TRAINING A GENERALLY CURIOUS AGENT

Fahim Tajwar* , **Yiding Jiang*** , **Abitha Thankaraj**
Carnegie Mellon University

Sumaita Sadia Rahman
North Carolina State University

J Zico Kolter, Jeff Schneider, Ruslan Salakhutdinov
Carnegie Mellon University

ABSTRACT

Efficient exploration is essential for intelligent systems interacting with their environment, but existing language models often fall short in scenarios that require strategic information gathering. In this paper, we present PAPRIKA, a fine-tuning approach that enables language models to develop general decision-making capabilities that are not confined to particular environments. By training on synthetic interaction data from different tasks that require diverse strategies, PAPRIKA teaches models to explore and adapt their behavior based on the environment feedback in context without gradient updates. Experimental results show that models fine-tuned with PAPRIKA can effectively transfer their learned decision-making capabilities to entirely unseen tasks without additional training. We also introduce a curriculum learning algorithm that improves PAPRIKA's sample efficiency. These results suggest a promising path towards AI systems that can autonomously solve novel sequential decision-making problems that require interactions with the external world.

1 INTRODUCTION

Large language models (LLMs) are considered to be a promising foundation for autonomous agents – systems capable of achieving goals independently with minimal human supervision or intervention. A crucial requirement for such systems is the ability to interact effectively with external environments and gather the information necessary to achieve their objectives. This capability can be formalized as solving sequential decision-making problems or performing reinforcement learning (RL) with language models as the agent. However, two challenges hinder the development of these interactive capabilities. First, most naturally occurring data lacks the structure and context needed to model interactions. Second, directly deploying models into the real world to collect interaction data can produce critical errors, which is expensive and potentially risky.

Given the impracticality of direct deployment in the wild, a natural alternative is to generate interaction data synthetically. Although generating synthetic data for every possible problem is infeasible, LLMs possess the capacity for *in-context learning* (ICL), which allows them to adapt to new tasks with minimal demonstrations (Brown et al., 2020). Instead of teaching the model to do all the interaction tasks that we care about, we should instead teach the model *in-context reinforcement learning* (Laskin et al., 2022) so that the model can solve new problems without being trained on them a priori. It shifts the focus from training the model on particular problems to training it on the general process of solving problems. This paradigm shares similarities with the supervised fine-tuning (SFT) and reinforcement learning from human feedback (RLHF) stages of training a language model (vs pretraining) where only a relatively small number of examples is needed to produce a model that can generate responses to a wide range of queries that they are not trained on. Our approach is also closely related to the principles of *meta reinforcement learning* (Beck et al., 2023).

In this work, we explore the feasibility of teaching LLMs to perform in-context RL that generalizes across different tasks. We begin by designing a diverse suite of textual decision-making tasks that require active information gathering and decision-making based on interaction outcomes. Using a base model, we generate interaction trajectories and assign scores based on their success in achieving

*First two authors contributed equally. Project website: `https://paprika-llm.github.io/`

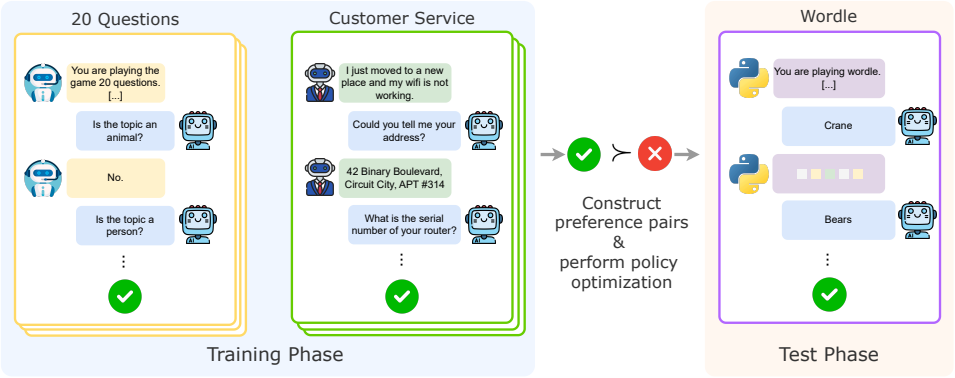

Figure 1: Overview of PAPRIKA. We design a diverse set of environments where an LLM agent needs strategic information gathering to succeed, then train an LLM on self-generated data to prefer higher performing trajectories. The resulting behavior learned by PAPRIKA transfers zero-shot to unseen tasks, showcasing its potential to build general decision making agents.

the tasks' objectives. We then apply a sequential variant of Direct Preference Optimization (Rafailov et al., 2024b, DPO) to increase the relative likelihood of successful trajectories. Unlike traditional training where computational costs are dominated by model updates, our approach's primary bottleneck lies in sampling useful interaction data. To further improve sample efficiency, we propose a curriculum learning strategy that prioritizes sampling trajectories from tasks with high learning potential.

