Language Guided Exploration for RL Agents in Text Environments

Hitesh Golchha[◇], **Sahil Yerawar**[◇], **Dhruvesh Patel**[◇], **Soham Dan**[△], **Keerthiram Murugesan**[△]
[◇]Manning College of Information & Computer Sciences, University of Massachusetts Amherst
[△]IBM Research
{hgolchha,syerawar,dhruveshpate}@cs.umass.edu
{soham.dan,keerthiram.murugesan}@ibm.com

Abstract

Real-world sequential decision making is characterized by sparse rewards and large decision spaces, posing significant difficulty for experiential learning systems like *tabula rasa* reinforcement learning (RL) agents. Large Language Models (LLMs), with a wealth of world knowledge, can help RL agents learn quickly and adapt to distribution shifts. In this work, we introduce Language Guided Exploration (LGE) framework, which uses a pre-trained language model (called GUIDE) to provide decision-level guidance to an RL agent (called EXPLORER). We observe that on ScienceWorld (Wang et al., 2022), a challenging text environment, LGE outperforms vanilla RL agents significantly and also outperforms other sophisticated methods like Behaviour Cloning and Text Decision Transformer. [1]

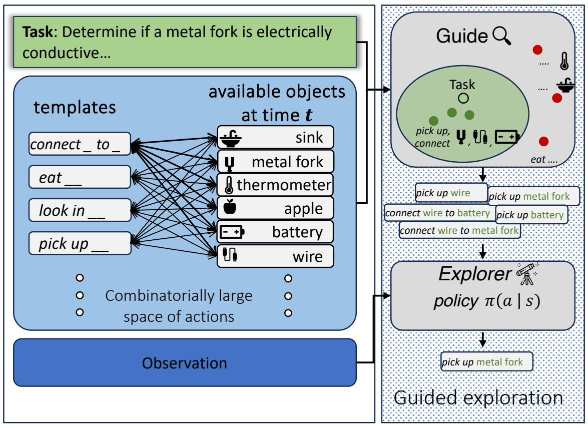

Figure 1: The Language Guided Exploration (LGE) Framework: The *Guide* uses contrastive learning to produce a set of feasible action given the task description thereby reducing substantially the space of possible actions. The *Explorer* , an RL agent, then uses the set of actions provided by the *Guide* to learn a policy and pick a suitable action using it.

1 Introduction

Reinforcement Learning (RL) has been used with great success for sequential decision making tasks. AI assistants whether text based (Li et al., 2022; Huang et al., 2022) or multi-modal (Chang et al., 2020; Patel et al., 2023), have to work with large action spaces and sparse rewards. In such settings, the approach of random exploration is inadequate. One needs to look for ways to use external information either to create a dense reward model or to reduce the size of action space. In this work we focus on the latter approach.

We make a simple observation that, in many cases, the textual description of the task or goal contains enough information to completely rule out certain actions, thereby greatly reducing the size of the effective action space. For example, as shown in Fig.1, if the task description is *"Determine if a metal fork is electrically conductive"*, then one can safely rule out actions that involve objects like sink, apple, and actions like eat, smell, etc. Motivated by this observation, we introduce the **L**anguage **G**uided **E**xploration (LGE) framework that uses an RL agent but augments it with a *Guide* model that uses world knowledge to rule out large number of actions that are infeasible or highly unlikely. Along with removing irrelevant actions, the frameworks supports generalization in unseen environments where new objects may appear. For example, if the model observed an apple in the environment during training, at test time, the environment may contain an orange instead. But the guide, which posses commonsense may understand that all fruits are equally relevant or irrelevant for the given task.

To test our framework, we use the highly challenging benchmark called SCIENCEWORLD (Wang et al., 2022), which consists of a purely text based environment where the observations, actions, and inventory are expressed using natural language text.

[1]The code for this work is available at https://github.com/hitzkrieg/drrn-scienceworld-clone.

Findings of the Association for Computational Linguistics: NAACL 2024, pages 93–102
June 16-21, 2024 ©2024 Association for Computational Linguistics

SCIENCEWORLD embodies the major challenges faced by RL agents in realy world applications: the template based actions with slots for verbs and objects produce a combinatorially large action space, the long natural language based observations make for a challenging state representation, and the rewards signals based mainly on the completion of challenging tasks create a delayed and sparse reward signal. Following are the main contributions of our work:

We propose a novel way to allow language guided exploration for RL agents. The task instructions are used to identify relevant actions using a contrastively trained LM. The proposed GUIDE model that uses contrastive learning has not been explored for text environments before.

We demonstrate significantly stronger results on the SCIENCEWORLD environment when compared to methods that use Reinforcement Learning, and more sophisticated methods like Behaviour Cloning (Wang et al., 2023) and Text Decision Transformer (Chen et al., 2021).

