Bayesian Inverse Reinforcement Learning for Modeling Conversational Agents in a Virtual Environment

Lina M. Rojas-Barahona[1] and Christophe Cerisara[2]

[1] Université de Lorraine/LORIA, Nancy
[2] CNRS/LORIA, Nancy
{lina.rojas,christophe.cerisara}@loria.fr

Abstract. This work proposes a Bayesian approach to learn the behavior of human characters that give advice and help users to complete tasks in a situated environment. We apply Bayesian Inverse Reinforcement Learning (BIRL) to infer this behavior in the context of a serious game, given evidence in the form of stored dialogues provided by experts who play the role of several conversational agents in the game. We show that the proposed approach converges relatively quickly and that it outperforms two baseline systems, including a dialogue manager trained to provide "locally" optimal decisions.

1 Introduction

Reinforcement Learning (RL) has been widely used for learning dialogue strategies [1–5]. Dialogues are modeled as an optimization problem, simulating the inherent dynamic behavior of conversations in order to find the globally optimal policy. However, the RL problem assumes the reward function is known. Indeed the reward function is usually handcrafted, as pointed out in [6], "the reward function is almost always set by intuition, not data". Inverse reinforcement learning (IRL) has been defined in [7] as the problem of recovering the reward function from experts' demonstrations. It tries to find an optimal reward, which leads to a decision policy that follows as closely as possible the examples provided by experts maximizing the expected cumulated reward in the long-run.

In this work we explore Bayesian Inverse Reinforcement Learning (BIRL) [8] to infer the reward function from humans who perform the task of instructing players in a serious game. We also apply the improvements to BIRL proposed in [9], namely the Modified BIRL (MBIRL), in order to reduce the computational complexity in large state spaces. This work covers a first step towards dialogue optimization with user simulation. Therefore, instead of designing in advance the reward function to "properly instruct players", which is a difficult and subjective task, we rather propose to learn it from humans. Once we have found the reward function we can apply classical reinforcement learning with user simulation for building a dialogue system and afterwards testing it with real users.

The adapted Bayesian approach is evaluated in terms of policy loss [9] and is compared against two baselines. The first one uses random rewards, while the second one exploits corpus-estimated locally-optimal rewards (i.e., supervised learning). The results show that the proposed approach converges relatively quickly and consistently outperforms both baselines, which confirms that taking into account the dynamic properties of the environment leads to virtual characters that better reproduce the behavior of experts. Qualitatively, our models have thus learned to adequately inform users and provide help when needed.

2 Reinforcement Learning for Dialogue Management

We focus on Markov decision processes for modeling dialogues because we aim to model unimodal conversations (i.e., a chatbot), thus we do not tackle speech recognition uncertainty. We first introduce Markov decision processes, then we present some reward functions commonly used in dialogue systems.

2.1 Markov Decision Processes

A finite Markov decision process (MDP) is a tuple $M = (S, A, T, \gamma, R)$ where:

- S: A set of possible states that represent the dynamic environment.
- A: A set of possible actions.
- $T : S \times A \times S \to [0, 1]$ is a transition probability function. For any action $a \in A(s)$ taken in a state $s \in S$, the probability of transiting to the next state s' is given by $T(s, s')$.
- γ: A discounting factor in the range of [0, 1), which controls the prediction horizon of the algorithm.
- R: The reward function that specifies the reward gained at every state. It contains the information that guides the agent towards the goal. R is a function of the state that is bounded in absolute value by R_{max}.

A stationary *policy* is a map $\pi : S \to A$ and the discounted infinite-horizon expected reward for starting in state s and following policy π thereafter is given by the value function $V^{\pi}(s)$ that satisfies the following *Bellman Equation*:

$$V^{\pi}(s) = R(s) + \gamma \sum_{s'} T(s, \pi(s), s') V^{\pi}(s') \tag{1}$$

The discounted infinite-horizon expected reward for starting in state s, taking action a and following policy π thereafter is given by the *Q-function* $Q^{\pi}(s, a)$ that satisfies the following equation:

$$Q^{\pi}(s, a) = R(s) + \gamma \sum_{s'} T(s, a, s') V^{\pi}(s') \tag{2}$$

A policy π is optimal in M iff, for all $s \in S$:

$$\pi(s) = \arg\max_{a \in A} Q^{\pi}(s, a) \tag{3}$$

$Q^*(s, a, \mathbf{R})$ is the optimal Q-function of the optimal policy π^* for a known reward function **R**.

