# Behavior Cloned Transformers are Neurosymbolic Reasoners

**Ruoyao Wang**[‡], **Peter Jansen**[‡], **Marc-Alexandre Côté**[♣], **Prithviraj Ammanabrolu**[♢]
[‡]University of Arizona, Tucson, AZ [♣]Microsoft Research Montréal
[♢]Allen Institute for AI, Seattle, WA
{ruoyaowang,pajansen}@arizona.edu
macote@microsoft.com, raja@allenai.org

## Abstract

In this work, we explore techniques for augmenting interactive AI agents with information from symbolic modules, much like humans use tools like calculators and GPS systems to assist with arithmetic and navigation. We test our agent's abilities in text games—challenging benchmarks for evaluating the multi-step reasoning abilities of game agents in grounded, language-based environments. Our experimental study indicates that injecting the actions from these symbolic modules into the action space of a behavior cloned transformer agent increases performance on four text game benchmarks that test arithmetic, navigation, sorting, and common sense reasoning by an average of 22%, allowing an agent to reach the highest possible performance on unseen games. This action injection technique is easily extended to new agents, environments, and symbolic modules.[1]

## 1 Introduction

Interactive fiction games (or *text games)* evaluate AI agents abilities to perform complex multi-step reasoning tasks in interactive environments that are rendered exclusively using textual descriptions. Agents typically find these games challenging due to the complexities of the tasks combined with the reasoning limitations of contemporary models. Overall performance is generally low, with agents currently solving only 30% of classic interactive fiction games such as Zork (Ammanabrolu and Hausknecht, 2020; Yao et al., 2021; Atzeni et al., 2022). Similarly, reframing benchmarks such as question answering into text games where agents must interactively reason with their environment and make their reasoning steps explicit causes performance to substantially decrease (Wang et al., 2022), highlighting both the capacity

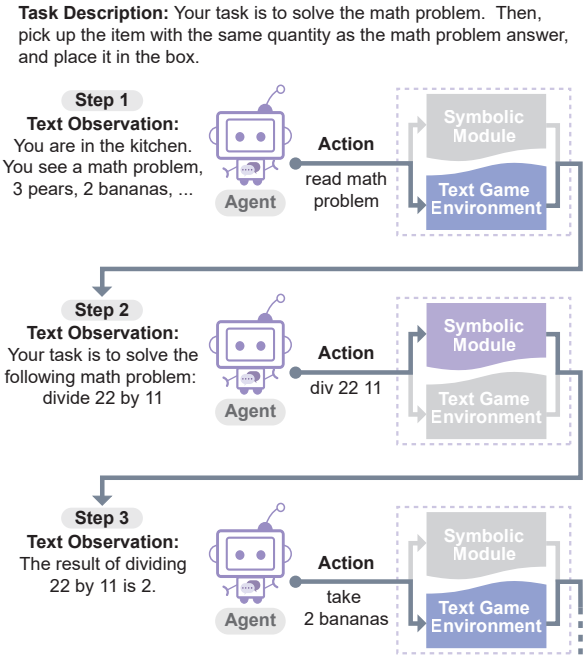


Figure 1: An overview of our approach on an example game evaluating arithmetic ability. At each step, the agent receives an observation from the environment, then takes an action. By providing actions that interface to symbolic modules (such as a calculator), the agent is able to use external knowledge to help solve the task.

of this methodology to evaluate multi-step reasoning, and the limitations of current language models.

While large language models are capable of a variety of common sense reasoning abilities (Liu et al., 2022b; Ji et al., 2020), contemporary agents typically struggle on tasks such as navigation, arithmetic, knowledge base lookup, and other tasks that humans typically make use of external tools (such as GPS systems, calculators, and books) to solve. This is at times frustrating, because the tasks they perform poorly on can sometimes be solved in a few dozen lines of code. In this work, we show that combining both approaches is possible for text game agents, with our approach shown in Figure 1. We develop symbolic modules for arithmetic, navigation, sorting, and knowledge base lookup in

[1]We release our system as open source, available at http://github.com/cognitiveailab/neurosymbolic/



*Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2777–2788

PYTHON, paired with new benchmark games for testing these capacities in interactive text game environments. We empirically demonstrate that injecting actions from those modules into the action space of each game can allow transformer-based agents to make use of that information, and achieve near-ceiling performance on unseen benchmark games that they otherwise find challenging.

## 2 Related Work

Neurosymbolic reasoning offers the promise of combining the inference capabilities of symbolic programs with the robustness of large neural networks. In the context of text games, Kimura et al. (2021a) develop methods to decompose text games into a set of logical rules, then combine these rules with deep reinforcement learning (Kimura et al., 2021b) or integer linear programming (Basu et al., 2021) to substantially increase agent performance while providing a more interpretable framework for understanding why agents choose specific actions (Chaudhury et al., 2021). More generally, neurosymbolic reasoning has been applied to a variety of multi-step inference problems, such as multi-hop question answering (Weber et al., 2019), language grounding (Zellers et al., 2021), and semantic analysis (Cambria et al., 2022).

Because text games require interactive multi-step reasoning, agents have most commonly been modelled using reinforcement learning (e.g. He et al., 2016; Zahavy et al., 2018; Yao et al., 2020), though overall performance on most environments remains low (see Jansen, 2022; Osborne et al., 2021, for reviews). Recently, alternative approaches modeling reinforcement learning as a sequence-to-sequence problem using imitation learning have emerged, centrally using behavior cloning (Torabi et al., 2018), decision transformers (Chen et al., 2021), and trajectory transformers (Janner et al., 2021). These approaches model interactive multi-step reasoning problems as a Markov decision process, where an agent's observation and action history up to some depth are provided as input, and the transformer must predict the next action for the agent to take. Behavior cloning and decision transformers have recently been applied to text games with limited success (Wang et al., 2022). Here, we show that the performance of a behavior cloned transformer can substantially increase when augmented with neurosymbolic reasoning.

