Supplementary Materials for Agent-Aware Dropout DQN for Safe and Efficient On-line Dialogue Policy Learning

Lu Chen and **Xiang Zhou** and **Cheng Chang** and **Runzhe Yang** and **Kai Yu**
Key Lab. of Shanghai Education Commission for Intelligent Interaction and Cognitive Eng.
SpeechLab, Department of Computer Science and Engineering
Brain Science and Technology Research Center
Shanghai Jiao Tong University, Shanghai, China
{chenlusz,owenzx,cheng.chang,yang_runzhe,kai.yu}@sjtu.edu.cn

A Details of Restaurant Information Domain

There are 7 slots in restaurant information domain, among which 4 slots are "informable". An informable slot is one which the user can provide a value for, to use as a constraint on their search. All 7 slots are "requestable", i.e. the user can request the value of any slot. The details are described in Table 1.

slots	informable	requestable
area	Yes	Yes
food	Yes	Yes
pricerange	Yes	Yes
name	Yes	Yes
addr	No	Yes
phone	No	Yes
postcode	No	Yes

Table 1: Slots in restaurant information domain

There are 16 summary actions for system: *repeat, request_area, request_food, request_pricerange, confirm_area, confirm_food, confirm_pricerange, confirm_name, select_area, select_food, select_pricerange, select_name, offer_1, offer_2, offer_3, offer_4*. More details are described in Table 2.

Action Type	Description
repeat	Ask the user to repeat.
request_x	Ask the user what their goal for slot *x* is.
confirm_x	Ask the user to confirm whether the most likely value for slot *x* is right or not.
select_x	Ask the user to pick from the suggested two values for slot *x*.
offer_n	Suggest a venue which matches n informable slot constraints.

Table 2: Description of different summary action types.

B Dialogue Belief State and Rule-based Policy

B.1 Dialogue Belief State

At t-th turn, for each informable slot s, a belief state $\mathbf{b}_s = \{b_s^1, \cdots, b_s^{n_s}, b_s^r\}$[1] is maintained, where $b_s^1 > b_s^2 > \cdots > b_s^{n_s}$, n_s is the number of candidate values for s, and b_s^r denotes the belief that s has not been mentioned by the user up to t-th turn, i.e. the belief for the special value "*none*". Mathematically, $b_s^r = 1 - \sum_{i=1}^{n_s} b_s^i$.

For each slot, if b_s^1 is more than b_s^r, the slot-value pair (s,v_s^1) with corresponding belief b_s^1 will be added into the database query constraints $\mathcal{DC}$.

B.2 Rule-based Policy

The rules R1, R2, R3, R1* and R4 are described in Algorithm 1∼5. Figure 1 gives an example to show the difference of decisions made by different ordered rules.

Algorithm 1 Rule **R1**

Require:
The belief state $\mathbf{b}_s = \{b_s^1, \cdots, b_s^{n_s}, b_s^r\}$ for each slot.

Initialize the summary action a with *null*.
for s = *area, food, pricerange, name* **do**
  if $b_s^1 >= 0.1$ and $b_s^1 < 0.6$ **then**
    $a \leftarrow$ *confirm_s*
    break
  end if
end for
return a

[1] For the sake of brevity, the subscript t is omitted.

Algorithm 2 Rule **R2**

Require:
The belief state $\mathbf{b}_s = \{b_s^1, \cdots, b_s^{n_s}, b_s^r\}$ for each slot.

Initialize the summary action a with *null*.
Initialize the database query constraints $\mathcal{DC}$ with $\{\}$.
for s = *area, food, pricerange, name* **do**
  if $b_s^1 > b_s^r$ **then**
    Add (s, v_s^1, b_s^1) into $\mathcal{DC}$.
    break
  end if
end for
$n \leftarrow \#\mathcal{DC}$
if $n > 0$ **then**
  $a \leftarrow$ *offer_n*
end if
return a

Algorithm 3 Rule **R3**

Equally sample a slot s from {*area, food, pricerange*}.
$a \leftarrow$ request_s
return a

Algorithm 4 Rule **R1***

Require:
The belief state $\mathbf{b}_s = \{b_s^1, \cdots, b_s^{n_s}, b_s^r\}$ for each slot.

Initialize the summary action a with *null*.
for s = *area, food, pricerange, name* **do**
  if s = *food* **then**
    if $b_s^1 >= 0.1$ and $b_s^1 < 0.6$ **then**
      $a \leftarrow$ *confirm_s*
      break
    end if
  else
    if $b_s^1 >= 0.1$ and $b_s^1 < b_s^r$ **then**
      $a \leftarrow$ *confirm_s*.
      break
    end if
  end if
end for
return a

Algorithm 5 Rule **R4**

Require:
The belief state $\mathbf{b}_s = \{b_s^1, \cdots, b_s^{n_s}, b_s^r\}$ for each slot.

