Efficient Reinforcement Learning via Large Language Model-based Search

Siddhant Bhambri
School of Computing and AI
Arizona State University
siddhantbhambri@asu.edu

Amrita Bhattacharjee
School of Computing and AI
Arizona State University
abhatt43@asu.edu

Huan Liu
School of Computing and AI
Arizona State University
huanliu@asu.edu

Subbarao Kambhampati
School of Computing and AI
Arizona State University
rao@asu.edu

Abstract

Reinforcement Learning (RL) suffers from sample inefficiency in sparse reward domains, and the problem is pronounced if there are stochastic transitions. To improve the sample efficiency, reward shaping is a well-studied approach to introduce intrinsic rewards that can help the RL agent converge to an optimal policy faster. However, designing a useful reward shaping function specific to each problem is challenging, even for domain experts. They would either have to rely on task-specific domain knowledge or provide an expert demonstration independently for each task. Given, that Large Language Models (LLMs) have rapidly gained prominence across a magnitude of natural language tasks, we aim to answer the following question: *Can we leverage LLMs to construct a reward shaping function that can boost the sample efficiency of an RL agent?* In this work, we aim to leverage off-the-shelf LLMs to generate a guide policy by solving a simpler deterministic abstraction of the original problem that can then be used to construct the reward shaping function for the downstream RL agent. Given the ineffectiveness of directly prompting LLMs, we propose **MEDIC**: a framework that augments LLMs with a **M**odel-based fe**ED**back crit**IC**, which verifies LLM-generated outputs, to generate a possibly sub-optimal but *valid* plan for the abstract problem. Our experiments across domains from the BabyAI environment suite show 1) the effectiveness of augmenting LLMs with MEDIC, 2) a significant improvement in the sample complexity of PPO and A2C-based RL agents when guided by our LLM-generated plan, and finally, 3) pave the direction for further explorations of how these models can be used to augment existing RL pipelines.

1 Introduction

Sample inefficiency of training Reinforcement Learning (RL) agents in sparse reward domains[1] has been a long-standing challenge Ng et al. [1999], Laud and DeJong [2003], Marthi [2007], Grzes and Kudenko [2008], Devlin and Kudenko [2011]. The number of environment interactions take a much severe hit if the domain further consists of stochastic transitions Grześ [2017], Ben-Porat et al. [2024]. In an effort to improve this sample complexity, reward shaping has been proven to be effective, which provides intrinsic rewards as a better training signal over just the sparse extrinsic

[1]We consider the case where the agent gets +1 reward at the goal state, and 0 otherwise.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

(environment) rewards Laud and DeJong [2003], Marthi [2007], Devlin and Kudenko [2011]. In particular, Potential-based Reward Shaping (PBRS) has been a popular approach which provides a reward shaping function as a difference in potential values Ng et al. [1999].

Underlying all reward shaping techniques including PBRS and its variants Devlin et al. [2011], Devlin and Kudenko [2012], Gao and Toni [2015], Eck et al. [2016], there is an inherent assumption made on how this reward shaping function can be constructed. One of the straightforward ways is for a domain expert to hand-engineer the reward shaping function, which can be cognitively demanding and additionally lead to a cognitive bias in the engineered rewards Wu et al. [2024], Lightman et al. [2023]. Yet another popular approach relies on learning the intrinsic rewards via Inverse RL (IRL) Russell [1998], Abbeel and Ng [2004], Russell and Norvig [2016], where the human records an expert demonstration for solving the task. While existing research tackles the problems associated with each of these approaches to some extent, an important issue that has been overlooked is the effort required to design or learn a task-specific reward shaping function. In doing so, the domain expert would either have to rely on domain knowledge pertaining to the specific task at hand, or determine a solution for reaching the goal in each task and provide an expert demonstration for the same.

Lately, Large Language Models (LLMs) have shown remarkable performance spanning a wide variety of natural language-based tasks Kocoń et al. [2023], Gilardi et al. [2023], Zhu et al. [2023] which can be attributed to the enormous and diverse data that they have been trained on. While some tasks have benefited from prompting off-the-shelf LLMs Bubeck et al. [2023], Bhattacharjee et al. [2024], others that are either highly domain-specific or require higher degrees of generalizability, require fine-tuning Li et al. [2024], Yang et al. [2024]. However, several recent studies have shown the performance of prompting LLMs directly to be brittle and unreliable Valmeekam et al. [2023], Stechly et al. [2024], Verma et al. [2024]. Similarly, the latter approach is bottlenecked by the need for sufficient task-specific data and expensive computation required for LLM fine-tuning. Yet, they continue to show some promise when tasked with solving a sufficiently relaxed version of the original problem Nirmal et al. [2024], or assisting in obtaining the final solution Kambhampati et al. [2024]. Keeping this trade-off in mind for our use case, we aim to answer: *Can we leverage LLMs to construct a reward shaping function that can boost the sample efficiency of an RL agent?*

From the limited exploratory works that currently lie at utilizing LLMs for guiding RL Liang et al. [2023], Du et al. [2023], Carta et al. [2023], Jiang et al. [2019], Kwon et al. [2023], Wang et al. [2023], Ma et al. [2023], LLMs have particularly been effective in providing either high-level (symbolic) policy guidance (Jiang et al. [2019], Liang et al. [2023]) or the reward function (Kwon et al. [2023], Ma et al. [2023]), which may only be feasible for tasks where there has been sufficient background knowledge or data that could have possibly been part of the language model's training. Other approaches have fine-tuned LLMs to learn the policy for text-based environments directly Carta et al. [2023], which carry all the limitations of fine-tuning that we listed above.

To this end, we take inspiration from the problem abstraction methods, which have been extensively studied in the planning and RL literature Dietterich et al. [1998], Sutton et al. [1999], Lane and Kaelbling [2002], Kattenbelt et al. [2010], Kulkarni et al. [2016], Gopalan et al. [2017], Jiang et al. [2019], Nashed et al. [2021], and query LLMs to obtain a solution for this sufficiently relaxed problem (in our case, we construct a deterministic abstraction of the original stochastic Markov Decision Process (MDP)). Furthermore, we propose **MEDIC**, a framework that consists of a **M**odel-based fe**ED**back crit**IC** which, at every step of the environment interaction, verifies the LLM-generated output and can generate a possibly sub-optimal but valid plan with soundness guarantees where possible Kambhampati et al. [2024]. By Model-based, we specifically refer to constructing a module based on the environment and task knowledge, which can back-prompt the LLM when it generates an invalid output. Specifically, in our case, an invalid action at any given environment state can be viewed as an invalid LLM output; and by back-prompt, we refer to the feedback that can be given to LLMs to improve the correctness of their outputs on the desired task. Note that while the idea of such critiques or verifiers has been recently seen in other contexts, primarily for checking the syntactic correctness of LLM-generated code Liang et al. [2023], Ma et al. [2023], Wang et al. [2023], we take one step further from syntactic verification and aim to construct and utilize Model-based critiques for improving LLM-generated responses. Finally, we leverage the LLM-generated plan as a basis for constructing the reward shaping function for the downstream RL sparse reward task.

The contributions of this work can be summarized as follows: 1) We propose **MEDIC**, a framework to augment an off-the-shelf LLM with a model-based feedback critic that can generate a valid plan

for a relaxed search problem. 2) Utilizing this LLM-generated plan, we construct a reward shaping function for the downstream stochastic sparse-reward problem. 3) With experiments on the BabyAI suite of environments, we show the utility of our approach for boosting the sample complexity of RL algorithms by demonstrating results with PPO and A2C algorithms.

For the rest of the paper, we begin with situating our work in the domain of LLM-guided RL works and give a brief background of the respective literature Section 2. Next, we formally define our problem statement and discuss the proposed framework and its utilization for reward shaping in detail in Section 3, followed by a thorough empirical analysis in Section 4. Finally, we conclude the work by discussing the possible applications and extensions of this work in Section 5. Appendix has also been attached which includes a detailed description of all our experiments, hyperparameters, additional ablations, results, anonymized code and a short note on the broader impact of this work.

2 Related Work

Sparse Reward RL and LLM-based Guidance: The seminal foundational work by Ng et al. [1999] provided policy invariance guarantees using Potential-based Reward Shaping (PBRS) for boosting the sample efficiency of RL agents, followed by further theoretical investigations by Laud and DeJong [2003], Wiewiora [2003]. Pathak et al. [2017] also showed the advantages of using intrinsic rewards for training RL agents. Reward shaping methods have been studied under several dimensions, including but not limited to automatic reward learning Grzes and Kudenko [2008], Marthi [2007], multi-agent domains Devlin and Kudenko [2011], Sun et al. [2018], meta-learning Zou et al. [2019], etc. Primarily, the sources of obtaining and/or learning intrinsic rewards includes domain experts hand-engineering the rewards Wu et al. [2024], Lightman et al. [2023], via providing feedback Lee et al. [2023], or via providing expert demonstrations Argall et al. [2009]. More recently, Large Language Models have been utilized to give feedback on RL agent's environment interaction Du et al. [2023], Ma et al. [2023], Kwon et al. [2023], Cao et al. [2024] or directly provide the reward function for the RL agent's task. Note, that directly querying LLMs for feedback or reward functions requires prompt engineering efforts, with the reliance on the domain experts to be able to inject cognitive bias such that LLMs are able to respond with the expert's intended results.

