The Exploration/Exploitation Trade-off in Reinforcement Learning for Dialogue Management

Sebastian Varges, Giuseppe Riccardi, Silvia Quarteroni, Alexei V. Ivanov
Department of Information Engineering and Computer Science
University of Trento
38050 Povo di Trento, Italy
{varges|riccardi|silviaq|ivanov}@disi.unitn.it

***Abstract*—Conversational systems use deterministic rules that trigger actions such as requests for confirmation or clarification. More recently, Reinforcement Learning and (Partially Observable) Markov Decision Processes have been proposed for this task. In this paper, we investigate action selection strategies for dialogue management, in particular the exploration/exploitation trade-off and its impact on final reward (i.e. the session reward after optimization has ended) and lifetime reward (i.e. the overall reward accumulated over the learner's lifetime). We propose to use *interleaved exploitation* sessions as a learning methodology to assess the reward obtained from the current policy. The experiments show a statistically significant difference in final reward of exploitation-only sessions between a system that optimizes lifetime reward and one that maximizes the reward of the final policy.**

I. Introduction

In recent years, Machine Learning techniques, in particular Reinforcement Learning (RL), have been applied to the task of dialogue management (DM) [1], [2], [3]. A major motivation is to improve robustness in the face of uncertainty, for example due to speech recognition errors. A further motivation is to improve adaptivity w.r.t. different noise levels/recognition environments, user behaviour and application contexts. The Reinforcement Learning framework [4] is attractive because it offers a statistical model representing the dynamics of the interaction between system and user. This is in contrast to the supervised learning approach of learning system behaviour based on annotated corpus [5], [6]. To explore the range of dialogue management strategies, a simulation environment is required that includes a simulated user [7] if one wants to avoid the prohibitive cost of using human subjects.

In this work, we investigate several aspects of action selection in RL for dialogue management. We use a domain for which we also built a more conventional dialogue manager to obtain, for example, an action set, a set of possible user responses, and dialogue ending options that are realistic and appropriate for a working SDS. We investigate the influence of the action selection strategy on learning of dialogue management policies. This issue, although studied in the general context of Machine Learning [8], has received little attention in RL for dialogue management. The action selection strategy is a key factor for determining the *exploration/exploitation trade-off*: to optimize the long-term reward and populate its policy with expected values, the learner needs to explore untried actions to gain more experience, and combine this with exploitation of the already known successful actions to also ensure high reward. A Reinforcement Learner trades off short-term rewards, i.e. rewards received at the end of each dialogue session, against long-term rewards in the form of values. There is no distinction between training and testing since the learner receives rewards in all sessions. However, we argue that there is a distinction between lifetime reward, i.e. the overall reward accumulated over the learner's lifetime, and the final short-term session reward obtained when exploiting the policy after optimization has ended.[1] The distinction of life-time vs final short-term reward is directly relevant to the choice of optimization settings of the RL system for simulated vs human user interaction. To measure a learner's progress, we propose to use interleaved exploitation sessions that expose the reward obtained from the current policy, and we investigate the effect of these sessions if they are used to update the policy.

This paper is structured as follows: after a brief introduction to RL for dialogue management (sec. II), we describe our RL-DM (sec. III), including state representation, action set, reward function and user simulation. After briefly describing the more conventional SDS that was used for a data collection (sec. IV), we describe interleaved exploitation (sec. V) and report on a series of simulation experiments (sec. VI), including the exploration/exploitation trade-off (sec. VI-A) and basic search properties of the RL learner (sec. VI-B). We conclude this paper with a discussion in section VII.

II. Reinforcement Learning for Dialogue Management

Dialogue management is modeled as taking a decision about an action a at time t in a dialogue state s_t given an observation o of the user input, and receiving a (possibly delayed) reward r. State s is the information state of the dialogue system with an explicit Markov assumption about the context window. In a Markov Decision Process (MDP), no uncertainty about the user input is assumed. This user input is

[1]Lifetime reward and values both represent long-term aspects of learning with rewards but are quite different in nature. For example, values pertain to the expected reward of state-action mappings (see also section II) whereas the lifetime reward simply sums over all sessions.

