# Interactive reinforcement learning for task-oriented dialogue management

**Pararth Shah** **Dilek Hakkani-Tür** **Larry Heck**
Google Research
Mountain View, CA 94043
pararth@google.com, {dilek, larry.heck}@ieee.org

## Abstract

Dialogue management is the component of a dialogue system that determines the optimal action for the system to take at each turn. An important consideration for dialogue managers is the ability to adapt to new user behaviors unseen during training. In this paper, we investigate policy gradient based methods for interactive reinforcement learning where the agent receives action-specific feedback from the user and incorporates this feedback into its policy. We show that using the feedback to directly shape the policy enables a dialogue manager to learn new interactions faster compared to interpreting the feedback as a reward value.

## 1 Introduction

Task-oriented dialogue systems assist users in completing specific tasks such as finding a restaurant or booking movie tickets through natural language interactions. The dialogue manager, also referred to as dialogue policy, is the decision-making module in a dialogue system, which chooses system actions at each step to guide the dialogue to successful task completion. The system actions include interacting with the user for getting specific requirements for accomplishing the task, as well as negotiating and offering alternatives. Optimization of statistical dialogue managers using Reinforcement Learning (RL) methods is an active and promising area of research (Fatemi et al. (2016); Su et al. (2016a)).

Dialogue management has significant differences compared to other discrete action domains that are the focus of much of deep RL research, such as game playing (Mnih et al. (2013)): an Atari game playing agent may be narrower in breadth, i.e., may have only a handful of moves such as going up, down, left or right, while a dialogue manager has a broader variety of system dialogue acts available, each associated with distinct semantics. An episode for a robot or game playing agent may be larger in depth, i.e., many games consist of hundreds of steps, where each action individually makes a small change in the environment state. A task-oriented dialogue, on the other hand, usually consists of fewer turns, and each system action can crucially alter the direction or length of the dialogue. Consequently, mistakes by the dialogue manager are both costlier and more temporally localized compared to these domains.

In these respects, dialogue management is similar to strategy games which require long term planning and where each individual move has a large impact on the game state. Deep RL (Mnih et al. (2013)) has been successfully applied to such strategy games, eg. Go (Silver et al. (2016)) and chess (Lai (2015)). Silver et al. (2016) applied a novel combination of three techniques to master the game of Go: (i) pretraining a policy network and a value network on a large corpus of human played games using supervised learning to narrow the search space to the most promising actions in each state; (ii) further refinement of policy and value networks through simulated self-play between different versions of the network during training; and (iii) employing Monte Carlo Tree Search (MCTS) rollouts (Browne et al. (2012)) to explore the game-tree below the current state to evaluate moves during actual play. In contrast, dialogue management is an asymmetric, imperfect information game with no predefined

Appears in: Workshop on Deep Learning for Action and Interaction (NIPS 2016), Barcelona, Spain.

set of rules, which complicates the application of these methods to dialogue management: (i) it is expensive to collect large, high quality data sets of dialogues with expert human agents and real users for every kind of task and user behavior that the dialogue system may be expected to handle; (ii) since the game is asymmetric, it is not straightforward to apply self-play to exhaustively explore the game tree; further, the flexibility of human conversations and lack of precise models of user goals and behavior make it laborious to engineer a realistic user simulator; and (iii) uncertainty over a user's goals and strict latency expectations for a real-time dialogue agent make it difficult to leverage MCTS rollouts at inference time.

What works in favor of dialogue management is that unlike the domains mentioned above, dialogue between a user and an assistant is a collaborative game where two players work together to accomplish a goal. One player, the user, needs to access some information or complete some action, and the other player, the dialogue system, has access to a database or service through which the user's goal can be achieved. The two players communicate with each other through dialogue moves (we refer to these as *dialog acts*). The user is usually willing to provide explicit or implicit feedback about the system's actions if it leads to demonstrable improvements in the system's performance (Knox et al. (2012)). Dialogue systems which can take advantage of this feedback could potentially accelerate their learning. Moreover, such interactive feedback from actual users of the system is valuable for adapting the system to handle dialogue flows that were not present in the training corpus or were not covered by the user simulator.

In this work, we investigate Interactive Reinforcement Learning (IRL) methods which introduce an action-specific feedback signal (Thomaz et al. (2005, 2006)) in addition to the task-level reward. The agent incorporates this feedback into its learning process, leading to faster convergence with minimal human involvement. The key contribution of this paper is the application of the Interactive RL paradigm to build task-oriented dialogue managers that can adapt to novel user behaviors that are unseen during training. We present preliminary experimental results showing that this can improve performance with respect to objective dialogue quality measures.

