Optimizing Policy via Deep Reinforcement Learning for Dialogue Management

Guanghao Xu*, Hyunjung Lee**, Myoung-Wan Koo*, Jungyun Seo*
Department of Computer Science and Engineering
Sogang University, Seoul, Korea*
Institute of Linguistics
University of Leipzig, 04107, Germany**
guanghao412@gmail.com, hyunjung.lee@uni-leipzig.de, {mwkoo, seojy}@sogang.ac.kr

***Abstract*—In this paper, we propose a dialogue manager model based on Deep Reinforcement Learning, which automatically optimizes a dialogue policy. The policy is trained within deep Q-learning algorithm, which efficiently approximates value of actions given a large space of dialogue state. Evaluation processes are conducted by comparing the performance of the proposed model to a rule-based one on the dialogue corpora of DSTC2 and 3 under three different levels of error rate in Spoken Language Understanding. Experimental results prove that given certain level of SLU error, the dialogue manager with self-learned policy shows higher completion rate and the robustness to SLU error. Overcoming the drawbacks of rule-based approach such as limited flexibility and high maintenance cost, our model shows the strength of self-learning algorithm in optimizing policy of dialogue manager without any hand-crafted features.**

Keywords—Deep Reinforcement Learning; Dialogue Management; Dialogue Policy

I. Introduction

Spoken Dialogue Systems (SDS) has shown innumerable benefits by interacting with users and provide some useful information such as weather forecast or personal schedule. Development of such systems has become more and more popular and many companies have launched their own dialogue systems such as Siri, Google Now and Echo that enables to interacts with users in Speech form. Dialogue manager plays a central role in building a successful SDS by apprehending a state of a dialogue in a current turn and deciding a proper action to take for a next turn, thereby implementing a human-like agent which interacts with actual users.

One straightforward way to build a dialogue manager is to define a set of rules that the system is supposed to follow during a dialogue. Though the rule-based approach is easy and undemanding, such model suffers from problems such as limited flexibility and high maintenance cost. To achieve an improved model, there has been attempt to design dialogue manager within Reinforcement Learning (RL) framework, since RL-based model is able to learn and train policy over time with experience. The RL-based dialogue model, however, still needs an intervention from a system developer to represent dialogue state, dialogue actions and a reward function which instructs the system on the right track of dialogues. In contrast to this supervised way of learning, a deep Reinforcement Learning (deep RL) algorithm is proposed to learn how to control policies from raw video data in complex RL environment such as Atari games, where the agent equipped with deep RL policy surpasses a human expert in several games [1].

Drawn on such insights of deep RL, we present a model of dialogue manager in deep DL applied to goal-oriented task without carefully designing hand-crafted features. In return for reducing the development efforts and increasing the system's flexibility as much as possible, our model has an ability to decide an optimal action for a next turn by representing dialogue state automatically from ontology. For simplification of training process, we adopt a simulated user in order to interact with the deep RL policy in semantic level. Experimental results prove that the policy learned by deep RL achieves a good completion rate of dialogues as well as robustness to noise in speech.

The rest of this paper is organized as follows. Section 2 gives a background on reinforcement learning and the core properties of deep RL. In Section 3, we describe the architecture of our model, and each component of dialogue manger is involved to optimize the policy via deep RL: dialogue state, dialogue action, reward function and a user simulator. Section 4 describes how we set up the experiments for training the dialogue policy and evaluate the proposed model. In Section 5, we provide our experimental results and discusses the implication of applying deep RL algorithm in designing dialogue manager. The Section 6 concludes this paper.

II. Background

In this section, we explore the main idea of RL and deep RL to understand how each algorithm train a policy, and introduce some previous works using RL and deep RL method.

