Incentivizing both Grounding and Reasoning in Large Language Models with Online Reinforcement Learning

Pavanpreet Singh Gandhi, Julian Ma, Xu Chen, Michael Yiu
Department of Computer Science, University College London
{pavanpreet.gandhi.24, julian.ma.24, xu.chen.24, chun.yiu.24}@ucl.ac.uk

Abstract

We investigate whether prompting LLM agents to reason before action selection improves sample efficiency and generalization during online reinforcement learning in BabyAI-Text environments. Unlike prior approaches, our method allows LLMs to generate actions token-by-token with a nested RL approach that attributes rewards across the decision process. Comparing agents prompted to reason before acting against those generating actions directly, we find explicit reasoning does not consistently yield performance advantages, with both agent types achieving comparable success rates after PPO fine-tuning. We identify a "reasoning collapse" phenomenon where agents occasionally simply state actions without providing thought processes. While reasoning agents exhibited similar performance to direct-action agents, they showed greater training instability. Our findings suggest explicit reasoning offers valuable interpretability without compromising effectiveness, contributing to understanding the interplay between reasoning, grounding, and reinforcement learning in LLM-based agents[1].

1 Introduction

Large Language Models (LLMs) have demonstrated promising capabilities in sequential decision-making problems (Yao et al., 2023a,b), leveraging pre-trained knowledge to navigate environments without task-specific fine-tuning (Wang et al., 2023a). However, trained primarily on next-token prediction, LLMs lack a *grounded* understanding of environment dynamics—they cannot reliably map tokens to actions, anticipate states, or comprehend reward implications. Recent research (Carta et al., 2023) addresses this limitation through online reinforcement learning (RL) to improve grounding.

On the other hand, RL has emerged as a method for enhancing LLMs by aligning with human preferences (Ouyang et al., 2022) and incentivizing reasoning capabilities (DeepSeek-AI et al., 2025; Lambert et al., 2024). Yet the combination of reasoning and online RL for grounding LLMs in sequential decision-making remains relatively unexplored.

We investigate whether explicit reasoning during online RL fine-tuning improves LLM-based agents' performance. Using BabyAI-Text (Carta et al., 2023), we compare (1) agents prompted to reason before action selection, and (2) agents generating actions directly without reasoning steps. Unlike prior work (Carta et al., 2023), our approach allows token-by-token action generation rather than selection from predefined options. We hypothesized reasoning-enabled agents would demonstrate superior sample efficiency and generalization to novel tasks. Our results reveal that explicit reasoning provides no consistent performance advantage over direct action generation, with both approaches achieving comparable success rates after PPO fine-tuning. Furthermore, reasoning agents sometimes exhibit greater training instability and occasionally a "reasoning collapse" phenomenon where explanatory text devolved into simply stating the intended action. Nevertheless, when they work, reasoning-based agents offered valuable interpretability benefits without compromising effectiveness.

2 Related Work

2.1 LLM Reasoning

LLM reasoning capabilities have seen substantial advances through both supervised prompt-

[1]Our source code can be found in https://github.com/pavanpreet-gandhi/rl-llm

ing and reinforcement learning. A foundational idea is Chain-of-Thought (CoT) prompting (Wei et al., 2022), where LLMs are prompted to articulate intermediate reasoning steps to improve performance on arithmetic and commonsense tasks. This idea is further refined in Self-Consistency (Wang et al., 2023b), which samples multiple CoT paths and selects the most coherent solution and in Tree of Thoughts (Yao et al., 2023a) where multiple reasoning paths are evaluated before final output. Similarly, Scratchpad-style reasoning (Nye et al., 2021) encourages LLMs to generate and retain intermediate states during problem solving, improving interpretability and generalization; while Reflexion (Shinn et al., 2023) proposed a framework to allow LLMs to verbally reflect on task feedback signals which are then maintained in a memory buffer to improve decision making.

Other works leverage reinforcement learning to enhance reasoning. RLHF (Ouyang et al., 2022) adapts PPO to fine-tune model outputs from human preferences, while DeepSeek-R1 (DeepSeek-AI et al., 2025) and TULU-3 (Lambert et al., 2024) leverage reinforcement learning and verifiable rewards to incentivise reasoning capabilities. VinePPO (Kazemnejad et al., 2024) proposes a PPO variant that replaces the value network with Monte Carlo rollouts for intermediate steps, directly supporting reasoning processes. Recent efforts like STaR (Zelikman et al., 2022) introduce iterative self-training to refine CoT traces over time, demonstrating that LLMs can bootstrap their own reasoning with minimal supervision.

More structured paradigms have also emerged. Towards System 2 Reasoning (Xiang et al., 2025) extends CoT via explicit modeling of the underlying reasoning ("Meta-CoT") and explored methods including process supervision, synthetic data generation and search algorithms. Cognitive Behaviors for Self-Improving Reasoners (Gandhi et al., 2025) identifies four key strategies—verification, backtracking, subgoal setting, and backward chaining—that are critical for performance improvements in RL training.

Collectively, these studies highlight the importance of structured, interpretable reasoning in LLMs. In our work, we evaluate the effect of prompting the agent to explicitly reason before generating actions, with the goal of improving sample efficiency and generalization in BabyAI-Text environments.

2.2 LLM Agents

A growing body of work investigates how LLMs can be adapted into interactive agents capable of grounding language into actions. These agents must interpret instructions, model the world, and make decisions based on partially observable environments. GLAM (Carta et al., 2023) demonstrates that grounded LLMs, when trained with PPO in BabyAI-text environments, achieve both strong task performance and high sample efficiency. Our work is closely related to GLAM, but we extend it by prompting the agent to explicitly reason before generating actions.

Others explore how LLMs can operate as agents by tightly coupling reasoning and action. ReAct (Yao et al., 2023b) interleaves "thoughts" and "acts" during each step, showing that such interleaving improves success rates in interactive decision-making tasks. Inner Monologue (Huang et al., 2023) applies this idea to robotics, allowing agents to maintain internal self-dialogue to better interpret sensor inputs and act appropriately. Although its domain differs from ours, the notion of internal reasoning aligns with our reasoning-before-acting policy design.

Language Models as Zero-Shot Planners (Huang et al., 2022) illustrates that pretrained models such as GPT-3 can compose multi-step symbolic plans without environment interaction or gradient updates, relying on latent world knowledge. SayCan (Ahn et al., 2022) adds an external value function to verify action feasibility, demonstrating the effectiveness of combining LLMs with decision-checking mechanisms — a strategy structurally similar to our use of value heads trained via PPO.

Recent works have expanded the scope of LLMs as agents by introducing real-world affordances and grounding. Language Models can Solve Computer Tasks (Kim et al., 2023) shows that LLMs can control a simulated computer environment to solve programming and OS-level tasks, learning to act through API interfaces. Meanwhile, Language Models Meet World Models (Xiang et al., 2023) investigates

how interaction with embodied world models — even simulated ones — improves reasoning and task completion, arguing that LLMs benefit from experiential learning.

To enhance LLM agency in multimodal or partially grounded environments, Grounding Multimodal Large Language Models in Actions (Szot et al., 2024) trains models that integrate language, vision, and proprioception. The authors show that with sufficient alignment, these models can directly translate instructions into physical actions, highlighting the importance of reward-aligned updates and latent action representations — themes echoed in our own use of PPO and value modeling for action refinement.

Other frameworks such as Voyager (Wang et al., 2023a) use LLMs to autonomously explore and craft tools in Minecraft, Toolformer (Schick et al., 2023) use LLM to interact with external tools using APIs while LOOP (Chen et al., 2025) and Learning to Search in Language (Gandhi et al., 2024) enable agents to iteratively improve their plans by simulating search processes in natural language. These systems represent a shift from prompt-only agents to learning systems that adapt through interaction, long-term memory, or reinforcement.

These works point toward a convergence between LLMs and traditional agentic systems. Through reinforcement learning, interaction, and grounding, LLMs are increasingly capable of planning, adapting, and acting in dynamic environments. Our study contributes further by examining how reasoning affects sample efficiency and policy generalization in a text-based BabyAI environment.

3 Methods

3.1 Environment: BabyAI-Text

BabyAI-Text (Carta et al., 2023) is a textual extension of the BabyAI platform (Chevalier-Boisvert et al., 2018) designed to study functional grounding of language models in interactive environments. It replaces the original symbolic observations of BabyAI with natural language descriptions, enabling interaction through a text-only interface. The environment retains the procedurally generated minigrid setup and low-level navigation tasks, while providing language-based goals and observations. Agents must interpret textual inputs, reason about spatial configurations, and execute actions like "go forward" or "pick up" to achieve sparse reward objectives. An example interaction is shown in Appendix A.