2 Related Work

Guided exploration for agents Language models have been used to aid with various parts of the RL pipeline (Du et al., 2023; Hendrycks et al., 2021). Due to its importance, the topic of exploration has received a lot of attention, with most classical methods focusing on intrinsically motivated exploration (Aubret et al., 2019). The recent advent of general-purpose language models has created a wave of resurgent interest in the topic, with works that use pre-trained language models in various ways to explore efficiently. Zhong et al. (2024) uses an LLM to remove undesirable actions by inspecting policy rollouts, followed by imitation learning using desirable rollouts. Yao et al. (2020), on the other hand simply trains a sequence model using imitation learning using gold trajectories. Triantafyllidis et al. (2023) proposes a framework for intrinsic exploration using LLMs. Previous works have also used symbolic knowledge from knowledge graphs along with language models to constrain or guide the exploration for RL agents (Basu et al., 2024; Atzeni et al., 2022; Ammanabrolu and Hausknecht, 2020). In this work, we take a different perspective, instead of guiding intrinsic exploration, we focus on reducing the action space in a meaningful manner such that simple classical exploration strategies like ϵ-exploration or entropy maximization can work well. We also propose a novel way of reducing the effective size of the action space using a contrastively trained language model.

Text-based games Text-based environments (Lebling et al., 1979; Yin and May, 2019; Murugesan et al., 2020; Côté et al., 2019) provide a low-cost alternative to complex 2D/3D environments, and real world scenarios, for the development of the high-level learning and navigation capabilities of the AI agents. As these games require multi-step reasoning, Reinforcement Learning serves as the default method to model these agents (He et al., 2016; Zahavy et al., 2018; Yao et al., 2020). Another way to model these agents is to view this task as a imitation learning task. Using knowledge graphs in conjunction with RL agents (Ammanabrolu and Hausknecht, 2020) utilizes the constrained actions from the Knowledge graphs created dynamically with each trajectory, however the filtered actions are still inefficient to distinguish relevant actions from irrelevant ones. Behavior cloning (Chen et al., 2021) shows success in the realm of text games, but the utilization of LLM's to predict the next action also suffers from the problem of removing spurious actions, even when the LLM's are fine-tuned on expert gold trajectories. In our work, we address the problem of reducing the action space of the agent, in order to learn the action selections in a much simpler way, directly addressing the problem of predicting irrelevant actions.

3 Methodology

Notation: The text environment, a partially observable Markov decision process (POMDP) consists of $(S, T, A, R, \tilde{O}, \Omega)$. In SCIENCEWORLD, along with the description of the current state, the observation also consists of a task description $\tau \in \mathcal{T}$ that stays fixed throughout the evolution of a single trajectory, i.e., $\tilde{O} = O \times \mathcal{T}$, where O is the set of textual descriptions of the state and $\mathcal{T}$ is the set of tasks (including different variations of each task). Note that the set of tasks are divided into different types and each type of task has different variations, i.e., $\mathcal{T} = \bigcup_{\gamma=1}^{\Gamma} \bigcup_{v=1}^{V_\gamma} \tau_{\gamma,v}$, where Γ is the number of task types and V_γ is the number of variations for the task type γ. Gold trajectories $G_{\gamma,v} = \{a_1, a_2, .., a_L\}$ are available for each γ, v.

3.1 The LGE framework

We propose a Language Guided Exploration Framework (LGE), which consists of an an RL agent called the EXPLORER, and an auxiliary model that scores each action called the GUIDE . The EXPLORER starts in some state sampled from initial state distribution d_0. At any time step t, a set of all valid actions $A_{\gamma,v,t}$ is provided by the environment. This set, constructed using the cross product of action templates and the set of objects (see Fig.1) is extremely large, typically in thousands. The pretrained GUIDE with frozen weights uses the task description $\tau_{\gamma,v}$, to produce a set of most relevant actions $\hat{A}_{\gamma,v,t} \subset A_{\gamma,v,t}$. With a probability $1-\epsilon$ (resp. ϵ), the EXPLORER samples an action from $\hat{A}_{\gamma,v,t}$ using its policy $\pi(a|s_t)$ (resp., from $A_{\gamma,v,t}$).

The pre-training of the GUIDE is outlined in 3.1.2, which is followed by the training of the EXPLORER. Algorithm 1 in Appendix A.1 outlines the steps involved in the LGE framework using a DRRN (He et al., 2016) based EXPLORER.

