User Simulation in Dialogue Systems using Inverse Reinforcement Learning

Senthilkumar Chandramohan [1,3]*, Matthieu Geist* [1]*, Fabrice Lefèvre* [3]*, Olivier Pietquin* [1,2]

[1] Supelec - Metz Campus, IMS Research Group, France
[2] UMI 2958 (CNRS - GeorgiaTech), France
[3] Université d'Avignon et des Pays de Vaucluse, LIA-CERI, France
[1]firstname.lastname@supelec.fr [3]firstname.lastname@univ-avignon.fr

Abstract

Spoken Dialogue Systems (SDS) are man-machine interfaces which use natural language as the medium of interaction. Dialogue corpora collection for the purpose of training and evaluating dialogue systems is an expensive process. User simulators aim at simulating human users in order to generate synthetic data. Existing methods for user simulation mainly focus on generating data with the same statistical consistency as in some reference dialogue corpus. This paper outlines a novel approach for user simulation based on Inverse Reinforcement Learning (IRL). The task of building the user simulator is perceived as a task of imitation learning.

Index Terms: Inverse Reinforcement Learning, User Simulation, Spoken Dialogue Systems.

1. Introduction

User simulation for spoken dialogue systems (SDS) is a research field aiming at generating artificial interactions so as to automatically assess the quality of dialogue strategies [1] or to train machine-learning-based dialogue management systems [2]. A majority of existing methods for user simulation mainly focus on reproducing user behaviors which have the same statistical consistency as in some dialogue corpus [3, 4, 5] in addition to ASR/NLU error models [6, 7]. However, with regard to man-machine interaction, human users tend to adapt themselves according to the dialogue management engine while corpora are usually obtained using a specific dialogue system. If user simulation is used for training machine-learning-based dialogue systems [8], the dialogue management is continuously modified and nothing guarantees that the user simulation remains close to what the user behavior would be facing a novel dialogue management strategy. This is also the case in human-human interaction where the communicating subjects choose what to say and how to say it based on the (alleged) ability of the other subject. Thus, simulating the user behavior only based on the frequency of occurrence of user acts (given the context) may fail to simulate the human behavior precisely. Moreover, statistical similarity is often used to assess the quality of user simulation [9].

In this paper the task of simulating human users is perceived as a sequential decision making under uncertainty problem and the user, when deciding to generate a dialogue act in a given context, is supposed to follow a long-term policy. One way to estimate the user policy from a dialogue corpus (annotated in user perspective) is to employ reinforcement learning (RL) [10] for policy optimization. However RL-based optimization requires a reward function, describing how good it is for the user to accomplish an action in a given context, to be defined in advance. Manually specifying the reward function of the user is one possible option but is often an impossible task if the behavior to be simulated is complex [11].

Here the task of simulating the user is defined as an imitation learning problem. Inverse Reinforcement Learning (IRL) [12] is employed to estimate the reward function of the user (*i.e.*, what satisfies the user when interacting with the machine). The estimated reward function is then used by a (direct) RL algorithm for retrieving the strategy of the user which is supposed to be optimal w.r.t. the inferred reward function. The novelty of the proposed approach from the authors perspective breaks down as follows: (i) user simulation is treated as a sequential decision making problem, (ii) IRL is used for the first time in the dialogue management domain (to estimate the reward function of the user) and (iii) the estimated objective function of the user can be used as a mean to assess the performance of the user simulator [9]. Overall the primary contribution of the paper is to settle the main framework of IRL-based user simulation and demonstrate the feasibility of the method on a simple problem.

The paper starts with a description of a formalism for modeling sequential decision making in Section 2. Then formal description of IRL is presented in Section 3. Following which casting the task of user modeling as a Markov Decision Process (MDP) and employing IRL to learn the reward function of the user is discussed in Section 4. Section 5 outlines the experimental setup for learning the user behavior of a hand-crafted user model, along with the evaluation strategy. Eventually, Section 6 concludes and outlines the future directions of work.