LLMs for Planning and Search: There is a research divide in the current literature regarding the planning, reasoning and verification abilities of Large Language Models. While popular works claiming LLM reasoning abilities have proposed several prompting methods Wei et al. [2022], Yao et al. [2023], Long [2023], Yao et al. [2024], Besta et al. [2024], there have been independent investigations refuting such claims using LLMs for solving deterministic planning and classical reasoning problems Valmeekam et al. [2023], Stechly et al. [2024], Verma et al. [2024]. While LLMs are themselves not reliable for providing accurate feedback Stechly et al. [2024], other than in natural language tasks Yao et al. [2023], recent works have augmented LLMs with task-specific verifiers (for example, a Python compiler that can check LLM-generated code for syntactic correctness) that can evaluate the validity (not necessarily correctness) of the LLM-generated output and provide feedback to the LLM accordingly Kambhampati et al. [2024], Ma et al. [2023], Wang et al. [2023], Liang et al. [2023]. These augmented LLM frameworks have been shown to be useful for improving the overall task performance when prompting LLMs. Our work is the first to showcase how such model-based verifiers can be constructed that can guide LLMs to solve relaxed deterministic problems. We further utilize these plans to construct the reward shaping function for the downstream RL agent.

3 Augmenting LLMs with MEDIC

In this section, we introduce **MEDIC**, our framework where an LLM augmented with a model-based feedback critic is used to generate a valid solution for a relaxed deterministic search problem. This plan is further used to construct a reward shaping function, resulting in a sample efficiency boost for the RL agent that learns a policy on the original stochastic, sparse reward problem. We first formalize our problem statement in Section 3.1. Next, we detail how we use an LLM augmented with MEDIC to interact with the BabyAI environment in Section 3.2. Finally, we present details on the reward shaping step that aids RL training, along with the training algorithm in Section 3.3.

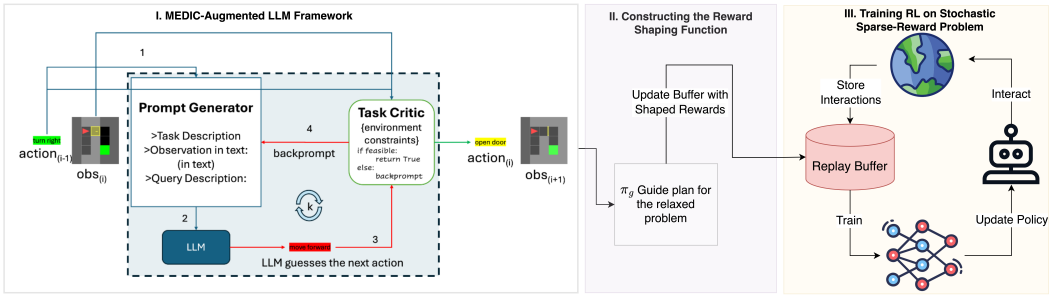


Figure 1: Augmenting LLMs with MEDIC: (I) We use our framework to generate a valid (guide) plan for the relaxed search problem. (II) We construct the reward shaping function using the guide plan to add intrinsic rewards by updating the RL agent's replay buffer. (III) The RL agent learns an optimal policy with the help of these intrinsic rewards on the original stochastic sparse-reward problem.

3.1 Problem Statement

We consider a finite horizon Markov Decision Process (MDP) $\mathcal{M}$ defined by the tuple ($\mathcal{S}$, $\mathcal{A}$, $\mathcal{P}$, $\mathcal{R}$, γ). Here, $\mathcal{S}$ represents the set of all possible states, $\mathcal{A}$ represents the set of all possible actions, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the stochastic state transition function where $\mathcal{P}(s'|s, a)$ is the transition probability for $s, s' \in \mathcal{S}$ and $a \in \mathcal{A}$, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function, and γ is the discount factor. In our case, we consider an MDP $\mathcal{M}$ with sparse rewards, i.e., $\mathcal{R} = 1$ for $g \in \mathcal{S}$ and $\mathcal{R} = 0$ otherwise, where g is the goal or termination state. The goal of the agent is to learn a parameterized policy $\pi_\theta(a|s)$ which maximizes the discounted cumulative reward for the trajectory τ, $\mathcal{J}(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} \gamma^t r_t\right]$. For the LLM to interact with the environment, we consider the modified MDP $\mathcal{M}'$ which is a deterministic abstraction of the original MDP $\mathcal{M}$. Hence, $\mathcal{M}'$ can be defined using the tuple ($\mathcal{S}$, $\mathcal{A}$, $\mathcal{T}$, $\mathcal{R}$, γ), where $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the deterministic transition function. We aim to obtain a guide policy π_g in $\mathcal{M}'$ using this LLM-environment interaction, that can further be used for the original downstream RL problem to learn π_θ.

3.2 Generating Guide Policy using LLM + MEDIC

Model-based Feedback Critic Construction: We will use the BabyAI *DoorKey* environment, as shown in Figure 1, as the example to discuss the details of how we construct the critique. Note, that this discussion also applies to any other environment that shares the same assumptions that we make with respect to domain knowledge. It is important to note that we eventually require a set of actions from this critique, that are feasible at any given state of the environment that the agent can be in. We will refer to this set of valid actions as $\{\mathcal{A}_v|s\}$ and set of invalid actions as $\{\mathcal{A}_{v^-}|s\}$ for $s \in \mathcal{S}$. Hence, for the BabyAI *DoorKey* environment, we begin with computing the agent's position (`x,y`) at the given state (s_i). Since we assume access to domain knowledge, particularly, the environment layout and the action space, we can verify each action that the agent can and can not take for a given state, i.e., either $a \in \{\mathcal{A}_v|s_i\}$ or $a \in \{\mathcal{A}_{v^-}|s_i\}$. For example, in the state (s_i) shown in Figure 1, the agent has already picked up the key but can only move forward if the door is open. Hence, our Model-based critique, given the agent's current position (`x,y`) and the set of valid actions $\{\mathcal{A}_v\}$ taken until step i, computes the set of actions that are feasible in the current state, i.e., `turn left`, `turn right`, and `toggle` (open door). Next, given the LLM's guessed action, i.e., `move forward`, the critique finds the action to be infeasible and generates a back-prompt that is appended to the LLM's original prompt. We further discuss prompt construction and design details.

Prompt Construction: At any given step i in the LLM's interaction with the environment, we construct the LLM prompt with three components. The first is the *Task Description* that defines the task that the LLM has to achieve. For example, as shown in Figure 1 for the BabyAI *DoorKey* environment, we specify that the environment is a 3x3 maze that consists of objects such as a key, a door, walls, and a goal location. We further include the goal of the agent that is obtained from the environment (referred to as the mission space in the environment specification), i.e., *"use the key to open the door and then get to the goal"*, followed by the environment's action space ($\mathcal{A}$).

While the *Task Description* is the same for the entire episode of the LLM's environment interaction, we construct the *Observation Description* independently at every step. This component describes the current view of the environment as shown in Figure 1. We admit that representing spatial relationships in text as an input to an LLM can be a challenging task. Inspired by the current works that have attempted to prompt LLMs with spatial descriptions Patel and Pavlick [2021], for the scope of this work, we represent the 3x3 grid as three rows of objects that are currently observed in the environment state (s_i). Naturally, there can exist prompt engineering techniques to better represent such spatial relationships, and can be a useful direction for future investigations. Finally, the third and the last component of this prompt is the *Query Description* which poses the question to the LLM to guess the next action that it should take in the environment. We refer to this as the *step-prompt* for our discussion. Once the LLM returns a valid action as a guess to the *step-prompt*, i.e. $\pi_{LLM}(s_i) = a$ for $a \in \{\mathcal{A}_v|s_i\}$, we execute the action in the environment and continue the interaction.

Consider the case where the LLM has guessed an invalid action, i.e. $\pi_{LLM}(s_i) = a$ for $a \in \{\mathcal{A}_{v^-}|s_i\}$. In this case, we do not execute the action in the environment, but rather, construct the *back-prompt* by appending the feedback given by our Model-based critique to the current prompt. To the existing *step-prompt*, we add this critique feedback that lists all the invalid actions ($\{\mathcal{A}_{v^-}|s_i\}$) guessed by the LLM for the current state s_i and prompt the LLM again to choose a different action that is not in $\{\mathcal{A}_{v^-}|s_i\}$. Once the LLM guesses a valid action, we break out of this back-prompting loop and continue our environment interaction as mentioned above (see Appendix D for the complete prompt).