## 2 Preliminaries

### 2.1 Task-oriented dialogue systems

A typical architecture of task-oriented dialogue systems consists of four components (Young (2006); Zhao and Eskenazi (2016)): natural language understanding (NLU) maps natural language user turns to a semantic representation called dialogue acts and slot value pairs; dialogue state tracker (DST) keeps track of belief state of user's goal over the course of the dialogue; dialogue manager (DM) maps the dialogue state to a system action; and natural language generation (NLG) maps the system action to a natural language system utterance. The semantic representation of the user and system utterances can be modeled as a discrete semantic frame or as a vector embedding in a continuous semantic space. We focus on the former in this work but the methods described here can be generalized to the latter representation as well.

A dialogue is termed successful if the user's task is completed and all the information requests are satisfied. Table 1 outlines a sample dialog interaction between a user and system agent for the task of finding a restaurant. The dialog manager operates at the semantic representation level: it receives a new observation in the form of a semantic dialog frame from NLU and DST modules, and emits a system dialogue action which the NLG module converts to natural language and communicates to the user.

Since conversations with humans are arbitrarily flexible, it is cumbersome to engineer a rule-based dialog manager to handle every possible conversation flow. Statistical dialogue managers that can learn from a corpus of dialogues or from interactions with real users can adapt themselves to user behaviors that were not foreseen by the system designer. Further, the dialogue manager must employ planning over multiple turns of the dialogue to favor strategies that result in task completion with shorter dialogues.

### 2.2 Reinforcement learning

Reinforcement Learning (RL) problems are characterized by an agent interacting with a dynamic environment with the objective of maximizing a long term reward (Sutton and Barto (1998)). The basic

| Dialogue turns | Dialogue acts |
|---|---|
| User: Suggest an Indian restaurant for dinner. | intent(find_restaurant), inform(cuisine=Indian, meal=dinner) |
| System: Which area are you interested in? | request(location) |
| User: Near Mountain View. | inform(location=Mountain View) |
| System: Sakoon and Shiva's are good choices in Mountain View. | select(restaurant_name=Sakoon, restaurant_name=Shiva's) |
| User: What is the price range for Sakoon? | request(price_range, restaurant_name=Sakoon) |
| System: Sakoon is in the moderate price range. | inform(price_range=moderate, restaurant_name=Sakoon) |
| User: Thanks, that works for me. | thank_you() |

Table 1: A sample dialogue interaction for finding a restaurant. Each turn is labeled with the user or system dialogue action that is input to or output from the dialogue manager.

RL model casts the task as a Markov Decision Process (MDP) defined by the tuple $\{S, A, T, R, \gamma\}$ of states, actions, transition function, reward, and discount factor. In the context of dialogue systems, states are dialogue contexts that are the agent's interpretation of the environment, and are usually represented as a distribution over user intents, dialogue acts and slots and their values (i.e., $intent(buy\_ticket)$, $inform(destination = Atlanta)$). Actions are possible communication behaviors that are available to the system at each state, and are usually designed as a combination of dialogue act tags, slots and possibly slot values (i.e., $request(departure\_date)$) (Williams (2006)).

The optimization task consists of finding a policy $\pi_\theta : S \rightarrow P(A)$ that maximizes the expected cumulative discounted rewards $J(\theta) = E\left[R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + ...\right]$, where $\theta$ parameterizes the agent's policy. In this work, we use the REINFORCE rule (Williams (1992)) to iteratively update $\theta$ using policy gradients. Although other RL techniques like actor-critic based models could have been used, we chose the simpler REINFORCE since the focus of this research is to study the benefits of adding interactive feedback to an RL agent, which would also apply to other variants.

The policy gradient can be empirically estimated as:

$$\nabla_\theta J(\theta) = \frac{1}{B} \sum_{i=1}^{B} \sum_{t=1}^{T} \nabla_\theta \log \pi(a_t|s_t;\theta)(v_{i,t} - b)$$

where $B$ is the number of episodes sampled in one batch, $v_{i,t} = \sum_{k=0}^{T-t} \gamma^k R_{i,t+k}$ is the cumulative discounted reward at step $t$ in episode $i$, and $b$ is a baseline function, which we compute as the exponential moving average of the previous rewards.

## 3 Interactive RL for dialogue

### 3.1 Motivation

Instead of teaching the system how to perform a task, RL allows the system designer to specify which end states are desirable, and the system figures out how to efficiently reach one of those states. Using RL to train a dialogue manager's policy lets the dialogue manager explore more dialogue flows and optimize its actions for higher chances of overall task completion. However, applying vanilla RL to dialogue management has a number of limitations. A significant issue is the difficulty of devising realistic user simulators that can cover the variability and flexibility of natural conversations exhibited by a human user. Popular methods for building user simulators employ simplifying assumptions about the user's behavior (Schatzmann et al. (2007)) and estimate hidden parameters of the user model from logs of conversations between agents and human users (Schatzmann and Young (2009)). These approaches, although intricate, are lossy models of human conversational behavior that lead to a "reality gap" between simulation and real-world usage of the dialogue manager, which similarly afflicts the domain of robot control (Koos et al. (2010)).