We systematically vary environment complexity by introducing different quantities of distractor objects (0, 3, or 5) that the agent must navigate around or distinguish from target objects. These distractors create more complex spatial configurations and increase the perceptual load, requiring the agent to filter relevant information from observations.

The reward structure is sparse and goal-driven: the agent receives a positive reward (maximum +1) only upon successful task completion, with higher values for faster completion. Most intermediate steps yield zero reward, except for invalid actions which incur a -1 penalty. This sparse, delayed reward signal presents a challenging credit assignment problem, motivating our nested reinforcement learning approach as described in Section 3.3.

3.2 Prompt Design and Action Output

To investigate how explicit reasoning affects the performance and sample efficiency of the LLM agent, we designed two different prompting strategies for the agent's action output:

Direct Action (No Reasoning) In this approach, the agent is prompted to output only a valid action command without additional commentary. The LLM's response at each time step is simply an environment action (e.g., "pick up") in free-form text. This setup treats the LLM purely as a policy $\pi(a|s)$, mapping observations and instructions directly to actions, analogous to standard RL agents. The full prompt is provided in Appendix B.

Reasoning For the reasoning variant, the prompt encourages the agent to articulate a brief "thought" process before outputting the action. The prompt suggests limiting reasoning to "at most 10 words", though this is not enforced as a hard limit but rather a guideline to prevent excessively long reasoning chains for computational reasons. For example, an agent might respond: *"The key is needed first. final answer: pick up"*. The text preceding *"final answer:"* represents the agent's internal reasoning, which is ignored by the environment;

only the action specified after the delimiter is executed.

In both modes, the context provided to the LLM includes the initial mission instruction and a history of previous observations and actions. Critically, when the agent produces an invalid response, we replace it in the context window with a warning message rather than preserving the original text. This design choice proved essential for training stability, as invalid responses often contain nonsensical characters that can trigger negative feedback loops where the model continues to produce similar invalid outputs. By sanitizing the context window in this way, we prevent the model from being influenced by its own previously invalid responses. More details on context window management and handling of invalid responses are provided in Appendices C and D.

3.3 PPO-Based LLM Fine-tuning

Following Carta et al.'s GLAM method (Carta et al., 2023), we employ Proximal Policy Optimization (PPO) (Schulman et al., 2017) to fine-tune the LLM's policy. Unlike GLAM, which computes conditional probabilities for each action, our approach allows the model to generate free-form responses, and we extend this framework even further to integrate explicit reasoning during training.

Our policy is based on LLaMA-3.2-3B-Instruct (chosen for its performance in preliminary tests; see Appendix E), augmented with a small value head (single linear layer applied to the final hidden state) that estimates state value $V_\phi(s_t, y_t^{1:m-1})$, where s_t is the query sequence at t and $y_t^{1:m-1}$ are the response sequence before token y_t^m. During PPO training, this value head is updated to minimise the mean-squared error to the empirical returns, while the policy parameters are updated by maximising the PPO clipped objective. We use a KL-divergence penalty to keep the policy close to pretrained distributions and employ Low-Rank Adaptation (LoRA) (Hu et al., 2021) for parameter-efficient fine-tuning.

Training LLM agents introduces a critical challenge: reconciling the nested nature of environment interaction and token generation. The agent operates simultaneously in two distinct loops with different granularity of observation, action, and reward. In the environment loop, the agent processes textual descriptions of the environment state and produces valid commands (e.g., "turn left", "go forward"). Rewards are typically sparse and often delayed until task completion. This loop represents the traditional reinforcement learning scenario where the agent's objective is to maximize cumulative environmental rewards. Meanwhile, within each environment step, the token generation loop unfolds as the LLM auto-regressively produces individual tokens conditioned on the context, history, and partial outputs. Each token prediction constitutes an LLM action drawn from the model's vocabulary, and the tokens themselves make up the environment action. The challenge lies in appropriately attributing credit from sparse environment rewards to individual token predictions.

Our nested-loop challenge is addressed using a dual-level credit assignment approach. Algorithm 1 details our nested PPO training implementation. To regularize training, we incorporate penalty-free future returns while restricting penalties to immediate TD returns, see more details in Appendix F. Training hyperparameters are fully documented in Appendix G.

For comparison, Transformer Reinforcement Learning's `PPOTrainer`[2] implements language-based PPO where query-response pairs receive scalar scores. This approach is more akin to contextual bandits where all the environment level (outer loop) actions recieve the same reward which is the final reward of the episode, which proves unsuitable for our task by neglecting autocorrelation in sequential environment interactions. We validate our algorithm choice through comparative analysis with `PPOTrainer` in Appendix F.

4 Experiments

To evaluate our hypothesis regarding the benefits of explicit reasoning during reinforcement learning, we conducted a series of controlled experiments across 10 distinct training runs. For all experiments, we used "go to" and "pick up" tasks in the BabyAI-Text environment, because GLAM (Carta et al., 2023) has shown that these tasks are simple enough for LLMs

[2] https://huggingface.co/docs/trl/en/ppo_trainer

Algorithm 1 Nested PPO Training Loops for LLM's

Input: Initial token generation policy parameters θ, value function parameters ϕ, learning rate η, number of iterations K

for $k = 1$ to K **do**

  Collect N trajectories $(s_1, y_1, r_2, s_2, y_2, r_3, \ldots, r_T)$ using policy π_θ

  for each timestep $t = T$ to 1 in a trajectory **do**

    Compute penalty-free future TD(λ) return

$$\hat{G}_{t+1} = \max(0, r_{t+2}) + \gamma((1-\lambda)V_\phi(s_{t+1}, y_{t+1}) + \lambda\hat{G}^\lambda_{t+2})$$

    Compute TD(λ) return

$$G^\lambda_t = r_{t+1} + \gamma((1-\lambda)V_\phi(s_t, y_t) + \lambda\hat{G}^\lambda_{t+1})$$

    for each token $m = 1$ to M **do**

      Compute KL

$$\mathrm{KL}^m_t \;=\; \log \pi_\theta(y^m_t \mid x^m_t) \;-\; \log \pi_{\mathrm{ref}}(y^m_t \mid x^m_t), \quad x^m_t := (s_t, y^{1:m-1}_t)$$

      Compute final reward for each token

$$R^m_t \;=\; \mathbb{1}\{m = M\} \cdot G^\lambda_t \;-\; \beta\,\mathrm{KL}^m_t$$

      Compute TD residual

$$\delta^m_t = R^{m+1}_t + \gamma_{\text{token}} V_\phi(s_t, y^{m+1}_t) - V_\phi(s_t, y^m_t)$$

      Compute GAE on sequence

$$\hat{A}^m_t = \delta^m_t + (\gamma_{\text{token}}\lambda_{\text{token}})\hat{A}^{m+1}_t \qquad \hat{A}^M_t = 0$$

    end for

  end for

  Update policy π_θ by maximizing $\mathcal{L}^{\text{CLIP}}(\theta)$ via gradient ascent

$$\mathcal{L}^{\text{CLIP}}(\theta) = \mathbb{E}_t\left[\min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}\,\hat{A}_t, \text{clip}(r_t(\theta),\, 1{-}\epsilon,\, 1{+}\epsilon)\,\hat{A}_t\right)\right],$$

  Update value function V_ϕ by minimizing

$$\frac{1}{2}\sum\left(V_\phi(s_t, y_t) - G^\lambda_t\right)^2$$

end for

to learn quickly. We established three environment configurations with increasing levels of complexity: (1) **0 distractors**: The agent must navigate to (or pick up) a single target object; (2) **3 distractors**: The agent must navigate with three randomly placed distractor objects; and (3) **5 distractors**: The agent must navigate with five distractor objects. Among all experiments, the only variables that differ are the number of distractors and the reasoning mechanism. All other hyperparameters, are kept constant as described in Table G in the appendix.

4.1 Performance Across Environment Configurations

For each configuration, we trained both reasoning and non-reasoning agents while maintaining consistent hyperparameters. We show the success rate and average reward of batches of episodes collected during training for the three-distractor case in Figure 1 and for the zero and five distractor cases in Appendix H.

In the simplest environment without distractors (Fig. 12), non-reasoning agents achieved perfect success rates (1.0) from the beginning of training, demonstrating effective zero-shot transfer. Reasoning agents initially struggled with generating valid actions but eventually converged to perfect success rates. The mixed results in average reward metrics suggest that in simple environments, explicit reasoning does not provide any significant advantage over the non-reasoning agent.

With three distractors (Fig. 1), we observed one reasoning agent outperform the non-reasoning agents in terms of average reward, while the other reasoning agent experienced unstable training and had to be terminated early. This instability was not observed in non-reasoning agents and required special mitigation techniques discussed in Appendix C.

In the most difficult configuration with five distractors (Fig. 13), both reasoning and non-reasoning agents demonstrated comparable performance in terms of success rates and average rewards.