A. Reinforcement Learning

An agent in RL learns its behavior by taking actions in an environment in discrete time steps. At each time t, the agent receives a representation of state s_t, $s_t \in S$, where S is a state space and selects an action a_t, $a_t \in A$, where A is a set of

2375-9356/18/$31.00 ©2018 IEEE
DOI 10.1109/BigComp.2018.00101

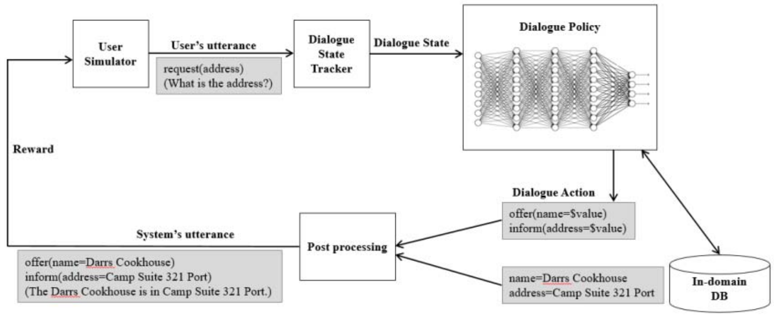

Fig. 1. The architecture of our Dialouge Manager

possible actions that the agent can take. As the result of its action, the agent receives a reward r and transits to a new state s_{t+1}. The goal of an agent is to select 'best' actions by maximizing its cumulative discounted reward, defined as

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^{T-1} r_T , \quad (1)$$

where γ is a discount factor and T is a final time step [2]. Given that the agent follows a policy $\pi: S \rightarrow A$, which defines a mapping from states to actions, an potential value of actions a in the current state s is estimated by Q-function as

$$Q^*(s,a) = max_\pi E[R_t | s_t = s, a_t = a, \pi]. \quad (2)$$

The more accurate the Q-function is, the better policy the agent learns.

B. Deep Reinforcement Learning

Though traditional RL algorithms like Q-learning [3] and SARSA could be used in learning the Q-function, they are quite inefficient, especially when the state space becomes large or even infinite. Recent advances in deep RL have achieved remarkable improvements in various tasks such as computer vision and speech recognition. Minh et al. proposed a model based on reinforcement learning method called Deep Reinforcement Learning (DRL, also called as deep Q-learning) to control policies directly from raw pixel inputs in Atari game. They proved that deep RL algorithm outperforms all previous methods on six of the games and surpasses even a human expert on three of them [1].

The core idea of deep RL is to adopt a function approximator based on deep neural network which is called Q-network to estimate the action-value function $Q(s,a;\theta) \approx Q^*(s,a)$, where θ is the parameters of the Q-network. The Q-network could be constructed in any form, such as a multi-layer feed forward network, a convolutional neural network or even a recurrent neural network.

In deep RL algorithm, the learning agent maintains two Q-networks: a value network and a policy network. The value network estimates value of target action, based on which policy network determines which action the agent should take for the next turn. The policy network is trained toward minimizing loss function $L_i(\theta_i)$ that changes at each iteration i,

$$L_i(\theta_i) = E[(y_i - Q(s,a;\theta_i))^2], \quad (3)$$

where y_i is the value of target action for iteration i which is estimated by the value network at iteration $i-1$,

$$y_i = E[r + \gamma max_{a'} Q(s',a';\theta_{i-1}) | s,a]. \quad (4)$$

To ensure adequate exploration of state space, the ϵ-greedy strategy is applied. The agent greedily chooses an action based on the value of agent's action calculated by the policy network,

$$a = max_a Q(s,a;\theta), \quad (5)$$

with probability $1-\epsilon$ and selects a random action with probability ϵ.

III. Deep RL for Dialogue Modeling

In this section, we present our deep RL approach to build a dialogue manager toward policy optimization. The overall architecture of our model is presented in Figure 1.

A. Dialogue Action

In our deep-RL based dialogue system, agent's responses and user's utterances are converted into semantic form called Dialogue Action with the form of '$Act\ (slot, value)$'. Rather than directly using raw utterances, we can have better control over the system's behaviors. The *act* is constructed from the given the previous defined set of system's action, the *slot* is acquired from in-domain ontology. Due to the sparsity issues, *value*[1] is temporarily left vacant in the level of constructing the value and policy network. The exact instance of *value* is later added to the corresponding slot as a post-processing step, once the deep-RL policy determines the best action. The full set of action-slot paradigm is presented in the Appendix III.

