules of the Memory game are respected.

7.1.1 User

The group of users that has participated to the test are 14 volunteers from the university with ages between 21 and 35 years old and different nationality. Only half of the group already had experience with robots before the simulation, while only few of them knew how reinforcement learning works.

It was not possible to find users with disabilities like blindness or absence of hearing, but we know that half of the participants need the glasses and only half of this half really uses glasses or other types of lenses.

7.1.2 Central Logic

The central logic is the brain of all the infrastructure. It contains the reinforcement learning algorithm and the game logic.

The learning part is responsible to select the actions, that Furhat will perform, and improve the policy. An important expedient, implemented for the learning, is a mechanism that identify when an actions is ignored from the user, based on the reaction time. Every time that the user select the two cards in a time interval smaller than 1,5 second the performed action is considered ignored and no reward is given to the agent.

The communications with Furhat may be implemented using *IrisTK* [Skantze and Al Moubayed(2012)], a java-based framework for developing multi-modal interactive systems, but this framework works only in a *windows* environment, and for this reason we were not able to use it. To supply to its absence we have manually implemented the communication. In the Appendix D are reported the detail of the implementation.

The game logic controls also all the dynamics of the game board. It uses a signal passing mechanism to receive notifications about which card has been selected and maintains a local representation of the board from which it can extract information about pairs match, pairs miss and number of picked cards in the current turn. All this information are used in the different phases of the learning algorithm.

7.1.3 Touch Screen

This device is used, in the scenario, to replace the physical cards. The game board (fig 7.2) is duplicated on the touch screen that allows a natural and immersive game play.

In the figure 7.2 a screenshot of the virtual board shows the disposition of the cards. In the picture are visible two details. The first, the highlighted card, is the one that the system recognized to be observed by the user but for a technical issue the *observed card recognition system* has not been implemented, so the first selected card of each turn is also considered as observed card. The second element is the button situated above the cards. It can be used by the user after a suggestion that is clearly wrong to give a punishment to the agent and to ask that a new action is performed. The chosen punishment is equal to -3 and is used to avoid that inconsistent actions were suggested more than one time.

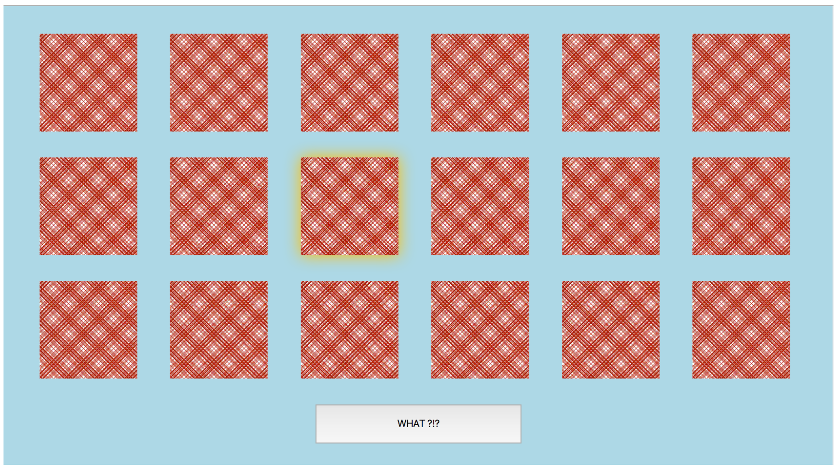


Figure 7.2: Picture of the game board.

7.1.4 Furhat

Furhat is interacting with both the user and the touch screen. Obviously he interacts with the user in order to give the selected suggestions, but it also interacts with the game board performing the gaze on the right card.

For a correct implementation of the interaction Furhat needs a camera that is used to evaluate the geometric triangulation of the point in the scene. Thanks to that Furhat is able to recognize the face of the user and the position of the cards.

7.2 Procedure

In the real scenario we have divided the simulation into two phases. The Phase 1 takes half of the users to train the policies required to generate the bias policy. Each user plays 2 games of 5 rounds and at the end of the simulation he has to complete a questionnaire relative to the experience. The number of games is the minimum required to generate the policies needed for the user information extraction process, while the reduced number of rounds is done to limit the game time in order to avoid stress in the users. Q-learning is used as learning algorithm. Like in the virtual simulation the ϵ-greedy method is used for the exploration. Each round the ϵ factor is decreased of $\frac{1}{5}$ of the initial value equal to 0.9.

At the user is requested to follow all the suggestions in a critic way, avoiding the selection of the right card after a suggestion considered wrong. This is very important to avoid bad trained policy that could affect the following steps.

In the Phase 2 all the users are used. They have to play 1 game of 5 rounds and, in the end, complete the questionnaire. The users of the Phase 1 have some additional questions used for the comparisons between the two phases.

The agent is running the policy reuse algorithm (sec 5.3) following the bias policy generated in the phase before. The exploration factor is set in order to initially favor the use of the past policy and increase the use of the current policy during the simulation. Indeed the last game is performed using only the new policy. The ϵ factor, used for the greedy exploration of the new policy, is set at an initial value of 0.9 and decrease uniformly among the rounds.

In this phase the users are free to play as they want in order to analyze natural behavior where for example a full level of attention by the user is not always guarantee.