4.2 Reasoning Collapse

We identified an intriguing pattern during training that we term "reasoning collapse", where agents' explicit reasoning simply states the action they are about to take rather than providing a coherent thought process. Normally, reasoning agents produced coherent thoughts before actions, as shown:

Normal Reasoning Example

Observation:
You see a wall 2 steps right
You see a blue key 3 steps left and 1 step forward
You see a yellow key 2 steps forward
Agent Response:
I should go forward to get slightly closer to the blue key.
final answer: go forward

However, as training progressed, we observed instances where reasoning text simply stated the action:

Collapsed Reasoning Example

Observation:
You see a wall 2 steps forward
You see a wall 3 steps right
You see a purple key 3 steps left
You see a blue key 2 steps left and 1 step forward
Agent Response:
Go forward.
final answer: go forward

Figure 2 displays the average reasoning text length for the 3-distractor environment, with similar patterns observed across all runs. The text length shows high variability throughout training, frequently collapsing to just two words. Later training batches typically contain both normal and collapsed reasoning outputs. This phenomenon suggests either that (1) the agent discovers explicit reasoning isn't necessary for maximizing rewards, or (2) reasoning processes become internalized within the model's latent representations as training advances.

5 Results and Discussion

5.1 Evaluation Results

We evaluated both trained and untrained models across three distinct task environments – varying in environment complexity with 0, 3 and 5 distractors – and compared the two prompting strategies outlined in Section

Figure 1: Training curves for agents in environments with three distractors.

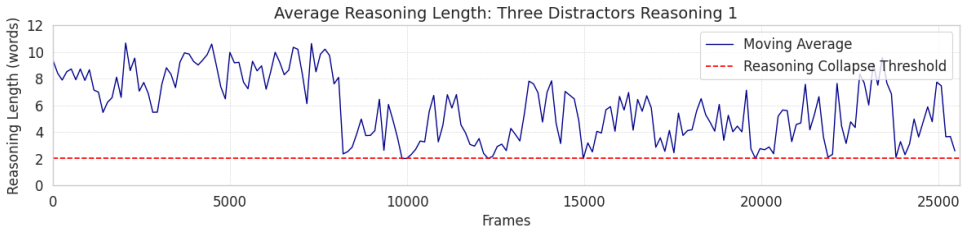

Figure 2: Average length of reasoning text (in words) generated by the reasoning agent over time during training in the 3-distractor environment.

3.2. Each envrionment configuration was assessed using four BabyAI tasks: `GoTo`, `Pickup`, `GoToTest` and `PickupTest`. Performance metrics used include success rate and average reward. The detailed evaluation results for the environment setting with 3 distractors can be found in Table 1, while those from environment setting with 0 and 5 distractors, as well as visualisations of these performance metrics can found in Appendix I. We also evaluated our models' performance on benchmarks outside of BabyAI-Text and found no clear difference between reasoning and non-reasoning models Appendix J.

5.2 Discussion

Zero-distractor Environment Table 3 in Appendix I summarises the performance metrics for both the zero-shot models and the models trained in this environment. In the simplest setting with no distractors, the zero-shot baseline without reasoning achieved a high success rate of 94%, while the reasoning-enabled baseline performed notably worse at 66%, likely due to the additional output formatting constraints and the complexity of reasoning generation. After training, both reasoning and non-reasoning agents consistently achieved 100% success rates, with only minor differences in average rewards and episode lengths. This confirms that, explicit reasoning does not yield tangible performance benefits in simple environments, and it imposes an initial learning overhead – namely, the need to learn to structure outputs according to the reasoning prompt format.

Three-distractor Environment Table 1 summarises the performance metrics for this intermediate-complexity environment. As before, all trained models outperformed their respective zero-shot baselines under direct action prompting. However, a stark discrepancy was observed between the two models trained with the reasoning prompt. Model `Three-Distractor-Reasoning-1` slightly outperformed the non-reasoning models across all four tasks, achieving near-perfect success rates. In contrast, `Three-Distractor-Reasoning-2` underperformed significantly, with success rates as low as 10%. Notably, this model expe-

Env	Model Variant	Reasoning Success Rate ± Std	Non-Reasoning Success Rate ± Std	Reasoning Avg Reward ± Std	Non-Reasoning Avg Reward ± Std
GoTo-v0	Zero-shot	0.46 ± 0.50	0.80 ± 0.40	0.34 ± 0.41	0.53 ± 0.34
	Three-Distractor_1	0.98 ± 0.14	0.98 ± 0.14	0.88 ± 0.15	0.86 ± 0.16
	Three-Distractor_2	0.14 ± 0.35	1.00 ± 0.00	0.13 ± 0.33	0.89 ± 0.08
Pickup-v0	Zero-shot	0.18 ± 0.39	0.42 ± 0.50	0.12 ± 0.28	0.30 ± 0.38
	Three-Distractor_1	0.96 ± 0.20	0.74 ± 0.44	0.86 ± 0.19	0.63 ± 0.40
	Three-Distractor_2	0.22 ± 0.42	0.82 ± 0.39	0.20 ± 0.38	0.68 ± 0.34
GoToTest-v0	Zero-shot	0.62 ± 0.49	0.86 ± 0.35	0.48 ± 0.41	0.62 ± 0.33
	Three-Distractor_1	1.00 ± 0.00	0.98 ± 0.14	0.91 ± 0.08	0.85 ± 0.16
	Three-Distractor_2	0.20 ± 0.40	1.00 ± 0.00	0.19 ± 0.39	0.90 ± 0.07
PickupTest-v0	Zero-shot	0.28 ± 0.45	0.38 ± 0.49	0.19 ± 0.33	0.27 ± 0.37
	Three-Distractor_1	0.92 ± 0.27	0.82 ± 0.39	0.80 ± 0.26	0.73 ± 0.35
	Three-Distractor_2	0.10 ± 0.30	0.72 ± 0.45	0.08 ± 0.26	0.60 ± 0.39

Table 1: Per-variant performance of reasoning and non-reasoning models in environments with three distractors.

rienced unstable training due to it falling into a negative-feedback-loop with invalid actions, resulting in early termination.

This disparity highlights the sensitivity of reasoning-based policies to training stability. However, reasoning agents may offer an advantage in interpretability. By explicitly articulating a short thought processes before taking an action, they make their decision-making more transparent and auditable. For example, reasoning traces such as *"I should go forward to get closer to the blue box. Final answer: go forward"* provide human-interpretable justifications for actions. This can be useful for debugging agent failures, improving human trust, and developing agents for safety-critical applications. Thus, while reasoning may not enhance performance, it can provide interpretability benefits at the expense of requiring more careful training.

Five-distractor Environment Table 4 in Appendix I presents results for the most complex setting (five distractors). Trained models consistently outperformed the zero-shot baselines, which showed success rates ranging from 20% to 72%. Trained agents, by comparison, achieved success rates between 80% and 100%, clearly demonstrating the benefit of PPO fine-tuning. However, in contrast to the three-distractor setting, the reasoning agent did not consistently outperform the non-reasoning agent. For instance, in both `GoTo` and `GoToTest`, the reasoning model achieved slightly lower success rates and exhibited higher variance. These findings suggest that there is no clear benefit of reasoning in terms of sample efficiency or generalization in our setup.

6 Conclusion

In this work, we investigated the impact of explicit reasoning on LLM agents trained with online reinforcement learning. Our approach enables language models to generate actions token-by-token in text-based environments, applying reinforcement learning in a nested fashion to improve performance - a more direct approach than previous methods that compute relative probabilities for each action.

Our experiments revealed that enabling agents to articulate explicit reasoning before generating actions did not significantly improve sample efficiency or generalization across our tested environments. The similar performance between reasoning and non-reasoning agents suggests that in relatively simple environments, the benefits of structured reasoning may be minimal. Nevertheless, we found that explicit reasoning provides valuable interpretability without compromising effectiveness.

Our work has several limitations that future research could address: we tested only on simple task types with varying distractor counts; we observed training instability in reasoning agents that could be mitigated through supervised fine-tuning before RL; and reward shaping techniques could potentially counteract the "reasoning collapse" phenomenon we documented. Future work might explore more complex environments, hybrid training approaches, and reward mechanisms that explicitly value coherent reasoning.

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. 2022. Do as i can, not as i say: Grounding language in robotic affordances. In *arXiv preprint arXiv:2204.01691*.

Thomas Carta, Clément Romac, Thomas Wolf, Sylvain Lamprier, Olivier Sigaud, and Pierre-Yves Oudeyer. 2023. Grounding large language models in interactive environments with online reinforcement learning. *arXiv preprint arXiv:2302.02662*.