3.1.1 EXPLORER

The EXPLORER learns a separate policy π_γ for each task type $\gamma \in \Gamma$ by exploring the the environment.[2] We use the Deep Reinforcement Relevance Network (DRRN) (He et al., 2016) as our EXPLORER, as it has shown to be the strongest baseline in Wang et al. (2022). However, our framework allows to swap the DRRN with any other RL agent. The DRRN uses Q-learning with with prioritized experience replay to perform policy improvement using a parametric approximation of the action value function $Q(s,a)$.[3] The current state s_t is represented by concatenating the representations of the past observation o_{t-1}, inventory i_t and look around l_t, each encoded by separate GRUs, i.e., $h_{s_t} = (\, f_{\theta_o}(o_{t-1}) : f_{\theta_i}(i_t) : f_{\theta_l}(l_t)\,)$. Each relevant action $a \in A_{\text{rel},t}$ is encoded in the same manner: $h_{a_t} = f_{\theta_a}(a_t)$. Here f_* are the respective GRU encoders, θ_* their parameters and ":" denotes concatenation. The value function $Q(s,a)$ is represented using a linear layer over the concatenation of the action and state representations $Q(s_t, a_t|\theta) = W^T \cdot (h_{s_t} : h_{a_t}) + b$, where θ is a collection of θ_o, θ_i, θ_l, θ_a, W and b. During training, a stochastic policy based on the value function is used: $\hat{a} \sim \pi(a|s) \propto Q(s,a|\theta)$, while at inference time we use greedy sampling: $\hat{a} = \arg\max_a Q(s,a|\theta)$.

3.1.2 GUIDE

While LLMs are capable of scoring the relevant actions without any finetuning, we observed that due to the idiosyncrasies of the SCIENCEWORLD environment, it is beneficial to perform some finetuning. We use SimCSE (Gao et al., 2021), a contrastive learning framework, to finetune the GUIDE LM. The training data $\{\tau_i, G_i\}_{i=1}^M$, which consists of task descriptions $\tau_i = \tau_{\gamma,v} \in \mathcal{T}$ along with the set of corresponding gold actions $G_i = G_{\gamma,v}$. The GUIDE model g_ϕ is used to embed the actions and the task descriptions into a shared representation space where the similarity score of a task and an action is expressed as $s(\tau,a) = \frac{g_\phi(\tau) \cdot g_\phi(a)}{\lambda}$, with λ being the temperature parameter. The training objective is such that the embeddings of a task are close to those of the corresponding relevant actions, expressed using the following loss function:

$$l(\phi; \tau_i, G_i) = -\log \frac{e^{s(\tau_i,a^+)}}{e^{s(\tau_i,a^+)} + \sum_{a^- \in N_i} e^{s(\tau,a^-)}},$$

where $a^+ \sim G_i$ is a relevant action and N_i is a fixed sized subset of irrelevant actions.[4]

Note that since we only have access to a small amount of gold trajectories (3442) for training, we take special steps to avoid overfitting, which is the main issue plaguing the imitation learning based methods. First, we only provide the task description to the GUIDE and not the full state information. Second, unlike the EXPLORER, which uses different policy for each task type, we train a common GUIDE across all tasks, and its weights are frozen during the training of the EXPLORER.

4 Experiments and Results

As done in Wang et al. (2022), the variations of each task type are divided into training, validation and test sets. Both GUIDE and EXPLORER are trained only using the training variations.

4.1 Evaluating the GUIDE

Before the joint evaluation, we evaluate the GUIDE in isolation. We sample 5 variations from the validation set for each task type and compute the three

[2]The agent learns a separate policy of each task type but this policy is common across all variations for that particular task type.

[3]We follow the implementation of DRRN provided in Hausknecht et al. (2019).

[4]Details of the models used and the training data are provided in Appendix A.1.

Model	top-k	RSR	MAP	GAR	GAR (%)	GARR
G_g	50	0.71	0.52	N/A	N/A	N/A
G_τ	50	0.9	0.66			
Guide	50	0.99	0.68	7.4 ± 16.2	1.8 ± 9.4	0.31 ± 2.3
	20	0.94	0.67			
	10	0.79	0.61			

Table 1: Various metrics used to evaluate the GUIDE in isolation. Note that for the baselines G_g and G_τ, we cannot compute GAR.

metrics: GAR, RST and MAP. We use the following two intuitive but strong baselines:

(1) Gold per-task (G_τ): We create a set of 50 most most used actions in gold trajectories of all training variations of a particular task. The Gold per-task baseline, predicts an action to be relevant if it belongs to this set.

(2) Gold Global (G_g) : Similar to Gold per-task but we use 50 most used actions in Gold trajectories of all training variations for all tasks.

Gold Action Rank (GAR): At any time step t, $GAR(\gamma, v, t)$ is defined as the rank of the gold a_t in the set of valid actions $A_{\gamma,v,t}$, and the Gold Action Reciprocal Rank (GARR) is defined as 1/GAR. Since the size of $A_{\gamma,v,t}$ is variable for every t, we also report percent GAR. As seen in Table 1, the gold action gets an average rank of 7.42, which is impressive because $|A_{\gamma,v,t}|$ averages around 2000.

