Efficient Dialog Policy Learning by Reasoning with Contextual Knowledge

Haodi Zhang[1], Zhichao Zeng[1], Keting Lu[2], Kaishun Wu[1], Shiqi Zhang[3]

[1] Computer Science and Software Engineering, Shenzhen University
[2] Baidu, Inc.
[3] Computer Science, SUNY Binghamton
{hdzhang, cengzhichao2019, wu}@szu.edu.cn, ktlu@mail.ustc.edu.cn, zhangs@binghamton.edu

Abstract

Goal-oriented dialog policy learning algorithms aim to learn a dialog policy for selecting language actions based on the current dialog state. Deep reinforcement learning methods have been used for dialog policy learning. This work is motivated by the observation that, although dialog is a domain with rich contextual knowledge, reinforcement learning methods are ill-equipped to incorporate such knowledge into the dialog policy learning process. In this paper, we develop a deep reinforcement learning framework for goal-oriented dialog policy learning that learns user preferences from user goal data, while leveraging commonsense knowledge from people. The developed framework has been evaluated using a realistic dialog simulation platform. Compared with baselines from the literature and the ablations of our approach, we see significant improvements in learning efficiency and the quality of the computed action policies.

1 Introduction

Artificial intelligence (AI) research in its early years was largely driven by the representation of and reasoning with human knowledge. Those algorithms and systems can use existing human knowledge to generate "new knowledge", or draw conclusions for different reasoning purposes, such as inference, diagnosis, and planning (Davis and Marcus 2015). Learning methods have shown great promises, thanks to the large-scale datasets and the massive computational power (LeCun, Bengio, and Hinton 2015). Given all the successes from (knowledge-based) reasoning and (data-driven) learning methods, there is the recent trend of research, including this work, that integrates the complementary advantages of knowledge-based and data-driven methods.

A typical dialog system includes at least the components for dialog state tracking, dialog management, and language synthesis (Young et al. 2013), while there are end-to-end dialog systems that integrate the components using deep neural networks (Bordes, Boureau, and Weston 2016; Serban et al. 2016). Dialog management focuses on selecting dialog actions based on the current dialog state toward achieving long-term dialog goals, e.g., reserving hotel rooms, and purchasing flight tickets. The dialog actions can be realized using language synthesis methods, and the current dialog state can be estimated using dialog state tracking methods. We focus on dialog management in this research.

Reinforcement learning (RL) algorithms enable agents to learn from trial-and-error experience (Sutton and Barto 2018), and have been applied to learning policies for dialog management (Singh et al. 2002; Schatzmann et al. 2006). On the one hand, existing RL-based dialog policy learning methods highly rely on the "rewards" from dialogs, which tend to be very sparse, unreliable, and potentially expensive. On the other hand, it frequently happens that there is widely available commonsense knowledge in dialog domains. For instance, "*rooms of 5-star hotels are usually expensive*," and "*many Chinese restaurants are located in Chinatowns*." However, it is very difficult to incorporate such knowledge into dialog policy learning, due to their fundamentally different paradigms of computation. The wide availability of contextual knowledge and highly sparse feedback in dialog domains together serve as the main motivation of this research.

In this paper, we develop a dialog agent that, for the first time, incorporates declarative human knowledge into deep RL-based dialog policy learning. In particular, the knowledge from people is used for reasoning about both internal and external dialog state factors. *Internal factors* are those for forming the dialog state space, e.g., *star* and *price* in hotel booking. *External factors* are those that are not modeled in dialog states but can be potentially useful. For instance, people prefer local hotels in bad weather, where *weather* is external. We use first-order logic for the representation of the dependencies between internal factors, and use Markov logic network (MLN) (Richardson and Domingos 2006) algorithms to learn the rules' weights for inference purposes. We use Answer set programming (Brewka, Eiter, and Truszczyński 2011) for the representation of and reasoning with human knowledge about external factors.

We have extensively conducted experiments using a realistic dialog platform PyDial (Ultes et al. 2017). Compared with baselines from the literature and ablations of our own approach, we observe significant improvements in dialog learning efficiency and policy quality.

2 Background

In this section, we briefly introduce the three building blocks of this work, namely dialog management with Deep rein-

forcement learning (DRL), Markov logic network (MLN), and Answer set programming (ASP).

2.1 Dialog Management with DRL

Dialog management can be modeled as a Markov decision problem (MDP) (Young et al. 2013), with a continuous belief state space $\mathcal{S}$, a finite set of action $\mathcal{A}$ and a reward function $\mathcal{R}$. A state s in $\mathcal{S}$ indicates the dialog status, including the distributions of slots, previous actions of the agent and the user and other necessary information. An action a in $\mathcal{A}$ is decided by a dialog policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$, and then executed by the agent. After the execution of some decided action a, the agent is rewarded or penalized according to the reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow R$. The objective of dialog management is to find an optimal policy π^* that maximizes the cumulative reward in dialogs, $R_t = \sum_{i=t}^{\infty} \gamma^{i-t} r_i$, where γ is a discount factor.

The cumulative reward can be approximated by a DQN (Mnih et al. 2015), which is a model-free reinforcement learning method. A neural network with parameters θ is used to approximate the Q-function (expected cumulative reward), $Q^*(s, a) = Q(s, a; \theta)$. The action is decided in a greedy way, either $\pi_Q(a) = \text{argmax}_{a \in \mathcal{A}} Q(s, a; \theta)$, or being a random exploration in probability ε.

