Learning Neuro-Symbolic World Models for Text-Based Game Playing Agents

Don Joven Agravante and **Michiaki Tatsubori**
IBM Research
don.joven.r.agravante@ibm.com, mich@jp.ibm.com

Abstract

Text-based games serve as important benchmarks for agents with natural language capabilities. To enable such agents, we are interested in the problem of learning useful world models. Our assumption is that such a world model is best represented by a logical form which underlies the structure of these games. We propose to tackle this problem by leveraging the expressivity of recent neuro-symbolic architectures, specifically the Logical Neural Networks (LNN). Here, we describe a method that can learn neuro-symbolic world models on the TextWorld-Commonsense set of games. We then show that planning on this learned world model results in optimal actions in the game world.

1 Introduction

Text-Based Games began as a form of entertainment in the 1980s. Players could read a narrative and imagine the state of the game world from text. They can then interact with this world through text input as well. In recent years, text-based games have become an interesting benchmark in the intersection of natural language processing and sequential decision making. For example, deep reinforcement learning (RL) was proposed as a possible solution starting with (Narasimhan et al., 2015). Recently, several sets of benchmarks and game environments were proposed such as TextWorld (Côté et al., 2018), Jericho (Hausknecht et al., 2020) and TextWorld Commonsense (Murugesan et al., 2021).

Text-based games present several different problems that are interesting topics for research. In this paper, we concentrate on the problem of learning logical world models. The main idea is that the colorful and complex natural language narrative of the text actually describes a fairly simple and compact world. This idea is used implicitly in other works appearing as Knowledge Graphs (Ammanabrolu and Riedl, 2019) or belief graphs (Adhikari et al., 2020) that are then used to enhance deep RL methods. In contrast to these methods we want to explicitly use the logical world models to plan optimal action sequences to be performed in the game. The main question to be addressed is then: How can we learn such models for text-based games?

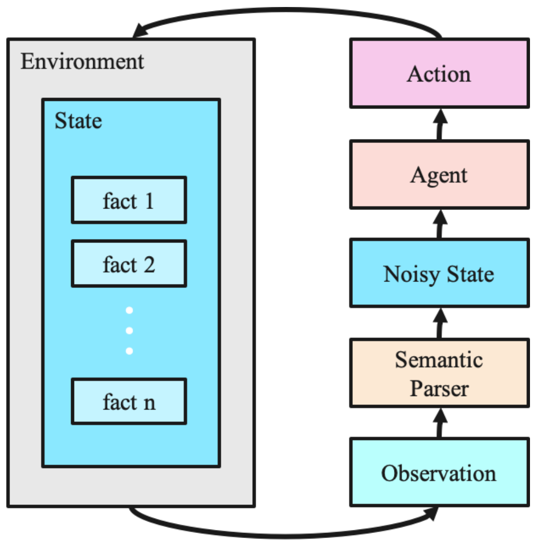


Figure 1: Model-based architecture with logical states

An overview of our proposed method is depicted by Figure 1. The left side depicts that the environment state can be sufficiently approximated as a set of logical facts. Continuing in the bottom right, the agent can get textual observations of the environment. We assume that we have a *semantic parser* that converts these observations into a logical form. In the real situation the semantic parsing is good, but won't be perfect, hence we require that our agent should be capable of handling noisy logical states. From such states, our agent should produce suitable actions for accomplishing its tasks in the environment. In the next sections we formally describe the problem setting of our agent and our proposed method that learns explicit logical world models from potentially noisy logical data.

2 Problem Definition

Text-based games are often modelled with the RL problem setting in mind as Partially Observable - Markov Decision Processes (PO-MDP) (Côté et al., 2018; Hausknecht et al., 2020). As a first approach, we add an additional assumption in this paper - that the semantic parser can remove partial observability and that we are dealing with an MDP. At each time step the agent uses the information in a state, s, to take an action, a, which transitions the state to the new state, s' according to the state transition function T such that $s' = T(s, a)$. While acting in this environment the agent also gets rewards, r, according to an unknown reward function, R, such that $r = R(s, a)$. In the model-free RL setting, the agent learns a policy or value function which directly governs the actions. Here, we are interested in the model-based RL setting where the agent learns a model of the world which usually consists of both T and R. This model can then be used with planning and search methods to find the optimal actions.

Based on the classical model-based RL setting, our problem has two more important specifications. First, we assume that our environment is *relational*, similar to (Lang et al., 2012). This means that all actions and states are composed of relational logic. They may be in the propositional form but there must be a corresponding lifted form that has a consistent *meaning*. For example, the propositional state, *on(book,table)* can be abstracted or *lifted* into *on(x,y)* with predicate, *on*, and the variables, (x, y). The first assumption is that all states and actions handled by the agent are in this relational lifted form. This assumption can be handled as a design specification of the semantic parser. The second assumption is that the goal state is given. This is a weaker assumption that is already used in current RL research, the so-called *goal-conditioned RL*. Here, it allows us to concentrate only on learning T since R is no longer required for planning when we are given the goal state.