Relevant Set Recall (RSR): GAR ranks a single optimal action at any time, but multiple valid action sequences may exist for task completion. Although all viable paths are not directly accessible, we estimate them. For each time step t in variation $\tau_{\gamma,v}$, a set of gold relevant actions $\tilde{A}_{\gamma,v,t}$ is identified by intersecting the gold trajectory $G_{\gamma,v}$ with valid actions at t, so $\tilde{A}_{\gamma,v,t} = \{a \mid a \in G_{\gamma,v} \cap A_{\gamma,v,t}\}$. The GUIDE's effectiveness is measured by its recall of this set, considering its top-k predicted actions $\hat{A}_{\gamma,v,t}$. Relevant Set Recall (RSR) is calculated as $RSR(\gamma, v, t) = \frac{|\hat{A}_{\gamma,v,t} \cap \tilde{A}_{\gamma,v,t}|}{|\tilde{A}_{\gamma,v,t}|}$. As seen in Table 1, the GUIDE has almost perfect average recall of 0.99 while selecting top 50 actions for the EXPLORER at every step of the episode.

Mean Avg. Precision (MAP): The GUIDE also functions as a binary classifier, predicting the relevance of each action in $A_{\gamma,v,t}$. Using a threshold-free metric like average precision score (Pedregosa et al., 2011), the GUIDE achieves a superior average precision score of 0.68 compared to baselines. Coupled with perfect recall at 50, this indicates the GUIDE's strong generalization ability on new variations and robust performance across various thresholds. We observe that the threshold that produces best MAP is 0.52, which corresponds to $|\hat{A}_{\gamma,v,t}| = 28$ on average. So, to be conservative, we use $k = 50$ in the subsequent evaluations. Table 5 shows an example of the set of actions selected by GUIDE for the task "Change of state".

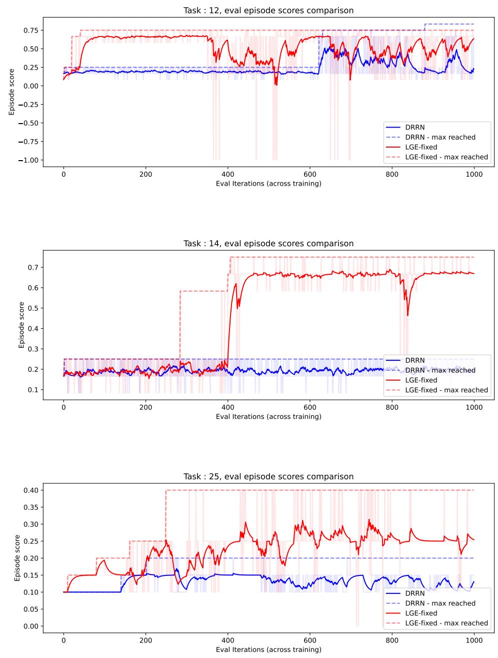

Figure 2: Episode scores on unseen variations of tasks 12, 14 and 25, plotted as the training progresses. The solid line plots are the exponential moving averages of episode scores, while the dotted lines are the maximum episode score achieved till that point. We see both the ability to reach higher scores and better sample efficiency for LGE as compared to the DRRN. This trend is observed in most tasks. Plots for more tasks are in the Appendix- Figure 3

4.2 Evaluating LGE

We follow the same evaluation protocol as (Wang et al., 2022) and evaluate two versions of the LGE framework, one with a fixed ϵ of 0.1 and the other with ϵ increasing from 0 to 1. Table 3 reports the means returns for each task.

LGE improves significantly over the RL baseline. The DRRN agent, which only uses RL, performs the best among the baselines. The proposed LGE framework (last two columns), improves the performance of DRRN on 20 out of 30 tasks. On average the LGE with $\epsilon = 0.1$, improves the mean

Relevant Gold Actions	Selected By GUIDE
open cupboard,focus on soap in kitchen, move soap in kitchen to metal pot, move metal pot to stove, go to outside,wait1, pick up thermometer, pick up metal pot,look around,activate stove	pick up metal pot,move metal pot to sink,pour metal pot into metal pot, open cupboard, activate stove,move metal pot to stove, pick up thermometer, open freezer, wait, go to outside, open glass jar,look around, open drawer in cupboard, open drawer in counter,open oven,move ceramic cup to sink,pick up ceramic cup,open fridge, open door to hallway, activate sink, mix metal pot, pour ceramic cup into ceramic cup, pick up sodium chloride, wait1, focus on metal pot, pick up soap in kitchen

Table 2: Column 1 shows the relevant gold actions for the task "Change of State (variation 1 from the dev set)", and column two shows the set of actions selected by the GUIDE. The missed gold actions are in Red, while selected gold actions are in Green

returns by 35% ($0.17 \rightarrow 0.23$). As seen in the Figure 2, for many tasks, LGE allows the Explorer to reach rewards earlier (leading to better sample efficiency), and often reach better states which were unattainable with DRRN.

