Composite Task-Completion Dialogue Policy Learning via Hierarchical Deep Reinforcement Learning

Baolin Peng[⋆] **Xiujun Li**[†] **Lihong Li**[†] **Jianfeng Gao**[†]
Asli Celikyilmaz[†] **Sungjin Lee**[†] **Kam-Fai Wong**[⋆]
[†]Microsoft Research, Redmond, WA, USA
[⋆]The Chinese University of Hong Kong, Hong Kong
{blpeng, kfwong}@se.cuhk.edu.hk
{xiul,lihongli,jfgao,aslicel,sule}@microsoft.com

Abstract

Building a dialogue agent to fulfill complex tasks, such as travel planning, is challenging because the agent has to learn to *collectively* complete multiple subtasks. For example, the agent needs to reserve a hotel and book a flight so that there leaves enough time for commute between arrival and hotel check-in. This paper addresses this challenge by formulating the task in the mathematical framework of *options* over Markov Decision Processes (MDPs), and proposing a hierarchical deep reinforcement learning approach to learning a dialogue manager that operates at different temporal scales. The dialogue manager consists of: (1) a top-level dialogue policy that selects among subtasks or options, (2) a low-level dialogue policy that selects primitive actions to complete the subtask given by the top-level policy, and (3) a global state tracker that helps ensure all cross-subtask constraints be satisfied. Experiments on a travel planning task with simulated and real users show that our approach leads to significant improvements over three baselines, two based on handcrafted rules and the other based on flat deep reinforcement learning.

1 Introduction

There is a growing demand for intelligent personal assistants, mainly in the form of dialogue agents, that can help users accomplish tasks ranging from meeting scheduling to vacation planning. However, most of the popular agents in today's market, such as Amazon Echo, Apple Siri, Google Home and Microsoft Cortana, can only handle very simple tasks, such as reporting weather and requesting songs. Building a dialogue agent to fulfill complex tasks remains one of the most fundamental challenges for the NLP community and AI in general.

In this paper, we consider an important type of complex tasks, termed *composite task*, which consists of a set of subtasks that need to be fulfilled collectively. For example, in order to make a travel plan, we need to book air tickets, reserve a hotel, rent a car, etc. in a collective way so as to satisfy a set of cross-subtask constraints, which we call *slot constraints*. Examples of slot constraints for travel planning are: hotel check-in time should be later than the flight's arrival time, hotel check-out time may be earlier than the return flight depart time, the number of flight tickets equals to that of hotel check-in people, and so on.

It is common to learn a task-completion dialogue agent using reinforcement learning (RL); see Su et al. (2016); Cuayáhuitl (2017); Williams et al. (2017); Dhingra et al. (2017) and Li et al. (2017a) for a few recent examples. Compared to these dialogue agents developed for individual domains, the composite task presents additional challenges to commonly used, *flat* RL approaches such as DQN (Mnih et al., 2015). The first challenge is reward sparsity. Dialogue policy learning for composite tasks requires exploration in a much larger state-action space, and it often takes many more conversation turns between user and agent to fulfill a task, leading to a much longer trajectory. Thus, the reward signals (usually provided by users at the end of a conversation) are delayed and sparse. As we will show in this paper, typical flat RL methods such as DQN with naive ϵ-greedy exploration is rather inefficient. The second challenge is to satisfy slot constraints across subtasks. This requirement makes most of the existing methods of learning *multi-domain dialogue* agents (Cuayáhuitl, 2009; Gasic et al., 2015b) inapplicable: these methods train a

collection of policies, one for each domain, and there is no cross-domain constraints required to successfully complete a dialogue. The third challenge is improved user experience: we find in our experiments that a flat RL agent tends to switch between different subtasks frequently when conversing with users. Such incoherent conversations lead to poor user experience, and are one of the main reasons that cause a dialogue session to fail.

In this paper, we address the above mentioned challenges by formulating the task using the mathematical framework of *options over MDPs* (Sutton et al., 1999), and proposing a method that combines deep reinforcement learning and hierarchical task decomposition to train a composite task-completion dialogue agent. At the heart of the agent is a dialogue manager, which consists of (1) a top-level dialogue policy that selects subtasks (options), (2) a low-level dialogue policy that selects primitive actions to complete a given subtask, and (3) a global state tracker that helps ensure all cross-subtask constraints be satisfied.

Conceptually, our approach exploits the structural information of composite tasks for efficient exploration. Specifically, in order to mitigate the reward sparsity issue, we equip our agent with an evaluation module (internal critic) that gives intrinsic reward signals, indicating how likely a particular subtask is completed based on its current state generated by the global state tracker. Such intrinsic rewards can be viewed as heuristics that encourage the agent to focus on solving a subtask before moving on to another subtask. Our experiments show that such intrinsic rewards can be used inside a hierarchical RL agent to make exploration more efficient, yielding a significantly reduced state-action space for decision making. Furthermore, it leads to a better user experience, as the resulting conversations switch between subtasks less frequently.

To the best of our knowledge, this is the first work that strives to develop a composite task-completion dialogue agent. Our main contributions are three-fold:

- We formulate the problem in the mathematical framework of options over MDPs.
- We propose a hierarchical deep reinforcement learning approach to efficiently learning the dialogue manager that operates at different temporal scales.
- We validate the effectiveness of the proposed approach in a travel planning task on simulated as well as real users.