2.2 Reward Functions for Dialogue Systems

Previous work on RL for learning dialogue strategies typically use reward functions that penalize long dialogues, returning a final positive reward for task completion or user satisfaction [1, 2, 10, 11]. This might be an intuitive reward function for slot-filling applications, such as train ticket or restaurant reservation, in which usually customers know exactly what they want and they expect to be accurately informed by the system as fast as possible. However, this reward function might be inappropriate in other domains or even for some other user profiles in the same domain. This is especially true in tutorial dialogues, where learners usually have to complete a task and may not know exactly how to do it. In such tutoring situations, the reward function might be designed according to the student-learning gains [12]. However, it is usually difficult even for tutors to write down the correct formula for being a good tutor. In our case, we are interested in building conversational virtual humans for a serious game. Although virtual characters can been seen as tutors because they provide information and help players to successfully complete different tasks, the learning-gain is relaxed since not only some conversations are optional, but also besides asking for help, players may also talk with virtual characters just for fun.

3 Related Work

Two dialogue systems were built in [13] for the same game scenario presented in this paper: (i) combining an information-state dialogue manager and a supervised model for interpretation; and (ii) using a supervised model for dialogue management. Both systems were evaluated with real users reaching a relatively low user satisfaction. These results motivated our interest to explore IRL because it leverages the local learning of supervised models in the context of MDPs, optimizing the cumulated reward at long term. IRL has been first introduced by [7], then it has been applied in car driving simulation [14] and autonomous helicopter aerobatics [15]. In dialogue systems, IRL has been first proposed as one strand of dialog research by [6]. It has been applied to user simulation in [16], learning the behaviour of users for simulating iterations in a RL dialogue manager with a known reward. Instead, we are applying IRL for dialogue management, learning the tutor (i.e. system) reward function from experts. In addition, their user simulator learns from human-computer data, which contains iterative turns between humans and a rule-based dialogue manager, while we are using human-human data, thus avoiding possible incoherent or unusual turns due to system errors. Unlike previous work [17, 16], we are using a Bayesian refined IRL algorithm instead of the original IRL algorithm proposed in [14] (the reader is referred to [18] for a review of IRL algoritms). Finally, [17, 16] applied IRL for slot-filling dialogue systems while we are applying it for building twelve distinct conversational agents in a serious game.

4 Conversational Agents in a Serious Game

In this section, we introduce the serious game and the dialog scenario. We then describe the dialogue states, the actions as well as the transition probability function.

Table 1. Description of the 12 dialogs in the game

Dialog Id	VC	Player	Goals	Location
1	Lucas	Ben	Find the address of the enterprise.	Uncle's place.
2	M.Jasper	Lucas	The manufacturing first step	Enterprise reception
3	Samir	Julie	Find the plans of the joystick *Optional: job, staff, studies, security policies*	Designing Office
4	Samir	Julie	Find out what to do next *Optional: jobs in the enterprise, staff in the enterprise*	Designing Office
5	Melissa	Lucas	Find the mould *Optional: where are the moulds*	Plant
6	Melissa	Lucas	Find the right machine	Plant
7	Melissa	Lucas	Confirm you have found the right mould and machine and find out what to do next	Plant
8	Operator	Julie	Knowing about the material space and about the job *Optional: find out what to do in the case of failure helping to feed a machine with the right material*	Material Space
9	Serge	Ben	Perform quality tests. *Optional: VC's job*	Laboratory Tests
10	Serge	Ben	Find out what to do next. *Optional: know what happens with broken items*	Laboratory Tests
11	Sophia	Julie	Find the electronic components, knowing about VC's job	Finishing
12	Sophia	Lucas	Finishing process *Optional: know about conditioning the product*	Finishing