In this work, we use deep reinforcement learning such as DQN as the infrastructure of dialog policy learning. Learning dialog policies interactively via DRL has emerged as a popular approach (Lipton et al. 2016; Dhingra et al. 2017; Zhao and Eskenazi 2016; Wu et al. 2019; Lu, Zhang, and Chen 2019). Compared to rule-based methods, DQN and other DRL algorithms do not require lots of domain-specific handcrafting.

2.2 Markov Logic Network

A Markov logic network (MLN) (Richardson and Domingos 2006) is a probabilistic logic which applies the ideas of a Markov network to first-order logic. MLN supports learning and inference under uncertainty to find the most likely distribution of the system. Given a set of first-order logic (FOL) formulas as a knowledge base, MLN gives the distributions of the groundings of each predicates. The distributions are calculated according to the weights of the FOL formulas,

$$Pr(X = x) = \frac{1}{Z} exp(\sum_i w_i f_i(x)) \tag{1}$$

where Z is known the partition function, the w_i is the weight of formula i, and f_i is the binary feature function. The weights and features can be learned from a given relational database by iteratively optimizing a pseudo-likelihood measure.

We use MLN for the specification of weighted rules about the factors in dialog states. The rule weights as prior knowledge are learned from historical data, and then used during the dialog policy learning. Some recent work (Zhang et al. 2020) combines MLN with graph neural networks (GNN), to improve MLN's inference efficiency. But as prior knowledge in dialog policy learning, MLN should not contain a large amount of rules and entities. So we just use MLN inference directly instead of deploying GNN as the inference network.

2.3 Answer Set Programming

Answer set programming (ASP) is a form of declarative programming (Brewka, Eiter, and Truszczyński 2011) based on stable model (answer set) semantics for logic programming (Gelfond and Lifschitz 1988), a nonmonotonic reasoning paradigm in knowledge representation. In ASP, search problems are reduced to computing stable models by answer set solvers, e.g., Clingo (Gebser et al. 2014). Syntactically, an ASP program consists of a set of rules of following form,

$$h_1 \; or \; \ldots \; or \; h_k \leftarrow b_1, \ldots, b_m, not \; b_{m+1}, \ldots, not \; b_n. \tag{2}$$

where h_i, b_i are literals, and not is negation as failure (default negation), which represents that there is no evidence that some literal is true. A stable model or an answer set of a logic program is defined as a fix point, namely, a set of literals X is an answer set of a logic program Π if X is a minimal model of the reduction program Π^X of Π w.r.t. X.

Traditionally, ASP reasons about discrete-valued symbols and does not quantify uncertainty explicitly. But in some works such as P-log (Baral, Gelfond, and Rushton 2009), RIL (Zhang et al. 2019), NeurASP (Yang, Ishay, and Lee 2020) and DILOG (Zhou et al. 2020), the paradigm is extended to allow probability atoms, and integrated with subsymbolic approaches. In this paper, we just use ASP without probabilities, to characterize and reason about some contextual factors that are not modeled in dialog states. In general, checking the existence of stable models is NP-hard, but some heuristic algorithms and available solvers are capable of solving large-scale problem instances in ASP efficiently.

3 Methodology

In this section, we propose a framework for dialog policy learning by reasoning with contextual knowledge. The framework integrates knowledge representation and reasoning (KRR) and model-free RL, and is illustrated in Figure 1. The KRR part includes logical-probabilistic reasoning (MLN) and nonmonotonic reasoning (ASP). These two components cooperate with deep reinforcement learning (DRL) in dialogs.

In each dialog turn, DRL decides which system action to execute. By executing the actions, the agent receives a big reward in successful dialogs, and has a small cost in each turn. Meanwhile, MLN and ASP components collect facts about internal and external factors, and reason with them to help DRL make better decisions. The use of MLN and ASP for knowledge representation and reasoning is motivated by two types of contextual knowledge in a dialog.

3.1 Internal and External Knowledge

In this paper, we consider knowledge about both internal factors and external factors. **Internal factors** are those for forming the dialog state space, and **external factors** are those that are not modeled in dialog states but can be potentially useful. For instance, the internal factors in a hotel