2 Related Work

Task-completion dialogue systems have attracted numerous research efforts. Reinforcement learning algorithms hold the promise for dialogue policy optimization over time with experience (Scheffler and Young, 2000; Levin et al., 2000; Young et al., 2013; Williams et al., 2017). Recent advances in deep learning have inspired many deep reinforcement learning based dialogue systems that eliminate the need for feature engineering (Su et al., 2016; Cuayáhuitl, 2017; Williams et al., 2017; Dhingra et al., 2017; Li et al., 2017a).

All the work above focuses on single-domain problems. Extensions to composite-domain dialogue problems are non-trivial due to several reasons: the state and action spaces are much larger, the trajectories are much longer, and in turn reward signals are much more sparse. All these challenges can be addressed by hierarchical reinforcement learning (Sutton et al., 1999, 1998; Singh, 1992; Dietterich, 2000; Barto and Mahadevan, 2003), which decomposes a complicated task into simpler subtasks, possibly in a recursive way. Different frameworks have been proposed, such as Hierarchies of Machines (Parr and Russell, 1997) and MAXQ decomposition (Dietterich, 2000). In this paper, we choose the *options framework* for its conceptual simplicity and generality (Sutton et al., 1998); more details are found in the next section. Our work is also motivated by hierarchical-DQN (Kulkarni et al., 2016) which integrates hierarchical value functions to operate at different temporal scales. The model achieved superior performance on a complicated ATARI game "Montezuma's Revenge" with a hierarchical structure.

A related but different extension to single-domain dialogues is multi-domain dialogues, where each domain is handled by a separate agent (Lison, 2011; Gasic et al., 2015a,b; Cuayáhuitl et al., 2016). In contrast to composite-domain dialogues studied in this paper, a conversation in a multi-domain dialogue normally involves one domain, so completion of a task does *not* require solving sub-tasks in different domains. Consequently, work on multi-domain dialogues focuses on different technical challenges such as transfer learning across different domains (Gasic

et al., 2015a) and domain selection (Cuayáhuitl et al., 2016).

3 Dialogue Policy Learning

Our composite task-completion dialogue agent consists of four components: (1) an LSTM-based language understanding module (Hakkani-Tür et al., 2016; Yao et al., 2014) for identifying user intents and extracting associated slots; (2) a state tracker for tracking the dialogue state; (3) a dialogue policy which selects the next action based on the current state; and (4) a model-based natural language generator (Wen et al., 2015) for converting agent actions to natural language responses. Typically, a dialogue manager contains a state tracker and a dialogue policy. In our implementation, we use a *global* state tracker to maintain the dialogue state by accumulating information across all subtasks, thus helping ensure all inter-subtask constraints be satisfied. In the rest of this section, we will describe the dialogue policy in details.

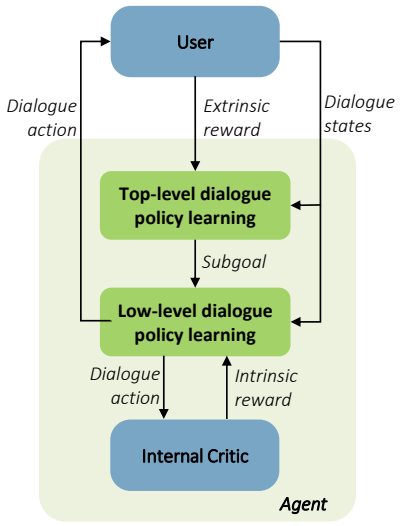

Figure 1: Overview of a composite task-completion dialogue agent.

3.1 Options over MDPs

Consider the following process of completing a composite task (e.g., travel planning). An agent first selects a subtask (e.g., book-flight-ticket), then takes a sequence of actions to gather related information (e.g., departure time, number of tickets, destination, etc.) until all users' requirements are met and the subtask is completed, and finally chooses the next subtask (e.g., reserve-hotel) to complete. The composite task is fulfilled after all its subtasks are completed collectively. The above process has a natural hierarchy: a top-level process selects which subtasks to complete, and a low-level process chooses primitive actions to complete the selected subtask. Such hierarchical decision making processes can be formulated in the *options* framework (Sutton et al., 1999), where options generalize primitive actions to higher-level actions. Different from the traditional MDP setting where an agent can only choose a primitive action at each time step, with options the agent can choose a "multi-step" action which for example could be a sequence of primitive actions for completing a subtask. As pointed out by Sutton et al. (1999), options are closely related to actions in a family of decision problems known as semi-Markov decision processes.

Following Sutton et al. (1999), an option consists of three components: a set of states where the option can be initiated, an intra-option policy that selects primitive actions while the option is in control, and a termination condition that specifies when the option is completed. For a composite task such as travel planning, subtasks like *book-flight-ticket* and *reserve-hotel* can be modeled as options. Consider, for example, the option *book-flight-ticket*: its initiation state set contains states in which the tickets have not been issued or the destination of the trip is long away enough that a flight is needed; it has an intra-option policy for requesting or confirming information regarding departure date and the number of seats, etc.; it also has a termination condition for confirming that all information is gathered and correct so that it is ready to issue the tickets.

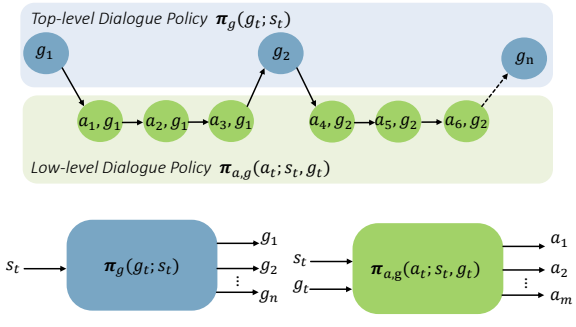

Figure 2: Illustration of a two-level hierarchical dialogue policy learner.