The data are collected on five days. The first two days are dedicated to the Phase 1 on which 8 users has conducted the simulation in time slot of 1 hour. In the middle, between the two phases, one day has been used to combine the policies and to test the simulation of the next phase. The last two days are used to perform the simulation of the Phase 2. Since this simulation is shorter, slots of 30 minutes were used to collect in the same time the data related to the double of the users.

7.3 Results

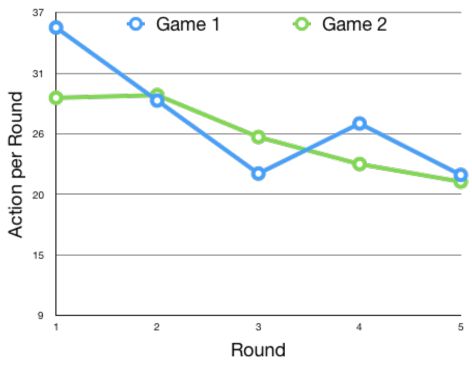
Starting from the results relative to the Phase 1, an important information is that only the 75% of the users has always followed the agent actions and the 87% of the users has never selects the corrects card when the action was completely or partially wrong. This happens because the users sometime has consider only a part of the multimodal actions. Indeed the 50% of the users has ignored at least one time the speech suggestions included in the robot actions and only the the 12% of the users has always followed the visual suggestions. Another data is that during the games only the 87% of the users has used the gaze in order to find a card, and the unexpected results is that the the 40% of the users has find an interpretation to the facial expression that they have used in the scenario.

In situation where speech and visual suggestion were contradictory the users has react in different way during the simulations. Most of them has considered only the speech suggestions or pressed the button for a new action, while the others were divided between select a random card or follow the visual suggestion.

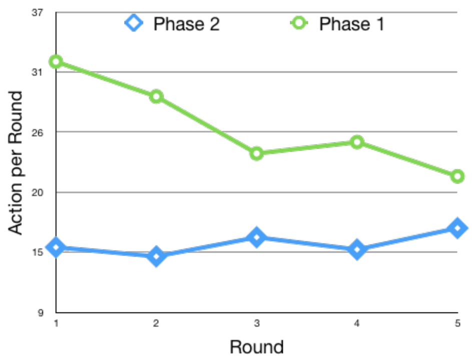
The first numeric results related to the learning that we want to report are the average performances in the different games. In all the future graphs the results represented with line and circle are associated to the Q-learning algorithm, while the results displayed with line and diamonds are associated to the policy reuse algorithm.

In the figure 7.3a are compared the average of the required actions of the first two games for all the users of the phase 1. Similarly in the figure 7.3b are compared the average actions per round between the two phases. The line labeled as “Phase 1” is the average of the two game reported in figure 7.3a.

Another important result that can be connected to the previous comparison is that, from the questionnaires, the 75% of the user that has take part to both the phases has recognize the round 1 of the phase 2 as the round in



(a)



(b)

Figure 7.3: Comparison of the average number of actions per round between the different game.

which the robot has perform better, while the other 25% has reply with the last round of Phase 2.

Always from the questionnaires we get the information that half of the users of the Phase 1 has changed the interpretation of some actions in the Phase 2. This change is motivated from more experience with the game and the view of new interpretations during the pause between the two phases.

Even if the results of the Phase 2 appears to be better we can find some user for which the policy learned at the end of the round 5 of the Phase 2 is worst that the others. This example is reported in the figure 7.4. The two lines with the circle represent the two games of the Phase 1, while the line with the diamonds is the representation of the performance of the Phase 2. The bars are used to show how many actions have been selected, in the Phase 2, from the indicated policy in the specific round.

Analyzing the behavior of all the users, that have taken part in the second phase, the most important data is that all the users has tried to learn how to interpret the robot actions after the first wrong suggestion. A positive results of the second phase is the fact that the 80% of the users has always

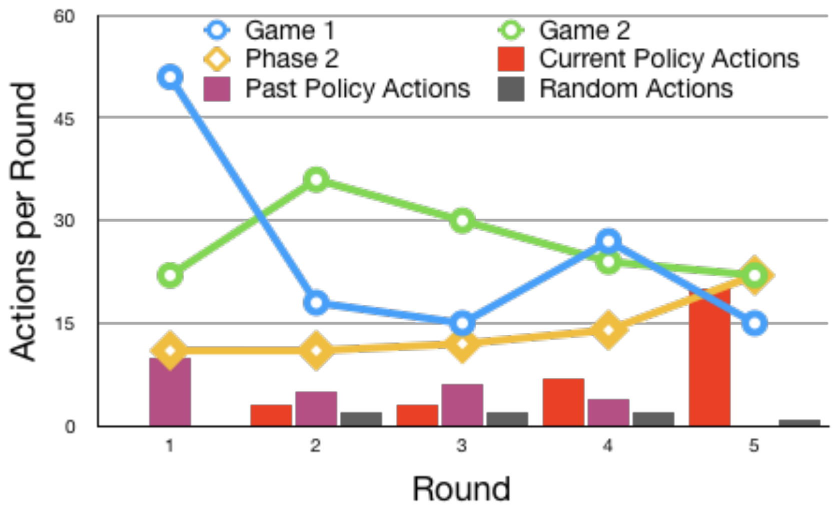


Figure 7.4: Example of bad training at the end of the Phase 2.

followed the agent suggestions ignoring it only when the position of the card was already known. An interesting data is that the 90% of these users has always payed attention to the speech suggestions while only the 40% has always payed attention to the visual suggestions and if we add the users that have partially payed attention they reach only the 75%. In this phase only 1 user has used the facial suggestions, but like a positive connotation for the other action parts.