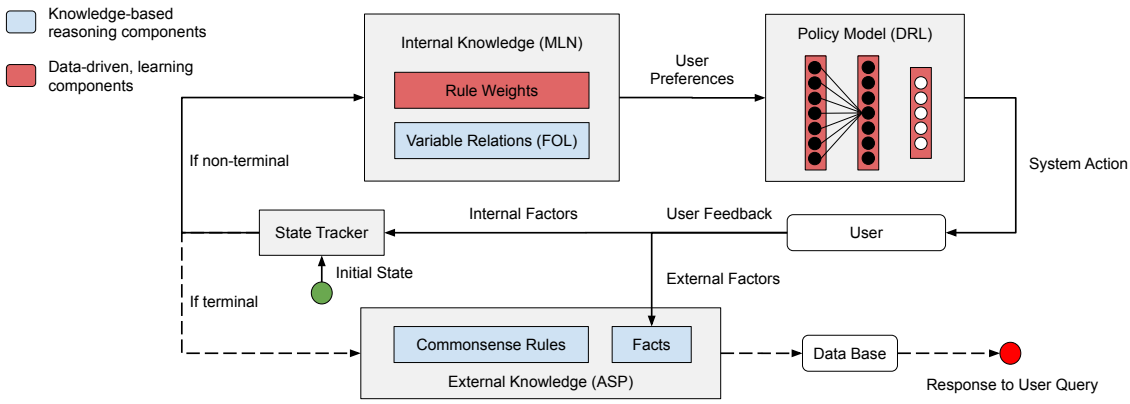


Figure 1: Overview of the framework. The framework mainly consists of three components, namely, internal knowledge (MLN-based), external knowledge (ASP-based) and dialog policy learning (DRL-based). The MLN-based probabilistic reasoner collects facts about internal factors, and provides user preferences to DRL for system action generation. The ASP-based logical reasoner collects facts about external factors, and reasons for potentially better service when the dialog terminates. Solid lines indicate the data flow, and dashed lines represent the function calls after a dialog terminates.

booking domain may include *price*, *star of the hotel*, etc. The external factors can be *weather*, *current location of the user*, etc. In the rest of the paper, we refer to the knowledge about the two kinds of factors as *internal knowledge* and *external knowledge*, respectively.

As the uncertainty of the internal factors is quantified by deep RL, it is natural to use logical-probabilistic reasoning to represent the internal knowledge. We use Markov logic network (MLN) to represent probabilistic knowledge such as "*95% rooms of 5-star hotels are expensive*," or "*75% users who order guest houses also require low prices*." Syntactically, MLN is a collection of weighted first-order logical formulas. The weights of the formulas are learnable from historical data.

We use ASP to represent and reason with external knowledge, for instance, "*it is a bad experience to travel in distance in a bad weather*." With that particular knowledge, the agent should avoid booking a hotel or a restaurant in distance when the weather is bad, if the user's original location is available. Factors like *weather* and *user's location* may not be modeled in a dialog state, but are potentially useful. As a nonmonotonic reasoning paradigm, ASP is good at representing dynamic domains and characterizing default facts. For example, the following ASP rule

$$suggest\_area(X) \leftarrow weather(bad), user\_loc(X), \\ not\ insist\_area(Y). \quad (3)$$

indicates that if the user's original location is X and the weather is bad, the agent should suggest some restaurant in the same area X, unless the user insists to have dinner in some area Y. The reasoning framework still works even when the information about *insist_area* is missing. As a solution of the *Frame Problem* (McCarthy and Hayes 1981) nonmonotonic reasoning such as ASP does not require monitoring all predicates. However, if the rule is written in classical logic, to reason about *suggest_area*, the assignment of predicate *insist_area* has to be known. We now formally introduce the algorithm.

Algorithm 1: Dialog policy learning by reasoning with contextual knowledge

Require: Internal knowledge K_I (MLN) , external knowledge K_E (ASP)

1: Train the rule weights of K_I with historical data
2: Initialize DRL model with random weights and empty replay
3: **repeat**
4: Initialize a new dialog d
5: Update dialog state s of d with K_I ▷ call Algorithm 2
6: **for** each dialog turn in d **do**
7: Select an action a with DRL model
8: Execute a, and collect reward r
9: Collect user's feedback, and track next state s'
10: Store tuple (s, a, s', r) in replay buffer
11: **if** s' is not a terminal state **then**
12: Collect facts F_E about E, and add F_E into K_E
13: Collect facts F_I about I
14: Update s with K_I and F_I ▷ call Algorithm 2
15: **else**
16: Reason with K_E ▷ call Algorithm 3
17: **end if**
18: Update dialog state $s \leftarrow s'$
19: **end for**
20: Update DRL model with samples from replay buffer
21: **until** Convergence

3.2 Algorithm Description

The main algorithm requires an internal knowledge base K_I (MLN-based) and an external knowledge base K_E (ASP-based), as illustrated in Algorithm 1, where I and E are internal and external factor sets, respectively. These two components help DRL model, e.g. DQN, make better decisions in dialog turns. The algorithm consists of the following two steps.

1. Train the weights of the rules in K_I with historical data (Line 1).
2. Initialize and repeatedly train DQN in dialogs, with trained K_I and K_E, until it converges (Lines 2-21).

Algorithm 2: State update by MLN

Require: State s to update, internal knowledge K_I, facts F_I about I

1: Let $I_{ev} = \{i \mid i \in I, i \text{ occurs in } F_I\}$ ▷ set of evidence factors
2: **for** $i \in I \backslash I_{ev}$ **do**
3: update the state s by $\mathbf{P}^s_{\sim i} \leftarrow \delta(i, K_I, F_I)$
4: **end for**
5: **return** s

Algorithm 3: Reasoning by ASP

Require: External knowledge K_E, database db

1: Compute the set AS of all answer sets of K_E
2: Query whether answer sets in AS are satisfiable in db
3: Let AS_{st} be the query result: $AS_{st} = \{x \mid x \in AS, \exists entity \in db : entity \models x\}$ ▷ set of satisfiable answer sets
4: **if** AS_{st} is not empty **then**
5: Override agent's service with AS_{st}
6: **end if**

The training process within one dialog is as follows. In Lines 4-5, once a new dialog d is initialized, the dialog state s of d is updated by MLN component K_I . K_I has already learned the statistical user preferences from historical data, and the knowledge is useful for characterizing the dialog state. Notice that here (Line 5) Algorithm 2 is called with an empty set of facts ($F^I = \emptyset$), where F^I will be augmented in later steps during the dialog.