3.2 Hierarchical Policy Learning

The intra-option is a conventional policy over primitive actions, we can consider an inter-option

policy over sequences of options in much the same way as we consider the intra-option policy over sequences of actions. We propose a method that combines deep reinforcement learning and hierarchical value functions to learn a composite task-completion dialogue agent as shown in Figure 1. It is a two-level hierarchical reinforcement learning agent that consists of a top-level dialogue policy π_g and a low-level dialogue policy $\pi_{a,g}$, as shown in Figure 2. The top-level policy π_g perceives state s from the environment and selects a subtask $g \in \mathcal{G}$, where $\mathcal{G}$ is the set of all possible subtasks. The low-level policy $\pi_{a,g}$ is shared by all options. It takes as input a state s and a subtask g, and outputs a primitive action $a \in \mathcal{A}$, where $\mathcal{A}$ is the set of primitive actions of all subtasks. The subtask g remains a constant input to $\pi_{a,g}$, until a terminal state is reached to terminate g. The internal critic in the dialogue manager provides intrinsic reward $r_t^i(g_t)$, indicating whether the subtask g_t at hand has been solved; this signal is used to optimize $\pi_{a,g}$. Note that the state s contains global information, in that it keeps track of information for all subtasks.

Naturally, we aim to optimize the low-level policy $\pi_{a,g}$ so that it maximizes the following cumulative intrinsic reward at every step t:

$$\max_{\pi_{a,g}} \mathbb{E}\left[\sum_{k\geq 0} \gamma^k r^i_{t+k} \middle| s_t = s, g_t = g, a_{t+k} = \pi_{a,g}(s_{t+k})\right],$$

where r^i_{t+k} denotes the reward provided by the internal critic at step $t+k$. Similarly, we want the top-level policy π_g to optimize the cumulative extrinsic reward at every step t:

$$\max_{\pi_g} \mathbb{E}\left[\sum_{k\geq 0} \gamma^k r^e_{t+k} \middle| s_t = s, a_{t+k} = \pi_g(s_{t+k})\right],$$

where r^e_{t+k} is the reward received from the environment at step $t+k$ when a new subtask starts.

Both the top-level and low-level policies can be learned with deep Q-learning methods, like DQN. Specifically, the top-level dialogue policy estimates the optimal Q-function that satisfies the following:

$$Q_1^*(s,g) = \mathbb{E}\Big[\sum_{k=0}^{N-1} \gamma^k r^e_{t+k} + \gamma^N \cdot \max_{g'} Q_1^*(s_{t+N}, g') | s_t = s, g_t = g\Big], \quad (1)$$

where N is the number of steps that the low-level dialogue policy (intra-option policy) needs to accomplish the subtask. g' is the agent's next subtask in state s_{t+N}. Similarly, the low-level dialogue policy estimates the Q-function that satisfies the following:

$$Q_2^*(s,a,g) = \mathbb{E}\Big[r^i_t + \gamma \cdot \max_{a_{t+1}} Q_2^*(s_{t+1}, a_{t+1}, g) | s_t = s, g_t = g\Big].$$

Both $Q_1^*(s,g)$ and $Q_2^*(s,a,g)$ are represented by neural networks, $Q_1(s,g;\theta_1)$ and $Q_2(s,a,g;\theta_2)$, parameterized by θ_1 and θ_2, respectively.

The top-level dialogue policy tries to minimize the following loss function at each iteration i:

$$\mathcal{L}_1(\theta_{1,i}) = \mathbb{E}_{(s,g,r^e,s')\sim\mathcal{D}_1}[(y_i - Q_1(s,g;\theta_{1,i}))^2]$$
$$y_i = r^e + \gamma^N \max_{g'} Q_1(s', g', \theta_{1,i-1}),$$

where, as in Equation (1), $r^e = \sum_{k=0}^{N-1} \gamma^k r^e_{t+k}$ is the discounted sum of reward collected when subgoal g is being completed, and N is the number of steps g is completed.

The low-level dialogue policy minimizes the following loss at each iteration i using:

$$\mathcal{L}_2(\theta_{2,i}) = \mathbb{E}_{(s,g,a,r^i,s')\sim\mathcal{D}_2}[(y_i - Q_2(s,g,a;\theta_{2,i}))^2]$$
$$y_i = r^i + \gamma \max_{a'} Q_2(s', g, a', \theta_{2,i-1}).$$

We use SGD to minimize the above loss functions. The gradient for the top-level dialogue policy yields:

$$\nabla_{\theta_{1,i}} L_1(\theta_{1,i}) = \mathbb{E}_{(s,g,r^e,s')\sim\mathcal{D}_1}[(r^e + \gamma^N \max_{g'} Q_2(s', g', \theta_{1,i-1}) - Q_1(s,g,\theta_{1,i})) \nabla_{\theta_{1,i}} Q_1(s,g,\theta_{1,i})] \quad (2)$$

The gradient for the low-level dialogue policy yields:

$$\nabla_{\theta_{2,i}} L_2(\theta_{2,i}) = \mathbb{E}_{(s,g,a,r^i,s')\sim\mathcal{D}_2}[(r^i + \gamma \max_{a'} Q_2(s', g, a', \theta_{2,i-1}) - Q_2(s,g,a,\theta_{2,i})) \nabla_{\theta_{2,i}} Q_2(s,g,a,\theta_{2,i})] \quad (3)$$

Following previous studies, we apply two most commonly used performance boosting methods: target networks and experience replay. Experience replay tuples (s,g,r^e,s') and (s,g,a,r^i,s'), are sampled from the experience replay buffers $\mathcal{D}_1$ and $\mathcal{D}_2$ respectively. A detailed summary of the learning algorithm for the hierarchical dialogue policy is provided in Appendix B.