4.1 Scenario and Demonstrations from Experts

The objective of the virtual agents is to engage the player in a conversation in the context of a serious [1] game called *Mission Plastechnologie*[2] (MP). In this game, the player seeks to build a joystick in order to free their uncle trapped in a video game. To build this joystick, the player must explore a factory and interact with different virtual humans through twelve distinct dialogs (i.e., chatbots), each of them occurring in a specific place of the virtual world with various mandatory goals to be achieved and optional goals to be discussed (See Table 1). Note that defining the reward for each of these dialogues is not as simple as giving a positive reward when the joystick is built by the player. Instead, virtual characters (i.e., tutors) have to instruct players, providing valuable information and supporting spontaneous conversations through a sort of fun relaxed tutoring (as mentioned in Section 2.2).

To learn human behavior, we are taking the experts' (i.e., seven subjects performing the Wizard of Oz) demonstrations from the corpus of the MP dialog scenario [19]. It contains 1250 Human-Human dialogues involving 6845 Wizard of Oz turns and 3610 player turns.

4.2 States, Actions and Transitions

As shown in Table 1, there are 12 distinct conversations in the game between 7 virtual characters (VC) and 3 player characters. Each of these dialogues talks about mandatory

[1] A serious game is a game designed for a primary purpose other than pure entertainment.

[2] The game is designed to promote careers in the plastic industry, is French speaking and was created by Artefacto, `http://www.mission-plastechnologie.com/`

and optional goals. The player either asks for information about these goals or asks for help. Accordingly, the virtual human either informs about the goals or provides help. It can also handle out of domain topics, misunderstandings or request information. The following example shows an excerpt of the third dialogue in the MP Game between Samir (the system) and Julie (the player) annotated with dialogues acts as in the corpus. The goal to be achieved by the player is to find the plans of the joystick, the goals to be discussed (which are optional) are: learn about the virtual character's job, his studies, his colleagues as well as the security policies of the enterprise.

Samir: Hello my name is Samir, the product designer {greet}
Samir: What are you doing here young people? {ask(task(X))}
Julie: We come to build the joystick of Professor Geekman {find_plans}
Samir: You are in the right place, the plans are in the closet ... {inform(do(find_plans))}
Samir: Before leaving would you like to hear about my job, the studies I did or my colleagues? {ask(domore(X))}
Julie: Ok, tell me about your job. {inform_job}

We use MDPs to model virtual humans in the game. We designed coarse-grained states containing user and system contributions to the dialogue; either by explicitly asking about the domain specific tasks (i.e. the dialogue goals) or by producing general dialogue acts (e.g., greeting, asking for help, acknowledgments, etc). A binary variable that indicates whether or not the dialogue has finished is also included. With this state representation we have 32 states for the shortest dialogue (the first dialogue in Table 1), and 432 states for the longest dialogue (i.e., the third dialogue in Table 1 with 5 goals).

State variables

1. Has any of the characters ended the dialogue with a farewell action ? : 1 for setting a terminal state, 0 otherwise.
2. The last goal either informed or requested by the system: 0 when the system has not informed/requested about any goal, otherwise the goal id (e.g., from 1 to up to 5 for the longest dialogue).
3. The last goal either asked or confirmed by the player: 0 when the user has not yet asked/confirmed about any goal, otherwise the id of the goal (e.g., from 1 to up to 5 for the longest dialogue).
4. The last general dialogue act produced by the system: 0 for absence of general dialog act, 1 when providing help, and 2 when asking the player about the task to be solved (e.g., "How may I help you").
5. The user has asked for help: 0 if the user has not asked for help, 1 otherwise.

Actions. We are considering only the following actions in our experiments.

- *quit*: farewell greeting.
- *inform(do(g_i))*: informing about how to achieve goal g_i.
- *inform(help)*: providing help
- *ask(task(X))*: Asking the player about the task, it corresponds to a general welcome sentence (e.g., "How may I help you"). Note that this action neither occurs in dialogue 1 nor in dialogue 7.

- *WAIT*: the system gives the turn back to the user.
- *ack*: the system acknowledges understanding.
- *other*: the system answers to out of context turns.

