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