## 3 Approach

Figure 1 illustrates the workflow of our approach. At each time step $t$, based on the observation $o_t$, the symbolic module will generate a set of valid actions $A_t^m$, and the text game environment will have a distinct set of valid actions $A_t^e$. Let the valid action set at step $t$ be $A_t = A_t^m \cup A_t^e$. Given $o_t$ and $A_t$, the agent needs to choose an action $a_t \in A_t$ to take. Note that in principle, any agent could be adapted to use this approach, since we simply inject actions from the symbolic modules into the environment action space. At a given time step, our approach checks if $a_t$ is a valid symbolic action. If $a_t \in A_t^m$ (e.g. *div 22 11* in Figure 1), the symbolic module will generate the next observation $o_{t+1}$, otherwise the text game environment will take $a_t$ and generate $o_{t+1}$ (e.g. *take 2 bananas* in Figure 1).

## 4 Environments and Symbolic Modules

We evaluate our approach to neurosymbolic reasoning using four text game benchmark environments centered around pick-and-place tasks, including one existing benchmark and three new developed for this work. Each environment supports parametric variation to generate many different games. These environments are outlined below, with additional details and example playthroughs found in APPENDIX B. All environments were implemented using the TEXTWORLDEXPRESS game engine (Jansen and Côté, 2022).

**Text World Common Sense (TWC):** A benchmark common sense reasoning task (Murugesan et al., 2021) where agents must collect objects from the environment (e.g. *dirty socks*), and place those objects in their canonical common sense locations (e.g. *washing machine*). The symbolic module for this game allows agents to query a knowledge base of *(subject, relation, object)* triples (e.g. *(cushion, hasCanonicalLocation, sofa)*).

**MapReader:** A navigation-themed pick-and-place game similar to Coin Collector (Yuan et al., 2018). An agent starts in a random location (e.g. *the kitchen*), and is provided with a target location (e.g. *the garage*). The agent must navigate to the target location, pick up a coin, then return to the starting location and place it in a box. The agent is further provided with a map that can be used for efficient route planning. The navigation symbolic module paired with this environment scrapes the observation space for location information (e.g. *you are*

| **Knowledge Base Module** |
|---|
| > *query cushion*<br>cushion located sofa<br>cushion located armchair |
| **Navigation Module** |
| You are currently in the kitchen.<br>> *next step to living room*<br>The next location to move to is: hallway. |
| **Arithmetic Module** |
| > *mul 3 6*<br>Multiplying 3 and 6 results in 18. |
| **Sorting Module** |
| > *sort ascending*<br>The objects in ascending order are: 8mg of steel, 2g of iron, 5kg of copper. |

Table 1: Example actions (inputs) and responses from the four symbolic modules investigated in this work.

*currently in the kitchen*), and both complete (e.g. *the map*) or partial (e.g. *to the north you see the hallway*) spatial connection information.

**Arithmetic:** A math-themed task, where agents must read and solve a math problem in order to know which object from a set of objects to pick-and-place. An example problem is *"take the bundle of objects that is equal to 3 multiplied by 6, and place them in the answer box"*, where the agent must complete the task by choosing *18 apples*. Distractor objects are populated with quantities that correspond to performing the arithmetic incorrectly (e.g. *3 oranges*, corresponding to subtracting 3 from 6). We pair the arithmetic game with a calculator module capable of performing addition, subtraction, multiplication, and division.

**Sorting:** A sorting-themed game where the agent begins in a room with three to five objects, and is asked to place them in a box one at a time in order of increasing quantity. To add complexity, quantities optionally include units (e.g. *5kg of copper, 8mg of steel*) across measures of volume, mass, or length. The sorting game is paired with a module that scrapes the observation space for mentions of objects that include quantities, and sorts these in ascending or descending order on command.

### 4.1 Symbolic Modules

Examples of symbolic modules and their responses are provided in Table 1. The number of valid actions injected by each module varies between 2 from the sorting module (*ascending/descending*) to over 500 from the knowledge base look-up (one for each object and its canonical locations present in the knowledge base). Symbolic modules were implemented in PYTHON as a wrapper around the TEXTWORLDEXPRESS API, allowing modules to monitor observations from the environment, inject actions, and provide responses for any actions they recognized as valid.

## 5 Models

In this section, we introduce the reinforcement learning and behavior cloning agents used in our experiments. Additional details and hyperparameters are provided in APPENDIX A.

**Deep Reinforcement Relevance Network (DRRN):** The DRRN (He et al., 2016) is a fast and strong reinforcement learning baseline that is frequently used to deliver near state-of-the-art performance in a variety of text games (e.g. Xu et al., 2020; Yao et al., 2020; Wang et al., 2022). At each step, the DRRN separately encodes the observation and candidate actions using several GRUs (Cho et al., 2014). A Deep Q-Network is then used to estimate Q-values for each *(observation, candidate action)* pair. The candidate action with the highest predicted Q-value will be chosen as the next action.