Kevin Chen, Marco Cusumano-Towner, Brody Huval, Aleksei Petrenko, Jackson Hamburger, Vladlen Koltun, and Philipp Krähenbühl. 2025. Reinforcement learning for long-horizon interactive llm agents. *arXiv preprint arXiv:2502.01600*.

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

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D Goodman. 2025. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars. *arXiv preprint arXiv:2503.01307*.

Kanishk Gandhi, Denise Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Sharma, and Noah D Goodman. 2024. Stream of search (sos): Learning to search in language. *arXiv preprint arXiv:2404.03683*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. 2022. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. *arXiv preprint arXiv:2201.07207*.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Noah Brown, Tomas Jackson, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. 2023. Inner monologue: Embodied reasoning through planning with language models. In *Proceedings of the 6th Conference on Robot Learning*, volume 205, pages 297–307.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. 2024. Vineppo: Unlocking rl potential for llm reasoning through refined credit assignment. *arXiv preprint arXiv:2410.01679*.

Geunwoo Kim, Pierre Baldi, and Stephen McAleer. 2023. Language models can solve computer tasks. *Advances in Neural Information Processing Systems*, 36:39648–39677.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. 2024. T\" ulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. 2021. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652.

Andrew Szot, Bogdan Mazoure, Harsh Agrawal, R Devon Hjelm, Zsolt Kira, and Alexander Toshev. 2024. Grounding multimodal large language models in actions. *Advances in Neural Information Processing Systems*, 37:20198–20224.

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

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations (ICLR)*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837.

Jiannan Xiang, Tianhua Tao, Yi Gu, Tianmin Shu, Zirui Wang, Zichao Yang, and Zhiting Hu. 2023. Language models meet world models: Embodied experiences enhance language models. *Advances in neural information processing systems*, 36:75392–75412.

Violet Xiang, Charlie Snell, Kanishk Gandhi, Alon Albalak, Anikait Singh, Chase Blagden, Duy Phung, Rafael Rafailov, Nathan Lile, Dakota Mahan, et al. 2025. Towards system 2 reasoning in llms: Learning how to think with meta chain-of-though. *arXiv preprint arXiv:2501.04682*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023a. Tree of thoughts: Deliberate problem solving with large language models.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023b. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488.

A BabyAI-Text Example Interaction

The following is an example of a BabyAI-Text environment interaction, consisting of a mission, observation, and potential actions that the agent can select.

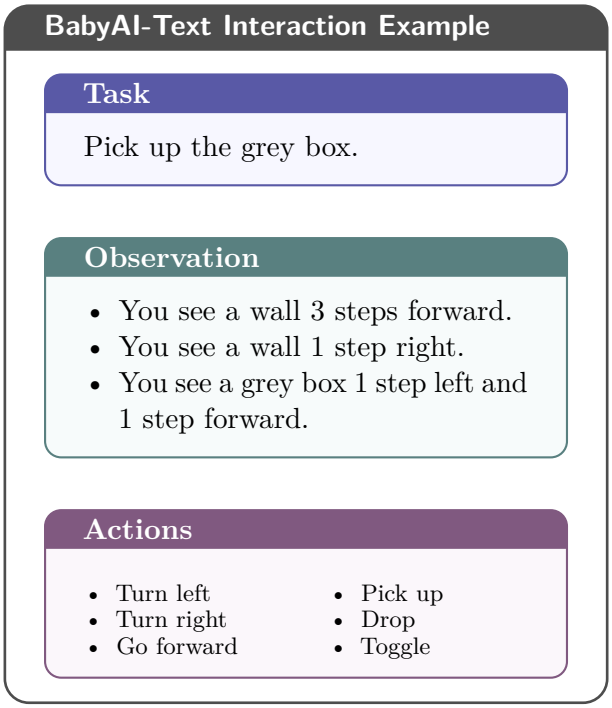

Figure 3: Example of task, observation, and available actions in BabyAI-Text.

B Prompts

System Prompt (No Reasoning)

You are an agent. Your goal is to **{goal}**.
Below are actions you may choose:

- `turn left`
- `turn right`
- `go forward`
- `pick up`
- `drop`
- `toggle`

Note: Only output a valid action (up to two words) and nothing else. **PLAY!**

System Prompt (Reasoning)

You are an intelligent agent. Your goal is to **{goal}**. Before outputting your final answer, think step-by-step (in at most 10 words) about which action best advances your goal.
Your response should end with "`final answer: [answer]`", where `[answer]` is a chosen valid action (up to two words and nothing else) from below:

- `turn left`
- `turn right`
- `go forward`
- `pick up`
- `drop`
- `toggle`

PLAY!

C Invalid Action Regularisation

During training, agents occasionally generated nonsensical output despite using `top k = 50` sampling. When preserved in context history, these invalid responses created problematic negative-feedback loops, causing the model to generate increasingly more invalid actions and destabilizing training.

To mitigate this issue, we implemented two specific approaches:

1. **Context sanitization**: We replaced invalid actions with warning messages in the context window, hiding problematic outputs from the model's history. This prevented the model from being influenced by its own previously invalid responses.

2. **Penalty exclusion**: We excluded the -1 penalty rewards from TD-target calculations, ensuring invalid action penalties only affected the value estimate of the specific action-state pair rather than propagating to future states.

These interventions significantly stabilized training by disrupting negative-feedback cycles, though instability occasionally persisted, as evident in our second reasoning experiment with three distractors.

Below is an example illustrating our invalid action replacement approach.

Without hiding the content of invalid actions from the context

system:

You are an intelligent agent. Your goal is to **pick up a blue box**. Before outputting your final answer, think step-by-step for at most 10 words about which action best advances your goal.
Your response should end with `"final answer: [answer]"`, where `[answer]` is a chosen valid action (up to two words and nothing else) from below:

- `turn left`
- `turn right`
- `go forward`
- `pick up`
- `drop`
- `toggle`

PLAY!

user:

You see a wall 2 steps left
You see a grey key 1 step left and 1 step forward
You see a purple ball 2 steps forward
You see a red ball 1 step right and 2 steps forward

assistant:

ragazialis.imgur castle Immediate Formatting mismehrtub dorsews ralloonexist Innov

user:

Invalid format. Think step-by-step and end your response with 'final answer: [answer]', where [answer] is one of: turn left, turn right, go forward, pick up, drop, toggle, done.

With hiding the invalid actions from the context

system:

You are an intelligent agent. Your goal is to **pick up a blue box**. Before outputting your final answer, think step-by-step for at most 10 words about which action best advances your goal.
Your response should end with `"final answer: [answer]"`, where `[answer]` is a chosen valid action (up to two words and nothing else) from below:

- `turn left`
- `turn right`
- `go forward`
- `pick up`
- `drop`
- `toggle`

PLAY!

user:

You see a wall 2 steps left
You see a grey key 1 step left and 1 step forward
You see a purple ball 2 steps forward
You see a red ball 1 step right and 2 steps forward

assistant:

Generated invalid action

user:

Invalid format. Think step-by-step and end your response with 'final answer: [answer]', where [answer] is one of: turn left, turn right, go forward, pick up, drop, toggle, done.

D Chat Template Example

Chat Template for Reasoning

system:

You are an intelligent agent. Your goal is to **go to the blue ball**. Before outputting your final answer, think step-by-step for at most 10 words about which action best advances your goal. You are navigating in a grid and can only see what is ahead.
Your response should end with `"final answer: [answer]"`, where `[answer]` is a chosen valid action (up to two words and nothing else) from below:

- `turn left`
- `turn right`
- `go forward`
- `pick up`
- `drop`
- `toggle`

PLAY!

user:

You see a wall 2 steps forward
You see a wall 2 steps left

assistant:

Turn right to avoid the wall.
`final answer: turn right`

user:

You see a wall 5 steps forward
You see a wall 2 steps left
You see a blue ball 1 step right and 4 steps forward
You see a purple box 2 steps right and 4 steps forward
You see a purple box 3 steps right and 2 steps forward
You see a blue box 3 steps right

assistant:

Go forward to get closer to the blue ball.
`final answer: go forward`

E Choice of Baseline Model

Prior to our main experiment runs, we conducted tests to pick the most reasonable baseline model and environment setting. We compared out-of-the-box performance across different architectures and model complexity against various BabyAI-Text environments under reasoning and non-reasoning modes.

We compared Llama 3.2 (3B and 1B instruct variants) and Qwen 2.5 (1.5B and 3B variants) and tested their performance across all five BabyAI-Text environments (GoTo, PickUp, Open, PutNext and PickUpSeqGoTo) under reasoning and non-reasoning modes.

We find that the Llama models outperform Qwen, and in particular the Llama-3B variant likely works best for experimentation under reasoning and non-reasoning modes. When we compare the different available BabayAI-Text tasks, it appears that GoTo and PickUp offer the best choice for our setup.

