LLMs are Greedy Agents: Effects of RL Fine-tuning on Decision-Making Abilities

Anonymous Author(s)
Affiliation
Address
email

Abstract

The success of LLMs has sparked interest in various agentic applications. A key hypothesis is that LLMs, leveraging common sense and Chain-of-Thought (CoT) reasoning, can effectively explore and efficiently solve complex domains. However, LLM agents have been found to suffer from sub-optimal exploration and the knowing-doing gap, the inability to effectively act on knowledge present in the model. In this work, we systematically study *why* LLMs perform sub-optimally in decision-making scenarios. In particular, we closely examine three prevalent failure modes: greediness, frequency bias, and the knowing-doing gap. We propose mitigation of these shortcomings by fine-tuning via Reinforcement Learning (RL) on self-generated CoT rationales. Our experiments across multi-armed bandits, contextual bandits, and Tic-tac-toe, demonstrate that RL fine-tuning enhances the decision-making abilities of LLMs by increasing exploration and narrowing the knowing-doing gap. Finally, we study both classic exploration mechanisms, such as ϵ-greedy, and LLM-specific approaches, such as self-correction and self-consistency, to enable more effective fine-tuning of LLMs for decision-making.

1 Introduction

Large Language Models (LLMs) pre-trained on massive internet-scale datasets have demonstrated success across diverse domains, including text generation and language understanding [Radford et al., 2019, Brown et al., 2020b, Team et al., 2023b, 2024a, Dubey et al., 2024]. Their broad pre-training distribution, enables generalization to a wide range of scenarios including coding assistance [Li et al., 2022], education [Team et al., 2024d], and medicine [Saab et al., 2024]. Therefore, their success has sparked interest in using LLMs for decision-making problems [Chen et al., 2023, Krishnamurthy et al., 2024, Nie et al., 2024] at the core of agentic AI systems [Durante et al., 2024].

One key hypothesis is that LLMs can generate informed action predictions without extensive environment interaction [Lu et al., 2024] due to "world knowledge" present in the model. Moreover, Chain-of-Thought (CoT) [Wei et al., 2022] equips models with the ability to reason about the observed history and their actions, which facilitates environment interaction. However, these advantages do not seem to materialize into strong performance when LLMs are faced with decision-making scenarios. Notably, Krishnamurthy et al. [2024] and Nie et al. [2024] found that LLMs do not robustly engage in *exploration* resulting in sub-optimal behavior. Similar shortcomings of LLMs have been observed by Paglieri et al. [2024] and Ruoss et al. [2024] on stateful environments commonly used in RL (e.g., grid-worlds, Atari). Both works attribute the shortcomings to the *knowing-doing gap*, which states that models can posses knowledge about a task or can describe the consequences of their behavior (i.e., they know what to do), but cannot materialize this knowledge when acting (i.e., incapable of doing). Consequently, sub-optimal exploration and the knowing-doing gap are considerable obstacles towards more powerful agentic LLMs.

Submitted to the First Exploration in AI Today Workshop at ICML (EXAIT at ICML 2025). Do not distribute.

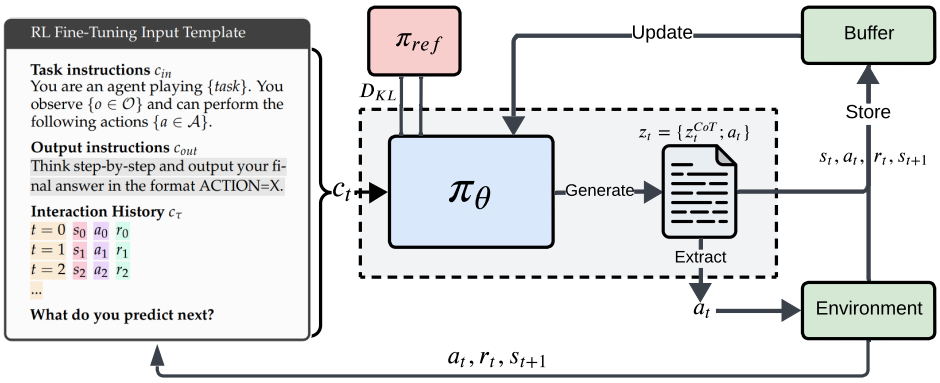


Figure 1: Illustration of our **Reinforcement Learning Fine Tuning (RLFT)** pipeline. We fine-tune a pre-trained LLM π_θ via self-generated Chain-of-Thought (CoT) rationales on environment rewards.

In this work, we aim to understand *why* LLMs often perform sub-optimally in simple decision-making scenarios. In particular, we systematically study three prevalent failure modes in small-to-medium-scale LLMs: greediness, frequency bias, and the knowing-doing gap (see Section 4.2). Our analysis shows that final performance often remains sub-optimal, because LLMs prematurely commit to greedy action selection strategies leading to stagnating action coverage that leave a large part of the action space unexplored (up to 55%). Moreover, we observe that small-scale LLMs (2B) tend to copy the most frequent actions in the context regardless of their respective reward, which we refer to as frequency bias. In contrast, larger LLMs (27B) mostly diminish the frequency bias, yet they remain prone to greedy behavior at the cost of exploration. Similarly, we quantify the knowing-doing gap and find that LLMs often know how to solve a task (87% correct rationales) but fail at acting on this knowledge as they prioritize greedy actions (64% of actions when rationale is correct).

To overcome these shortcomings, we propose Reinforcement Learning Fine-Tuning (RLFT) on self-generated CoT rationales. RL is the pre-dominant learning paradigm in decision-making scenarios and has been successful in game-playing [Silver et al., 2016, Vinyals et al., 2019], robotics [Tirumala et al., 2025], plasma-control [Degrave et al., 2022], or navigating stratospheric balloons [Bellemare et al., 2020]. We study the effects of RLFT on pre-trained Gemma2 models [Team et al., 2024b,c] in three sizes (2B, 9B, and 27B) in multi-arm bandit (MAB) and contextual bandit (CB) settings proposed by Nie et al. [2024], and the textual Tic-tac-toe environment released by Ruoss et al. [2024]. Across environments, we find that RLFT enhances the decision-making abilities of LLMs by increasing exploration and narrowing the knowing-doing gap. While RLFT positively affects exploration of LLM agents, their exploration strategies remain sub-optimal. Therefore, we empirically evaluate both "classic" exploration mechanisms commonly employed in RL, such as ϵ-greedy, and LLM-specific approaches, such as self-correction and self-consistency, to enable more effective fine-tuning for decision-making scenarios. Finally, in our ablations we investigate the importance of CoT reasoning for decision-making, highlight the effectiveness of leveraging expert data, and show the benefits of giving the agent more reasoning tokens to solve the decision-making problem.

In summary, we make the following **contributions**:

- We systematically examine three failure modes of small-to-medium scale LLMs in decision-making scenarios: greediness, frequency bias, and the knowing-doing gap.
- We study how RL fine-tuning on self-generated CoT rationales affects these shortcomings, highlighting positive effects of RLFT on exploration and decision-making abilities.
- We evaluate a variety of exploration mechanisms (e.g., ϵ-greedy) and LLM-specific approaches (e.g., self-consistency), to enable more effective RLFT for LLMs.

2 Related Work

We discuss related works in Appendix A.

3 Methodology

3.1 Background

Reinforcement Learning. We assume the standard RL formulation via a Markov Decision Process (MDP) represented by a tuple of $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$, where $\mathcal{S}$ and $\mathcal{A}$ denote state and action spaces, respectively. At every timestep t the agent observes state $s_t \in \mathcal{S}$, predicts action $a_t \in \mathcal{A}$, and receives a reward r_t given by the reward function $\mathcal{R}(s_t, a_t)$. $\mathcal{P}(s_{t+1} \mid s_t, a_t)$ defines the transition dynamics constituting a probability distribution over next states s_{t+1}. The goal of RL is to learn a policy $\pi_\theta(a_t \mid s_t)$ with parameters θ that predicts an action a_t in state s_t that maximizes cumulative reward.

Reinforcement Learning from Human Feedback. RLHF aims to fine-tune pre-trained models towards human preferences [Christiano et al., 2017]. Preferences are typically encoded via a reward model r_ϕ with parameters ϕ learned from a human annotated dataset $\mathcal{D}$ consisting of query-response pairs x and y, respectively. RLHF optimizes a constrained REINFORCE estimator [Williams, 1992]:

$$\max_\theta \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_\theta(\cdot|x)} \left[(r_\phi(x, y) - b) \nabla_\theta \log \pi_\theta(y \mid x) - \beta D_{KL}(\pi_\theta(\cdot \mid x) \mid\mid \pi_{ref}(\cdot \mid x)) \right] \quad (1)$$

Here π_{ref} is a reference policy, which is typically the frozen pre-trained model, and β is as weighting term. The baseline b represents a baseline to reduce variance and is commonly instantiated by a value function [Schulman et al., 2017, Ouyang et al., 2022] or a Monte-Carlo (MC) estimate of the returns [Ahmadian et al., 2024, Ramesh et al., 2024, Shao et al., 2024].

3.2 Reinforcement Learning Fine-Tuning (RLFT)

Our RLFT approach relies on fine-tuning on self-generated CoT rationales on rewards obtained from environment interaction. During RLFT the model learns to iteratively refine its reasoning process, favoring CoT patterns and actions that lead to higher rewards (see Figure 1). Our approach is similar to Guo et al. [2025] and Zhai et al. [2025], but specialized for decision-making scenarios.

Context Representation. The input tokens to our model at step t consists of input instructions c_t^{in}, output instructions c_t^{out}, and the most recent interaction history $c_t^{\tau_{t-C:t}}$ (see Figure 1). The history representation contains the trajectory $\tau_{t-C:t} = (s_{t-C}, a_{t-C}, r_{t-C}, \ldots, s_t, a_t, r_t)$ of the C most recent states, actions, and rewards. We opt for task-specific instructions for c_t^{in} rather than a generic instruction template, providing the agent with information about the observations, the possible actions, and its objective. Consequently, c_t is represented by the concatenation of the instruction and history tokens $c_t = [c_t^{in}; c_t^{out}; c_t^{\tau_{t-C:t}}]$.

Factorization of Action Tokens. At every interaction step t, the agent generates action tokens $z_t = [z_t^{CoT}; a_t]$ containing both the CoT reasoning tokens z_t^{CoT} and the action to be executed in the environment a_t. To extract a_t from z_t, we make use of an extraction function $a_t = g(z_t)$. In practice, g consists of regular expressions to match the output pattern given by c_t^{out}. If no valid action is found a random action is executed. To allow for flexibility in refining the reasoning process, we opt for a permissive output template (i.e., `ACTION=X`), rather than enforcing a structured output template (e.g., `<thought>` and `<action>` blocks). We employ a token generation budget of G tokens ($G = 256$ by default), therefore $|z_t| \leq G$.

Reward Shaping for Valid Actions. In addition to the environment reward r_t^{env}, we employ a reward shaping term r_t^{valid} to encourage the model to adhere to the output template, $r_t = r_t^{env} + r_t^{valid}$. More specifically, we make use of a reward penalty of -5 if g cannot extract a valid action, $r_t^{valid} = -5 \cdot 1(g(a_t^{act}) \notin \mathcal{A})$. To ensure that the reward penalty does not overly bias optimization, we employ reward normalization to the environment rewards.

Fine-tuning objective. We fine-tune using the clipping objective introduced by Schulman et al. [2017] with and additional KL constraint to the reference policy π_{ref}:

$$\max_\theta \mathbb{E}_{(c,z)\sim\mathcal{D}} \left[\min \left(\frac{\pi_\theta(z|c)}{\pi_{\theta_{old}}(z|c)} A_{adv}, \text{clip}_\epsilon \left(\frac{\pi_\theta(z|c)}{\pi_{\theta_{old}}(z|c)} \right) A_{adv} \right) - \beta D_{KL}(\pi_\theta(\cdot|c) || \pi_{ref}(\cdot|c)) \right] \quad (2)$$

Here $\pi_{\theta_{old}}$ refers to the rollout generating policy, D is the rollout buffer, and ϵ is a hyperparameter. To allow for memory efficient fine-tuning in environments with fixed episode lengths (bandits), we make use of a Monte Carlo baseline to estimate A_{adv}. Instead of exploiting multiple rollouts, as used by

Ahmadian et al. [2024] and Ramesh et al. [2024], we compute rewards-to-go. For environments with variable episode lengths (Tic-tac-toe), we learn a separate state-value head on top of the last layer LLM representations and make use of generalized advantage estimation [Schulman et al., 2015].