4 Experiments and Results

To evaluate the proposed method, we conduct experiments on the composite task-completion dialogue task of travel planning.

4.1 Dataset

In the study, we made use of a human-human conversation data derived from a publicly available multi-domain dialogue corpus[1] (El Asri et al., 2017), which was collected using the Wizard-of-Oz approach. We made a few changes to the schema of the data set for the composite task-completion dialogue setting. Specifically, we added inter-subtask constraints as well as user preferences (soft constraints). The data was mainly used to create simulated users, as will be explained below shortly.

4.2 Baseline Agents

We benchmark the proposed *HRL agent* against three baseline agents:

- A *Rule Agent* uses a sophisticated hand-crafted dialogue policy, which requests and informs a hand-picked subset of necessary slots, and then confirms with the user about the reserved tickets.
- A *Rule+ Agent* requests and informs all the slots in a pre-defined order exhaustedly, and then confirms with the user about the reserved tickets. The average turn of this agent is longer than that of the *Rule* agent.
- A *flat RL Agent* is trained with a standard flat deep reinforcement learning method (DQN) which learns a flat dialogue policy using extrinsic rewards only.

4.3 User Simulator

Training reinforcement learners is challenging because they need an environment to interact with. In the dialogue research community, it is common to use simulated users as shown in Figure 3 for this purpose (Schatzmann et al., 2007; Asri et al., 2016). In this work, we adapted the publicly-available user simulator, developed by Li et al. (2016), to the composite task-completion dialogue setting using the human-human conversation data described in Section 4.1.[2] During training, the simulator provides the agent with an (extrinsic) reward signal at the end of the dialogue. A dialogue is considered to be successful only when a travel plan is made successfully, and the information provided by the agent satisfies user's constraints. At the end of each dialogue, the agent receives a positive reward of $2*max\_turn$ ($max\_turn = 60$ in our experiments) for success, or a negative reward of $-max\_turn$ for failure. Furthermore, at each turn, the agent receives a reward of -1 so that shorter dialogue sessions are encouraged.

User Goal A user goal is represented by a set of slots, indicating the user's request, requirement and preference. For example, an *inform slot*, such as dst_city="Honolulu", indicates a user requirement, and a *request slot*, such as price="?", indicates a user asking the agent for the information.

In our experiment, we compiled a list of user goals using the slots collected from the human-human conversation data set described in Section 4.1, as follows. We first extracted all the slots that appear in dialogue sessions. If a slot has multiple values, like "or_city=[San Francisco, San Jose]", we consider it as a user preference (soft constraint) which the user may later revise its value to explore different options in the course of the dialogue. If a slot has only one value, we treat it as a user requirement (hard constraint), which is unlikely negotiable. If a slot is with value "?", we treat it as a user request. We removed those slots from user goals if their values do not exist in our database. The compiled set of user goals contains 759 entries, each containing slots from at least two subtasks: *book-flight-ticket* and *reserve-hotel*.

User Type To compare different agents' ability to adapt to user preferences, we also constructed three additional user goal sets, representing three different types of (simulated) users, respectively:

- *Type A*: All the informed slots in a user goal have a single value. These users have hard constraints for both the flight and hotel, and have no preference on which subtask to accomplish first.
- *Type B*: At least one of informed slots in the *book-flight-ticket* subtask can have multiple values, and the user (simulator) prefers to start with the *book-flight-ticket* subtask. If the user receives "no ticket available" from the agent during the conversation, she is willing to explore alternative slot values.

[1] https://datasets.maluuba.com/Frames

[2] A detailed description of the user simulator is presented in Appendix A.

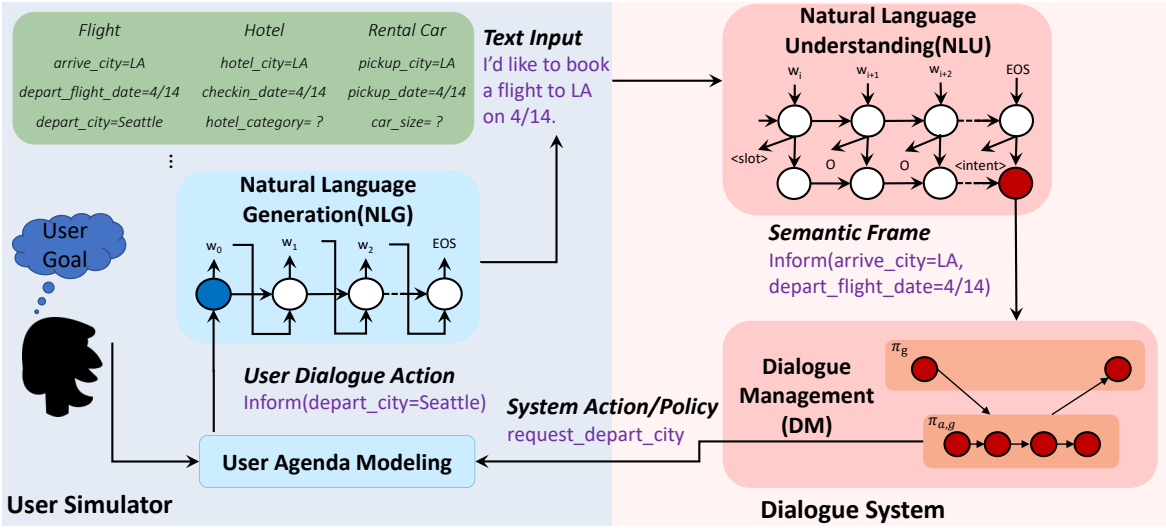

Figure 3: Illustration of the Composite Task-Completion dialogue System

- *Type C*: Similar to *Type B*, at least one of informed slots of the *reserve-hotel* subtask in a user goal can have multiple values. The user prefers to start with the *reserve-hotel* subtask. If the user receives a "no room available" response from the agent, she is willing to explore alternative slot values.