LGE is better than much more complex, specialized methods. The behaviour cloning (BC) model, uses a Macaw (Tafjord and Clark, 2021) model fine-tuned on the gold trajectories to predict the next action. The Text Decision Transformer (TDT) (Chen et al., 2021) models the complete POMDP trajectories as a sequence and is capable of predicting actions that maximize long-term reward. As seen in Table 3, the simpler LGE framework outperforms both TDT and BC. This shows the importance of having an RL agent in the framework that can adapt to the peculiarities of the environment.

Using an increasing ϵ schedule is slightly worse LGE-inc uses an increasing ϵ schedule, where the reliance on the Guide is slowly weaned away, as the Explorer's learning matures. We observe that as the actions provided by the GUIDE almost always contain the right action, LGE-inc is almost competitive with LGE-fix, but slightly worse. LGE-inc should be better in more difficult environments with not-so-perfect Guide for better generalization.

Task	DRRN*	BC*	TDT*	LGE inc	LGE fix	Delta
T0	0.03	0.00	0.00	**0.04**	0.03	0.01(↑)
T1	0.03	0.00	0.00	0.03	0.03	0.00
T2	0.01	0.01	0.00	0.00	0.00	-0.01(↓)
T3	**0.04**	0.00	0.01	0.03	0.03	-0.01(↓)
T4	0.08	0.01	0.02	**0.09**	0.06	0.01(↑)
T5	0.06	0.01	0.02	0.07	**0.07**	0.01(↑)
T6	0.10	0.04	0.04	0.10	0.10	0.00
T7	0.13	0.03	0.07	0.13	0.13	0.00
T8	0.10	0.02	0.05	0.09	0.10	0.00
T9	0.07	0.05	0.05	0.06	0.07	0.00
T10	0.20	0.04	0.05	0.23	**0.27**	0.07(↑)
T11	0.19	0.21	0.19	**0.48**	0.47	0.29(↑)
T12	0.26	0.29	0.16	0.28	**0.6**	0.34(↑)
T13	0.56	0.19	0.17	0.55	**0.64**	0.08(↑)
T14	0.19	0.17	0.19	0.43	**0.65**	0.46(↑)
T15	0.16	0.03	0.05	**0.17**	0.17	0.01(↑)
T16	0.09	0.08	0.03	0.09	0.09	0.00
T17	0.20	0.06	0.10	**0.26**	0.25	0.06(↑)
T18	0.29	0.16	0.20	**0.30**	0.27	0.01(↑)
T19	0.11	0.05	0.07	0.11	0.11	0.00
T20	0.48	0.26	0.20	0.55	**0.62**	0.14(↑)
T21	0.31	0.02	0.20	**0.33**	0.28	0.02(↑)
T22	0.47	0.14	0.16	**0.54**	0.50	0.07(↑)
T23	0.10	0.02	0.07	**0.28**	0.14	0.18(↑)
T24	0.09	0.04	0.02	**0.18**	0.08	0.09(↑)
T25	0.13	0.05	0.04	0.24	**0.26**	0.13(↑)
T26	0.13	0.05	0.04	**0.25**	0.24	0.12(↑)
T27	0.13	0.04	0.04	0.23	**0.23**	0.10(↑)
T28	0.19	0.06	0.06	0.19	**0.22**	0.03(↑)
T29	0.17	0.13	0.05	0.17	0.16	0.00
Avg.	0.17	0.08	0.08	0.22	**0.23**	0.06(↑)

Table 3: Zero-shot performance of the agents on test variations of across all tasks. The columns with * are reported from Wang et al. (2022). The Delta column is the difference between DRRN and the best LGE model. The LGE performance values are averaged across 3 separate runs. The names of the tasks are in Table 4 in Appendix.

5 Conclusion

We proposed a simple and effective framework for using the knowledge in LMs to guide RL agents in text environments, and showed its effectiveness on the SCIENCEWORLD environment when used with DRRN. Our framework is generic and can extend to work with other RL agents. We believe that the positive results observed in our work will pave the way for future work in this area.

6 Limitations

This paper focuses on the ScienceWorld environment, which is an English only environment. Moreover, it focuses mainly on scientific concepts and skills. To explore other environments in different languages with different RL agents will be an interesting future work. There is also a large room for improvement to approach human-level performance which may come from improving the RL agent and using a larger LM as the Guide. Also, our current methodology requires supervised demonstrations to train the GUIDE, so extending this method to semi/un-supervised settings now becomes a relevant direction for future work.

Acknowledgement

We want to acknowledge the support by the IBM Research AI through the AI Horizons Network for this work. We also thank Shib Sankar Dasgupta for the helpful initial discussions.

References

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

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*.