Virtual characters always greet the player at the beginning, thus we do not need to learn this behaviour.

Transition Function. The transition function is not deterministic when the next state reflects an (unpredictable) user action. This is typically the case after the WAIT system action. However, BIRL requires this transition function to be given, and we have thus estimated such non-deterministic transition probabilities using smoothed counts from the observed corpus as follows:

$$P(s'|s,a) = \frac{N(s,a,s')+\alpha}{N(s,a)+N_\chi\alpha}$$

Where $N(s,a,s')$ and $N(s,a)$ are respectively the number of times the transition (s,a,s') and the state-action pair (s,a) have been seen in the corpus, and N_χ is the number of observed state-action pairs. α is a smoothing constant arbitrarily set to 0.1.

The other transitions that reflect a system action are deterministic and have been defined as:

$$P(s'|s,a) = \begin{cases} 1, \text{ if } s' = next\_s(s,a) \\ 0, \text{ otherwise} \end{cases}$$

Where $next\_s(s,a)$ is a function that computes the next state given a system action a. For instance, when the system informs about the first goal, g_1, the action $a_t = inform(do(g_1))$ yields the next state s' to have the state variable 2 set to 1.

5 Bayesian Inverse Reinforcement Learning

The IRL problem as defined in [7] is described as follows: given a finite state space S, a set of actions $A = \{a_1, a_2, ...a_k\}$, a transition probability $P^a_{ss'}$, a discount factor γ, and a policy π, determine a set of possible reward functions R such that π is the optimal policy for the given MDP. The IRL problem is an ill-posed problem [14], because potentially an infinite number of rewards may be optimal. Bayesian IRL approaches model this uncertainty by inferring the posterior distribution of the reward vector **R**, treating the demonstration sequences as the evidence and relying on a prior on the reward function [8].

The IRL agent receives a sequence of observations of the expert's behaviour $O_\chi = \{(s_1,a_1),(s_2,a_2),...,(s_k,a_k)\}$, which means that at time step i, the virtual character χ that mimics the expert is in state s_i and takes the action a_i. After applying Bayes Theorem, the posterior can be written as:

$$Pr(\mathbf{R}|O_\chi) = \frac{Pr(O_\chi|\mathbf{R})Pr(\mathbf{R})}{Pr(O_\chi)} \tag{4}$$

We model next the reward function by a simple n-dimensional real vector, where n is the number of different states. Then, $Pr(\mathbf{R}|O_\chi)$ is the posterior distribution of the reward vector given the observed state-action pairs of the expert. $Pr(O_\chi|\mathbf{R})$ is the likelihood of the observed expert state-action pairs given the reward vector **R**. This likelihood is modeled in [8] with a parameter α representing the degree of confidence we have in the expert's ability to choose a good action as follows:

$$Pr(O_\chi|\mathbf{R}) = \frac{1}{Z}e^{\alpha \sum_i Q^*(s_i,a_i,\mathbf{R})} \tag{5}$$

$Pr(\mathbf{R})$ is the prior distribution and $Pr(O_\chi)$ is the probability of the evidence over the entire space of reward vectors **R**, which is not needed in the BIRL algorithm. The original BIRL algorithm, namely PolicyWalk, follows a Markov Chain Monte Carlo (MCMC) technique iterating as follows: Given a reward vector **R**, it performs random walks over the neighbors of **R** on a grid of length δ, finding a new proposal $\bar{\mathbf{R}}$, such that: $\bar{\mathbf{R}}(s) = \mathbf{R}(s) \pm \delta$. The proposal is accepted with probability $\min\{1, \frac{Pr(\bar{\mathbf{R}}|O)}{Pr(\mathbf{R}|O)}\}$, where the posterior is given by Eq (4).

The expected value of the reward given this posterior is then computed over all these samples. Note that the normalizing constants cancel out in the ratio used to accept the proposed $\bar{\mathbf{R}}(s)$ and that finding Q^* in Eq (5) requires to solve the MDP at every MCMC iteration. This can be done for example with the policy iteration (PI) algorithm [20].