The results below indicate to us that there is a sweet spot between context window length (we observe that too short a context window will likely cause very poor performance and an ineffective use of reasoning), model complexity (we observe that for reasoning to work we need model of at least around 3B parameters) and task type. Based on our results we chose Llama-3.2-3B-Instruct under a context-window of five on the GoTo and Pickup tasks.

We present below the full performance data across different settings.

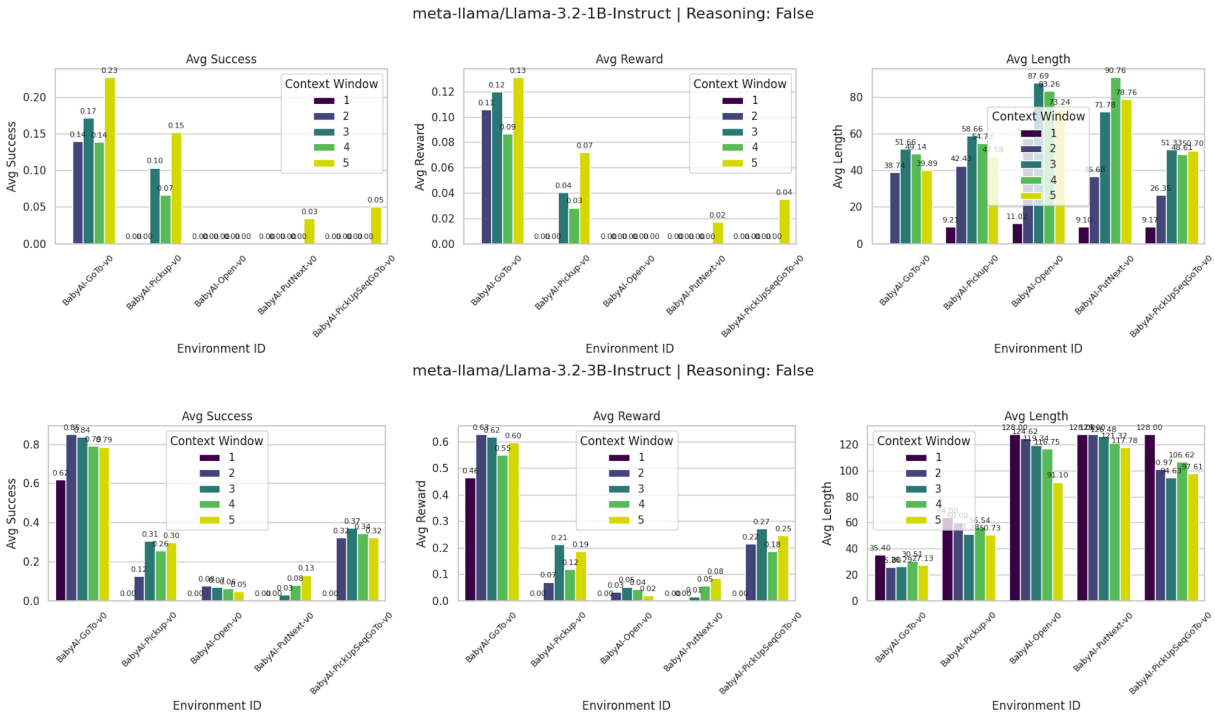

Figure 4: Llama models under non-reasoning mode

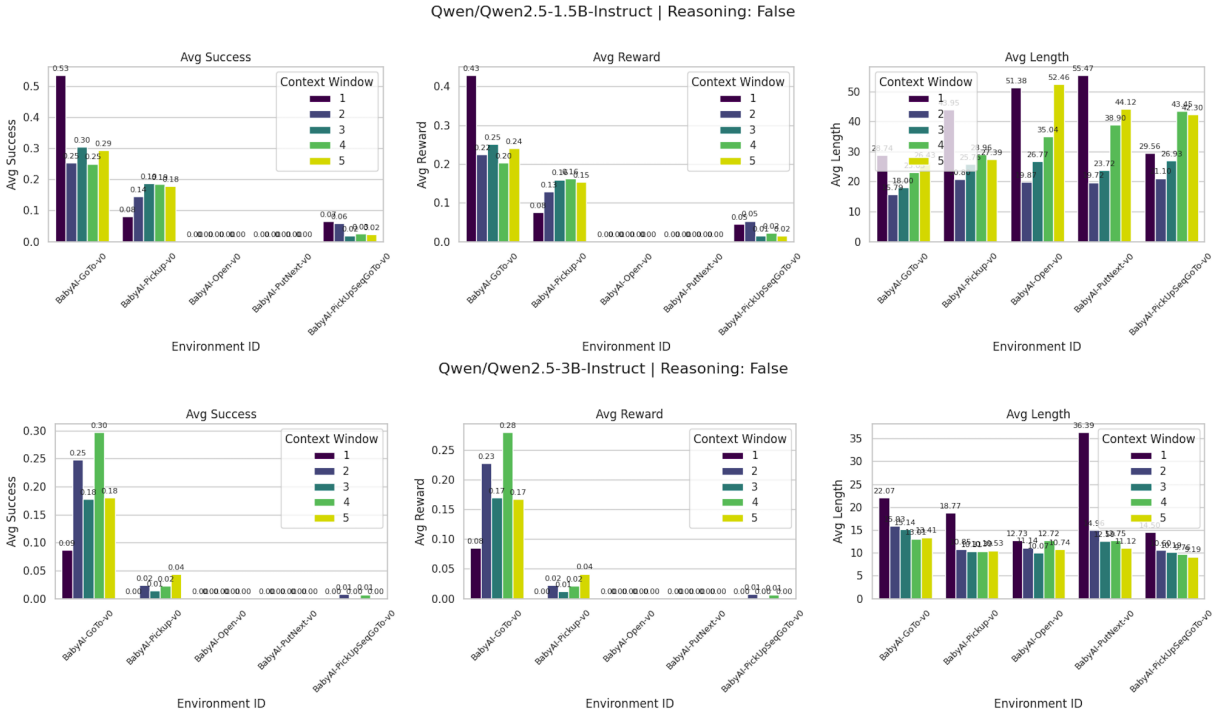

Figure 5: Qwen models under non-reasoning mode

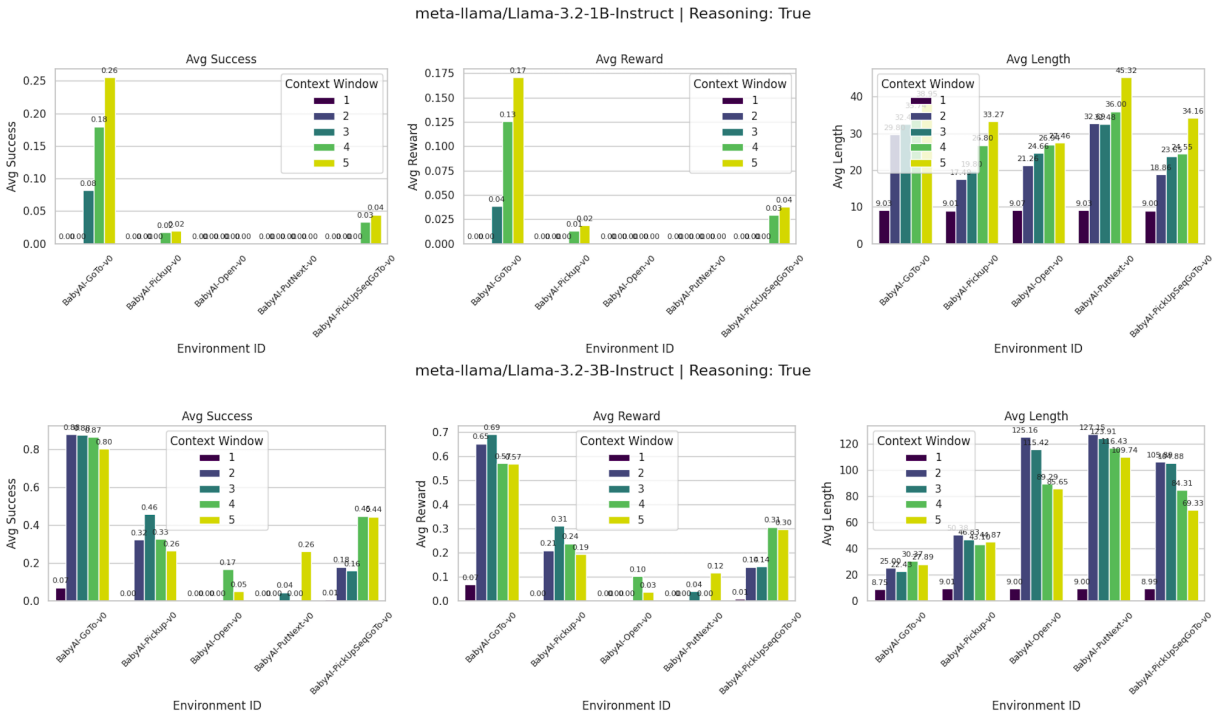

Figure 6: Llama models under reasoning mode

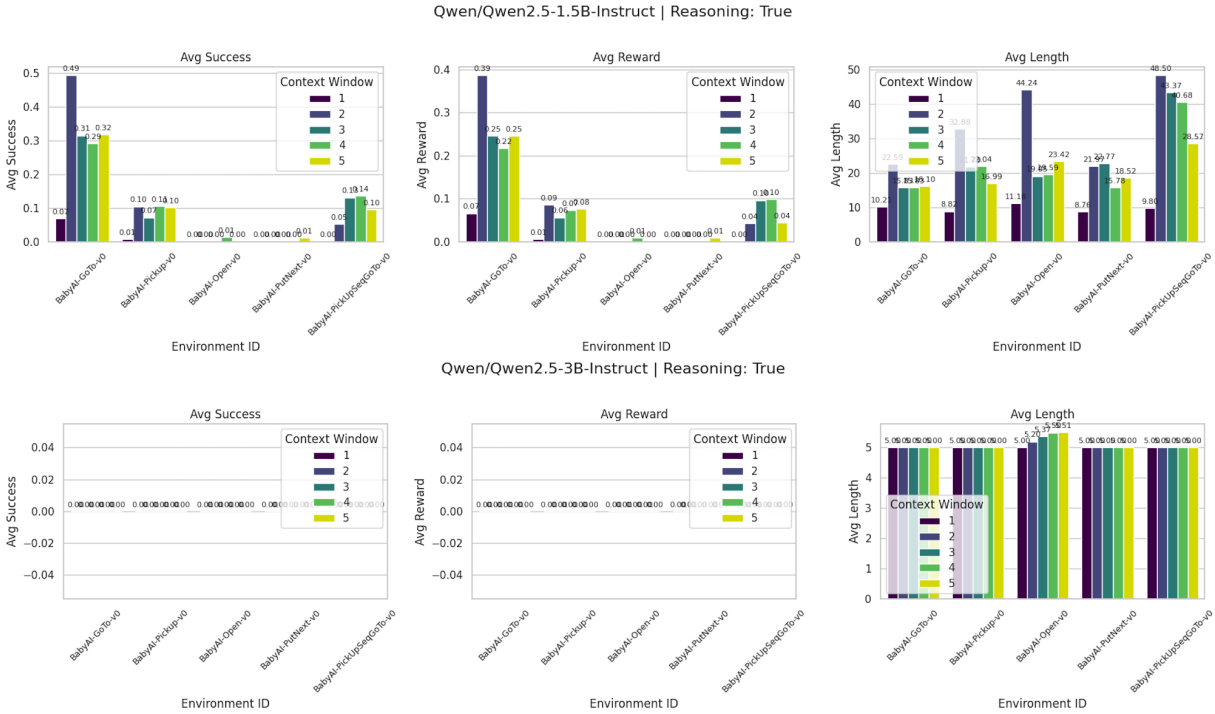

Figure 7: Qwen models under non-reasoning mode

F Contextual Bandit v.s. Nested PPO

We compare the average award and success rate on each batch for both algorithms. Nested PPO has better performance over contextual bandit in both non-reasoning and reasoning setups.

F.1 Non-reasoning

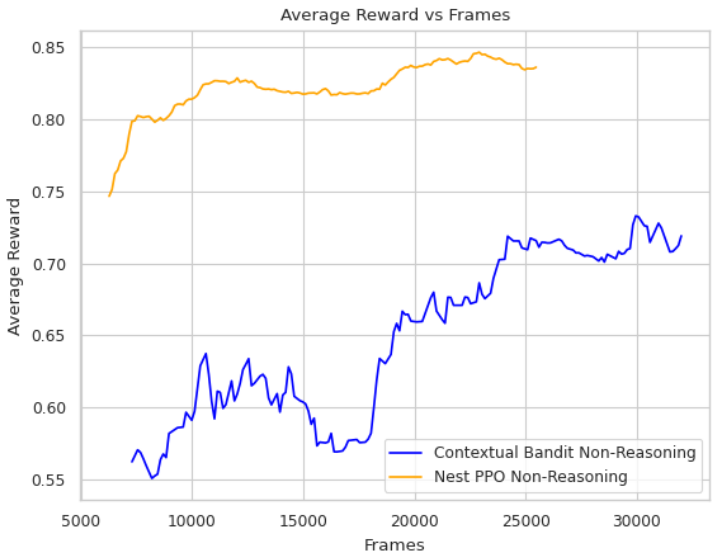

Figure 8: Average Reward in Non-Reasoning Setup