In the end, our problem definition is very close to the line of work on learning symbolic models of complex domains (Pasula et al., 2007). It may seem surprising to revert to an older problem setting, however, we believe that recent advances in semantic parsing and logical rule learning might provide breakthroughs.

3 Learning Logical World Models

The problem of learning logical rules that explain a given set of logical examples can be cast into the general problem called Inductive Logic Programming (ILP) (Muggleton and De Raedt, 1994). What needs to be done is then to cast our relational model-based RL problem into ILP form. But before going into that detail, it is important to note that relying on classical ILP has significant failings. In particular, it is not well suited to noisy data to the extent that a single erroneous data point may cause the whole system to fail. However, newer methods that leverage neural networks have shown great promise on working even with noisy data (Evans and Grefenstette, 2018). These are sometimes called neural ILP, differentiable ILP or neurosymbolic ILP. These advances are the main impetus for us to revisit the classical problem of learning logical world models.

We may use any such ILP method that is noise-resistant but here we propose to use the Logical Neural Network (LNN) (Riegel et al., 2020) as a Neuro-Symbolic AI framework because it has two main features. It is an end-to-end differentiable system that enables scalable gradient-based learning and it has a real-valued logic representation of each neuron that enables logical reasoning (Riegel et al., 2020). (Riegel et al., 2020; Sen et al., 2021). These features may prove useful in future works.

Now, getting back to the task of expressing our relational model-based RL problem as ILP, we first gather data samples which are triples of lifted logic, (s, a, s'). This is gathered by using an exploration policy to generate actions. Here, we used a policy that uniformly randomly samples the action space but better exploration methods may be used, such as that outlined in (Lang et al., 2012). This data collection may be done in an offline or online RL setting but we assume that a large enough *batch* is available in the online RL setting before we start the learning procedure.

Given a batch of data samples, the learning procedure must produce an estimate of T. This T will be the *hypothesis* to be generated by our ILP. To make learning more efficient we need to narrow down the definition of T. Because we are ultimately interested in using T for planning, we define it as a set of STRIPS-like operators where each one is a quadruple of $(\alpha, \beta, \gamma, \sigma)$. Each element is a set of logical conditions where α are conditions that must be true for the action to be executable, β are ones that must be false, γ are ones made true by the action and σ are ones made false. These conditions are the lifted logic statements that comprise a state, s, and the set of all possible conditions is P. We model each of the operator elements as an LNN conjunction operator whose inputs are P. The LNN learning procedure can learn weights for each of these inputs that correspond to real-valued logic (Riegel et al., 2020; Sen et al., 2021). For the LNNs of α and β, the inputs are given the corresponding logical values of the conditions in s. The output is true when action, a, corresponds and $s \neq s'$ otherwise it is false. For the LNNs of γ and σ, the inputs are given the logical values corresponding to the difference in the conditions of s and s' such that γ are the the conditions made true and σ those that are made false. The output is true when action, a, corresponds otherwise it is false. Using these inputs and outputs to the LNN, gradient-based optimization can be used for supervised learning (Riegel et al., 2020; Sen et al., 2021). When learning converges, we have a set of weights for each of the corresponding elements. These may be interpreted as probabilistic transitions but here we simply threshold them and maintain a deterministic transition system for our final estimate of T. Given this operator transition model and the goal, we can be in any state and use classical planning methods to find a series of actions to reach the goal.

4 Results and Discussions

In this paper, we experiment on the TextWorld Commonsense (TWC) set of games (Murugesan et al., 2021). In general, TextWorld provides an ideal testbed for us because it gives an interface to the underlying logical states (Côté et al., 2018). In our problem setting this conveniently corresponds to a perfect semantic parsing. Although our methods are chosen with a view to being able to handle real world noise, our preliminary results here show the soundness of our methodology. Once we have the logical world model in the form of STRIPS operators we can use this with a planner to complete our game-playing agent. Here, we use the Fast-Downward Planner (Helmert, 2006). For our convenience we also convert the STRIPS operators into PDDL operators in the form of preconditions and effects by combining (α, β) into the preconditions and (γ, σ) into the effects.