3.3 Potential-based Reward Shaping using LLM policy

Potential-based Reward Shaping (PBRS) Ng et al. [1999] allows for injecting intrinsic rewards for training an RL agent on a sparse reward problem while guaranteeing the so-called policy invariance property. Formally, we adopt the definition which utilizes a shaping reward function $\mathcal{F}$ such that for any state s and action a, our updated reward function can be defined as $\mathcal{R}'(s,a) = \mathcal{R}(s,a) + \mathcal{F}(s,a)$.

Once we have obtained π_{LLM} using our Model-based critique-augmented LLM framework as explained in Section 3.2, we store this goal-reaching trajectory $\tau_g \sim \pi_{LLM}$ and utilize it to construct our reward shaping function $\mathcal{F}$. Until we reach the goal state s_g, we sample $a \sim \pi_{LLM}(s)$ and assign potentials to each (s,a) pair. In the exploration phase, our RL agent will be storing the environment interactions, i.e., (s,a,s',r) tuples, in a dataset buffer $\mathcal{D}$. Given $\mathcal{F}$ constructed using $\tau_g \sim \pi_{LLM}$, we update the buffer $\mathcal{D}$ with the shaped rewards such that $(s,a,s',r) \rightarrow (s,a,s',r')$ where r' is the shaped reward. Our RL agent then continues the learning over this updated dataset buffer $\mathcal{D}'$. We present the complete pipeline in Figure 1 (see Algorithm 1 in Appendix C).

4 Experiments & Results

Recall that we first aim to investigate the advantages of augmenting off-the-shelf LLMs with MEDIC in solving relaxed deterministic search problems. Next, given the reward shaping function constructed using this LLM-generated plan, we want to measure the boost in sample efficiency for training the RL agents on the original stochastic sparse-reward problem. To do so, we utilize the BabyAI suite of environments, namely - *DoorKey* which is a complex environment that requires an understanding of the sub-goals (using key to open door and reaching the goal) that need to be achieved; *Empty-Random* where there are no obstacles but the initial position of the agent is randomized for each episode; and finally, *LavaGap* in which the agent has to reach the goal location while avoiding the adversarial objects (the lava tile) present in the environment. We aim to answer the following: **RQ1:** How does our MEDIC framework perform in terms of plan length and total rewards? **RQ2:** How effective is the reward shaping with the use of LLM-generated plans for boosting the RL sample efficiency?

4.1 Testing our Model-based critique-augmented LLM framework (RQ1)

In order to study RQ1, we run all our experiments using the OpenAI API for `gpt-3.5-turbo`. We highlight the experimental setup, baselines and evaluation metrics in the following paragraphs.

Experimental Setup: To test the robustness of our evaluations for RQ1, we run all our LLM-based experiments on three different layouts for the *DoorKey-5x5* environment (varying the object and goal placements), three different layouts for the *Empty-Random-5x5* environment (varying the

Table 1: **RQ1 Results:** We train the *RL* agents for 1e5 steps, across 3 seeds and average the metrics by evaluating each agent over 100 episodes. For *Direct LLM prompting* and *LLM+MEDIC (Ours)* results, we prompt `gpt-3.5-turbo` three times with temperature=0 for each environment, and show the average across each metric.

Environment	Performance Metric	RL	Direct LLM prompting	LLM + MEDIC *Ours*	Optimal Search
DoorKey	*Avg Task Success Rate*	100%	0%	100%	100%
	Avg Success Plan Length	18.43 ± 0.601	–	15.2 ± 6.57	11
	Avg Success Rewards	0.83 ± 0.005	–	0.945 ± 0.02	0.96
Empty-Random	*Avg Task Success Rate*	100%	0%	100%	100%
	Avg Success Plan Length	5.47 ± 0.079	–	27 ± 1	4
	Avg Success Rewards	0.95 ± 0.0	–	0.757 ± 0.009	0.96
LavaGap	*Avg Task Success Rate*	100%	0%	100%	100%
	Avg Success Plan Length	9.45 ± 0.354	–	16 ± 8	6
	Avg Success Rewards	0.91 ± 0.005	–	0.8559 ± 0.07	0.95

agent's initial state independent of any environment objects being present), and one layout each of *LavaGap-5x5* and *DoorKey-6x6* environments (testing our framework's performance by varying environment adversariality and scalability). For each environment layout, we query our framework by bounding the maximum number of steps it can take to reach the goal to 30, and the maximum number of back-prompts allowed at any step to 10. Since *temperature* 0 does not guarantee complete determinism in LLM generation, we query each LLM three times across every environment layout. We include the ablations on the plan length and back-prompt attempts in Section 4.3, and provide other experimental details including prompts for these experiments in Appendix C and D.

Baselines & Evaluation Metrics: One may argue that while it may be computationally expensive given the same sparse reward problem in the deterministic setting, we can always train an RL agent to generate our guide plan. Note that this is only valid where we can assume that the cost of interaction with the environment is nearly negligible. As an ablation and for comparative purposes, we also evaluate directly prompting LLMs without augmenting MEDIC. Also, for the current scope of this work, since we are interested in deterministic search problems, it is easy to compute the optimal plans, for example, by using A* search. Hence, to thoroughly investigate the effectiveness of our framework, we compute the optimal plans as an upper bound, and compare our augmented LLM-generated plans in terms of 1) task success rate, 2) average length of a successful plan, and 3) average total rewards. Note, that while we only show results for deterministic search problems, it is worth investigating the applicability of our approach in domains where computing the optimal plan can be very costly Bhambri et al. [2022], or infeasible Silver et al. [2018].

Results: From the results in Table 1, we note that while both RL and our MEDIC-augmented LLM framework get 100% success rate, direct LLM prompting can not generate a valid plan in all the cases across the three environment settings. Also, the number of environment interactions required for our approach is a maximum of 30 *step-prompts* and 10 *back-prompts* for each step, compared to 1e5 steps required for training the RL agents. Moreover, our approach performs the closest to the optimal search results in terms of average success plan length and average success rewards. Once again, we emphasize that these results hold promise for further exploring the applications of using the critic-augmented LLM framework where obtaining search heuristics and optimal plans can be costly or computationally infeasible, respectively.

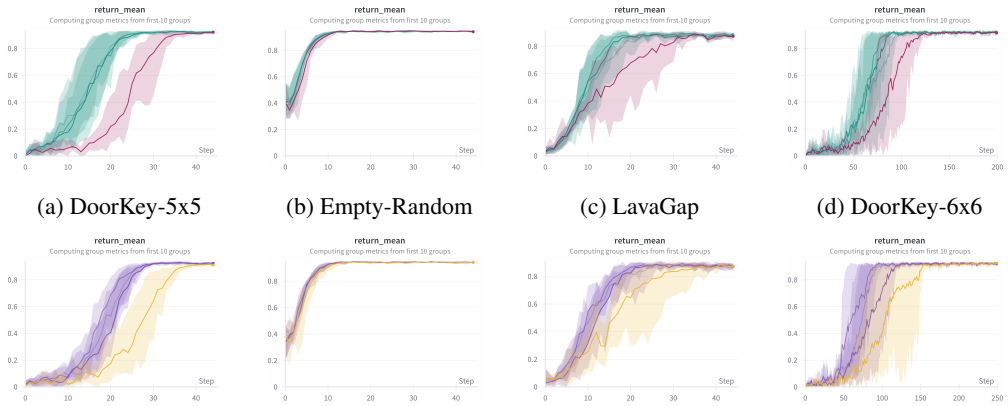
4.2 Evaluating the sample efficiency boost in RL training (RQ2)

To study RQ2, we train PPO and A2C algorithms on each environment layout that we mentioned in Section 4.1. We provide all training and hyperparameters details in Appendix C.

Experimental Setup: Recall, from Section 4.1, that we generate three guide plans for each environment layout. We train the PPO and A2C agents on our original stochastic sparse-reward problem, with the reward shaping function $\mathcal{F}(s, a)$ constructed using each of these guide plans. Due to space constraints, we show the results for one layout for each environment. For each experiment setting, we compare and plot our training results in terms of the total environment returns in Figure 2

and Figure 3. Results on other environment layouts and plots on the number of steps to reach the goal are shown in Appendix C.

Baselines & Evaluation Metrics: We adopt the RL baselines for both PPO and A2C algorithms from Chevalier-Boisvert et al. [2018]. Along with the base RL setting (labeled as *vanilla* in Figure 2 and Figure 3), we also consider an extended *vanilla-text* setting which also encodes the mission space (for example, for *DoorKey* environment, the mission space is defined as: "use the key to open the door and then get to the goal") as an input feature along with the environment state, and has been shown to be an improvement over the base RL for certain environment settings.



