A User Answer Understanding Module

We define the user answer understanding module in Figure 2 based on a bidirectional Recurrent Neural Network with attention. Specifically, for the j-th word in the user answer d_i, we concatenate its forward and backward hidden states (i.e., $h_{d_{i,j}} = [\overrightarrow{h}_{d_{i,j}}; \overleftarrow{h}_{d_{i,j}}]$) as its semantic representation. Attention weights w_{att_j} over all words are computed with a trainable context vector c, i.e., $w_{att_j} = \text{softmax}(c^T h_{d_{i,j}})$, which will help the agent identify important words in d_i. User answer d_i is then represented as $v_{d_i} = \sum_j w_{\text{att}_j} h_{d_{i,j}}$.

B Training Algorithm for HRL

In our work, policy functions are all updated via Stochastic Gradient Ascent. Given the objective Eq (4), we deduce the updates for the high-level policy function below:

$$\nabla_\theta J(\theta) = E_{\pi^h}[\nabla_\theta \log \pi^h(g; s) u_t], \tag{6}$$

where $u_t = \sum_{\kappa=0}^{\infty} \gamma^\kappa r^h(s_{t+\sum_{n=1}^{\kappa} N_n}, g_{t+\sum_{n=1}^{\kappa} N_n})$ is the summation item within the two brackets in Eq (4).

Similarly, at low level, for the ongoing subtask $g_t \in \mathcal{G} = \{st_1, st_2, st_3, st_4\}$ at time step t, the updates of its policy function $\pi^l_{g_t}$ are given by:

$$\nabla_{\phi_{g_t}} J_{g_t}(\phi_{g_t}) = E_{\pi^l_{g_t}}[\nabla_{\phi_{g_t}} \log \pi^l_{g_t}(a; s) u_{g_t,t}], \tag{7}$$

where $u_{g_t,t} = \sum_{k \geq 0} \gamma^k r^l_{g_{t+k}}(s_{t+k}, a_{t+k})$.

Algorithm 1 details the entire training procedure. Line 1-2 initialize each policy network. Particularly, each low-level policy network is pre-trained via supervised learning (Section 4). We train the agent on M episodes (i.e., recipes). At the beginning of each episode, the state vector $s^l_{st_i}$ is initialized by calculating Eq (1-2) with $s^l_{st_{-i}} = \overrightarrow{0}$ (Line 7). Gradients of parameters θ and ϕ_{st_i} are accumulated during every episode. To improve efficiency, we perform gradient update for every 64 episodes (Line 31-34).

C Implementation Details

To accelerate the training of HRL-based agents, we adopt a similar skill as (Gordon et al. 2017; Wang et al. 2017) to initialize their low-level policy functions with the well-trained LAM-sup model. Particularly for HRL, we apply an "action mask" at the high level, so that the completed subtask will not be repeatedly chosen as the next subtask. However, the agent does not rely on this mask after being fully trained. The validation check is performed every 2K iterations on a randomly sampled 1K validation recipes. The best parameter setting is the one that obtains the most high-level reward in validation.

D User Simulator

For a trigger/action function, we design the following strategies to simulate user answers:

1. Revised function description: Each function has an official description on IFTTT.com, e.g., for function `Create a link note`, its description is "*This action will create a new link note from a source URL in the notebook you specify*." We revised the function description to make a simulated user answer, e.g., "*Create a new link note from a source URL in the notebook I specify*."
2. Paraphrases of the revised function description: Specifically, we replaced words and phrases with their paraphrases, which were collected and marked as semantically "Equivalence" by the PPDB paraphrase database (Pavlick et al. 2015).
3. Extractions from a large collection of real-user recipe descriptions in our training set (Ur et al. 2016): The extractions are based on six manually defined templates and will embrace language variation in simulated user answers. For example, for a recipe description "If X then Y," X will be considered as an answer to questions about the ground-truth trigger function and Y as an answer to those about the ground-truth action function.

E User Evaluation Examples

We present three user evaluation examples in Table 4.