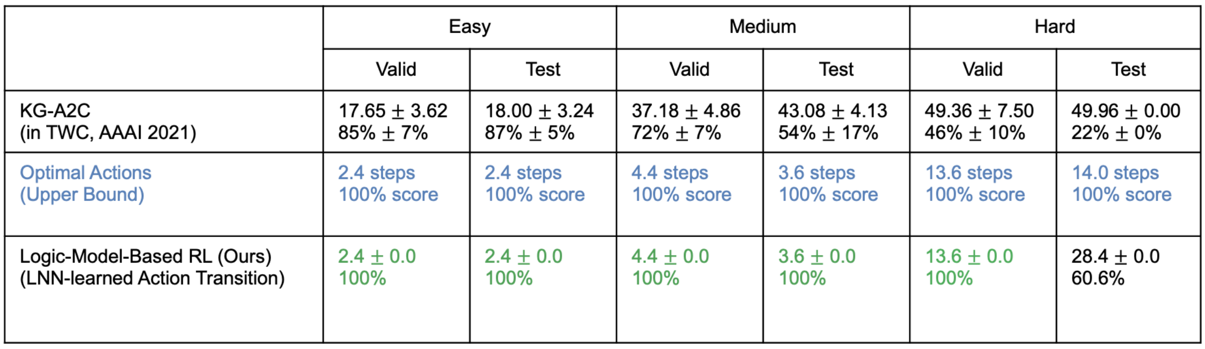
For our results, we first show some examples of the learned rules in our logical world model in Figure 3. This is in the converted PDDL form. Here, we can visually inspect the validity of the rules. For example, for the *take* action the effect would be that the object *v*0 is no longer *at*(*v*1) but now it is in the inventory *v*3. This level of *explanability* is inherent in logical models although it requires careful inspection.

It would be tedious to inspect the rules individually, but what would be more interesting is if taken altogether can these rules allow us to plan optimal actions in the world. To answer this, we present our results in the table shown in Figure 2. Here, there are 3 rows, the first two rows are for comparison while the third row is our method. For comparison, we show the best performing deep RL agent in (Murugesan et al., 2021) and the optimal possible actions using perfect game knowledge. Note that we have additional assumptions differing from the plain deep RL setting of the original setup in (Murugesan et al., 2021) but we give this as a reference on the potential improvement our overall approach might provide. The TWC games are categorized into Easy-Medium-Hard with a validation and testing set for each as shown in the columns. Our results show that planning on our learned model can produce the same optimal actions for all the Easy and Medium games and for the validation set of the Hard games. An interesting limitation appears in the test set of the Hard games wherein novel predicates appear in the test set that do not appear in any of the training or validation set. This is a current limitation of our system which does not have any mechanism for handling such novel predicates.

5 Conclusion

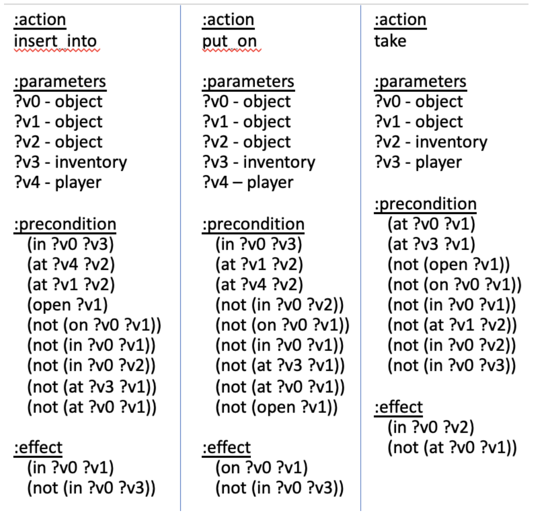
We outlined and proposed a model-based RL setting for text-based games which is comprised of a semantic parser which produces logical states, a neuro-symbolic ILP module for learning logical world models and an off-the-shelf planning system to produce optimal actions in the game world. We believe this approach shows promise and we present initial results and experiments on the key component which learns the logical world models.

The natural progression of this work is to use real semantic parsers. Although our method is designed with noisy logical states in mind, the extent and type of noise might differ in practice. We are also working on relaxing more assumptions in our problem setting to be closer to those assumptions



	Easy		Medium		Hard	
	Valid	Test	Valid	Test	Valid	Test
KG-A2C (in TWC, AAAI 2021)	17.65 ± 3.62 85% ± 7%	18.00 ± 3.24 87% ± 5%	37.18 ± 4.86 72% ± 7%	43.08 ± 4.13 54% ± 17%	49.36 ± 7.50 46% ± 10%	49.96 ± 0.00 22% ± 0%
Optimal Actions (Upper Bound)	2.4 steps 100% score	2.4 steps 100% score	4.4 steps 100% score	3.6 steps 100% score	13.6 steps 100% score	14.0 steps 100% score
Logic-Model-Based RL (Ours) (LNN-learned Action Transition)	2.4 ± 0.0 100%	2.4 ± 0.0 100%	4.4 ± 0.0 100%	3.6 ± 0.0 100%	13.6 ± 0.0 100%	28.4 ± 0.0 60.6%

