INTERSPEECH 2009 BRIGHTON

A Policy-Switching Learning Approach for Adaptive Spoken Dialogue Agents

Heriberto Cuayáhuitl, Juventino Montiel-Hernández

Intelligent Systems Research Group, Autonomous University of Tlaxcala
Apartado Postal #140, Apizaco, Tlaxcala, Mexico, 90300
{h.cuayahuitl, j.montiel74}@gmail.com

Abstract

The reinforcement learning paradigm has been adopted for inferring optimized and adaptive spoken dialogue agents. Such agents are typically learnt and tested without combining competing agents that may yield better performance at some points in the conversation. This paper presents an approach that learns dialogue behaviour from competing agents—switching from one policy to another competing one—on a previously proposed hierarchical learning framework. This policy-switching approach was investigated using a simulated flight booking dialogue system based on different types of information request. Experimental results reported that the induced agent using the proposed policy-switching approach yielded 8.2% fewer system actions than three baselines with a fixed type of information request. This result suggests that the proposed approach is useful for learning adaptive and scalable spoken dialogue agents.

Index Terms: spoken dialogue systems, adaptive behaviours, hierarchical reinforcement learning, policy-switching.

1. Introduction

Spoken dialogue systems use a dialogue strategy (also referred to as 'dialogue policy', 'dialogue controller', or 'dialogue agent') to control their dialogue behaviour in human-machine conversations. Such dialogue agents typically employ static behaviour that do not adapt to the conversational environment such as the user or the speech recognition and understanding component. Previous investigations have argued and tested experimentally that adaptive spoken dialogue behaviours produce better behaviours (i.e., more efficient and successful dialogues, and with higher user satisfaction) than non-adaptive ones [1, 2].

Adaptive spoken dialogue agents can be specified with three main approaches: (1) hand-crafted behaviour, (2) supervised learning, and (3) reinforcement learning. Firstly, hand-crafted behaviour can use heuristics to alternate from different types of behaviour, but it may not be easy to specify and lack of optimization. Secondly, supervised approaches use the following iterative process during the course of the dialogue: (a) monitor dialogue features, (b) predict whether the dialogue was problematic, and (c) start the dialogue with more natural interactions but adapt to more conservative dialogue strategies if the dialogue was classified as problematic [1, 2]. Thirdly, reinforcement learning aims to provide with optimal behaviour—lacked by the previous approaches—by inferring a policy performing action selection that is expected to maximize its performance function for each perceived dialogue state [3]. This latter approach has been identified as more promising for learning adaptive and optimized spoken dialogue behaviour.

The research gap that we address in this paper is that reinforcement learning dialogue agents typically infer behaviour without combining the behaviour from multiple competing agents[1]. This is attractive if we assume that choosing the optimal action for each dialogue state from a set of competing dialogue agents may yield better performance rather than using a single dialogue agent for the whole conversational interaction. To the best of our knowledge this idea has not yet been applied to reinforcement learning dialogue agents. In the rest of this paper we describe a policy-switching approach for such a purpose.

2. Using reinforcement learning for optimized spoken dialogue control

The dynamics of a reinforcement learning dialogue agent can be defined as follows: (1) the machine is in a given state s_t at time t, (2) the dialogue agent chooses an action a_t given the current state by following policy $\pi(s_t)$, (3) the action is executed in a real or simulated conversational environment in order to observe the next state s_{t+1} and an immediate reward r_{t+1} for executing a_t in s_t, and (4) the dialogue agent receives the new state and reward in order to compute a cumulative discounted reward for each state or state-action pair that specifies what is good in the long run. A conversation follows such sequence of interactions in an iterative process between both conversants until one of them terminates it at T time steps, which can be expressed as dialogue $D = \{s_1, a_1, r_2, s_2, a_2, r_3, ..., s_{T-1}, a_{T-1}, r_T, s_T\}$. Although real human-machine conversations can be used for optimizing spoken dialogue behaviour, a more common practice is to use simulations due to the large number of dialogues required to learn optimal policies.