We refer to the overall framework as PAPRIKA[1]. Our results demonstrate that training on different subsets of these games improves the performance of the model on unseen tasks. More broadly, our result highlights the potential of using synthetical data to learn in-context RL which would equip LLMs with the capability to interact with the world and solve different decision-making problems without requiring task-specific fine-tuning.

2 PRELIMINARY

Many decision making problems can be formalized as a partially observable Markov decision process (POMDP). We assume each *task*, τ is a POMDP although we will not draw on the details of the POMDP formalism in this work. As a concrete example, guessing the word "`apple`" would be a task in 20 questions. We will use *group*, $G = \{\tau_1, \tau_2, \ldots, \tau_{|G|}\}$, to refer to a high-level grouping of different tasks (e.g., the game 20 questions would be a group). Tasks in a group should share similar strategies but it is not always true that they share the same optimal policy as such constraints may be overly stringent. From the agent's perspective, each task is a black box function that takes in the agent's action a_t (and possibly the whole interaction history) and outputs an observation o_t. Both a_t and o_t are strings. In a game of 20 questions, a_t could be "`Is the word an animal?`" and the o_t could be "`No.`"

An episode contains the agent's interaction trajectory with a single environment. Unlike the conventional RL structure, we will assume that the transition-level reward is either 0 or must be inferred from o_t, and that the individual tasks can flexibly implement different observation spaces and termination conditions. An episode terminates when the agent achieves the objective of the environment or when the maximum number of interactions allowed within the environment is reached. We will use $h = (o_0, a_0, \ldots, o_H, a_H)$ to denote an episode of length H, $h_t = (o_t, a_t)$ to denote a single step of h, and $h_{p:q} = (o_p, a_p, \ldots, o_q, a_q)$ to denote a slice of h similar to array slicing. At the end of an episode, the environment emits a single score, $r(h)$, that evaluates the performance of the agent. Let π denote the LLM agent and $h \sim \pi \circ \tau$ denote sampling a trajectory from task τ using policy π. The performance of a policy on a group would be: $\texttt{Perf}(G) = \frac{1}{|G|} \sum_{\tau \in G} \mathbb{E}_{h \sim \pi \circ \tau}[r(h)]$. The agent is trained in a finite set of groups, $\mathcal{G}_{\text{train}}$, and the goal is to perform well on unseen groups, $\mathcal{G}_{\text{test}}$.

[1]The name is inspired by the movie "Paprika" (2006), where a dream detective navigates vast and strange dream worlds to solve different mysteries.

3 PAPRIKA

The goal of our paper is to develop a scalable method to instill better strategic exploration and sequential decision-making capabilities into LLMs. Prior works (Krishnamurthy et al., 2024) have shown that LLMs can perform poorly on even the simple decision making task of multi-armed bandits. Nie et al. (2024) has since then demonstrated that LLMs can be taught to perform better on bandits after fine-tuning them on synthetic trajectories generated by known algorithms such as UCB. However, this idea is limited in scope for three reasons: **(1)** we want LLMs to perform strategic exploration and decision making in more complex settings, **(2)** for most tasks, there is no known algorithm like UCB to generate good synthetic trajectories from, **(3)** it can be infeasible to collect data for all tasks that we care about.

We aim to solve these issues using our method, PAPRIKA. First, we design a suite of complex decision-making tasks that require strategic information gathering to succeed. Next, we show that in the absence of known good algorithms, existing LLMs can generate trajectories with better decision making behaviors through diversity-encouraging sampling. We then finetune the LLMs to prefer higher performing trajectories and show that this leads to better decision making abilities at test-time. More importantly, these behaviors generalize to unseen tasks without additional training. Finally, we propose a general curriculum learning algorithm that can dynamically choose which subset of tasks to train on next to improve data efficiency of such training methods. We next describe each component of PAPRIKA.

3.1 ENVIRONMENT DESIGN

Table 1: Summary of the environments used by PAPRIKA.

Environment	# Train Tasks	# Test Tasks	Maximum Turns	Env Feedback	Uses COT
Twenty questions	1500	367	20	LLM generated	✗
Guess my city	500	185	20	LLM generated	✗
Wordle	1515	800	6	Hardcoded program	✓
Cellular automata	1000	500	6	Hardcoded program	✓
Customer service	628	200	20	LLM generated	✗
Murder mystery	203	50	20	LLM generated	✗
Mastermind	1000	500	12	Hardcoded program	✓
Battleship	1000	200	20	Hardcoded program	✓
Minesweeper	1000	200	20	Hardcoded program	✓
Bandit best arm selection	81	1	20	Hardcoded program	✓

The first component of PAPRIKA is to design a set of decision making tasks that we can evaluate and train LLMs on. The environments we want should have the following desired properties: **(1)** they are purely text based, **(2)** they require multi-turn interaction, where the agents have to both understand prior history in its context and choose actions that maximize the probability of success in the future, **(3)** they are partially observable, i.e., the observations do not capture the full state or hidden information, so the agents must simultaneously explore to reveal more information and exploit to solve the task efficiently, **(4)** they are diverse and require different strategies to succeed.