Initialize the summary action a with *null*.
Initialize the database query constraints $\mathcal{DC}$ with $\{\}$.
for s = *area, food, pricerange, name* **do**
  if $b_s^1 > b_s^r$ **then**
    Add (s, v_s^1, b_s^1) into $\mathcal{DC}$.
    break
  end if
end for
$n \leftarrow \#\mathcal{DC}$
if $n = 1$ and *area* is not in $\mathcal{DC}$ **then**
  $a \leftarrow$ *request_area*
end if
return a

C Details of Companion Function

In our experiments, the probability of guidance from the teacher p_{tea} is determined by the value of ΔC_e. The chosen function should meet the following requirements: monotonously increasing and the value has domain $[0, 1]$. According to the results of our early experiments, the use of simple linear function does not provide good result, the system suffers from great performance loss in the early stage so the safety of the system is not guaranteed. Therefore, we choose to use a concave function. In detail, the function used in our experiment is a hyperbola:

$$P_{tea}(\Delta C_e) = \phi(\lambda) = -\frac{1}{\lambda + \frac{\sqrt{5}-1}{2}} + \frac{\sqrt{5}+1}{2},$$

where $\lambda = \frac{\Delta C_e}{C_{th} - \overline{C}_{min}}$, and $\overline{C}_{min}$ is the minimum value of $\overline{C}_e$ reached by the system in the experiments. This function meets all the requirements above while maintaining its simplicity. It performs well in the experiments.

D Illustrations of Safety Loss and Efficiency Loss

D.1 Safety Loss

As mentioned in section 4, the value of safety loss is equivalent to the area of the region between the acceptable performance line and the system performance curve. An illustration is shown in Figure 3, the safety loss is equal to the area of the blue region.

	area	food	pricerange	name
dialogue belief state	east = 0.09 West = 0.04 center = 0.01 none= 0.86	Chinese = 0.08 French = 0.02 none = 0.9	cheap = 0.5 expensive = 0.2 none = 0.3	none = 1.0
database query constraints	(pricerange, cheap) 0.5			
ordered rules	R1, R2, R3	R1, R4, R2, R3	R1*, R2, R3	R1*, R4, R2, R3
summary action	confirm_pricerange	confirm_pricerange	offer_1	request_area

Figure 1: An example to show the difference of decisions made by different ordered rules.

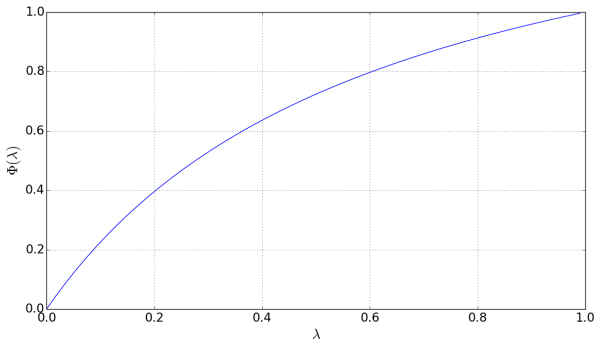

Figure 2: Graph of function $\phi(\lambda)$

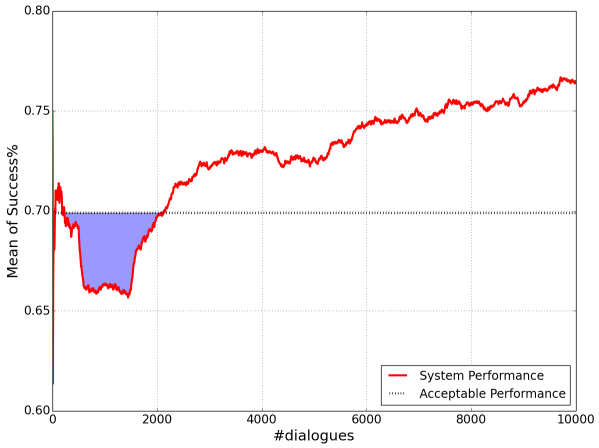

Figure 3: Illustration of Safety Loss

D.2 Efficiency Loss

In comparison to the safety loss, the efficiency loss gives a greater penalty on the time. The intuition behind the mathematical equation is that we want to penalize the performance loss after a longer time period more than the performance loss at the very beginning. This is achieved by multiplying e to each loss $\max(0, S_i - S_e)$ in the summation.

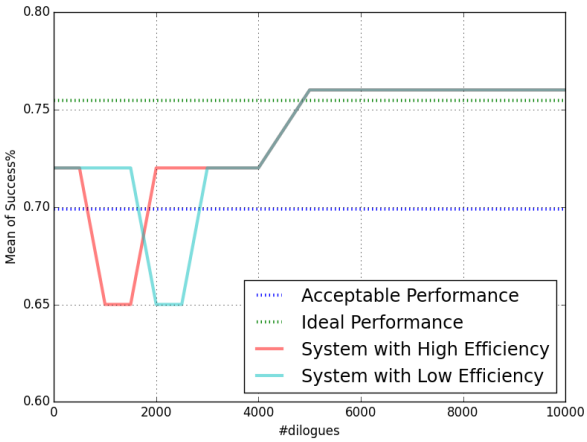

Figure 4: Comparison of Efficiency Loss between two Systems

For instance, shown in Figure 4, the area between the ideal performance line and the blue/red line are exactly the same. However, the blue line presents a large performance drop in later dialogues, which has a greater time penalty leading to a greater efficiency loss value. Therefore, the efficiency loss will prefer the red line, which is consistent with the normal intuition.