A reinforcement learning dialogue agent aims to learn its behaviour from interaction with an environment, where situations are mapped to actions by maximizing a long-term reward signal. The reinforcement learning paradigm works by using the formalism of Markov Decision Processes (MDPs), or generalizations from it. An MDP is characterized by a set of states S, a set of actions A, a state transition function, and a performance function that rewards the agent for each selected action. Solving the MDP means finding a mapping from states to actions corresponding to $\pi^*(s_t) = \arg\max_{a_t \in A} Q^*(s_t, a_t)$, where the Q function specifies the cumulative rewards for each state-action pair. The optimal policy π^* can be found using reinforcement learning algorithms such as Q-learning or SARSA [4].

Because the flat tabular reinforcement learning framework (using a single policy) lacks scalability, in this paper we used a hierarchical reinforcement learning framework that decomposes an MDP into a hierarchy of Semi-Markov decision processes (representing sub-dialogues), which has been applied before to spoken dialogue agents with large state spaces [5].

[1] In this paper we assume that competing spoken dialogue agents have the same set of dialogue states and equivalent actions (e.g., with different prompts). Such competing agents reach the same goal state(s).

3. Hierarchical reinforcement learning using a policy-switching approach

3.1. Dialogue control using hierarchical SMDPs

Spoken dialogue control has been treated as a hierarchy of discrete Semi-Markov Decision Processes (SMDPs) in order to address the problem of scalable dialogue optimization [5]. In this way an MDP can be decomposed into multiple SMDPs hierarchically organized into L levels and N models per level, denoted as $\mathcal{M} = \{M_j^i\}$, where $j \in \{0, ..., N-1\}$ and $i \in \{0, ..., L-1\}$. Thus, any given SMDP in the hierarchy is denoted as $M_j^i = <S_j^i, A_j^i, T_j^i, R_j^i>$ (see the environment shown in Figure 1), where S_j^i is a set of states, A_j^i is a set of primitive and/or composite actions, T is a transition function that specifies the next state s' given the current state s and action a with probability $P(s', \tau | s, a)$, and R_j^i is a reward function that specifies the reward given to the agent for choosing action a when the environment makes a transition from state s to state s'. The random variable τ denotes the number of time-steps taken to execute action a in state s. In this form of dialogue control the execution of primitive actions yields single rewards and the execution of composite actions lasting τ time steps yields cumulative discounted rewards expressed as

$$r_{t+\tau} = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... + \gamma^{\tau-1}_{t+\tau},$$

where the discount rate $0 \leq \gamma \leq 1$ makes future rewards less valuable than immediate rewards as it approaches 0.

The goal in an SMDP is to find an optimal policy π^*, that maximizes the reward of each visited state. The optimal action-value function $Q^*(s, a)$ specifies the expected cumulative reward for executing action a in s and then following π^*. The Bellman equation for Q^* of subtask M_j^i can be expressed as

$$Q_j^{*i}(s, a) = \sum_{s', \tau} P_j^i(s', \tau | s, a)[R_j^i(s', \tau | s, a) + \gamma^\tau \max_{a'} Q_j^{*i}(s', a')],$$

and the optimal policy for each dialogue subtask is defined by

$$\pi_j^{*i}(s) = \arg\max_{a \in A_j^i} Q_j^{*i}(s, a).$$

Several methods have been investigated for learning a hierarchy of SMDPs such as Hierarchical Semi-Markov Q-Learning (HSMQ-Learning) [6], where the action-value function Q_j^{*i} of the previous equation is approximated according to the following update rule:

$$Q_j^i(s_t, a_t) \leftarrow (1-\alpha) Q_j^i(s_t, a_t) + [r_{t+\tau} + \gamma^\tau \max_{a'} Q_j^i(s_{t+\tau}, a')].$$

The summation over all τ time steps as appears in the Bellman equation is reflected here by using cumulative rewards $r_{t+\tau}$ received for executing actions a_t, and by raising γ to the power τ. See [5] for a detailed explanation of the HSMQ-learning algorithm applied to spoken dialogue. Briefly, this learning algorithm receives dialogue subtask M_j^i and a knowledge base used to initialize state s, performs similarly to Q-learning for primitive actions, but for composite actions it invokes recursively with a child subtask. When the subtask is completed with τ time steps it returns a cumulative reward $r_{t+\tau}$, and continues its execution until finding a terminal state for the root subtask M_0^0. This algorithm is iterated until convergence occurs to optimal context-independent policies [6].