In Lines 7-11, in each turn of the dialog d, DQN selects an action a for each dialog turn in ε-greedy way, and then executes it. After the execution, the agent collects the user's feedback and the reward r, tracks the next state s', stores the tuple (s, a, s', r) in experience replay, and updates dialog state $s \leftarrow s'$. This process is the same as standard DQN-based dialog systems.

In Lines 12-18, for each non-terminal dialog turn, the agent collects facts about the internal and external factors, F_I and F_E, respectively. The former is immediately used to update dialog state s by MLN (Algorithm 2), and the latter is continually added into the ASP program for reasoning purposes until the dialog terminates (Algorithm 3).

In the following, we introduce in detail the MLN and ASP components.

State Update by MLN (Algorithm 2) The MLN component consists of variable relations in form of first-order logical (FOL) rules and the rule weights associated with them. The rules in FOL characterize the dependencies of internal factors. For instance, the postcode is decided by the address

$$address = x \supset postcode = y,$$

and the price of a hotel is related with its star and location,

$$star = x \wedge area = y \supset price = z.$$

The weight associated with each FOL rule indicates the importance of the rule, and is learnable from historical data. After pretraining, the weighted rules of MLN are used for internal knowledge inference. In each turn, MLN component collects facts about internal factors from the state tracker, and then infers with them to find out the user's preferences. The information about user preferences helps precisely characterize the dialog state and generate better system actions.

The state update by pretrained internal knowledge is illustrated in Algorithm 2. The MLN component uses K_I and collected facts F_I about I to update the dialog state. The facts in F_I are also called *evidence atoms* for MLN. We call those factors that are mentioned in F_I *evidence factors*. The set of all evidence factors is denoted by I_{ev},

$$I_{ev} = \{i \mid i \in I, i \text{ occurs in } F_I\}. \quad (4)$$

With the facts F_I, MLN infers about other internal factors in I and updates dialog state s, with function δ in Line 3 of Algorithm 2. Specifically, the function updates distribution of i in s, $\mathbf{P}^s_{\sim i}$, according to F_I and the weighted rules in K_I.

Definition 1 *Given an internal factor $i \in I$, a dialog state s, pretrained internal knowledge K_I, and evidence atoms F_I, the* update function $\delta(i, K_I, F_I)$ *updates i's distribution in s as follows,*

$$\mathbf{P}^s_{\sim i} = \delta(i, K_I, F_I) = \mathbf{P}'_{\sim i} \quad (5)$$

where for each possible value v for i, $\mathbf{P}'_{\sim i}(i = v) = Pr(i = v | F_I; K_I)$ and $Pr(i = v | F_I; K_I)$ is the conditional probability inferred by the weighted rules in K_I, with the evidence atoms in F_I.

In each non-terminal turn of a dialog d, the MLN component uses newly collected evidence atoms to update the dialog state.

Reasoning by ASP (Algorithm 3) The ASP component consists of commonsense knowledge and the facts about external factors. The commonsense knowledge is in form of logic program rules. The facts are collected from user feedback in each dialog turn, and are incrementally added into the logic program. When the dialog is terminated, ASP post-processes and overrides agent's service by reasoning with the commonsense rules and collected facts.

The reasoning by ASP is illustrated in Algorithm 3. With external knowledge K_E, in which the facts F_E have been integrated, ASP computes all answer sets of the logic program when dialog terminates. Then the ASP component queries to the database to check whether there is any satisfiable answer set. Let AS_{st} be the set of all satisfiable answer sets,

$$AS_{st} = \{x \mid x \in AS, \exists entity \in db : entity \models x\} \quad (6)$$

If AS_{st} is empty, the agent's service can not be improved anymore. In this case, the ASP component does nothing after reasoning. If AS_{st} is not empty, ASP overrides the agent's origin service with these satisfiable answer sets. It is worth mentioning that, to avoid conflicts with internal knowledge, those internal factors mentioned by FOL rules in K_I do not appear in the heads of ASP rules in K_E.

To sum up, Algorithm 1 learns the dialog policy interactively, uses the internal knowledge learned by Algorithm 2 to update states, and uses the external knowledge reasoned by Algorithm 3 to make better decisions.

3.3 Algorithm Instantiation (Hotel Booking)

In this subsection, we use a *hotel booking* domain to demonstrate how the reasoning and learning system works. Consider the following example.

- Internal factors:

$$I = \{star, price, kind, area, parking\}$$

- External factors: $E = \{weather, user\_loc\}$
- K_I contains first-order logical rules

$$\begin{aligned} &star = 4 \wedge kind = hotel \supset price = expensive : p_1 \\ &star = 5 \wedge kind = hotel \supset price = expensive : p_2 \end{aligned} \tag{7}$$

  where p_1 and p_2 are the pretrained weights of the two rules, respectively.

- K_E contains following rules,

$$\begin{aligned} &suggest\_area(X) \leftarrow weather(bad), user\_loc(X). \\ &suggest\_parking(true) \leftarrow weather(bad). \end{aligned} \tag{8}$$

The agent is supposed to book a hotel according to the user goal and the contextual knowledge. The above internal knowledge (7) indicates that the rooms of 4-star and 5-star hotels are usually expensive. If some facts about internal factors, such as *star* and *room kind*, are collected, they can be used to infer more facts. Formally, a fact about a factor i is a grounding truth of form $i = w$ such that $s \models Pr(i = w) = 1$, where $\models$ is semantic satisfaction. For instance, when a user says that he/she wants to book a 5-star hotel room, the fact of *star*=5 is given to MLN as an evidence atom. Using the atom and the knowledge (K_I), MLN then infers that the room is expensive in probability 0.9. So the distribution of factor *price* in dialog state is updated.