B. Dialogue State

Dialogue state of each dialogue turn represents the information that the user want the system to do. To make appropriate response to the user, the system should keep track of the changes in dialogue state during the entire dialogue, and it is also responsible for representing the dialogue state over

[1] We would like to make a clear-cut between *value* and the value network. The italicized *value* denotes the specific value of slots. For example, 'French', 'American' and 'Mexican' are possible *value*s for the corresponding *slot* 'Food'.

TABLE I. THE EXAMPLE OF INPUT LAYER OF Q-NETWORK

	Output of Dialogue State Tracker			SLU N-best results of user's utterance			Results of DB query
Components	Goals	Methods	Requested	SLU 1-best	SLU 2-best	SLU 3-best	Matched count
No. of dimension	5	5	9	78	78	78	1

food	pricerange	name	area	this
0.9458	0.6613	0.0	0.0613	0.0

turns by using the SLU results.[2] The dialogue state tracker outputs for each turn distributions for each of the three components of the dialogue state: *Goals*, *Method* and *Requested slots.* We represent the dialogue state in the form of continuous vector. Table 1 shows the examples of dialogue state in terms of *Goal.* To let the dialogue agent aware of backend system such as a database, we also add the results of a database query to the state vector. It is possible to automatically construct the dialogue state vector beforehand by using the domain-specific ontology information, which can reduce a lot of human effort in designing the possible dialogue state.

C. User Simulator

As stated above, the deep RL agent learns over times by experiences. In other word, the dialogue manager needs a lot of dialogues to train and feedbacks to its corresponding actions to optimize a good policy. Normally, the number of dialogues required to train a real-world dialogue manager is approximately the order of 100k, which is impractical to collect such many dialogues from real users [4]. Therefore, we develop an agenda-based user simulator which is similar to the one proposed by Schatzmann, to automatically train the policy [5].

Let us briefly describe the process of how user simulator operates. The simulator is initialized with a certain agenda which consists of *constraints* (i.e. food=korean, price=cheap, area=east etc.) and *requests* (i.e. address, phone, signature etc.) by using ontology information in a given domain. During the dialogue, the simulator interacts with the dialog agent based on its agenda and provides a reward to agent's actions to evaluate the success rate of dialogues. The interaction process is illustrated in Figure 1.

D. Reward Function

During scoring the success rate of a dialogue, we say that a dialogue is successful if the dialogue agent successfully searches a restaurant and answered all the user's requests within 10 dialogue turns. Based on this setting, a reward function is set to give a reward of 20 for successful dialogues, and a penalty -10 for failed dialogues. To encourage the agent to produce actions effectively, an additional -1 is charged for each dialogue turn so that the agent behaves as fast as possible.

E. Q-network and Training

The architecture of the Q-network consists of a fully-connected multi-layer perceptron with 254 nodes in the input layer, 100 nodes in the first and second hidden layer, and 51 nodes in the output layer.[3] The Q-network outputs a probability distributions over all agent's actions given the current dialogue state vector and optimize the dialogue policy automatically.

During the training of the Q-network, we adopt a ϵ-greedy strategy, where the probability ϵ is initially set to 1.0 and gradually decreased to 0.1 over the first 10k dialogues. Then we set the ϵ to 0 and train the policy for another 10k dialogues.

IV. EXPERIMENTAL SETUP

We evaluate the proposed RL dialogue policy on the DSTC2 and 3 dialogue corpora by comparing to a rule-based dialogue policy. Experiments are conducted on three levels of SLU error rates: *None*, *Low* and *High*.

A. DSTC2 & 3 Dialogue Corpora

The DSTC2 and 3 dialogue corpora were collected using Amazon Mechanical Turk and released in 2013 and 2014 respectively, initially intended for evaluating the performance of Dialogue State Trackers [6, 7]. The domain of DSTC2 provides restaurant information and the DSTC3 embraces tourist information to extend the domain of restaurant information, including bars, cafes and several new slots. The lists of slots and the number of their possible values of DSTC2 & 3 corpora are presented in Appendix I and II. Both the DSTC2 & 3 corpora and their ontology file are publicly available.[4]

B. SLU error rates: None, Low and High

One of the advantages of deep RL-based dialogue policy over the rule-based policy is the robustness to SLU errors. To test the SLU error robustness, we mimic three environments with different levels of noise by using the SLU N-best results stated in the corpora. Table 2 and Table 3 summarize the SLU error rate of DSTC2 and DSTC3 dialogue corpora, respectively.