With these requirements in mind, we design 10 environments in our paper. On all environments, we employ an LLM as the agent that is given a task it needs to solve through sequential interaction with the environment, which provides both observations for intermediate timesteps given the agent's actions and also a task reward at the end of an episode. For tasks requiring general knowledge about the world to generate intermediate observations, we employ another LLM (typically GPT-4o-mini) as the environment. For tasks that have rule-based observations and rewards, we find that using hardcoded programs as the verifier for these tasks is more reliable than LLMs, similar to DeepSeek-AI et al. (2025). In order to prevent environment hacking, we also use either another LLM or a hardcoded program as a judge to filter out unsuccessful trajectories that got labeled as successful by the environment. We also find that for environments requiring complex reasoning, letting the agent think using chain-of-thought (COT) prompting (Wei et al., 2022; Kojima et al., 2022) before generating a final answer improves its performance significantly, so we use COT for these environments. We provide a brief description of our environments here, please refer to 1 for a summary of our environments and C for more details.

Following prior work (Abdulhai et al., 2023), we include classic guessing games like *twenty questions* and *guess my city* in our list of environments. They require guessing a secret topic as quickly as possible by asking a sequence of questions and observing the answers. We also employ *Wordle* and *Mastermind*, where the player needs to guess a secret 5-letter word and 4-digit code respectively. The environments for these tasks provide feedback in terms of similarity between the guess and the target word/code, and the player needs to refine their guesses in future turns to maximize information gathering. We design *customer service* and *murder mystery* as dynamic text-based environments: an LLM playes the role of the environment in these tasks, which is provided with the criterion for task success and generates dynamic intermediate observations based on this criterion and agent actions.

A desirable capability in LLMs is to code and refine based on interpreter feedback. To simulate this process with a toy case, we design the environment of *Cellular Automata*, where the agent needs to make inferences about the transition rule in 1D elementary cellular automata (Wolfram, 1983; Cook et al., 2004) by observing inputs and outputs. The agent receives the outputs generated from their predicted transition rule and they have to refine their predictions based on it (see C for more details). Next, we incorporate *Minesweeper* and *Battleship* based on classical games, which require the player to interact with 2D grids to find hidden items within a fixed number of turns and refine their guesses based on per-turn observations.

Finally, we incorporate a modified version of the multi-armed bandit (Slivkins, 2024) environment from prior works (Krishnamurthy et al., 2024; Nie et al., 2024) with the following distinctions: **(1)** we let the agent employ chain-of-thought reasoning before choosing arms so that they can transfer good strategies learned from other environments, **(2)** we let the agent interact with the environment in a multiturn way, **(3)** instead of reducing regret, we work on the bandit best arm selection (Audibert & Bubeck, 2010; Wang et al., 2024a) problem, where we let the agent choose arms and observe rewards for a fixed number of turns and then measure its accuracy in deciding the arm with the highest reward. This is done to reduce computational cost over generating COTs for a large number of turns, since the difference in regret between different models is not meaningful when the number of turns is not large enough.

3.2 Dataset construction

In order to learn from these environments, we must first generate data from them. It is crucial that the data we generate are diverse which would allow the model to learn different strategies without the risk of overfitting. We accomplish this by generating a large number of trajectories at a high temperature with Min-p sampling (Nguyen et al., 2024). Min-p sampling works by using an adaptive threshold $p_{\text{scaled}} \propto p_{\text{max}}$, where p_{max} is the highest probability predicted by the model on the next token, to truncate the vocabulary to only those tokens that have a probability larger than p_{max} and samples from them — this enables us to generate diverse yet coherent trajectories at a higher temperature.

For each task in a set of chosen tasks (e.g., uniformly sampled), we generate n_{sample} trajectories and then construct a preference pair (h_w, h_l) where h_w is the highest scoring trajectory (trajectory that succeeds and does so at the fewest number of turns) and h_l is randomly sampled from the lower scoring (failed or takes substantially more turns to succeed) trajectories. We choose h_l randomly instead of choosing the worst one to increase the diversity of our dataset. We treat h_w and h_l as proxies for desirable and undesirable behaviors. A dataset $\mathcal{D} = \left\{ \left(h^w, h^l\right)^{(i)} \right\}_{i=1}^{N}$ is a collection of such trajectory pairs.

3.3 Optimization

Supervised fine-tuning. If we take the winning episodes as the expert behavior, then we can discard the losing episode and maximize the likelihood of winning episodes:

$$\mathcal{L}_{\text{SFT}}(\mathcal{D}) = \mathbb{E}_{\mathcal{D}} \left[\frac{1}{\sum_{t=0}^{|h_w|} |a_t^w|} \sum_{t=0}^{|h_w|} \log \pi_\theta \left(a_t^w \mid h_{:t}^w \right) \right]. \tag{1}$$

where $|a|$ is the number of tokens for the agent response (discarding the environment generation). This is akin to rejection sampling fine-tuning (Gulcehre et al., 2023; Dong et al., 2023; Mukobi et al., 2023) seen in prior work.

