A Teacher-Student Framework for Maintainable Dialog Manager

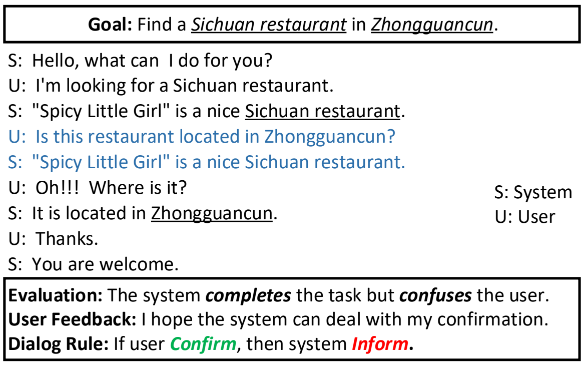
Weikang Wang[1,2], **Jiajun Zhang**[1,2], **Han Zhang**[3],
Mei-Yuh Hwang[3], **Chengqing Zong**[1,2,4] and **Zhifei Li**[3]
[1] National Laboratory of Pattern Recognition, Institute of Automation, CAS, Beijing, China
[2] University of Chinese Academy of Sciences, Beijing, China
[3] Mobvoi AI Lab, Seattle, US
[4] CAS Center for Excellence in Brain Science and Intelligence Technology, Beijing, China
{weikang.wang, jjzhang, cqzong}@nlpr.ia.ac.cn
{hanzhang, mhwang, zfli}@mobvoi.com

Abstract

Reinforcement learning (RL) is an attractive solution for task-oriented dialog systems. However, extending RL-based systems to handle new intents and slots requires a system redesign. The high maintenance cost makes it difficult to apply RL methods to practical systems on a large scale. To address this issue, we propose a practical teacher-student framework to extend RL-based dialog systems without retraining from scratch. Specifically, the "student" is an extended dialog manager based on a new ontology, and the "teacher" is existing resources used for guiding the learning process of the "student". By specifying constraints held in the new dialog manager, we transfer knowledge of the "teacher" to the "student" without additional resources. Experiments show that the performance of the extended system is comparable to the system trained from scratch. More importantly, the proposed framework makes no assumption about the unsupported intents and slots, which makes it possible to improve RL-based systems incrementally.

1 Introduction

With the flourish development of virtual personal assistants (e.g., Amazon Alexa and Google Assistant), task-oriented dialog systems, which can help users accomplish tasks naturally, have been a focal point in both academic and industry research. In the early work, the task-oriented dialog system is merely a set of hand-crafted mapping rules defined by experts. This is referred to as a rule-based system. Although rule-based systems often have acceptable performance, they are inconvenient and difficult to be optimized. Recently, reinforcement learning approaches have been applied to optimize dialog systems through interaction with a user simulator or employed real users online (Gašić et al., 2011; Su et al., 2016a; Li et al., 2016, 2017b). It has been proven that RL-based dialog systems can abandon hand-crafted dialog manager and achieve more robust performance than rule-based systems (Young et al., 2013).



Goal: Find a *Sichuan restaurant* in *Zhongguancun*.

S: Hello, what can I do for you?
U: I'm looking for a Sichuan restaurant.
S: "Spicy Little Girl" is a nice Sichuan restaurant.
U: Is this restaurant located in Zhongguancun?
S: "Spicy Little Girl" is a nice Sichuan restaurant.
U: Oh!!! Where is it?
S: It is located in Zhongguancun.
U: Thanks.
S: You are welcome.

S: System
U: User

Evaluation: The system ***completes*** the task but ***confuses*** the user.
User Feedback: I hope the system can deal with my confirmation.
Dialog Rule: If user ***Confirm***, then system ***Inform***.

Figure 1: An example of a task-oriented dialog after the system comes online. The user is confused because the "*confirm*" intent has not been considered in the deployed system. Dialog rules should be embedded in a new system to handle such situations.

Typically, the first step of building RL-based dialog systems is defining a user model[1] and necessary system actions to complete a specific task (e.g., seek restaurants information or book hotels). Based on such ontology, developers can extract dialog features and train the dialog manager model in an interaction environment. Such systems work well if real users are consistent with the predefined user model. However, as shown in Fig. 1, the unanticipated actions[2] of real users will lead to a poor user experience.

In this situation, the original system should be extended to support new user actions based on user feedback. However, adding new intents or slots will change the predefined ontology. As a consequence, developers need to extract additional

[1] The user model defines what users can do in a dialog system, including domain specific intents and slots.

[2] User actions consist of intents and slots.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3803–3812
Brussels, Belgium, October 31 - November 4, 2018. ©2018 Association for Computational Linguistics

dialog features based on new ontology. Besides, new system actions may be added to deal with new user actions. The network architecture of the new system and the original one will be different. The new system can not inherit the parameters from the old one directly. It will make the original dialog manager model invalid. Therefore, developers have to retrain the new system by interacting with users from scratch. Though there are many methods to train a RL-based dialog manager efficiently (Su et al., 2016a, 2017; Lipton et al., 2017; Chang et al., 2017; Chen et al., 2017), the unmaintainable RL-based dialog systems will still be put on the shelf in real-world applications (Paek and Pieraccini, 2008; Paek, 2006).