C. Rule-based Dialogue Policy

To compare the performance of deep RL-policy, we build a rule-based dialogue policy as a baseline model. The model

[2] In this work, we use a public available, rule-based state tracker that is described in DSTC2.

[3] See Table 1. The number of nodes in the input layer is calculated by the total sum of the number of vectors dimension in Dialogue State. The 51 nodes of output layers are described in Appendix III.

[4] http://camdial.org/~mh521/dstc/

TABLE II. SLU ERROR RATE OF DSTC2CORPORA

SLU Error Level	Top-1 Error Rate	Top-10 Error Rate
None	0%	0%
Low	29.02%	16.69%
High	36.98%	23.71%

* The "Top-1 Error Rate" means the probability that the ground truth dialogue acts are not presented in Top-1 SLU result. Similarly, the "Top-10 Error Rate" means the probability that the ground truth dialogue acts are not presented in Top-10 SLU result.

TABLE III. SLU ERROR RATE OF DSTC3 CORPORA

SLU Error Level	Top-1 Error Rate	Top-10 Error Rate
None	0%	0%
Low	16.17%	6.78%
High	31.22%	19.43%

Algorithm 1 – Rule-based dialogue policy.

$G \leftarrow$ the 'goal' component of the state tracker output.
$R \leftarrow$ the 'requested slot' component of the state tracker output.
$S \leftarrow$ the DB query result with constrains in G.
A_m: placeholder for output system dialogue acts.
if *length*(S) = 0 **then**
    A_m=canthelp(slot=value), fill slot=value using G.
if *length*(G) < 2 **then**
    A_m=request(slot), fill slot using slots that not yet included in G.
else:
    venue=*random*(S)
    A_m=offer(name=venue.name)
    for slot **in** R **do**
        A_m=A_m+inform(venue.slot=venue.value)
output system response A_m.

issues a query and makes a response to user's utterance using a set of predefined rules. The pseudo code of the rule-based dialogue policy is presented in Algorithm 1.

V. RESULTS AND DISCUSSION

Table 4 and Table 5 summarize the comparative results of the dialogue manager with the deep RL policy and the rule-based policy. As shown in Table 4 and 5, the rule-based policy always achieves a 100% dialogue success rate if there exists no SLU error. When the SLU error is relatively low, the deep RL policy outperforms the rule-based policy 4~5% in terms of dialogue success rate. Also, the Deep RL policy has another advantage in terms of the average dialogue turns which are required for completing a dialogue, which is shorter than the baseline model with rule-based policy. The result suggests that the deep RL policy interacts with the simulated user more effectively than the rule-based policy.

We observe that the differences of two policies in terms of dialogue success rate and average dialogue turns are more noticeable in the extended dialogue domain, DSTC3. Appendix IV and Appendix V presents an example of dialogue between the simulated user and the dialogue manager with rule-based and deep RL based policy.

Figure 2 presents the changes of dialogue success rate with respect to the number of dialogues used in training. The success rate is converged after 10k dialogues under the *None* SLU error

TABLE IV. COMPARATIVE RESULTS IN DSTC2 DOMAIN.

SLU Error Level	Policy	Dialogue Success Rate	Average Dialog Turns
None	Rule-based	100%	7.42
	Deep RL	99.38%	5.84
Low	Rule-based	85.57%	7.47
	Deep RL	**90.35%**	**7.74**
High	Rule-based	77.14%	7.37
	Deep RL	**89.55%**	**8.16**

TABLE V. COMPARATIVE RESULTS IN DSTC3DOMAIN.

