Neuro-Symbolic Approaches for Text-Based Policy Learning

Subhajit Chaudhury, Prithviraj Sen, Masaki Ono, Daiki Kimura,
Michiaki Tatsubori and **Asim Munawar**
IBM Research
subhajit@jp.ibm.com, senp@us.ibm.com,
{moono, daiki, mich}@jp.ibm.com, asim@ibm.com

Abstract

Text-Based Games (TBGs) have emerged as important testbeds for reinforcement learning (RL) in the natural language domain. Previous methods using LSTM-based action policies are uninterpretable and often overfit the training games showing poor performance to unseen test games. We present **S**ymbo**L**ic **A**ction policy for **T**extual **E**nvironments (**SLATE**), that learns interpretable action policy rules from symbolic abstractions of textual observations for improved generalization. We outline a method for end-to-end differentiable symbolic rule learning and show that such symbolic policies outperform previous state-of-the-art methods in text-based RL for the coin collector environment from $5-10$x fewer training games. Additionally, our method provides human-understandable policy rules that can be readily verified for their logical consistency and can be easily debugged. [1]

1 Introduction

Text-based games are increasingly being used as a benchmark for progressing the state of the art in natural language RL. These games typically require solving goals that are defined by natural language descriptions such as "retrieve the coin in the cellar". The agent receives a textual description of the scene and can interact with the environment using only textual commands such as "go north", "take knife" upon which it receives a reward signal for completing the goal (or sub-goal). The action policy model is trained using such reward signals.

Previous methods in text-based RL typically use memory-based recurrent systems for feature extraction from textual observations (Narasimhan et al., 2015; Adolphs and Hofmann, 2020) or knowledge graph extraction (Ammanabrolu and Riedl, 2019; Ammanabrolu and Hausknecht, 2020) for better state representation. However, Chaudhury et al. (2020) showed that previous methods in text-based RL do not generalize well to unseen test games. Furthermore, the learned policy in such cases is not interpretable and is difficult to debug leading to potential unforeseen behavior in real-life applications.

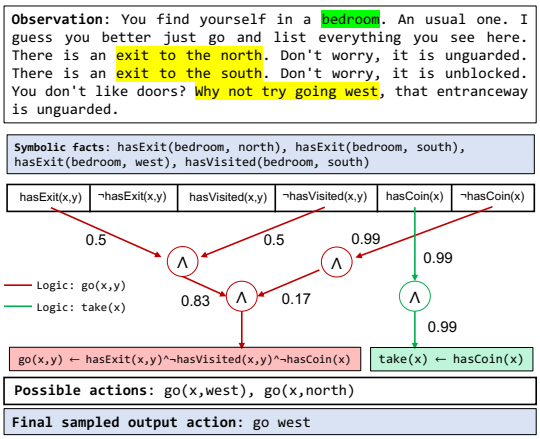

Figure 1: Overview of our neuro-symbolic rule learning. SLATE learns interpretable action policy for each action verb, `go` and `take`, from first-order symbolic states. We show the raw weights for each AND gate ($\wedge$) which are thresholded to obtain the logical rules.

In this paper, we bridge the gap between gradient-based weight learning and symbolic reasoning applied to text-based RL. We introduce **S**ymbo**L**ic **A**ction policy for **T**extual **E**nvironments (**SLATE**), a method for interpretable policy learning in text-based games from symbolic state representation. Our goal is to learn symbolic rules as logical connectives for generating action commands by gradient-based training. We present a symbolic rule learning framework using both MLP with symbolic inputs and Logical Neural Network (LNN) (Riegel et al., 2020), a recent symbolic reasoning-based approach, to learn lifted rules from the first-order symbolic abstraction of textual observations. We show that our symbolic

[1] **Reproducibility**: We will release our code at https://github.com/subhajit1411/slate-text-based-rl

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 3073–3078
November 7–11, 2021.

action policy learning framework outperforms previous text-based RL methods in terms of generalization to unseen games even when the previous methods use symbolic state representation.

2 Related Work

Most previous works on text-based RL handle the problems of partial observability and large action space. LSTM-DQN (Narasimhan et al., 2015) is an early work on text-based RL that used an LSTM-based encoder for feature extraction from textual observations and Q-learning (Watkins and Dayan, 1992) for learning the action policy model. LSTM-DRQN (Yuan et al., 2018) used memory units in the action score to handle the issue of partial observability. CREST (Chaudhury et al., 2020) showed that previous methods overfit the training data and improved generalization by training a bootstrapped model on context-relevant observation text only. Adolphs and Hofmann (2020) presents the winning strategies in the First- TextWorld Competition, which uses recurrent feature extraction along with the A2C (Mnih et al., 2016) RL algorithm for training the policy. Previous works that extract knowledge graphs from observations (Ammanabrolu and Riedl, 2019; Ammanabrolu and Hausknecht, 2020) showed improved performance compared to processing raw textual observations. The recent method of Adhikari et al. (2020) use a dynamic belief graph learning for better generalization than text-based policy learning.