We provide additional implementation and training details in Appendix C.

4 Experiments

We study the effect of fine-tuning Gemma2 [Team et al., 2024b,c] models in MAB and CB settings proposed by Nie et al. [2024], and on a text-based version of Tic-tac-toe released by Paglieri et al. [2024]. We describe our environments and baselines in Section 4.1. For our experiments, we compare Gemma2 [Team et al., 2024c] models at three model scales: 2B, 9B and 27B. In Section 4.2, we first analyze three common failure modes of LLM agents in MAB scenarios: **(1)** greediness, **(2)** frequency bias, and **(3)** the knowing-doing gap. Then we investigate the effects of fine-tuning on self-generated CoT rationales or expert rationales in MABs and CBs (see Section 4.3), and in Tic-tac-toe (see Section 4.5). In Section 4.4, we study the effects of a exploration mechanisms on the fine-tuning performance. Finally, in Section 4.5 we empirically examine important components of our approach.

4.1 Environments & Baselines

Multi-armed and Contextual Bandits. MABs [Slivkins et al., 2019, Lattimore and Szepesvári, 2020] are a classic problem setting in RL that isolates the *exploration-exploitation* trade-off. For our MAB experiments, we leverage the text-based bandit scenarios released by Nie et al. [2024]. We focus on the *continuous* and *button* variants, as illustrated in Figure 2. We report results for MAB with $k \in \{5, 10, 20\}$ arms ($|\mathcal{A}| = k$) and payoffs of the arms being either Gaussian or Bernoulli distributed. In addition, we consider three levels of stochasticity (low/medium/high) that determine the standard deviation or delta gap in Gaussian or Bernoulli bandits, respectively. For all MAB settings, we limit the horizon T to 50 interaction steps. We compare against two commonly used baselines for MABs: Upper-confidence Bound (UCB) [Auer, 2002] and a random agent that selects actions uniformly at random. UCB is considered optimal and represents the upper-bound for agent performance, whereas the random baseline represents the lower bound. We provide more details on our MAB and CB setups in Appendices B.1 and B.2, respectively.

Tic-tac-toe. In addition, we use the text-based Tic-tac-toe environment released by Ruoss et al. [2024], which exhibits proper state transitions. Ruoss et al. [2024] demonstrated that frontier models struggle to achieve strong performance in this environment and barely beat a random opponent. Consequently, it is a good target to investigate the efficacy of RLFT. In Appendix B.3, we provide addition details on our environment and training setup.

Button Multi-armed Bandit (Gaussian)

You are a bandit algorithm in a room with 5 buttons labeled red, green, blue, yellow, orange. [...]. Your goal is to maximize the total reward. [More instructions]

Think step-by-step and output your final answer in the format ACTION=X where X is one of the arms listed above. IMPORTANT: Provide your (SHORT!) thinking process and your answer ACTION=X

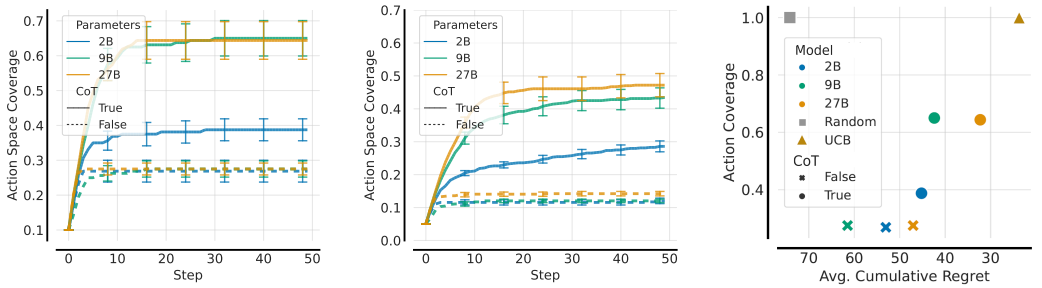
So far you have tried/seen:
Step=0 Action=green Reward=0.3
Step=1 Action=blue Reward=0.1
Step=2 Action=orange Reward=-0.5
Step=3 Action=red Reward=0.5
Step=4 Action=green Reward=0.24
...
What do you predict next?

Figure 2: Illustration of a Gaussian MAB for the *button* scenario from [Nie et al., 2024] using our context representation and instructions.

4.2 Why do LLMs perform suboptimally in decision-making?

Prior works found that LLM agents perform suboptimally and fail to explore sufficiently in interactive settings [Paglieri et al., 2024, Ruoss et al., 2024]. Therefore, we first examine *why* models perform suboptimally and identify three prevalent failure modes: **(1)** greediness, **(2)** frequency bias, and **(3)** the knowing-doing gap. In this section, we present analyses of Gemma2 models when given input

contexts that elucidate the failure modes. We conduct our analyses on the *button* instance of our MAB experiments at three model scales, and find that the failure modes persist across model scales (see Appendix D.1 for *continuous* instance).



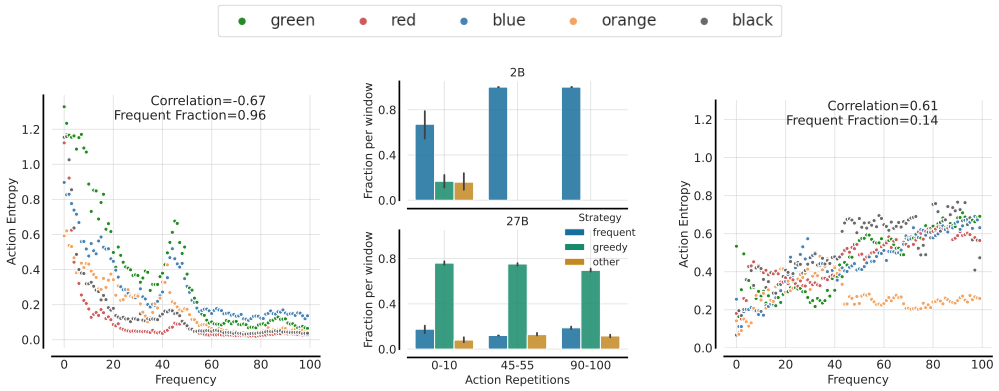
(a) Action Coverage: 10 arms (b) Action Coverage: 20 arms (c) Coverage vs. Regret: 10 arms

Figure 3: Illustration of **Greediness**. We show action coverage for Gemma2 2B/9B/27B w/ and w/o CoT for **(a)** 10 and **(b)** 20 arms over 50 interaction steps. Agents favor the best performing action among the set of selected actions, leading to stagnating action coverage, despite benefits of larger models and CoT. In **(c)**, we plot cumulative regret against action coverage. The agents exhibit suboptimal regret, because of greedy action selection strategies.

Greediness. The first and most pervasive failure mode is *greediness*, which is characterized by the LLM overly favoring the best performing action among a small set of actions seen so far. To illustrate this failure mode, we show the average action coverage achieved by Gemma2 2B/9B/27B with and without CoT across 64 MABs with 10 and 20 arms over 50 interaction steps (see Figure 3 a and b). We define action coverage C_t at step t as the fraction of available actions that have been selected at least once, $C_t = \frac{\{a \in \mathcal{A}: N_t(a) > 0\}}{|\mathcal{A}|}$ with $N_t(a)$ representing the number of times action $a \in \mathcal{A}$ has been selected until t. For 10 arms and averaged over 64 parallel environments, we find that Gemma2 2B covers 40% of all actions, while 9B/27B cover 65% (i.e., 6.5 actions), leaving a significant part of the action space unexplored. Note that without CoT all models explore merely 25% of all actions in the 10 arms setting. The suboptimal coverage is caused by the model overly favoring high-reward actions (see Figure 12 in Appendix D.1.1). Consequently, the model prematurely commits to a greedy strategy leading to a stagnating action coverage beyond 10 steps. Increasing the number of arms makes the greediness even more apparent, with the largest models only covering 45% of all actions. Due to this, the regret remains high compared to UCB, even though the models improve significantly over a random agent (see Figure 3c).

Frequency Bias. The next prevalent failure mode is *frequency bias*, which is characterized by repeatedly selecting the most frequently occurring action in the context, even when that action gives low reward. To understand how the model's behavior is influenced by the frequency of actions, we construct prefix histories using a random policy, vary the number of repetitions of the last action in the context history (0 to 100) and record the entropy over all actions (see Figure 4a and c). We provide details on the context generation in Appendix D.1.2. To quantify frequency bias, we categorize an action as *frequent* action $a_f = \arg\max_{a \in A} N_T(a)$, *greedy* $a_g = \arg\max_{a \in \{a \in A: N_T(a) > 0\}} R_T(a)$, or *other* if they are neither frequent nor greedy. Note that action is optimal with 10% probability. Subsequently, we compute the frequent F_f, greedy F_g and other F_o fractions as reported in Figure 4 (see Appendix 4 for definitions).

Gemma2 2B heavily suffers from repeated actions, exhibiting a decreasing entropy with increasing repetitions (96% F_f, see Figure 4a). In contrast, 27B escapes the frequency bias (14%, see Figure 4c) and interestingly becomes less certain of its action prediction with increasing repetitions. To examine this further, we show the bucketized fractions with 0-10, 45-55 and 90-100 repetitions for 2B and 27B in Figure 4b. Indeed, for 2B F_f keeps increasing with increasing repetitions. While 27B escapes the frequency bias it suffers heavily from greediness. Similar biases have been identified in Behavior Cloning (BC) settings and termed *copycat* bias [Wen et al., 2020, Schmied et al., 2024b]. This suggests that frequency bias is an artifact of supervised pre-training, and motivates the use of RL as a counter-measurement.



(a) 2B: Correlations (b) Strategy per Window (c) 27B: Correlations

Figure 4: Illustration of **Frequency Bias**. We plot the frequency of the repeated action in the context against the action entropy across all actions for 10 armed MABs. **(a)** Gemma2 2B heavily suffers from frequency bias, becoming more certain of the most frequent action, the more often it occurs in the context. **(c)** Gemma2 27B overcomes the frequency bias, but instead behaves greedily. In **(b)** we show the action strategies for three repetition windows.

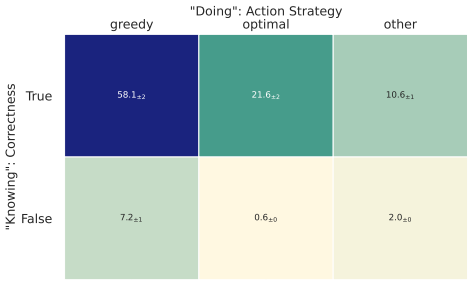


Figure 5: Confusion matrix for the **Knowing-Doing Gap** of Gemma2 27B. The agent "knows" how to solve the task (87% correct rationales, sum of top row), but fails at "doing" (58% greedy actions among correct rationales). See Figure 21, for instructions and an agent response.

Knowing-Doing Gap. The *knowing-doing gap* has been observed by Paglieri et al. [2024] and Ruoss et al. [2024]. To illustrate the gap in our setting, we first task Gemma2 27B to produce the UCB algorithm, to compute the relevant quantities accordingly ("knowing"), and finally to act according to the computed quantities ("doing", see Figure 21 for the instructions and an agent response). We let Gemma2 27B interact with the environment (64 instances) for 50 timesteps with $G = 2048$ per step, and extract the UCB quantities from the rationales. To quantify "knowing", we compare the UCB values computed by the model against the real UCB values, and consider the rationale z_{CoT} as correct if the arm with the highest UCB values match (see Appendix D.1.3 for details). To quantify "doing", we categorize the generated actions as *optimal* action if the model selects the action with the highest UCB value, as *greedy* if it selects the action with the highest UCB value among the set of actions tried so far, and as *other* if the action is neither optimal nor greedy. Subsequently, we compute the percentages of greedy/optimal/other actions. The agent clearly knows how to solve the task, with 87% of all rationales being correct (see Figure 5). However, even for correctly computed rationales, the model often selects the greedy action (58%) over the optimal action (21%). This discrepancy highlights the shortcomings of the LLM when it comes to "acting" even when "knowing" the algorithm.