The users were free to behave as they prefer and we have recorded that in case of contradictory suggestions more than half of their has followed the speech suggestions while the others were split between follow the visual suggestions, press the button or take a random cards.

As we have said at the beginning of the research we are interested to understand if a policy can be combined and work for the involved users, and later we have also hypothesized that this policy could work also for other users. The answers at this two questions are in the figures 7.5 and 7.6. The first shows the graphical representation of two merged policies. One is the policy gained from the combination of the Deaf and Blind users of the *vir-*

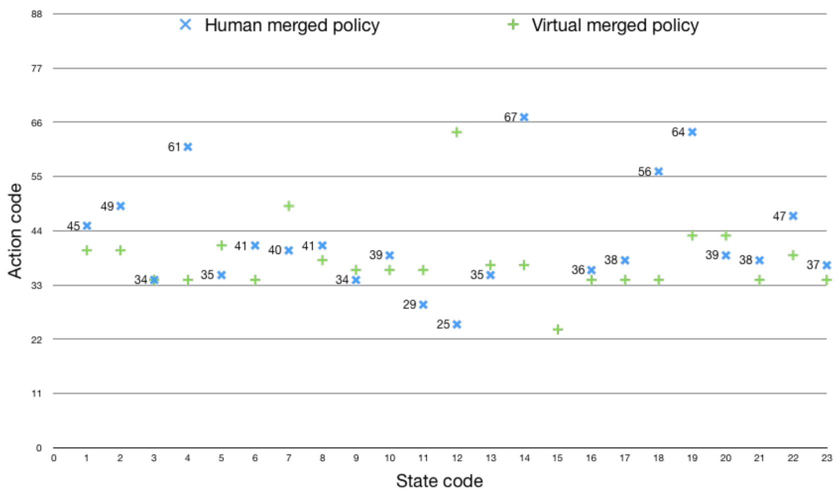


Figure 7.5: Comparison of the representations of the merged policies.

tual simulation, while the second is the policy obtained from the combination of all the users that have participated to the Phase 1 of the *human world simulation*. The second figure (fig 7.6) represents a comparison of the average performances gained with the use of the policy reuse algorithm between the users of the Phase 1, dashed line, and the new users added in the Phase 2.

An important analysis has to be done on how the users interpret the agent actions. The first unexpected event has been the mistake of some users in the localization of the position of the card when the line and column number were known. The user was counting the column starting from the right side and not the left, as it is normal in the Occidental culture. If this action can be misunderstood also all the others directional suggestions can be affected by the same problem.

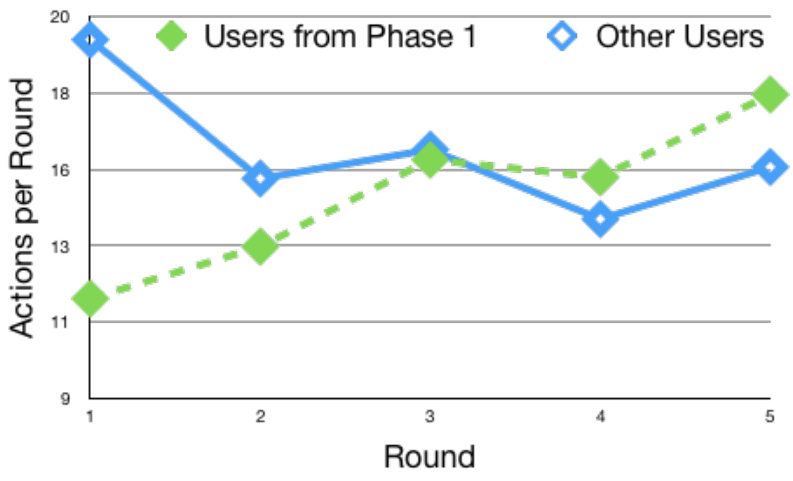


Figure 7.6: Graphical representation of the average performance in the Phase 2.

7.4 Discussion

The first results about the behaviors of the users in the Phase 1 are of essential importance, because the bias policy has been generated on their behaviors. The way on which these users have followed the agent actions condition the learned policy. It is not a big deal if, for example, a user ignore the visual suggestions, because the agent will learn how to reach the target using the speech suggestions, but it's essential that the user plays with consistency because the online simulation are not too long to correct an initial wrong learned behavior.

What emerges from the results of the questionnaires of the Phase 1 is that two users on eight have ignored parts of the agent's actions. One was ignoring the wrong actions while the second was not so focused on the robot and sometimes has played alone. The first behavior is not wrong on a normal game play, but could lead to unexpected results as associate wrong suggestions to the concept of correct action. In the same way the behavior of the second user is equivalent to give random feedback, making the policy full of noisy actions.

As we said before, in our system are privileged the consistent players, because they are going to have similar policies with few differences in the perceived parts and for this reasons the combining policies algorithm will give more importance to their learned suggestions.

About the remaining users, we can say that they perform a modest training. They have payed attention to don't keep the correct card after a wrong action, but a common error has been to select the right card when the exact position was explicitly said even if the visual part of the suggestion was pointing to the opposite side. This has leads to policies that are well performing for user able to hear, but confusing because the right suggestion is paired with a wrong movement that, in the future, will make the user feel insecure and with low confidence in the agent.