Direct preference optimization. A popular approach for finetuning LLMs is DPO (Rafailov et al., 2024b) where one directly optimizes the Bradley-Terry model (Bradley & Terry, 1952) for preferences. In our setting, each trajectory consists of multiple rounds of interactions so the original DPO objective does not apply. We instead use a multi-turn version of DPO introduced in Rafailov et al. (2024a):

$$\mathcal{L}_{\text{DPO}}(\mathcal{D}) = \mathbb{E}_{\mathcal{D}}\left[\log\sigma\left(\sum_{t=0}^{|h^w|}\beta\log\frac{\pi_\theta(a_t^w \mid h_{:t}^w)}{\pi_{\text{ref}}(a_t^w \mid h_{:t}^w)} - \sum_{t=0}^{|h^l|}\beta\log\frac{\pi_\theta(a_t^l \mid h_{:t}^l)}{\pi_{\text{ref}}(a_t^l \mid h_{:t}^l)}\right)\right] \tag{2}$$

where a_t^w is the action token generated by the model at turn t in the preferred trajectory h^w. π_{ref} is the reference policy, for which we use the initial model. The main difference with standard DPO here is that we only want to calculate the loss on the action tokens — the log probability ratios of the environment generated tokens are not included in the loss.

We note that we use DPO because it is less memory intensive. DPO allows us to decouple the data collection and policy improvement steps and offload them on different machines. However, in principle, one could also employ online RL with more resources. Following prior work that shows the efficacy of online RL compared to offline algorithms (Xu et al., 2024; Tajwar et al., 2024), we expect doing PAPRIKA with online RL would lead to even stronger results.

Combining objectives. Finally, prior works have noted DPO having the unintended effect of reducing the probability of preferred trajectories as well, knowns as unintentional unalignment (Razin et al., 2024), which can affect model performance. The RPO objective (Pang et al., 2024), by combining SFT and DPO loss, has shown promising results in mitigating this issue. Formally, we have:

$$\mathcal{L}_{\text{RPO}}(\mathcal{D}) = \mathcal{L}_{\text{DPO}}(\mathcal{D}) + \alpha\mathcal{L}_{\text{SFT}}(\mathcal{D}) \tag{3}$$

where α is a hyperparameter. Following Pang et al. (2024), we set α to be 1.0 for this paper.

3.4 SCALABLE ONLINE CURRICULUM LEARNING

The core idea of PAPRIKA is to fine-tune the model on a large number of decision making problems to acquire general decision making ability. It is relatively easy to design a large number of tasks, but it is harder to decide which task to train on. A major obstacle is that different tasks may have a large range of difficulty. Unlike pre-training where the model can generally make progress on any given sample (i.e., decrease next-token prediction loss), an RL agent cannot make meaningful progress without collecting good experience. As such, if a task is too difficult for the current model, the model would not generate trajectories with meaningful learning signals. Since generating a trajectory is expensive, it stands to reason that we want to prioritize the tasks where the model can make meaningful progress, which is a form of curriculum learning (Bengio et al., 2009).

Without additional assumptions, the only way to know whether a task would yield good learning signals is to actually perform a rollout in that task, which is expensive. In fact, in this particular scenario, the major cost for training is actually the data generation rather than model updates. As such, this naive approach would not save us time or computation. A desideratum for an efficient curriculum is the ability to know whether certain tasks will yield data with learning signals without actually performing the rollout. A natural assumption is that similar tasks would have similar levels of learning signal. These groupings can be obtained through meta data or prior knowledge.[2]

Measuring learning potential. We will use $h \sim \pi \circ \tau$ to denote sampling one episode from the task τ using the policy π. The average performance of π on τ is $R_\pi(\tau) = \mathbb{E}_{h\sim\pi\circ\tau}[r(h)]$ and the

[2]While this requirement may seem restrictive, we believe assumptions of similar effects are likely needed for any form of curriculum learning to be computationally efficient.

Algorithm 1 Task selection with UCB

Input: Number of arms K, batch size B, number of samples C, number of rounds T
Initialize: $s_k = 0$, $n_k = 0$, `Buffer`
for each round $t = 1, 2, \ldots, T$ **do**
  Compute $\theta_k = \frac{s_k}{n_k} + \sqrt{\frac{2 \log \sum_{k=1}^{K} n_k}{n_k}}$ for each k
  Select $k^\star = \arg\max_k \theta_k$
  Sample τ from each group $k^\star$
  Sample C trajectories from τ and add to `Buffer`
  Compute an estimate for $\hat{\nu}_\pi(\tau)$ using Eq 4
  Update: $s_{k^\star} = s_{k^\star} + \hat{\nu}_\pi(\tau)$, $n_{k^\star} = n_{k^\star} + 1$
end for
Construct $\mathcal{D}$ from `Buffer` and update the model π

variance is $\sigma_\pi^2(\tau) = \mathbb{E}_{h \sim \pi \circ \tau}\left[(r(h) - R_\pi(\tau))^2\right]$. Based on these, we can define:

$$\nu_\pi(\tau) = \frac{\sqrt{\sigma_\pi^2(\tau)}}{R_\pi(\tau)}. \tag{4}$$

This quantity is known as the coefficient of variation in statistics, a dimensionless quantity that measures the population's variability relative to the mean.