Even with a reasonable user simulator, a further complication with using RL for dialogue is the complexity of a task-independent reward function that scores a dialogue for task completion and

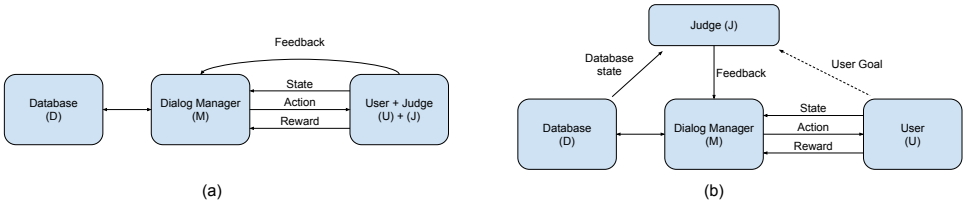


Figure 1: Interactive RL for training a dialogue manager. (a) The user provides a feedback signal to the dialogue manager. (b) A third agent, J, observes the interactions and provides feedback to the dialogue manager.

naturalness of the conversation. This is probably unique to the realm of natural language interactions, which lacks a concise set of rules that encode the feasibility and desirability of various conversational paths and end states (Biber (1992)). Other domains do not suffer from this issue: Atari emulators have precise game dynamics, Go and chess are fully specified by a small set of rules, and robot control environments are governed by the laws of physics. Discourse analysis is a widely studied area of research (Grosz and Sidner (1986); Jørgensen and Phillips (2002); Paltridge (2012)), but it lacks a unifying theory of discourse which can be boiled down to a manageable set of rules that define a general-purpose reward function for scoring the dialogue manager's actions. Most of the previous work in RL for dialogue management has tackled this issue by defining an ad-hoc reward function which runs the completed dialogue through a series of manually crafted checks to score the dialogue on various dimensions, for example, did the system answer the user's information requests, did the system offer a valid suggestion, etc. This does not scale well as the reward is path-dependent and the system designer must take care to write rules which are general enough to cover all possible dialogue flows, otherwise the RL agent may learn to optimize for undesirable behavior that is unintentionally highly rewarded.

We propose an alternate approach using Interactive RL (IRL) (Thomaz et al. (2005, 2006)) to alleviate these issues by shaping the dialogue manager's behavior with two independent signals. The first is a task-level reward which is path-independent and depends only on (i) whether the user's overall goal was achieved, and (ii) number of turns taken to reach the goal (Gašić et al. (2010)). This reward is used to optimize the dialogue manager's policy using regular policy gradients. The second signal is a turn-level feedback signal which assigns a correctness probability to each state-action pair, and is used to reweight the probability of choosing an action at each turn. Since the interactive feedback is received during the actual use of the system, it can help to overcome deficiencies of the training data or user simulations.

### 3.2 Modeling interactive feedback

Our setup for IRL for dialogue consists of a dialogue manager agent $M$, a user agent $U$, a database $D$, and a judge $J$ (Figure 1). $D$ consists of a list of entities with attributes, and is hidden from $U$. $U$ has an internal goal $G$ that is hidden from $M$. $G$ assigns a score of 0 or 1 to each entity in $D$, $\forall d \in D : g(d) \rightarrow \{0, 1\}$, and $G$ is achieved if $M$ offers an entity with score 1 to $U$. $U$ and $M$ interact using dialogue acts which represent conversational actions, such as *request*, *offer*, *confirm*, etc. An episode consists of a complete dialogue between $U$ and $M$. The dialogue is successful if the goal $G$ was achieved. $M$ receives the task-level reward $r$ at the end of a dialogue. The judge $J$ can choose to provide feedback to $M$ at any turn, which changes $M$'s behavior. There are two configurations for $J$: (a) the user agent can serve as the judge, or (ii) $J$ can be a third agent who observes the interactions between $M$ and $U$ and advises $M$. In the first case, $J$ has access to the user goal but not to the database, while in the second case $J$ can inspect the database $D$ when choosing its feedback, but may not have access to the user goal apart from what it can infer from the dialogue history. The former configuration models the scenario where a human user is providing feedback to improve the system, while the latter case models the scenario where a human agent is observing the interactions of the dialogue system with a user, either in real-time or from conversation logs, and rating the actions of the dialogue system.