**Behavior Cloning:** Behavior cloning (Torabi et al., 2018) is a form of imitation learning similar to the Decision Transformer (Chen et al., 2021) that models reinforcement learning as a sequence-to-sequence problem, predicting the next action given a series of previous observations. We follow the strategy of Ammanabrolu et al. (2021) in adapting behavior cloning to text games, where the model input at step $t$ includes the task description, current state observation, previous action, and previous state observation $(d, o_t, a_{t-1}, o_{t-1})$. During training, the agent is fine-tuned on gold trajectories, where the training target is to generate action $a_t$ from the gold trajectories. During evaluation, the agent performs inference online in the text game environment. For experiments reported here, we used a T5-base model (Raffel et al., 2020).

### 5.1 Oracle Agents and Gold Trajectories

To generate training data for the behavioral cloning model, we implement oracle agents that generate optimal and generalizable solution trajectories for each benchmark. For example, an oracle agent for an arithmetic game always reads the math problem, picks up the object with the same quantity as the

| | DRRN | | | | Behavior Cloned Transformer | | | |
|---|---|---|---|---|---|---|---|---|
| | Baseline | | NeuroSymbolic | | Baseline | | NeuroSymbolic | |
| Benchmark | Score | Steps | Score | Steps | Score | Steps | Score | Steps |
| MapReader | 0.02 | 50 | 0.02 | 50 | 0.71 | 27 | **1.00** | **10** |
| Arithmetic | 0.17 | 10 | 0.14 | 7 | 0.56 | 5 | **1.00** | **5** |
| Sorting | 0.03 | 21 | 0.03 | 18 | 0.72 | 7 | **0.98** | **8** |
| TWC | 0.57 | 27 | 0.37 | 34 | 0.90 | 6 | **0.97** | **3** |
| Average | 0.20 | 27 | 0.14 | 27 | 0.72 | 11 | **0.99** | **7** |

Table 2: Average model performance across 100 games in the unseen test set. *Scores* are normalized to between 0 and 1 (higher is better), while *steps* represents the number of steps an agent takes in the environment (lower is better). Neurosymbolic performance reflects when models have access to symbolic modules in their action space.

math problem answer, then places that object in the answer box. For experiments using symbolic modules, we further insert appropriate module actions when the agent requires that information to complete the next step – for example, using the calculator module after reading the math problem in the arithmetic game.

## 6 Results and Discussion

The results of both DRRN and behavior cloning experiments across each benchmark are shown in Table 2. We report the average model performance across 100 games in the unseen test set. The DRRN achieves a low average performance of 0.20 without modules, while adding symbolic modules into the action space does not improve performance. In contrast, the behavior cloned T5 model has a moderate average performance of 0.72 without modules, while adding symbolic modules increases average task performance to 0.99, nearly solving each task. Symbolic modules also make the behavior cloned agent more efficient, reducing the average steps required to complete the tasks from 11 to 7, matching oracle agent efficiency.

**Why does behavior cloning perform well?** The baseline behavior cloned transformer achieves moderate overall performance, likely owing at least in part due to its use of gold trajectories for training. Large pretrained transformers contain a variety of common sense knowledge and reasoning abilities (Zhou et al., 2020; Liu et al., 2022c) which likely contributes to the high performance on TWC, where the model only needs to match objects with their common sense locations. In contrast, while transformers have some arithmetic abilities, their accuracy tends to vary with the frequency of specific tokens in the training data (Razeghi et al., 2022), likely causing the modest performance on the Arithmetic game. Here, we show that instead of increasing the size of training data, transformers can be augmented with symbolic modules that perform certain kinds of reasoning with high accuracy. Compared to the DRRN, the presence of gold trajectories for training allows the behavior cloned transformer to efficiently learn how to capitalize on the knowledge available from those modules.

**Why does the DRRN perform poorly?** We hypothesize that two considerations make these tasks difficult for the Deep Reinforcement Relevance Network. The model frequently tries to select actions that lead to immediate reward (such as immediately picking the correct number of objects in the arithmetic game), without having first done the prerequisite actions (like reading or solving the math problem) that would naturally lead it to select that action. This creates an ungeneralizable training signal, causing the model to fail to learn the task. In addition, the action spaces for each game are generally large – baseline games contain between 5 and 30 possible valid actions at each step (see Table 4 in the APPENDIX), resulting in up to 24 million possible trajectories up to 5 steps, which is challenging to explore. Inspired by Liu et al. (2022a), our future work will aim to overcome these limitations, and allow reinforcement learning models to learn to efficiently and effectively exploit information from symbolic modules.

**How does performance compare against other agents?** While most environments used in this work are new, TEXTWORLD COMMON SENSE is an existing benchmark. Figure 3 compares the Neurosymbolic Behavior Cloned Transformer against recent models that use a combination of reinforcement learning, logic, knowledge resources, and case-based reasoning. While the performance is not directly comparable – here, we use the TEXTWORLDEXPRESS reimplementation of TWC with supervised learning, while other models use

| Model | Score | Steps |
|---|---|---|
| SceneIt (Murugesan et al., 2022) | 0.88 | 20 |
| Bike+CBR (Atzeni et al., 2022) | 0.93 | 17 |
| SceneGraph (Tanaka et al., 2022) | 0.91 | 17 |
| IG (Basu et al., 2021) | 0.92 | 13 |
| BCT Baseline (Ours) | 0.90 | 6 |
| **BCT+NeuroSymbic (Ours)** | **0.97** | **3** |

Table 3: A comparison of performance on TWC on unseen games on the "easy" setting. Note that models may not be directly comparable, as this work uses the TEXTWORLDEXPRESS reimplementation of TWC, and supervised learning. *Scores* are normalized to between 0 and 1 (higher is better), while *steps* represents the number of steps an agent takes in the environment (lower is better).

the original implementation with a mix of reinforcement learning and case-based reasoning – we can make the high-level observation that the performance of both the baseline and Neurosymbolic Behavior Cloned Transformer meets or exceeds the scores of previous models, while generating paths that are more efficient – by a factor of up to 7x.