4.3 Effectiveness of RL Fine-Tuning

Next, we study the effects of RLFT on cumulative regret (w.r.t. optimal policy) and whether it alleviates the highlighted failure modes. We fine-tune Gemma2 2B and 9B on self-generated CoT rationales for 30K updates with an (accumulated) batch size of 128. To avoid memorization of reward distributions, we maintain a pool of 512 MABs and randomly select a subset of 16 MABs per rollout. We refer to Appendix C for training details and hyperparameters.

RLFT lowers regret. In Figure 6, we report the cumulative regrets across model sizes and arms for a medium noise $\sigma = 1.0$ scenario (see Appendix D.2 for low/high noise). Across environments, the LLMs clearly outperform the random baseline and RLFT lowers regret for both 2B and 9B. For 2B, RLFT narrows the gap to it's larger counterparts and UCB. Similarly, RLFT lowers regret for

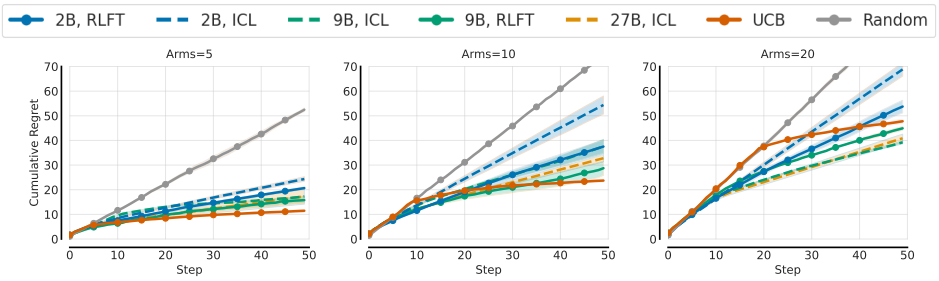


Figure 6: Main Comparison on **Gaussian MABs** button scenario in the medium noise ($\sigma = 1$) setting. We compare cumulative regrets (lower is better) of classic baselines against ICL and RLFT performances for 5, 10, and 20 arms. See Figure 16 for $\sigma = 0.1$ and $\sigma = 3$.

Gemma2 9B. Note, that the lower cumulative regret of Gemma2 9/27B compared to UCB after 50 environment steps in the 20 arms scenario is an artifact of the limited interaction steps, but the trends remain clear. We repeat RLFT for CBs, and observe similar performance improvements for Gemma2 2B (see Appendix D.3). Consequently, reinforcing self-generated CoT rationales towards environment rewards improves performance on simple decision-making scenarios.

RLFT mitigates greediness. In Figure 14a, we report the action coverage for 2B after RLFT at different numbers of gradient steps (10K, 20K, 30K). Indeed, we observe that RLFT results in increased action coverage (+12%) after 30K updates. Interestingly, we first observe a decrease (at 10K) followed by an increase in action coverage (20K, 30K). We observe similar effects for the 20 arms scenario (see Figure 14b). Via RLFT the agent learns to explore and mitigates greediness.

RLFT counteracts frequency bias. We find that RLFT counteracts frequency bias (Figure 15). In particular, for 0-10 repetitions we observe a strong decrease in the fraction of frequent actions ($70\% \rightarrow 35\%$) and increase in "other" actions ($8\% \rightarrow 35\%$). However, F_f remains elevated for high repetitions. Consequently, RLFT counteracts frequency bias, but does not fully alleviate it.

4.4 Effect of Exploration Mechanisms

For RLFT, we relied solely on the exploration properties for CoT reasoning. However, it is common practice to employ additional exploration strategies [Mnih et al., 2015, Schulman et al., 2017, Haarnoja et al., 2018]. Therefore, we study the effects of classic exploration mechanisms and LLM-specific strategies to encourage exploration (see Appendix C.4). Across model scales, we observe that the mechanisms result in varied effects on action coverage (see Figure 18). First, we find that the simple *try-all* strategy, which reduces the need for additional exploration by trying all actions, results in the biggest performance improvements. Second, a simple *exploration bonus* (+1 reward for untried actions during RLFT), significantly increases exploration ($50\% \rightarrow 70\%$) and lowers regret towards the expert compared to regular RLFT. This highlights the importance of reward shaping for fine-tuning LLMs to elucidate a desired behavior.

4.5 Ablations

We conduct additional ablations on the effect of RLFT in Tic-tac-toe, the importance of CoT for RLFT, leveraging pre-collected datasets via SFT, and the effect of "thinking" tokens in Appendix E.

5 Conclusion

We study *why* LLMs perform sub-optimally in decision-making scenarios and examine three prevalent failure modes: greediness, frequency bias, and the knowing-doing gap. We show that RLFT on CoT rationales mitigates greediness, counteracts frequency bias, and improves final performance. While RLFT improves the exploration, it remains sub-optimal compared to bandit algorithms. Therefore, we investigate a variety of mechanisms to improve exploration. Models act near-optimally if provided with sufficient information underscoring their shortcomings in exploration. Finally, we highlight the importance of reward shaping for RLFT. In Appendix F, we discuss limitations and future work.

References

R. Agarwal, M. Schwarzer, P. S. Castro, A. C. Courville, and M. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

A. Ahmadian, C. Cremer, M. Gallé, M. Fadaee, J. Kreutzer, A. Üstün, and S. Hooker. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms. *arXiv preprint arXiv:2402.14740*, 2024.

J. A. Arjona-Medina, M. Gillhofer, M. Widrich, T. Unterthiner, J. Brandstetter, and S. Hochreiter. Rudder: Return decomposition for delayed rewards. *Advances in Neural Information Processing Systems*, 32, 2019.

P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

M. Beck, K. Pöppel, M. Spanring, A. Auer, O. Prudnikova, M. Kopp, G. Klambauer, J. Brandstetter, and S. Hochreiter. xlstm: Extended long short-term memory. *Advances in Neural Information Processing Systems*, 37:107547–107603, 2025.

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

M. G. Bellemare, S. Candido, P. S. Castro, J. Gong, M. C. Machado, S. Moitra, S. S. Ponda, and Z. Wang. Autonomous navigation of stratospheric balloons using reinforcement learning. *Nature*, 588(7836):77–82, 2020.

A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020a.

T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020b.

Y. Burda, H. Edwards, A. Storkey, and O. Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

S. Chan, A. Santoro, A. K. Lampinen, J. Wang, A. Singh, P. H. Richemond, J. L. McClelland, and F. Hill. Data distributional properties drive emergent in-context learning in transformers. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

L. Chen, L. Wang, H. Dong, Y. Du, J. Yan, F. Yang, S. Li, P. Zhao, S. Qin, S. Rajmohan, et al. Introspective tips: Large language model for in-context decision making. *arXiv preprint arXiv:2305.11598*, 2023.

P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

W. Chu, L. Li, L. Reyzin, and R. Schapire. Contextual bandits with linear payoff functions. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 208–214. JMLR Workshop and Conference Proceedings, 2011.

K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

R. Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pages 72–83. Springer, 2006.

C. Cundy and S. Ermon. Sequencematch: Imitation learning for autoregressive sequence modelling with backtracking. *arXiv preprint arXiv:2306.05426*, 2023.

S. De, S. L. Smith, A. Fernando, A. Botev, G. Cristian-Muraru, A. Gu, R. Haroun, L. Berrada, Y. Chen, S. Srinivasan, et al. Griffin: Mixing gated linear recurrences with local attention for efficient language models. *arXiv preprint arXiv:2402.19427*, 2024.

J. Degrave, F. Felici, J. Buchli, M. Neunert, B. Tracey, F. Carpanese, T. Ewalds, R. Hafner, A. Abdolmaleki, D. de Las Casas, et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419, 2022.

Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel. Rl2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Z. Durante, Q. Huang, N. Wake, R. Gong, J. S. Park, B. Sarkar, R. Taori, Y. Noda, D. Terzopoulos, Y. Choi, et al. Agent ai: Surveying the horizons of multimodal interaction. *arXiv preprint arXiv:2401.03568*, 2024.

A. Ecoffet, J. Huizinga, J. Lehman, K. O. Stanley, and J. Clune. Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995*, 2019.

C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

S. Flennerhag, A. A. Rusu, R. Pascanu, F. Visin, H. Yin, and R. Hadsell. Meta-learning with warped gradient descent. *arXiv preprint arXiv:1909.00025*, 2019.

O. Groth, M. Wulfmeier, G. Vezzani, V. Dasagi, T. Hertweck, R. Hafner, N. Heess, and M. Riedmiller. Is curiosity all you need? on the utility of emergent behaviours from curious exploration. *arXiv e-prints*, pages arXiv–2109, 2021.

D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

D. Hafner. Benchmarking the spectrum of agent capabilities. *arXiv preprint arXiv:2109.06780*, 2021.

F. M. Harper and J. A. Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.

H. He, W. Yao, K. Ma, W. Yu, Y. Dai, H. Zhang, Z. Lan, and D. Yu. Webvoyager: Building an end-to-end web agent with large multimodal models. *arXiv preprint arXiv:2401.13919*, 2024.

D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

S. Hochreiter, A. S. Younger, and P. R. Conwell. Learning to learn using gradient descent. In *Artificial Neural Networks—ICANN 2001: International Conference Vienna, Austria, August 21–25, 2001 Proceedings 11*, pages 87–94. Springer, 2001.

E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

S. Hu and J. Clune. Thought cloning: Learning to think while acting by imitating human thinking. *Advances in Neural Information Processing Systems*, 36:44451–44469, 2023.

L. Kirsch, S. van Steenkiste, and J. Schmidhuber. Improving generalization in meta reinforcement learning using learned objectives. *arXiv preprint arXiv:1910.04098*, 2019.

L. Kirsch, J. Harrison, J. Sohl-Dickstein, and L. Metz. General-purpose in-context learning by meta-learning transformers. *arXiv preprint arXiv:2212.04458*, 2022.

L. Kirsch, J. Harrison, C. Freeman, J. Sohl-Dickstein, and J. Schmidhuber. Towards general-purpose in-context learning agents. In *NeurIPS 2023 Workshop on Generalization in Planning*, 2023.

M. Klissarov, P. D'Oro, S. Sodhani, R. Raileanu, P.-L. Bacon, P. Vincent, A. Zhang, and M. Henaff. Motif: Intrinsic motivation from artificial intelligence feedback. *arXiv preprint arXiv:2310.00166*, 2023.

M. Klissarov, M. Henaff, R. Raileanu, S. Sodhani, P. Vincent, A. Zhang, P.-L. Bacon, D. Precup, M. C. Machado, and P. D'Oro. Maestromotif: Skill design from artificial intelligence feedback. *arXiv preprint arXiv:2412.08542*, 2024.

A. Krishnamurthy, K. Harris, D. J. Foster, C. Zhang, and A. Slivkins. Can large language models explore in-context? *arXiv preprint arXiv:2403.15371*, 2024.

A. Kumar, V. Zhuang, R. Agarwal, Y. Su, J. D. Co-Reyes, A. Singh, K. Baumli, S. Iqbal, C. Bishop, R. Roelofs, et al. Training language models to self-correct via reinforcement learning. *arXiv preprint arXiv:2409.12917*, 2024.

M. Laskin, L. Wang, J. Oh, E. Parisotto, S. Spencer, R. Steigerwald, D. Strouse, S. Hansen, A. Filos, E. Brooks, et al. In-context reinforcement learning with algorithm distillation. *arXiv preprint arXiv:2210.14215*, 2022.

T. Lattimore and C. Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

K.-H. Lee, O. Nachum, M. Yang, L. Lee, D. Freeman, W. Xu, S. Guadarrama, I. Fischer, E. Jang, H. Michalewski, et al. Multi-game decision transformers. *arXiv preprint arXiv:2205.15241*, 2022.

Y. Li, D. Choi, J. Chung, N. Kushman, J. Schrittwieser, R. Leblond, T. Eccles, J. Keeling, F. Gimeno, A. Dal Lago, et al. Competition-level code generation with alphacode. *Science*, 378(6624): 1092–1097, 2022.

C. Lu, S. Hu, and J. Clune. Intelligent go-explore: Standing on the shoulders of giant foundation models. *arXiv preprint arXiv:2405.15143*, 2024.