We argue that this quantity is an ideal measure of the learning potential for a single task. DPO requires a pair of positive and negative samples [3]. Intuitively, the pair should be sufficiently different so the model can tell the two apart. Variance naturally measures the possibility of getting diverse trajectories from sampling. On the other hand, different tasks could have vastly different reward scales. Without loss of generality, if we assume that all rewards are positive, the average reward of each task is a measurement of the reward scale. Normalizing the standard deviation with the reward scale allows us to compare different tasks directly.

Sampling tasks. Each group contains a large number of different tasks. Since it is infeasible to evaluate $\nu_\pi(\tau)$ for all tasks, we instead sample tasks from the group. This induces a scalar distribution that describes the distribution of $\nu_\pi(\tau)$ for all tasks in the group G. Given a collection of K groups $(G_1, \ldots, G_K)$, a reasonable objective would be to maximize the learning potential of the tasks sampled. This problem can be formulated as a multi-armed bandit (MAB). Many algorithms for MAB exist; for simplicity, we choose the Upper Confidence Bound (Auer, 2000, UCB).

We conduct the task selection in a sequential manner using the original UCB algorithm, but we expect a batched variant of UCB could be used to parallelize the experience collection. Each action corresponds to a group of tasks, and we then uniformly sample one task from the chosen group to evaluate the model performance with C rollouts. These statistics are then used to update the mean estimate of that group. After a sufficient amount of episodes are sampled, we construct the dataset and train the model with objectives in Section 3.3. See Algorithm 3.4 for the pseudocode.

Note. An important role of ν_π is to make different environments comparable. The specific selection algorithms could likely be replaced with other more sophisticated online learning methods. More importantly, recent breakthroughs such as OpenAI et al. (2024b) and DeepSeek-AI et al. (2025) mark the beginning of applying RL to a broad range of reasoning problems. Moving forward, we anticipate a proliferation of different RL environments for LLMs. In this emerging paradigm, a scalable meta algorithm for selecting which environments to train on will be essential, and we believe PAPRIKA's curriculum learning approach will be a promising foundation for future algorithms.

4 EXPERIMENTS

In this section, we will present the results of our empirical study to answer the following research questions: **(1)** Can training on self-generated trajectories from a diverse range of tasks equip LLMs

[3] We hypothesize this quantity would also apply to online RL since if all sampled trajectories have the same reward the policy gradient update would be 0.

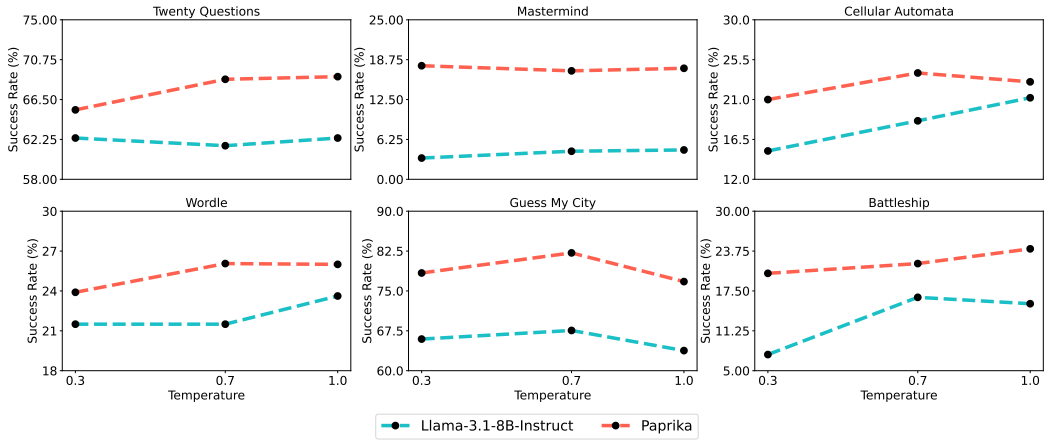

Figure 2: **(PAPRIKA improves success rate on a diverse range of environments)** Success rate (best-of-4) on 6 environments. PAPRIKA improves performance on all individual environments after fine-tuning on only roughly 20,000 total trajectories.

with sequential decision making capabilities that generalize to unseen tasks without the need to train on them? **(2)** Can curriculum learning improve the data efficiency of our training mechanism? **(3)** Finally, does PAPRIKA hurt the model on regular tasks, and can fine-tuning on existing multi-turn interaction data that do not have any sequential decision making structure also improve these capabilities? We first describe our experimental setup, and then describe our empirical observations.