Figure 2: Scores on the TextWorld Commonsense(TWC) set of games



```
:action
insert_into

:parameters
?v0 - object
?v1 - object
?v2 - object
?v3 - inventory
?v4 - player

:precondition
  (in ?v0 ?v3)
  (at ?v4 ?v2)
  (at ?v1 ?v2)
  (open ?v1)
  (not (on ?v0 ?v1))
  (not (in ?v0 ?v1))
  (not (in ?v0 ?v2))
  (not (at ?v3 ?v1))
  (not (at ?v0 ?v1))

:effect
  (in ?v0 ?v1)
  (not (in ?v0 ?v3))
```

```
:action
put_on

:parameters
?v0 - object
?v1 - object
?v2 - object
?v3 - inventory
?v4 – player

:precondition
  (in ?v0 ?v3)
  (at ?v1 ?v2)
  (at ?v4 ?v2)
  (not (in ?v0 ?v2))
  (not (on ?v0 ?v1))
  (not (in ?v0 ?v1))
  (not (at ?v3 ?v1))
  (not (at ?v0 ?v1))
  (not (open ?v1))

:effect
  (on ?v0 ?v1)
  (not (in ?v0 ?v3))
```

```
:action
take

:parameters
?v0 - object
?v1 - object
?v2 - inventory
?v3 - player

:precondition
  (at ?v0 ?v1)
  (at ?v3 ?v1)
  (not (open ?v1))
  (not (on ?v0 ?v1))
  (not (in ?v0 ?v1))
  (not (at ?v1 ?v2))
  (not (in ?v0 ?v2))
  (not (in ?v0 ?v3))

:effect
  (in ?v0 ?v2)
  (not (at ?v0 ?v1))
```

Figure 3: Examples of the learned action models

used. Another recent trend that will help is the emmergence of *foundation models* (Bommasani et al., 2021) which are large general-purpose pre-trained models. In our context this would be repuposed for translating the natural data into logical states. We also believe our method can be made general enough for even the most difficult text-based games.

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.

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.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri S. Chatterji, Annie S. Chen, Kathleen Creel, Jared Quincy Davis, Dorottya Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah D. Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark S. Krass, Ranjay Krishna, Rohith Kuditipudi, and et al. 2021. On the opportunities and risks of foundation models. *CoRR*, abs/2108.07258.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Ruo Yu Tao, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. 2018. Textworld: A learning environment for text-based games. *CoRR*, abs/1806.11532.

Richard Evans and Edward Grefenstette. 2018. Learning explanatory rules from noisy data. *Journal of Artificial Intelligence Research*, 61:1–64.

Matthew Hausknecht, Prithviraj Ammanabrolu, Marc-Alexandre Côté, and Xingdi Yuan. 2020. Interactive fiction games: A colossal adventure. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7903–7910.

Malte Helmert. 2006. The fast downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246.

Tobias Lang, Marc Toussaint, and Kristian Kersting. 2012. Exploration in relational domains for model-based reinforcement learning. *Journal of Machine Learning Research*, 13(119):3725–3768.

Stephen Muggleton and Luc De Raedt. 1994. Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, 19:629–679.

Keerthiram Murugesan, Mattia Atzeni, Pavan Kapanipathi, Pushkar Shukla, Sadhana Kumaravel, Gerald Tesauro, Kartik Talamadupula, Mrinmaya Sachan, and Murray Campbell. 2021. Text-based rl agents with commonsense knowledge: New challenges, environments and baselines. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(10):9018–9027.

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, Lisbon, Portugal. Association for Computational Linguistics.

Hanna M Pasula, Luke S Zettlemoyer, and Leslie Pack Kaelbling. 2007. Learning symbolic models of stochastic domains. *Journal of Artificial Intelligence Research*, 29:309–352.

Ryan Riegel, Alexander G. Gray, Francois P. S. Luus, Naweed Khan, Ndivhuwo Makondo, Ismail Yunus Akhalwaya, Haifeng Qian, Ronald Fagin, Francisco Barahona, Udit Sharma, Shajith Ikbal, Hima Karanam, Sumit Neelam, Ankita Likhyani, and Santosh K. Srivastava. 2020. Logical neural networks. *CoRR*, abs/2006.13155.

Prithviraj Sen, Breno W. S. R. de Carvalho, Ryan Riegel, and Alexander G. Gray. 2021. Neuro-symbolic inductive logic programming with logical neural networks. *CoRR*, abs/2112.03324.