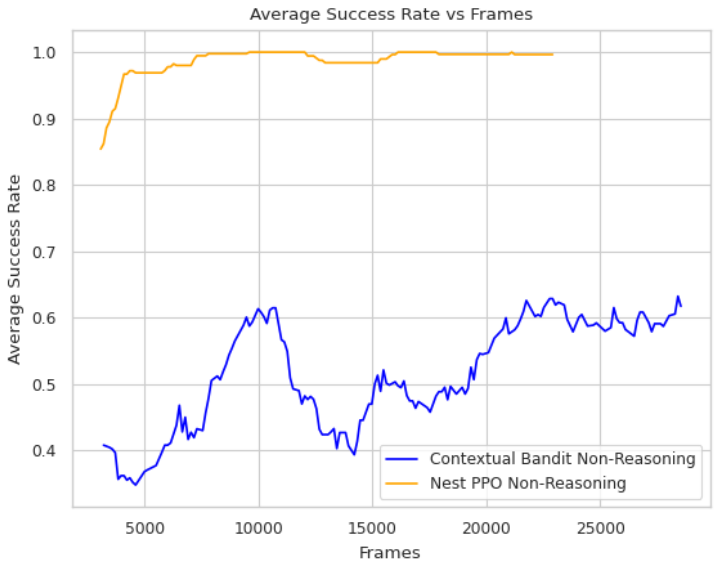

Figure 9: Average Success Rate in Non-Reasoning Setup

F.2 Reasoning

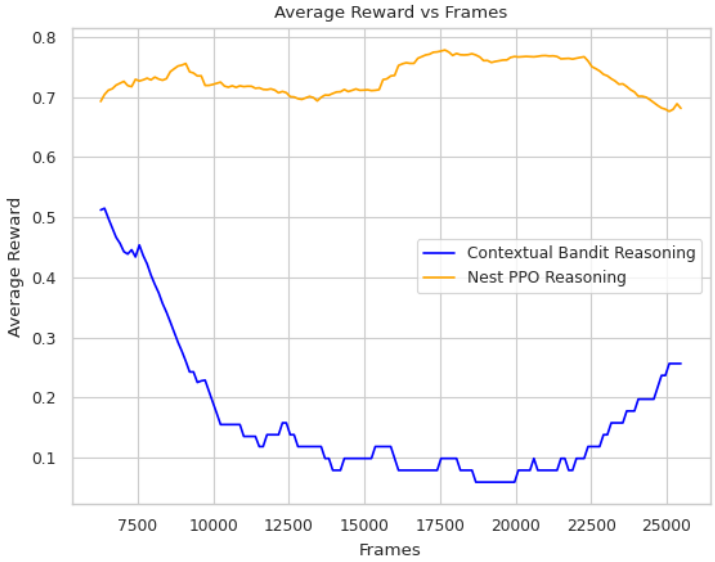

Figure 10: Average Reward in Reasoning Setup

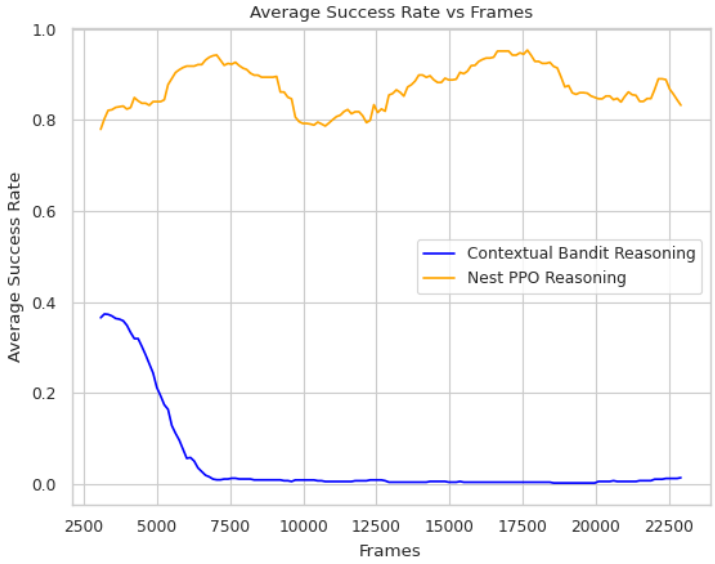

Figure 11: Average Success Rate in Reasoning Setup

G Hyperparameter Table

Hyperparameter	**Value**	**Justification**
num envs	6	Parallel envs, subject to memory capacity
batch size	128	Generated from current policy for policy update
mini batch size	16	For A100 GPU, this is the largest mini batch size we can use
learning rate	1.41E-05	Default in `TRL PPOTrainer`
top k	50	Constrain the token space to avoid nonsense
top p	0.9	
temperature	0.7	
Invalid action penalty	-1	Penalty if the generated action is not valid
early stopping	TRUE	Stop the PPO optimization loop early is the KL too high.
context window	5	Trade-off between history queries and memory capacity
γ	0.9	Discount factor for trajectory return
λ	0.95	Decay factor for trajectory return
ϵ	0.2	Clip parameter in PPO
γ_{token}	1	Discount factor for token GAE
λ_{token}	0.98	Decay factor for token GAE

Table 2: Hyperparameter List

H Training Curves

Figure 12: Training curves for agents in environments without distractors.