4.4 Implementation

For the RL agent, we set the size of hidden layer to 80. For the HRL agent, both top-level and low-level dialogue policies had a hidden layer size of 80. RMSprop was applied to optimize the parameters. We set batch size to 16. During training, we used the ϵ-greedy strategy for exploration. For each simulation epoch, we simulated 100 dialogues and stored these state transition tuples in an experience replay buffer. At the end of each simulation epoch, the model was updated with all the transition tuples in the buffer in a batch manner.

The experience replay strategy is critical to the success of deep reinforcement learning. In our experiments, at the beginning, we used a rule-based agent to run N ($N = 100$) dialogues to populate the experience replay buffer, which was an implicit way of imitation learning to initialize the RL agent. Then, the RL agent accumulated all the state transition tuples and flushes the replay buffer only when the current RL agent reached a success rate threshold no worse than that of the *Rule* agent.

This strategy was motivated by the following observation. The initial performance of an RL agent was often not strong enough to result in dialogue sessions with a reasonable success rate. With such data, it was easy for the agent to learn the locally optimal policy that "failed fast"; that is, the policy would finish the dialogue immediately, so that the agent could suffer the least amount of per-turn penalty. Therefore, we provided some rule-based examples that succeeded reasonably often, and did not flush the buffer until the performance of the RL agent reached an acceptable level. Generally, one can set the threshold to be the success rate of the *Rule* agent. To make a fair comparison, for the same type of users, we used the same *Rule* agent to initialize both the RL agent and the HRL agent.

4.5 Simulated User Evaluation

On the composite task-completion dialogue task, we compared the HRL agent with the baseline agents in terms of three metrics: success rate[3], average rewards, and the average number of turns per dialogue session.

Figure 4 shows the learning curves of all four agents trained on different types of users. Each learning curve was averaged over 10 runs. Table 1 shows the performance on test data. For all types of users, the HRL-based agent yielded more robust dialogue policies outperforming the hand-crafted rule-based agents and flat RL-based agent measured on success rate. It also needed fewer turns per dialogue session to accomplish a task than the rule-based agents and flat RL agent. The results

[3]Success rate is the fraction of dialogues where the tasks are successfully accomplished within the maximum turns.

	Type A			Type B			Type C		
Agent	Succ.	Turn	Reward	Succ.	Turn	Reward	Succ.	Turn	Reward
Rule	.322	46.2	-24.0	.240	54.2	-42.9	.205	54.3	-49.3
Rule+	*.535*	*82.0*	*-3.7*	*.385*	*110.5*	*-44.95*	*.340*	*108.1*	*-51.85*
RL	.437	45.6	-3.3	.340	52.2	-23.8	.348	49.5	-21.1
HRL	**.632**	**43.0**	**33.2**	**.600**	**44.5**	**26.7**	**.622**	**42.7**	**31.7**

Table 1: Performance of three agents on different User Types. Tested on 2000 dialogues using the best model during training. Succ.: success rate, Turn: average turns, Reward: average reward.

(a) Success Rate of User Type A (b) Success Rate of User Type B (c) Success Rate of User Type C

Figure 4: Learning curves of dialogue policies for different User Types under simulation

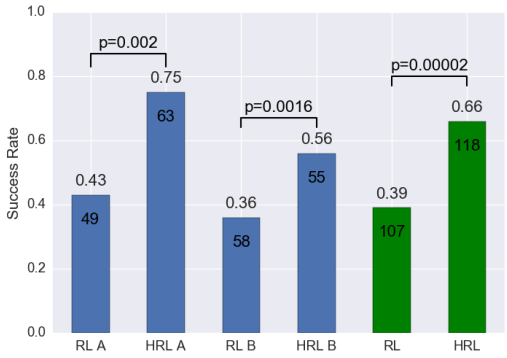

Figure 5: Performance of HRL agent versus RL agent tested with real users: success rate, number of tested dialogues and p-values are indicated on each bar; the rightmost green ones are for total (difference in mean is significant with $p < 0.01$).

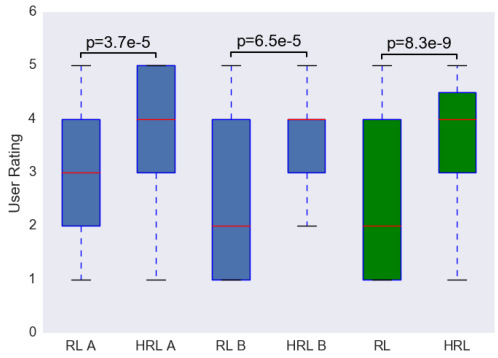

Figure 6: Distribution of user ratings for HRL agent versus RL agent, and total.

across all three types of simulated users suggest the following conclusions.