G. Mialon, C. Fourrier, T. Wolf, Y. LeCun, and T. Scialom. Gaia: a benchmark for general ai assistants. In *The Twelfth International Conference on Learning Representations*, 2023.

S. Mirchandani, F. Xia, P. Florence, B. Ichter, D. Driess, M. G. Arenas, K. Rao, D. Sadigh, and A. Zeng. Large language models as general pattern machines. *arXiv preprint arXiv:2307.04721*, 2023.

N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel. A simple neural attentive meta-learner. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=B1DmUzWAW.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

G. Monea, A. Bosselut, K. Brantley, and Y. Artzi. Llms are in-context reinforcement learners. *arXiv preprint arXiv:2410.05362*, 2024.

N. Muennighoff, Z. Yang, W. Shi, X. L. Li, L. Fei-Fei, H. Hajishirzi, L. Zettlemoyer, P. Liang, E. Candès, and T. Hashimoto. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025.

A. Nie, Y. Su, B. Chang, J. N. Lee, E. H. Chi, Q. V. Le, and M. Chen. Evolve: Evaluating and optimizing llms for exploration. *arXiv preprint arXiv:2410.06238*, 2024.

P.-Y. Oudeyer, F. Kaplan, and V. V. Hafner. Intrinsic motivation systems for autonomous mental development. *IEEE transactions on evolutionary computation*, 11(2):265–286, 2007.

L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

D. Paglieri, B. Cupiał, S. Coward, U. Piterbarg, M. Wolczyk, A. Khan, E. Pignatelli, Ł. Kuciński, L. Pinto, R. Fergus, et al. Balrog: Benchmarking agentic llm and vlm reasoning on games. *arXiv preprint arXiv:2411.13543*, 2024.

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

D. A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

R. Raileanu and T. Rocktäschel. Ride: Rewarding impact-driven exploration for procedurally-generated environments. *arXiv preprint arXiv:2002.12292*, 2020.

S. S. Ramesh, Y. Hu, I. Chaimalas, V. Mehta, P. G. Sessa, H. B. Ammar, and I. Bogunovic. Group robust preference optimization in reward-free rlhf. *arXiv preprint arXiv:2405.20304*, 2024.

D. Rao, F. Sadeghi, L. Hasenclever, M. Wulfmeier, M. Zambelli, G. Vezzani, D. Tirumala, Y. Aytar, J. Merel, N. Heess, et al. Learning transferable motor skills with hierarchical latent mixture policies. In *International Conference on Learning Representations*, 2021.

S. C. Raparthy, E. Hambro, R. Kirk, M. Henaff, and R. Raileanu. Generalization to new sequential decision making tasks with in-context learning, 2023.

S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-Maron, M. Gimenez, Y. Sulsky, J. Kay, J. T. Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

A. Ruoss, F. Pardo, H. Chan, B. Li, V. Mnih, and T. Genewein. Lmact: A benchmark for in-context imitation learning with long multimodal demonstrations. *arXiv preprint arXiv:2412.01441*, 2024.

K. Saab, T. Tu, W.-H. Weng, R. Tanno, D. Stutz, E. Wulczyn, F. Zhang, T. Strother, C. Park, E. Vedadi, et al. Capabilities of gemini models in medicine. *arXiv preprint arXiv:2404.18416*, 2024.

A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pages 1842–1850. PMLR, 2016.

J. Schmidhuber. Evolutionary principles in self-referential learning. on learning now to learn: The meta-meta-meta...-hook. Diploma thesis, Technische Universitat Munchen, Germany, 14 May 1987.

J. Schmidhuber. Curious model-building control systems. In *Proc. international joint conference on neural networks*, pages 1458–1463, 1991a.

J. Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, pages 222–227, 1991b.

T. Schmied, M. Hofmarcher, F. Paischer, R. Pascanu, and S. Hochreiter. Learning to modulate pre-trained models in rl. *Advances in Neural Information Processing Systems*, 36:38231–38265, 2023.

T. Schmied, T. Adler, V. Patil, M. Beck, K. Pöppel, J. Brandstetter, G. Klambauer, R. Pascanu, and S. Hochreiter. A large recurrent action model: xlstm enables fast inference for robotics tasks. *arXiv preprint arXiv:2410.22391*, 2024a.

T. Schmied, F. Paischer, V. Patil, M. Hofmarcher, R. Pascanu, and S. Hochreiter. Retrieval-augmented decision transformer: External memory for in-context rl. *arXiv preprint arXiv:2410.07071*, 2024b.

J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. Li, Y. Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

N. Shazeer and M. Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pages 4596–4604. PMLR, 2018.

N. Shinn, F. Cassano, B. Labash, A. Gopinath, K. Narasimhan, and S. Yao. Reflexion: Language agents with verbal reinforcement learning.(2023). *arXiv preprint cs.AI/2303.11366*, 2023.

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

A. Slivkins et al. Introduction to multi-armed bandits. *Foundations and Trends® in Machine Learning*, 12(1-2):1–286, 2019.

S. Still and D. Precup. An information-theoretic approach to curiosity-driven reinforcement learning. *Theory in Biosciences*, 131(3):139–148, 2012.

Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. d. L. Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

A. A. Team, J. Bauer, K. Baumli, S. Baveja, F. M. P. Behbahani, A. Bhoopchand, N. Bradley-Schmieg, M. Chang, N. Clay, A. Collister, V. Dasagi, L. Gonzalez, K. Gregor, E. Hughes, S. Kashem, M. Loks-Thompson, H. Openshaw, J. Parker-Holder, S. Pathak, N. P. Nieves, N. Rakicevic, T. Rocktäschel, Y. Schroecker, J. Sygnowski, K. Tuyls, S. York, A. Zacherl, and L. M. Zhang. Human-timescale adaptation in an open-ended task space. In *International Conference on Machine Learning*, 2023a.

G. Team, R. Anil, S. Borgeaud, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth, K. Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023b.

G. Team, P. Georgiev, V. I. Lei, R. Burnell, L. Bai, A. Gulati, G. Tanzer, D. Vincent, Z. Pan, S. Wang, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024a.

G. Team, T. Mesnard, C. Hardin, R. Dadashi, S. Bhupatiraju, S. Pathak, L. Sifre, M. Rivière, M. S. Kale, J. Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024b.

G. Team, M. Riviere, S. Pathak, P. G. Sessa, C. Hardin, S. Bhupatiraju, L. Hussenot, T. Mesnard, B. Shahriari, A. Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024c.

L. Team, A. Modi, A. S. Veerubhotla, A. Rysbek, A. Huber, B. Wiltshire, B. Veprek, D. Gillick, D. Kasenberg, D. Ahmed, et al. Learnlm: Improving gemini for learning. *arXiv preprint arXiv:2412.16429*, 2024d.

D. Tirumala, M. Wulfmeier, B. Moran, S. Huang, J. Humplik, G. Lever, T. Haarnoja, L. Hasenclever, A. Byravan, N. Batchelor, N. sreendra, K. Patel, M. Gwira, F. Nori, M. Riedmiller, and N. Heess. Learning robot soccer from egocentric vision with deep reinforcement learning. In P. Agrawal, O. Kroemer, and W. Burgard, editors, *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 165–184. PMLR, 06–09 Nov 2025. URL https://proceedings.mlr.press/v270/tirumala25a.html.

O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *nature*, 575(7782):350–354, 2019.

J. X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Z. Leibo, R. Munos, C. Blundell, D. Kumaran, and M. Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Y. Wang, X. Yue, and W. Chen. Critique fine-tuning: Learning to critique is more effective than learning to imitate. *arXiv preprint arXiv:2501.17703*, 2025.

J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

S. Welleck, X. Lu, P. West, F. Brahman, T. Shen, D. Khashabi, and Y. Choi. Generating sequences by learning to self-correct. *arXiv preprint arXiv:2211.00053*, 2022.

C. Wen, J. Lin, T. Darrell, D. Jayaraman, and Y. Gao. Fighting copycat agents in behavioral cloning from observation histories. *Advances in Neural Information Processing Systems*, 33:2564–2575, 2020.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

M. Wulfmeier, M. Bloesch, N. Vieillard, A. Ahuja, J. Bornschein, S. Huang, A. Sokolov, M. Barnes, G. Desjardins, A. Bewley, S. M. E. Bechtle, J. T. Springenberg, N. Momchev, O. Bachem, M. Geist, and M. Riedmiller. Imitating language via scalable inverse reinforcement learning. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 90714–90735. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/a5036c166e44b731f214f41813364d01-Paper-Conference.pdf.

S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

E. Zelikman, Y. Wu, J. Mu, and N. Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

E. Zelikman, G. Harik, Y. Shao, V. Jayasiri, N. Haber, and N. D. Goodman. Quiet-star: Language models can teach themselves to think before speaking. *arXiv preprint arXiv:2403.09629*, 2024.

S. Zhai, H. Bai, Z. Lin, J. Pan, P. Tong, Y. Zhou, A. Suhr, S. Xie, Y. LeCun, Y. Ma, et al. Fine-tuning large vision-language models as decision-making agents via reinforcement learning. *Advances in Neural Information Processing Systems*, 37:110935–110971, 2025.

T. Zhang, A. Madaan, L. Gao, S. Zheng, S. Mishra, Y. Yang, N. Tandon, and U. Alon. In-context principle learning from mistakes. *arXiv preprint arXiv:2402.05403*, 2024.

S. Zhou, F. F. Xu, H. Zhu, X. Zhou, R. Lo, A. Sridhar, X. Cheng, T. Ou, Y. Bisk, D. Fried, et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023.

Contents

A Related Work

Exploration in RL and LLMs. The trade-off between *exploration* and *exploitation* is a long-standing challenge in the field of RL [Schmidhuber, 1991a,b, Still and Precup, 2012, Oudeyer et al., 2007]. Widely used RL agents have often relied on random schemes [Mnih et al., 2015], heuristics such as state-visitation counts [Ecoffet et al., 2019, Raileanu and Rocktäschel, 2020], intrinsic curiosity [Pathak et al., 2017, Burda et al., 2018, Groth et al., 2021], behavior priors [Rao et al., 2021], or maximum entropy regularization [Schulman et al., 2017, Haarnoja et al., 2018]. Naturally, a number of works looked into leveraging LLMs for improving exploration of RL agents either as a source of rewards [Klissarov et al., 2023, Lu et al., 2024] or to orchestrate exploration strategies [Klissarov et al., 2024]. Krishnamurthy et al. [2024] investigate the in-context exploration abilities of LLMs when acting directly as a policy. Similarly, Nie et al. [2024] study the exploration abilities of LLMs when fine-tuned on expert trajectories. In contrast, our work investigates the effects of RLFT on the exploration abilities of LLMs and focuses on *why* models fail.

In-context Learning for Decision-Making. ICL is a form of Meta-learning, also referred to as learning-to-learn [Schmidhuber, 1987]. While meta-learning is targeted via a meta-training phase [Santoro et al., 2016, Mishra et al., 2018, Finn et al., 2017, Wang et al., 2016, Duan et al., 2016,

Kirsch et al., 2019, Flennerhag et al., 2019, Team et al., 2023a], ICL emerges as a result of the pre-training data distribution [Chan et al., 2022, Kirsch et al., 2022]. ICL has been rediscovered in LLMs [Brown et al., 2020a] after initial observations by Hochreiter et al. [2001] in LSTMs [Hochreiter and Schmidhuber, 1997]. Mirchandani et al. [2023] leverage the ICL abilities of LLMs to operate as general pattern machines. A number of works leverage the CoT abilities [Wei et al., 2022] of LLMs in simple text-based scenarios [Shinn et al., 2023, Yao et al., 2022]. Similar in-context abilities have been observed in decision-making with models trained from scratch, albeit in restricted environments [Laskin et al., 2022, Lee et al., 2022, Kirsch et al., 2023, Raparthy et al., 2023, Schmied et al., 2024b,a].