(a) DoorKey-5x5 (b) Empty-Random (c) LavaGap (d) DoorKey-6x6

(e) DoorKey-5x5 (f) Empty-Random (g) LavaGap (h) DoorKey-6x6

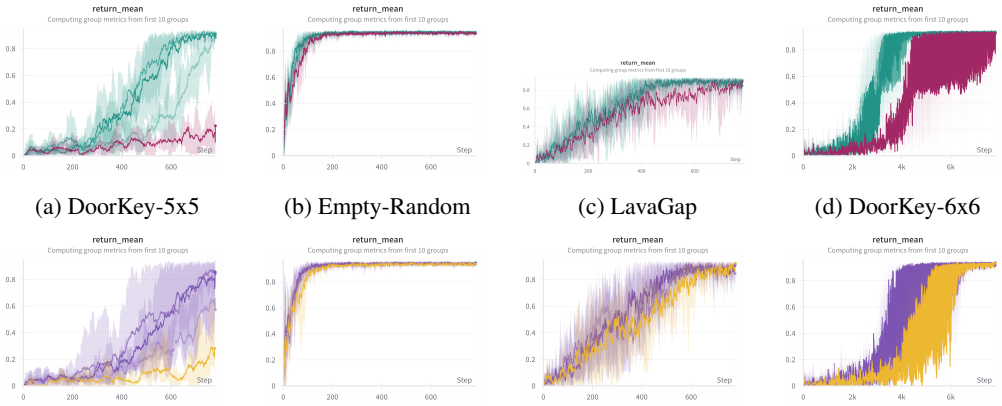
Figure 2: **RQ2 Results:** *Top-* Learning curves across five runs comparing vanilla PPO training against vanilla PPO with reward shaping using our augmented LLM-generated plan 1, plan 2, and, plan 3, as measured on the episodic returns. *Bottom-* Learning curves across five runs comparing vanilla-text PPO training against vanilla-text PPO with reward shaping using our augmented LLM-generated plan 1, plan 2, and, plan 3, as measured on the episodic returns. The solid lines and shaded regions represent the mean and min/max range, respectively.

Results: From results shown in Figure 2 and Figure 3, we note the most significant boost in sample efficiency due to the reward shaping using our augmented LLM-generated plan in the BabyAI *DoorKey-5x5* environment, followed by *LavaGap*, and results in *Empty-Random-5x5* environment are similar to the vanilla RL and vanilla RL w/ text baselines. We believe that the commonsensical domain knowledge available in the *DoorKey-5x5* environment (i.e., an agent first needs to pick the key, open the door, and then reach the goal location) is possibly exploited and utilized by the LLM to successfully solve the deterministic abstraction of the problem. Constructing the reward shaping function with this knowledge further helps the RL agent learn the optimal policy faster. For the *LavaGap* environment, the difference between reward-shaped policy training and the baselines is relatively smaller than that in *DoorKey-5x5*. One possible reason here could be that while the action space for this environment is smaller (three actions only), using the LLM-generated plan for reward shaping allows the RL agent to learn to avoid the lava tiles faster and reach the goal location. It is further useful to see that reward shaping using our approach performs at par with the baselines for the *Empty-Random-5x5* environment, despite the complexity of varying agent initializations. Along with results on other layouts for *DoorKey-5x5* and *Empty-Random-5x5*, we further show the results for the number of steps to reach the goal in Appendix C.

4.3 Ablations and Additional Experiments

In relation to the experiments for RQ1 (Section 4.1) and RQ2 (Section 4.2), we perform the following additional experiments and briefly discuss our findings for each:

Number of *step-prompts* (RQ1): By varying the number of allowed *step-prompts* across 10, 20, 30, & 50 for two environment layouts of the *DoorkKey-5x5* environment and five trials, we find that our framework is successfully able to find a valid plan solving the task each time when given 30 and 50



(a) DoorKey-5x5 (b) Empty-Random (c) LavaGap (d) DoorKey-6x6

(e) DoorKey-5x5 (f) Empty-Random (g) LavaGap (h) DoorKey-6x6

Figure 3: **RQ2 Results:** *Top-* Learning curves across five runs comparing vanilla A2C training against vanilla A2C with reward shaping using our augmented LLM-generated plan 1, plan 2, and, plan 3, as measured on the episodic returns. *Bottom-* Learning curves across five runs comparing vanilla-text A2C training against vanilla-text A2C with reward shaping using our augmented LLM-generated plan 1, plan 2, and, plan 3, as measured on the episodic returns. The solid lines and shaded regions represent the mean and min/max range, respectively.

step-prompts along with 10 back-prompts. We observe x and y failures for 10 and 20 step-prompt experiments, respectively. This indicates, that given enough trials which may be dependent on the complexity and scale of the gridworld, our framework is successfully able to find a valid plan.

Number of *back-prompts* (RQ1): By varying the number of allowed *back-prompts* across 3, 5, & 10 for two environment layouts of the *DoorkKey-5x5* environment and five trials, we find that our framework is successfully able to find a valid plan solving the task each time when given 10 back-prompts, along with 30 step-prompts. On average, we observe 10% and 30% failures for 3 and 5 back-prompt experiments, respectively. This indicates, that given enough trials which may be dependent on the size of the state and action space, our framework is able to find a valid plan.

Environment Scalability (RQ1): By running our experiments on a larger gridworld, i.e., *DoorKey-6x6*, we are able to achieve the reward shaping plan using our approach by querying the LLM with 100 *step-prompts* limit and 20 *back-prompts* limit. Furthermore, from Figures 2 and 3, we observe a sample efficiency boost close to 29% and 52% over PPO and A2C baselines, respectively.

Reward Shaping without MEDIC (RQ2): We also construct a reward shaping function by using the partially correct plan generated by directly prompting the LLM. For one layout of *DoorKey-5x5* environment, we get a plan with only one action correct and assign a +0.1 reward for that (state, action) pair. Expectedly, we observe no improvement in sample efficiency of the vanilla RL setting, thereby, reaffirming the importance of reward shaping using only a correct plan that can be obtained with our MEDIC-augmented LLM framework. Note that future work can relax this assumption by better utilizing the partially correct LLM-generated plan for constructing the reward shaping function. We show the plots for this ablation in Appendix C.5.

5 Conclusion & Future Work

Task-specific reward shaping to construct intrinsic rewards useful for training Reinforcement Learning agents requires expert domain knowledge and engineering efforts. Obtaining reward shaping functions from domain experts can also lead to the injection of cognitive bias, which can influence the downstream RL training. Lately, Large Language Models have shown remarkable success in a variety of natural language tasks, while also encountering limitations when it comes to prompting them for planning and reasoning domains. Keeping these limitations in mind, we leveraged LLMs in this work to guide RL training in sparse reward tasks across a variety of environments from the BabyAI suite. Instead of relying on prompting off-the-shelf LLMs or fine-tuning them, we proposed an approach to build Model-based critiques that can be augmented with LLMs to verify their guessed outputs

and provide feedback in an automated fashion. Using these generated plans on the deterministic abstraction of the original problem, we constructed a Potential-based Reward Shaping function. We noticed a boost in the sample efficiency of downstream RL training on the original sparse reward stochastic problem using PPO and A2C algorithms. From these results, we believe that our work can pave the direction for further exploring the meaningful advantages Large Language Models can offer to Reinforcement Learning.

Acknowledgement

This research is supported in part by ONR grant N0001423-1-2409, and gifts from Qualcomm, J.P. Morgan and Amazon.

References

Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1, 2004.

Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

Omer Ben-Porat, Yishay Mansour, Michal Moshkovitz, and Boaz Taitler. Principal-agent reward shaping in mdps. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 9502–9510, 2024.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17682–17690, 2024.

Siddhant Bhambri, Amrita Bhattacharjee, and Dimitri Bertsekas. Reinforcement learning methods for wordle: A pomdp/adaptive control approach. *arXiv preprint arXiv:2211.10298*, 2022.

Amrita Bhattacharjee, Raha Moraffah, Joshua Garland, and Huan Liu. Towards llm-guided causal explainability for black-box text classifiers. In *AAAI 2024 Workshop on Responsible Language Models, Vancouver, BC, Canada*, 2024.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

Meng Cao, Lei Shu, Lei Yu, Yun Zhu, Nevan Wichers, Yinxiao Liu, and Lei Meng. Beyond sparse rewards: Enhancing reinforcement learning with language model critique in text generation, 2024.

Thomas Carta, Clément Romac, Thomas Wolf, Sylvain Lamprier, Olivier Sigaud, and Pierre-Yves Oudeyer. Grounding large language models in interactive environments with online reinforcement learning. In *International Conference on Machine Learning*, pages 3676–3713. PMLR, 2023.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. Babyai: A platform to study the sample efficiency of grounded language learning. *arXiv preprint arXiv:1810.08272*, 2018.

Sam Devlin and Daniel Kudenko. Theoretical considerations of potential-based reward shaping for multi-agent systems. In *Tenth International Conference on Autonomous Agents and Multi-Agent Systems*, pages 225–232. ACM, 2011.

Sam Devlin, Daniel Kudenko, and Marek Grześ. An empirical study of potential-based reward shaping and advice in complex, multi-agent systems. *Advances in Complex Systems*, 14(02): 251–278, 2011.

Sam Michael Devlin and Daniel Kudenko. Dynamic potential-based reward shaping. In *11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, pages 433–440. IFAAMAS, 2012.