The agent-environment interaction for hierarchical dialogue control is illustrated in Figure 1(a). Whilst the environment is modelled with a hierarchy of dialogue SMDPs, the

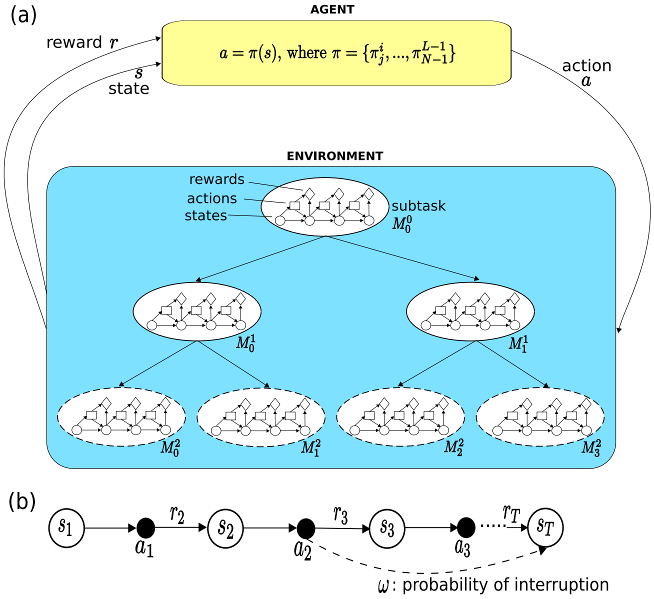

Figure 1: *(a) Architecture of the agent-environment interaction using a hierarchy of SMDPs M_j^i, where SMDPs denoted with dashed ellipses allow early termination. (b) Sample state-action trajectory with interrupted execution allowing policy-switching in their parent SMDP in order learn how to act in each state.*

learning agent takes action $a \in A_j^i$ in state $s \in S_j^i$ by using a hierarchy of policies executed with a top-down mechanism, where each invoked dialogue subtask is pushed into a stack and popped off when it terminates its execution. Note that decision-making on each SMDP uses its corresponding dialogue policy, e.g., the behaviour in the root dialogue subtask M_0^0 follows policy $\pi_0^0(s)$. Note also that the indexes i and j only identify a subtask in a unique way in the hierarchy, they do not specify the execution sequence of subtasks because that is learnt by the reinforcement learning agent.

3.2. Hierarchical control using a policy-switching approach

Typically, a Semi-Markov Decision Process (SMDP) is executed from a start state to a terminal goal state. But it may be possible that competing policies might be better than the one being executed (e.g., dialogue policies with different types of help). *Suppose that the reinforcement learning dialogue agent is executing policy $\pi_j^i(s_t)$ and that at each time step t there is a competing policy $\pi_k^i(s_t)$. If the cumulative reward of the latter policy is higher, why not interrupt policy π_j^i and switch to policy π_k^i?* Our proposal (inspired by [7]) is as follows: Competing behaviours during training are interrupted with a small probability ω in a similar way as the ϵ-greedy action-selection strategy [4]. That is, the agent continues its execution with probability $1-\omega$, and with probability ω interrupts its execution, see Figure 1(b):

$$execution = \begin{cases} \text{continue} & \text{if} \quad p(random) \leq 1 - \omega \\ \text{terminate} & \text{otherwise.} \end{cases}$$

When a policy is being learnt and its termination occurs, the algorithm's update rule is omitted (i.e., only performs learning on non-interrupted experiences). However, during policy testing, its termination occurs iff the cumulative reward $Q_j^{*i}(s, a)$ is lower than a competing cumulative reward $Q_k^{*i}(s, a)$ so that the parent policy can adapt its behaviour accordingly.

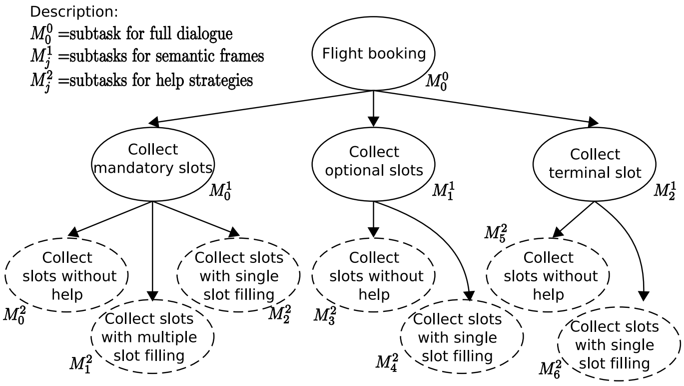

Figure 2: *A subtask hierarchy for the flight booking spoken dialogue system. The corresponding state variables and actions for each dialogue subtask M_j^i can be found in Tables 1, 2, 3.*