To alleviate this problem, we propose a teacher-student framework to maintain the RL-based dialog manager without training from scratch. The idea is to transfer the knowledge of existing resources to a new dialog manager.

Specifically, after the system is deployed, if developers find some intents and slots missing before, they can define a few simple dialog rules to handle such situations. For example, under the condition shown in Fig. 1, a reasonable strategy is to inform the user of the location of this restaurant. Then we encode information of such hand-crafted logic rules into the new dialog manager model. Meanwhile, user logs and dialog policy of the original system can guide the new system to complete tasks like the original one. Under the guidance of the "teacher" (logic rules, user logs, and original policy), we can reforge an extended dialog manager (the "student") without a new interaction environment.

We conduct a series of experiments with simulated and real users on restaurant domain. The extensive experiments demonstrate that our method can overcome the problem brought by the unpredictable user behavior after deployment. Owing to reuse of existing resources, our framework saves time in designing new interaction environments and retraining RL-based systems from scratch. More importantly, our method does not make any assumptions about the unsupported intents and slots. So the system can be incrementally extended once developers find new intents and slots that are not taken into account before. As far as we know, we are the first to discuss the maintainability of deep reinforcement learning based dialog systems systematically.

2 Related Work

Dialog Manager The dialog manager of task-oriented dialog systems, which consists of a state tracker and a dialog policy module, controls the dialog flow. Recently, deep reinforcement learning (Mnih et al., 2013, 2015) has been applied to optimize the dialog manager in an "end-to-end" way, including *deep Q-Network* (Lipton et al., 2017; Li et al., 2017b; Peng et al., 2017; Zhao and Eskenazi, 2016) and *policy gradient* methods (Williams et al., 2017; Su et al., 2016b; Dhingra et al., 2017). RL methods have shown great potential in building a robust dialog system automatically. However, RL-based approaches are rarely used in real-world applications because of the maintainability problem (Paek and Pieraccini, 2008; Paek, 2006). To extend the domain of dialog systems, Gašic et al. (2014) explicitly defined kernel functions between the belief states that come from different domains. However, defining an appropriate kernel function is non-trivial when the ontology has changed drastically. Shah et al. (2016) proposed to integrate turn-level feedback with a task-level reward signal to learn how to handle new user intents. This approach alleviates the problem that arises from the difference between training and deployment phases. But it still fails when the developers have not considered all user actions in advance. Lipton et al. (2017) proposed to use BBQ-Networks to extend the domain. However, similar to Shah et al. (2016), the BBQ-Networks have reserved a few bits in the feature vector for new intents and slots. And system actions for handling new user actions have been considered in the original system design. This assumption is not practical enough. Compared to the existing domain extension methods, our work addresses a more practical problem: new intents and slots are unknown to the original system. If we need to extend the dialog system, we should design a new network architecture to represent new user actions and take new system actions into account.

Knowledge Distillation Our proposed framework is inspired by recent work in knowledge distillation (Bucilu et al., 2006; Ba and Caruana, 2014; Li et al., 2014). Knowledge distillation means training a compact model to mimic a larger teacher model by approximating the function learned by the teacher. Hinton et al. (2015) introduced knowledge distillation to transfer knowledge from

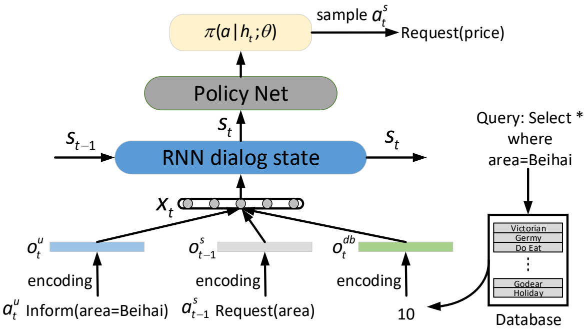


Figure 2: An overview of the RL-based dialog manager used in our work[3]. In the last turn, the system inquires "Where do you want to go?". In current turn, the user input is "Find a restaurant in Beihai.".

a large highly regularized model into a smaller model. The knowledge which can be transferred has not been restricted to models. Stewart and Ermon (2017) proposed to distill the physics and domain knowledge to train neural networks without labeled data. Hu et al. (2016) enabled a neural network to learn simultaneously from labeled instances as well as logic rules. Zhang et al. (2017) integrated multiple prior knowledge sources into neural machine translation using posterior regularization. Our experiments are based on such insights. Through defining appropriate regularization terms, we can distill different knowledge (e.g., trained model or prior knowledge) to a new designed model, alleviating the need for new labeled data or expensive interaction environments.

3 RL-based Dialog Manager

Before going to the details of our method, we provide some background on the RL-based dialog manager in this section. Fig. 2 shows an overview of such dialog manager. We describe each of the parts briefly below.