Example 1
User: "pinterest post to wordpress" **Ground-truth recipe**: [tc: `Feed`, tf: `New feed item`, ac: `WordPress`, af: `Create a photo post`] **HRL agent**: "Which event triggers the action?" **User**: "when there's a new pinterest post' **HRL agent**: "Which event results from the trigger?" **User**: "create a post with photo" **Agent Prediction**: [tc: `Feed`, tf: `New feed item`, ac: `WordPress`, af: `Create a photo post`]
Example 2
User: "foto do instagram" **Ground-truth recipe**: [tc: `Instagram`, tf: `Any new photo by you`, ac: `Email`, af: `Send me an email`] **HRL agent**: "Which event triggers the action?" **User**: "I share new ins photo" **HRL agent**: "Which channel should act per your request?" **User**: "email" **Agent Prediction**: [tc: `Instagram`, tf: `Any new photo by you`, ac: `Email`, af: `Send me an email`]
Example 3
User: "send fiverr orders to wunderlist" **Ground-truth recipe**: [tc: `Fiverr`, tf: `New order received`, ac: `Gmail`, af: `Send an email`] **HRL agent**: "Which event triggers the action?" **User**: "a new order is recievied" **Agent Prediction**: [tc: `Fiverr`, tf: `New email in inbox from search`, ac: `Gmail`, af: `Send an email`]

Table 4: Two examples (Example 1-2) from user evaluation where the HRL agent correctly interpreted the user instructions and two (Example 3) where it did not. The agent failed when there is a typo in the word "recievied".

Algorithm 1 Learning algorithm for HRL

Initialize parameters θ of the high-level policy network randomly.
Initialize parameters ϕ_{st_i} $(i = 1, 2, 3, 4)$ of each low-level policy network with supervised pre-training.
Initialize gradients: $d\theta \leftarrow 0$, $d\phi_{st_i} \leftarrow 0$.
for $\#episode = 1$ to M **do**
    Reset the user simulator and get a recipe description $\mathcal{I}$.
    Initialize $b_i \leftarrow 0, d_i \leftarrow \emptyset, \forall i = 1, 2, 3, 4$. Observe $s_0 = \{\mathcal{I}, b_i, d_i\}$.
    Calculate $s^l_{st_i}, \forall i = 1, 2, 3, 4$, according to Eq (1-2), with $s^l_{st_{-i}} = \overrightarrow{0}$.
    $global\_turn \leftarrow 1$.
    $t \leftarrow 0$.
    while s_t is not terminal **and** $global\_turn \leq Max\_Global\_Turn$ **do**
        Sample a subtask $g_t \sim \pi^h(g; s_t)$ according to Eq (3). We denote g_t as st_{i_t}, i.e., the i_t-th subtask in the subtask set $\mathcal{G} = \{st_1, st_2, st_3, st_4\}$.
        $t_{start} \leftarrow t$.
        $local\_turn \leftarrow 1$, $a_t \leftarrow \varnothing$.
        while ($a_t == \varnothing$ **or** $a_t ==$ "AskUser") **and** $local\_turn \leq Max\_Local\_Turn$ **do**
            Sample a primitive action $a_t \sim \pi^l_{st_{i_t}}(a; s_t)$.
            Execute and receive a low-level reward $r^l_{st_{i_t}}(s_t, a_t)$.
            $d_{i_t} \leftarrow d_{i_t} \cup$ retrieved simulated user answer.
            Observe new state s_{t+1} with new d_{i_t}.
            Update $s^l_{st_{i_t}}$ according to Eq (1-2).
            $g_{t+1} \leftarrow g_t$.
            $t \leftarrow t + 1$.
            $local\_turn \leftarrow local\_turn + 1$.
        end while
        $d\phi_{st_{i_t}} \leftarrow d\phi_{st_{i_t}} +$ accumulated gradient according to Eq (7).
        Receive a high-level reward $r^h(s_{t_{start}}, g_{t_{start}})$.
        $b_{i_t} \leftarrow 1$.
        $t \leftarrow t + 1$. Observe state s_t.
        $global\_turn \leftarrow global\_turn + 1$.
    end while
    $d\theta \leftarrow d\theta +$ accumulated gradient according to Eq (6).
    if $\#episode \ \% \ 64 \ == 0$ **then**
        Perform update of θ and ϕ_{st_i} $(i = 1, 2, 3, 4)$ by gradient ascent using $d\theta$ and $d\phi_{st_i}$.
        $d\theta \leftarrow 0$, $d\phi_{st_i} \leftarrow 0$.
    end if
end for