Arthur Aubret, Laetitia Matignon, and Salima Hassas. 2019. A survey on intrinsic motivation in reinforcement learning.

Kinjal Basu, Keerthiram Murugesan, Subhajit Chaudhury, Murray Campbell, Kartik Talamadupula, and Tim Klinger. 2024. Explorer: Exploration-guided reasoning for textual reinforcement learning.

Chien-Yi Chang, De-An Huang, Danfei Xu, Ehsan Adeli, Li Fei-Fei, and Juan Carlos Niebles. 2020. Procedure planning in instructional videos. In *European Conference on Computer Vision*, pages 334–350. Springer.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. 2021. Decision transformer: Reinforcement learning via sequence modeling. *arXiv preprint arXiv:2106.01345*.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. 2019. Textworld: A learning environment for text-based games. *Computer Games*, page 41–75.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, abs/1810.04805.

Yuqing Du, Olivia Watkins, Zihan Wang, Cédric Colas, Trevor Darrell, Pieter Abbeel, Abhishek Gupta, and Jacob Andreas. 2023. Guiding pretraining in reinforcement learning with large language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 8657–8677. PMLR.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Matthew J. Hausknecht, Prithviraj Ammanabrolu, Marc-Alexandre Côté, and Xingdi Yuan. 2019. Interactive fiction games: A colossal adventure. In *AAAI Conference on Artificial Intelligence*.

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.

Dan Hendrycks, Mantas Mazeika, Andy Zou, Sahil Patel, Christine Zhu, Jesus Navarro, Dawn Song, Bo Li, and Jacob Steinhardt. 2021. What would jiminy cricket do? towards agents that behave morally. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. 2022. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning*, pages 9118–9147. PMLR.

Lebling, Blank, and Anderson. 1979. Special feature zork: A computerized fantasy simulation game. *Computer*, 12(4):51–59.

Shuang Li, Xavier Puig, Chris Paxton, Yilun Du, Clinton Wang, Linxi Fan, Tao Chen, De-An Huang, Ekin Akyürek, Anima Anandkumar, et al. 2022. Pre-trained language models for interactive decision-making. *Advances in Neural Information Processing Systems*, 35:31199–31212.

Keerthiram Murugesan, Mattia Atzeni, Pavan Kapanipathi, Pushkar Shukla, Sadhana Kumaravel, Gerald Tesauro, Kartik Talamadupula, Mrinmaya Sachan, and Murray Campbell. 2020. Text-based rl agents with commonsense knowledge: New challenges, environments and baselines.

Dhruvesh Patel, Hamid Eghbalzadeh, Nitin Kamra, Michael Louis Iuzzolino, Unnat Jain, and Ruta Desai. 2023. Pretrained language models as visual planners for human assistance. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15302–15314.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in

Python. *Journal of Machine Learning Research*, 12:2825–2830.

Oyvind Tafjord and Peter Clark. 2021. General-purpose question-answering with Macaw. *ArXiv*, abs/2109.02593.

Eleftherios Triantafyllidis, Filippos Christianos, and Zhibin Li. 2023. Intrinsic language-guided exploration for complex long-horizon robotic manipulation tasks.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. 2023. Behavior cloned transformers are neurosymbolic reasoners. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2777–2788, Dubrovnik, Croatia. Association for Computational Linguistics.

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

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.

Xusen Yin and Jonathan May. 2019. Learn how to cook a new recipe in a new house: Using map familiarization, curriculum learning, and bandit feedback to learn families of text-based adventure games.

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 *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 3566–3577, Red Hook, NY, USA. Curran Associates Inc.

Victor Zhong, Dipendra Misra, Xingdi Yuan, and Marc-Alexandre Côté. 2024. Policy improvement using language feedback models.

A Appendix

A.1 Implementation details

A.1.1 GUIDE's architecture

We use a BERT-base model (Devlin et al., 2019) as the GUIDE. We also performed a rudimentary experiment of fine-tuning the Encoder part of the 770M Macaw (Tafjord and Clark, 2021) model (T5 Large model pretrained on Question Answering datasets in Science Domain), but could not achieve the same quality of pruning post training as the smaller BERT-base model. This could be attributed to two reasons:

1. The size of the training dataset may not be enough to train the large number of parameters in the bigger Macaw model (thus leading to underfitting).

2. We used a smaller batch size for training the Macaw model using similar compute as the BERT-base model (16GB GPU memory). As the contrastive loss depends on in-batch examples for negative samples, the smaller batch-size could mean less effective signal to train the model. We would explore a fairer comparison with similar training settings as the BERT model in future work.

The code for this work is available at `https://github.com/hitzkrieg/drrn-scienceworld-clone`.