Feature Extraction At the t-th turn of a dialog, the user input u_t is parsed into domain specific intents and slots to form a semantic frame a_t^u by a language understanding (LU) module. o_t^u and o_{t-1}^s are the one-hot representations of such semantic frames for the current user input and the last system output respectively. Alternatively, o_t^u can be a simple n-grams representation of u_t. But the vocabulary size is relatively large in real-world applications. It will yield slow convergence in the absence of a LU module. Based on the slot-value pair output with the highest probability, a query is sent to a database to retrieve user requested information. o_t^{db} is the one-hot representation of the database result. As a result, the observable information x_t is the concatenation of o_t^u, o_{t-1}^s and o_t^{db}.

State Representation Based on the extracted feature vector x_t and previous internal state s_{t-1}, recurrent neural networks (RNNs) are used to infer the latent representation of dialog state s_t at step t. Current state s_t can be interpreted as the summary of dialog history h_t up to current step.

Dialog Policy Next, the dialog state representation s_t is fed into a policy network. The output $\pi(a|h_t;\theta)$ of the policy network is a probability distribution over a predefined system action set A_s. Lastly, the system samples an action $a_t^s \in A_s$ based on $\pi(a|h_t;\theta)$ and receives a new observation x_{t+1} with an assigned reward r_t. The policy parameters θ can be learned by maximizing the expected discounted cumulative rewards:

$$\mathcal{J}(\theta) = E\left(\sum_{k=0}^{T-t} \gamma^k r_{t+k}\right) \quad (1)$$

where T is the maximal step, and γ is the discount factor. Usually the parameters θ can be iteratively updated by *policy gradient* (Williams, 1992) approach. The policy gradient can be empirically estimated as:

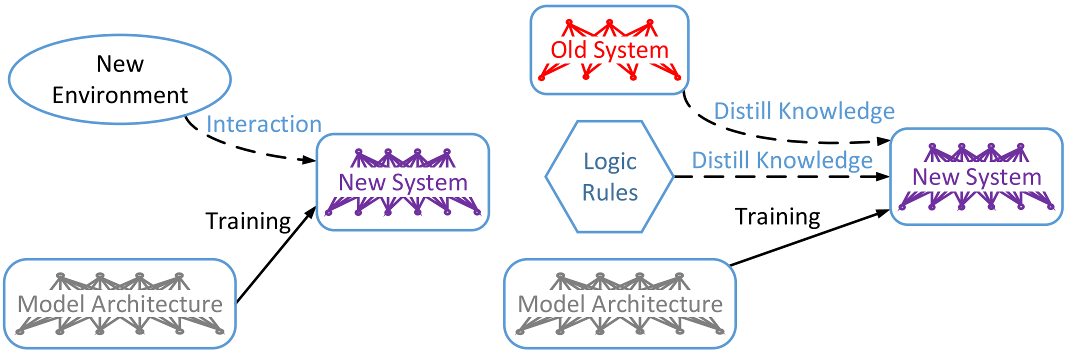
$$\nabla_\theta \mathcal{J}(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta\, log\, \pi(a_{i,t}^s|h_{i,t};\theta)(G_{i,t}-b) \quad (2)$$

where N is the number of sampled episodes in a batch, $G_{i,t} = \sum_{k=0}^{T-t}\gamma^k r_{i,t+k}$ is the sum of discounted reward at step t in the episode i, and b is a baseline to estimate the average reward of current policy.

4 Notations and Problem Definition

Let A_u and A_s denote the supported user and system action sets in the original system design respectively. u_t denotes the user input in the t-th turn. The LU module converts u_t into a domain specific intent and associated slots to form a user action $a_t^u \in A_u$. The system will return an action $a_t^s \in A_s$ according to the dialog manager $\pi(\theta)$. Note that not all user actions are taken into account at the beginning of system design.

[3] Similar dialog model architectures can be found in recent work (Liu and Lane, 2017; Williams et al., 2017; Su et al., 2016b). But designing a dialog model architecture is not our main purpose. We focus on endowing RL-based dialog systems with maintainability and scalability. The dialog model used in our work can be replaced with the similar architectures in the related work.



(a) Retraining a new system from scratch. (b) An overview of our proposed teacher-student framework.

Figure 3: Two kinds of strategies to extend the original system. (a) means redesigning and retraining a new system in an expensive interaction environment from scratch and (b) means transferring knowledge from existing resources to a new system. The network in red indicates the old system based on the original ontology. The networks in gray and purple represent the initialized and trained dialog manager models based on a new ontology respectively. On account of the change in ontology, the extended system has a different network architecture. The dash lines in (a) and (b) show the ability of a new model derives from various sources.

After deployment, the developers can find that some user actions A_{u_new} cannot be handled by the original system based on the human-machine interaction logs D. Generally speaking, A_{u_new} consists of new intents and slots. Our goal is to extend the original system to support the new user action set $A'_u = A_u \cup A_{u\_new}$. The extended dialog manager and new system action set are denoted as $\pi(\theta')$ and A'_s respectively. To handle new user actions, more system actions may be added to the new system. It means that A_s is a subset of A'_s.