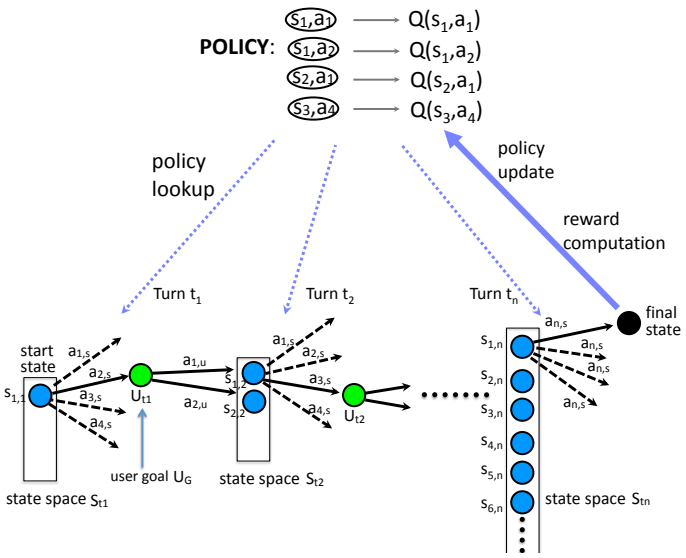

Fig. 1. MDP Dialogue Manager

assumed to unambiguously refer to the system state, i.e. only a single (dialogue) state hypothesis is maintained. In a Partially Observable Markov Decision Process (POMDP), ambiguity in the user input is taken into account by accepting an N-best 'list' of Spoken Language Understanding (SLU) results, i.e. concept-value pairs with their confidences. This yields an (often large) number of parallel state hypotheses.

The RL-DM maintains an internal data structure called a *policy* to keep track of the values (accumulated rewards) of past state-action pairs. The goal of the learner is to optimize the *long-term* reward by maximizing the 'Q-Value' $Q^\pi(s_t, a)$ of a policy π for taking action a at time t. In a POMDP, the policy consists of a mapping from distributions over states to actions. Since an analytic solution to finding optimal Q-Values is not possible for realistic dialogue scenarios, $Q^\pi(s_t, a)$ is estimated by dialogue simulations.

III. RL-DM for a real-world task

The RL-DM implemented in this work is shown in figure 1: at each turn at time t, the incoming N user act hypotheses $a_{n,u}$ split the state space S_t to represent the complete set of interpretations from the start state (N=2). A belief update is performed resulting in a probability assigned to each state. The resulting ranked state space S_t is used as a basis for action selection. In our current implementation, belief update is based on probabilistic user responses that include SLU confidences (see below). Action selection to determine system action $a_{m,s}$ is based on the best state (m is a counter for actions in action set A; see section III-B). In each turn, the system uses an ϵ-greedy action selection strategy to decide probabilistically if to exploit the policy or explore any other action at random. (An alternative would be softmax, for example.) At the end of each dialogue/session a reward is assigned (section III-C) and policy entries are added or updated for each state-action pair involved. These are stored in tabular form. The conditioning of action decisions on individual states at the policy level implies that the implemented RL-DM is an MDP. The ranked state space is a step toward handling uncertainty (see [9], for example). We perform Monte Carlo updating similar to [1]:

$$Q_t(s,a) = R(s,a)/n + Q_{t-1} \cdot (n-1)/n, \qquad (1)$$

where n is the number of sessions, R the reward and Q the estimate of the state-action value.

At the beginning of each dialogue, a user goal U_G (a set of concept-value pairs) is generated randomly and passed to a user simulator. The user simulator takes U_G and the current dialogue context to produce plausible SLU hypotheses. These are a subset of the concept-value pairs in U_G along with a confidence estimate bootstrapped from a small corpus of 74 in-domain dialogs. We assume that the user 'runs out of patience' after 15 turns and ends the call.