BIRL converges slowly when applied to large state spaces. One reason for this is that it infers the reward of every state, although many states have little expert evidence. Second, searching over a reward function space easily increases the number of MCMC iterations needed to approximate the mean of the posterior. To solve these limitations, [9] proposed a modified BIRL (MBIRL) that:

– infers only those states that are similar to the observed ones according to a *kernel-based relevance function*.
– uses simulated annealing to focus the sampled distribution around its maximum, hence reducing the number of samples needed to converge. Therefore, they use a modified acceptance probability of $\left(\frac{Pr(\bar{\mathbf{R}}|O)}{Pr(\mathbf{R}|O)}\right)^{\frac{1}{T_i}}$ where T_i is a decreasing *cooling schedule*.

6 Experiments

In this section we introduce the baselines, the evaluation metrics and the experiment setup for 12 dialogues in the game. We have defined for each dialogue the state and action space as explained in Section 4.

6.1 Baselines

We evaluate the performances of the proposed system by comparing it with two baselines:

– Using random rewards (RR);

- Exploiting "locally-estimated" rewards (LR), i.e., rewards that are trained on the corpus with the additional assumptions that the reward prior $Pr(\mathbf{R})$ is uniform, that the states are conditionally independent given the reward $P(O_\chi|\mathbf{R}) = \prod_i P(s_i|\mathbf{R})$ and that the state likelihood is multinomial with parameters representing the reward $P(s = k|\mathbf{R}) = R_k$, so that the path that maximizes the cumulated reward also maximizes the likelihood. Then:

$$\begin{aligned} \arg\max_R Pr(\mathbf{R}|O_\chi) &= \arg\max_R Pr(O_\chi|\mathbf{R}) \\ &= \arg\max_R \prod_i P(s_i|\mathbf{R}) \end{aligned} \tag{6}$$

  Let n_k be the number of times the k^{th} state of the states space occurs in the expert observations: $n_k = |\{(s_i = k, a_i)\}_{i \in O_\chi}|$
  Then we want to maximize the likelihood $\prod_k P(s = k|\mathbf{R})^{n_k}$ under the constraint $\sum_k R_k = 1$, which gives the locally optimum reward:

$$\hat{R}_k = \frac{n_k}{N_\chi}$$

  with $N_\chi = |\{(s_i, a_i)\}_{i \in O_\chi}|$ the number of observed state-action pairs.

6.2 Evaluation Metrics

We consider two evaluation metrics: the policy loss [9] and the system training time.

- *Policy loss*: The policy loss is the ratio $\frac{n_{\neq}}{N_\chi}$, where $n_{\neq} = |\{(s_i, a_i \neq \pi(s_i))\}_{i \in O_\chi}|$ is the number of expert state-action pairs that disagree with the learned policy π and $N_\chi = |\{(s_i, a_i)\}_{i \in O_\chi}|$ is the number of observed state-action pairs.
- *Elapsed time*: The time in milliseconds it takes to MBIRL and to the policy iteration algorithms to finish.

6.3 Experimental Setup

We used 2000 MCMC iterations and 20 policy iterations (PI) with a discount factor $\gamma = 0.9$. The experiments were run 5 times (except for dialogues 3 and 8 that were run only once) and the averaged measures are reported in Table 2. Dialogue 3, which has the largest state and action spaces ($|S| = 432, |A| = 11$), was run with 1000 MCMC iterations and 10 PI iterations.

Parameters for BIRL. For solving the BIRL posterior defined in Eq (4), we set the parameter α of Eq(5), representing the degree of confidence we have in the expert , to $\alpha = 0.85$, based on the Inter-Annotator Agreement between experts. We use the $Beta$ distribution as prior, where $R_{max} = 6$ and $R_{min} = -6$, R_{max} was set taking into account the maximum number of goals that can be discussed in a dialogue (i.e., 5) plus the action of providing help when requested.

$$P_{Beta}(R(s) = r) = \frac{1}{(\frac{r - R_{min}}{R_{max} - R_{min}})^{\frac{1}{2}} (\frac{R_{max} - r}{R_{max} - R_{min}})^{\frac{1}{2}}}$$

Table 2. Results of MBIRL. The dialogues marked with (*) are optional. The first four columns stand for dialogue id (Id), number of dialogues ($ND.$), number of states $|S|$ and number of actions $|A|$. The next three columns represent the policy loss of the baselines and MBIRL. The last two columns shown the policy iteration (PI) and MCMC elapsed time.