3 SymboLic Action policy for Textual Environments (SLATE)

To improve generalization in TBGs, we propose **SLATE** that learns logical rules based on first-order symbolic inputs from the environment. The symbolic inputs are fed into the logical rule learner to obtain the likelihood of action commands.

3.1 Extracting Symbolic Facts

We explain symbolic fact extraction with Textworld (Côté et al., 2018) Coin Collector environment where the goal is to start from a room and collect the coin from a final room. However, this method generally applies to all text-based environments. The agent uses action commands having a verb and a noun token, such as "go north" and "take coin", etc. We extract symbolic facts from the textual observations of the current step using keyword-based matching. We use lifted representations of three symbolic predicates - $\texttt{hasCoin}(x)$, $\texttt{hasExit}(x, y)$ and $\texttt{hasVisited}(x, y)$, where x represents the grounding for the current room and $y \in \{\text{north}, \text{south}, \text{east}, \text{west}\}$ represents the direction of travel for the agent. $\texttt{hasCoin}(x)$ and $\texttt{hasExit}(x, y)$ symbolic predicates represent if the current room, x has a coin present and if there is any exit available in the direction, y. The predicate $\texttt{hasVisited}(x, y)$ tells if the agent has already visited the direction y at room x which we keep track of using a hash table. A representative observation and corresponding symbolic inputs from the observation are shown in Figure 1.

3.2 Differentiable Logical Rule Learning

Let us consider the original textual observation as O_t and corresponding symbolic abstraction as $S_t(x, y)$, which is the lifted input to the SLATE model. For coin-collector, $S_t(x, y) = [F_t(x, y); \neg F_t(x, y)]$, where $F_t(x, y) = [\texttt{hasCoin}(x), \texttt{hasExit}(x, y), \texttt{hasVisited}(x, y)]$ and $\neg$ refers to the negated forms of the predicate, defined as $\neg p = 1 - p$. The groundings for each predicates (for example, $\texttt{hasExit}$(kitchen,east)) is parametrized by (x, y). The goal of differential rule learning is to down-weight irrelevant and up-weight relevant symbolic predicates towards each action verb generation through gradient based learning.

We wish to find an action policy π that is represented as logical connectives of symbolic inputs. We identify the action verbs "go" and "take" for symbolic rule learning and refer to the corresponding models as $\texttt{go}$-SLATE and $\texttt{take}$-SLATE. For each (x, y) grounding at time step t, we obtain the probabilities of possible actions commands $\{go(x_1, y_1), go(x_1, y_2), \ldots\} = \{f_\theta(S_t(x_1, y_1)), f_\theta(S_t(x_1, y_2)) \ldots\}$ from which the action command is sampled, where $f(.)$ is the forward function of the learning model and θ represents the corresponding learnable parameters of the model. We use two kinds of learning models, which we describe below.

Symbolic MLP: In this setting, we use single feed-forward layers for $\texttt{go}$ and $\texttt{take}$ models, with the symbolic input of $S_t(x, y)$, producing the likelihood of each action command which are passed through a softmax to convert into probabilities.

Logical Neural Networks: We also used the recently proposed LNN model (Riegel et al., 2020)

Table 1: Average success rate (3 random seeds) on 20 unseen test games with a varying number of training games. Our symbolic rule learning method trained on $5-10\times$ fewer data has a similar success rate to state-of-the-art methods on the coin collector environment. Nx denotes that the agent was trained on x number of games.