Experimental Setup We use a Llama-3.1-8B-Instruct model (MetaAI et al., 2024) for all our experiments. For data generation, we use Min-p sampling (Nguyen et al., 2024) with temperature 1.5 and p_{max} 0.3, as we saw that this setting consistently generated diverse data that resulted in higher test-time accuracy. For each task in the training split, we generate $n_{\text{sample}} = 20$ trajectories to construct our training dataset. After filtering, this results in 15,744 training trajectories for supervised fine-tuning and 4,384 trajectory pairs for RPO. Unless explicitly mentioned otherwise, we use learning rate of 10^{-6} for supervised fine-tuning and 2×10^{-7} for RPO. We use batch size 32 for all training runs. We generally always run supervised fine-tuning first and then further fine-tune it with the RPO objective to obtain the final model unless explicitly mentioned otherwise. We use an AdamW optimizer (Loshchilov & Hutter, 2019) with a cosine annealing learning rate scheduler and warmup ratio 0.04 (Loshchilov & Hutter, 2017) to train all our models.

During evaluation, in order to account for variability of both the environment and the agent, we generate 4 trajectories for each task in the test set and take the best of them to measure success rate (i.e. best-of-4). we use min-p parameter 0.3 and temperature 0.7 unless performance for other temperatures are also reported.

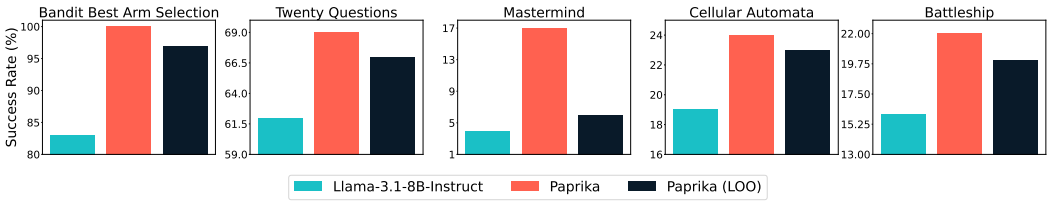

Figure 3: **(Testing generalization of PAPRIKA via leave-one-out experiments)** We test PAPRIKA's zero-shot performance on unseen tasks by leave-one-out (LOO) experiments, where we train the LLM on trajectories from every task except the task we test on. All evaluations are done at temperature 0.7 and we report the best-of-4 success rate. This demonstrates that PAPRIKA can teach an LLM strategic exploration and sequential decision making abilities that transfer well to new tasks without additional training needed.

PAPRIKA improves LLM decision making abilties We motivate this question by looking into the toy environment of bandit best arm selection more closely. This task requires strategic use of the fixed sampling budget (20) to quickly discard arms that are unlikely to have a high mean reward, and use majority of the sampling budget on the few top arms to decide the best arm among them.

Prior work (Nie et al., 2024) has shown that training on synthetic trajectories from optimal bandit algorithms can significantly improve LLMs' performance on them. Contrary to that, we show that LLMs can learn generalizable strategies from other decision making tasks that then transfer to this bandit task, without needing an optimal algorithm to generate synthetic trajectories to then finetune the LLMs on. 3 (left) shows that PAPRIKA improve best-of-4 success rate from 82.5% to 97.5% on the bandit task after only seeing trajectories from other tasks, and including trajectories from the bandit task in the training set can increase success rate to 100%.

Motivated by this, we next study whether PAPRIKA can also improve performance on more complex tasks. 2 shows our main findings: PAPRIKA, when trained on a dataset consisting of filtered trajectories from all 10 tasks, improve the success rate of the regular instruct model on all of them by a significant margin (please see 5 for complete results). PAPRIKA also improves the average number of turns it takes to achieve success on all tasks — 6 shows the improvement in task efficiency achieved by our fine-tuning method. Averaged across all 10 environments, PAPRIKA increase the model's performance by 47% of its original success rate with only about 20,000 trajectories.

PAPRIKA can teach LLMs generalizable strategies The next important question we want to study is if the strategies learned by PAPRIKA can zero-shot transfer to entirely different groups of tasks. We saw already that PAPRIKA improved the success rate on the bandit group without the need to train on it, now we explore this possibility for more complex decision making tasks. To do so, we perform a set of leave-one-out (LOO) experiments: we randomly choose one group (e.g., 20 questions) from our set of environments, train the LLM on trajectories generated from every other group, and test the resulting model's performance on the left-out group. 3 shows our results on a representative set of tasks: PAPRIKA improves success rate on all of them and maintains a similar level of performance gain compared to the model trained on all groups, except one environment (mastermind), where it still outperforms the instruct model. This confirms our hypothesis that PAPRIKA is a viable solution for teaching LLMs to do in-context RL.

Curriculum learning can improve data efficiency of PAPRIKA The biggest bottleneck of PAPRIKA is the time required to generate a large number of trajectories for each. Some tasks are naturally harder than others, which means that spending an equal sampling budget on the harder tasks gives us a smaller learning signal. We study a curriculum learning version of PAPRIKA where we have a grouping over our tasks according to task difficulty. For this, we use GPT-4o-mini to classify the tasks in twenty questions into 3 categories: easy, medium, and hard. This results in 477 easy, 727 medium, and 296 hard questions in the train split and 127 easy, 172 medium, and 68 hard questions in the test split.