Id	$ND.$	$\|S\|$	$\|A\|$	LR (p-loss)	RR (p-loss)	MBIRL (p-loss)	PI (ms)	MCMC (ms)
1	105	32	6	0.79	0.73	**0.66**	559	177062
2	112	48	7	0.82	0.82	**0.66**	852	415988
3	113	432	11	0.80	0.95	0.85	112231	590935186
4	106	192	9	0.74	0.85	0.79	15693	50988645
5	107	108	8	0.81	0.86	**0.72**	7661	5129894
6*	102	48	7	0.94	0.75	**0.69**	876	397773
7	105	72	7	0.95	0.88	**0.64**	8649	5183773
8	105	300	10	0.86	0.92	0.89	109863	351388062
9	104	108	8	0.79	0.82	**0.73**	6513	5620255
10*	93	108	8	0.79	0.80	0.82	7625	4770796
11	115	108	8	0.81	0.80	**0.71**	6949	4938122
12*	82	108	8	0.80	0.83	**0.79**	6614	4994690

Parameters for MBIRL. The state relevance kernel exploits a radial basis kernel that uses the Euclidean distance as a measure of similarity as follows:

$$k(s, s') = e^{\frac{-||s-s'||_2}{2\xi^2}},$$

where $\xi = 100$. The cooling schedule parameter is set to $\frac{1}{T_i} = 25 + \frac{i}{50}$ where i is the MCMC iteration.

7 Results and Discussion

In this section we present the results of the quantitative evaluation measured in terms of policy loss and elapsed time as explained in Section 6.2. We also present a qualitative evaluation in which we compare the trajectories of the experts and the trajectories of the optimal policy π obtained by MBIRL.

Performance. Table 2 shows the performances of MBIRL in the context of our serious game. Two important issues affect performance: the size of the state-space and the limited number of expert observations. In general MBIRL outperforms both locally-optimal and random rewards, reaching a policy loss of 0.66 ± 0.10 for the shortest dialogue, dialogue 1 ($|S| = 32, |A| = 6$) and around 0.72 for dialogues 5,9 and 11 ($|S| = 108, |A| = 8$). However, with a larger state-action space such as in dialogue 3 ($|S| = 432, |A| = 11$), 4 ($|S| = 192, |A| = 9$) and 8 ($|S| = 300, |A| = 10$), the models do not improve over the locally-optimal reward, which suggests that the number of samples that are generated is not large enough. Moreover, for state spaces greater than 300 states, MBIRL takes a prohibitively long running time to finish. For instance, dialog 3 took 590935186 milliseconds (around 7 days) to finish when running with 10 policy iterations and 1000 MCMC iterations. Similarly, dialogue 8 took 351388062 milliseconds, which is equivalent to 4 days, to finish.

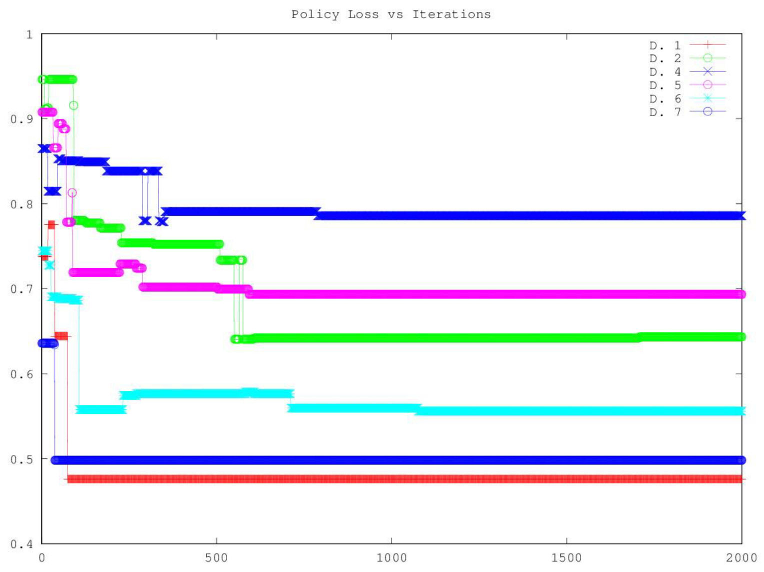

Fig. 1. 0-1 policy loss as a function of the number of MCMC iterations for the first six dialogues that finished with 20 PI