Methods	Easy				Medium			Hard	
	N5	N10	N25	N50	N10	N25	N50	N25	N50
LSTM-DQN (+attn)	0.0	0.0	0.0	0.03	0.0	0.0	0.0	0.0	0.0
LSTM-DRQN (+attn)	0.0	0.0	0.32	0.47	0.03	0.02	0.02	0.0	0.02
LSTM-DRQN (+attn+dropout)	0.0	0.13	0.58	0.80	0.0	0.0	0.02	0.0	0.0
CREST (glove+att)	0.0	0.18	0.70	0.97	0.0	0.07	0.67	0.17	0.10
CREST (conceptNet+att)	0.0	0.3	0.82	0.93	0.08	0.25	0.67	0.57	0.93
LSTM-DQN (symbolic)	0.0	0.0	0.42	0.33	0.08	0.85	0.80	0.25	0.0
SLATE-MLP (teacher)	0.98	0.95	0.98	0.98	0.63	0.02	0.03	0.0	0.0
SLATE-MLP (rollouts)	1.0	1.0	0.98	1.0	0.97	0.98	0.93	**0.90**	0.67
SLATE-MLP (teacher+rollouts)	1.0	1.0	1.0	1.0	0.93	1.0	1.0	0.57	**1.0**
SLATE-LNN (teacher)	1.0	0.97	1.0	1.0	0.78	0.88	0.93	0.22	0.32
SLATE-LNN (rollouts)	1.0	1.0	1.0	1.0	**1.0**	1.0	0.98	0.47	0.98
SLATE-LNN (teacher+rollouts)	**1.0**	**1.0**	**1.0**	**1.0**	0.97	**1.0**	**1.0**	0.27	0.47

for action probability generation, which we illustrate for a typical 2-input conjunction (AND) node. Let us consider (x_1, x_2) as two logical inputs to the conjunction node (represented as $\wedge$) given as $f(x_1, x_2)$. Unlike conventional logical gates, LNNs define a threshold level for noise tolerance α such that values in $[\alpha, 1]$ signify a logical *high* and values in $[0, 1-\alpha]$ signify a logical *low*. Following the standard truth table of the conjunction (AND) gate, we can get the LNN constraints, that make the LNN behave as a logical conjunction connective with the forward function as the weighted Łukasiewicz t-norm, $f(x_1, x_2; \beta, w_1, w_2) = \max(0, \min(1, \beta - w_1(1-x_1) - w_2(1-x_2)))$. Parameters β, w_1, w_2 are tuned to match target labels during training. We use double description optimization (Frerix et al., 2020) for our constrained weight learning.

3.3 Training

For textworld coin collector games, we use the counting reward introduced in Yuan et al. (2018), where the agent receives a reward of 1.0 for visiting a new room. An additional reward of 1.0 is obtained on successfully retrieving the coin. We perform experiments on three difficulty levels: easy, medium and hard games having 0, 1 and 2 distractor rooms respectively. We outline two major methods for training the symbolic policy corresponding to the mainstream idea of (i) reinforcement learning from rewards obtained from the environment via policy **rollouts**, (ii) imitation learning by bootstrapping from the trajectories of a **teacher** agent (typically text-based LSTM agent) that overfits the training data.

Rollouts: This approach learns purely from environment interactions. Let us assume a policy $\pi_\theta(a_t|s_t)$ with θ being the policy parameters, a_t the action at time t, s_t being the state input, and r_t is the step-wise reward. In this learning method, at each time step t, we sample an action a_t from the policy and label the state-action pair as either negative or positive samples based on whether the reward is positive or not. Since we wish to learn lifted rules, we extract the symbolic state corresponding to the ground entity (noun) in the action and choose the MLP/LNN to train based on the action verb. For instance, consider that the action "go south" results in a positive reward whereas "take coin" causes a zero reward. In the first case, we label the state as a positive sample for the `go`-SLATE model and we only extract the predicate values $\texttt{hasCoin}(x)$, $\texttt{hasExit}(x, \text{south})$ and $\texttt{hasVisited}(x, \text{south})$ and their negated ($\neg$) forms. Similarly, the second case is a negative sample for `coin`-SLATE model.

Therefore, for each SLATE model (either MLP or LNN) corresponding to the action verbs `go` and `take`, we obtain samples with binary labels. The model parameters are then trained using maximum likelihood training with cross-entropy loss. We collect samples from an evolving policy that is trained every 10 episodes for 100 episodes.

Teacher imitation: This corresponds to the imitation learning framework where we collect samples from an overfitted LSTM-based textual teacher model (Chaudhury et al., 2020) on the training games and learn the rules on the SLATE-based policies using behavior cloning (Pomerleau, 1991). At each time step, we sample the lifted symbolic state for the teacher action as positive samples, and

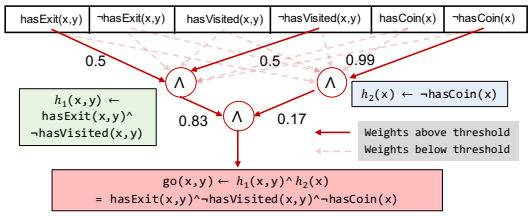

Figure 2: Extracting rules from weights for `go` LNN model using threshold of $\frac{1}{N_{input}}$, with $N_{input} = 6$.