4. Experiments and results

Our hypothesis was that the policy-switching approach—using a parent behaviour to perform adaptation from competing sub-behaviours—would yield better spoken dialogue behaviours in comparison to policies without interruption. Our experiments are based on a mixed-initiative flight booking dialogue system using different types of information request, and employed a simulated conversational environment at the dialogue-act level.

4.1. Experimental setup

The flight booking system used the state variables described in Table 1, and the action set shown in Table 2. It can be noted that the state-action space (421.8 million state-actions) is large and not suitable for learning using a single tabular decision-maker. In contrast, the hierarchical state-action space representation used 11 dialogue subtasks: one parent, three children and seven grandchildren. Figure 2 illustrates the subtask hierarchy and Table 3 shows the state variables and actions per subtask. It can also be noted that the dialogue subtasks are applying state abstraction by ignoring irrelevant variables. The root subtask applies state abstraction by using high-level state variables as follows: variable MAN represents the status of subtask M_0^1, variable OPT represents the status of subtask M_1^1, and variable TER represents the status of subtask M_2^1. These state abstractions represented dialogue states in a more compact way (e.g., the policy-switching agent only used 106.8K state-actions). The reward function focused on efficient conversations, and is defined by the following rewards given to the agent for choosing action[2] a when the environment makes a transition from s to s':

$$R = \begin{cases} 0 & \text{for} \quad \text{successful (sub)dialogue} \\ \text{-10} & \text{for} \quad \text{an already collected subtask } M_j^i \\ \text{-10} & \text{for} \quad \text{collecting subtask } M_i^i \text{ before } M_{i-1}^i \\ \text{-10} & \text{for} \quad \text{presenting many/none items of information} \\ \text{-10} & \text{for} \quad \text{multiple greetings or closings} \\ \text{-10} & \text{for} \quad \text{collecting slots without a greeting} \\ \text{-10} & \text{for} \quad \text{executing action } a \text{ and yielding } s' = s \\ \text{-1} & \quad\quad\;\; \text{otherwise} \end{cases}$$

The learning setup used the HSMQ-learning algorithm for hierarchical reinforcement learning. The learning rate parameter α decays from 1 to 0 according to $\alpha = 100/(100 + \tau)$,

[2] Illegal actions had no effect in the dialogues and only wasted time, e.g., request an already filled slot, request an already confirmed slot, etc.

Table 1: *State variables for the flight booking dialogue system.*

Variable	Values	Description
SAL	$\{0, 1, 2\}$	Status of salutation: null, greeting, closing
C00	$\{0, ..., 4\}$	Status of mandatory slot 'departure city'
...	$\{0, ..., 4\}$	slots: departure city, date, time, airline
C05	$\{0, ..., 4\}$	Status of mandatory slot 'flight type'
SIF	$\{0, ..., 5\}$	Slot in focus
REI	$\{0, ..., 4\}$	# retries + explicit/implicit confirmations
DBT	$\{0, 1, 2\}$	Amount of database tuples
PRE	$\{0, 1\}$	Status of information presentation
ACK	$\{0, 1\}$	Status of acknowledgement

Values of variables: C_{ij}={0=unfilled, 1=low confidence, 2=medium confidence, 3=high confidence, 4=confirmed}, DBT={0=none, 1=few, 2=many}.