The feedback, $f_{(s,a),t} \in \{-1, +1\}$ is a label on the current action taken by $M$, and can be used to prune the search space explored by $M$, focusing the exploration on actions that are likely to lead to

successful dialogues and thereby increasing the frequency of seeing positive rewards during training. However, human feedback can be irregular and inconsistent. To account for this uncertainty in the correctness and completeness of feedback, consider $P_F(a|s)$ as the probability that $a$ is the optimal action to take in state $s$ based on the explicit feedback received from the judge, and let $\pi_F$ be a policy which takes actions according to the feedback probability, i.e. $\pi_F(s,a) = P_F(a|s)$. Griffith et al. (2013) describe a model for estimating the optimality of a state-action pair $(s,a)$ based on feedback:

$$P_F(a|s) = \frac{C^{\delta_{s,a}}}{C^{\delta_{s,a}} + (1-C)^{\delta_{s,a}}}$$

$$f_{(s,a),t} \in \{-1, +1\}$$

$$\delta_{s,a} = \sum_t f_{(s,a),t}$$

where $C$ ($0 < C < 1$) is the probability that the feedback $F$ is consistent with the optimal policy, and $\delta_{s,a}$ is the difference between the number of positive and negative labels received for action $a$ in state $s$. We use this formulation to estimate $\pi_F$ based on the feedback. $M$ samples actions according to the combined policy $\pi \propto \pi_R \times \pi_F$, where $\pi_R$ is $M$'s current estimate of the policy that optimizes the reward signal from the user. Multiplying distributions together is the Bayes optimal method for combining probabilities from conditionally independent sources (Griffith et al. (2013)).

## 4 Related work

Using reinforcement learning based approaches to train a dialogue manager has been extensively researched (Levin and Pieraccini (1997); Williams (2006); Gašić et al. (2012); Pietquin (2013); Cuayáhuitl (2016); Dhingra et al. (2016)). Recent work (Gašić et al. (2013); Cuayáhuitl et al. (2015)) has shown the success of RL approaches to train dialog managers using little to none supervised training data, but they require careful engineering and domain expertise to create summary actions or restricted action sets to narrow the search space. The approach we present in this paper removes the need for domain-specific engineering of the action space by effectively learning soft action masks based on incremental feedback from non-expert judges.

Fatemi et al. (2016); Su et al. (2016a) describe a two-stage training method which also alleviates the need for summary spaces or restricted action sets, by teaching the network to identify good actions in each state by first training on a corpus of Wizard of Oz (WOz) dialogues, where human agents play the role of the system. Wen et al. (2016) also described an approach for collecting WOz dialogues to train a dialogue system end-to-end using supervised learning. All of these approaches require collecting dialogues with crowd workers trained as system agents, which is expensive, and the interaction style can vary across workers. The dialogues collected from humans talking to each other can be too complex to learn from, whereas incrementally improving the capability of a dialogue agent allows it to gradually tackle more complex interactions. Our approach reduces the need to collect dialogues from human agents, and benefits from feedback from actual users of the dialogue system who critique the actions taken by the agent, guiding the agent to improve its performance.

Su et al. (2016b) presented a method for actively learning a reward function for a dialogue task by querying the user for feedback about the overall success of the dialogue. Their approach is complementary to ours as it uses feedback to estimate the quality of the full dialogue, while our approach uses feedback to identify optimality of individual system actions.

Interactive learning from human-provided rewards has been extensively studied (Thomaz et al. (2005, 2006); Knox and Stone (2012); Loftin et al. (2014)), especially in the robot control domain. Suay and Chernova (2011) showed that incorporating teacher feedback into the policy learning process of a real-world robot agent significantly improves the policy learning time by reducing the number of states the agent explores. However, these approaches interpret the human feedback as a reward value, while we follow the approach described in Griffith et al. (2013) to treat the feedback as a label on the specific action taken by the agent. Cederborg et al. (2015) present experimental results of applying this approach with human teachers, showing that the assumptions made by the policy shaping algorithm about human teachers are reasonable.

# 5 Experiments

## 5.1 Setup

We present an experiment to evaluate the use of interactive feedback to shape the behavior of a dialogue manager. The dialogues are restricted to a toy task of using an assistant to search a restaurant database for results that match a set of goal constraints. Our experiment setup is entirely synthetic: we construct simulated user, judge and expert system agents to control the content of the dialogues used for training and evaluating the dialogue manager. An important concern for dialogue managers is their ability to effectively handle user interactions that were not present in the training corpus or covered by the user simulator used for training. The experiment is designed to simulate unseen behavior by the user in the test dialogues, and we study the dialogue manager's ability to learn to handle these interactions, with and without interactive feedback.