2. Markov Decision Processes

In machine learning, the problem of sequential decision making under uncertainty is generally treated in the framework of the Markov Decision Processes [13]. Formally, an MDP is defined as a tuple $\{S, A, P, R, \gamma\}$ where S is the state space, A is the action space, $P : S \times A \rightarrow \mathcal{P}(S)$ a set of Markovian transition probabilities, $R : S \rightarrow \mathbb{R}$ the reward function and γ a discount factor weighting long-term rewards. A learning agent steps from state to state accomplishing actions. At any given time step, the agent is in a state $s_i \in S$ and transits to s_{i+1} according to $p(.|s_i, a_i)$ upon (choosing and) performing an action $a_i \in A$ according to a policy $\pi : S \rightarrow A$. After each transition the agent receives a reward $r_i = R(s_i)$. The quality of the policy π followed by the agent can be quantified by the state-action

The authors want to thank the European INTERREG IVa program for funding (ALLEGRO project) and the Région Lorraine.

value function or Q-function ($Q^\pi : S \times A \rightarrow \mathbb{R}$) defined as:

$$Q^\pi(s,a) = E[\sum_{i=0}^{\infty} \gamma^i r_i | s_0 = s, a_0 = a] \quad (1)$$

The optimal policy π^* is the one for which the Q-function is maximum for each state-action pair: $\pi^* = \arg\max_\pi Q^\pi(s,a)$. The optimal policy leads to an optimal Q-function $Q^*(s,a)$ and $\pi^*(s) = \arg\max_a Q^*(s,a)$. There exist many RL algorithms to compute the optimal policy π^* [10] and the associated $Q^*(s,a)$.

3. Inverse reinforcement learning

Given that the MDP framework is adopted, IRL [12] aims at learning the reward or utility function optimized by an *expert* from a set of observed interactions between this expert and the MDP. However, this is an ill-posed problem and there exist several possible reward functions that can match the expert behavior [12]. Thus the primary focus of most existing IRL algorithms is to retrieve some reward function (not necessarily the true reward function) which can exhibit a behavior similar to the expert's behavior. Let us assume an MDP defined by the tuple $\{S, A, P, \gamma\}/R$, where $/R$ means that the reward function of the MDP is not available. Let the feature space of the reward function be defined using a vector ϕ. Then the reward function R of the MDP can be expressed as: $R_\theta(s,a) = \theta^T \phi(s,a) = \sum_{i=1}^{k} \theta_i \phi_i(s,a)$ (where θ (resp. ϕ) is a vector of which the i^{th} component is θ_i (resp. $\phi_i(s,a)$). Using this in Eq (1):

$$Q^\pi(s,a) = E[\sum_{i=0}^{\infty} \gamma^i \theta^T \phi(s,a) | s_0 = s, a_0 = a] = \theta^T \mu^\pi(s,a) \quad (2)$$

where $\mu^\pi(s,a)$ is the *feature expectation* of the policy π. Feature expectation in simple terms can be defined as the discounted measure of features according to state visitation frequency (based on when the state is visited in the trajectory). It provides a compact yet simple mean to summarize the behavior of the user observed in the form of trajectories. Given a set of m trajectories (where H_i is the length of the i^{th} trajectory) from the expert (who acts based on some unknown policy π), the feature expectation $\mu^\pi(s_0,a)$ can be estimated by:

$$\mu^\pi(s_0,a) = \frac{1}{m} \sum_{i=0}^{m} \sum_{t=0}^{H_i} \gamma^t \phi(s_t^i, a_t^i) \quad (3)$$

The primary focus of IRL is to retrieve some reward function (through its parameters θ) used to predict a user behavior (π_{predict}) being similar to the behavior of the expert (π_{expert}).