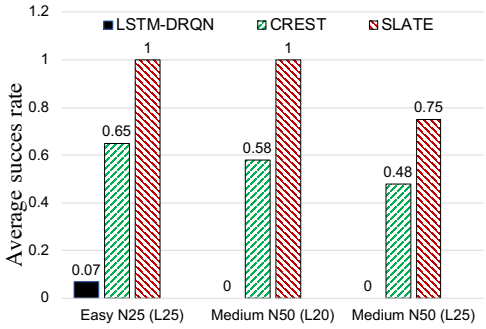

Figure 3: Zero-shot transfer from L15 training games to L20 and L25 testing games for various agent.

the symbolic states corresponding to other possible actions as the negative sample. SLATE parameter learning is performed by cross-entropy loss-based training with logical constraints.

4 Results

For baseline methods we use three previous agents: LSTM-DQN (Narasimhan et al., 2015), LSTM-DRQN (Yuan et al., 2018) and CREST (Chaudhury et al., 2020). The results show that our proposed SLATE successfully generalizes to unseen test games learned from $5-10$x fewer games and performs better zero-shot generalization.

Learned interpretable rules: Figure 2 shows how we extract the learned rules for easy N25 games case. We use a weight threshold of $\frac{1}{N_{input}}$, and all connections greater than the threshold contribute to the learned rule. The rule for `go`-LNN action verb looks at exits that are not visited from the current room given the coin is not present in the room. `take`-LNN takes coin only if the coin is present in the room. These rules match with the logic a human agent would typically use for navigation in such an environment.

Comparison to text-based agents: Table 1 shows that SLATE shows better generalization to unseen test games trained from $5-10x$ fewer training games with close to perfect learning. SLATE learns compact rules in the space of logical connectives that best fit the data and hence it generalizes better to unseen data as well (simple rules generalize better - Occam's Razor).

We also compare SLATE with LSTM-DQN trained/tested on symbolic inputs. LSTM-DQN agent is given the symbolic input as textual facts like "has exit east has exit south has visited south". Since SLATE does not have a recurrent action policy and makes single-step decisions, for a fair comparison we do not compare it with LSTM-DRQN that has a recurrent action scorer unit. The results in Table 1 show that our proposed method generalizes better suggesting that LSTM-based agents might require more training data.

Table 1 show that SLATE with both MLP and LNN achieves generalization close to perfect for the RL (**rollouts**) setting. However, SLATE with LNN gives an interpretable representation in the form of logical rules due to its logical constraints that MLP does not have. For the teacher imitation setting, LNN exhibits better success rates than MLP especially for medium and hard games. Since such games have multiple possible exits and only one direction is chosen by the teacher's action, the other possible directions are labeled as negative samples. We believe MLP based rule learning is not robust to handle such conflicting (or noisy) information that can be handled by LNN.

Zero-shot transfer: Similar to the experiments in Chaudhury et al. (2020), Figure 3 shows the zero-shot performance of SLATE compared to LSTM-DRQN and CREST when trained on games having 15 rooms (L15) and tested on 20 (L20) and 25 (L25) rooms. SLATE (with LNN) shows better generalization compared to previous text-based agents due to compact and logically consistent rule learning. This is because SLATE learns lifted rules that inherently provide the structure to generalize to unseen configurations that pure deep learning approaches struggle to achieve.

Results on Cooking World: We also show results on cooking games (Adolphs and Hofmann, 2020), where the agent has to navigate to the kitchen, gather items that a present in the recipe, and prepare and eat the meal. For this game, we use 6 predicates: `inRoom`(x) (if item x is in the current room), `isIngredient`(x) (if item x is a required ingredient), `inInventory`(x) (if item x is currently in the inventory), `closed`(x)

(if item x is in closed state), $\texttt{hasExit}(x)$ and $\texttt{hasVisited}(x)$.

The actions verbs to learn rules for are: $\texttt{take}(x)$, $\texttt{open}(x)$, and $\texttt{go}(x)$. We use the game walkthrough to learn LNN rules in the teacher imitation mode. The learned rules for three verbs are: $\texttt{open}(x) \leftarrow \texttt{inRoom}(x) \wedge \texttt{closed}(x)$, $\texttt{take}(x) \leftarrow \texttt{inRoom}(x) \wedge \texttt{isIngredient(x)}, \texttt{go}(x) \leftarrow \neg\texttt{inRoom}(x) \wedge \texttt{hasExit}(x)$. We assume that learning rules with first-order quantifiers with forall ($\forall$) and exists ($\exists$) are out-of-scope of this paper. Therefore, the rule for "prepare meal" is specified as $\forall_z(\texttt{isIngredient}(z) \wedge \texttt{inInventory}(z))$ and for "eat meal" is $\texttt{inInventory}(x = \text{meal})$.