Table 2: *Action space for the flight booking dialogue system.*

Action	Description
req1	Request slot in focus without help
req2	Request slot in focus with help of multiple slot filling
req3	Request slot in focus with help of single slot filling
apo+req1	Apology for mis-recognition + request without help
apo+req2	Apology for mis-recognition + request type two
apo+req3	Apology for mis-recognition + request type three
sic+req1	Single implicit confirmation + request without help
sic+req2	Single implicit confirmation + request type two
sic+req3	Single implicit confirmation + request type three
mic+req1	Multiple implicit confirmation + request without help
mic+req2	Multiple implicit confirmation + request type two
mic+req3	Multiple implicit confirmation + request type three
sec	Single explicit confirmation of the slot in focus
mec	Multiple explicit confirmation of filled slots
acc	Move to the next ascending slot with lower-value
dbq+sta	Perform a database query + inform database status
pre+ofr1	Information presentation + offer without help
pre+ofr3	Information presentation + offer with help
apo+ofr1	Apology for mis-recognition + offer without help
apo+ofr3	Apology for mis-recognition + offer with help
ofr1	Offer database options
ofr3	Offer database options
ack	Acknowledgement of flight booking
gre	Greeting
clo	Good bye

Table 3: *Subtask hierarchy for the flight booking system.*

Subtask	State Variables	Actions
M_0^0	DBT,MAN,OPT, SAL,TER	M_0^1,M_1^1,M_2^1, dbq+sta,gre,clo
M_0^1	MAN,REI	M_0^2,M_1^2,M_2^2
M_1^1	OPT,REI	M_3^2,M_4^2
M_2^1	ACK,REI,TER	M_5^2,M_6^2,ack
M_0^2	END,SIF,C00,C01, C02,C03	req1,apo+req1,sic+req1, mic+req1,sec,mec,acc
M_1^2	END,SIF,C00,C01, C02,C03	req2,apo+req2,sic+req2, mic+req2,sec,mec,acc
M_2^2	END,SIF,C00,C01, C02,C03	req3,apo+req3,sic+req3, mic+req3,sec,mec,acc
M_3^2	END,C04	req1,apo+req1,sec
M_4^2	END,C04	req3,apo+req3,sec
M_5^2	END,PRE,C05	pre+ofr1,apo+ofr1,ofr1,sec
M_6^2	END,PRE,C05	pre+ofr3,apo+ofr3,ofr3,sec

Value of variables: {MAN, OPT, TER}={0=unfilled subtask, 1=filled subtask, 2=confirmed subtask}, END={0=continue execution, 1=interrupt execution}.

Table 4: *Heuristic conditional probabilities for simulated user/machine behaviour, given three types of information request. Notation: ob=user obedience (for filling the slot in focus), msf=multi-slot filling, ker=machine keyword error rate.*

Probability	Value	Description
$p(ob\|req1)$	0.5	Obedience given no help
$p(ob\|req2)$	0.7	Obedience given help type one
$p(ob\|req3)$	0.9	Obedience given help type two
$p(msf\|req1)$	0.5	Multi-slot filling given no help
$p(msf\|req2)$	0.3	Multi-slot filling given help type one
$p(msf\|req3)$	0.1	Multi-slot filling given help type two
$p(ker\|req1)$	0.4	ASR error given no given help
$p(ker\|req2)$	0.3	ASR error given help type one
$p(ker\|req3)$	0.2	ASR error given help type two

where τ represents elapsed time-steps in the current subtask. Each subtask M_j^i had its own learning rate. The discount factor $\gamma = 1$ makes future rewards equally as valuable as immediate rewards, as in [3]. The action selection strategy used ϵ-Greedy with $\epsilon = 0.01$, initial Q-values initialized to 0, and $\omega = 0.01$.

For training the reinforcement learning spoken dialogue agents we used the simulated conversational environment described in [5], and modified it to act according to the following types of information request so that the learning agent could infer adaptive behaviour: always requesting information without help, e.g., *When would you like to fly?* (req1); always requesting information with a multi-slot filling help example, e.g., *When would you like to fly? For example you can say "a flight on the 15th of December in the morning travelling with British Airways"* (req2); and always requesting information with a single-slot filling help example, e.g., *What is your destination city? For example you can say "a flight to London"* (req3). For such a purpose, whilst user responses were modified according to user obedience (ob) and multi-slot filling (msf), machine behaviour modified its behaviour according to keyword error rate (ker), see Table 4. The simulated speech recognition errors assigned equal amounts of substitutions, insertions and deletions. In a similar way, equal amounts of three tiered ASR (Automatic Speech Recognition) confidence levels were assigned to each keyword in the user utterances.