$$R^*_\theta(s,a) = \arg\min\{J(\pi_{\text{expert}}, \pi_{\text{predict}})\}$$

where J is some dissimilarity measure between the expert behavior and the predicted user behavior. From Eq (2) it can be observed that comparing two different user behaviors (policies) in terms of feature expectation is indeed comparing the behaviors based on their value function. Notice that from Eq (1), a reward function that is non-zero only in some states can lead to a Q-function that is non-zero in every state. From the Q-function, a greedy policy $\arg\max_a Q(s,a)$ can be inferred for every state and therefore, IRL-based user simulation can generalize to unseen situations which is harder to obtain from traditional statistical simulation. Assuming that $\|\theta\| \leq 1$ (which is not restrictive: rewards are bounded and scale invariant), one has $\|Q^\pi\| \leq \|\mu^\pi\|$. Thus an easy way of computing the dissimilarity between the expert behavior π_{expert} and the predicted user behavior π_{predict} is to compute a distance between their feature expectations, that is μ_{expert} and μ_{predict}:

$$J(\pi_{\text{expert}}, \pi_{\text{predict}}) = \|\mu_{\text{expert}} - \mu_{\text{predict}}\|^2$$

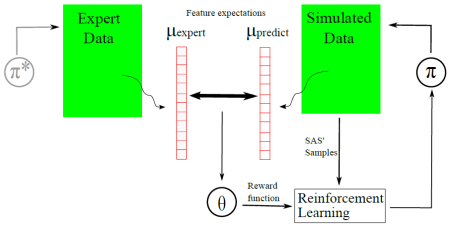

Figure 1: *User simulation using imitation learning*

Apprenticeship learning or imitation learning [11] focuses on learning to imitate the behavior observed in the sample trajectories. In this case IRL is used as a first step to retrieve the underlying reward function of the expert, which in turn is used in policy optimization to imitate the expert. Imitation learning can be pictorially represented as shown in Figure 1.

4. User simulation using IRL

To begin with, the user modeling task is casted as an MDP: *user-MDP*, where S is the state space of the user (formally defined in Section 5), A is the set of all possible user acts. Let the state-action space of the userMDP be defined by a vector of features ϕ: $S \times A \rightarrow [0,1]^k$. Also it is assumed that a set of m dialogue episodes (trajectories) from human users is made available. The goal here is to build a user simulator which can imitate the behavior observed in the dialogue corpus. Let us term π_{human} the policy of the human users. Feature expectation μ_{human} can be computed as shown in Eq (3). In the imitation learning algorithm (Algorithm (1)), step 3 is an IRL step where, based on the dissimilarity between the human and simulated user behavior, the reward function of the human user is estimated. It searches for the reward function which maximizes the distance between the value of the expert and any policy computed in prior iterations. The updated reward function is used to estimate the optimal behavior (policy for userMDP) of the simulated user in step 6. Steps 3 to 6 are performed iteratively until some convergence criteria is met, *i.e.* the distance between the human and simulated user behavior is within an acceptable range ξ. Upon convergence the algorithm outputs a set of policies Π.

Algorithm 1 User simulation using imitation learning

1: Compute μ_{expert} from dialogue corpus
2: Initiate Π with random policy $\pi_{\text{predict}} = \pi_0$ and compute μ_{predict}
3: Compute t and θ such that

$$t = \max_\theta \{ \min_{\pi_{\text{predict}} \in \Pi} \theta^T (\mu_{\text{expert}} - \mu_{\text{predict}}) \} \text{ s.t. } \|\theta\|^2 \leq 1$$

4: **if** $t \leq \xi$ **then** Terminate
5: **end if**
6: Train a new policy π_{predict} for userMDP optimizing $R = \theta^T \phi(s,a)$ with RL (LSPI).
7: Compute μ_{predict} for π_{predict}; $\Pi \leftarrow \pi_{\text{predict}}$
Goto to step 3.

In order to simulate the user one can choose the best policy from Π based on its distance with μ_{expert} or choose to use multiple policies with an associated selection probability (explained in Section 5). The algorithm does not guarantee to retrieve the true reward function of the expert but to retrieve a reward function which can exhibit a similar behavior. Even though it may seem that using the state visitation frequency in the form of feature expectation is comparable to existing approaches for user simulation (such as the n-gram method), it is worth noting that the feature expectation is not directly used. It is used to predict a reward function which in turn is used to simulate the expert behavior. Most existing ML based approaches for user simulation focus on simulating the user at the transition level when the primary focus of this work is to simulate user trajectories. Also IRL user simulation generalizes through the value function which is non-zero everywhere.