In the first two graphs (fig 7.3) we can see an evident improvement of performances for the user involved in the Phase 1, while the performances are basically invariant for the users of the Phase 2. The improvement of the users of the Phase 1 means that the agent is learning, but also that the users are learning how to interpret the agent actions. This facts is evident in figure 7.3a where we can see how better are performing the users in the first round of the game 2 compared to the game 1. Indeed after the first games the users were more confident and reactive at the agent actions, while at the beginning of the simulation they were more focused on all the environment and less on the difference between the suggestions.

The advantage of being expert of the environment is even more evident in the figure 7.6 where the user of the Phase 1 were performing, in the first round, with results close to the perfect match, while the new users, that are not familiar with the system, have needed the double of picks to complete the round. These users were able to immediately match the correct card after suggestions like "tell the exact position", but were disoriented and unable to interpret head movements and general voice suggestions. The presence of inexpert users in the Phase 2 partially explains the constant performance of the entire group of users. Indeed the initial average performance are the results of a great results of the expert users joint with the results of the rest of the users. In the end remains only to understand how can we explain the performances of the last round?

To understand better the results of the Phase 2 we need to examine better the figure 7.6 where the results of the two groups of users that compose the Phase 2 are compared. The two groups have opposite trends. The new users, introduced in this phase, are following a learning curve, improving almost every round. On the opposite side the users that had taken part also in the Phase 1 are deteriorating their performance every round.

The learning of the second phase is done using the policy reuse algorithm that balance the exploration selecting actions between the bias and current policy. Essentially the agent starts following the bias policy and moves the action selection to the new policy round after round until when, in the last round, the actions are selected only from the current policy. The expert users, that since the first round are performing really well, are unfortunately visiting a low number of states leaving parts of the current policy untrained. This leads to the situation where these users, in the last round, are assisted by partially trained policies that can't always provide the correct suggestions. It is important to clarify that this happens because we are using a really short train and it isn't a fault of the policy reuse algorithm.

A possible solution is to delay the use of the current policy in order to hope that the users visit all the states before the last round, but this limit the adaptation of the current policy to the specific needs of the current user. Depending on the situation the balance of the policy selection can leads to initial good performances followed by a partially untrained policy or to initial mediocre performances followed by a more trained policy.

An other interesting comparison is suggested again by the figure 7.3. Now the interesting data is that the average performance of the round 1 of the Phase 2 is better than round 5 of the Phase 1 of each users. We have supposed that the merged policy has to be sub-optimal for the users interested in the combination, but in the results it appears even better that the policies used in the combination. First we have to say that the supposition about the sub-optimality of the merged policy is valid only in the scenario where all the policy used in the combination are optimal, but this remain an interesting results.

How is possible that the policies combination has improved the learned policies? In a normal scenario with optimal policy the policies combination combine the resulting multimodal actions with the suggestions from the users recognized more precise in the perception of the corresponding channel of communication. In a scenario of fast training the combination algorithm prefers the suggestions supplied by the users more consistent in the actions interpretation. In the case of users with equivalent level of perception the actions are summed and the defined algebra favors the actions considered more specific. This could be considered as an introduction of supervision in the method that introduce a bias depending on the preference of the task designer, but in this case allow the specific suggestions to survive at the combination with random suggestions. In fact it is as consequence of this implementation choice that the merged policy is performing better than the other, because it privilege the learned suggestions at the innate random actions inside the non optimal learned policy.

After the new evidence related to the merged policy is interesting analyze his composition. In figure 7.5 is reported a graphical representation of two merged policies. If we consider the policy represented by green plus, generate combining the Blind and Deaf users of the virtual scenario, as correct, we can notice many selected actions of the human merged policy differing from it. These difference can be, in term of action code, small or bigger and both the cases require a detailed studies. In the case of a small differences, usually happens for actions that are composed by the same speech suggestion. All the actions that are saying the exact position of the card are included in the range of the codes 33-44 and the different code are given by the different combination of visual suggestions. Counting only the states where both the policies have the learned actions in this range, there are at least six states where the actions can't be both correct. These are the states where both the actions are included between 35 and 44. In fact these have different head movement, and so the visual part of the suggestion is moving the attention to different area of the board and for this reason one of the two actions is wrong. For the rest of the actions, these out of the range 33-44, we have personally tested their goodness and we have discover that for the human merged policy four are completely wrong.

To conclude we can say that the human merged policy is for the 45% composed by wrong or partially wrong actions. The explanation to all this wrong suggestions is more than one. As first thing the quality of the training, because the majority of the users have always paid more attention to the voice than to the visual suggestions. A second major explanation are the different interpretations that users has given to the suggestions. Indeed the same suggestions have assumed different meaning depending on the position of observed card on the board, the selected system of reference and in combination with other suggestions. For example suggestions pointing to the border with the observed card on the board have changed their meaning to all the card on the border. Another examples are the different interpretation assumed by the head pointing up between the different users. For someone was the card closer to him, but for others was the opposite. The same is happened for the head down, while a similar situation has been encountered for the speech suggestions "Maybe left" and "Maybe right".

All the same interpretation associated to different actions are not identified by the user information extraction algorithm that can't abstract to the concept thought by the user.

Even if the analysis of the merged policy has revealed some problem the results are really good and in a way unexpected. The credits of these performances are all for the human brain that is able to learn and interpret the different situations. Indeed in all the case of partially wrong actions the human user is able to understand from the context, which part of the action is wrong in order to match the right card.