The external knowledge (8) states that, if the current weather is bad, the agent better books a hotel with a parking lot in the the user's area. Suppose that in non-terminal dialog turns, ASP collects the following facts: 1) the weather is bad, 2) the user is in the *north* area. When the dialog terminates, the agent originally decides to book a room for the user in hotel h, which is 4-star hotel in the *center* area without parking lot. Now, by reasoning with K_E and the above facts, the ASP component gets one answer set

$$\begin{aligned} \{&suggest\_area(north), suggest\_parking(true), \\ &weather(bad), user\_loc(north)\} \end{aligned} \tag{9}$$

which concludes that the agent should change the booking area from *center* to *north*, and a parking lot is needed. The service is then updated to booking a room in some 4-star hotel in the *north* area with a parking lot. If such a hotel is available in the database, say h', it then overrides previously suggested hotel h. With the new service of booking a room of h', ASP successfully prevents the user from traveling in distance or walking in bad weather.

The above framework and algorithms learn from historical user data and leverage commonsense knowledge reasoning. Our experiments show that the learned information and knowledge are helpful for dialog policy learning.

4 Experiments

External knowledge is introduced in the experiments to examine how contextual knowledge impacts the dialog system and revises final recommendation. Hence, our success criteria are different from those used in dialog systems that do not reason about external factors. In particular, a dialog is deemed successful, only if all goal constraints are satisfied in the end of the dialog, and the query is fulfilled in such a way that the fulfillment complies with all external factors in the scenario. For instance, when a user is looking for a *3-star guesthouse* room, and the *weather* and the *traffic* are bad, then it will be a successful dialog, only if the agent recommends a *3-star* guesthouse room, where the user does not need to travel a long distance in bad weather or heavy traffic.

4.1 Hypotheses

Experiments have been extensively conducted to evaluate the following three hypotheses:

1. Reasoning with contextual knowledge improves the learning efficiency and the quality of computed action policies within the dialog context.
2. When more data are provided for training MLN, the learning efficiency of our dialog agent is further improved.
3. Our approach is robust enough for policy learning to both incomplete and inaccurate knowledge bases.

Next, we describe our experiment setup, and then present the experimental results collected from a realistic dialog simulation platform.

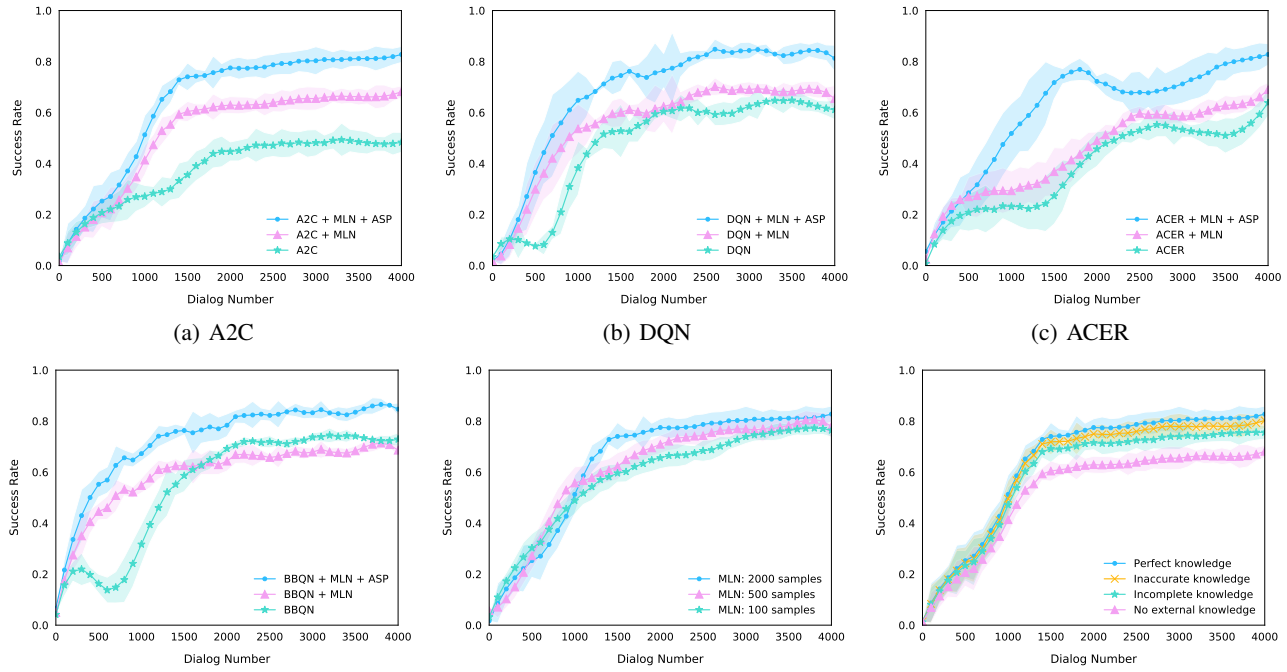
4.2 Experiment Setup

In the experiments, we use a revised version of a *hotel booking* domain in PyDial (Casanueva et al. 2017), where the main slots, i.e., internal factors, are the same with I in Section 3.3. Besides the revised evaluation criteria, we also modified the database to evaluate the reasoning capabilities of our developed approach. We enlarged the original database, so that the user goals would not be frequently rejected due to the lack of diverse data entities. More details about the domain are available in the supplementary appendix.