5. Experiment

In this last section, a simple experiment is presented that shows the feasibility of the method. The goal of the experiment is to learn a user model which can simulate the behavior of a hand-crafted user model. To begin with, the state-action space of the SDS and the userMDP are defined. Then the experimental setup for training and evaluation of the IRL user model is described.

5.1. Town-Information dialogue system

The problem studied in this paper is a task-oriented, form-filling dialogue system in the tourist information domain, similar to the one studied in [14]. The aim of the dialogue system is to give information about restaurants in a city based on specific user preferences. It may be recalled that the imitation learning algorithm outlined in Section 4 requires the feature expectation (defined in Eq (3)) for a user behavior π to be computed. One way to estimate this is to run a set of dialogue episodes between the dialogue manager and the user simulator (using π). The dialogue context is composed of the knowledge of 3 slots: location, cuisine and price-range of the restaurant. The list of possible system acts includes 13 actions: Ask-slot (3 actions), Explicit-confirm (3 actions), Implicit-confirm and Ask-slot value (6 actions) and finally Close-dialogue. For the sake of simplicity a hand-crafted dialogue policy is used for dialogue management. The hand-crafted dialogue strategy tries to fill and confirm all the 3 slots one after the other.

The task of user simulation is casted as an MDP. The state of the user simulator is represented by the Information State paradigm [15]. The user state is a summary of the dialogue course from a user simulation's perspective. Apart from encoding the exchange of information between the user model and the dialogue manager, the user state also includes the most recent action performed by the dialogue manager. For the 3-slot town-information dialogue problem the user model has the following state representation: {System-Act} {Slot1} {Slot2} {Slot3}, where the system-act field takes values in 0:13 representing the corresponding system acts defined above. Slot1, Slot2, Slot3 fields take values in 0:2; *i.e.* (0) the slot is empty (never provided by the user), (1) the slot has been provided by the user, (2) the slot is confirmed. The action space of userMDP includes the following 10 user acts: remain silent (Silent), provide-all-values (AllSlots), provide-one-value (OneSlot: 3 actions), confirm slot value (Confirm: 3 actions), negate slot value (Negate: 3 actions) and hangup (CloseDialogue). The discount factor of the userMDP is set to 0.95.

Table 1: *Hand-crafted user behavior*

SystemAct	UserActs (probability)
Greet	Silent (0.7) AllSlots (0.3)
AskSlot	OneSlot (0.95) AllSlots (0.05)
Explicit-Conf	Confirm (1.0)
Implicit-Conf	OneSlot (0.9) Negate (0.1)
CloseDialogue	Silent (1.0)

5.2. Learning to imitate

The main originality of this paper is to cast the user simulation problem as an IRL problem. To exemplify the method, this section describes how data generated from a hand-crafted (expert) user simulation can be used to perform imitation learning. The user behavior to be imitated (expert policy for userMDP) is detailed in Table 1. A set of trajectories is created using the dialogue manager and the user simulation. Using the generated data the feature expectation of the expert user behavior π_{expert} is computed as shown in Eq (3). The computed feature expectation will be the μ_{expert} since the task here is to imitate the expert behavior. First a set of dialogue episodes are generated using the userMDP with a fully random action selection π_{random} and the dialogue manager. The feature expectation of the random policy can then be computed from the dialogue trajectories. Since the necessary expert behavior is available in the form of dialogue trajectories, μ_{expert} can be computed.

The reward function of the user simulation is estimated based on the distance between the feature expectations of the expert and simulated user behavior. At each iteration of imitation learning the intermediate reward function is used to estimate the optimal behavior for userMDP using LSPI [16]. Notice that LSPI is a batch algorithm which allows learning from fixed sets of data (this is an important feature to extend the proposed method to real applications). Upon convergence (ξ set to 0.1) the best policy π^* of the userMDP (chosen from Π based on the least dissimilarity measure between π_{predict} and π_{expert}) will exhibit the expert user behavior.