Thomas G Dietterich et al. The maxq method for hierarchical reinforcement learning. In *ICML*, volume 98, pages 118–126, 1998.

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

Adam Eck, Leen-Kiat Soh, Sam Devlin, and Daniel Kudenko. Potential-based reward shaping for finite horizon online pomdp planning. *Autonomous Agents and Multi-Agent Systems*, 30:403–445, 2016.

Yang Gao and Francesca Toni. Potential based reward shaping for hierarchical reinforcement learning. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

Fabrizio Gilardi, Meysam Alizadeh, and Maël Kubli. Chatgpt outperforms crowd workers for text-annotation tasks. *Proceedings of the National Academy of Sciences*, 120(30):e2305016120, 2023.

Nakul Gopalan, Michael Littman, James MacGlashan, Shawn Squire, Stefanie Tellex, John Winder, Lawson Wong, et al. Planning with abstract markov decision processes. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 27, pages 480–488, 2017.

Marek Grześ. Reward shaping in episodic reinforcement learning. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 565–573, 2017.

Marek Grzes and Daniel Kudenko. Learning potential for reward shaping in reinforcement learning with tile coding. In *Proceedings AAMAS 2008 Workshop on Adaptive and Learning Agents and Multi-Agent Systems (ALAMAS-ALAg 2008)*, pages 17–23, 2008.

Yiding Jiang, Shixiang Shane Gu, Kevin P Murphy, and Chelsea Finn. Language as an abstraction for hierarchical deep reinforcement learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Kaya Stechly, Mudit Verma, Siddhant Bhambri, Lucas Saldyt, and Anil Murthy. Llms can't plan, but can help planning in llm-modulo frameworks. *arXiv preprint arXiv:2402.01817*, 2024.

Mark Kattenbelt, Marta Kwiatkowska, Gethin Norman, and David Parker. A game-based abstraction-refinement framework for markov decision processes. *Formal Methods in System Design*, 36: 246–280, 2010.

Jan Kocoń, Igor Cichecki, Oliwier Kaszyca, Mateusz Kochanek, Dominika Szydło, Joanna Baran, Julita Bielaniewicz, Marcin Gruza, Arkadiusz Janz, Kamil Kanclerz, et al. Chatgpt: Jack of all trades, master of none. *Information Fusion*, 99:101861, 2023.

Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

Minae Kwon, Sang Michael Xie, Kalesha Bullard, and Dorsa Sadigh. Reward design with language models. *arXiv preprint arXiv:2303.00001*, 2023.

Terran Lane and Leslie Pack Kaelbling. Nearly deterministic abstractions of markov decision processes. In *AAAI/IAAI*, pages 260–266, 2002.

Adam Laud and Gerald DeJong. The influence of reward on the speed of reinforcement learning: An analysis of shaping. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 440–447, 2003.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Lu, Thomas Mesnard, Colton Bishop, Victor Carbune, and Abhinav Rastogi. Rlaif: Scaling reinforcement learning from human feedback with ai feedback. *arXiv preprint arXiv:2309.00267*, 2023.

Rumeng Li, Xun Wang, and Hong Yu. Llamacare: An instruction fine-tuned large language model for clinical nlp. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 10632–10641, 2024.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9493–9500. IEEE, 2023.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Jieyi Long. Large language model guided tree-of-thought. *arXiv preprint arXiv:2305.08291*, 2023.

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv:2310.12931*, 2023.

Bhaskara Marthi. Automatic shaping and decomposition of reward functions. In *Proceedings of the 24th International Conference on Machine learning*, pages 601–608, 2007.

Samer B Nashed, Justin Svegliato, Matteo Brucato, Connor Basich, Rod Grupen, and Shlomo Zilberstein. Solving markov decision processes with partial state abstractions. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 813–819. IEEE, 2021.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pages 278–287, 1999.

Ayushi Nirmal, Amrita Bhattacharjee, Paras Sheth, and Huan Liu. Towards interpretable hate speech detection using large language model-extracted rationales. *arXiv preprint arXiv:2403.12403*, 2024.

Roma Patel and Ellie Pavlick. Mapping language models to grounded conceptual spaces. In *International conference on learning representations*, 2021.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pages 2778–2787. PMLR, 2017.

Stuart Russell. Learning agents for uncertain environments. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 101–103, 1998.

Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Pearson, 2016.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419): 1140–1144, 2018.

Kaya Stechly, Karthik Valmeekam, and Subbarao Kambhampati. On the self-verification limitations of large language models on reasoning and planning tasks. *arXiv preprint arXiv:2402.08115*, 2024.

Fan-Yun Sun, Yen-Yu Chang, Yueh-Hua Wu, and Shou-De Lin. Designing non-greedy reinforcement learning agents with diminishing reward shaping. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pages 297–302, 2018.

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. On the planning abilities of large language models-a critical investigation. *Advances in Neural Information Processing Systems*, 36:75993–76005, 2023.

Mudit Verma, Siddhant Bhambri, and Subbarao Kambhampati. Theory of mind abilities of large language models in human-robot interaction: An illusion? In *Companion of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*, pages 36–45, 2024.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Eric Wiewiora. Potential-based shaping and q-value initialization are equivalent. *Journal of Artificial Intelligence Research*, 19:205–208, 2003.

Zeqiu Wu, Yushi Hu, Weijia Shi, Nouha Dziri, Alane Suhr, Prithviraj Ammanabrolu, Noah A Smith, Mari Ostendorf, and Hannaneh Hajishirzi. Fine-grained human feedback gives better rewards for language model training. *Advances in Neural Information Processing Systems*, 36, 2024.

Haoran Yang, Yumeng Zhang, Jiaqi Xu, Hongyuan Lu, Pheng Ann Heng, and Wai Lam. Unveiling the generalization power of fine-tuned large language models. *arXiv preprint arXiv:2403.09162*, 2024.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36, 2024.

Yao Yao, Zuchao Li, and Hai Zhao. Beyond chain-of-thought, effective graph-of-thought reasoning in large language models. *arXiv preprint arXiv:2305.16582*, 2023.

Yiming Zhu, Peixian Zhang, Ehsan-Ul Haq, Pan Hui, and Gareth Tyson. Can chatgpt reproduce human-generated labels? a study of social computing tasks. *arXiv preprint arXiv:2304.10145*, 2023.

Haosheng Zou, Tongzheng Ren, Dong Yan, Hang Su, and Jun Zhu. Reward shaping via meta-learning. *arXiv preprint arXiv:1901.09330*, 2019.

A Broader Impact

Designing a reward function is not a trivial challenge for many real-world problems where Reinforcement Learning can be useful. Moreover, in domains where interacting with the environment is limited or costly, sample inefficiency of RL algorithms can become a bottleneck for learning a policy. While domain experts can hand-engineer intrinsic rewards to aid the RL agent training, it is unreasonable to expect them to design useful reward shaping functions for each independent task. As a single step towards relaxing this expectation, we aim to leverage the universality of using Large Language Models (which can loosely be referred to "*Jack of all trades, master of none*") to provide useful guidance for training the RL agent. Hence, one of the direct impacts of our work is tapping into the potential of LLMs for obtaining useful guidance across multiple tasks thereby reducing the cognitive load on domain experts. Moreover, we hope that our work opens up future possibilities of exploring more domains where LLMs can assist in aiding the downstream learning process.

B Environments

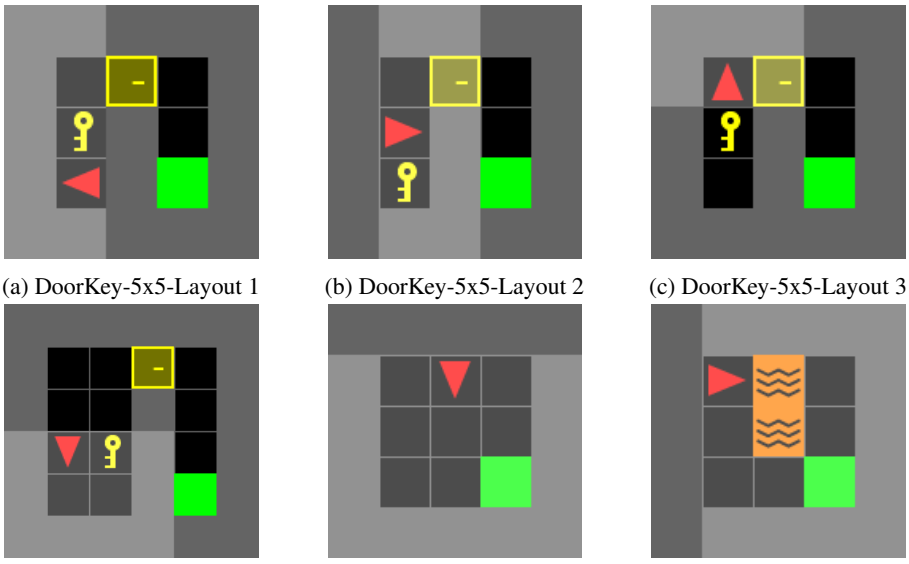
B.1 BabyAI

In the BabyAI suite of environments, the agent has to reach the goal location in a fixed number of steps (environment max steps is set to 100). This platform relies on a gridworld environment (MiniGrid) to generate a set of complex environments, and includes challenging domains to test sample efficiency of training RL agents. Each gridworld environment is populated with the agent and objects such as doors, keys, and lava, which are placed in varying sized grids. Each grid is procedurally generated, i.e., the object placements can vary at the start of each episode. However, using a fixed environment seed, the layout stays the same for each episode run. The doors in the environment can also be locked, and can be opened by first picking up the key first. Objects such as lava can not be crossed by the agent, and lead to episode termination if the agent steps on one of these.