## 7 Conclusion

In this paper, we present an approach to neurosymbolic reasoning for text games using action space injection that can be easily adapted to existing text game environments. For models that are capable of exploiting the information provided by the symbolic modules, this technique allows agents to inexpensively augment their reasoning skills to solve more complex tasks. We empirically demonstrate this approach can substantially increase task performance on four benchmark games using a behavior cloned transformer.

## Limitations

Two assumptions highlight core limitations in the scope of our results for augmenting models with neurosymbolic reasoning: the privileged access to a list of valid actions, and the use of gold trajectories for training the behavior cloned transformer.

**Valid Actions:** One of the central challenges with text games is that the space of possible action utterances is large, and text game parsers recognize only a subset of possible actions (e.g. *take apple on the table*) while being unable to successfully interpret a broader range of more complex utterances (e.g. *take the red fruit near the fridge*). As a result, nearly all contemporary models (e.g. Ammanabrolu and Hausknecht, 2020; Adhikari et al., 2020; Murugesan et al., 2021) make use of the *valid action aid* (Hausknecht et al., 2020), where at a given step the model is provided with an exhaustive list of possible valid actions from the environment simulator, from which one action is chosen. The models presented here similarly use this aid. The DRRN functions essentially as a ranker to select the most probable next action. The behavior cloned transformer generates a candidate action that is aligned using cosine similarity with the list of valid actions, where the action with the highest overlap is chosen as the next action. Overcoming the *valid action aid* will generally require either more complex simulation engines capable of interpreting a wider variety of intents from input actions, or models that learn sets of valid actions from a large amount of training data – though these generally demonstrate lower performance than those using valid actions (e.g. Yao et al., 2020).

**Gold Trajectories:** In this work we demonstrate a substantial improvement in the performance of a behavior cloned transformer when augmented with neurosymbolic reasoning, but this requires the use of gold trajectories demonstrating the use of those symbolic modules. Gold training data is not available in many reinforcement learning applications, and the model comparison we perform (DRRN versus behavior cloning) is meant to highlight the capacity for the behavior cloned model to learn to make use of symbolic modules through gold demonstrations, rather than to suggest the DRRN is incapable of this. In future work, we aim to develop training procedures to allow models that do not have the benefit of using gold trajectories to make use of symbolic modules.

## Ethics Statement

**Broader Impacts:** As noted by Ammanabrolu and Riedl (2021), the ability to perform long-term multi-step reasoning in complex, interactive, partially-observable environments has downstream applications beyond playing games. Text games are platforms upon which to explore interactive, situated communication such as dialogue. Although reinforcement learning is applicable to many sequential decision making domains, our setting is most relevant to creating agents that affect change via language. This mitigates physical risks prevelant in robotics, but not cognitive and emotional risks, as any system capable of generating natural

language is capable of biased language use (Sheng et al., 2021).

**Intended Use:** The method described in this paper involves fine-tuning a large pretrained transformer model. The data generated for fine-tuning was generated by gold agents, and not collected from human participants. The trained models are intended to operate on these benchmark tasks that assess reasoning capacities in navigation, arithmetic, and other common sense competencies. Large language models have been shown to exhibit a variety of biases (e.g. Nadeem et al., 2021) that may cause unintended harms, particularly (in the context of this work) in unintended use cases.

**Computation Time:** Training large models can involve a large carbon footprint (Strubell et al., 2019), or decrease the availability of a method due to the barriers in accessing high performance compute resources. The proposed technique can reduce the need for large models by augmenting smaller models with more complex reasoning through symbolic modules. The behavior cloning experiments achieve strong performance with T5-base, highlighting the capacity of modest models that can be run with workstation GPUs to be better exploited for complex reasoning tasks.

## Acknowledgements

This work supported in part by National Science Foundation (NSF) award #1815948 to PJ, and the Allen Institute for Artificial Intelligence (AI2).

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

Prithviraj Ammanabrolu and Mark Riedl. 2021. Learning knowledge graph-based world models of textual environments. In *Thirty-fifth Conference on Neural Information Processing Systems (NeurIPS)*.

Prithviraj Ammanabrolu, Jack Urbanek, Margaret Li, Arthur Szlam, Tim Rocktäschel, and Jason Weston. 2021. How to motivate your dragon: Teaching goal-driven agents to speak and act in fantasy worlds. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 807–833, Online. Association for Computational Linguistics.

Mattia Atzeni, Shehzaad Zuzar Dhuliawala, Keerthiram Murugesan, and MRINMAYA SACHAN. 2022. Case-based reasoning for better generalization in textual reinforcement learning. In *International Conference on Learning Representations*.

Kinjal Basu, Keerthiram Murugesan, Mattia Atzeni, Pavan Kapanipathi, Kartik Talamadupula, Tim Klinger, Murray Campbell, Mrinmaya Sachan, and Gopal Gupta. 2021. A hybrid neuro-symbolic approach for text-based games using inductive logic programming. In *Proceedings of the 1st Workshop on Combining Learning and Reasoning: Programming Languages, Formalisms, and Representations*.