First, he HRL agent significantly outperformed the RL agent. This, to a large degree, was attributed to the use of the hierarchical structure of the proposed agent. Specifically, the top-level dialogue policy selected a subtask for the agent to focus on, one at a time, thus dividing a complex task into a sequence of simpler subtasks. The selected subtasks, combined with the use of intrinsic rewards, alleviated the sparse reward and long-horizon issues, and helped the agent explore more efficiently in the state-action space. As a result, as shown in Figure 4 and Table 1, the performance of the HRL agent on types B and C users (who may need to go back to revise some slots during the dialogue) does not drop much compared to type A users, despite the increased search space in the former. Additionally, we observed a large drop in the performance of the RL Agent due to the increased complexity of the task, which required more dialogue turns and posed a challenge for temporal credit assignment.

Second, the HRL agent learned much faster than the RL agent. The HRL agent could reach the same level of performance with a smaller number of simulation examples than the RL agent,

Table 2: Sample dialogue by RL and HRL agents with real user: Left column shows the dialogue with the RL agent; Right column shows the dialogue with the HRL agent; bolded slots are the joint constraints between two subtasks.

<table>
<tr><td colspan="2">User Goal</td></tr>
<tr><td>
reserve-hotel subtask:
{
“request_slots”: {
“hotel_price”: “?”
“hotel_date_checkout”: “?”
“hotel_name”: “?”
},
“inform_slots”: {
“hotel_city”: “Cancun”,
“hotel_numberofpeople”: “3”,
“hotel_date_checkin”: “09/20”
}
}
</td><td>
book-flight-ticket subtask:
{
“request_slots”: {
“price”: “?”
“return_time_dep”: “?”
“return_date_dep”: “?”
“depart_time_dep”: “?”
“seat”: “?”
},
“inform_slots”: {
“or_city”: “Campinas”,
“dst_city”: “Cancun”,
“numberofpeople”: “3”,
“depart_date”: ”9/20”
}
}
</td></tr>
<tr><td>RL agent</td><td>HRL agent</td></tr>
<tr><td>
Agent: Welcome to our travel agent. May I know your original city, destination city and departure date?
User: I depart from Campinas to Cancun and prefer to leave on 09-20.
Agent: There is a return flight on 9/26 and hotel checkout date is on 9/26 too.
User: My wife and son will travel with me, three people in total.
Agent: There is a hotel called Hotel Tropic available for you.
User: What depart time is available?

Agent: Departure time is at 10:00AM.
User: What is the departure time of return flight?
Agent: It is 4:00PM.
User: Which seat is available?
Agent: Only Business class available.
User: What’s the price for the flight?
Agent: Total is 1399 dollars for flight.
User: What’s the price for the hotel?
Agent: It is 1091 dollars for hotel.
User: Could you please help me book the tickets?
Agent: Departure flight on 9/20 and hotel checkin date on 9/20 are available.
User: Could you please help me book the tickets?
Agent: Okay. Tickets are issued.
User: Thanks!
Agent: Thanks!
</td><td>
Agent: Welcome to our travel agent. May I know your original city, destination city and departure date?
User: I depart from Campinas to Cancun and prefer to leave on 09-20.
Agent: Only Business class seat is available.

User: What is the depart time?
Agent: Departure time is at 10:00AM.
User: My wife and son will travel with me, three people in total.
Agent: There is one return flight departed at 4:00PM.
User: What is the date of the return flight?
Agent: It is on 9/26 and hotel checkout date is also 9/26.
User: What’s the price for the flight?
Agent: Total is 1399 dollars for flight.
User: Which hotel is available?
Agent: A hotel called Hotel Tropic is available for you.
User: What’s the price for the hotel?
Agent: It is 1091 dollars for hotel.
User: Could you please help me book the tickets?
Agent: Okay. Tickets are issued.

User: Thanks!
Agent: Thanks!
</td></tr>
<tr><td>Success: but the agent does not behave naturally where is an inter-subtask transition from book-flight-ticket to reserve-hotel.</td><td>Success</td></tr>
</table>

demonstrating that the hierarchical dialogue policies were more sample-efficient than flat RL policy and could significantly reduce the sample complexity on complex tasks.

Finally, we also found that the *Rule+* and *flat* RL agents had comparable success rates, as shown in Figure 4. However, a closer look at the correlation between success rate and the average number of turns in Table 1 suggests that the *Rule+* agent required more turns which adversely affects its success, whilst the *flat* RL agent achieves similar success with much less number of turns in all the user types. It suffices to say that our hierarchical RL agent outperforms all in terms of success rate as depicted in Figure 4.

4.6 Human Evaluation

We further evaluated the agents, which were trained on simulated users, against real human users, recruited from the authors’ affiliation. We conducted the study using the HRL and RL agents, each tested against two types of users: *Type A* users who had no preference for subtask, and *Type B* users who preferred to complete the *book-flight-ticket* subtask first. Note that *Type C* users were symmetric to Type B ones, so were not included in the study. We compared two (agent, user type) pairs: {RL A, HRL A} and {RL B, HRL B}; in other words, four agents were trained against their specific user types. In each dialogue session, one of the agents was randomly picked to converse with a user. The user was presented with a user

goal sampled from our corpus, and was instructed to converse with the agent to complete the task. If one of the slots in the goal had multiple values, the user had multiple choices for this slot and might revise the slot value when the agent replied with a message like "No ticket is available" during the conversation. At the end of each session, the user was asked to give a rating on a scale from 1 to 5 based on the naturalness and coherence of the dialogue. (1 is the worst rating, and 5 the best). We collected a total of 225 dialogue sessions from 12 human users.