The huge computational expense for large state spaces is an important limitation of MBIRL since it needs to solve one RL problem per iteration. A potential solution to this issue might be to use appropriate function approximation both for the Q function and for modeling the reward function **R**, but this is left for future work.

Unsurprisingly, the MBIRL policy loss is higher for optional dialogues such as dialogue 10 and 12 than for mandatory dialogues within the same state-action space size (i.e., dialogues 5,9 and 11). Indeed, MBIRL does not improve significantly over either the locally-optimum or the random reward in the optional dialogue 10, showing that the scarce number of observations in this dialogue significantly affects performance (see column (ND.), number of dialogues, in Table 2).

Convergence. Figure 1 shows that MBIRL converges in around 600 iterations when using 20 PI, towards a policy loss that is better than the one obtained with the initial random rewards.

Trajectories. Figure 2 shows two dialogue trajectory excerpts with both the gold (or expert) trajectory and the trajectory inferred by MBIRL. Interestingly, in most of the dialogues, both trajectories coincide in the first state and in those states where the system has to inform about mandatory goals just after explicitly requested by the user. This is also the case of the states where the system properly provides help as requested by the user. On the other hand, the learned policy usually fails to close the dialogue and it sometimes contains repetitions e.g., once it has informed about a goal, it may inform again later on.

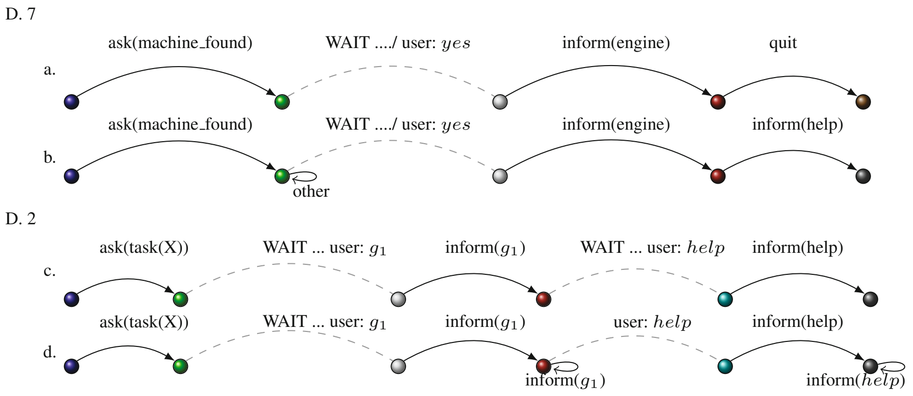

Fig. 2. Comparison of expert vs. MBIRL trajectories for dialogues 7 (top) and 2 (bottom). (a) and (c) depict expert trajectories, while (b) and (d) show the trajectories of MBIRL optimal policy π.

8 Conclusion

In this work we applied a Bayesian algorithm for apprenticeship learning to model the behaviour of distinct conversational agents in a virtual environment. The reward function is then learned from expert demonstrations. Most noticeably, the learned reward tends to reproduce quantitatively and qualitatively the expert decisions, in particular all models learned to provide information and help as requested. We conclude that the proposed approach is a viable option to learn a reward that leads to human-like policies while still benefiting from the interesting dynamic properties of Markov Decision Processes. We found that two main factors affect performance, the size of the state-action space and the limited number of expert demonstrations. Certainly, the computational expense drastically increases when dealing with complex dialogues in a large state-action space ($|S| >= 192$, $|A| >= 9$), without improving the locally optimum.