A.1.2 Training the GUIDE

The supervised contrastive loss framework in (Gao et al., 2021) needs a dataset consisting of example triplets of form (x_i, x_i^+ and x_i^-) where x_i and x_i^+ are semantically related and x_i^- is an example of a hard negative (semantically unrelated to x_i, but more still more similar than any random sample).

For training the Guide, we want to anchor the task descriptions closer in some embedding space to relevant actions and away from irrelevant actions. Thus we prepare a training data $\{(\tau_i, a_i^+, a_i^-)\}_{i=1}^M$, consists of tuples of task descriptions $\tau_i = \tau_{\gamma,v} \in \mathcal{T}$ along with a relevant action $a_i^+ \sim G_{\gamma,v}$ and an irrelevant action $a_i^- \sim \mathcal{N}_\gamma$ (fixed size set of irrelevant actions for every task γ).

Preparing $\mathcal{N}_\gamma$: We simulate gold trajectories from 10 random training variations for each task-type $\gamma \in \Gamma$, and keep taking a union of the valid actions at each time step to create a large union of valid actions for that task-type. $\mathcal{N}_\gamma = \bigcup_{v=1}^{10} \bigcup_t A_{\gamma,v,t}$. Now, this set is used for sampling hard negatives for a given task description. For a batch of size N, the loss is computed as:

$$l(\phi) = -\sum_{i=1}^{N} \log \frac{e^{s(\tau_i, a_i^+)}}{\sum_{j=1}^{N} e^{s(\tau_i, a_j^-)} + e^{s(\tau_i, a_j^+)}}, \quad (1)$$

The final training dataset to train the GUIDE LM on 30 task-types consisting of 3442 training variations had 214535 tuples. The LM was trained with a batch size of 128, on 10 epochs and with a learning rate of 0.00005.

TaskID	Task Name
T0	Changes of State (Boiling)
T1	Changes of State (Any)
T2	Changes of State (Freezing)
T3	Changes of State (Melting)
T4	Measuring Boiling Point (known)
T5	Measuring Boiling Point (unknown)
T6	Use Thermometer
T7	Create a circuit
T8	Renewable vs Non-renewable Energy
T9	Test Conductivity (known)
T10	Test Conductivity (unknown)
T11	Find an animal
T12	Find a living thing
T13	Find a non-living thing
T14	Find a plant
T15	Grow a fruit
T16	Grow a plant
T17	Mixing (generic)
T18	Mixing paints (secondary colours)
T19	Mixing paints (tertiary colours)
T20	Identify longest-lived animal
T21	Identify longest-then-shortest-lived animal
T22	Identify shortest-lived animal
T23	Identify life stages (animal)
T24	Identify life stages (plant)
T25	Inclined Planes (determine angle)
T26	Task 26 Friction (known surfaces)
T27	Friction (unknown surfaces)
T28	Mendelian Genetics (known plants)
T29	Mendelian Genetics (unknown plants)

Table 4: List of Task Names with their task ID's

A.1.3 Training and evaluating the Explorer

We use similar approach as (Wang et al., 2022) to train and evaluate the Explorer. The DRRN architecture is trained with embedding size and hidden size = 128, learning rate = 0.0001, memory size = 100k, priority fraction (for experience replay) =

0.5. The model is trained simultaneously on 8 environment threads at 100k steps per thread. Episodes are reset if they reach 100 steps, or success/failure state.

After every 1000 training steps, evaluation is performed on 10 randomly chosen test variations. The final numbers reported in table 4 are the average score of last 10% test step scores.

A.2 More examples

Table 2 and Table 5 show examples of the out sample usage of the GUIDE.

Table 5: Qualitative analysis of Validation set trajectories for the ScienceWorld Task "Friction Known Surfaces" for variation 0 at step 17. Note: Missed gold actions are in Red, while selected gold actions are in Green.

Relevant Gold Actions
look around, move block to inclined plane with a steel surface, focus on inclined plane with a steel surface, go to hallway, wait1, look at inclined plane with a soapy water surface, move block to inclined plane with a soapy water surface, look at inclined plane with a steel surface
Selected By Guide
focus on inclined plane with a soapy water surface, look at inclined plane with a soapy water surface, move block to inclined plane with a soapy water surface, look at inclined plane with a steel surface, move block to inclined plane with a steel surface, focus on inclined plane with a steel surface, go to hallway, look around, wait1, connect red wire terminal 2 to anode in green light bulb, connect red wire terminal 2 to cathode in green light bulb, connect battery cathode to red wire terminal 1, connect black wire terminal 2 to anode in green light bulb, connect red wire terminal 2 to anode in red light bulb, connect black wire terminal 2 to cathode in green light bulb, connect battery cathode to black wire terminal 1, connect red wire terminal 2 to cathode in red light bulb, connect black wire terminal 2 to anode in red light bulb, connect black wire terminal 2 to cathode in red light bulb, open freezer, wait, pick up red wire, focus on red light bulb, pick up black wire, focus on green light bulb, pick up green light bulb, pick up black wire, focus on green light bulb, pick up green light bulb