5 Approach

Fig. 3 shows two kinds of strategies to extend the original system. The first strategy requires a new interaction environment. However, building a user simulator or hiring real users once the system needs to be extended is costly and impractical in real-world applications. By contrast, our method enhances the reuse of existing resources. The basic idea is to use the existing user logs, original dialog policy model and logic rules ("teacher") to guide the learning process of a new dialog manager ("student"). Without an expensive interaction environment, the developers can maintain RL-based dialog systems as efficiently and straightforwardly as in rule-based systems.

5.1 Distill Knowledge from the Original System

Although the ontology of the new system is different from the original one, the extended dialog manager can still reuse dialog policy of the ill-considered system circuitously. Given user logs D and the original dialog manager $\pi(\theta)$, we define a loss $\mathcal{L}(\theta'; D, \theta)$ to minimize the difference between new dialog manager $\pi(\theta')$ and the old one:

$$\mathcal{L}(\theta'; D, \theta) = \sum_{d \in D} \sum_{t=1}^{|d|} KL(\,\pi(a|h_t;\theta)\,||\,\pi(a|h_t;\theta')\,) \quad (3)$$

where $\pi(a|h_t;\theta)$ and $\pi(a|h_t;\theta')$ are the policy distributions over A_s and A'_s given the same dialog history h_t. $|d|$ means turns of a specific dialog $d \in D$. To deal with unsupported user actions, A_s will be a subset of A'_s. As a result, the KL term in equation (3) can be defined as follows:

$$\begin{aligned} & KL(\,\pi(a|h_t;\theta)\,||\,\pi(a|h_t;\theta')\,) \\ = & \sum_{k=1}^{|A_s|} \pi(a_k|h_t;\theta)\left(log\pi(a_k|h_t;\theta) - log\pi(a_k|h_t;\theta')\right) \end{aligned} \quad (4)$$

As the original policy parameters θ are fixed, the loss function in equation (3) can be rewritten as:

$$\mathcal{L}(\theta'; D, \theta) = -\sum_{d \in D} \sum_{t=1}^{|d|} \sum_{k=1}^{|A_s|} \pi(a_k|h_t;\theta) log\pi(a_k|h_t;\theta') \quad (5)$$

This objective will transfer knowledge of the original system to the "student" at the turn level. Under the guidance of the original system, the extended system will be equipped with the primary strategy to complete a task.

5.2 Distill Knowledge from Logic Rules

It's easy for the developers to give logic rules on the system responses to handle new user actions.

For example, if users ask to confirm a slot, the system should inform the value of that slot immediately. Note that these system actions which handle new user actions may not exist in the old model. It means the architecture of the new system is different from the old one.

We define a set of logic constraints $\mathcal{R} = \{(h_l, a_l)\}_{l=1}^{L}$, where $h_l \in H_{\mathcal{R}}$ indicates the dialog context condition in the l-th rule, and $a_l \in A'_s$ is the corresponding system action. The number of logic rules L is equal to the number of new user actions. These rules can be seen as triggers: if dialog context h_t in current turn t meets the context condition h_l defined in logic rules, then the system should execute a_l. In our work, we use the output of the LU module to judge whether the current dialog context meets the condition defined by logic rules. An alternative method is simple rules matching. To distill the knowledge of rules to a new system, we define a loss function $\mathcal{L}(\theta'; D, \mathcal{R})$ to embed such constraints in the new system:

$$\begin{aligned}\mathcal{L}(\theta'; D, \mathcal{R}) = -\sum_{d\in D}\sum_{t=1}^{|d|}\sum_{h_l\in H_{\mathcal{R}}} \mathbb{1}\{h_t = h_l\} \times \\ \sum_{k=1}^{|A'_s|} \mathbb{1}\{a_k = a_l\}\, log\, \pi(a_k|h_t;\theta')\end{aligned} \tag{6}$$

Where $\mathbb{1}\{\cdot\}$ is the indicate function. Equation (6) suggests the new dialog manager $\pi(\theta')$ will be penalized if it violates the instructions defined by the dialog rules. Note that, for simplicity, we assume these rules are absolutely correct and mutually exclusive. Although this hypothesis may lead to a non-optimal dialog system, these rules define reasonable system actions to corresponding dialog contexts. It implies that the new system can be further refined by reinforcement learning once a new interaction environment is available.

5.3 Extension of Dialog Manager

In the absence of a new training environment, learning is made possible by exploiting structure that holds in the new dialog manager. On one hand, we expect the new system can complete tasks like the original one. On the other hand, it should satisfy the constraints defined by dialog rules. So, the learning objective of new dialog manager $\pi(\theta')$ can be defined as follows:

$$\mathcal{L}(\theta'; D, \theta, \mathcal{R}) = \begin{cases}\mathcal{L}(\theta'; D, \mathcal{R}) & \text{if } h_t \in H_{\mathcal{R}}\,; \\ \mathcal{L}(\theta'; D, \theta) & \text{else}\end{cases} \tag{7}$$

When the dialog context h_t in the t-th turn satisfies a condition defined in $H_{\mathcal{R}}$, we distill knowledge of rules into the new system. Otherwise, we distill knowledge of the original system into the new one. Instead of retraining from scratch, developers can extend RL-based systems by reusing existing resources.