A potentially interesting future work concerns the study of function approximation to model both Q and $\mathbf{R}$ functions. Such an approach may provide an elegant way to solve the states space dimensionality issue, as well as give the possibility to model more complex behaviour such as handling misunderstandings, repetitions and out of context inputs as well as proposing new topics to be discussed. Finally, exploiting BIRL for both dialogue management and user simulation might constitute an interesting strand of research to better model uncertainty at every stage of dialogue modeling.

Acknowledgments. This work was partially supported by the French ANR (*Agence Nationale de la Recherche*) funded project ContNomina.

References

1. Levin, E., Pieraccini, R., Eckert, W.: A stochastic model of human-machine interaction for learning dialog strategies. IEEE Transactions on Speech and Audio Processing 8(1), 11–23 (2000)

2. Rieser, V., Lemon, O.: Reinforcement learning for adaptive dialogue systems. Springer (2011)
3. Pietquin, O., Dutoit, T.: A probabilistic framework for dialog simulation and optimal strategy learning. IEEE Transactions on Audio, Speech, and Language Processing 14(2), 589–599 (2006)
4. Cuayáhuitl, H.: Hierarchical reinforcement learning for spoken dialogue systems. PhD thesis, Citeseer (2009)
5. Williams, J.D., Young, S.: Partially observable markov decision processes for spoken dialog systems. Computer Speech & Language 21(2), 393–422 (2007)
6. Paek, T., Pieraccini, R.: Automating spoken dialogue management design using machine learning: An industry perspective. Speech Communication 50(8), 716–729 (2008)
7. Ng, A.Y., Russell, S.J.: Algorithms for inverse reinforcement learning. In: Icml, pp. 663–670 (2000)
8. Ramachandran, D., Amir, E.: Bayesian inverse reinforcement learning. Urbana 51, 61801 (2007)
9. Michini, B., How, J.P.: Improving the efficiency of bayesian inverse reinforcement learning. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 3651–3656. IEEE (2012)
10. Walker, M.A.: An application of reinforcement learning to dialogue strategy selection in a spoken dialogue system. Journal of Artificial Intelligence Research 12, 387–416 (2000)
11. Young, S., Gašić, M., Keizer, S., Mairesse, F., Schatzmann, J., Thomson, B., Yu, K.: The hidden information state model: A practical framework for pomdp-based spoken dialogue management. Computer Speech & Language 24(2), 150–174 (2010)
12. Tetreault, J.R., Litman, D.J.: A reinforcement learning approach to evaluating state representations in spoken dialogue systems. Speech Communication 50(8), 683–696 (2008)
13. Rojas Barahona, L.M., Lorenzo, A., Gardent, C.: An end-to-end evaluation of two situated dialog systems. In: Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue, pp. 10–19. Association for Computational Linguistics, Seoul (2012)
14. Abbeel, P., Ng, A.Y.: Apprenticeship learning via inverse reinforcement learning. In: Proceedings of the Twenty-First International Conference on Machine Learning, p. 1. ACM (2004)
15. Abbeel, P., Coates, A., Ng, A.Y.: Autonomous helicopter aerobatics through apprenticeship learning. The International Journal of Robotics Research 29(13), 1608–1639 (2010)
16. Chandramohan, S., Geist, M., Lefevre, F., Pietquin, O., et al.: User simulation in dialogue systems using inverse reinforcement learning. In: Proceedings of the 12th Annual Conference of the International Speech Communication Association, pp. 1025–1028 (2011)
17. Boularias, A., Chinaei, H.R., Chaibdraa, B.: Learning the reward model of dialogue pomdps from data. In: NIPS Workshop on Machine Learning for Assistive Techniques, Citeseer (2010)
18. Zhifei, S., Joo, E.M.: A review of inverse reinforcement learning theory and recent advances. In: 2012 IEEE Congress on Evolutionary Computation (CEC), pp. 1–8. IEEE (2012)
19. Rojas-Barahona, L.M., Lorenzo, A., Gardent, C.: Building and exploiting a corpus of dialog interactions between french speaking virtual and human agents. In: Proceedings of the 8th International Conference on Language Resources and Evaluation (2012)
20. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)