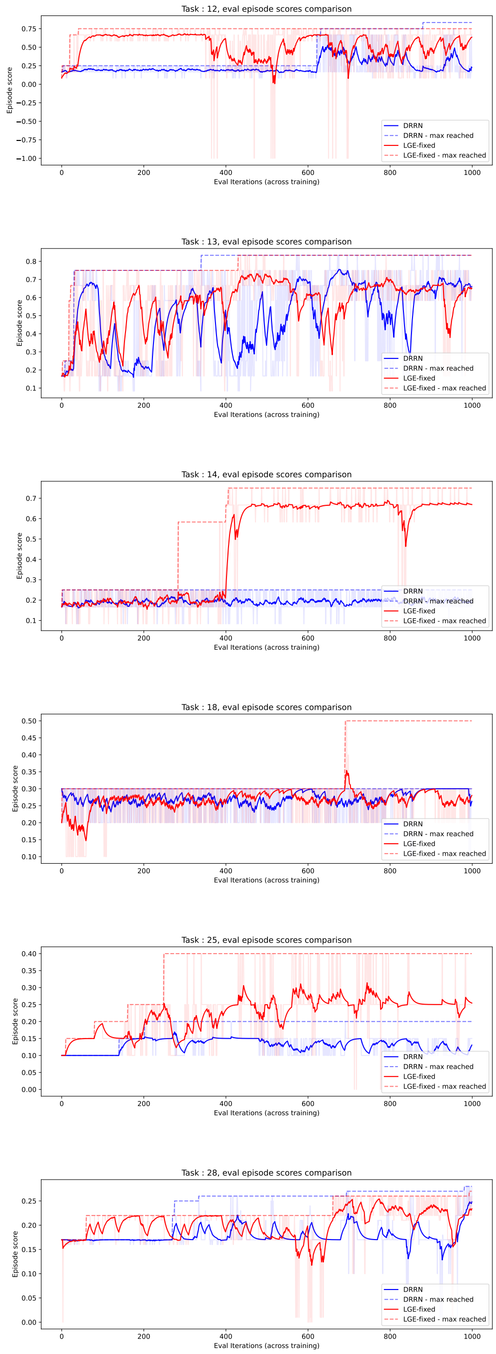

Figure 3: Episode scores on unseen variations validated throughout training. The line plots for DRRN and LGE-fixed are plotted with exponential moving average. We see both the ability to reach higher scores and better sample efficiency for LGE as compared to the DRRN baseline on some tasks.

Algorithm 1 Training Algorithm: LANGUAGE GUIDED EXPLORATION FRAMEWORK with DRRN Explorer

Initialize replay memory D to capacity C
Initialize Explorer's Q-network with random weights θ
Initialize $updateFrequency, totalSteps$
for episode $= 1$ to M **do**
    $env, v, d \leftarrow$ sampleRandomEnv('train', γ)
    Sample initial state s_1 from d_0 and get $A_{\gamma,v,1}$
    for $t = 1$ to N **do**
        $totalSteps \mathrel{+}= 1$
        Identify k most relevant actions using Guide:
        $\hat{A}_{\gamma,v,t} \leftarrow$ Guide.top_k$(A_{\gamma,v,t}, k, d_{T,v})$
        $randomNumber \sim$ Uniform$(0, 1)$
        if $randomNumber > \epsilon$ **then**
            $a_t \sim$ Multinomial(softmax($\{Q(s_t, a|\theta)$ for $a \in \hat{A}_{\gamma,v,t}\}$))
        else
            $a_t \sim$ Multinomial(softmax($\{Q(s_t, a|\theta)$ for $a \in A_{\gamma,v,t}\}$))
        Execute a_t, observe r_{t+1}, s_{t+1}, $A_{\gamma,v,t+1}$
        Store $(s_t, a_t, r_{t+1}, s_{t+1}, A_{\gamma,v,t+1})$ in D
        if $totalSteps \mod updateFrequency = 0$ **then**
            Sample batch from D
            $L_{\text{cumulative}} = 0$
            for each (s, a, r, s', A') in batch **do**
                $\delta = r + \gamma \max_{a' \in A'} Q(s', a'|\theta) - Q(s, a|\theta)$
                Compute Huber loss L:
                $L = \begin{cases} \frac{1}{2}\delta^2 & \text{if } |\delta| < 1 \\ |\delta| - \frac{1}{2} & \text{otherwise} \end{cases}$
                $L_{\text{cumulative}} \mathrel{+}= L$
            Update θ with Adam optimizer:
            $\theta \leftarrow$ AdamOptimizer$(\theta, \nabla_\theta L_{\text{cumulative}})$
        Update state: $s_t \leftarrow s_{t+1}$