The restaurant database consists of rows having these attributes, also referred to as 'slots': restaurant name, location, price range, cuisine, meal, rating. The user goal is the tuple: location, price range, cuisine, meal, rating, which denotes the user's constraints. Some constraints may be unspecified, in which case the user is indifferent to the value of those slots. A dialogue is termed successful if the dialogue manager offered a restaurant that matches all the constraints of the user goal. The simulated user agent is a stochastic agenda-based user simulator (Schatzmann et al. (2007)), which chooses user actions based on (i) a goal, (ii) a user profile (co-operative/aggressive, verbose/succinct, etc), and (iii) previous system actions. The dialogue acts available to the user are: *greeting*, *inform*, *request*, *confirm*, *affirm*, and *thank_you*. (See Appendix for a full description.) At each turn of the dialogue, the user emits one or more user actions, which are combinations of user dialogue acts and slot-value pairs. The dialogue act transitions are based on manually crafted rules which are parameterized by the user profile parameters. During training, the goal and user profile are randomly sampled at the start of each dialogue. For evaluation, we sampled 1000 user goal and user profile pairs, and use these as a fixed set of seeds for parameterizing the user simulator across evaluations of each version of the dialogue manager. (We will refer to this set as *EvalSeeds*.)

The dialogue manager is modeled as a feedforward deep neural network with two hidden layers followed by a softmax layer that produces a distribution over all system actions. The inputs to the network are (i) user action, (ii) previous system action, (iii) the dialogue state, i.e. a binary vector indicating which slots have been specified by the user, and (iv) the database state encoded as a binary vector comprising of flags which indicate if the database contains zero, one, two, or three and more results matching the constraints specified till the current turn of the dialogue. The dialogue acts available to the dialogue manager are: *request*, *offer*, *select*, *inform* and *nomatch*. (See Appendix for a full description.) At each turn of the dialogue, the dialogue manager emits one or more system actions, which are combinations of system dialogue acts and slot-value pairs. We also construct a handcrafted rule-based expert dialogue manager which can perfectly handle dialogues with both simulators. The expert agent is used for simulating the collection of a corpus of dialogues of expert human agents interacting with users.

To evaluate the dialogue manager, we construct two versions of the user simulator, which we will refer to as Sim1 and Sim2. Sim2 has all of the dialogue acts available for use, while Sim1 has all except the *confirm* dialogue act, which lets the user verify if a particular constraint is satisfied by an offered restaurant, eg. "Does Sakoon have a lunch menu?" is represented as *confirm(restaurant_name=Sakoon,meal=lunch)*. A dialogue manager trained with Sim1 and evaluated with Sim2 will see *confirm* actions only during evaluation. We are interested in analyzing whether the dialogue manager can learn to respond to a *confirm($s_1$,$s_2$)* user action with the appropriate system action, which is an *inform($s_1$,$s_2$)* action with the same pair of slots. We simulate a feedback signal which is received from the user whenever the system responds to a *confirm* user action. The feedback is +1 if the system responded correctly and -1 otherwise.

The dialogue manager model is pretrained using supervised learning on a small dataset of 50 dialogues collected with the expert agent interacting with Sim1. This bootstraps the model by reducing the space of actions it needs to consider. After this the dialogue manager is made to interact with Sim2 and receive a reward at the end of each dialogue. We follow a normalized reward scheme similar to Fatemi et al. (2016): the model receives a reward of +1 for successful dialogues and -1 for unsuccessful dialogues, with a penalty of -0.03 for every turn, to encourage shorter dialogues. The model parameters are updated using the REINFORCE rule to calculate the policy gradients.

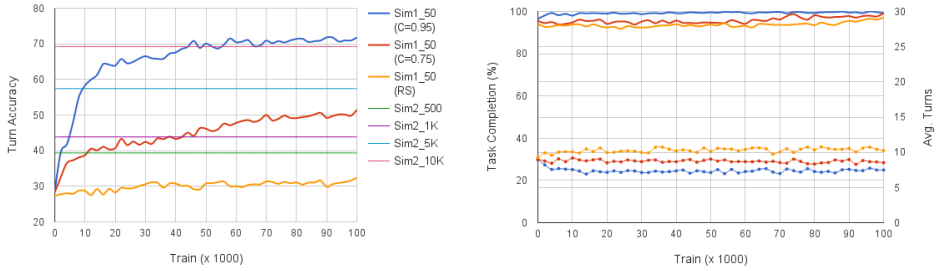


Figure 2: (a) The turn level accuracies of dialogue managers trained with and without interactive feedback. (b) The task completion rates (solid lines) and avg. turns per dialogue (linespoints) for dialogue managers incorporating interactive feedback.