In the experiment, we used several popular dialog strategy algorithms as baselines, including A2C (Fatemi et al. 2016), DQN, ACER (Weisz et al. 2018) and BBQN (Lipton et al. 2018). The environment parameters are selected via a validation set. In each run, we use 40 batches and each of them contains 100 dialogs. After training with each batch, the policy is evaluated using 100 dialogs. Each data point is an average over multiple runs, where we also evaluate the standard deviation.

For internal knowledge, we utilize Alchemy (Kok et al. 2005) to train a MLN to represent the probabilistic relationships among slots. The trained MLN is then used to update the marginal distributions of the internal factors, once some facts about them are collected from dialogs.

For external knowledge, we use Clingo (Gebser et al. 2014) to ground and solve our ASP logic programs. The



(a) A2C (b) DQN (c) ACER

(d) BBQN (e) Different sample sizes (f) Different knowledge types

Figure 2: Performance comparison. The subgraphs (a)-(d) show the results of our approach deployed on different popular dialog policy algorithms (A2C, DQN, ACER and BBQN). Each subgraph shows our approach (DRL+MLN+ASP), its ablation (DRL+MLN), and the standard DRL-based dialog agent. The subfigure (e) shows policy learning with different sizes of training data for MLN. The subfigure (f) compares different levels of external knowledge (with RL algorithm A2C).

computed answer sets are used to finalize the service when a dialog terminates. We consider three types of external knowledge in the experiments, 1) **Perfect knowledge:** complete and precise knowledge about all external factors, 2) **Incomplete knowledge:** precise knowledge about only a part of the external factors, 3) **Inaccurate knowledge:** knowledge about all external factors, but with errors. For instance, in the "incomplete knowledge" case, the dialog agent does not know the effect of bad weather, so it cannot reason about the external factor of *weather* in fulfilling user queries. More details about the knowledge are available in the supplementary appendix.

4.3 Experimental Results

The first experiment demonstrates how MLN and ASP components improve the dialog agent's learning rate and policy quality, as shown in Figure 2 (a)-(d). Our approach outperforms its ablation (DRL+MLN) in learning efficiency and policy quality. The ASP component is useful for reasoning about the influence of external factors, which potentially improves the service quality. The ablation of our approach performs better than the standard DRL-based dialog agent. The MLN component provides contextual knowledge in the form of conditional distributions given real-time observations that facilitate our agent to predict user preferences. The experimental result supports the first hypothesis that both internal and external knowledge are useful for dialog policy learning.

The second experiment tests on three MLNs, trained with 100, 500 and 2000 data samples, respectively. As shown in Figure 2 (e), the network with larger training data size performs better. When MLN is trained with more data, the dialog agent's learning rate is improved more significantly. Larger training sets enable the agent to learn the underlying distributions and dependencies of the slots in a more precise way, and the learned parameters further better support the agent to assess dialog states and user preferences. The result supports our second hypothesis that larger training data for MLN further improves the dialog policy learning efficiency.

Thirdly, it is worth discussing the robustness to different levels of commonsense knowledge. As shown in Figure 2 (f), the incomplete and inaccurate knowledge bases still improve the dialog agent's performance in comparison to the no-knowledge approach (DRL), although their performances are not as good as perfect knowledge. Incomplete knowledge ignores some of the external factors, and inaccurate knowledge contains some error information, for instance, wrong topological information of a city. Both of them can potentially lead to failures or many turns in dialog. Overall, our method shows the robustness of our dialog policy learning approach given either incomplete or inaccurate knowledge. More training dialogs are needed for learning with imperfect contextual knowledge. Hence, the result supports our third hypothesis that our approach is robust to both incomplete and inaccurate knowledge for policy learning.

4.4 Illustrative Trial

Lastly, we demonstrate a sample dialog between the user and the trained agent by our approach. Figure 3 shows that how