Self-Correction in LLMs. A critical component for LLM agents is the ability to self-correct over previously explored attempts. Existing works focus primarily on math benchmarks [Cobbe et al., 2021, Hendrycks et al., 2021, Welleck et al., 2022]. Zelikman et al. [2022] leverage hints to iteratively generate correct answers and fine-tune on the respective CoT rationales. Kumar et al. [2024] employ RLFT over multiple trials to induce self-correction. Similarly, Zelikman et al. [2024] make use of RL fine-tuning, but instead generate rationales at every token position. Instead of imitation, Wang et al. [2025] rely on critique fine-tuning to induce self-correction. Wulfmeier et al. [2024] make use of inverse RL to avoid compounding errors. Other works rely on ICL abilities to learn from previous mistakes [Zhang et al., 2024, Monea et al., 2024]. While conceptual corrections are possible, exact token-level correction is usually difficult for autoregressive generation [Cundy and Ermon, 2023].

B Environments & Datasets

We conduct experiments on three sets of environments: multi-armed bandits, contextual bandits and tic-tac-toe. For the SFT experiments reported in Section 4.5, we generate our own expert datasets. In this section, we provide additional details on our environments and datasets.

B.1 Multi-arm Bandits: BanditBench

MABs [Slivkins et al., 2019, Lattimore and Szepesvári, 2020] are a classic problem setting in RL that isolates the *exploration-exploitation* trade-off. In contrast, commonly used RL environments [Bellemare et al., 2013, Tassa et al., 2018] often conflate exploration with other RL-specific aspects, such as delayed rewards [Arjona-Medina et al., 2019]. We rely on the MAB scenarios released in BanditBench [Nie et al., 2024] and also used by [Krishnamurthy et al., 2024]. MABs come with a number of variable dimensions including the scenario type (textual description of the task), the type of reward distribution (Gaussian, Bernoulli) and its corresponding noise level (low/medium/high), the number of arms (i.e., actions), and the number of interaction steps per episode. Consequently, MABs are a good testbed for LLM agents.

We focus on the *continuous* and *button* variants released by Nie et al. [2024]. We report results for MAB with $k \in \{5, 10, 20\}$ arms ($|\mathcal{A}| = k$) for three levels of stochasticity (low/medium/high). In our experiments, for every arm the corresponding reward is sampled from a Gaussian distribution $r \sim \mathcal{N}(\mu, \sigma)$ where $\mu \sim \mathcal{U}(0, 1)$ and is a fixed scalar $\sigma \in \{0.1, 1, 3\}$ for the three levels of stochasticity, respectively. For all MAB settings, we limit the horizon T to 50 interaction steps. Limiting the horizon is necessary to handle the increasing lengths and consequently RAM requirements for fine-tuning. While we consider 50 interaction steps sufficient for 5 and 10 arms, it is insufficient for the 20 arms scenario. However, note that the general trends are well observable for the 20 arms scenario. In Figure 7, we show the continuous and button Gaussian MABs with CoT instructions for the agent. Similarly, in Figure 8 we show the same instances without CoT instructions.

B.1.1 Baselines.

We compare against two commonly used baselines for MABs: Upper-confidence Bound (UCB) [Auer, 2002] and a random agent that selects actions uniformly at random (see Appendix C for details). UCB is considered optimal and represents the upper-bound for agent performance, whereas the random baseline represents the lower bound. We provide implementations details for all baselines in Appendix C.

B.1.2 SFT Datasets.

In our main experiments, we focused on self-generated CoT rationales and action predictions produced by our fine-tuned agents, which do not require a pre-collected dataset. In contrast, for our SFT experiments reported in Section 4.5, we generated UCB expert datasets. In particular, we construct two dataset instances: a *behavior cloning* dataset that only contains expert actions and a

Continuous MAB (Gaussian)

You are a bandit algorithm and interact with 5 arms labeled 0,1,2,3,4. Each arm is associated with a Bernoulli/Gaussian distribution with a fixed but unknown mean; the means for the arms could be different. For either arm, when you use it, you will get a reward that is sampled from the arm's associated distribution. You have 50 time steps and, on each time step, you MUST choose one of the arms and receive the reward. Your goal is to maximize the total reward.

[More Instructions]

Think step-by-step and output your final answer in the format ACTION=X where X is one of the arms listed above. IMPORTANT: Provide your (SHORT!) thinking process and your answer ACTION=X

So far you have tried/seen:
Step=0 Action=1 Reward=0.3
Step=1 Action=2 Reward=0.1
Step=2 Action=0 Reward=-0.5
Step=3 Action=3 Reward=0.5
Step=4 Action=1 Reward=0.24
...
What do you predict next?

Button MAB (Gaussian)

You are a bandit algorithm in a room with 5 buttons labeled red, green, blue, yellow, orange. Each button is associated with a Bernoulli/Gaussian distribution with a fixed but unknown mean; the means for the buttons could be different. For either button, when you press it, you will get a reward that is sampled from the button's associated distribution. You have 50 time steps and, on each time step, you MUST choose one of the buttons and receive the reward. Your goal is to maximize the total reward over the 50 time steps.

[More Instructions]

Think step-by-step and output your final answer in the format ACTION=X where X is one of the arms listed above. IMPORTANT: Provide your (SHORT!) thinking process and your answer ACTION=X

So far you have tried/seen:
Step=0 Action=green Reward=0.3
Step=1 Action=blue Reward=0.1
Step=2 Action=orange Reward=-0.5
Step=3 Action=red Reward=0.5
Step=4 Action=green Reward=0.24
...
What do you predict next?

Figure 7: Illustration of *continuous* and *button* Gaussian multi-armed bandits scenarios from Bandit-Bench [Nie et al., 2024] using our context representation and **with CoT** instructions.

thought cloning (TC) dataset that incorporates expert actions alongside a thought process for coming up with the respective action (i.e., w/ CoT). For every arm and stochasticity level combination, we construct 32K rollouts from different MABs, which amounts to 1.6M transitions (state, action, reward) per dataset.

To provide the thought process in the TC datasets, we reconstruct the computations of the UCB values conducted by the UCB expert in textual format. The thought process ends with a final conclusion why a particular action was selected (i.e., highest UCB value or exploratory action). Consequently, the action z_t at step t contains the thought process z_{CoT} and the action to execute a_t (see Section 3.2). We illustrate the actions contained in the dataset for a trajectory at steps 4 and 11 in Figure 9. The BC datasets do not contain the thought process. Instead, they only contain the final predictions made by the model, for example `ACTION=yellow` as shown in Figure 9.

B.2 Contextual Bandits

MABs do not emit states. In contrast, CBs emit state representations at every interaction step, making them contextual. Consequently, CBs are interesting to test abilities of LLMs to make use of the given context when predicting the next action. For our CB experiments, we leverage the MovieLens environment released by Nie et al. [2024], a semisynthetic bandit task based on the MovieLens dataset [Harper and Konstan, 2015]. In this setting, the agent operates as a movie recommendation engine given a contextual description of a user (10K users in total) and a list of K possible movies. The context representation provides a textual description of the user to recommend the movie to. This description includes the user's gender, age, profession, location, and a numeric description of the user's preferences for each of the possible movies. As for MABs, we report results for $K \in \{5, 10, 20\}$, limit the horizon to 50 interaction steps. In Figure 10, we provide an example for a MovieLens CB with 5 actions with our context representation and CoT instructions.

Baselines. Similar to MABs, we compare against LinUCB [Chu et al., 2011] and an agent selecting actions uniformly at random. We provide implementation details on our baselines in Appendix C.

Continuous MAB (Gaussian)

You are a bandit algorithm and interact with 5 arms labeled 0,1,2,3,4. Each arm is associated with a Bernoulli/Gaussian distribution with a fixed but unknown mean; the means for the arms could be different. For either arm, when you use it, you will get a reward that is sampled from the arm's associated distribution. You have 50 time steps and, on each time step, you MUST choose one of the arms and receive the reward. Your goal is to maximize the total reward.

[More Instructions]

Output ONLY your final answer in the format ACTION=X.

So far you have tried/seen:
Step=0 Action=1 Reward=0.3
Step=1 Action=2 Reward=0.1
Step=2 Action=0 Reward=-0.5
Step=3 Action=3 Reward=0.5
Step=4 Action=1 Reward=0.24
...
What do you predict next?

Button MAB (Gaussian)

You are a bandit algorithm in a room with 5 buttons labeled red, green, blue, yellow, orange. Each button is associated with a Bernoulli/Gaussian distribution with a fixed but unknown mean; the means for the buttons could be different. For either button, when you press it, you will get a reward that is sampled from the button's associated distribution. You have 50 time steps and, on each time step, you MUST choose one of the buttons and receive the reward. Your goal is to maximize the total reward over the 50 time steps.

[More Instructions]

Output ONLY your final answer in the format ACTION=X.

So far you have tried/seen:
Step=0 Action=green Reward=0.3
Step=1 Action=blue Reward=0.1
Step=2 Action=orange Reward=-0.5
Step=3 Action=red Reward=0.5
Step=4 Action=green Reward=0.24
...
What do you predict next?

Figure 8: Illustration of *continuous* and *button* Gaussian multi-armed bandits scenarios from Bandit-Bench [Nie et al., 2024] using our context representation **without CoT** instructions.

B.3 Tic-tac-toe

Finally, we use the text-based Tic-tac-toe environment released by Ruoss et al. [2024] (see Figure 11 for an example). Unlike MABs and CBs, Tic-tac-toe is a stateful environment with proper state transitions (i.e., action predicted at step t affects the state observed at step $t+1$). The agent receives scalar rewards of 1, 0, and -1 for winning, drawing, and loosing against its opponent, respectively. Episodes last until either of the players wins, draws or looses. To enable easy extraction of actions from the generated rationales, we represent the action space as a discrete set of 9 actions, corresponding to the grid positions on the 3×3 grid used in Tic-tac-toe ($| \mathcal{A} | = 9$). However, only at the start of an episode, all 9 actions are valid. Subsequently, only a subset is valid, because of taken board positions. We (optionally) provide the set of valid actions at a particular step in textual form in the context given to the agent. Ruoss et al. [2024] demonstrated that frontier models struggle to achieve strong performance in this environment and barely beat a random opponent. Consequently, we deem it a good target to investigate the efficacy of RLFT.

Baselines. Following Ruoss et al. [2024], we compare against a random agent by default. In addition, we also compare against (MCTS) [Coulom, 2006], and a noisy variant of MCTS that selects an action randomly with 50% chance and according to MCTS otherwise.

C Experimental & Implementation Details

C.1 Training & Evaluation

In our experiments, we fine-tune Gemma2 models in three model sizes (2B/9B/27B). For all experiments, we use the instruction-tuned versions of Gemma2 and leverage the respective instruction pre-and-postfixes. For bandits, we fine-tune all models for a total of 30K updates and evaluate after every 10K steps. with an accumulated batch size of 128. Similarly, we fine-tune for 12K updates and evaluate every 4K updates on Tic-tac-toe. We report the mean and 95% confidence intervals over three seeds, as suggested by Agarwal et al. [2021].

General. We train all agents with an accumulated batch size of 128. We use a learning rate of $1e^{-4}$, 100 linear warm-up steps followed by a cosine decay to $1e^{-6}$. To allow for memory-efficient

Multi-armed Bandit at step 4

Let's think step-by-step. We calculate the counts and means for every action.

[More Thoughts]

Count for action blue = 1, Mean = (-1.91) / 1 = -1.91, UCB = -1.91 + sqrt(1 / 1)) = -0.91
Count for action green = 1, Mean = (1.41) / 1 = 1.41, UCB = 1.41 + sqrt(1 / 1)) = 2.41
Count for action red = 1, Mean = (0.45) / 1 = 0.45, UCB = 0.45 + sqrt(1 / 1)) = 1.45
Count for action yellow = 0, Mean = NaN, UCB = NaN
Count for action orange = 0, Mean = NaN, UCB = NaN

We have not yet selected all actions. Therefore, we select the next one.

ACTION=yellow

Multi-armed Bandit at step 11

Let's think step-by-step. We calculate the counts and means for every action.

[More Thoughts]

Count for action blue = 1, Mean = (-1.91) / 1 = -1.91, UCB = -1.91 + sqrt(1 / 1)) = -0.91
Count for action green = 4, Mean = (1.41 + 0.17 + 0.67 + -0.1) / 4 = 0.54, UCB = 0.54 + sqrt(1 / 4)) = 1.04
Count for action red = 3, Mean = (0.45 + 0.78 + 2.16) / 3 = 1.13, UCB = 1.13 + sqrt(1 / 3)) = 1.71
Count for action yellow = 1, Mean = (-1.03) / 1 = -1.03, UCB = -1.03 + sqrt(1 / 1)) = -0.03
Count for action orange = 1, Mean = (-1.2) / 1 = -1.2, UCB = -1.2 + sqrt(1 / 1)) = -0.2

We select actions according to the highest UCB value. Therefore, action red is selected.