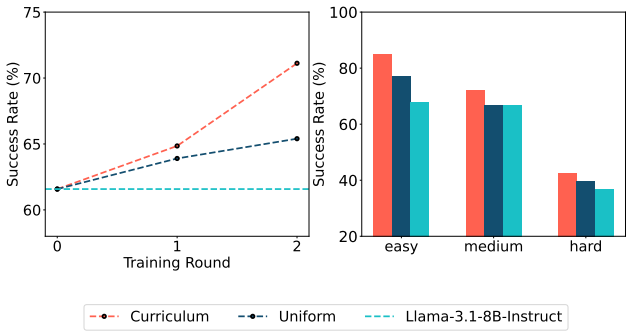

Figure 4: **(Multi-round training with curriculum on twenty questions)** We demonstrate the efficacy of our curriculum learning algorithm for sampling training tasks by comparing its performance against uniform sampling for multi-round training. (*Left*) Success rate (best-of-4) in each round at temperature 0.7. Our curriculum shows strong improvement over the uniform sampling baseline. (*right*) Success rate per category.

Next, we run the curriculum learning algorithm described in 3.4 for 2 rounds: at each round, we sample 500 tasks from the train set according to 3.4. We use the number of turns it took the agent to solve a task across sampled trajectories as a proxy for reward in 4 to calculate ν_π, see D for more details. We generate 20 trajectories for each task using the previous round's model checkpoint and train that checkpoint on the resulting dataset (for DPO, we use the prior round's checkpoint instead of the initial model as the reference policy). We compare our curriculum against the baseline of sampling 500 tasks uniformly at random from the train set at each round. 4 shows our results: after two rounds of training with our curriculum, the test accuracy reaches 71% whereas training with uniform sampling only reaches 65%.

5 Related Works

LLM alignment. Alignment or post-training is a crucial step for creating helpful LLM assistant. Existing post-training pipeline typically involves instruction tuning and then reinforcement learning from human feedback (Christiano et al., 2017, RLHF) where one either performs RL against a reward model trained on human preference data via Proximal Policy Optimization (Schulman et al., 2017, PPO) or sidesteps reward model training via Direct Preference Optimization (Rafailov et al., 2024b, DPO). Most methods focus on *single-turn* interactions where the model generates a single response to a query. We focus on the *multi-turn* setting where the agent has to interact with an environment iteratively. There are a few existing environments and datasets that focus on multi-turn interactions (Abdulhai et al., 2023; Sun et al., 2023; Kwan et al., 2024; Wang et al., 2024b). LMRL-Gym (Abdulhai et al., 2023) implements a suite of textual RL environment, some of which we build on. Most of these environments focus on interactions with humans. Rather than any particular tasks, We focus on evaluating the general ability to solve a sequential decision making problem where the agent needs to explore (e.g., gather necessary information for a task) and exploit (e.g., solving a task in an efficient manner).

In-context reinforcement learning. In-context learning (ICL) is the ability where LLMs can learn a new task from a small number of demonstrations without any gradient update (Brown et al., 2020). Existing ICL usually focuses on a single-turn interaction. We focus on in-context reinforcement learning (Laskin et al., 2022; Lee et al., 2024; Lin et al., 2024) instead. Existing work in this field has focused on environments where RL is conventionally applied (e.g., grid world, bandits, and maze), and the training data are generated by either random policies or pre-existing RL algorithms. In comparison, we focus on diverse environments and study how well the decision making abilities generalize to completely new environments.

Curriculum learning in RL. Curriculum learning (Bengio et al., 2009) shows the data to the model in a non-uniform order. This idea is inspired by the fact that humans tend to learn skills in a sequential order (Skinner, 1958), and is particularly appealing for RL because learning easier tasks first could build scaffold toward solving difficult tasks that the agent could not solve otherwise (Andrychowicz et al., 2017; Florensa et al., 2017; Fang et al., 2019; Portelas et al., 2020a). Another related line of work is environment design where a second process controls the distribution over different environments or directly generates the details of the environments in a procedural manner to maximize some notion of learning progress (Wang et al., 2019; Dennis et al., 2020; Jiang et al., 2021b;a; Bruce et al., 2024). Since this is a field of extensive existing literature, we refer interested reader to Portelas et al. (2020b) for a comprehensive survey.

6 Discussion

In this paper, we presented a scalable fine-tuning method to improve information gathering and decision making abilities of LLMs. Moreover, we showed that the strategies learned by the LLM from our method can generalize zero-shot to unseen tasks. There are a few limitations of our approach. Firstly, we use rejection sampling on self-generated data to teach the model better behaviors. In order to get good performance, the starting model need to exhibit good behavior within a reasonable generation budget, so Paprika would perform worse in the absence of a good base model. Next, we use offline preference tuning algorithms to train our models due to lack of computational resources needed for online reinforcement learning. A possible future direction for our work is to run online RL on diverse tasks instead: due to its recent success in other domains (DeepSeek-AI et al., 2025), we expect it will give a larger improvement in LLM capabilities and lead to generally curious agents that can perform in-context RL on a large number of tasks without being explicitly trained on them. Finally, our environments, despite being designed with the help of GPT-4o-mini, required a lot of human effort for implementation. A new axis of improvement for Paprika can be training an LLM to scalably generate suitable tasks that can then be used to train the agent. We leave these directions for future work.