Chapter 8

Conclusion

In a world where the robot are entering in the daily life, their use to assist the human require always new studies to improve their cognitive skills. We have tried to find a solution to the slow learning connected to the autonomous agents by proposing a solution that try to maintain the learned task related knowledge in the succession of different users. We have build an environment in which recreate the proposed problem and with the use of the reinforcement learning we have trained an agent in the task of assist different users. The information learned in this way has been combined in a new policy that the agent can use as base in the learning for the future users that will need assistance. This technique has been tested on virtual and human scenarios finding out that the idea is a valid solution, but that can be used in a limited set of environment. That are all the scenarios where multimodal actions are used and their interpretation are performed with no ambiguity between the different users.

This good results can be improved with some small future works. The first idea is to improve the user info extraction algorithm in order to work at action level and not comparing the single suggestions. This could allow the identification of multi-modal suggestion that are normally used by the human users. At the current state of the art these are recognized as incompatibilities reducing the quality of the features associated to the user in cause.

A second major works is the use of a preliminary test to understand the point of reference that the user has decided for the system. This information permit the agent to convert the learned actions in the system of reference used by the user and memorize the actions in a standard representation. In this way the propose process will lose the limitation about the ambiguity in the performed actions.

An additional improvement of the combination process can be done performing a combination of the policies of each user and use the obtained merged policy for the combination with the others. This idea can improve the learned policy in scenarios where the learned policies are not optimal. Until now all the combination were performed on a random policies between the policies learned assisting the user, but in the future the policy used for the combination can be the combination of the policies learned by the user. This idea come out after the results of the merged policy on the human's world simulation, where we have find out how the merged policy was outperforming the other learned policy.

This new approach try to solve a problem not so much considered until now. The hope with this research is to give inspiration for new possible solution or better implementation of the proposed solution.

List of Figures

List of Algorithms

List of Tables

Appendix A

Personas

In the following pages the stories of different personas are presented. From their routine, fears and virtues a set of features are extracted. The meaning of the features are explained in the following descriptions:

Focus The bent at follow instructions and the level of attention.

Memory User memory training level.

Sight Quality of the user's view.

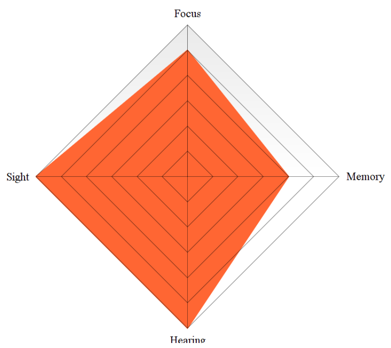
Hearing Quality of the user's hearing.

The features are represented on a 1 to 6 scale, where 1 is the minimum and 6 the maximum value.

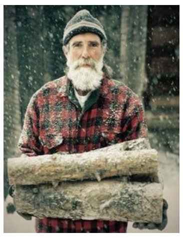
Ashley Tracy

Thirteen years old Ashley is the second of four brothers. She studies in the local school where she is also a cheerleader. She doesn't like to study and prefers to text with her friends and post pictures on the social network.

Ashley lives with her family and fights very often with her parents, because she thinks that they give more attentions and privileges to the two younger sons. For this reason she spends a loot of time close in her room using the smartphone or writing her diary.



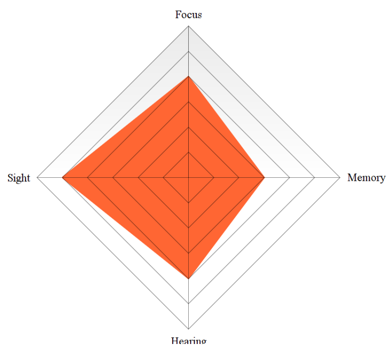
Helmut Volker



Sixty-two years old Helmut is a retired man that live in the suburbs with his wife. They have two sons that live by their own. The oldest is Ianna that works as secretary in a office in a near city. The second Arne is a car seller for local branch of the wolkswagen. The two sons come often to visit the parents and when is not possible they call Helmut on his old feature phone.

Every morning Helmut starts the day with a big breakfast while read the newspaper. After that he goes in the garden to check his vegetable garden and the other plants. In the afternoon he like to play chess with his wife, game that he has played during all his life in the free time.

He was a lumberjack for 35 year since he got an accident where he became cripple and has stopped to work.



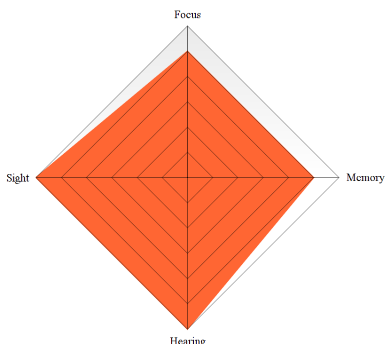
Tatiana Modric

Thirty-four years old Tatiana is a freelance with her own course of yoga, where she is the main instructor. She lives alone with her two dogs, nirvana and rog.

She works only 5 hour per day, 4 day at week. In the free time she likes to walk and run with the dogs, and practice meditation in the park or if the weather is bad in her house.

Tatiana is single since two years after she has broken with her ex boyfriend. She has some friend that sees every Sunday morning for breakfast. Usually on Friday she goes out with some guy that has meet during the week or with her sister. She really likes her sister and they are very connected. They are used to go to see ballet together, sport that they have played in their childhood.