A. State Representation

The application domain is a tourist information system for accommodation and events in the local area. The domain of the trained DMs is identical to that of a rule-based DM that was used for a data collection with human users (section IV-B), allowing us to compare the two directly.

The state of the RL-DM keeps track of the SLU hypotheses in the form of domain concepts. There are 8 of these in the application domain: main activity, location, star rating of hotels, duration, start and end day and month. Furthermore, there are 2 'pseudo-concepts' for user-initiated acts: transfer to the operator and dialogue ending. The state representation of the DM keeps track of the actual values. In policy space, we abstract concept values into `'KNOWN/UNKNOWN'`, thus increasing the likelihood that the system re-visits a dialogue state which it can exploit. The actual policy state space is thus of size $2^8 = 256$.

B. Action set

The system selects actions that are *relative to the current belief state*. We distinguish 3 types of actions (size of action set $|A| = 26$ overall): concept questions, clarification questions, and non-linguistic actions such as database queries. There are more concept questions than domain concepts: a question for the start date expects both a month and a day concept from the user. Alternatively, these concepts can be obtained by separate `question-start-day` and `question-start-month` actions. (The tutorial day-month dialogue of [1] is thus a subtask of our domain). For each action, we specify the expected concepts that the DM is able to recognize at the next turn according to a domain ontology. This is motivated by the Automatic Speech Recognition (ASR) of the data collection system (section IV-A). Hence, the probability of any other concept occurring is 0, regardless of the actual user simulator (section III-D).

Clarification questions verify a specific concept value. Since in many policy state representations these values will be `UNKNOWN`, the question arises if clarification questions should be possible in these cases. One option is to constrain the set of

possible actions (given the current best state) manually. This results in a hybrid approach that integrates rule-based knowledge with RL learning [10]. In the start state, for example, the system may only have the opening prompt (e.g. *'How May I help You?'*, HMIHY) available. This effectively constrains the search space of RL-learning since there are fewer actions to explore. However, it also introduces additional (and potentially error prone) manually provided knowledge into the learning process. In this work, we explore an alternative: we always make clarification actions available. This is possible without the need for special-purpose rules since the user simulator answers automatically negatively if an `UNKNOWN` value is verified but the user goal actually contains a concept value.

Informed by the action set of the data collection system (section IV), we define three actions that result in an 'end state': passing the call to an operator, ending the call system-side, and performing a database lookup.

C. Reward Function

The reward function has a crucial influence on learning. A key ingredient of any *objective function* is the correctness of the acquired concept values. Furthermore, the interaction cost, often measured by the number of turns, needs to be taken into account. Additionally, we consider the cost of ending the dialogue session. Reward R is thus defined as

$$R = w_1 M - w_2 S - w_3 I - w_4 E, \tag{2}$$

where M is the number of matches and S the number of mismatches between user goal and DM state representation. Note that we need to compare actual values rather than the abstracted ones of policy space. I is the interaction cost and E the cost of the specific dialog ending. Defining a reward function and assigning empirically-based weights is a research topic in its own right [11]. Based on our experiences, we amplify the benefits of matches by a factor of 10 and assign a database lookup a cost of 5, operator transfer a cost of 10 and a hangup carries a cost of 20. Hangups, i.e. sudden dialogue endings, can occur as a result of a system action (section III-B) and also as a result of a user hangup, which is defined by the user model (section III-D). Thus, an end state can be reached by 3 system actions and 1 user action overall. An optimal session for a hotel enquiry receives a reward of 73.5. (The cost function is sufficiently simple to calculate this manually.)