6 Experiments

To evaluate our method, we conduct experiments on a dialog system extension task of restaurant domain.

6.1 Domain

The dialog system provides restaurant information in Beijing. The database we use includes 2988 restaurants. This domain consists of 8 slots (*name*, *area*, *price range*, *cuisine*, *rating*, *number of comments*, *address* and *phone number*) in which the first four slots (*inform slots*) can be used for searching the desirable restaurant and all of these slots (*request slots*) can be asked by users. In each dialog, the user has a goal containing a set of slots, indicating the constraints and requests from users. For example, an *inform slot*, such as "*inform(cuisine=Sichuan cuisine)*", indicates the user finding a Sichuan restaurant, and a *request slot*, such as "*request(area)*", indicates the user asking for information from the system (Li et al., 2016, 2017b; Peng et al., 2017).

6.2 Measurements

A main advantage of our approach is that the unconsidered user actions can be handled in the extended system. In addition to traditional measurements (e.g., success rate, average turns and average reward), we define an objective measurement called "Satis." (user satisfaction) to verify this feature in the simulated evaluation. "Satis." indicates the rate at which the system takes reasonable actions in unsupported dialog situations. It can be calculated as follows:

$$Satis. = \frac{\sum_{d\in D}\sum_{t=1}^{|d|}\sum_{l=1}^{L}\mathbb{1}\{h_t = h_l\}\mathbb{1}\{a_t^s = a_l\}}{\sum_{d\in D}\sum_{t=1}^{|d|}\sum_{l=1}^{L}\mathbb{1}\{h_t = h_l\}} \tag{8}$$

where h_t and a_t^s are the dialog history and system action in the t-th turn, h_l and a_l are dialog context condition and corresponding system action defined in the l-th rules. Intuitively, an unreasonable system reply will frustrate users and low "Satis." indicates a poor user experience.

	*Sim*1				*Sim*2			
LU Error Rate	Succ.	Turn	Reward	Satis.	Succ.	Turn	Reward	Satis.
0.00	0.962	13.6	3.94	-	0.901	13.2	2.95	0.57
0.05	0.937	13.7	3.41	-	0.877	14.4	2.41	0.48
0.10	0.910	14.3	2.65	-	0.841	13.9	1.41	0.47
0.20	0.845	15.2	0.58	-	0.784	14.7	0.01	0.47

Table 1: Performance of the original system when interacting with different user simulators. LU error means simulating slot errors and intent errors in different rates. Succ.: success rate, Turn: average turns, Reward: average reward.

LU Error Rate	0.00	0.05	0.10	0.20
Total	25857	26166	27077	28385
New User Actions	1853	1859	1998	1912

Table 2: Statistics of turns when S_1 interacts with *Sim*2.

Although "Satis." is obtained based on our handcrafted dialog rules, it approximately measures the subjective experience of real users after system deployment.

6.3 User Simulator

Training RL-based dialog systems requires a large number of interactions with users. It's common to use a user simulator to train RL-based dialog systems in an online fashion (Pietquin and Dutoit, 2006; Scheffler and Young, 2002; Li et al., 2016). As a consequence, we construct an agenda-based user simulator, which we refer to as *Sim*1, to train the original RL-based system. The user action set of *Sim*1 is denoted as A_u, which includes such intents[4]: "*hello*", "*bye*", "*inform*", "*deny*", "*negate*", "*affirm*", "*request*", "*reqalts*" and "*null*". The slots of *Sim*1 are shown in section 6.1. In each turn, the user action consists of a intent and slots and we append the value of slots according to the user goal.

6.4 Implementation of the Original System

For the original RL-based dialog system, a feature vector x_t of size 191 is extracted. This vector is the concatenation of encodings of LU results, the previous system reply, database results and the current turn number. The LU module is implemented with an SVM[5] for intent detection and a CRF[6] for slot filling. The language generation module is implemented by a rule-based method. The hidden dialog state representation is inferred by a GRU (Chung et al., 2014). We set the hidden states of the GRU to be 120. The policy network is implemented as a Multilayer Perceptron (MLP) with one hidden layer. The size of the hidden layer is 80. The output dimension of policy network is 15, which corresponds to the number of system actions. To encourage shorter interaction, we set a small per-turn negative reward $R_{turn} = -1$. The maximal turn is set to be 40. If the user goal is satisfied, the policy will be encouraged by a large positive reward $R_{succ} = 10$; otherwise the policy will be penalized by a negative reward $R_{fail} = -10$. Discounted factor $\gamma = 0.9$. The baseline b of current policy is estimated on sampled episodes in a batch. The batch size N is set to be 32. Adadelta (Zeiler, 2012) method is used to update model parameters. The **original system** S_1 is trained by interacting with *Sim*1. After about 2400 interactions, the performance of S_1 starts to converge.