ACTION=red

Figure 9: Illustration of UCB rationales contained in our **SFT expert datasets** at two timesteps (4 and 11) in the same trajectory. Both examples show the Thought Cloning dataset instance containing both the produced CoT rationale along with the predicted action. The Behavior Cloning instances contain only the final action prediction (in red).

fine-tuning of 2B and 9B models, we train using the AdaFactor optimizer [Shazeer and Stern, 2018]. We experiment with LoRA [Hu et al., 2022] for fine-tuning the 9B and 27B models but found it insufficient for improving the agent's decision-making abilities in our setting. However, LoRA considerably reduces the amount of memory required for RLFT and has been shows to work well for supervised fine-tuning of decision-making agents [Schmied et al., 2023]. Therefore, we deem it a promising candidate for RLFT in decision-making scenarios. Furthermore, we employ gradient clipping of 1.0. We list all hyperparameters in Table 1.

Context Lengths & Generation Budget. For all model sizes and tasks, we use a context length of 1792 for the input context. By default, we set the generation budget to 256 tokens, except for the knowing-doing gap analyses reported in Section 4.2, which require a larger budget of 2048 tokens. Consequently, the effective sequence length for fine-tuning is 2048.

Hardware Setup. We train all models on a server equipped with 8 $\times$ H100 GPUs.

C.2 RLFT

For our RLFT experiments on bandits, we employ the context representation, action factorization, reward shaping terms, and training objectives described in Section 3.2. To extract the target action a_t from z_t, we make use of a stack of regex expressions against the target pattern (i.e., `ACTION=X`) and consider the last match in the generated tokens as a_t. In addition to being fairly robust, we found that this approach allows for more flexibility during the RLFT process and led to better outcomes than a more structured approach. Furthermore, across model sizes, we found it essential to introduce a reward shaping term to penalize rationales that contain no valid actions. By default, we use a reward penalty of -5 for invalid actions. Empirically, we found that this reward shaping term is sufficient for the models to produce valid actions early on in the training.

MovieLens Contextual Bandit

You are an AI movie recommendation assistant for a streaming platform powered by a bandit algorithm that offers a wide variety of films from different studios and genres. There are 5 unique movies you can recommend, named star_wars_(1977), contact_(1997), fargo_(1996), return_of_the_jedi_(1983), liar_liar_(1997). When a user visits the streaming platform, you assess their demographic description to choose a movie to suggest. You aim to match the user with movies they are most likely to watch and enjoy.

[More Instructions]

Think step-by-step and output your final answer in the format ACTION=X where X is one of the arms listed above. IMPORTANT: Provide your (SHORT!) thinking process and your answer ACTION=X

So far you have tried/seen:
...
Step=4 This person is a 28-year-old man, working as a administrator and live in Santa Clara county, CA. The user has some numerical values that represent their true implicit preference or taste for all movies: [-0.04, 0.02, -0.02, -0.0, 0.02]
What do you predict next?

Figure 10: Illustration of **contextual MovieLens scenario** from BanditBench [Nie et al., 2024] using our context representation and instructions.

We fine-tune using the clipping objective introduced by Schulman et al. [2017] with an additional KL constraint to the reference policy π_{ref}. We set $\beta = 0.05$ and $\epsilon = 0.2$ for all experiments. We make use of the approximated (per-token) KL divergence instead of computing the full KL. While we found that computing the full KL slightly improves performance, it slows down training considerably. In contrast to Ahmadian et al. [2024] and Ramesh et al. [2024], we do not rely on producing multiple rollouts, because it is impractical for the mutli-step nature of decision-making tasks. While generating multiple actions at a particular timestep is possible for simulated environments, it requires environment resets. Therefore, we rely on standard MC-baselines to estimate A_{adv}.

For bandit experiments, we maintain a pool of 512 stochastic MABs. For every rollout, we let the agent interact with a subset of 16 bandits for a single episode (50 timesteps). Consequently, every rollout contains 800 transitions. Similarly, for Tic-tac-toe, we maintain 16 parallel environments and collect 2048 rollout steps. We conduct 1 and 2 update epochs over the rollout buffer for bandits and Tic-tac-toe, respectively.

C.3 SFT

For our SFT experiments on MABs, we fine-tune on either on the expert action or expert rationales produced by UCB. We employ standard SFT training using a cross-entropy objective on the target tokens.

C.4 Exploration Mechanisms

In Section 18, we compare a variety of classic exploration mechanisms and LLM-specific approaches and study their effects on agent performance on Gaussian MABs with 10 arms. Here, we provide a description for each mechanism.

Try-all. The try-all strategy is inspired by UCB, which incorporates an initial phase for trying all untried actions. This is because the UCB values for all untried actions are ∞. Therefore, we incorporate the same exploration phase when performing ICL and RLFT at the beginning of every episode. To enable fine-tuning on exploration actions, we provide an action rationale template to the model (e.g., `Action X has not been tried yet, let's explore it. ACTION=X`). While simple, we find that this try-all strategy is effective for lowering regret across all model sizes (see Figure 18). This suggests that the model is able to select appropriate actions if given sufficient information, but struggles to explore.

Tic-tac-toe

You are an agent playing tic-tac-toe. You observe a board with 9 entries that looks like this:
000
100
002
1 indicates that player 1 has placed a stone in that square. 2 indicates that player 2 has placed a stone in that square. 0 indicates that no stone has been placed in that square. You play as 1.
There are 9 possible actions: 0, 1, 2, 3, 4, 5, 6, 7, 8. The actions correspond to the following board locations
012
345
678

[More Instructions]

Think step-by-step and output your final answer in the format ACTION=X where X is one of the arms listed above. IMPORTANT: Provide your (SHORT!) thinking process and your answer ACTION=X

So far you have tried/seen:
Step=0 State=000000000 Action=0 Reward=0
Step=1 State=102000000 Action=4 Reward=0
Step=2 State=102010002 Action=5 Reward=0
What do you predict next?

Figure 11: Illustration of the text-based **Tic-tac-toe** environment.

ϵ-greedy. ϵ-greedy is classic exploration mechanism and commonly used in RL algorithms [Mnih et al., 2015, Hessel et al., 2018]. For our experiments, we use $\epsilon = 0.1$ both during training and evaluation. We explored other values for ϵ but did not observe performance improvements. As for the try-all strategy, we provide an action rationale template to enable fine-tuning on exploration actions.

Context Randomization. Context Randomization is an LLM-specific mechanism designed to introduce randomness in the action predictions by modifying the context representation. At every interaction step, we construct a mapping from the original action labels to a shuffled list of the same action labels. Subsequently, we remap action in the context history according to the constructed mapping. Finally, the predicted action is mapped back to the original action label space and executed environment. Besides introducing randomness, context randomization acts as a control mechanism to ensure that the observed biases do not only stem from biases towards particular action-tokens (e.g., blue occurs more often than magenta in the pre-training dataset).

Context Summary. Similar to Krishnamurthy et al. [2024] and Nie et al. [2024], we evaluate the effects of providing a context summary to the agent. After the context history, we provide the model with a summary of that history that contains the number of times every action has been selected so far, along with their respective mean rewards.

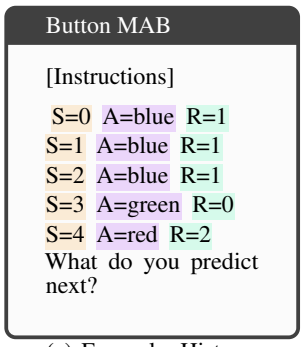
Self-Correction. Inspired by Kumar et al. [2024] and Wang et al. [2025], we employ self-correction to the model's predicted actions. First, we let the model generate its is initial rationale and corresponding action prediction. Then we append the generated rationale along with a self-correction message (similar to Kumar et al. [2024]) to the input context, and repeat the action generation. Finally, we extract the action from the final response and execute it in the environment. For RLFT, we only fine-tune on the final response, but retain the initial response along with the self-correction message in the context.

Self-Consistency. Instead of generating a single answer, self-consistency [Wang et al., 2022] relies on generating multiple responses. Subsequently, self-consistency employs a majority voting mechanism to determine the final response. For our experiments in Figure 18, we report results for self-consistency with 16 generated responses. Instead of majority voting, we experimented with sampling from the respective response distribution or random mechanisms.

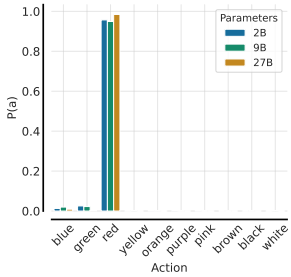
Exploration Bonus. Finally, we evaluate a reward shaping mechanism in the form of an exploration bonus. In particular, we give an exploration bonus of +1 if the agents selects an action not yet tried within the respective episode. While simple, we find that the exploration bonus effectively narrows the gap to the UCB expert. This highlights the importance of reward shaping for fine-tuning LLMs in decision-making scenarios.

Name	Value	Description
Training		
`training_steps`	30K or 12K	Number of training steps.
`eval_freq`	10K or 4K	Evaluation frequency (in updates).
`batch_size`	128	Accumulated batch size.
`lr_scheduler`	Linear + cosine	Learning rate scheduler
`warmup steps`	100	Warmup steps.
`lr`	$1e^4$ to $1e^6$	Maximum learning rate.
`optimizer`	AdaFactor	Optimizer.
Sequence Length & Generation Budget		
`context_length`	1792	Input context length.
`num_tokens`	256	Generation budget.
RLFT		
`rollout_steps`	800 or 2048	Rollout steps in-between updates.
`update_epochs`	1 or 2	Update epochs over rollout-buffer.
`reward_penalty`	-5	Reward penalty for invalid actions.
`loss`	PPO clipping objective + KL constraint	Objective function.
`baseline`	MC-baseline or state-value head	Baseline.
`envs`	16	Number of parallel envs.
ϵ	0.2	Clipping value.
β	0.05	KL coefficient.
`reward_norm`	True	Whether reward normalization is used.
`train_temp`	1.0	Sampling temp during rollouts.
`eval_temp`	0.0	Sampling temp during evaluation.
`top_p`	1.0	Sampling top-p.
Hardware		
`accelerator`	8 × H100	Hardware accelerator.

Table 1: Default **hyperparameters** used in our experiments.



(a) Example: History



(b) Example: Probability Mass

Figure 12: Illustration of **action probabilities** leading to greediness behavior. Models exhibit overly high action probabilities in the presence of rewards, potentially resulting in repeatedly selecting sub-optimal actions

D Additional Results

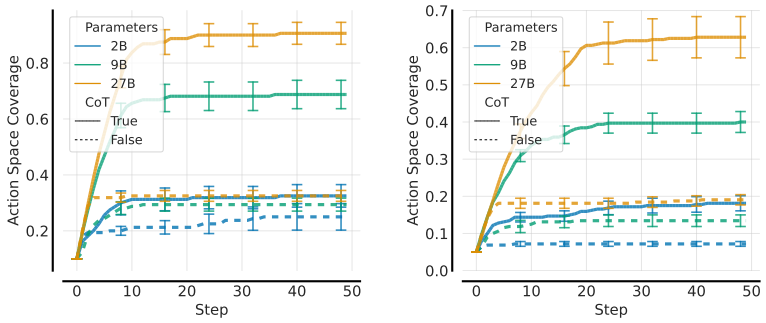
D.1 Failure Modes

D.1.1 Greediness

Greediness is characterized by the LLM overly favoring the best performing action among a small set of actions seen so far. We define action coverage C_t at step t as the fraction of available actions that have been selected at least once, $C_t = \frac{\{a \in \mathcal{A} : N_t(a) > 0\}}{|\mathcal{A}|}$ with $N_t(a)$ representing the number of times action a has been selected until t.

Action probabilities. The suboptimal action coverage reported in Section 4.2 is caused by the model overly favoring high-reward actions (i.e., overly high action probabilities). In Figure D.1.1, we provide an illustration of the action probabilities for a given input history. Across model sizes, Gemma2

exhibits overly high action probabilities in the presence of reward, which results in repeatedly selecting a potentially suboptimal action.