D. User Simulation

In order to conduct thousands of simulated dialogues, the DM needs to deal with heterogeneous but plausible user input. For this purpose, we have designed a User Simulator (US) which bootstraps likely user behaviors starting from a small corpus of 74 in-domain dialogs, acquired using the rule-based version of the SDS (section IV). The task of the US is to simulate the output of the SLU module to the DM, hence providing it with a ranked list of SLU hypotheses. At each turn, the US mines the previous system dialog act to obtain the concepts required by the DM and obtains the corresponding values (if any) from the current user goal.

The output of the user model is passed to an error model that simulates "noisy channel" recognition errors based on statistics from the dialogue corpus. These affect concept values as well as other dialogue phenomena such as $NoInput$, $NoMatch$ and $HangUp$. If the latter phenomena occur, they are propagated to the DM directly; otherwise, the following US step is to attach plausible confidences to concept-value pairs, also based on the dialogue corpus. Finally, the concept-value pairs are combined in an SLU hypothesis and, as in the regular SLU module, a cumulative utterance-level confidence is computed, determining the rank of each of the n hypotheses. The probability of a given concept-value observation o_{t+1} at time $t+1$, given the chosen system act at time t, $a_{s,t}$ and the known session user goal g_u, is obtained by combining the error model and the user model:

$$P(o_{t+1}|a_{s,t}, g_u) = P(o_{t+1}|a_{u,t+1}) \cdot P(a_{u,t+1}|a_{s,t}, g_u),$$

where $a_{u,t+1}$ is the true user action. However, in the presented experiments (section VI) we do not use an error model. The probabilistic user model is used to weigh the selection of the user's response. The user goal determined by the simulation environment is used to ensure consistent responses from the simulated user, i.e. repeated questions for the same concept yield identical values.

IV. Data Collection Baseline SDS

We also developed a more conventional SDS for interaction with human users to obtain training data from human-system interaction.

A. ASR and SLU

The data collection system uses a VXML platform as an interface to ASR and TTS functionality. Speech recognition in this version of the system is grammar-based; dialogue manager actions – both of the RL-DM and the rule-based DM (section IV-B) – are associated with specific grammars.

The SLU module receives the top 10 ASR interpretations as produced by the VXML grammars and is responsible for communicating a ranked list of the top 2 interpretations to the DM. As VXML grammars used for ASR already recognize concepts, the objective of the SLU module is to normalize concept values (e.g. numbers) and to estimate concept-level confidences. These are computed as the average ASR confidence of the words underlying each concept; the final ranking in the SLU output depends on an utterance-level confidence score which is in turn the average concept-level confidence for all the concepts in the current interpretation.

B. Rule-based Dialogue Management

The rule-based dialogue manager works in two stages: retrieving and preprocessing facts (tuples) taken from a dialogue state database, and inferencing over those facts to generate a system response. We distinguish between the 'context model' of the first phase – essentially allowing more recent values for a concept to override less recent ones – and the 'dialog move engine' of the second phase. In the second stage, acceptor

rules match SLU results to dialogue context, for example perceived user concepts to open questions. This may result in the decision to verify the application parameter in question, and the action is verbalized by language generation rules. If the parameter is accepted, application dependent task rules determine the next parameter to be acquired, resulting in the generation of an appropriate request.

The system architecture and further details are described in [12].

V. Interleaving Exploration and Exploitation

Action selection in Reinforcement Learning can either try to exploit the policy to carry out the currently believed best action, or explore untried actions w.r.t. the current belief state. This choice is determined by the *action selection strategy* which is thus crucial for determining the exploration/exploitation trade-off. Generally, a certain amount of exploration always takes place: even if the system decides to exploit, if the current state has not been visited before, the system has no choice but to fall back on exploration. In this work, we use an ϵ-greedy action selection strategy.