The updates are done in batch and transitions are sampled from an experience pool which holds the most recent transitions observed during training. After every 2000 training dialogues, the models are evaluated against Sim2 for 1000 simulations (seeded with *EvalSeeds*) with the parameter updates turned off.

We consider two approaches to incorporating the feedback. As a baseline, we implemented an agent which interprets the feedback as a reward of +0.5 or -0.5 on the current turn. This reward is incorporated into the policy gradient update, and the dialogue manager samples actions in an $\epsilon$-greedy manner during training. In the second approach, the dialogue manager caches all labels received for every state-action pair, and uses the counts to compute $\pi_F$ using the policy shaping formulation described in Section 3.2. Actions are always sampled from the distribution $\pi_R \times \pi_F$. We tried two variations for this approach with two values of $C$, $C = 0.95$ and $C = 0.75$, which sets the consistency of the received feedback.

### 5.2 Results

We compare the turn-level accuracy for the different approaches, computed as the ratio of number of times positive feedback was received to the total number of times feedback was received at all. For a fixed feedback signal, an increasing trend in this value indicates that the dialogue manager is able to adapt to the feedback and correct its mistakes. The first graph of Figure 2 shows the turn-level accuracy for the three variations of incorporating interactive feedback: a reward shaping baseline (*Sim1_50 (RS)*), and two versions of policy shaping (*Sim1_50 (C=0.95)* and *Sim1_50 (C=0.75)*). All three models were pretrained on 50 dialogues generated with Sim1, followed by 50 iterations of interaction with Sim2 consisting of training on 2000 simulated dialogues with RL updates and evaluation with 1000 dialogues. The accuracies are averaged over 5 runs of the entire training process.

As seen from the graph, policy shaping in the ideal case (*Sim1_50 (C=0.95)*) is able to rapidly learn to handle *confirm* requests based on the feedback received, while in the case of policy shaping with a reduced feedback consistency (*Sim1_50 (C=0.75)*), the model is able to learn but at a slower rate. In contrast, in case of interpreting the feedback as a reward value (*Sim1_50 (RS)*), the model is learning at a much slower rate as it randomly explores the space of system actions to identify the correct responses. Accumulating the action-specific feedback into a single model which is also optimizing for long term reward does not allow the model to use the full amount of information available in the feedback signal as effectively as having a separate distribution which holds the feedback policy (Griffith et al. (2013)).

The second graph plots the task completion and average number of turns in the evaluation dialogues for each of the three models. The task completion is high since the models have learnt the basic interaction of requesting constraints and offering restaurant names from training on the 50 dialogues generated by the expert agent and Sim1. The user simulator Sim2 ignores turn-level mistakes made by the dialogue manager and optimistically continues the dialogue, which is why it is possible for the models to have high overall task completion in spite of lower turn-level accuracy. Since the task completion and average number of turns are consistent through the interactive RL phase, this serves as a sanity check that incorporating the interactive feedback does not hamper the model's ability to complete the overall task.

Finally, to understand the cost of using interactive feedback to train the dialogue manager, we compare the data required to reach similar turn-level accuracy when training a dialogue manager on dialogues collected using an expert agent interacting with similarly behaving users. To simulate this, we generated 4 sets of dialogues, of sizes 500, 1000, 5000 and 10000, between the expert agent and Sim2. We trained 4 instances of the same dialogue manager model described above, using only supervised learning on these generated dialogues. The first graph of Figure 2 is marked with the turn-level accuracies of these models on 1000 simulated evaluation dialogues with Sim2 (seeded with *EvalSeeds*), averaged over 5 runs. The policy shaping based model *Sim1_50 (C=0.95)* is able to meet the accuracy of *Sim2_5K* after training for 10,000 dialogues, and that of *Sim2_10K* after 46,000 dialogues. This allows us to estimate that in our case, training using RL with interactive feedback requires 2x-5x as many dialogues as compared to training directly from expert policies using supervised learning. However, interactive feedback can be obtained from actual users or non-expert crowd workers, while the latter approach requires training human agents as expert assistants, which has an additional overhead.[1]

# 6 Conclusion

We presented a method for integrating turn-level feedback with a task-level reward signal and outlined how this can enable a dialogue manager to learn to handle new behaviors while maintaining overall task completion performance. We described results from a synthetic experiment using simulated agents, which confirms the benefits of using interactive feedback to shape an RL-based agent's policy, under assumptions of ideal feedback.