SLU Error Level	Policy	Dialogue Success Rate	Average Dialog Turns
None	Rule-based	100%	8.58
	Deep RL	99.16%	5.84
Low	Rule-based	91.49%	8.16
	Deep RL	**95.15%**	**6.86**
High	Rule-based	52.49%	11.53
	Deep RL	**86.85%**	**8.05**

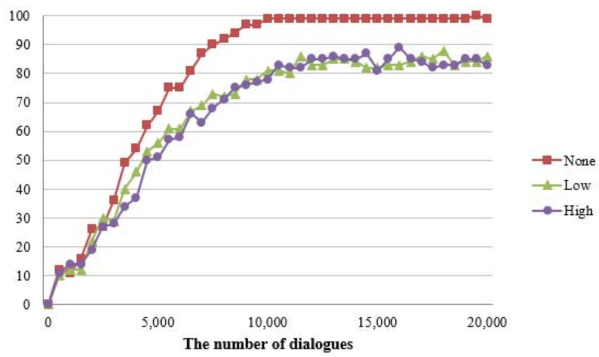

Fig. 2. The Success Rate of Dialogues in Different SLU Errors.

level, under the *Low* and *High* case, the policy needs another 5k dialogues to converge. Nevertheless, the number of dialogues required to converge is much smaller than the traditional Q-learning based RL policy which needs approximately 90k~700k dialogues [8]. It shows our dialogue manager with deep Q-learning economically trains a policy with small size of dialogues.

The experimental results suggest that by optimizing the policy in deep RL algorithm, dialogue agent can be trained automatically to successfully complete a dialogue within much shorter turns and to be more robust to SLU error than the rule-based policy. In addition, our proposed model requires even smaller size of train data to learn the best action.

VI. CONCLUSION

In this paper, we propose the dialogue manager by optimizing the dialogue policy using deep Reinforcement Learning algorithm. It shows the deep RL policy is more robust to SLU error and flexible to complex domain of dialogues than the rule-based policy.

ACKNOWLEDGMENT

This research was supported by the MISP (Ministry of Science, ICT & Future Planning), Korea, under the National Program for Excellence in SW) (2015-0-00910) supervised by the IITP (Institute for Information & communications Technology Promotion).

REFERENCES

[1] V. Mnih, et al, "Playing atari with deep reinforcement learning," arXiv:1312.5602, 2013.

[2] R. Sutton and A. Barto. *Reinforcement learning I: An introduction*, ser. Adaptive Computation and Machine Learning. Cambridge, Massachusetts: MIT press, 1998.

[3] C. Watkins and P. Dayan. "Q-learning," *Machine learning*, pp. 279-292, 1992.

[4] M. Gašic, *Statistical dialogue modeling*. Diss. PhD thesis, University of Cambridge, 2011.

[5] S. Jost, et al, "Agenda-based user simulation for bootstrapping a POMDP dialogue system," *Human Language Technologies 2007*. Association for Computational Linguistics, pp. 149-152, 2007.

[6] H. Matthew, B. Thomson and J. D. Williams, "Dialog state tracking challenge 2 & 3 handbook," 2013.

[7] H. Matthew, B. Thomson and J. D. Williams, "The third dialog state tracking challenge." in *Proceedings of IEEE Spoken Language Technology Workshop (SLT)*, 2014.

[8] E. Levin, R. Pieraccini and W. Eckert. "Using markov decision process for learning dialogue strategies," in *Proceedings of the IEEE International Conference*. vol. 1, pp. 201-204, 1998.

Appendices

Appendix I. Slots in DSTC2 (Restaurant Information Domain)

Slot	User may give as a constraint?
area	Yes, 5 possible values.
food	Yes, 91 possible values.
name	Yes, 113 possible values.
pricerange	Yes, 3 possible values.
addr	No
phone	No
postcode	No
signature	No

Appendix II. Slots in DSTC3 (Tourist Information Domain)

Slot	User may give as a constraint?
area	Yes, 15 possible values.
children allowed	Yes, 2 possible values.
food	Yes, 28 possible values.
has internet	Yes, 2 possible values.
has tv	Yes, 2 possible values.
name	Yes, 163 possible values.
near	Yes, 52 possible values.
pricerange	Yes, 4 possible values.
type	Yes, 3 possible values.
addr	No
phone	No
postcode	No
price	No

APPENDIX III. SYSTEM ACTIONS AND CORRESPONDING EXAMPLE UTTERANCES.