To measure a learner's progress, we propose to use interleaved exploitation sessions that expose the reward obtained from the current policy, and contrast this with the reward obtained from those sessions that mix exploration and exploitation. We thus obtain three reward measures overall for each simulation experiment: the average reward of all sessions, and separate reward averages for exploitation-only sessions and mixed exploration/exploitation sessions. This will be shown in section VI. The exploitation-only sessions can serve to estimate the reward obtained with human user interaction if simulation were to stop *at this point*, for example. The mixed exploration/exploitation sessions show how the learner performs while populating the policy with values – the rewards of these sessions could be important to decide if human subjects should be involved, for example. There is also an option to update the policy during exploitation-only sessions. If no updating is performed, the interleaved exploitation-only sessions are purely for observational purposes and do not alter the value function.

Every RL system obviously has to balance exploration and exploitation. However, the previous work we are aware of usually presents single reward measures, often in the form of average rewards, e.g. [8], [13]. In many cases, previous work does not elaborate on the chosen exploration/exploitation trade-off although this has a high impact on learning. We thus cautiously suggest that separating rewards to expose different aspects of learning is a novel experimental method.

VI. Simulation Experiments

We conducted several simulation experiments of 10 simulation 'runs' each. For each run, 1000 dialogue sessions were conducted. For each dialogue, system-user interactions are simulated up to maximally the user patience level, i.e. 15 turn pairs. We investigated both the search behavior of the learner and the resulting dialogue strategies. The experiments employed the RL-DM described in section III.

A. The exploration/exploitation trade-off

The parameter we investigated is the impact of the action selection strategy, more precisely the probabilistic exploitation rate in ϵ-greedy action selection, for ϵ= 0.0, 0.01, 0.2, 0.4, 0.7, 1.0.[2]

Figure 2 shows the average rewards for 10 simulation runs of 1000 sessions each, with $\epsilon = 0.7$ (left figure) and $\epsilon = 0.2$ (right figure). Each figure shows three reward lines that belong *to the same system*: each dot on the bold line 'a)' shows the average reward for 100 sessions (e.g. session 1-100, 101-200 etc.) in 10 runs, i.e. it averages 1000 rewards.

To measure the reward of the current policy at different time points, we switched to exploitation-only sessions at every 10th session. Lines labeled 'b)' show the reward of these interleaved exploitation sessions: this is the best action choice the learner can make at a given session number (10th, 20th, 30th etc). Each dot is the average of 100 rewards (10 rewards for each window of 100 sessions, for 10 runs).

Lines labeled 'c)' show the rewards during ϵ-search, i.e. while populating the policy. Each dot represents the average of 900 rewards, for example for session numbers 1-9, 11-19, 21-29, etc. (90 rewards for each window of 100 sessions, for 10 runs). The vertical lines are error bars that indicate the 95% confidence interval.

As figure 2 shows, the overall reward increases for both experiments (bold lines). However, as the interleaved exploitation sessions show, with $\epsilon = 0.7$ learning is faster and a higher final exploitation reward is achieved. On the other hand, the overall average reward for $\epsilon = 0.2$ is larger. The same is true for the mixed exploration/exploitation sessions used to populate the policy. The overall reward is closer to the mixed session line because it contains a larger component of mixed session rewards than exploitation-only rewards.

Table I shows average final rewards of exploitation sessions ('$\bar{R}$ Final'), percentage of optimal strategy found ('% Opt. Final'), and the overall average lifetime reward per session ('$\bar{R}$ Lifetime') and overall ('$\sum R$ Lifetime') excluding exploitation sessions for two types of simulations: one that uses the interleaved exploitation sessions for updating the policy (table I, left), and one that skips updating (table I, right). In the former case, the actual exploration rate is reduced, e.g. to $0.9 \cdot 0.7 = 0.63$ for ϵ=0.7.

Table I shows that without exploration ($\epsilon = 0.0$) average and lifetime rewards remain low. Using only exploration ($\epsilon = 1.0$) yields low lifetime reward and is not likely to result in optimal final policies. In between the extremes, there seems to be a range of exploration rates for which high final rewards can be achieved. However, while $\epsilon = 0.7$ makes finding optimal final rewards likely (see also figure 2 left), it yields relatively low lifetime reward. In contrast, $\epsilon = 0.2$ achieves maximum

[2]The value of 0.01 was chosen because in a pilot experiment very little exploration proved sufficient.