Learning directly from actual users of the system helps to bridge the "reality gap" between simulations and real-world use. Having a simple formulation for the task-level reward function reduces the burden on the system designer when scaling the system to new domains, while the subtleties of natural interactions are encoded in a feedback model which is incrementally constructed by leveraging the "wisdom of the crowds". Interactive reinforcement learning of task-oriented dialogue management strikes a good balance between supervised and reinforcement learning by enabling faster learning with users and non-expert crowd workers.

## 6.1 Future work

The dialogue manager's policy $\pi_R$ can be modeled with other methods like actor-critic models (Sutton and Barto (1998)) where a DQN (Mnih et al. (2013)) is used to estimate the Q-value of each state to reduce variance in estimated rewards. Another extension is to use a decay function when calculating the feedback sum, $\delta_{s,a} = \Sigma_t \phi(t) f_{t,(s,a)}$, giving lower weight to older feedback, which will enable the agent to drift its behavior towards newer feedback. In this work, we assume that feedback is regular and consistent and we have not explored more sophisticated models of human-provided explicit and implicit feedback (Loftin et al. (2016)).

The feedback counts for each state-action pair are cached and used to compute the probabilities $P_F(a|s)$ to shape the agent's policy. With lots of feedback instances this data structure can become unwieldy over time. An extension would be to use the accumulated feedback as labelled examples and train a neural network model using supervised learning to estimate the feedback policy $\pi_F$ directly.

We have modeled feedback as a label on an individual state-action pair. However, it is easy for humans to give higher level feedback using natural language instructions which can generalize to more states (Goldwasser and Roth (2014)). This could be encoded as logical constraints on the system's behavior as a function of the dialogue context (eg. "If the user asks to confirm a slot, inform the value of that slot."). Assimilating these logic rules into the policy network (Hu et al. (2016)) and guiding the agent to pick actions within these constraints would result in the agent learning much faster from very few instances of feedback from the judge.

[1] It should also be noted that dialogues collected from human interactions tend to be noisy and would be expected to result in lower performance as compared to training from dialogues generated with a simulated rule-based expert agent.

## References


Biber, D. (1992). On the complexity of discourse complexity: A multidimensional analysis. *Discourse Processes*, 15(2):133–163.

Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., and Colton, S. (2012). A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43.

Cederborg, T., Grover, I., Isbell, C. L., and Thomaz, A. L. (2015). Policy shaping with human teachers. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*.

Cuayáhuitl, H. (2016). Simpleds: A simple deep reinforcement learning dialogue system. *arXiv preprint arXiv:1601.04574*.

Cuayáhuitl, H., Keizer, S., and Lemon, O. (2015). Strategic dialogue management via deep reinforcement learning. *arXiv preprint arXiv:1511.08099*.

Dhingra, B., Li, L., Li, X., Gao, J., Chen, Y.-N., Ahmed, F., and Deng, L. (2016). End-to-end reinforcement learning of dialogue agents for information access. *arXiv preprint arXiv:1609.00777*.

Fatemi, M., Asri, L. E., Schulz, H., He, J., and Suleman, K. (2016). Policy networks with two-stage training for dialogue systems. *arXiv preprint arXiv:1606.03152*.

Gašić, M., Breslin, C., Henderson, M., Kim, D., Szummer, M., Thomson, B., Tsiakoulis, P., and Young, S. (2013). On-line policy optimisation of bayesian spoken dialogue systems via human interaction. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8367–8371. IEEE.

Gašić, M., Henderson, M., Thomson, B., Tsiakoulis, P., and Young, S. (2012). Policy optimisation of pomdp-based dialogue systems without state space compression. In *Spoken Language Technology Workshop (SLT), 2012 IEEE*, pages 31–36. IEEE.

Gašić, M., Jurčíček, F., Keizer, S., Mairesse, F., Thomson, B., Yu, K., and Young, S. (2010). Gaussian processes for fast policy optimisation of pomdp-based dialogue managers. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 201–204. Association for Computational Linguistics.

Goldwasser, D. and Roth, D. (2014). Learning from natural instructions. *Machine learning*, 94(2):205–232.

Griffith, S., Subramanian, K., Scholz, J., Isbell, C., and Thomaz, A. L. (2013). Policy shaping: Integrating human feedback with reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2625–2633.

Grosz, B. J. and Sidner, C. L. (1986). Attention, intentions, and the structure of discourse. *Computational linguistics*, 12(3):175–204.

Hu, Z., Ma, X., Liu, Z., Hovy, E., and Xing, E. (2016). Harnessing deep neural networks with logic rules. *arXiv preprint arXiv:1603.06318*.

Jørgensen, M. W. and Phillips, L. J. (2002). *Discourse analysis as theory and method*. Sage.

Knox, W. B., Glass, B. D., Love, B. C., Maddox, W. T., and Stone, P. (2012). How humans teach agents. *International Journal of Social Robotics*, 4(4):409–421.