Dialogue Act 1	Dialogue Act 2	Dialogue Act 3	Intended Meaning of System's Action
select(pricerange=$x)	select(pricerange=$y)		Do you want an expensive or cheap restaurant?
select(food=$x)	select(food=$y)		Do you want an Indian or French restaurant?
select(area=$x)	select(area=$y)		Do you want a restaurant in south or north?
request(pricerange)			Which price range do you want?
request(food)			Which food do you want?
request(area)			Which area do you want?
request(pricerange)	impl-conf(food=$x)		There are restaurants that serve French food, which price range do you want?
request(pricerange)	impl-conf(area=$x)		You want restaurants in north, which price range do you want?
request(food)	impl-conf(pricerange=$x)		You want expensive food, which food do you want?
request(food)	impl-conf(area=$x)		There are restaurants in East, which food do you want?
request(area)	impl-conf(pricerange=$x)		There are restaurants in East, which price range do you want?
request(area)	impl-conf(food=$x)		You want Korean food, which price range do you want?
request(pricerange)	impl-conf(food=$x)	impl-conf(area=$y)	There are restaurants that serve Chinese food in East, which price range do you want?
request(food)	impl-conf(pricerange=$x)	impl-conf(area=$y)	There are restaurants that serve moderate price range in south, which food do you want?
request(area)	impl-conf(pricerange=$x)	impl-conf(food=$y)	There are restaurants that serve expensive Italian food, which area do you want?
offer(name=$x)			"Tree house" is a good restaurant.
offer(name=$x)	inform(pricerange=$y)		"Tree house" is a moderate priced restaurant.
offer(name=$x)	inform(food=$y)		"Tree house" is a good place and they serve Italian food.
offer(name=$x)	inform(area=$y)		"Tree house" is in west area.
offer(name=$x)	inform(phone=$y)		The phone number of "Tree house" is 010-0000-5059.
offer(name=$x)	inform(postcode=$y)		The postcode of "Tree house" is 157-840.
offer(name=$x)	inform(addr=$y)		"Tree house" is located at Westin Bayshore.
offer(name=$x)	inform(signature=$y)		The signature dish of "Tree house" is Vongole pasta.
offer(name=$x)	inform(pricerange=$y)	inform(food=$z)	"Tree house" serves expensive Italian food.
offer(name=$x)	inform(pricerange=$y)	inform(area=$z)	
offer(name=$x)	inform(pricerange=$y)	inform(phone=$z)	
offer(name=$x)	inform(pricerange=$y)	inform(postcode=$z)	
offer(name=$x)	inform(pricerange=$y)	inform(addr=$z)	
offer(name=$x)	inform(pricerange=$y)	inform(signature=$z)	
offer(name=$x)	inform(food=$y)	inform(area=$z)	
offer(name=$x)	inform(food=$y)	inform(phone=$z)	
offer(name=$x)	inform(food=$y)	inform(postcode=$z)	
offer(name=$x)	inform(food=$y)	inform(addr=$z)	
offer(name=$x)	inform(food=$y)	inform(signature=$z)	
offer(name=$x)	inform(area=$y)	inform(phone=$z)	
offer(name=$x)	inform(area=$y)	inform(postcode=$z)	
offer(name=$x)	inform(area=$y)	inform(addr=$z)	
offer(name=$x)	inform(area=$y)	inform(signature=$z)	
offer(name=$x)	inform(postcode=$y)	inform(phone=$z)	
offer(name=$x)	inform(postcode=$y)	inform(addr=$z)	
offer(name=$x)	inform(phone=$y)	inform(signature=$z)	
offer(name=$x)	inform(phone=$y)	inform(addr=$z)	
offer(name=$x)	inform(signature=$y)	inform(addr=$z)	
expl-conf(pricerange=$x)			You want to find an expensive restaurant, right?
expl-conf(food=$x)			You want to find a Mexica restaurant, right?
expl-conf(area=$x)			You want to find a restaurant in south, right?
canthelp(pricerange=$x)			There is no restaurant serving low priced food.
canthelp(food=$x)			There is no restaurant serving Korean food.
canthelp(area=$x)			There is no restaurant in West
repeat()			I am not sure what you just said, can you repeat that?