5.3. Evaluation of user behavior

To facilitate evaluation, 1000 dialogue episodes are generated using the dialogue manager, the trained IRL user simulation and the expert user simulation (1000 episodes each). Figure 2 presents the average choice of user actions taken by the two different user simulations. The behavior of the expert user simulation is stochastic (see Table 1) and the behavior of the IRL user simulation is deterministic as only one of the policies retrieved during training is used during evaluation. However it is also possible to obtain a stochastic user behavior by employing a *policy mixer* to pick up a different policy before starting each dialogue episode. Let the probability of choosing a policy $\pi^i_{\text{predict}} \in \Pi$ be λ_i . The values of $\lambda_{1\ldots n}$ can be heuristically determined using a Gibbs distribution: $\lambda_i = e(-d_i/\tau)/\sum_{j=1}^{n} e(-d_j/\tau)$, where d_i is the distance between μ_{expert} and μ^i_{predict} estimated during training. During the evaluation, τ is set to 0.01 (ensures that the policies closest to π_{expert} are given more preference). Let the behavior of IRL user simulation which employs policy mixer be termed as MixIRL-UserSim.

Using IRL to estimate the reward function of the user provides a unique opportunity to evaluate the simulated behavior using the reward function itself. Since the reward function represents the intention of the user, the reward estimated using it can be used to quantify the effectiveness of the user model to

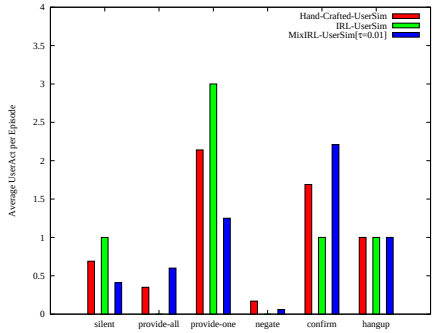

Figure 2: *Frequency of user actions per episode*

Table 2: *Hand-crafted vs IRL user behavior*

UserSim	AvgLength	AvgReward
Hand-crafted	6.03	2.6
IRL	6.0	2.7
Mixed-IRL	5.5	2.9

simulate the user. Table 2 shows the average discounted reward obtained by the user simulations based on the estimated reward function. It can be observed that the rewards obtained by the IRL user simulations and the hand-crafted user simulation, along with the average dialogue lengths, are almost the same, whereby consolidating the hypothesis of similar behaviors.

A similarity measure suggested by [17] is used to compare the behaviors. This metric is used to measure the correlation between the trajectories of the expert user model, π_{hc} and the IRL user model, π_{irl}:

$$Sim(\pi_{hc}, \pi_{irl}) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{1 + \mathrm{rank}(\mathrm{a_i})}$$

with n the number of user acts observed in the trajectory generated by the expert user model. Ranking of user acts (selected by π_{hc}) based on their Q-value (obtained using Q-function of the IRL used model) gives an indication to what extent the user acts selected by the expert model correlates with the learned IRL user model. Upon comparing the expert user model with the IRL user model, the similarity measure is found to be 0.48 for IRL-UserSim and 0.46 for MixIRL-UserSim which are close to the optimal value 0.5 (both rankings are identical, and then $\mathrm{rank}()$ is 1 constantly). This confirms that the behavior of the IRL user model correlates very well with the expert's behavior.

6. Conclusion and future work

This paper presents a novel approach for building user simulations for dialogue systems using IRL. Preliminary experimental results show that using IRL is possible to learn deterministic and stochastic user simulation strategies. The dialogue iterations generated using trained IRL user models closely correlates with that of hand-crafted users. IRL based methods provide a unique opportunity to learn even complex user models since they are based on the computation of a utility function and generate adaptive behaviors (goal-driven and not frequency-based action selection). We recognize that the experiments are currently not large enough and the user simulation is primarily reflecting the data similar to other methods. Even if not demonstrating yet the full potentiality of the approach, the experimental results provide evidence that the method is worth being investigated further. Since IRL user simulation can generalize from data it provides an opportunity for schemes such as co-adaptation of dialogue manager and user simulator.