Figure 13: Training curves for agents in environments with five distractors.

I Evaluation Metrics

The evaluation metrics on environment with 0 distractors are as follows:

Env	Model Variant	Reasoning Success Rate ± Std	Non-Reasoning Success Rate ± Std	Reasoning Avg Reward ± Std	Non-Reasoning Avg Reward ± Std
GoTo-v0	Zero-shot	0.66 ± 0.48	0.94 ± 0.24	0.55 ± 0.41	0.69 ± 0.28
	Zero-Distractor_1	1.00 ± 0.00	1.00 ± 0.00	0.85 ± 0.10	0.91 ± 0.05
	Zero-Distractor_2	1.00 ± 0.00	1.00 ± 0.00	0.90 ± 0.07	0.92 ± 0.05
Pickup-v0	Zero-shot	0.64 ± 0.48	0.68 ± 0.47	0.47 ± 0.39	0.47 ± 0.36
	Zero-Distractor_1	0.88 ± 0.33	1.00 ± 0.00	0.69 ± 0.30	0.85 ± 0.10
	Zero-Distractor_2	1.00 ± 0.00	1.00 ± 0.00	0.88 ± 0.06	0.86 ± 0.11
GoToTest-v0	Zero-shot	0.44 ± 0.50	0.96 ± 0.20	0.36 ± 0.42	0.75 ± 0.23
	Zero-Distractor_1	0.98 ± 0.14	1.00 ± 0.00	0.82 ± 0.18	0.91 ± 0.05
	Zero-Distractor_2	1.00 ± 0.00	1.00 ± 0.00	0.90 ± 0.06	0.92 ± 0.05
PickupTest-v0	Zero-shot	0.72 ± 0.45	0.82 ± 0.39	0.53 ± 0.36	0.50 ± 0.31
	Zero-Distractor_1	0.96 ± 0.20	1.00 ± 0.00	0.80 ± 0.21	0.87 ± 0.10
	Zero-Distractor_2	1.00 ± 0.00	1.00 ± 0.00	0.87 ± 0.06	0.87 ± 0.08

Table 3: Summary statistics of zero-shot model and trained models on environments with 0 distractors

The evaluation metrics on environment with 5 distractors are as follows:

Env	Model Variant	Reasoning Success Rate ± Std	Non-Reasoning Success Rate ± Std	Reasoning Avg Reward ± Std	Non-Reasoning Avg Reward ± Std
GoTo-v0	Zero-shot	0.46 ± 0.50	0.72 ± 0.45	0.33 ± 0.39	0.52 ± 0.40
	Five-Distractor	0.86 ± 0.35	1.00 ± 0.00	0.75 ± 0.32	0.88 ± 0.13
Pickup-v0	Zero-shot	0.20 ± 0.40	0.32 ± 0.47	0.15 ± 0.33	0.21 ± 0.34
	Five-Distractor	0.86 ± 0.35	0.80 ± 0.40	0.74 ± 0.32	0.68 ± 0.37
GoToTest-v0	Zero-shot	0.41 ± 0.50	0.90 ± 0.30	0.34 ± 0.42	0.63 ± 0.31
	Five-Distractor	0.88 ± 0.33	1.00 ± 0.00	0.79 ± 0.30	0.87 ± 0.12
PickupTest-v0	Zero-shot	0.22 ± 0.42	0.30 ± 0.46	0.17 ± 0.33	0.22 ± 0.36
	Five-Distractor	0.82 ± 0.39	0.80 ± 0.40	0.72 ± 0.35	0.70 ± 0.37

Table 4: Summary statistics of zero-shot model and trained models on environments with 5 distractors

We also include summary bar charts that resembles the evaluation metrics that can be found in Tables 3, 1 and 4 for easy of readability and interpretability.

The plots for evaluation on environments with no distractors are as follows:

Figure 14: Bar chart comparing zero-shot and first trained model performance in zero-distractor environments

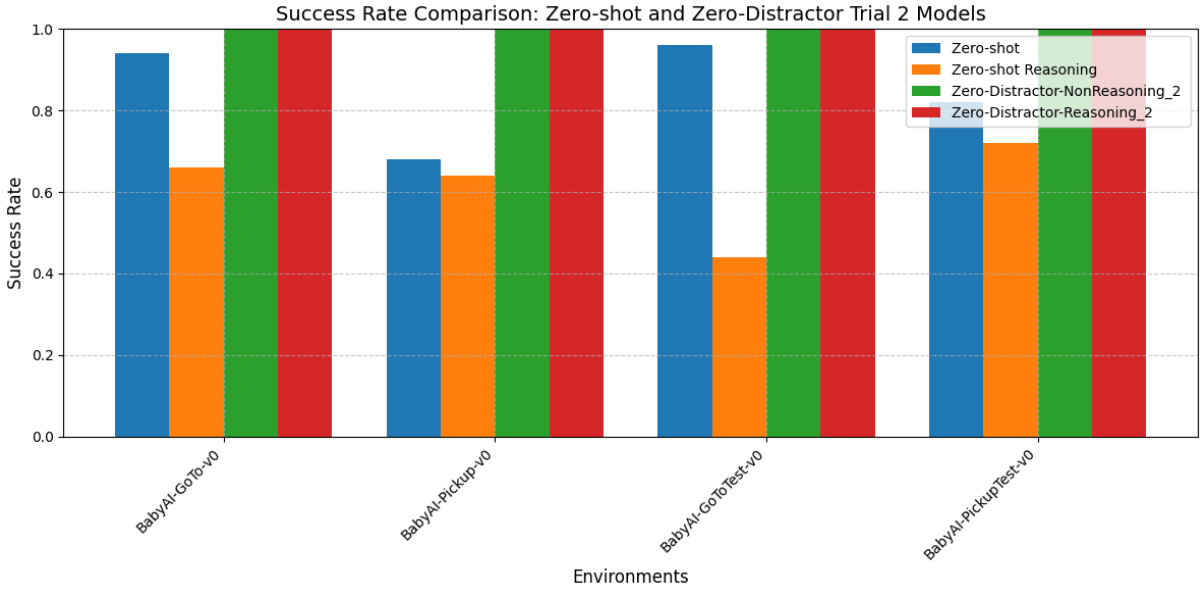

Figure 15: Bar chart comparing zero-shot and second trained model performance in zero-distractor environments

The plots for evaluation on environments with three distractors are as follows:

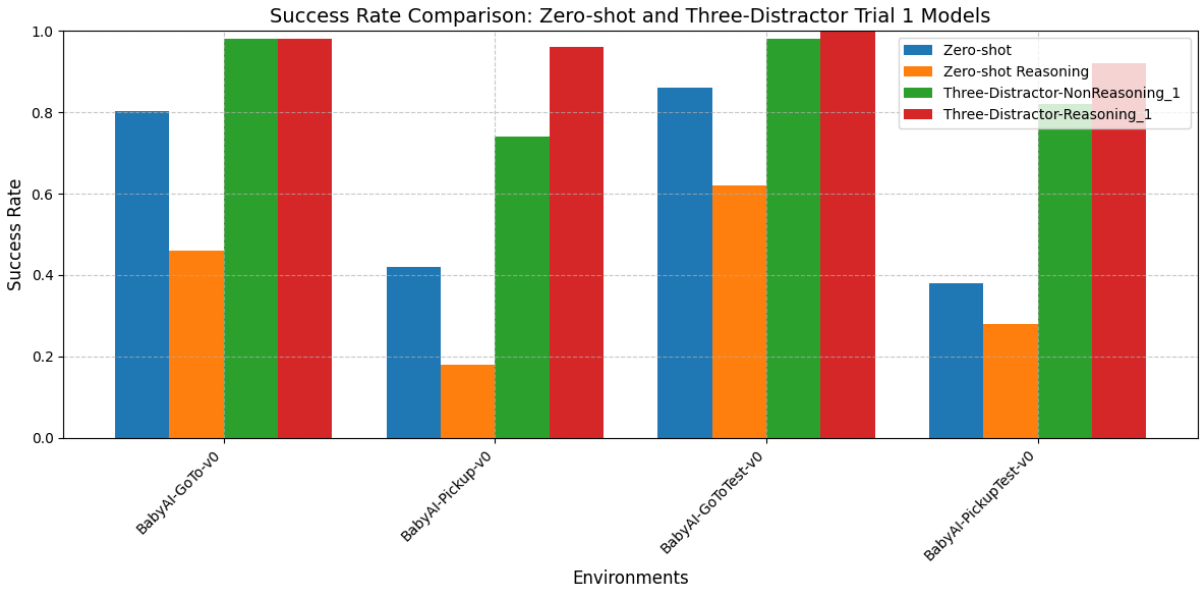

Figure 16: Bar chart comparing zero-shot and first trained model performance in three-distractor environments