Figure 5 presents the performance of these agents against real users in terms of success rate. Figure 6 shows the comparison in user rating. For all the cases, the HRL agent was consistently better than the RL agent in terms of success rate and user rating. Table 2 shows a sample dialogue session. We see that the HRL agent produced a more coherent conversation, as it switched among subtasks much less frequently than the *flat* RL agent.

5 Discussion and Conclusions

This paper considers composite task-completion dialogues, where a set of subtasks need to be fulfilled collectively for the entire dialogue to be successful. We formulate the policy learning problem using the options framework, and take a hierarchical deep RL approach to optimizing the policy. Our experiments, both on simulated and real users, show that the hierarchical RL agent significantly outperforms a flat RL agent and rule-based agents. The hierarchical structure of the agent also improves the coherence of the dialogue flow.

The promising results suggest several directions for future research. First, the hierarchical RL approach demonstrates strong adaptation ability to tailor the dialogue policy to different types of users. This motivates us to systematically investigate its use for dialogue *personalization*. Second, our hierarchical RL agent is implemented using a two-level dialogue policy. But more complex tasks might require multiple levels of hierarchy. Thus, it is valuable to extend our approach to handle such deep hierarchies, where a subtask can invoke another subtask and so on, taking full advantage of the options framework. Finally, designing task hierarchies requires substantial domain knowledge and is time-consuming. This challenge calls for future work on automatic learning of hierarchies for complex dialogue tasks.

Acknowledgments

Baolin Peng was in part supported by General Research Fund of Hong Kong (14232816). We would like thank anonymous reviewers for their insightful comments.

References

Layla El Asri, Jing He, and Kaheer Suleman. 2016. A sequence-to-sequence model for user simulation in spoken dialogue systems. In *Interspeech 2016*, pages 1151–1155. ISCA.

Andrew G. Barto and Sridhar Mahadevan. 2003. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, 13(1-2):41–77.

Heriberto Cuayáhuitl. 2009. Hierarchical reinforcement learning for spoken dialogue systems.

Heriberto Cuayáhuitl. 2017. SimpleDS: A simple deep reinforcement learning dialogue system. In *Dialogues with Social Robots*, pages 109–118. Springer.

Heriberto Cuayáhuitl, Seunghak Yu, Ashley Williamson, and Jacob Carse. 2016. Deep reinforcement learning for multi-domain dialogue systems. *arXiv preprint arXiv:1611.08675*.

Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2017. End-to-end reinforcement learning of dialogue agents for information access. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Thomas G. Dietterich. 2000. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303.

Layla El Asri, Hannes Schulz, Shikhar Sharma, Jeremie Zumer, Justin Harris, Emery Fine, Rahul Mehrotra, and Kaheer Suleman. 2017. Frames: A corpus for adding memory to goal-oriented dialogue systems. *arXiv preprint arXiv:1704.00057*.

Milica Gasic, Dongho Kim, Pirros Tsiakoulis, and Steve J. Young. 2015a. Distributed dialogue policies for multi-domain statistical dialogue management. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2015, South Brisbane, Queensland, Australia, April 19-24, 2015*, pages 5371–5375. IEEE.

Milica Gasic, Nikola Mrksic, Pei-hao Su, David Vandyke, Tsung-Hsien Wen, and Steve J. Young. 2015b. Policy committee for adaptation in multi-domain spoken dialogue systems. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding, ASRU 2015, Scottsdale, AZ, USA, December 13-17, 2015*, pages 806–812. IEEE.

Dilek Hakkani-Tür, Gokhan Tur, Asli Celikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. 2016. Multi-domain joint semantic frame parsing using bi-directional RNN-LSTM. In *Proceedings of The 17th Annual Meeting of the International Speech Communication Association*.

Tejas D. Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. 2016. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3675–3683.

Esther Levin, Roberto Pieraccini, and Wieland Eckert. 2000. A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Trans. Speech and Audio Processing*, 8(1):11–23.

Xiujun Li, Yun-Nung Chen, Lihong Li, and Jianfeng Gao. 2017a. End-to-end task-completion neural dialogue systems. *arXiv preprint arXiv:1703.01008*.

Xiujun Li, Yun-Nung Chen, Lihong Li, Jianfeng Gao, and Asli Celikyilmaz. 2017b. Investigation of language understanding impact for reinforcement learning based dialogue systems. *arXiv preprint arXiv:1703.07055*.

Xiujun Li, Zachary C Lipton, Bhuwan Dhingra, Lihong Li, Jianfeng Gao, and Yun-Nung Chen. 2016. A user simulator for task-completion dialogues. *arXiv preprint arXiv:1612.05688*.

Pierre Lison. 2011. Multi-policy dialogue management. In *Proceedings of the SIGDIAL 2011*, pages 294–300. The Association for Computer Linguistics.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Ronald Parr and Stuart J. Russell. 1997. Reinforcement learning with hierarchies of machines. In *Advances in Neural Information Processing Systems 10, [NIPS Conference, Denver, Colorado, USA, 1997]*, pages 1043–1049. The MIT Press.

Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve J. Young. 2007. Agenda-based user simulation for bootstrapping a POMDP dialogue system. In *Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, April 22-27, 2007, Rochester, New York, USA*, pages 149–152. The Association for Computational Linguistics.

Jost Schatzmann and Steve Young. 2009. The hidden agenda user simulation model. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(4):733–747.

Konrad Scheffler and Steve J. Young. 2000. Probabilistic simulation of human-machine dialogues. In *IEEE International Conference on Acoustics, Speech, and Signal Processing. ICASSP 2000, 5-9 June, 2000, Hilton Hotel and Convention Center, Istanbul, Turkey*, pages 1217–1220. IEEE.