Tatiana doesn't like to go out for shopping and love to buy everything online from the most famous website.

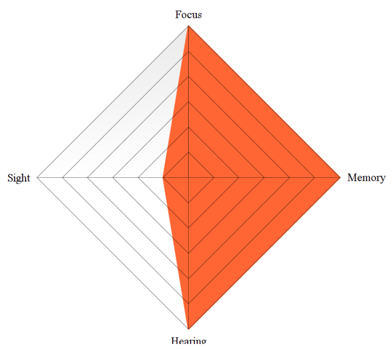


Vincent Cornaros

Twenty-three years old Vincent is a blind student at the university of physics. He lives with his parents and has no brother or sister.

Vincent has a guide dog which them can walk alone around the city. He also uses the iPhone in the day life with the help of the voice over technology.

He likes to listen instrumental music such as electronic and progressive. He is also heated for the role play games and plays it a lot with his group of friends. They meet every week end and one or two day in the week, based on the availability of everyone.



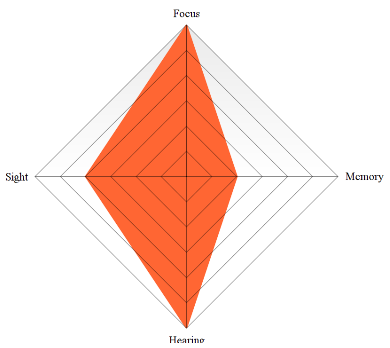
Wasim Taider

Forty-one years old Wasim is a Syrian refuge. He moved from Syria at the beginning of the fight using all the money saved since that day. Arrived in Germany he start a new life with his family composed of the wife and 2 daughters.

Now are 3 year that Wasim is in the new country and live in a old flat. He is the only worker of the familiar unit and the money are never enough.

He work as mechanical fitter all the week and in the week end work like delivery man in the kebab restaurant near home. He has very few free time and uses it to relax, served by his entire family.

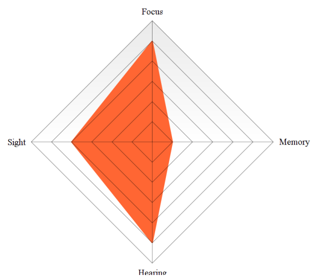
He has a cheap smartphone that use to maintain the contact with his old friends that now are around Europe. He finds it difficult to use because the characters are to small, but the real problem is that he needs glass, yet he never wears it in his entire life and is to proud to admit that he needs a pairs.



Frances Miller

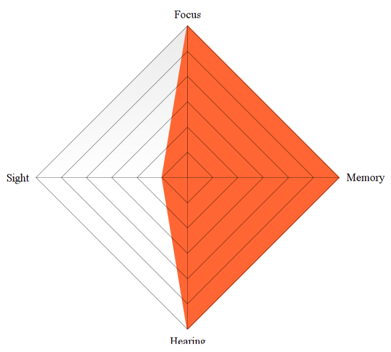
Sixty-seven year-old Frances is the mother of four children and the grandmother of twelve. She lives in her own home, bakes a pie once a week so that she has something to serve for Sunday visitors (usually one of her children and their immediate family), and has two cats. The cats' names are Fred and Wilma, names given to them by four-year old grandson Bobby. She likes to knit and do needlework, which she either gives away as presents to her family or donates to the annual sale to raise money for the church she belongs to.

She is a middle-class retiree living on a fixed income. Her mortgage has been paid off and she has one credit card which she seldom uses. She has been a customer of the bank for 57 years although has never used an automated teller machine (ATM) and never intends to. She has no patience for phone banking and does not own a computer. Every Monday at 10:30 am she will visit her local bank branch to withdraw enough cash for the week. She prefers to talk with Selma the branch manager or with Robert, a CSR who was a high-school friend of her oldest son.



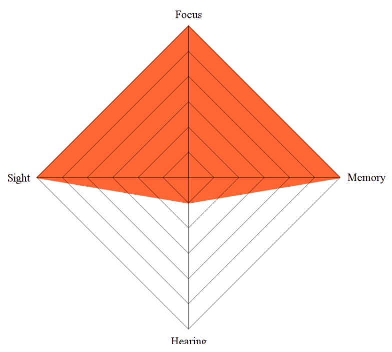
Blind

Blind user expressly designed for the simulation.



Deaf

Deaf user expressly designed for the simulation



Appendix B

Suggestion Interpretation

Action	**Suggested Cards**
Say Nothing	None[1]
“It is the right card”	The observed card
“Try at row x and column y”	The card at row x and column y
“Maybe up”	The cards above the observed card
“Maybe down”	The cards below the observed card
“Maybe left”	The cards on the left of the observed card
“Maybe right”	The cards on the right of the observed card

Table B.1: Interpretation: Speech Suggestion

Action	**Suggested Cards**
Do Nothing	None[1]
Smile	The observed card

Table B.2: Interpretation: Facial Expression

Action	**Suggested Cards**
Do Nothing	None[1]
Nod	The observed card
Turn the head up	The cards above the observed card
Turn the head down	The cards below the observed card
Turn the head left	The cards on the left of the observed card
Turn the head right	The cards on the right of the observed card
Turn the head up left	The cards above the left of the observed card
Turn the head up right	The cards above the right of the observed card
Turn the head down left	The cards below the left of the observed card
Turn the head down right	The cards below the right of the observed card

Table B.3: Interpretation: Head Movement

Action	**Suggested Cards**
Do Nothing	None[1]
Gaze the card at row x and column y	The card at row x and column y plus the cards UP, DOWN, LEFT and RIGHT[2]

Table B.4: Interpretation: Gaze

[1]No Information Set.