We present the normalized score on test games for SLATE for difficulty level 3 with training 20 at 66.67% which outperforms the scores of 41.7% for GATA, 41.7% for GATA-GTP, and 46.7% for GATA-GTF from (Adhikari et al., 2020), showing SLATE's efficiency for complicated cases. However, there is still room for improvement for SLATE on the cooking games because the rule for $\texttt{go}(x) \leftarrow \neg\texttt{inRoom}(x) \wedge \texttt{hasExit}(x)$ is inconsistent and the term $\neg\texttt{inRoom}(x)$ is not required on the right-hand side for the correct rule.

5 Conclusions

We present **SLATE**, a neuro-symbolic approach for action policy learning in text-based games using a differentiable rule learner from first-order symbolic inputs. Our method outperforms previous text-based state-of-the-art methods on textworld coin collector games from $5-10$x fewer training games and shows zero-shot generalization to unseen test configurations since it learns compact, interpretable, and logically correct rules. In this paper, we presented neuro-symbolic rule learning for the case, where the list of possible predicates require knowledge of the domain and are known before the training. For a more generalized rule learning on an apriori unknown set of predicates, we plan to use information extraction techniques like Abstract Meaning Representation (AMR) and OpenIE for obtaining a domain-agnostic graphical state representation for rule induction as a future extension of this work.

Ethical Statement

Our paper describes a method for action policy learning in text-based games that is unlikely to produce ethically questionable action commands since the vocabulary of possible actions is limited and does not contain ethically problematic tokens. On the contrary, our model is fully interpretable, thus leading to a transparent analysis of the model's action outputs. Such a neuro-symbolic approach is ideal for analyzing the reason behind ethically questionable outputs like racial bias or hate speech learned by deep models from large amounts of public data. Therefore, our approach and related extensions of neuro-symbolic approaches are possible methods for reducing ethically problematic outputs from traditional deep models. Therefore, the ethical risk of our proposed neuro-symbolic SLATE approach is low and can be a likely means to address bias in deep neural network-based learning.

References

Ashutosh Adhikari, Xingdi Yuan, Marc-Alexandre Côté, Mikuláš Zelinka, Marc-Antoine Rondeau, Romain Laroche, Pascal Poupart, Jian Tang, Adam Trischler, and Will Hamilton. 2020. Learning dynamic belief graphs to generalize on text-based games. In *Advances in Neural Information Processing Systems*, volume 33, pages 3045–3057. Curran Associates, Inc.

Leonard Adolphs and Thomas Hofmann. 2020. Ledeepchef deep reinforcement learning agent for families of text-based games. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7342–7349.

Prithviraj Ammanabrolu and Matthew Hausknecht. 2020. Graph constrained reinforcement learning for natural language action spaces. In *International Conference on Learning Representations*.

Prithviraj Ammanabrolu and Mark Riedl. 2019. Playing text-adventure games with graph-based deep reinforcement learning. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3557–3565, Minneapolis, Minnesota. Association for Computational Linguistics.

Subhajit Chaudhury, Daiki Kimura, Kartik Talamadupula, Michiaki Tatsubori, Asim Munawar, and Ryuki Tachibana. 2020. Bootstrapped Q-learning with context relevant observation pruning to generalize in text-based games. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3002–3008.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud

Adada, et al. 2018. Textworld: A learning environment for text-based games. *arXiv preprint arXiv:1806.11532*.

Thomas Frerix, Matthias Nießner, and Daniel Cremers. 2020. Homogeneous linear inequality constraints for neural network activations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 748–749.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR.

Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. 2015. Language understanding for text-based games using deep reinforcement learning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1–11.

Dean A Pomerleau. 1991. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 3(1):88–97.

Ryan Riegel, Alexander Gray, Francois Luus, Naweed Khan, Ndivhuwo Makondo, Ismail Yunus Akhalwaya, Haifeng Qian, Ronald Fagin, Francisco Barahona, Udit Sharma, et al. 2020. Logical neural networks. *arXiv preprint arXiv:2006.13155*.

Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine learning*, 8(3-4):279–292.

Xingdi Yuan, Marc-Alexandre Côté, Alessandro Sordoni, Romain Laroche, Remi Tachet des Combes, Matthew Hausknecht, and Adam Trischler. 2018. Counting to explore and generalize in text-based games. *arXiv preprint arXiv:1806.11525*.