Observation Space: By default, the environment offers an observation space of shape (7,7,3) which is a partial view of the complete environment grid, and has been shown to accelerate the RL agent's training Chevalier-Boisvert et al. [2018]. This observation space consists of a symbolic mapping for 3 matrices each of size 7x7. The first contains the object-id, the second contains the property (color) of the object, and the third consists of the state of the object (e.g., if the door is open or locked).

Action Space: The agent's action space consists of : turn left, turn right, move forward, pick up an object, drop an object, toggle/activate an object, and done. For each environment, some of these actions are unused, and we refer readers to the specific environment's documentation for the respective details.

Extrinsic/Environment Rewards: If the agent reaches the goal location in N steps, the total reward is calculated as R = 1 - 0.9*(N/H) where H is the maximum number of allowed steps (i.e., 100). Otherwise, the agent get a reward of 0 for that episode.



(a) DoorKey-5x5-Layout 1 (b) DoorKey-5x5-Layout 2 (c) DoorKey-5x5-Layout 3

(d) DoorKey-6x6 (e) Empty (f) LavaGap

Figure 4: Environment Layouts from the BabyAI environment suite used for experiments.

B.1.1 DoorKey

Description: The agent has to pick up the key to unlock the door and then reach to the goal location shown by the green square.

Mission Space: "use the key to open the door and then get to the goal"

Action Space: turn left, turn right, move forward, pick up an object, toggle/activate an object

B.1.2 Empty-Random

Description: The agent has to reach the goal location shown by the green square in a completely empty room. The agent's starting position can be varied by varying the environment seed.

Mission Space: "get to the green goal square"

Action Space: turn left, turn right, move forward

B.1.3 LavaGap

Description: The agent has to reach to the goal location shown by the green square by avoiding all lava tiles, where the episode will terminate immediately.

Mission Space: "avoid the lava and get to the green goal square"

Action Space: turn left, turn right, move forward

C Algorithm, Experiment Details & Additional Results

All the experiments were run on an Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz, with GeForce GTX 1080 GPU.

C.1 Algorithm

We present the algorithm for our framework below:

Algorithm 1 Training an RL policy π_θ using π_{LLM}

Require: Input $\pi_{\mathbf{LLM}}$
Ensure: Output π_θ
Initialize parameters θ, dataset $\mathcal{D}$
// EXPLORATION PHASE
for each iteration **do**
    Store transitions $\mathcal{D} \leftarrow (s, a, s', r) \sim \pi_\theta$
    Relabel dataset with shaped rewards $\mathcal{D}' \leftarrow \mathcal{D}, \pi_{LLM}$
end for
// REINFORCEMENT LEARNING PHASE
for each gradient step **do**
    Sample transitions $\{\tau_j\}_{j=1}^{\mathcal{D}'} \sim \mathcal{D}'$
    train π_θ using $\{\tau_j\}$
end for
return π_θ

C.2 RQ1: RL Experiments

For learning an RL policy on the deterministic MDP, we train PPO to learn an optimal policy and construct the reward shaping function by storing the (state, action) pairs generated by this policy during evaluation. The hyperparameters for this training can be found in Table 2.

C.3 RQ1: Visualizing Guide Plans

Here, we provide visualization examples of the state sequence that we obtain by executing the actions generated by our MEDIC-augmented LLM framework.

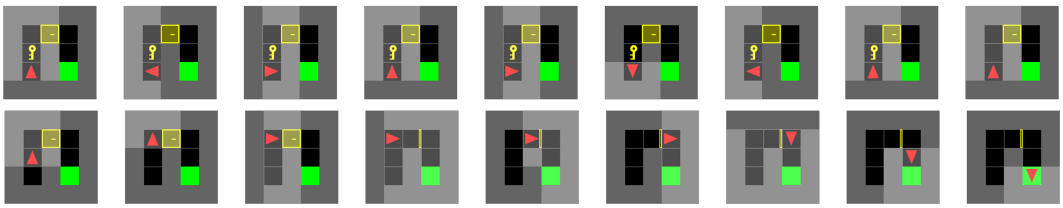


Figure 5: MEDIC-augmented LLM-generated plan for DoorKey-5x5 Layout 1.

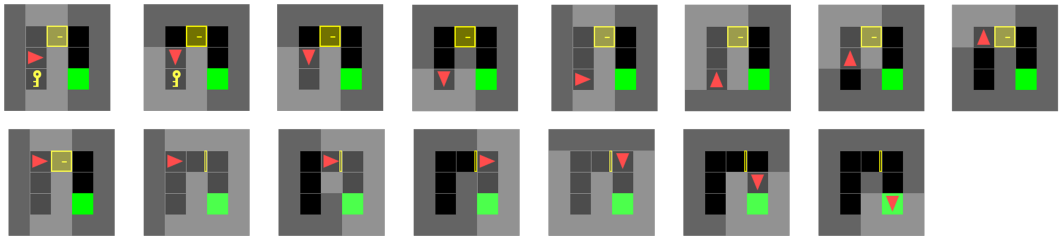


Figure 6: MEDIC-augmented LLM-generated plan for DoorKey-5x5 Layout 2.

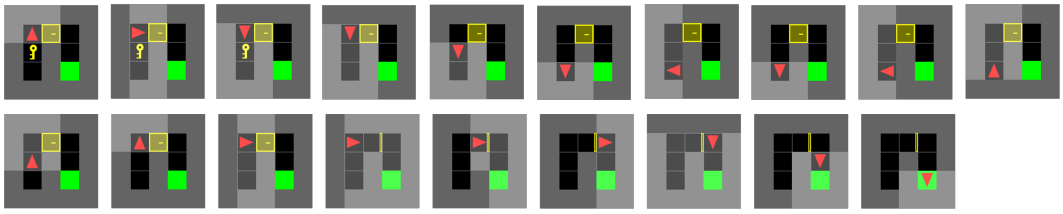


Figure 7: MEDIC-augmented LLM-generated plan for DoorKey-5x5 Layout 3.

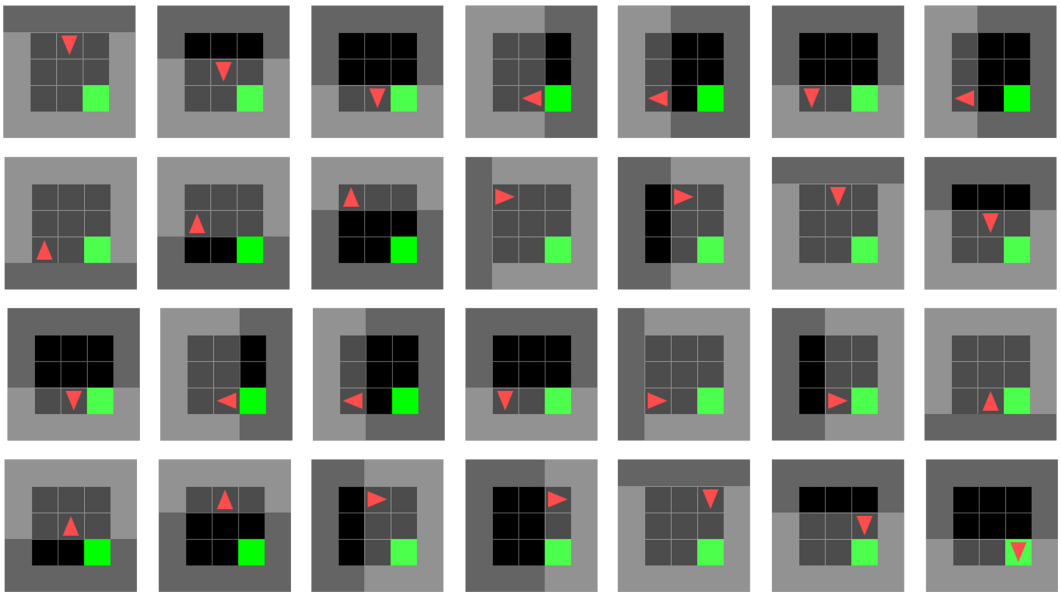


Figure 8: MEDIC-augmented LLM-generated plan for Empty-Random.