Knox, W. B. and Stone, P. (2012). Reinforcement learning from simultaneous human and mdp reward. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 475–482. International Foundation for Autonomous Agents and Multiagent Systems.

Koos, S., Mouret, J.-B., and Doncieux, S. (2010). Crossing the reality gap in evolutionary robotics by promoting transferable controllers. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 119–126. ACM.

Lai, M. (2015). Giraffe: Using deep reinforcement learning to play chess. *arXiv preprint arXiv:1509.01549*.

Levin, E. and Pieraccini, R. (1997). A stochastic model of computer-human interaction for learning dialogue strategies. In *Eurospeech*, volume 97, pages 1883–1886.

Loftin, R., Peng, B., MacGlashan, J., Littman, M. L., Taylor, M. E., Huang, J., and Roberts, D. L. (2016). Learning behaviors via human-delivered discrete feedback: modeling implicit feedback strategies to speed up learning. *Autonomous Agents and Multi-Agent Systems*, 30(1):30–59.

Loftin, R. T., MacGlashan, J., Peng, B., Taylor, M. E., Littman, M. L., Huang, J., and Roberts, D. L. (2014). A strategy-aware technique for learning behaviors from discrete human feedback. In *AAAI*, pages 937–943.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Paltridge, B. (2012). *Discourse analysis: An introduction*. Bloomsbury Publishing.

Pietquin, O. (2013). Inverse reinforcement learning for interactive systems. In *Proceedings of the 2nd Workshop on Machine Learning for Interactive Systems: Bridging the Gap Between Perception, Action and Communication*, pages 71–75. ACM.

Schatzmann, J., Thomson, B., Weilhammer, K., Ye, H., and Young, S. (2007). Agenda-based user simulation for bootstrapping a pomdp dialogue system. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 149–152. Association for Computational Linguistics.

Schatzmann, J. and Young, S. (2009). The hidden agenda user simulation model. *IEEE transactions on audio, speech, and language processing*, 17(4):733–747.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

Su, P.-H., Gasic, M., Mrksic, N., Rojas-Barahona, L., Ultes, S., Vandyke, D., Wen, T.-H., and Young, S. (2016a). Continuously learning neural dialogue management. *arXiv preprint arXiv:1606.02689*.

Su, P.-H., Gasic, M., Mrksic, N., Rojas-Barahona, L., Ultes, S., Vandyke, D., Wen, T.-H., and Young, S. (2016b). On-line active reward learning for policy optimisation in spoken dialogue systems. *arXiv preprint arXiv:1605.07669*.

Suay, H. B. and Chernova, S. (2011). Effect of human guidance and state space size on interactive reinforcement learning. In *2011 Ro-Man*, pages 1–6. IEEE.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

Thomaz, A. L., Hoffman, G., and Breazeal, C. (2005). Real-time interactive reinforcement learning for robots. In *AAAI 2005 workshop on human comprehensible machine learning*.

Thomaz, A. L., Hoffman, G., and Breazeal, C. (2006). Reinforcement learning with human teachers: Understanding how people want to teach robots. In *ROMAN 2006-The 15th IEEE International Symposium on Robot and Human Interactive Communication*, pages 352–357. IEEE.

Wen, T.-H., Gasic, M., Mrksic, N., Rojas-Barahona, L. M., Su, P.-H., Ultes, S., Vandyke, D., and Young, S. (2016). A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562*.

Williams, J. (2006). *Partially Observable Markov Decision Processes for Spoken Dialogue Management*. PhD thesis, Churchill College and Cambridge University Engineering Department.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256.

Young, S. (2006). Using pomdps for dialog management. In *2006 IEEE Spoken Language Technology Workshop*, pages 8–13. IEEE.

Zhao, T. and Eskenazi, M. (2016). Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. *arXiv preprint arXiv:1606.02560*.

## A: Dialogue acts

| **Action** | **Description** |
|---|---|
| greeting | Greet the system. |
| inform | Inform the system of a slot value from the user's goal. |
| request | Request a slot value of a restaurant. |
| confirm | Ask the system to confirm a slot value of a restaurant. |
| affirm | Agree to the system's offer. |
| negate | Deny a system's offer. |
| thank_you | Thank the system. |

Table 2: User dialogue actions.

| **Action** | **Description** |
|---|---|
| request | Request a slot's value from the user. |
| offer | Offer a single restaurant to the user. |
| select | Ask the user to choose from two or more restaurants. |
| inform | Inform the user of a slot value of a restaurant. |
| nomatch | Notify the user that no restaurants match the specified constraints. |

Table 3: System dialogue actions.