[2]If the card at row x and column y is on a border one or more cards can miss.

Appendix C

Furhat

"Furhat is a robotic platform that combines natural voice and gesture to create a more social, human experience" — Forbes

Furhat is a robot with a character and a personality, a platform to build socially aware spoken conversations with computers. It is an a human-like interface with social intelligence. An interface with personality and intention. An interface that understands the human social protocol, language, gesture, and dynamics. An interface that is built from the bottom up on computational models of engagement, group dynamics, joint-attention, and with social intentions.

Furhat comes with supports for several video and depth cameras including Kinect, it has a speech synthesis system and it is the most expressive robot head in the world. The high quality and great customization of Furhat's facial repertoire of facial gesture combined with the video device give the requirement required for a natural conversation. Thanks to the camera it is able to follow the face of the user giving an idea of involvement in the conversation. The facial gesture give the shade of skin and mouth movement like a real face. The two factors together plus the voice make Furhat a perfect platform to build social interactions.

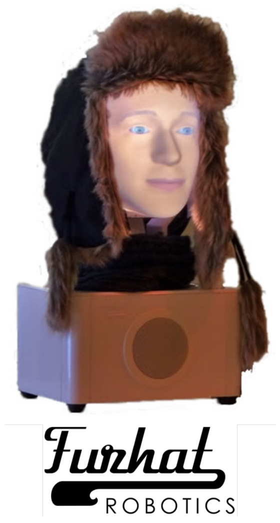


Figure C.1: A picture of the robot head Furhat.

Appendix D

Communicate with Furhat

The communication with Furhat can be implemented via the irisTk Framework, but it supports only windows environment. To deal with this problem we have used a serialized communication over TCP/IP. In this way is also possible to use programming language different from java.

First, we need to start a Broker that relays all messages. Thus, all systems only need to know the address of the Broker and inform the Broker of which events they subscribe to.

Then, it is possible to connect the system to the broker like this:

```
socket.connect((host, port))
```

After the connection of the socket we continue creating two threads, one for writing to the socket and one for reading from it.
When the threads are running we write

```
CONNECT [ticket] [myname]\n
```

on the socket, where `[ticket]` is the name of the ticket we want to share events with, and `[myname]` is the name of our client.
After the connect message we have to wait, reading the socket, until the receipt of the `"CONNECTED\n"` message.
Now that we are connected and identified to the broker we have to write

```
SUBSCRIBE [filter]\n
```

to tell the broker which events we are interested in (e.g., `"**"` for all events).

The messages are received in the form `EVENT [name] [nBytes]\n` followed by a serialized event in JSON format. The `[nBytes]` is a number which tells us how many bytes the JSON event takes. Thus, we have to continue to read these bytes and then parse the JSON.

To send events to the broker we have to write to the socket

```
EVENT [name] [nBytes]\n
```

followed by the serialized JSON event, where `[nBytes]` denotes the number of bytes in the JSON event.

To terminate your client, we have to write `"CLOSE\n"` and the system will shut down.

Event JSON Format

We are now going to introduce an example of event for each suggestion family, but first we will see a general examples.

```
{
  "class" : "iristk.system.Event",
  "event_name" : "action.speech",
  "event_id" : "my_unique_id_123",
  "text" : "Hello there"
}
```

Figure D.1: Example of serialized JSON event.

Every event has some standard parameters:

event_name the name of the event

event_id a unique ID for the event

event_sender the name of the module that sent the event

event_time a timestamp when the event was created

Particular events require additional parameter that are better explain in the irisTk documentation [Skantze(2013)].

Speech Event

This event represents an action that can be sent to a synthesizer to add an utterance to the speech queue.
The `"text"` parameter contains the text to speak.

```
{
    "abort": "true",
    "class": "iristk.system.Event",
    "event_id": "MemoryGame." + str(time.time()),
    "event_name": "action.speech",
    "event_sender": "MemoryGame",
    "event_time": str(datetime.now()),
    "text": msg
}
```

Figure D.2: Example of serialized speech event.

With the use of the event `action.speech.stop` we can say to a synthesizer to stop speaking (and clearing the speech queue).

Gaze Event

This event makes the agent shift gaze to a certain location in 3D space. Concatenations of multiple gaze events product a head movement.

For this event three additional parameters are used. The first is `"location"` used to store the 3D location where the agent should gaze. The parameter `"mode"` is used to specify the type or movement. This could be done by both the eyes and neck or by only one of the two. Using the default mode the robot use eyes and neck in a flexible way. The `"speed"` indicate how fast has to be the gaze to the 3D location.

```
{
    "class": "iristk.system.Event",
    "event_id": "MemoryGame." + str(time.time()),
    "event_name": "action.gaze",
    "event_sender": "MemoryGame",
    "event_time": str(datetime.now()),
    "location": {"x": "0", "y": "0", "z": "2"},
    "mode": "default",
    "speed": "medium"
}
```

Figure D.3: Example of serialized gaze event.