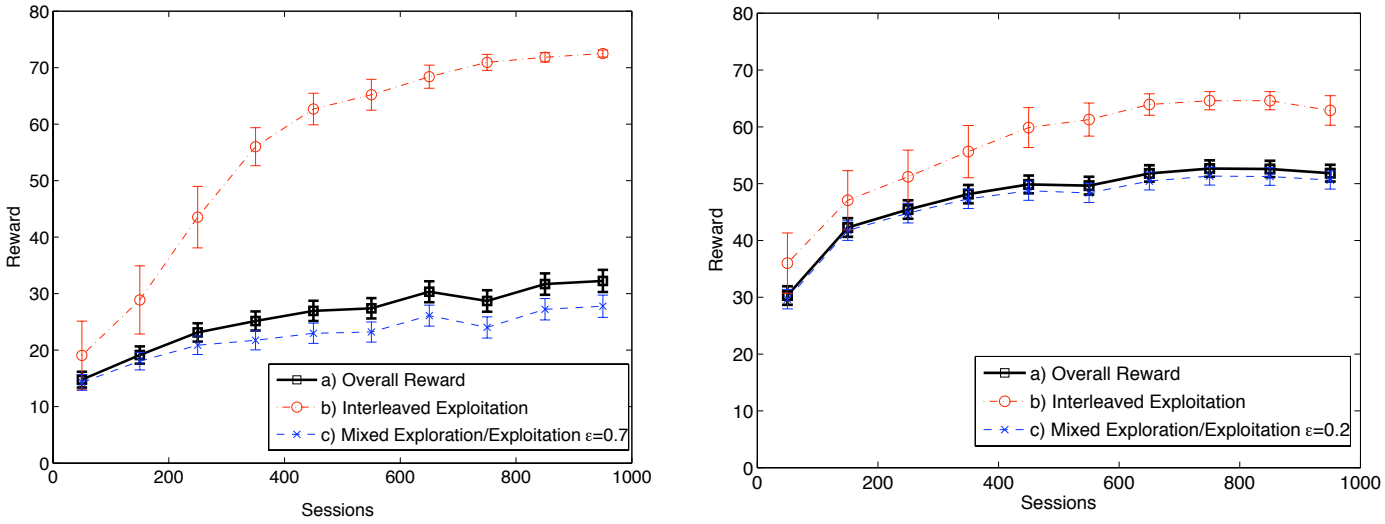

Fig. 2. Exploration/exploitation trade-off for simulated dialogue sessions for RL systems with $\epsilon = 0.7$ (LEFT) and $\epsilon = 0.2$ (RIGHT). Bold lines with squares 'a)' show the overall average reward of 10 simulation runs. Lines with circles 'b)' show average reward of interleaved exploitation sessions only. Lines with stars 'c)' show average reward for mixed exploration/exploitation sessions only. The reward curves 'b)' and 'c)' in each figure represent different components of the average reward 'a)' of the RL system.

TABLE I
SIMULATION RESULTS FOR DIFFERENT EXPLORATION RATES (ϵ) IN ϵ-GREEDY ACTION SELECTION: WITH POLICY UPDATING AFTER INTERLEAVED EXPLOITATION SESSIONS (LEFT) AND WITHOUT (RIGHT).