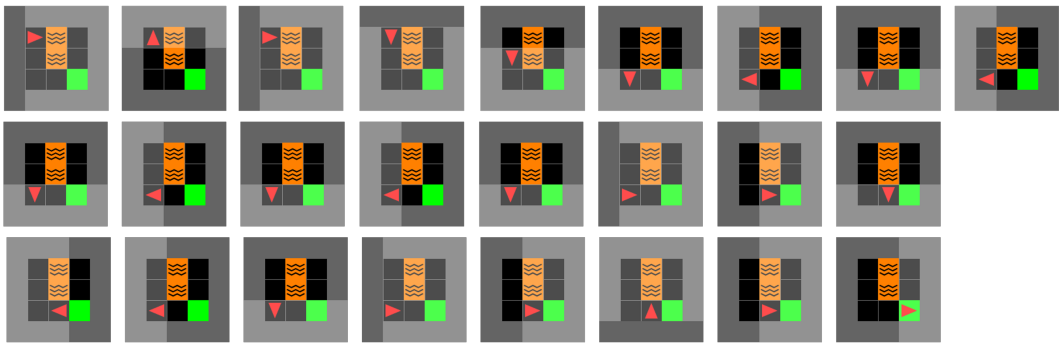


Figure 9: MEDIC-augmented LLM-generated plan for LavaGap.

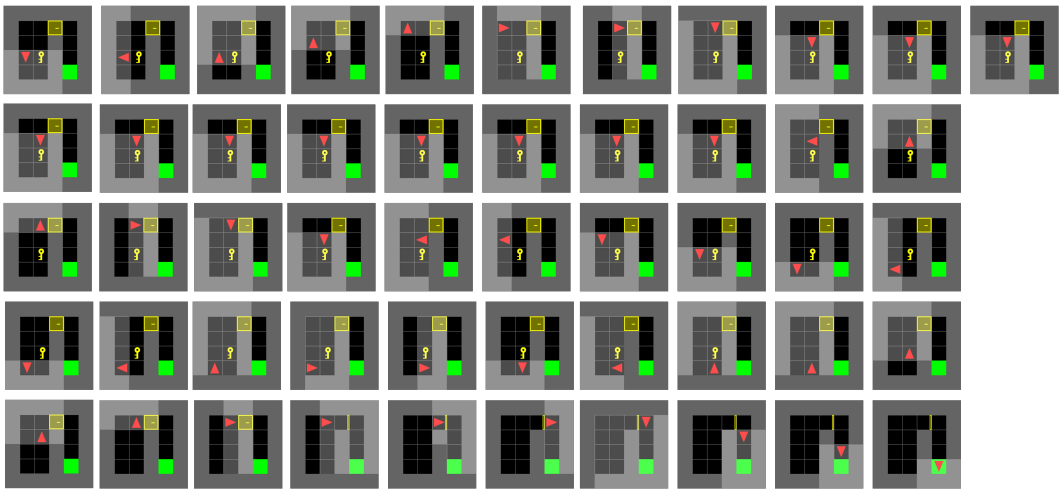
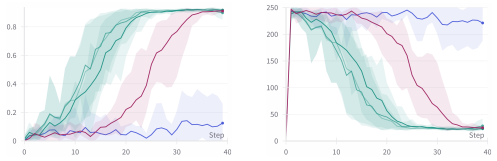


Figure 10: MEDIC-augmented LLM-generated plan for DoorKey-6x6.

C.4 RQ2: Ablation on the Reward Shaping Function

Given a valid plan for the relaxed deterministic problem generated by our MEDIC-augmented LLM framework, we assign uniformly increasing rewards to each (state, action) pair that is part of the plan. The total reward assigned to this plan is +1. During the RL training stage, we check for (state, action) pairs in the training buffer that correspond to those that are part of the MEDIC-augmented LLM-generated plan. Finally, we update the buffer by assigning these intrinsic rewards and continuing the training process. For ablation, we assign a reward of + $\frac{1}{n}$ to each (state, action) pair where n is the number of steps to reach the goal. From Figure 11, we note that assigning uniformly increasing rewards using the MEDIC-augmented LLM-generated plan performs significantly better than the ablation case.

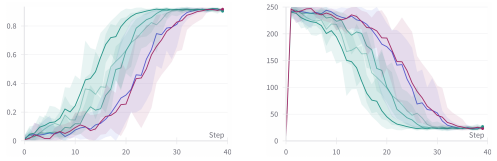


(a) Episodic Returns (b) Average # Steps to Goal

Figure 11: Ablating with reward shaping: Learning curves across five runs comparing vanilla PPO, vanilla PPO with uniformly increasing intrinsic reward shaping using our augmented LLM-generated plan 1, plan 2, and, plan 3, and uniform distribution of intrinsic rewards in ablation as measured on episodic returns and the number of steps to reach the goal in *DoorKey-5x5* environment. The solid lines and shaded regions represent the mean and min/max range, respectively.

C.5 RQ2 Ablation: Reward Shaping without MEDIC

Given a partially correct plan for the relaxed deterministic problem generated by directly prompting LLM without our MEDIC framework, we assign a +0.1 reward to the correct (state, action) pairs that are part of the plan until the first incorrect action occurs. Same as above, during the RL training stage, we check for (state, action) pairs in the training buffer that correspond to those that are part of this LLM-generated plan. Finally, we update the buffer by assigning these intrinsic rewards and continuing the training process. From the comparison shown in Figure 12, we note that as expected, the sample efficiency boost is much lesser with a partially correct plan than with a valid plan. In our case, the partially correct plan only consists of a single correct action generated by the LLM for one layout of *DoorKey-5x5* environment.

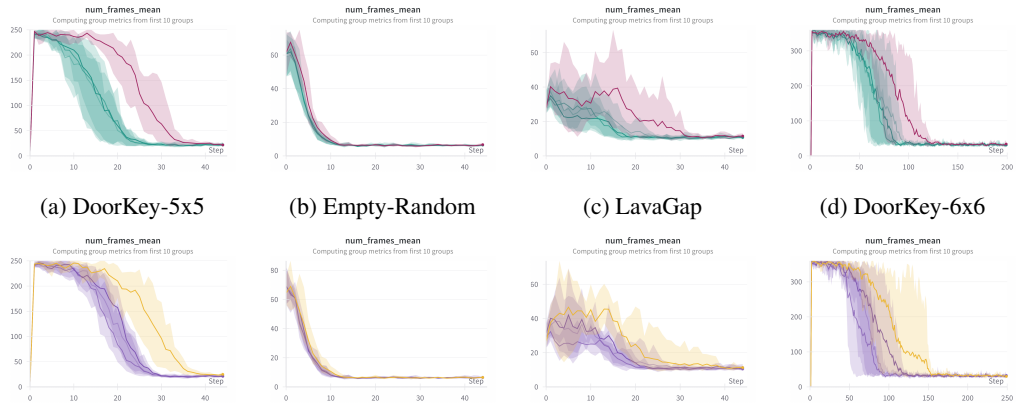


(a) Episodic Returns (b) Average # Steps to Goal

Figure 12: Ablating without MEDIC: Learning curves across five runs comparing vanilla PPO, vanilla PPO with uniformly increasing intrinsic reward shaping using our augmented LLM-generated plan 1, plan 2, and, plan 3, and uniform distribution of intrinsic rewards in ablation as measured on episodic returns and the number of steps to reach the goal in *DoorKey-5x5* environment. The solid lines and shaded regions represent the mean and min/max range, respectively.

C.6 RQ2: RL Training with Shaped Rewards

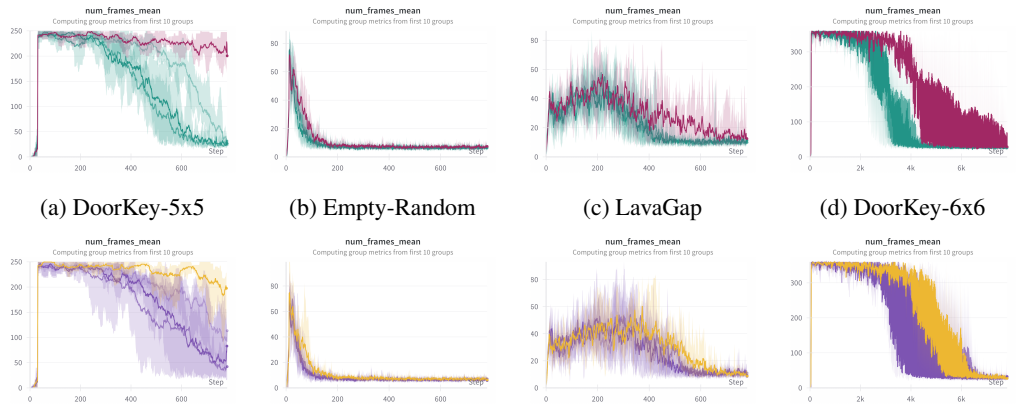
We show additional results from our RQ2 experiment setting. For both PPO and A2C training across the four environments, we showed the episodic returns in Section 4. Here, we show the average number of steps taken by the RL agent to reach the goal, in Figures 13 and 14. Furthermore, we run additional experiments on 2 additional layouts of the *DoorKey-5x5* environment as shown in Figure 4. The training results on episodic returns and average number of steps to reach the goal for PPO have been shown in Figure 15, and for A2C in Figure 16.