Erik Cambria, Qian Liu, Sergio Decherchi, Frank Xing, and Kenneth Kwok. 2022. SenticNet 7: A commonsense-based neurosymbolic AI framework for explainable sentiment analysis. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 3829–3839, Marseille, France. European Language Resources Association.

Subhajit Chaudhury, Prithviraj Sen, Masaki Ono, Daiki Kimura, Michiaki Tatsubori, and Asim Munawar. 2021. Neuro-symbolic approaches for text-based policy learning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3073–3078, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. 2021. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar. Association for Computational Linguistics.

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.

Ji He, Mari Ostendorf, Xiaodong He, Jianshu Chen, Jianfeng Gao, Lihong Li, and Li Deng. 2016. Deep reinforcement learning with a combinatorial action

space for predicting popular Reddit threads. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1838–1848, Austin, Texas. Association for Computational Linguistics.

Michael Janner, Qiyang Li, and Sergey Levine. 2021. Offline reinforcement learning as one big sequence modeling problem. In *Advances in Neural Information Processing Systems*.

Peter Jansen. 2022. A systematic survey of text worlds as embodied natural language environments. In *Proceedings of the 3rd Wordplay: When Language Meets Games Workshop (Wordplay 2022)*, pages 1–15, Seattle, United States. Association for Computational Linguistics.

Peter A Jansen and Marc-Alexandre Côté. 2022. Textworldexpress: Simulating text games at one million steps per second. *arXiv preprint arXiv:2208.01174*.

Haozhe Ji, Pei Ke, Shaohan Huang, Furu Wei, Xiaoyan Zhu, and Minlie Huang. 2020. Language generation with multi-hop reasoning on commonsense knowledge graph. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 725–736, Online. Association for Computational Linguistics.

Daiki Kimura, Subhajit Chaudhury, Masaki Ono, Michiaki Tatsubori, Don Joven Agravante, Asim Munawar, Akifumi Wachi, Ryosuke Kohita, and Alexander Gray. 2021a. LOA: Logical optimal actions for text-based interaction games. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 227–231, Online. Association for Computational Linguistics.

Daiki Kimura, Masaki Ono, Subhajit Chaudhury, Ryosuke Kohita, Akifumi Wachi, Don Joven Agravante, Michiaki Tatsubori, Asim Munawar, and Alexander Gray. 2021b. Neuro-symbolic reinforcement learning with first-order logic. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3505–3511, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Iou-Jen Liu, Xingdi Yuan, Marc-Alexandre Côté, Pierre-Yves Oudeyer, and Alexander G. Schwing. 2022a. Asking for knowledge: Training rl agents to query external knowledge using language. *ArXiv*, abs/2205.06111.

Jiacheng Liu, Alisa Liu, Ximing Lu, Sean Welleck, Peter West, Ronan Le Bras, Yejin Choi, and Hannaneh Hajishirzi. 2022b. Generated knowledge prompting for commonsense reasoning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3154–3169, Dublin, Ireland. Association for Computational Linguistics.

Jiacheng Liu, Alisa Liu, Ximing Lu, Sean Welleck, Peter West, Ronan Le Bras, Yejin Choi, and Hannaneh Hajishirzi. 2022c. Generated knowledge prompting for commonsense reasoning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3154–3169.

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. In *Thirty Fifth AAAI Conference on Artificial Intelligence*.

Keerthiram Murugesan, Subhajit Chaudhury, and Kartik Talamadupula. 2022. Eye of the beholder: Improved relation generalization for text-based reinforcement learning agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 11094–11102.

Moin Nadeem, Anna Bethke, and Siva Reddy. 2021. StereoSet: Measuring stereotypical bias in pretrained language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5356–5371, Online. Association for Computational Linguistics.

Philip Osborne, Heido Nomm, and André Freitas. 2021. A survey of text games for reinforcement learning informed by natural language. *Transactions of the Association for Computational Linguistics*, 10:873–887.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Yasaman Razeghi, Robert L Logan IV, Matt Gardner, and Sameer Singh. 2022. Impact of pretraining term frequencies on few-shot reasoning. *arXiv preprint arXiv:2202.07206*.

Emily Sheng, Kai-Wei Chang, Prem Natarajan, and Nanyun Peng. 2021. Societal biases in language generation: Progress and challenges. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4275–4293, Online. Association for Computational Linguistics.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.

Tsunehiko Tanaka, Daiki Kimura, and Michiaki Tatsubori. 2022. Commonsense knowledge from scene graphs for textual environments. *arXiv preprint arXiv:2210.14162*.

Faraz Torabi, Garrett Warnell, and Peter Stone. 2018. Behavioral cloning from observation. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 4950–4957. International Joint Conferences on Artificial Intelligence Organization.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. 2022. Scienceworld: Is your agent smarter than a 5th grader? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Leon Weber, Pasquale Minervini, Jannes Münchmeyer, Ulf Leser, and Tim Rocktäschel. 2019. NLProlog: Reasoning with weak unification for question answering in natural language. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6151–6161, Florence, Italy. Association for Computational Linguistics.

Yunqiu Xu, Meng Fang, Ling Chen, Yali Du, Joey Tianyi Zhou, and Chengqi Zhang. 2020. Deep reinforcement learning with stacked hierarchical attention for text-based games. *Advances in Neural Information Processing Systems*, 33:16495–16507.

Shunyu Yao, Karthik Narasimhan, and Matthew Hausknecht. 2021. Reading and acting while blindfolded: The need for semantics in text game agents. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3097–3102, Online. Association for Computational Linguistics.