	INTERLEAVED WITH UPDATING				INTERLEAVED WITHOUT UPDATING			
ϵ	R Final	% Opt. Final	R Lifetime	$\sum R$ Lifetime	R Final	% Opt. Final	R Lifetime	$\sum R$ Lifetime
0.0	8.58	0.0	8.51	76631.3	3.40	0.0	3.39	30502.0
0.01	58.65	10.0	41.83	376446.3	52.33	10.0	32.97	296693.0
0.2	64.60	10.0	**46.43**	**417902.0**	70.65	30.0	**46.73**	**420512.3**
0.4	69.03	20.0	39.18	352608.0	69.05	50.0	35.29	317604.0
0.7	**73.43**	**70.0**	22.63	203661.5	**71.35**	**80.0**	22.30	200725.0
1.0	58.68	0.0	6.55	58926.5	52.33	10.0	32.97	296693.0

lifetime reward in our experiments but does not achieve the highest rewards in exploitation-only sessions (see also figure 2 right). As the confidence intervals in figure 2 show, the difference between the final exploitation-only rewards (lines 'b') between the two systems is statistically significant at the 95% confidence interval (see also the discussion in section VII).

B. Non-monotonicity of reward increase

In many simulation runs, we observe what appear to be 'glitches' in the otherwise largely increasing reward, i.e. the reward is not strictly monotonically increasing. This is an important property to establish for any search strategy. Upon closer inspection, the low-reward sessions turn out to repeat actions or action sequences until the user patience limit. However, because of the low reward received, the glitches are immediately corrected, so that the maximal reward returns to the previous value at the next exploitation session. We find examples of such temporary glitches in several simulation runs. They are characterized by repeated actions or action sequences without reaching a dialogue end state. The experimental results suggest that the use of exploitation may actually help improve the current policy: comparing the rewards for $\epsilon = 1.0$ in table I, i.e. 100% exploration, we find that using interleaved exploitation sessions for updating increases the final reward. However, it seems to reduce the overall lifetime reward, and the effect cannot be observed for all settings of ϵ.

The correcting effect of exploitation should always be available since for all simulations with $\epsilon < 1.0$ a certain amount of exploitation takes place. However, in normal operation the choice of exploration vs exploitation is made for each system turn rather than the entire dialogue. Thus, it seems likely that the effect is stronger in exploitation-only sessions.

C. Resulting dialogue strategies

We now look at the actually learnt dialogue strategies as evidenced by exploitation sessions at the end of simulations. We observe that even under the same experimental conditions, different final policies are learnt. For example, for $\epsilon = 0.40$ we obtain the following the action sequence in the final exploitation sessions:

```
(1)  QUESTION-HMIHY
     QUESTION-START-DATE
     QUESTION-END-DATE
     QUESTION-STARRATING
     QUESTION-DURATION
     DATABASE-LOOKUP
```

This sequence yields optimal reward given the state representation and reward function. Each question is only asked once, and questions that return two concepts rather than one are preferred: the HMIHY question wins over individual activity and location questions, and questions for dates win over individual day and month questions. The dialogue ends with a database query that contains all required concepts. The system avoids clarification actions since these do not contribute to higher reward in this experiment; we do not yet use the error model of the user simulator that would make the use of clarifications beneficial.

It should be noted that a dialogue such as (1) is the result of interaction with the simulator user, and thus not the only dialogue the policy can conduct. Inspecting the final policies, we find that in a state in which only the location is known, for example, the highest ranked action is typically the question for activity, or at least a HMIHY question (which also expects an activity concept). This is in fact the desired behaviour, which we previously programmed manually into the rule-based DM (section IV-B): we found that in the data collection experiments with human users, for the opening HMIHY question often only the location was recognized, requiring an additional question for activity (*'Please tell me what you would like to do in [LOCATION]'*).

VII. Discussion and conclusions

Regarding the exploration/exploitation trade-off (section VI-A) we show that the reward of the final policy and the overall lifetime reward are not necessarily optimized simultaneously. There is a statistically significant difference in the final exploitation-only reward between the two. Given this, we suggest the following methodology: simulations should be conducted with a relatively high exploration rate to find a policy that maximizes the reward of the resulting policy. Interaction with human users, in contrast, should optimize the lifetime reward since this measure directly relates to the user experience. In our system, we can use policies trained in simulation for real user interaction with further adaptation, i.e. policy updating.