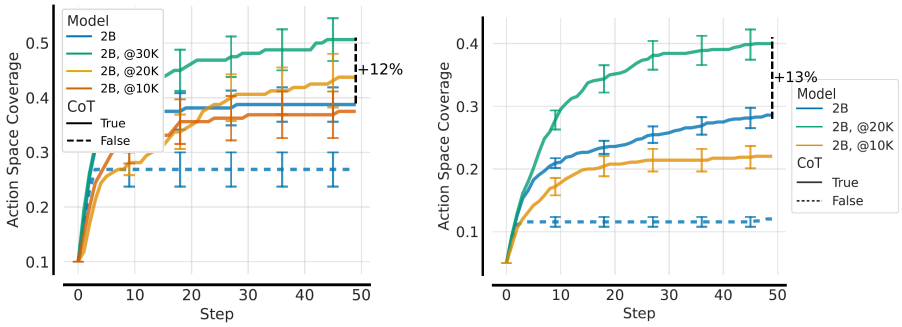


(a) Action Coverage: 10 arms (b) Action Coverage: 20 arms

Figure 13: Illustration of greediness for the **numbers** scenario.

Greediness on Continuous MABs. We repeat the analyses conducted in Section 4.2 using *numbers* instead of *buttons* as the possible actions. Indeed, we find that the same trends hold. Without CoT the performance remains low. For Gemma2 27B, we observe an increase in the action coverage to almost 90% for the 10 arms scenario, and to 60% for the 20 arms scenario.

Post RLFT. In line with Figure 14a, we present the post RLFT action coverage on the 20 arms scenario in Figure 14b. Similar to the effects on the 10 arms scenario, we observe that RLFT improves the action coverage by 13%.



(a) Action Coverage: 10 arms (b) Action Coverage: 20 arms

Figure 14: Effect of RLFT on **greediness** for 2B on 10 and 20 arms (medium noise).

D.1.2 Frequency Bias

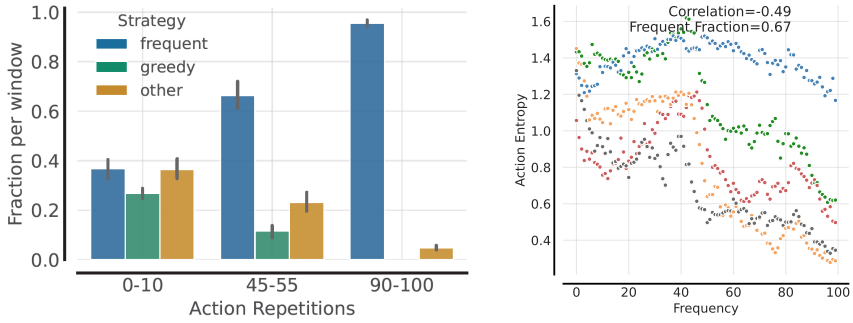
Frequency bias is characterized by repeatedly selecting the most frequently occurring actions in the context, even when the dominant action gives low reward. To measure frequency bias, we first construct a variety of interaction histories (occurred during environment interaction) containing between 2 and 10 transitions. This interaction history is collected using a random policy. Given an initial interaction history, we repeat the last action in the history, which we also refer to as target action, between 0 and 100 times. Finally, we report the entropy all actions, $H(\theta) = -\sum_{a \in A} \pi_\theta(a \mid \tau) \log \pi_\theta(a \mid \tau)$. To achieve this, we conduct a separate forward pass for every possible action in the action space and report the respective log probabilities. We repeat the same procedure for different interaction histories and target actions (see Figure 4a and c). For the 10 arms scenario, every interaction history therefore results in 1000 (10 arms * 100 repetitions of the target action) forward passes. We repeat this procedure for the 5 target actions reported in Figure 4 using 5 interaction histories per action, accumulating to a total of 25K model forward passes (1000 * 5 * 5) per figure.

To quantify frequency bias, we categorize the resulting actions as *frequent* action, *greedy*, or *other* if they are neither frequent nor greedy. Subsequently, we compute the frequent F_f, greedy F_g and other

F_o fractions as reported in Figure 4:

$$F_f = \frac{N_T(a_f)}{N}; \quad F_g = \frac{N_T(a_g)}{N}; \quad F_o = \frac{\sum_{a \in A \setminus \{a_f, a_g\}} N_T(a)}{N}, \quad \text{with } N = \sum_{a \in A} N_T(a). \tag{3}$$

Note that there can be an overlap between greedy and frequent actions. In these (rare) cases, the greedy action category is dominant, i.e., we categorize the action as greedy even if it would also be the frequent action. This implies that the actions classified as frequent in Figure 4, are always suboptimal/bad compared to the respective greedy action. Consequently, a high F_f indicates that the model prefers the most frequent action even when observing a better action in the context.



(a) Action Strategies (b) Corrleations

Figure 15: Effect of RLFT on **frequency bias** for 2B (10 arms, medium noise).

Post RLFT. In Section 4.3, we observed that RLFT counteracts frequency bias. In addition to frequency buckets reported in Figure 15a, we provide the plot frequency against action entropy post RLFT in Figure 15b. Compared to Figure 4a, we observe that after RLFT the models maintains a higher action entropy for longer. Only at high repetition frequencies the action entropy decreases severely. Consequently, RLFT counteracts frequency bias, but does not completely alleviate it.

D.1.3 Knowing-Doing Gap

The *knowing-doing gap* has been observed by Paglieri et al. [2024] and Ruoss et al. [2024]. It states that models possess knowledge about a task or consequences of their behavior (i.e., they *know* what to do), but cannot materialize this knowledge when acting (i.e., they are incapable of *doing*). We illustrate the knowing-doing gap empirically in Figure 5. To this end, we first task Gemma2 27B to produce the UCB algorithm and to compute the relevant quantities required to act according to UCB ("knowing"). This involves counting how often every action was selected, computing the mean rewards for every action, and computing the final UCB values. After producing the quantities, the model is tasked to act according to them (i.e., "doing"). In Figure 21, we present and example of the respective instructions given to the model along with a response produced by Gemma2 27B.

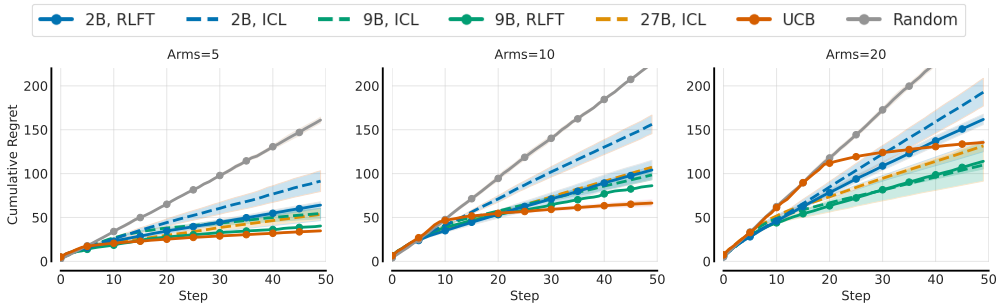
To evaluate performance empirically, we let Gemma2 27B interact with the environment (64 parallel instances) for 50 timesteps. We extend the token generation budget to 2048 tokens per step, to accommodate the additional required computations. Every produced action z contains both the CoT rationale z_{CoT} and the final selected action a. We first extract the computed UCB values from the produced rationale z_{CoT}. To achieve this, we task Gemma2 27B to enclose the computed values by <ucb_values> and </ucb_values> blocks. Then we extract the selected action a and execute it in the environment. For this experiment, we use Gemma2 27B, because we found that 2B and 9B struggled with computing the relevant UCB quantities and with enclosing them appropriately under the desired blocks.

Quantifying "Knowing". To quantify "knowing", we compare the UCB values computed by the model and extracted from z_{CoT} against the real UCB values. To this end, we recompute the real UCB values for every action at every time-step given the observed history. We consider the rationale as correct if the arm with the highest UCB values match. We opt for this choice rather than checking for exact equality, because we observed that the model struggles with exact calculations for complex operations. This is expected, because the necessary computations involve logarithm and square roots of floating point values. While tool use (e.g., calculator) could mitigate this issue, we observed that Gemma2 27B gets the quantities approximately right, resulting in valid rationales. Thus, the fraction of correct rationales is $F_c = \frac{1}{N} \sum_{i=1}^{N} g(z_{CoT}^i)$ given a classifier g.

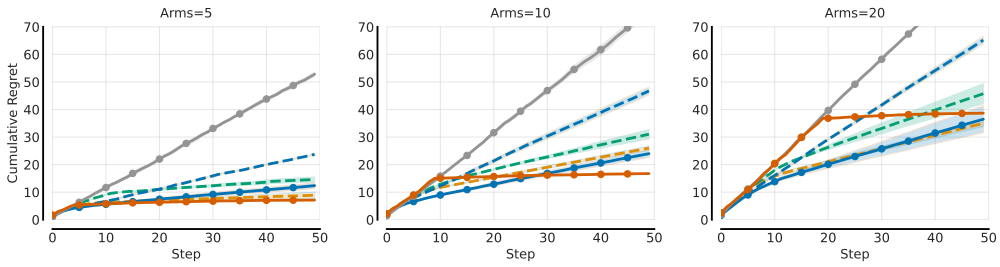
Quantifying "Doing". To quantify "doing", we categorize the generated actions as *optimal* action if the model selects the action with the highest UCB value, as *greedy* if it selects the action with the highest UCB value among the set of actions tried so far, and as *other* if the action is neither optimal nor greedy. It is possible that the greedy action is the optimal action. However, in this case the action is considered optimal instead of greedy. Subsequently, we compute the percentages of greedy/optimal/other actions (e.g., $F_g \times 100$). We find that the model clearly *knows* how to solve the task, with 89% of all rationales being correct (see Figure 5).

D.2 Multi-armed Bandits

In Figure 6, we report the cumulative regrets across model sizes and arms for a medium noise ($\sigma = 1.0$) scenario. In addition, we repeat the same experiment in the low-noise ($\sigma = 0.1$) and the high-noise $\sigma = 3.0$ setting in Figure 16. For both noise levels, we observe similar trends as for the medium noise setting. In particular, we observe that LLMs clearly outperform the random baseline and RLFT lowers the cumulative regret for Gemma2 2B across all arm scenarios.



(a) High Noise ($\sigma = 3$)



(b) Low Noise ($\sigma = 0.1$)

Figure 16: Main Comparison on **Gaussian MABs** button scenario in the **(a)** high $\sigma = 3$ and **(a)** low $\sigma = 0.1$ noise settings. We compare cumulative regrets (lower is better) of classic baselines against ICL and RLFT performances for Gemma2 2/9/27B for 5, 10, and 20 arms.

D.3 Contextual Bandits

We repeat the same fine-tuning experiment for the contextual MovieLens bandits described in Section B.2. In Figure 17, we report the cumulative regrets attained by Gemma2B across different model sizes and for 5, 10 and 20 arms. Furthermore, we compare against a LinearUCB and a Random baseline. Overall, we observe similar performance improvements for RLFT on CBs as on MABs. While the ICL performances barely attain the same performance as a Random agent, RLFT fine-tuned Gemma2 2B perform similar to UCB.

D.4 Effect of Exploration Mechanisms

E Ablations

Finally, we provide additional details on the ablations conducted in this work.

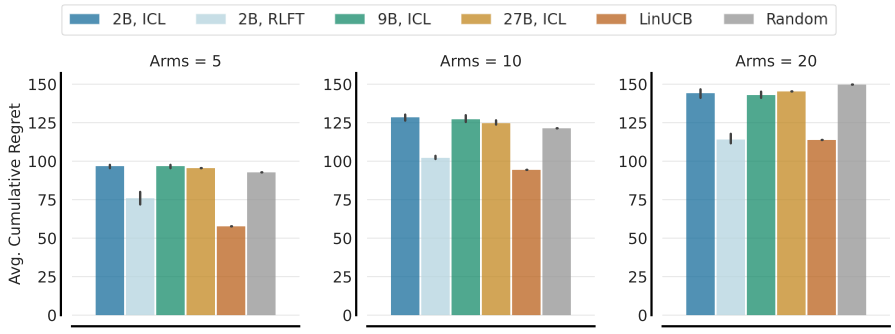


Figure 17: Main Comparison on Gaussian **MovieLens CBs** for **(a)** 5, **(b)** 10, and **(c)** 20 arms. We compare classic baselines against ICL and RLFT performances for Gemma2 2/9/27B.