Shunyu Yao, Rohan Rao, Matthew Hausknecht, and Karthik Narasimhan. 2020. Keep CALM and explore: Language models for action generation in text-based games. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8736–8754, Online. Association for Computational Linguistics.

Xingdi Yuan, Marc-Alexandre Côté, Alessandro Sordoni, Romain Laroche, Rémi Tachet des Combes, Matthew J. Hausknecht, and Adam Trischler. 2018. Counting to explore and generalize in text-based games. *ArXiv*, abs/1806.11525.

Tom Zahavy, Matan Haroush, Nadav Merlis, Daniel J Mankowitz, and Shie Mannor. 2018. Learn what not to learn: Action elimination with deep reinforcement learning. In *NeurIPS*.

Rowan Zellers, Ari Holtzman, Matthew Peters, Roozbeh Mottaghi, Aniruddha Kembhavi, Ali Farhadi, and Yejin Choi. 2021. PIGLeT: Language grounding through neuro-symbolic interaction in a 3D world. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2040–2050, Online. Association for Computational Linguistics.

Xuhui Zhou, Yue Zhang, Leyang Cui, and Dandan Huang. 2020. Evaluating commonsense in pre-trained language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9733–9740.

# A Appendix: Experiment Details

## A.1 Training and evaluation sets

For each game, we randomly generate 100 parametric variations for each of the train, development, and test sets. To encourage and evaluate generality, problems are unique across sets – for example, arithmetic problems (for the Arithmetic game) or task objects (for TWC) found in the training set are not found in the development or test sets.

## A.2 Hyperparameters

Following standard practice (e.g. (Wang et al., 2022; Xu et al., 2020; Hausknecht et al., 2020)), the DRRN models are trained for 100k steps. We parallelly train DRRN on 16 environment instances with five different random seeds and the average results are reported. The behavior cloned transformers are trained for between 2 and 20 epochs, with the best model (as evaluated on the development set) used for evaluating final performance on the test set. Trained models are evaluated on all 100 parametric variations in the development or test set. Environments are limited to 50 steps, such that if the agent exceeds this many steps without reaching an end state, the score at the last step is taken to be the final score, and the environment resets. Model training time varied between 1 hour and 12 hours, with the TWC model that includes a large number of symbolic module actions requiring the largest training time.

## A.3 Implementation details

We make use of an existing DRRN implementation[2] and adapted it to the TEXTWORLDEXPRESS environment. At each step, the current game state observation, task description, inventory information, and the current room description are concatenated into one string and encoded by a GRU. All

[2] https://github.com/microsoft/tdqn

| Benchmark | No Modules | | | With Symbolic Modules | | |
|---|---|---|---|---|---|---|
| | Min | Avg | Max | Min | Avg | Max |
| MapReader | 4 | 6.2 | 22 | 6 | 9.3 | 26 |
| Arithmetic | 9 | 14.3 | 52 | 17 | 21.5 | 53 |
| Sorting | 5 | 9.3 | 35 | 7 | 11.0 | 37 |
| TWC | 3 | 6.3 | 14 | 544 | 547.8 | 561 |
| Average | 5.3 | 9.0 | 30.8 | 143.5 | 147.4 | 169.3 |

Table 4: The minimum, mean, and maximum number of valid actions per step, for each benchmark. Values represent averages determined using a random agent that is run to 50 steps for on 10 training episodes per benchmark.

candidate actions are encoded by another GRU. The Q-value of each encoded (observation, candidate action) pair is then estimated by a Q-network consists of two linear layers. During training, the next action is sampled from all candidate actions based on the estimated Q-values. During evaluation, the action with the highest estimated Q-value is chosen as the next action.

For the behavior cloned transformer, the input string of the T5 model at step $t$ are formatted as:

$d$ **</s> OBS** $o_t$ **</s> INV** $o_t^{inv}$ **</s> LOOK** $o_t^{look}$ **</s> <extra_id_0> </s> PACT** $a_{t-1}$ **</s> POBS** $o_{t-1}$ **</s>**

where $d$ is the task description, **</s>** and **<extra_id_0>** are the special tokens for separator and mask for text to generate used by the T5 model, **OBS**, **INV**, **LOOK**, **PACT**, and **POBS** are the special tokens representing observation $o_t$, inventory information $o_t^{inv}$, the current room description obtained by the "look around" action $o_t^{look}$, previous action $a_{t-1}$, and previous observation $o_{t-1}$, respectively. We use beam search to generate the top 16 strings from the T5 model, and choose the first string that is a valid action as the action to take. In the case where the model does not generate an exact match, we use cosine similarity to pick the valid action that has the highest unigram overlap with an action generated by T5.

## B Environments and Symbolic Modules

**Action Space:** The number of valid actions per step for each benchmark is shown in Table 4, with these values collected by a random agent that runs for 50 steps across 10 training episodes. Environments contain an average of 9 valid actions per step (range 5 to 30), not including actions injected from the symbolic modules. After adding the module actions, the action space becomes up to twice as large for the MapReader, Arithmetic, and Sorting benchmarks. The knowledge base module for TWC adds approximately 530 actions, substantially increasing the action space.