More research is needed on setting the exploration rate in different applications. In a different set of experiments with a comparable state size (417 vs 256 here) but considerably smaller action set (4 vs 26 here), an exploration rate of 1% proved sufficient to quickly find an optimal policy. This suggests that the number of state-action pairs in the policy may be used as an estimate of the complexity of the learning task, i.e. $|S| \cdot |A|$, rather than the size of state set $|S|$ alone. It also suggests that more exploration is needed in more complex domains.

The policy's state representation has an obvious impact on learning. Here, we abstracted concept values into `'KNOWN/UNKNOWN'`. This still does not result in exploring all possible states in the experiments: the average number of policy state entries for $\epsilon = 0.6$ is 140.5 out of 256 possible states, for example. However, these are obviously states that were not visited during the simulation, and they may be unlikely to occur for a given domain and user model.

The RL-DM learns to ask each question only once, and to prefer questions that return several concepts (sections VI-C). In future research we will employ error models in the user simulations to learn using clarification questions, confidence thresholds, and reranking of the state space. Furthermore, we will extend the RL-DM implementation to map complete belief states to actions rather than 'best' states and conduct user evaluations.

Acknowledgment

This work was partially supported by the European Commission Marie Curie Excellence Grant for the ADAMACH project (contract No. 022593) and by LUNA STREP project (contract No. 33549).

References

[1] E. Levin, R. Pieraccini, and W. Eckert, "A stochastic model of human-machine interaction for learning dialog strategies," *IEEE Transactions on Speech and Audio Processing*, vol. 8, no. 1, 2000.

[2] J. D. Williams and S. Young, "Partially Observable Markov Decision Processes for Spoken Dialog Systems," *Computer Speech and Language*, vol. 21, no. 2, pp. 393–422, 2006.

[3] S. Young, M. Gasic, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson, and K. Yu, "The Hidden Information State Model: a practical framework for POMDP-based spoken dialogue management." *Computer Speech and Language, To appear*, 2009.

[4] R. S. Sutton and A. G. Barto, *Reinforcement Learning. An Introduction*. MIT Press, 1998.

[5] R. Higashinaka, M. Nakano, and K. Aikawa, "Corpus-based discourse understanding in spoken dialogue systems," in *Proc. of ACL*, 2003.

[6] D. Griol, L. F. Hurtado, E. Segarra, and E. Sanchis, "A statistical approach to spoken dialog systems design and evaluation," *Speech Communication*, vol. 50, no. 8-9, pp. 666–682, 2008.

[7] 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, no. 2, pp. 97–126, 2006.

[8] S. Whiteson, M. E. Taylor, and P. Stone, "Empirical studies in action selection with reinforcement learning," *Adaptive Behavior - Animals, Animats, Software Agents, Robots, Adaptive Systems*, pp. 33–50, 2007.

[9] J. Henderson and O. Lemon, "Mixture Model POMDPs for Efficient Handling of Uncertainty in Dialogue Management," in *Proc. of the 46th Annual Meeting of the Association for Computational Linguistics (ACL/HLT)*, Columbus, Ohio, 2008.

[10] J. D. Williams, "The best of both worlds: Unifying conventional dialog systems and POMDPs," in *Intl. Conf on Speech and Language Processing (ICSLP)*, 2008.

[11] V. Rieser and O. Lemon, "Learning Effective Multimodal Dialogue Strategies from Wizard-of-Oz data: Bootstrapping and Evaluation," in *Proc. ACL/HLT*, 2008.

[12] S. Varges and G. Riccardi, "A Data-centric Architecture for Data-driven Spoken Dialogue Systems," in *Proceedings of IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, Kyoto, Japan, 2007.

[13] J. D. Williams, "Applying POMDPs to Dialog Systems in the Troubleshooting Domain," in *Proceedings of the HLT/NAACL Workshop on Bridging the Gap: Academic and Industrial Research in Dialog Technology*, Rochester, NY, USA, 2007.