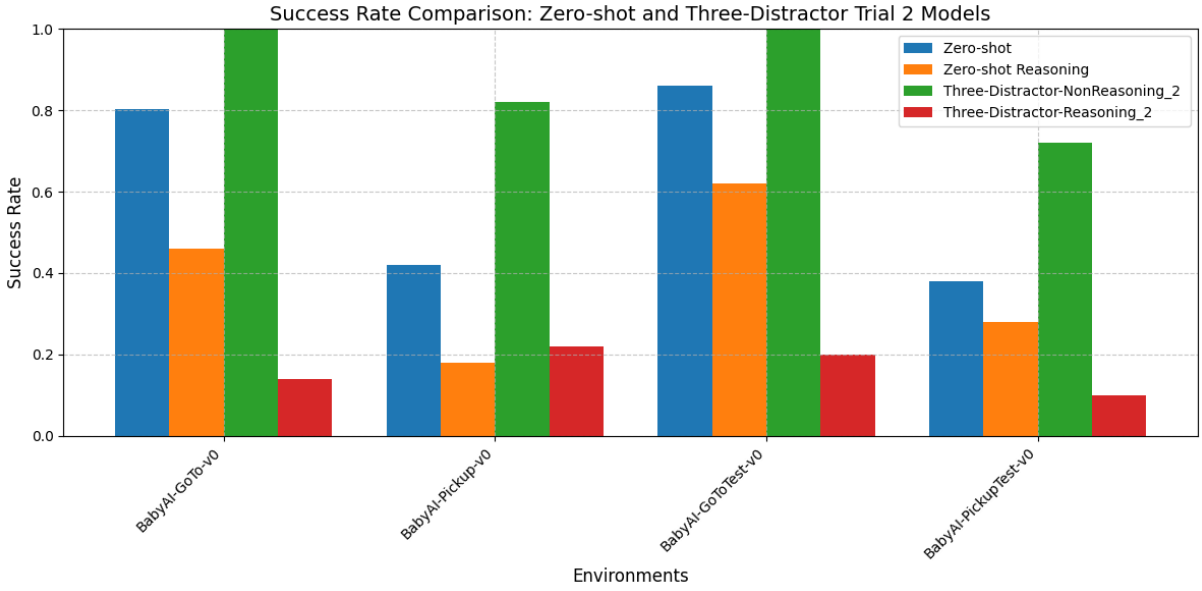

Figure 17: Bar chart comparing zero-shot and second trained model performance in three-distractor environments

The plot for evaluation on environments with five distractors is as follows:

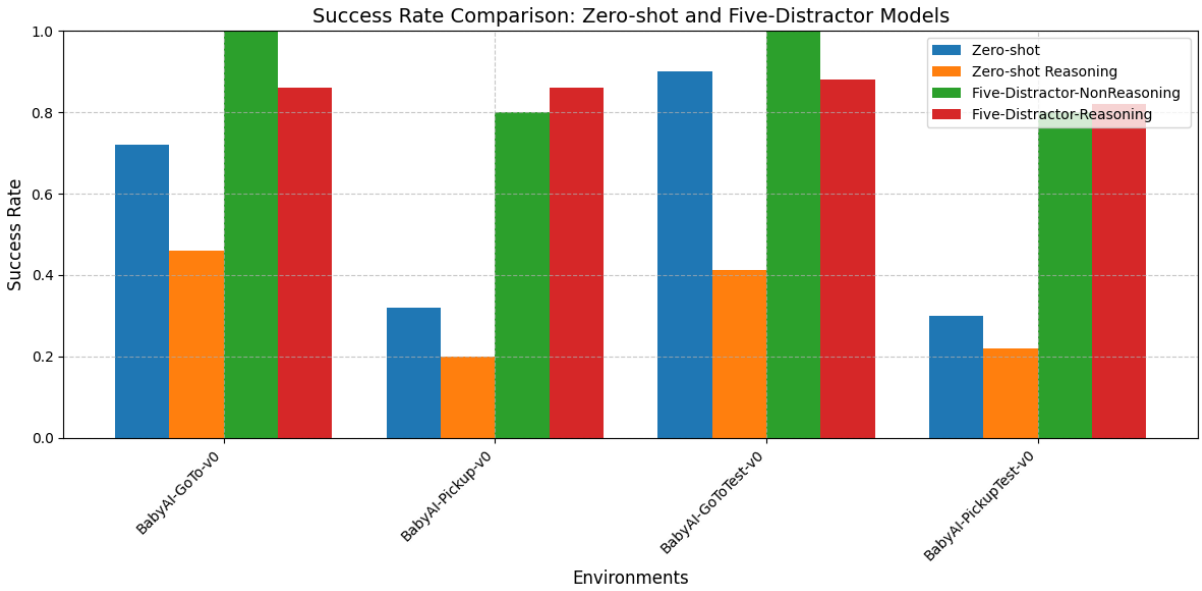

Figure 18: Bar chart comparing zero-shot and trained model performance in five-distractor environments

J Model Performance on General Metrics

We evaluated model performance on two benchmarks unrelated to the BabyAI-Text environment, ARC-C and a subset of 1000 examples in HellaSwag. We discover that training the model under reasoning modes result in very similar performance to those trained under non-reasoning modes. In the below bar charts, we show performance of the model when trained under 1) no distractors; 2) three distractors and 3) five distractors, under reasoning and non-reasoning mode, respectively.

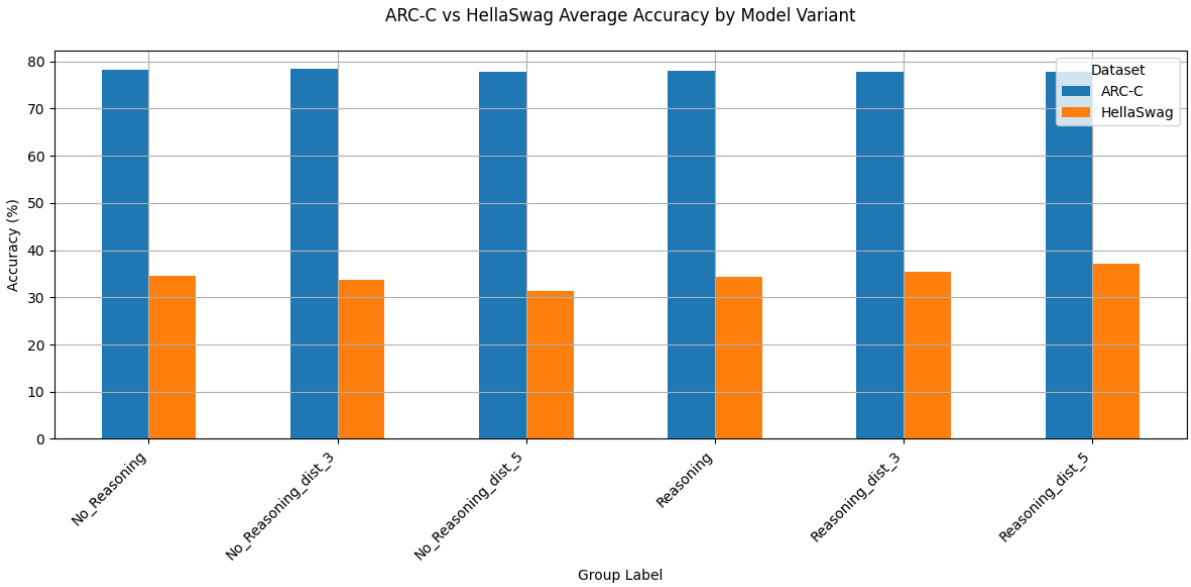

Figure 19: Model performance on general benchmarks