**Text World Common Sense (TWC) Game**

**Task Description:** Your task is to pick up objects, then place them in their usual locations in the environment.

You are in the corridor. In one part of the room you see a shoe cabinet that is closed. There is also **a white coat**. You also see a key holder, that has nothing on it. In another part of the room you see a hat rack, that has nothing on it. In one part of the room you see **a coat hanger**, that has nothing on it. There is also a umbrella stand, that has nothing on it.
Inventory:
Your inventory is currently empty.
> *query* ***white coat***

The results are:
**white coat** located **coat hanger**
**white coat** located wardrobe
Inventory:
Your inventory is currently empty.
> *take* ***white coat***

You take the **white coat**.
Inventory:
**a white coat**
> *put* ***white coat*** *in* ***coat hanger***

*Game completed.*

Table 5: An example of a Text World Common Sense (TWC) game. *Actions* for the solution are italicized. Relevant **task objects** and **target locations** are bolded, while distractor locations are underlined. Symbolic module actions and their responses are highlighted in *blue*.

### B.1 Text World Common Sense (TWC)

A benchmark common sense reasoning task where agents must collect objects from the environment (e.g. *dirty socks*) and place them in their canonical locations (e.g. *washing machine*). We use the TEXTWORLDEXPRESS implementation of the original environment (Murugesan et al., 2021), in a setting analogous to easy difficulty, where the agent must pick and place one misplaced item in the environment. The agent receives a score of 0.5 for picking up the object, and 1.0 for placing it in the correct location. Table 5 shows an example of the TWC Game in which the agent places a clean shirt in the wardrobe with the help of querying the

**Arithmetic Game**

**Task Description:** Your first task is to solve the math problem. Then, pick up the item with the same quantity as the math problem answer, and place it in the box.

You are in the kitchen. In one part of the room you see a fridge that is closed. There is also a dining chair that has 11 tangerines, 33 papayas, 242 strawberries, and 20 peaches on it. You also see a box, that is empty. In another part of the room you see a math problem. In one part of the room you see a dishwasher that is closed. There is also a trash can that is closed. You also see an oven. In another part of the room you see a cutlery drawer that is closed. In one part of the room you see a stove. There is also a kitchen cupboard that is closed. You also see a counter that has 6 oranges, and **2 bananas** on it.
Inventory:
Your inventory is currently empty.
> *take math problem*

You take the math problem.
Inventory:
a math problem
> *read math problem*

Your task is to solve the following math problem: divide 22 by 11. Then, pick up the item with the same quantity as the answer, and place it in the box.
Inventory:
a math problem
> *div 22 11*

The result of dividing 22 by 11 is 2.
Inventory:
a math problem
> *take **2 bananas***

You take the **2 bananas**.
Inventory:
a math problem
**2 bananas**
> *put **2 bananas** in box*

*Game completed.*

Table 6: An example Arithmetic game. *Actions* for the solution are italicized. Relevant **task objects** are bolded, while distractor objects are underlined. Symbolic module actions and their responses are highlighted in *blue*.

common sense location of a clean shirt.

*Knowledge Base Module:* We pair TWC with a symbolic module that provides queries to a simple knowledge base of <*object, hasCanonicalLocation, container*> triples. The symbolic module generates and accepts actions of the form QUERY <QUERY TOKENS>, where <QUERY TOKENS> corresponds to all object and container names in the knowledge base. This results in an increase of approximately 530 actions in the action space.

### B.2 Arithmetic Game

The Arithmetic game requires agents to read a math problem, solve it, then perform a pick-and-place task based on the answer. For example, the agent may read the math problem (*"Take the bundle of objects that is equal to 3 multiplied by 6, and place them in the box"*), and must then perform the arithmetic then *take 18 apples* and place them in the answer box. Distractor objects are populated corresponding to performing the arithmetic incorrectly (for example, including 3 oranges, corresponding to subtracting 3 from 6, and 2 pears, corresponding to 6 divided by 3), with the condition that results are positive integer values. Agents receive a score of 0.5 for picking up the correct object, and 1.0 for completing the task successfully. An example playthrough of the Arithmetic game is in Table 6.

*Arithmetic Module:* We pair the Arithmetic game with an Arithmetic module that adds actions for addition, subtraction, multiplication, and division. To reduce the complexity of the action space, only actions with arguments from the current math problem are enumerated (e.g. *add 3 6, sub 3 6, sub 6 3, mul 3 6, div 3 6, div 6 3*).

### B.3 Sorting Game

The sorting game is a pick-and-place game that presents an agent with 3 to 5 objects, and asks the agent to place them in an answer box one at a time based on order of increasing quantity. To add complexity to the game, quantities optionally include units (e.g. *5kg of copper, 8mg of steel, 2g of iron*) across measures of volume, mass, or length. The agent score is the normalized proportion of objects sorted in the correct order, where perfect sorts receive a score of 1.0, and errors cause the score to revert to zero and the game to end. An example playthrough of the Sorting game is in Table 7.

*Sorting Module:* The sorting module monitors observations for mentions of objects (nouns) that include quantities, while also interpreting and normalizing quantities based on known units. The module injects two actions: sort ascending, and sort descending, that provides the user with a sorted list of objects.

### B.4 MapReader Game

MapReader is a navigation oriented pick-and-place game similar to Coin Collector (Yuan et al., 2018), with the added complexity that the agent is provided with a map of the environment that can be used to more efficiently navigate. Environments and their maps are randomly generated to contain up to 15 locations drawn from 50 locations in Coin Collector. The agent begins in a randomly chosen location, and is asked to move to a target location (e.g. *the kitchen*), take a coin, then return to the starting location and place it in a box. Target locations are randomly chosen to be between 1 and 4 steps from the starting location. The most efficient solution method is to read the map, determine the shortest path between the agents current location

**Sorting Game**

**Task Description:** Your task is to sort objects by quantity. First, place the object with the smallest quantity in the box. Then, place the objects with the next smallest quantity in the box, and repeat until all objects have been placed in the box.