4.2. Experimental results

The following four hierarchical reinforcement learning agents were inferred: always requesting information without help (Baseline1), always requesting information with a multi-slot filling example (Baseline2), always requesting information with a single-slot filling example (Baseline3), and requesting information using using a combination of the baseline behaviours (PS=policy-switching). Each learning agent used 10^5 training dialogues. Test results[3] averaged over 10 runs of 1000 dialogues showed that the policy-switching dialogue policy learning approach yielded more efficient conversations than the baseline policies, on average 8.2% fewer system actions (see Table 5 for a detailed comparison of actions per dialogue). This result suggests that our proposed approach successfully adapted its behaviour to the given simulated conversational environment.

[3]The reinforcement learning agents were tested following policy $\pi^*(s)$ if $s <> s'$ and policy $\pi^{det}(s)$ otherwise, where $\pi^*(s)$ is the learnt dialogue policy and $\pi^{det}(s)$ is a hand-crafted deterministic dialogue policy as described in [5]. This form of action selection has been suggested for more coherent behaviour rather than a purely learnt one.

Table 5: *Average number of actions per test dialogue for the policy-switching approach (PS) and three baseline policies.*

Action	Baseline1	Baseline2	Baseline3	PS
acc	1.65	1.68	2.32	0.36
apo+ofr1	0.03	0.00	0.00	0.00
apo+ofr3	0.00	0.02	0.14	0.03
apo+req1	0.63	0.00	0.00	0.30
apo+req2	0.00	0.39	0.00	0.16
apo+req3	0.00	0.02	0.20	0.11
mec	0.67	0.47	0.22	0.59
mic+req1	0.97	0.00	0.00	0.74
mic+req2	0.00	0.82	0.00	0.09
mic+req3	0.00	0.00	0.49	0.04
ofr1	0.02	0.00	0.00	0.00
ofr3	0.00	0.01	0.09	0.16
pre+ofr1	1.15	0.00	0.00	1.00
pre+ofr3	0.00	1.08	1.22	0.00
req1	1.95	0.00	0.00	0.63
req2	0.00	1.75	0.00	1.20
req3	0.00	0.32	2.05	0.22
sec	3.19	3.25	3.12	3.11
sic+req1	1.07	0.00	0.00	0.90
sic+req2	0.00	1.27	0.00	0.27
sic+req3	0.00	0.00	2.17	0.29
SUM	**15.30**	**15.07**	**16.03**	**14.20**

Note: Four actions with occurrence of '1' were omitted.

5. Conclusions and future work

We have presented a method for learning adaptive modular dialogue behaviour using a policy-switching approach, where a learnt dialogue agent (with competing agents) interrupts its execution itself at any dialogue state if its competing dialogue agent shows to be better (with higher cumulative reward). Experimental results—in a simulated flight booking dialogue system—showed that the dialogue agent using policy-switching adapted its behaviour to the given environment, and yielded more efficient dialogues than three baselines that followed a unique type of information request. The importance of this approach lies in its application to a wide range of adaptive dialogue behaviours (e.g., adaptation to the user and ASR). This work can be extended by investigating other state variables and reward functions, by using more realistic simulated environments, and by evaluating the effectiveness of adaptation in real environments.

6. References

[1] Litman, D. and Pan, S., "Designing and evaluating an adaptive spoken dialogue system", in *UMUAI*, 12 (2/3), pp. 111-137, 2002.

[2] Chu-Carroll, J., "MIMIC: An adaptive mixed initiative spoken dialogue system for information queries", in ANLP, 2000.

[3] Singh, S., Litman, D., Kearns, M., and Walker, M., "Optimizing dialogue management with reinforcement learning: Experiments with the NJFun system", in *JAIR*, 16, pp. 105-133, 2002.

[4] Sutton, R. and Barto, A. *Reinforcement learning: An introduction*. MIT Press, 1998.

[5] Cuayáhuitl, H., *Hierarchical reinforcement learning for spoken dialogue systems*, Ph.D. thesis, University of Edinburgh, 2009.

[6] Dietterich, T., "An overview of MAXQ hierarchical reinforcement learning", in Symposium on Abstraction, Reformulation, and Approximation (SARA), pp. 26-44, HorseshoeBay, TX, USA, 2000.

[7] Sutton, R., Precup, D. and Singh, S., "Between MDPs and Semi-MDPs: A framework for temporal abstraction in reinforcement learning", in *Artificial Intelligence*, 112 (1), pp. 181-211, 1999.