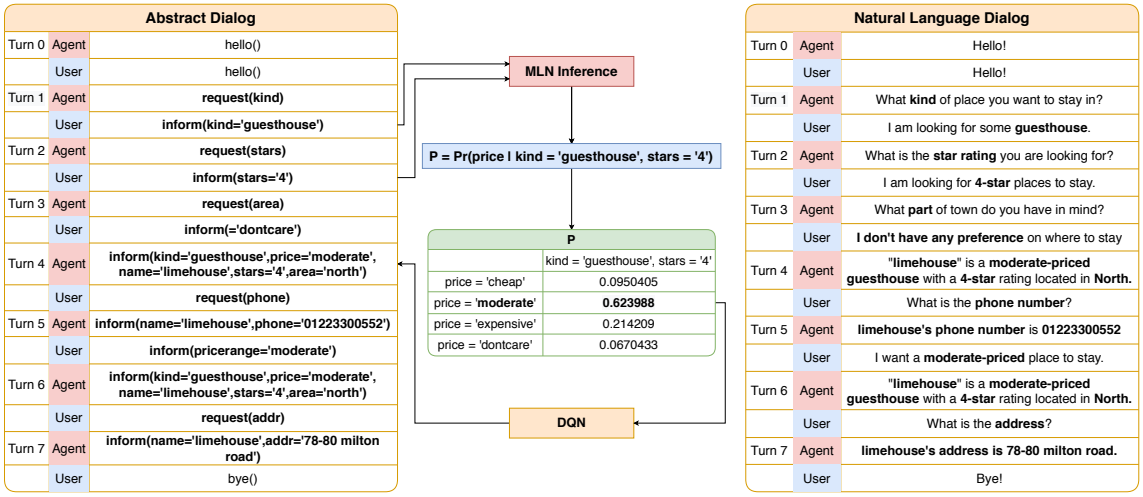


Figure 3: A sample dialog. The left part shows a successful dialog in abstract form, and the corresponding dialog in natural language are in the right part. The middle part illustrates how contextual knowledge is used to help the agent predict user preferences according to current observation.

contextual knowledge is useful for the agent to better predict user preferences with observation. Once some evidence facts about *kind* and *star* are obtained in Turns 1 and 2 respectively, the agent successfully predicts user's preference on *price* in Turn 4 by suggesting a proper hotel that satisfies user's requirements without explicitly querying more information.

The reasoning by the ASP component follows the dialog turns in the figure. The ASP component utilizes collected facts about external factors, such as $user\_loc$, $traffic$ and $weather$, to reason for potentially better suggestion. With **perfect knowledge**, denoted by K_I, ASP component gives the following possible suggestions according to the answer sets of the logic program,

$$update\_area(south).\ \ update\_area(west). \qquad (10)$$

The ASP component simply samples one of them as the new suggestion, i.e., updating the *area* of the hotel to be *south* or *west*. With **incomplete knowledge**, denoted by K_I', the ASP component gives result (10) that is exactly the same as that of K_I, since the missing logic rule is not activated under F. If *weather* in F is changed to *bad*, besides the suggestion to update *parking* to be *true*, K_I will also remove the option $update\_area(south)$ from (10), while K_I' will not remove any option. With **inaccurate knowledge**, denoted by K_I'', the ASP component gives the suggestion $update\_area(west)$, which is one of the options in (10) given by K_I. More details about the external factors and knowledge are in the supplementary appendix.

5 Conclusions and Future Work

In this paper, we integrate two knowledge representation and reasoning (KRR) paradigms into the deep reinforcement learning (RL), as applied to dialog policy learning problems. In particular, we used Answer set programming, a form of non-monotonic KRR, for default reasoning, and Markov logic network, a form of logical-probabilistic KRR, for probabilistic reasoning. Results collected using a realistic dialog simulation platform showed that the KRR paradigms can significantly improve the agents' performance in dialog policy learning. This is the very first work that bridges the gap between KRR and RL within the dialog policy learning context.

The communities of KRR and RL are still largely isolated due to their fundamentally different representations. We show a novel way of integrating the two, as applied to dialog systems. There are a number of ways to further make progress in this line of research. We plan to investigate the possibilities of applying our approach to continuous domains, such as robot control, where the challenge will be the discrete-continuous gap between the state spaces of KRR and RL. Also, the MLN and ASP components of this paper rely on human knowledge (though MLN can learn from data), and we are interested in exploring ways of acquiring such knowledge from human-agent interactions, where dialog can be one possible way of realizing it. While experimenting with real humans are challenging (this is particular true under the pandemic), we would like to apply our approach to learning from real conversations.

Acknowledgments

The work is partially supported by the National Natural Science Foundation of China (61806132, U2001207, 61872248), Guangdong NSF 2017A030312008, Shenzhen STF (ZDSYS20190902092853047, R2020A045), the Project of DEGP (2019KCXTD005, 2021ZDZX1068), the Guangdong "Pearl River Talent Recruitment Program" (2019ZT08X603). A portion of this research has taken place at the Autonomous Intelligent Robotics (AIR) Group, SUNY Binghamton. AIR research is supported in part by grants from the National Science Foundation (NRI-1925044), Ford Motor Company (URP Awards 2019-2021), OPPO (Faculty Research Award 2020), and SUNY Research Foundation. Kaishun Wu is the corresponding author.

References

Baral, C.; Gelfond, M.; and Rushton, J. N. 2009. Probabilistic Reasoning with Answer Sets. *Theory Practice of Logic Programming*, 9(1): 57–144.

Bordes, A.; Boureau, Y.-L.; and Weston, J. 2016. Learning End-to-End Goal-Oriented Dialog. *CoRR*, abs/1605.07683.

Brewka, G.; Eiter, T.; and Truszczyński, M. 2011. Answer Set Programming at A Glance. *Communications of the ACM*, 54(12): 92–103.

Casanueva, I.; Budzianowski, P.; Su, P.; Mrksic, N.; Wen, T.; Ultes, S.; Rojas-Barahona, L. M.; Young, S. J.; and Gasic, M. 2017. A Benchmarking Environment for Reinforcement Learning Based Task Oriented Dialogue Management. *CoRR*, abs/1711.11023.

Davis, E.; and Marcus, G. 2015. Commonsense Reasoning and Commonsense Knowledge in Artificial Intelligence. *Communications of the ACM*, 58(9): 92–103.