6.5 Simulated Evaluation

To evaluate our approach, we design another user simulator, which we denote as *Sim*2, to simulate the unpredictable real customers. The user action set of *Sim*2 is denoted as A'_u. The difference between A_u and A'_u is reflected on the domain specific intents[7]. Specifically, in addition to the intents of *Sim*1, A'_u includes the "*confirm*" intent. The difference in user action sets will result in different interaction strategies between *Sim*1 and *Sim*2. To verify whether a recommended restaurant meets his (her) constraints, *Sim*1 can **only** request what the value of a specific slot is, but *Sim*2 can request or confirm.

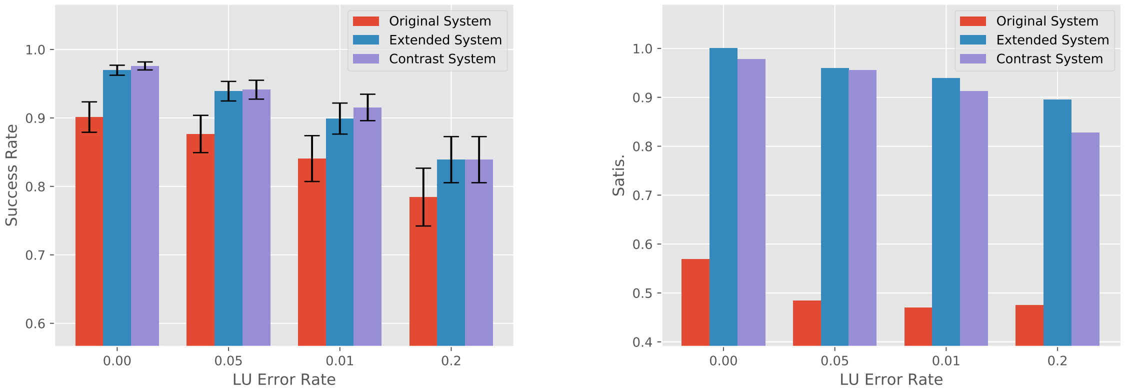
After obtaining the original system S_1, we deploy it to interact with *Sim*1 and *Sim*2 respectively, under different LU error rates (Li et al., 2017a). In each condition, we simulate 3200 episodes to obtain the performance. Table 1 shows the

[4] A detail explanation of these intents is in DSTC2 (Henderson et al., 2013).

[5] We use the publicly available SVM tool at http://scikit-learn.org.

[6] We use the publicly available CRF tool at https://pypi.python.org/pypi/sklearn-crfsuite.

[7] A more complex situation is shown in the human evaluation.



(a) Dialog success rate of different systems.

(b) Satis. of different systems.

Figure 4: Performance of different systems under simulation. The original, extended and contrast systems are shown in red, blue and purple bars. We test these systems by interacting with *Sim*2.

details of the test performance. Table 2 shows the statistics of turns when S_1 interacts with *Sim*2.

As shown in Table 1, S_1 achieves higher dialog success rate and rewards when testing with *Sim*1. When interacting with *Sim*2, nearly half of the responses to unsupported user actions are not reasonable. Notice even though *Sim*2 contains new user actions, some of the new actions might be appropriately handled by S_1. It may be due to the robustness of our RL-based system. But it's far from being desired. The unpredictable real user behavior in the deployment stage will lead to a poor user experience in real-world applications. It proves the importance of a maintainable system.

To maintain the original system, we define a few simple logic rules to handle unsupported user actions: if users confirm the value of a slot in current turn, the system should inform users of that value. These rules[8] are intuitive and reasonable to handle queries such as "Is this restaurant located in Zhongguancun?". There are four slots[9] that can be used for confirmation, so we define four logic rules in all. Due to the change in ontology, we add a new status in dialog features to represent the "confirm" intent of users. It leads to a change in the model architecture of extended dialog manager. Then we distill knowledge of the S_1 and logic rules into the **extended system**. No additional data is used to obtain the extended system.

For comparison, we retrain another new system (**contrast system**) from scratch by interacting with *Sim*2. After about 2600 interactions with *Sim*2, the performance of contrast system starts to converge. Note that in order to build the contrast system, the developers need to redesign a new user simulator or hire real users. It's expensive and impractical in industrial applications. Then we simulate 3200 interactions with *Sim*2 to obtain its performance. Fig. 4 illustrates the performance of different systems.

As can be seen, the extended system performs better than the original system in terms of dialog success rate and "Satis.". This is to a large degree attributed to the consideration of new user actions. Fig. 4(a) shows that the contrast system achieves higher dialog success rate than the extended system. But the gap is negligible. However, the contrast system is trained from scratch under a new interaction environment and the extended system is trained by transferring knowledge of the original system and logic rules. To train the contrast system, about 2600 episodes are sampled by interacting with a new interaction environment. But no additional data is used to train the extended system.

In Fig. 4(b), the "Satis." of the extended system is slightly higher than the contrast system. This is due to the fact that the extended system learns how to deal with new user actions from logic rules but the contrast system obtains dialog policy by exploring the environment. As a result, the contrast system learns a more flexible dialog policy than the extended system[10]. However, the "Satis." has a bias to the suboptimal rules,

[8] In the practical dialog system, we can inject more complex logic rules and take dialog history into account. These rules are not limited to question/answer mapping.