In future work, we want to measure the generalization capabilities of the method as well as the co-adaptation phenomenon when the dialogue system changes. We also mean to measure the quality of RL-based dialogue management policies learned using IRL user simulation. This may reveal the effectiveness of IRL user simulation when compared to other approaches for user modeling. And it would be also very interesting to explore the possibility of evaluating if not quantifying the performance of other user simulations using the reward function of human user retrieved by IRL [9]. We will also investigate batch methods for IRL [18].

7. References

[1] W. Eckert, E. Levin, and R. Pieraccini, "User Modeling for Spoken Dialogue System Evaluation," in *Proc. of ASRU*, 1997, pp. 80–87.

[2] J. Schatzmann, K. Weilhammer, M. Stuttle, and S. Young, "A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies," *Knowledge Engineering Review, vol. 21(2), pp. 97–126*, 2006.

[3] O. Pietquin and T. Dutoit, "A probabilistic framework for dialog simulation and optimal strategy learning," *IEEE Transactions on Audio, Speech & Language Processing, 14(2): 589-599*, 2006.

[4] J. Schatzmann, B. Thomson, K. Weilhammer, H. Ye, and S. Young., "Agenda-based User Simulation for Bootstrapping a POMDP Dialogue System," in *Proc. of HLT/NAACL*, 2007.

[5] K. Georgila, J. Henderson, and O. Lemon, "Learning User Simulations for Information State Update Dialogue Systems," in *Eurospeech*, 2005.

[6] O. Pietquin and R. Beaufort, "Comparing ASR Modeling Methods for Spoken Dialogue Simulation and Optimal Strategy Learning," in *Proc. of Eurospeech'05*, 2005, pp. 861–864.

[7] O. Pietquin and T. Dutoit, "Dynamic Bayesian Networks for NLU Simulation with Application to Dialog Optimal Strategy Learning," in *Proc. of ICASSP'06*, 2006, pp. 49–52.

[8] O. Lemon and O. Pietquin, "Machine learning for spoken dialogue systems," in *Proc. of InterSpeech'07*, Belgium, 2007.

[9] O. Pietquin and H. Hastie, "A survey on metrics for the evaluation of user simulations," *Knowledge Engineering Review*, 2011.

[10] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 3rd ed. The MIT Press, March 1998.

[11] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proc. of ICML*, 2004.

[12] A. Y. Ng and S. Russell, "Algorithms for inverse reinforcement learning," in *Proc. of ICML*, 2000.

[13] R. Bellman, "A markovian decision process," *Journal of Mathematics and Mechanics, vol. 6, pp. 679–684*, 1957.

[14] O. Lemon, K. Georgila, J. Henderson, and M. Stuttle, "An ISU dialogue system exhibiting reinforcement learning of dialogue policies: generic slot-filling in the TALK in-car system," in *Proc. of EACL'06*, Morristown, NJ, USA, 2006.

[15] S. Larsson and D. R. Traum, "Information state and dialogue management in the TRINDI dialogue move engine toolkit," *Natural Language Engineering, vol. 6, pp 323–340*, 2000.

[16] M. G. Lagoudakis and R. Parr, "Least-squares policy iteration," *Journal of Machine Learning Research*, vol. 4, pp. 1107–1149, 2003.

[17] J. Schatzmann, M. N. Stuttle, K. Weilhammer, and S. Young, "Effects of the user model on simulation-based learning of dialogue strategies," in *Proc. of ASRU'05, Puerto Rico*, 2005.

[18] E. Klein, M. Geist, and O. Pietquin, "Batch, Off-policy and Model-Free Apprenticeship Learning," in *IJCAI-ALIHT Workshop 2011*.