Dhingra, B.; Li, L.; Li, X.; et al. 2017. Towards End-to-End Reinforcement Learning of Dialogue Agents for Information Access. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, 484–495.

Fatemi, M.; Asri, L. E.; Schulz, H.; He, J.; and Suleman, K. 2016. Policy Networks with Two-Stage Training for Dialogue Systems. In *Proceedings of SIGDIAL 2016*, 101–110.

Gebser, M.; Kaminski, R.; Kaufmann, B.; and Schaub, T. 2014. Clingo = ASP + Control: Preliminary Report. *CoRR*, abs/1405.3694.

Gelfond, M.; and Lifschitz, V. 1988. The Stable Model Semantics for Logic Programming. In *Logic Programming, Proceedings of the Fifth International Conference and Symposium*, 1070–1080. MIT Press.

Kok, S.; Singla, P.; Richardson, M.; and Domingos, P. 2005. The Alchemy System for Statistical Relational AI. Technical report.

LeCun, Y.; Bengio, Y.; and Hinton, G. 2015. Deep learning. *Nature*, 521(7553): 436–444.

Lipton, Z. C.; Gao, J.; Li, L.; Li, X.; Ahmed, F.; and Deng, L. 2016. Efficient Exploration for Dialogue Policy Learning with BBQ Networks & Replay Buffer Spiking. Technical report.

Lipton, Z. C.; Li, X.; Gao, J.; Li, L.; Ahmed, F.; and Deng, L. 2018. BBQ-Networks: Efficient Exploration in Deep Reinforcement Learning for Task-Oriented Dialogue Systems. In *Proceedings of AAAI 2018*, 5237–5244.

Lu, K.; Zhang, S.; and Chen, X. 2019. Goal-Oriented Dialogue Policy Learning from Failures. In *Proceedings of AAAI 2019*, 2596–2603.

McCarthy, J.; and Hayes, P. J. 1981. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In Webber, B. L.; and Nilsson, N. J., eds., *Readings in Artificial Intelligence*, 431 – 450. Morgan Kaufmann.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level Control through Deep Reinforcement Learning. *Nature*, 518(7540): 529–533.

Richardson, M.; and Domingos, P. 2006. Markov Logic Networks. *Machine learning*, 62(1-2): 107–136.

Schatzmann, J.; Weilhammer, K.; Stuttle, M.; and Young, S. 2006. A Survey of Statistical User Simulation Techniques For Reinforcement-Learning of Dialogue Management Strategies. *The Knowledge Engineering Review*, 21(2): 97–126.

Serban, I. V.; Sordoni, A.; Bengio, Y.; Courville, A.; and Pineau, J. 2016. Building End-to-End Dialogue Systems Using Generative Hierarchical Neural Network Models. In *Proceedings of AAAI 2016*, 3776–3783.

Singh, S.; Litman, D.; Kearns, M.; and Walker, M. 2002. Optimizing dialogue management with reinforcement learning: Experiments with the NJFun system. *Journal of Artificial Intelligence Research*, 16: 105–133.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement Learning: An Introduction*. MIT press.

Ultes, S.; Barahona, L. M. R.; Su, P.-H.; Vandyke, D.; Kim, D.; Casanueva, I.; Budzianowski, P.; Mrkšić, N.; Wen, T.-H.; Gasic, M.; et al. 2017. Pydial: A Multi-Domain Statistical Dialogue System Toolkit. In *Proceedings of ACL 2017, System Demonstrations*, 73–78.

Weisz, G.; Budzianowski, P.; Su, P.; and Gasic, M. 2018. Sample Efficient Deep Reinforcement Learning for Dialogue Systems With Large Action Spaces. *IEEE/ACM Transactions on Audio, Speech and Language Processing*, 26(11): 2083–2097.

Wu, Y.; Li, X.; Liu, J.; et al. 2019. Switch-Based Active Deep Dyna-Q: Efficient Adaptive Planning for Task-Completion Dialogue Policy Learning. In *Proceedings of AAAI 2019*, 7289–7296.

Yang, Z.; Ishay, A.; and Lee, J. 2020. NeurASP: Embracing Neural Networks into Answer Set Programming. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI)*, 1755–1762.

Young, S.; Gašić, M.; Thomson, B.; and Williams, J. D. 2013. Pomdp-based Statistical Spoken Dialog Systems: A Review. *Proceedings of the IEEE*, 101(5): 1160–1179.

Zhang, H.; Gao, Z.; Zhou, Y.; Zhang, H.; Wu, K.; and Lin, F. 2019. Faster and Safer Training by Embedding High-Level Knowledge into Deep Reinforcement Learning. *CoRR*, abs/1910.09986.

Zhang, Y.; Chen, X.; Yang, Y.; Ramamurthy, A.; Li, B.; Qi, Y.; and Song, L. 2020. Efficient Probabilistic Logic Reasoning with Graph Neural Networks. In *8th International Conference on Learning Representations (ICLR)*.

Zhao, T.; and Eskenazi, M. 2016. Towards End-to-End Learning for Dialog State Tracking and Management using Deep Reinforcement Learning. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 1–10.

Zhou, Z.; Beirami, A.; Crook, P. A.; Shah, P.; Subba, R.; and Geramifard, A. 2020. Resource Constrained Dialog Policy Learning Via Differentiable Inductive Logic Programming. In *Proceedings of the 28th International Conference on Computational Linguistics (COLING)*, 6775–6787.