[9] They are "*name*", "*area*", "*price range*" and "*cuisine*".

[10] Table 6 in the Appendix shows sample dialogs from the extended system and contrast system.

Average Rating	2.35
Average Turns	12.1
Unseen *Intents*	takeTaxi, bookTable
Unseen *Inform Slots*	withoutWaiting, carryoutService, goodforDating, privateRoom, hasInternet
Unseen *Request Slots*	waitingLine, discount, businessHours
Unseen *Confirm Slots*	carryoutService, goodforDating, privateRoom, hasInternet

Table 3: Details of the real user logs. Users are encouraged to interact with the original system by unsupported intents and slots. We find there are 14 user actions unseen before.

Dialog Condition	System Action
takeTaxi	API call
bookTable	API call
inform unseen slots	recommend a restaurant
request unseen slots	offer information of slots
confirm unseen slots	offer information of slots

Table 4: Different types of rules for new user actions. Left column shows the dialog context condition; Right column shows the corresponding system action. We define 14 rules in all to handle newfound intents and slots shown in Table 3.

rather than the optimal policy gained from the environment. It suggests the extended system can be further refined by reinforcement learning once a new interaction environment is available.

6.6 Human Evaluation

In any case, the developers can't guarantee all user actions are considered. Fortunately, our method makes no assumptions about the new user actions and new dialog model architecture. As a result, the system can be extended over multiple iterations.

To evaluate this characteristic, we deploy the extended system[11] in section 6.5 to interact with real human users. Users are given a goal sampled from our corpus for reference. To elicit more complex situations, they are encouraged to interact with our system by new intents and slots related to the restaurant domain. At the end of each dialog, they are asked to give a subjective rating on the scale from 1 to 5 based on the naturalness of the system (1 is the worst, 5 is the best.). After filtering dialog sessions unrelated to our task, we collect 315 episodes in total. Table 3 shows the details of the user logs. As shown in Table 3, after deployment, there are a few slots and intents unseen before. For example, users may ask for the discount information or take a taxi to the restaurant. To represent the new intents and slots, the dimension of extracted dialog features is extended to 236. Meanwhile, the number of system actions is extended to 29 to handle new user actions. To deal with the newfound user actions, we define 14 rules in total. Table 4 shows the details of new defined logic rules. Then we distill the knowledge of the original system and logic rules into a new system. Fig. 5 shows the comparison in user ratings. The extended system significantly gets a higher subjective rating than the original one. It proves that the extended system can give reasonable responses in unseen dialog situations.

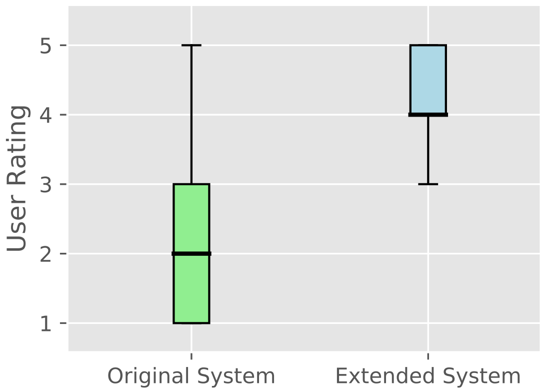


Figure 5: Distribution of user ratings.

Table 5 shows sample dialogs from the original system and extended system with real users. We can see that the extended system is much more coherent since it takes new user actions into account. It inspires us that a complicated RL-based dialog system can start from a simple one and be improved incrementally based on our framework. This design pattern will be much more efficient if the interaction environment is hard to access.

7 Conclusion

In this paper, we present a practical solution to maintain RL-based dialog systems without retraining from scratch. By reusing existing resources, developers can extend the RL-based dialog manager to deal with unpredictable user actions after deployment. Furthermore, our method holds no constraints on the architecture of the new system and retains the key benefit of data-driven methods. It means the new system can be maintained iteratively and further refined by RL once a new interaction environment is available. The sim-

[11] The extended system in the simulated evaluation will be the original system in our human evaluation.

Original System	Extended System
User: I want to find a Sichuan restaurant. *System*: Where do you want to go? *User*: Zhongguancun area. *System*: “Hot Charge” is a nice Sichuan restaurant in Zhongguancun area. *User*: **Can you tell me the discount?** *System*: **The average cost is 42 CNY.** *User*: **Tell me the discount!** *System*: **“Hot Charge” is a nice Sichuan restaurant in Zhongguancun area.** *User*: Bye. *System*: Bye.	*User*: I want to find a Sichuan restaurant in Zhongguancun area. *System*: “Hot Charge” is a nice Sichuan restaurant in in Zhongguancun area. *User*: **What’s the discount of this restaurant?** *System*: **This restaurant is offering 10% off.** *User*: **Is there a waiting line?** *System*: **Currently there are 5 parties waiting in line.** *User*: **Can you book a table for me?** *System*: **Ok.** *User*: Thanks. *System*: You are welcome.
Subjective Rating = 2	Subjective Rating = 5

Table 5: Sample dialogs of the original and extended systems in the human evaluation. The original system will collapse when encountering unseen slots and intents. After extension, the new system can handle more user actions.

ulation and human experiments show our proposed framework guarantees the maintainability and scalability in RL-based systems, which are necessary for any industrial application.