Satinder P. Singh. 1992. Reinforcement learning with a hierarchy of abstract models. In *Proceedings of the 10th National Conference on Artificial Intelligence. San Jose, CA, July 12-16, 1992.*, pages 202–207. AAAI Press / The MIT Press.

Pei-Hao Su, Milica Gasic, Nikola Mrksic, Lina Rojas-Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Continuously learning neural dialogue management. *arXiv preprint arXiv:1606.02689*.

Richard S. Sutton, Doina Precup, and Satinder P. Singh. 1998. Intra-option learning about temporally abstract actions. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998), Madison, Wisconsin, USA, July 24-27, 1998*, pages 556–564.

Richard S. Sutton, Doina Precup, and Satinder P. Singh. 1999. Between MDPs and Semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-hao Su, David Vandyke, and Steve J. Young. 2015. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1711–1721.

Jason D Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid code networks: Practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Kaisheng Yao, Baolin Peng, Yu Zhang, Dong Yu, Geoffrey Zweig, and Yangyang Shi. 2014. Spoken language understanding using long short-term memory neural networks. In *2014 IEEE Spoken Language Technology Workshop, SLT 2014, South Lake Tahoe, NV, USA, December 7-10, 2014*, pages 189–194.

Steve J. Young, Milica Gasic, Blaise Thomson, and Jason D. Williams. 2013. POMDP-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

A User Simulator

User Goal In the task-completion dialogue setting, the first step of user simulator is to generate a feasible user goal. Generally, a user goal is defined with two types of slots: *request* slots that user does not know the value and expects the agent to provide it through the conversation; *inform* slots is slot-value pairs that user know in the mind, serving as *soft*/*hard* constraints in the dialog; slots that have multiple values are termed as *soft* constraints, which means user has preference, and user might change its value when there is no result returned from the agent based on the current values; otherwise, slots that have with only one value serve as *hard* constraint. Table 3 shows an example of a user goal in the composite task-completion dialogue.

	book-flight-ticket	*reserve-hotel*
inform	**dst_city**=LA **numberofpeople**=2 **depart_date_dep**=09-04 or_city=Toronto seat=economy	**hotel_city**=LA **hotel_numberofpeople**=2 **hotel_date_checkin**=09-04
request	price=? return_time_dep=? return_date_dep=? depart_time_dep=?	hotel_price=? hotel_date_checkout=? hotel_name=?

Table 3: An example of user goal

First User Act This work focuses on user-initiated dialogues, so we randomly generate a user action as the first turn (a user turn). To make the first user-act more reasonable, we add some constraints in the generation process. For example, the first user turn can be inform or request turn; it has at least two informable slots, if the user knows the original and destination cities, *or_city* and *dst_city* will appear in the first user turn etc.; If the intent of first turn is request, it will contain one requestable slot.

During the course of a dialogue, the user simulator maintains a compact stack-like representation named as *user agenda* (Schatzmann and Young, 2009), where the user state s_u is factored into an agenda A and a goal G, which consists of constraints C and request R. At each time-step t, the user simulator will generate the next user action $a_{u,t}$ based on the its current status $s_{u,t}$ and the last agent action $a_{m,t-1}$, and then update the current status $s'_{u,t}$. Here, when training or testing a policy without natural language understanding (NLU) module, an error model (Li et al., 2017b) is introduced to simulate the noise from the NLU component, and noisy communication between the user and agent.

B Algorithms

Algorithm 1 outlines the full procedure for training hierarchical dialogue policies in this composite task-completion dialogue system.

Algorithm 1 Learning algorithm for HRL agent in composite task-completion dialogue

Initialize experience replay buffer $\mathcal{D}_1$ for meta-controller and $\mathcal{D}_2$ for controller.
Initialize Q_1 and Q_2 network with random weights.
Initialize dialogue simulator and load knowledge base.
for $episode$=1:N **do**
  Restart dialogue simulator and get state description s
  while s is not terminal **do**
    $extrinsic\_reward := 0$
    $s_0 := s$
    select a subtask g based on probability distribution $\pi(g|s)$ and exploration probability ϵ_g
    while s is not terminal and subtask g is not achieved **do**
      select an action a based on the distribution $\pi(a|s,g)$ and exploration probability ϵ_c
      Execute action a, obtain next state description s', perceive extrinsic reward r^e from environment
      Obtain intrinsic reward r^i from internal critic
      Sample random minibatch of transitions from $\mathcal{D}_1$
      $y = \begin{cases} r^i & \text{if} s' \text{ is terminal} \\ r^i + \gamma * max_{a'} Q_1(\{s', g\}, a'; \theta_1) & \text{oterwise} \end{cases}$
      Perform gradient descent on loss $\mathcal{L}(\theta_1)$ according to equation 2
      Store transition($\{s,g\}$,a,r^i,$\{s',g\}$) in $\mathcal{D}_1$
      Sample random minibatch of transitions from $\mathcal{D}_2$
      $y = \begin{cases} r^e & \text{if} s' \text{ is terminal} \\ r^e + \gamma * max_{a'} Q_2(s', g', a'; \theta_2) & \text{oterwise} \end{cases}$
      Perform gradient descent on loss $\mathcal{L}(\theta_2)$ according to equation 3
      $extrinsic\_reward$ += r^e
      $s = s'$
    end while
    Store transition (s_0, g, $extrinsic\_reward$, s') in $\mathcal{D}_2$
  end while
end for