You are in the kitchen. In one part of the room you see a fridge that is closed. There is also a counter that has **15kg of cedar**, and **21kg of marble** on it. You also see a dining chair that has **25g of oak**, and **47g of brick** on it. In another part of the room you see a box, that is empty. In one part of the room you see a dishwasher that is closed. There is also a trash can that is closed. You also see an oven. In another part of the room you see a cutlery drawer that is closed. In one part of the room you see a stove. There is also a kitchen cupboard that is closed.
Inventory:
Your inventory is currently empty.
> *sort ascending*

The observed items, sorted in order of increasing quantity, are: **25 g of oak**, **47 g of brick**, **15 kg of cedar**, **21 kg of marble**.
Inventory:
Your inventory is currently empty.
> *take **25g of oak***

You take the **25g of oak**.
Inventory:
**25g of oak**
> *put **25g of oak** in box*

You put the **25g of oak** in the box.
Inventory:
Your inventory is currently empty.
> *take **47g of brick***

You take the **47g of brick**.
Inventory:
**47g of brick**
> *put **47g of brick** in box*

You put the **47g of brick** in the box.
Inventory:
Your inventory is currently empty.
> *take **15kg of cedar***

You take the **15kg of cedar**.
Inventory:
**15kg of cedar**
> *put **15kg of cedar** in box*

You put the **15kg of cedar** in the box.
Inventory:
Your inventory is currently empty.
> *take **21kg of marble***

You take the **21kg of marble**.
Inventory:
**21kg of marble**
> *put **21kg of marble** in box*

*Game completed.*

Table 7: An example Sorting game. *Actions* for the solution are italicized. Relevant **task objects** are bolded. Symbolic module actions and their responses are highlighted in *blue*.

and target location, follow that path to retrieve the coin, then follow the path in reverse to return the coin to the starting location. The agent receives a score of 0.5 for retrieving the coin, and 1.0 for placing the coin in the box at the start location. An example of the MapReader game is shown in Table 8.

*Navigation Module:* We pair MapReader with a navigation module that scrapes the environment for both complete map information (obtained if the agent chooses to read the map), as well as partial information such as the current location (e.g. *"You are in the kitchen"*) and connecting locations (e.g. *"To the north you see the living room"*) that can be used to incrementally build a map. The module adds actions that, if selected, provide the next step in the shortest path to navigate to all known locations in the environment (e.g. *next step to living room*, *next step to garage*, ...).

**MapReader Game**

**Task Description:** Your task is to take the coin that is located in **the laundry room**, and put it into the box found in **the foyer**. A map is provided, that you may find helpful.

You are in **the foyer**. In one part of the room you see a box, that is empty.

To the East you see the corridor.
Inventory:
a map
> *read map*

The map reads:
The living room connects to the backyard and corridor.
The garage connects to the driveway.
**The laundry room** connects to the corridor.
The backyard connects to the living room, alley, kitchen and sideyard.
The bedroom connects to the corridor.
The sideyard connects to the backyard and driveway.
The kitchen connects to the bathroom, pantry and backyard.
The supermarket connects to the alley.
**The foyer** connects to the corridor.
The pantry connects to the kitchen.
The driveway connects to the sideyard, alley and garage.
The street connects to the alley.
The alley connects to the driveway, supermarket, street and backyard.
The bathroom connects to the kitchen.
The corridor connects to the living room, foyer, bedroom and laundry room.
Inventory:
a map
> *next step to laundry room*

The next location to go to is: corridor
Inventory:
a map
> *move east*

You are in the corridor. In one part of the room you see a shoe cabinet that is closed. There is also a key holder, that has nothing on it. You also see a hat rack, that has nothing on it. In another part of the room you see a coat hanger, that has nothing on it. In one part of the room you see a umbrella stand, that has nothing on it.
To the North you see the laundry room. To the South you see the living room. To the East you see the bedroom. To the West you see **the foyer**.
Inventory:
a map
> *next step to **laundry room***

The next location to go to is: **laundry room**
Inventory:
a map
> *move north*

*continued...*

You are in **the laundry room**. In one part of the room you see a coin. There
is also a bench, that has nothing on it. You also see a washing machine that is closed. In another part of the room you see a work table, that has nothing on it. In one part of the room you see a laundry basket, that has nothing on it. There is also a clothes drier that is closed.
To the South you see the corridor.
Inventory:
a map
> *take coin*

You take the coin.
Inventory:
a map
a coin
> *next step to **foyer***

The next location to go to is: corridor
Inventory:
a map
a coin
> *move south*

You are in the corridor. In one part of the room you see a shoe cabinet that is closed. There is also a key holder, that has nothing on it. You also see a hat rack, that has nothing on it. In another part of the room you see a coat hanger, that has nothing on it. In one part of the room you see a umbrella stand, that has nothing on it.
To the North you see **the laundry room**. To the South you see the living room. To the East you see the bedroom. To the West you see **the foyer**.
Inventory:
a map
a coin

> *next step to **foyer***
The next location to go to is: **foyer**
Inventory:
a map
a coin
> *move west*

You are in **the foyer**. In one part of the room you see a box, that is empty.
To the East you see the corridor.
Inventory:
a map
a coin
> *put coin in box*

*Game completed.*

Table 8: An example of a MapReader game. *Actions* for the solution are italicized. The **starting location** and the **target location** are bolded. Symbolic module actions and their responses are highlighted in *blue*.