Gesture Event

The gesture event makes the agent perform a specific gesture.

```
{
    "class": "iristk.system.Event",
    "event_id": "MemoryGame." + str(time.time()),
    "event_name": "action.gesture",
    "event_sender": "MemoryGame",
    "event_time": str(datetime.now()),
    "name": "smile_open"
}
```

Figure D.4: Example of serialized gesture event.

Attend Event

This event is used to move the attendance of the agent to a specific target or location.

```
{
    "class": "iristk.system.Event",
    "event_name" : "action.attend",
    "event_sender": self.agent,
    "event_time": str(datetime.now()),
    "event_id": "MemoryGame." + str(time.time()),
    "target": targetId,
    "mode": "headpose",
    "agent": self.agent,
    "speed": "x-fast"
}
```

Figure D.5: Example of serialized attend event.

The target could be an user from the SystemAgentFlow or an item. The item can be added to the system in order to record when are detected or moved.

In the figure D.6 we show an example of how record a card. The `"sensor"` parameter is the ID of the camera or sensor that detected the items, while the `"items"` takes a record with all the items. The record is structured with item Id as key and item objects as values like in figure D.7.

```
{
    "class": "iristk.system.Event",
    "event_name" : "sense.item",
    "event_sender": "MemoryGame",
    "event_time": str(datetime.now()),
    "event_id": "MemoryGame." + str(time.time()),
    "sensor" : "kinect",
    "items" : record
}
```

Figure D.6: Example of item creation.

```
record = {
    "card1": {
        "class": "iristk.situated.Item",
        "id": id,
        "expire": -1,
        "location": {"x": 1, "y": 1, "z": 1},
        "shape": {
            "class": "iristk.situated.Shape$Box",
            "xsize": 0.07, "ysize": 0.01, "zsize": 0.07
        }
    }
}
```

Figure D.7: Example of item record.

Bibliography

[AB(2017)] Furhat Robotics AB. This is furhat, 2017. URL `https://www.furhatrobotics.com/furhat/`.

[Buşoniu et al.(2010)Buşoniu, Ernst, De Schutter, and Babuška] Lucian Buşoniu, Damien Ernst, Bart De Schutter, and Robert Babuška. Online least-squares policy iteration for reinforcement learning control. In *American Control Conference (ACC), 2010*, pages 486–491. IEEE, 2010.

[Fernández and Veloso(2006)] Fernando Fernández and Manuela Veloso. Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 720–727. ACM, 2006.

[Griffith et al.(2013)Griffith, Subramanian, Scholz, Isbell, and Thomaz] Shane Griffith, Kaushik Subramanian, Jonathan Scholz, Charles Isbell, and Andrea L Thomaz. Policy shaping: Integrating human feedback with reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2625–2633, 2013.

[Hemminghaus and Kopp(2017)] Jacqueline Hemminghaus and Stefan Kopp. Towards adaptive social behavior generation for assistive robots using reinforcement learning. In *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, pages 332–340. ACM, 2017.

[Konidaris and Barto(2006)] George Konidaris and Andrew Barto. Autonomous shaping: Knowledge transfer in reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 489–496. ACM, 2006.

[Lagoudakis and Parr(2003)] Michail G Lagoudakis and Ronald Parr. Least-squares policy iteration. *Journal of machine learning research*, 4(Dec): 1107–1149, 2003.

[Rosenblatt(2000)] Julio K Rosenblatt. Optimal selection of uncertain actions by maximizing expected utility. *Autonomous Robots*, 9(1):17–25, 2000.

[Russell and Zimdars(2003)] Stuart J Russell and Andrew Zimdars. Q-decomposition for reinforcement learning agents. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 656–663, 2003.

[Schulz and Fuglerud(2012)] Trenton Schulz and Kristin Skeide Fuglerud. Creating personas with disabilities. In *International Conference on Computers for Handicapped Persons*, pages 145–152. Springer, 2012.

[Sherstov and Stone(2005)] Alexander A Sherstov and Peter Stone. Improving action selection in mdp's via knowledge transfer. In *AAAI*, volume 5, pages 1024–1029, 2005.

[Skantze(2013)] Gabriel Skantze. Iristk intelligent real-time interactive systems toolkit, 2013. URL `https://http://iristk.net/`.

[Skantze and Al Moubayed(2012)] Gabriel Skantze and Samer Al Moubayed. Iristk: a statechart-based toolkit for multi-party face-to-face interaction. In *Proceedings of the 14th ACM international conference on Multimodal interaction*, pages 69–76. ACM, 2012.

[Sutton(2016)] Richard S. Sutton. Tile coding software – reference manual, version 2.1, 2016. URL http://incompleteideas.net/rlai.cs.ualberta.ca/RLAI/RLtoolkit/tiles.html.

[Sutton and Barto(2011)] Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction. 2011.

[Taylor and Stone(2009)] Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(Jul):1633–1685, 2009.

[Thomaz and Breazeal(2008)] Andrea L Thomaz and Cynthia Breazeal. Teachable robots: Understanding human teaching behavior to build more effective robot learners. *Artificial Intelligence*, 172(6-7):716–737, 2008.

[Watkins and Dayan(1992)] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[Wikipedia(2017)] Wikipedia. Concentration, 2017. URL https://en.wikipedia.org/wiki/Concentration_(game).

Statement of authorship

I hereby certify that I wrote this thesis independently and that it has not been submitted for any other degree. I have not used other than the stated sources and aids. All direct or indirect sources are acknowledged as references.

Bologna, December 7, 2017 [illegible]