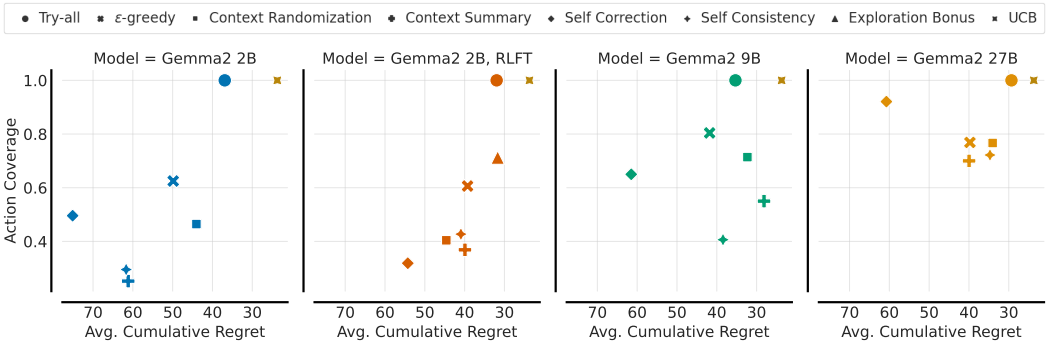


Figure 18: Effect of **exploration mechanisms** on action coverage and cumulative regret.

E.1 RLFT in Tic-tac-toe.

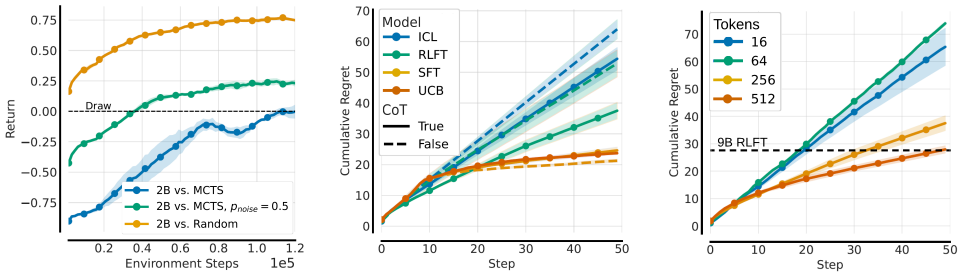
To investigate the efficacy of RLFT in stateful environments, we evaluate on Tic-tac-toe from Ruoss et al. [2024], in which frontier models struggle to achieve strong performance (see Appendix C for training details). We fine-tune against three opponents: a random agent, Monte Carlo Tree Search (MCTS) [Coulom, 2006], and noisy MCTS (50% of actions selected at random). We find that RLFT significantly enhances the win-rate of Gemma2 2B against all opponent compared to ICL (see Figure 19a). Against the random agent, RLFT elevates the average return from 0.15 (i.e., winning 15% of games) to 0.75. Notably, the agent even manages to draw against the optimal MCTS baseline ($-0.95 \rightarrow 0.0$), underscoring the effectiveness of RLFT for decision-making. However, for high performance it is essential to provide the legal actions in the context (see Figure 20).

E.2 Tic-tac-toe: Effect of Legal Actions in State

By default, we provided the legal actions available at the current turn within the input context to the agent. We found this design choice to be essential for effective fine-tuning compared to training without legal actions (see Figure 19b). Without legal actions in the context the average return drops from 0.75 (w/ legal actions) to 0.45. This suggests that the LLM fails at identifying the appropriate actions among the set of all possible actions when not given legal actions at the current state. In contrast, when provided with sufficient information (i.e., legal actions), the LLM is able to select actions appropriately (similar to Section 4.4). Providing the legal actions in the agent's context alleviates the need to explore/identify invalid actions. Consequently, this shortcoming further highlights the need for principled exploration strategies for LLMs in decision-making scenarios.

E.3 Importance of CoT for RLFT

CoT reasoning is critical for ICL performance (see Figure 3), but the question remains how CoT influences RLFT. Therefore, we run RLFT on Gemma2 2B on the 10 arms Gaussian MAB both w/ and w/o CoT (see Figure 19b, RLFT). Indeed, without CoT, RLFT barely attains the performance of ICL w/ CoT. This highlights the function of CoT as a vital exploration and rationalization mechanism



(a) Tic-tac-toe: Opponents (b) MAB: Effect of CoT (c) MAB: "Thinking" Time

Figure 19: Ablations. **(a)** Effect of RLFT in Tic-tac-toe from Ruoss et al. [2024]. **(b)** Effect of CoT on ICL, RLFT and SFT (expert data) performance on MABs. **(c)** Effect of increasing the number of "thinking" tokens to generate during RLFT.

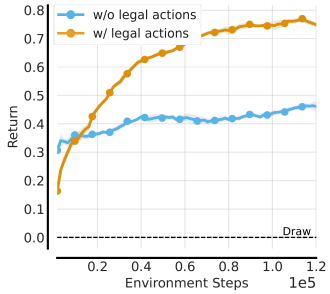


Figure 20: Effect of removing **legal actions** from the context in Tic-tac-toe.

for decision-making. For our results without CoT reported in Figure 19b, we remove the CoT instructions given to our agents. Instead, we instruct the agents to not perform any reasoning steps and to only produce the action to execute in the environment a. In addition, we limit the token generation budget G to 16 to avoid that the model ignores the instructions and makes use of the additional tokens. Furthermore, this considerably speeds up training due faster rollout times and shorter context lengths.

E.4 Expert Behavior Cloning vs. Thought Cloning

A prevalent approach in sequence models for decision-making is behavior cloning (BC) [Pomerleau, 1988, Reed et al., 2022, Brohan et al., 2022, 2023], which relies on expert datasets. Consequently, we construct two UCB expert dataset comprising 32K rollouts either w/o CoT (*behavior cloning*) or w/ CoT (*thought cloning*), as described in Appendix B.1. Notably, both SFT variants successfully mimic the expert achieving comparable regret to the UCB expert (see Figure 19b, SFT). This result underscores the efficacy of training on expert data in decision-making scenarios when available, echoing recent findings in reasoning tasks [Muennighoff et al., 2025]. While BC and TC attain similar performance levels on the simplistic MABs, we anticipate that TC is advantageous in more complex decision-making scenarios as found by Hu and Clune [2023].

E.5 "Thinking" Time

We investigate the effect of giving the agent more time to "think" in Figure 19c. To achieve this, we vary the maximal number of tokens that the agent can generate per action $G \in \{16, 64, 256, 512\}$. By default, we set G to 256. Indeed, we observe that the performance improves consistently with more thinking tokens. Decreasing G to 16 or 64 results in poor performance, because the agent is unable to rationalize its decisions within the restricted generation budget. This is similar to the performance without CoT, but in contrast, the agent is instructed to produce the reasoning process. Over the course of RLFT, the agents learn to produce short rationales z_{CoT}, including the action a due to our reward shaping mechanism (see Section 3.2). However, the produced short rationales are unhelpful to improving agent performance.

In contrast, doubling G from 256 to 512 results in a considerable performance increase to the level of Gemma2 9B with RLFT (see Figure 6). We observe an increase in the average sequence length over the course of the RLFT process. This suggests that the agent learns to effectively leverage the additional "thinking time" and reflects recent observations in mathematical reasoning [Guo et al., 2025]. However, the increased performance comes with additional training cost due to the multistep nature of decision-making scenarios. In fact, we observed that rollout generation can make up the majority of the training time required by the RLFT process. This is because the agent has to produce more tokens at every environment interaction step. For example, for our default horizon of 50 timesteps and a generation budget of 500, the agent produces 25K tokens (at maximum).

F Limitations & Future Work

We focused our evaluation on the Gemma2 series and small-to-medium scale models. While we expect that our findings transfer to larger models, we deem research into frontier models important. Moreover, our MAB experiments were conducted with a limited horizon of 50 environment steps, which is sufficient for 5 and 10 arms, but insufficient for 20 arms. For future work, we believe that evaluating the exploration abilities of LLM agents is particularly interesting in environments that require targeted exploration towards an end-goal. First, this includes other stateful environments from Paglieri et al. [2024] and Ruoss et al. [2024], such as Crafter [Hafner, 2021]. Second, we deem a systematic investigation into exploration abilities of LLMs in existing agentic benchmarks [Mialon et al., 2023, He et al., 2024, Zhou et al., 2023] interesting. In our ablation studies, we found that LLMs benefit from additional "thinking" time and believe that allowing for a larger generation budget will become increasingly important for agentic scenarios, especially for scenarios with high-stakes decisions (e.g., economics or ethics). We deem investigations into such high-stakes scenarios fruitful for future work. While increasing "thinking" time improves performance, it comes with excessive computational cost at training time due to the rollout generation and the multi-step nature of decision-making. Therefore, modern recurrent architectures [De et al., 2024, Beck et al., 2025] that allow for faster inference may be promising alternatives for decision-making.

Instructions for Gemma2 27B as UCB agent

[Instructions]

Your task is to act according to the Upper-Confidence-Bound (UCB) algorithm. First, write down the UCB algorithm. Then compute the relevant UCB values for every button (approximate if necessary). Finally, select your action according to the computed quantities.

[More Instructions]

So far you have tried/seen:
Step=0 Action=blue Reward=1.06
Step=1 Action=blue Reward=1.82
Step=2 Action=green Reward=1.0
Step=3 Action=green Reward=-0.26
Step=4 Action=blue Reward=-0.58
Step=5 Action=blue Reward=-0.34
Step=6 Action=green Reward=1.19
Step=7 Action=green Reward=2.21
Step=8 Action=green Reward=0.07
Step=9 Action=green Reward=1.45
Step=10 Action=green Reward=-0.11
What do you predict next?

Generated response by Gemma2 27B

Here's how I'd approach this as a UCB bandit algorithm:
UCB Algorithm (Short Version)

- **Initialization:** For each button, set a count (number of times pressed) to 0 and an estimated reward average to 0.
- **Iteration:**
  - For each button, calculate the UCB value: $UCB = \text{average\_reward} + \sqrt{\frac{2 \cdot \ln(\text{total\_steps})}{\text{button\_count}}}$
  - Choose the button with the highest UCB value.
  - Update the button's count and average reward based on the observed reward.

UCB Calculations (Approximate)
Let's denote the buttons by their colors. Here's a breakdown after 10 steps:

- **blue:**
  - Count = 4
  - Average Reward = (1.06 + 1.82 - 0.58 - 0.34) / 4 = 0.49
  - UCB $\approx 0.49 + \sqrt{\frac{2 \cdot \ln(11)}{4}} \approx$ 1.17
- **green:**
  - Count = 7
  - Average Reward = (1.0 + -0.26 + 1.19 + 2.21 + 0.07 + 1.45 - 0.11) / 7 $\approx$ 0.92
  - UCB $\approx 0.92 + \sqrt{\frac{2 \cdot \ln(11)}{7}} \approx$ 1.47
- **Other buttons:** Count = 0, Average Reward = 0, UCB = $\sqrt{\frac{2 \cdot \ln(11)}{0}} = \infty$ (We'll explore these later)

Action Selection: The button with the highest UCB value is **green**.
Therefore, ACTION=green
Explanation: The UCB algorithm balances exploration (trying buttons we haven't seen much) and exploitation (choosing buttons that seem to have high rewards).

- Green has a higher average reward than blue, but blue has been explored less.
- The UCB formula takes this into account, giving a higher value to green because its average reward is more reliable due to more observations.
- The other buttons have infinite UCB values because we haven't tried them yet, so they are highly prioritized for exploration. However, green's current UCB is high enough to warrant another try.

Figure 21: Illustration of the **knowing-doing gap**. **(a)** Instructions for the agent, which is prompted to act like a UCB algorithm. **(b)** The response generated by Gemma2 27B with greedy decoding (temperature=0). The LLM "knows" the UCB algorithm and computes the relevant quantities approximately correctly, but acts erroneously by selecting the next action greedily.