(a) DoorKey-5x5 (b) Empty-Random (c) LavaGap (d) DoorKey-6x6

(e) DoorKey-5x5 (f) Empty-Random (g) LavaGap (h) DoorKey-6x6

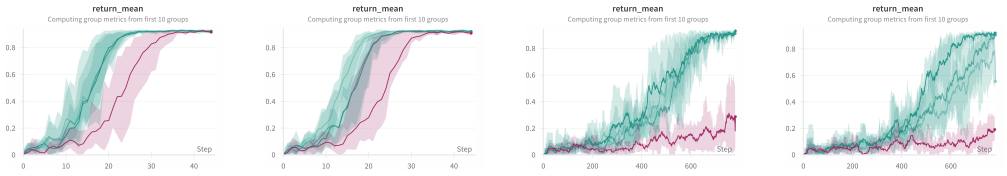
Figure 13: **RQ2 Results:** *Top-* Learning curves across five runs comparing vanilla PPO training against vanilla PPO with reward shaping using our augmented LLM-generated plan 1, plan 2, and, plan 3, as measured on the number of steps to reach the goal. *Bottom-* Learning curves across five runs comparing vanilla-text PPO training against vanilla-text PPO with reward shaping using our augmented LLM-generated plan 1, plan 2, and, plan 3, as measured on the number of steps to reach the goal. The solid lines and shaded regions represent the mean and min/max range, respectively.



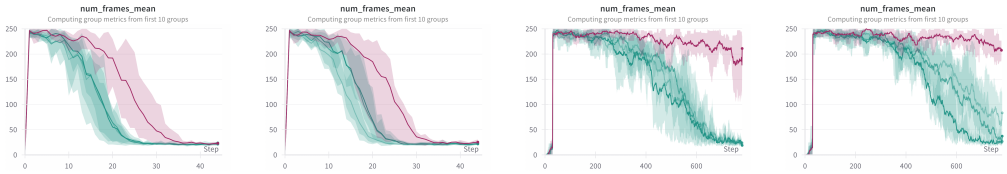
(a) DoorKey-5x5 (b) Empty-Random (c) LavaGap (d) DoorKey-6x6

(e) DoorKey-5x5 (f) Empty-Random (g) LavaGap (h) DoorKey-6x6

Figure 14: **RQ2 Results:** *Top-* Learning curves across five runs comparing vanilla A2C training against vanilla A2C with reward shaping using our augmented LLM-generated plan 1, plan 2, and, plan 3, as measured on the number of steps to reach the goal. *Bottom-* Learning curves across five runs comparing vanilla-text A2C training against vanilla-text A2C with reward shaping using our augmented LLM-generated plan 1, plan 2, and, plan 3, as measured on the number of steps to reach the goal. The solid lines and shaded regions represent the mean and min/max range, respectively.

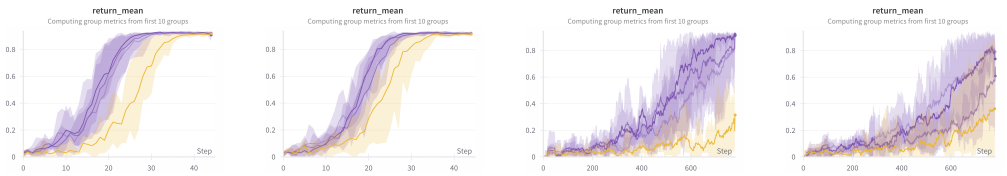


(a) DoorKey-5x5-Layout 2 (b) DoorKey-5x5-Layout 3 (c) DoorKey-5x5-Layout 2 (d) DoorKey-5x5-Layout 3

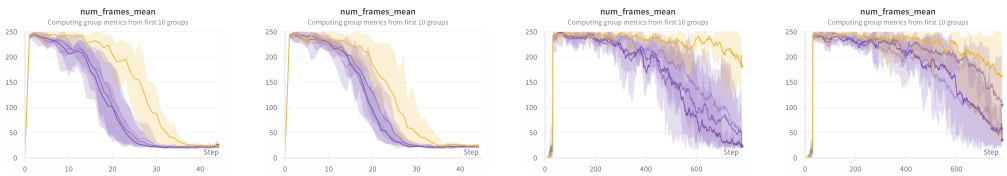


(e) DoorKey-5x5-Layout 2 (f) DoorKey-5x5-Layout 3 (g) DoorKey-5x5-Layout 2 (h) DoorKey-5x5-Layout 3

Figure 15: **RQ2 Results:** Learning curves across five runs comparing vanilla PPO training against vanilla PPO with reward shaping using our augmented LLM-generated plan 1, plan 2, and, plan 3, as measured on (*Top*) episodic returns, and (*Bottom*) the number of steps to reach the goal. The solid lines and shaded regions represent the mean and min/max range, respectively.



(a) DoorKey-5x5-Layout 2 (b) DoorKey-5x5-Layout 3 (c) DoorKey-5x5-Layout 2 (d) DoorKey-5x5-Layout 3



(e) DoorKey-5x5-Layout 2 (f) DoorKey-5x5-Layout 3 (g) DoorKey-5x5-Layout 2 (h) DoorKey-5x5-Layout 3

Figure 16: **RQ2 Results:** Learning curves across five runs comparing vanilla-text PPO training against vanilla-text PPO with reward shaping using our augmented LLM-generated plan 1, plan 2, and, plan 3, as measured on *Top* episodic returns, and (*Bottom*) the number of steps to reach the goal. The solid lines and shaded regions represent the mean and min/max range, respectively.

C.7 Hyperparameters

The hyperparameters used for all RL training experiments are listed in Table 2.

Table 2: Hyperparameters for all RL training experiments.

Hyperparameter	**Value**
# of training steps	1e6
# of epochs	4
batch size	256
discount	0.99
learning rate	0.001
gae lambda	0.95
entropy coefficient	0.01
value loss coefficient	0.5
max gradient norm	0.5
optimizer epsilon	1e-08
optimizer alpha	0.99
clip epsilon	0.2

D Prompts

D.1 RQ1: MEDIC-augmented LLM prompts

Here we show representative step-prompts and corresponding back prompts for one seed of the DoorKey 5x5 environment. Prompts for other environments also follow a similar structure.

[STEP PROMPTING]
You are tasked with solving a 3x3 maze where you will encounter objects like a key and a door along with walls. Your task is 'use the key to open the door and then get to the goal'. You can be facing in any of the four directions. To move in any direction, to pick up the key, and to open the door, you need to face in the correct direction. You will be given a description of the maze at every step and you need to choose the next action to take. The available actions are 'turn left', 'turn right', 'move forward', 'pickup key', 'open door'.
The current maze looks like this:

unseen door unseen
key wall unseen
agent wall goal

You (agent) are currently facing left.
What is the next action that the agent should take? Only choose from the list of available actions. Do not include anything else in your response. For example, if you choose 'move forward', then only write 'move forward' in your response.
LLM Response: pickup key
[BACK PROMPTING]

Information: You cannot 'pickup key' in this state as you are not facing the key. Please choose another action.The following actions are feasible in this state: ['turn right', 'turn left'].
The current maze looks like this:

unseen door unseen
key wall unseen
agent wall goal

You (agent) are currently facing left.
What is the next action that the agent should take? Only choose from the list of available actions. Do not include anything else in your response. For example, if you choose 'move forward', then only write 'move forward' in your response. You have already tried the following actions: pickup key. Please choose another action.
LLM Response: turn right
...

D.2 RQ1: Direct LLM prompts

Here we show the exact prompt for one seed of the DoorKey 5x5 environment.

D.2.1 *Querying the entire policy at once*

You are tasked with solving a 3x3 maze where you will encounter objects like a key and a door along with walls. Your task is 'use the key to open the door and then get to the goal'. You can be facing in any of the four directions. To move in any direction, to pick up the key, and to open the door, you need to face in the correct direction. The available actions at each step are 'turn left', 'turn right', 'move forward', 'pickup key', 'open door'.
The current maze looks like this:

unseen door unseen
key wall unseen
agent wall goal

You (agent) are currently facing left.
What is the sequence of actions you will take to reach the goal? Output as a comma separated list. Do not include anything else in your response.

D.2.2 *Querying actions step by step*

You are tasked with solving a 3x3 maze where you will encounter objects like a key and a door along with walls. Your task is 'use the key to open the door and then get to the goal'. You can be facing in any of the four directions. To move in any direction, to pick up the key, and to open the door, you need to face in the correct direction. You will be given a description of the maze at every step and you need to choose the next action to take. The available actions are 'turn left', 'turn right', 'move forward', 'pickup key', 'open door'.
The current maze looks like this:

unseen door unseen
key wall unseen
agent wall goal

You (agent) are currently facing left.
What is the next action that the agent should take? Only choose from the list of available actions. Do not include anything else in your response. For example, if you choose 'move forward', then only write 'move forward' in your response.