8 Acknowledgments

The research work described in this paper has been supported by the National Key Research and Development Program of China under Grant No. 2017YFB1002103 and also supported by the Natural Science Foundation of China under Grant No. 61333018.

References

Jimmy Ba and Rich Caruana. 2014. Do deep nets really need to be deep? In *Advances in neural information processing systems*, pages 2654–2662.

Cristian Bucilu, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541. ACM.

Cheng Chang, Runzhe Yang, Lu Chen, Xiang Zhou, and Kai Yu. 2017. Affordable on-line dialogue policy learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2190–2199.

Lu Chen, Xiang Zhou, Cheng Chang, Runzhe Yang, and Kai Yu. 2017. Agent-aware dropout dqn for safe and efficient on-line dialogue policy learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2444–2454.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2017. Towards end-to-end reinforcement learning of dialogue agents for information access. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 484–495.

Milica Gašić, Filip Jurčíček, Blaise Thomson, Kai Yu, and Steve Young. 2011. On-line policy optimisation of spoken dialogue systems via live interaction with human subjects. In *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*, pages 312–317. IEEE.

Milica Gašic, Dongho Kim, Pirros Tsiakoulis, Catherine Breslin, Matthew Henderson, Martin Szummer, Blaise Thomson, and Steve Young. 2014. Incremental on-line adaptation of pomdp-based dialogue managers to extended domains. *In Proceedings on InterSpeech.*

Matthew Henderson, Blaise Thomson, and Jason Williams. 2013. Dialog state tracking challenge 2 & 3.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531.*

Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. 2016. Harnessing deep neural networks with logic rules. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2410–2420.

Jinyu Li, Rui Zhao, Jui-Ting Huang, and Yifan Gong. 2014. Learning small-size dnn with output-distribution-based criteria. In *Fifteenth Annual Conference of the International Speech Communication Association.*

Xiujun Li, Yun-Nung Chen, Lihong Li, Jianfeng Gao, and Asli Celikyilmaz. 2017a. 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*.

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

Zachary Lipton, Xiujun Li, Jianfeng Gao, Lihong Li, Faisal Ahmed, and Li Deng. 2017. Bbq-networks: Efficient exploration in deep reinforcement learning for task-oriented dialogue systems. *arXiv preprint arXiv:1711.05715*.

Bing Liu and Ian Lane. 2017. Iterative policy learning in end-to-end trainable task-oriented neural dialog models. *arXiv preprint arXiv:1709.06136*.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. In *NIPS Deep Learning Workshop*.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Tim Paek. 2006. Reinforcement learning for spoken dialogue systems: Comparing strengths and weaknesses for practical deployment. In *Proc. Dialog-on-Dialog Workshop, Interspeech*.

Tim Paek and Roberto Pieraccini. 2008. Automating spoken dialogue management design using machine learning: An industry perspective. *Speech communication*, 50(8):716–729.

Baolin Peng, Xiujun Li, Lihong Li, Jianfeng Gao, Asli Celikyilmaz, Sungjin Lee, and Kam-Fai Wong. 2017. Composite task-completion dialogue policy learning via hierarchical deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2231–2240.

Olivier Pietquin and Thierry Dutoit. 2006. A probabilistic framework for dialog simulation and optimal strategy learning. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(2):589–599.

Konrad Scheffler and Steve Young. 2002. Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. In *Proceedings of the second international conference on Human Language Technology Research*, pages 12–19. Morgan Kaufmann Publishers Inc.

Pararth Shah, Dilek Hakkani-Tür, and Larry Heck. 2016. Interactive reinforcement learning for task-oriented dialogue management. In *NIPS 2016 Deep Learning for Action and Interaction Workshop*.

Russell Stewart and Stefano Ermon. 2017. Label-free supervision of neural networks with physics and domain knowledge. In *AAAI*, pages 2576–2582.

Pei-Hao Su, Pawel Budzianowski, Stefan Ultes, Milica Gasic, and Steve Young. 2017. Sample-efficient actor-critic reinforcement learning with supervised data for dialogue management. *arXiv preprint arXiv:1707.00130*.

Pei-Hao Su, Milica Gasic, Nikola Mrkšić, Lina M Rojas Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016a. Online active reward learning for policy optimisation in spoken dialogue systems. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2431–2441.

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

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 (Volume 1: Long Papers)*, volume 1, pages 665–677.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

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

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Jiacheng Zhang, Yang Liu, Huanbo Luan, Jingfang Xu, and Maosong Sun. 2017. Prior knowledge integration for neural machine translation using posterior regularization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1514–1523.

Tiancheng Zhao and Maxine Eskenazi. 2016. Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. In *17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, page 1.