APPENDIX IV. EXAMPLE DIALOGUES OF **RULE-BASED POLICY** WITH HIGH SLU ERROR RATE (IN DSTC2 DOMAIN).

Turn No.	**Speaker**	**Dialogue acts (Semantic form)**	**Intended Meaning (Surface form)**
0	System	welcomemsg()	How can I help you ?
1	User	inform(area=centre)	Any restaurant in the centre area?
1	System	request(pricerange)	What pricerange do you want?
2	User	inform(pricerange=moderate)	Moderate.
2	System	request(pricerange)	What pricerange do you want?
3	User	inform(pricerange=moderate)	I want moderate priced restaurant.
3	System	offer(name=sesame restaurant) inform(pricerange=moderate) inform(area=centre)	“Sesame restaurant” is located at centre, with a moderate pricerange.
4	User	request(food)	What kind of food do they serve?
4	System	offer(name=sesame restaurant) inform(food=spanish)	“Sesame restaurant” serves Spanish food.
5	User	inform(food=romanian)	Romanian.
5	System	offer(name=sesame restaurant) inform(food=spanish)	“Sesame restaurant” serves Spanish food.
6	User	inform(food=romanian)	How about Romanian restaurant?
6	System	offer(name=eraina) inform(pricerange=moderate) inform(food=romanian)	“Eraina” serves moderate priced Romanian food.
7	User	request(postcode)	What is the postcode?
7	System	offer(name=eraina) inform(postcode=336-404)	The postcode of “Eraina” is 336-404.
8	User	request(signature)	What is the signature dish?
8	System	offer(name=eraina) inform(signature=flat_bone)	signature dish of “Eraina” is “Flat Bone”.
9	User	request(addr)	What is the address?
9	System	offer(name=eraina) inform(addr=3791 Charles Trace Suite)	The address is 3791 Charles Trace Suite.
10	User	thankyou()	Thanks!
10	System	bye()	Bye!

APPENDIX V. EXAMPLE DIALOGUES OF **DEEP RL POLICY** WITH HIGH SLU ERROR RATE (IN DSTC2 DOMAIN).

Turn No.	Speaker	Dialogue act (Semantic form)	Utterance (Surface form)
0	System	welcomemsg()	How can I help you ?
1	User	inform(pricerange=cheap) inform(food=indian)	I want cheap Indian food.
1	System	offer(name=panahar) inform(area=west)	“Panahar” is a good place in west area.
2	User	inform(area=south)	South area.
2	System	offer(name=the river bar steakhouse) inform(area=east)	“the river bar steakhouse” is in east area.
3	User	inform(area=south)	I want south area.
3	System	canthelp(pricerange=cheap)	I’m sorry, but there is no place in cheap pricerange.
4	User	inform(pricerange=moderate)	How about moderate pricerange?
4	System	canthelp(food=indian)	I’m sorry, but there is no Indian restaurant in this area.
5	User	inform(food=thai)	Find me Thai restaurant.
5	System	canthelp(food=indian)	I’m sorry, but there is no Indian restaurant area.
6	User	inform(food=thai)	Thai restaurant.
6	System	offer(name=venue) inform(food=thai) inform(area=south)	“Venue” is a Thai restaurant in south area.
7	User	request(postcode)	Postcode?
7	System	offer(name=venue) inform(postcode=370-356) inform(signature=picnic_shoulder)	The postcode is 370-356 and their signature dish is “Pinic Shoulder”.
9	User	request(addr)	What is the address?
9	System	offer(name=venue) inform(addr=9558 Ramirez Villages Apt.)	The address is 9558 Ramirez Villages Apt.
10	User	thankyou()	Thankyou!
10	System	bye()	Bye!