REFERENCES

Marwa Abdulhai, Isadora White, Charlie Snell, Charles Sun, Joey Hong, Yuexiang Zhai, Kelvin Xu, and Sergey Levine. Lmrl gym: Benchmarks for multi-turn reinforcement learning with language models. *arXiv preprint arXiv:2311.18232*, 2023.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

Jean-Yves Audibert and Sébastien Bubeck. Best Arm Identification in Multi-Armed Bandits. In *COLT 2010 - Proceedings*, pp. 13 p., Haifa, Israel, June 2010. URL https://enpc.hal.science/hal-00654404.

Peter Auer. Using upper confidence bounds for online learning. In *Proceedings 41st annual symposium on foundations of computer science*, pp. 270–279. IEEE, 2000.

Jacob Beck, Risto Vuorio, Evan Zheran Liu, Zheng Xiong, Luisa Zintgraf, Chelsea Finn, and Shimon Whiteson. A survey of meta-reinforcement learning. *arXiv preprint arXiv:2301.08028*, 2023.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48, 2009.

Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, et al. Genie: Generative interactive environments. In *Forty-first International Conference on Machine Learning*, 2024.

Jiuhai Chen, Rifaa Qadri, Yuxin Wen, Neel Jain, John Kirchenbauer, Tianyi Zhou, and Tom Goldstein. Genqa: Generating millions of instructions from a handful of prompts, 2024. URL https://arxiv.org/abs/2406.10323.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Matthew Cook et al. Universality in elementary cellular automata. *Complex systems*, 15(1):1–40, 2004.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. *Advances in neural information processing systems*, 33:13049–13061, 2020.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, KaShun SHUM, and Tong Zhang. RAFT: Reward ranked finetuning for generative foundation model alignment. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=m7p5O7zblY.

Meng Fang, Tianyi Zhou, Yali Du, Lei Han, and Zhengyou Zhang. Curriculum-guided hindsight experience replay. *Advances in neural information processing systems*, 32, 2019.

Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. In *Conference on robot learning*, pp. 482–495. PMLR, 2017.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, Wolfgang Macherey, Arnaud Doucet, Orhan Firat, and Nando de Freitas. Reinforced self-training (rest) for language modeling, 2023. URL https://arxiv.org/abs/2308.08998.

Matthew Hausknecht, Prithviraj Ammanabrolu, Marc-Alexandre Côté, and Xingdi Yuan. Interactive fiction games: A colossal adventure, 2020. URL https://arxiv.org/abs/1909.05398.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Minqi Jiang, Michael Dennis, Jack Parker-Holder, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Replay-guided adversarial environment design. *Advances in Neural Information Processing Systems*, 34:1884–1897, 2021a.

Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. In *International Conference on Machine Learning*, pp. 4940–4950. PMLR, 2021b.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

Akshay Krishnamurthy, Keegan Harris, Dylan J. Foster, Cyril Zhang, and Aleksandrs Slivkins. Can large language models explore in-context?, 2024. URL https://arxiv.org/abs/2403.15371.

Wai-Chung Kwan, Xingshan Zeng, Yuxin Jiang, Yufei Wang, Liangyou Li, Lifeng Shang, Xin Jiang, Qun Liu, and Kam-Fai Wong. Mt-eval: A multi-turn capabilities evaluation benchmark for large language models. *arXiv preprint arXiv:2401.16745*, 2024.

Michael Laskin, Luyu Wang, Junhyuk Oh, Emilio Parisotto, Stephen Spencer, Richie Steigerwald, DJ Strouse, Steven Hansen, Angelos Filos, Ethan Brooks, et al. In-context reinforcement learning with algorithm distillation. *arXiv preprint arXiv:2210.14215*, 2022.

Jonathan Lee, Annie Xie, Aldo Pacchiano, Yash Chandak, Chelsea Finn, Ofir Nachum, and Emma Brunskill. Supervised pretraining can learn in-context reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Licong Lin, Yu Bai, and Song Mei. Transformers as decision makers: Provable in-context reinforcement learning via supervised pretraining. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=yN4Wv17ss3.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts, 2017. URL https://arxiv.org/abs/1608.03983.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

MetaAI, Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Gabriel Mukobi, Peter Chatain, Su Fong, Robert Windesheim, Gitta Kutyniok, Kush Bhatia, and Silas Alberti. Superhf: Supervised iterative learning from human feedback, 2023. URL https://arxiv.org/abs/2310.16763.

Minh Nguyen, Andrew Baker, Clement Neo, Allen Roush, Andreas Kirsch, and Ravid Shwartz-Ziv. Turning up the heat: Min-p sampling for creative and coherent llm outputs. *arXiv preprint arXiv:2407.01082*, 2024.

Allen Nie, Yi Su, Bo Chang, Jonathan N. Lee, Ed H. Chi, Quoc V. Le, and Minmin Chen. Evolve: Evaluating and optimizing llms for exploration, 2024. URL https://arxiv.org/abs/2410.06238.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, et al. Gpt-4 technical report, 2024a. URL https://arxiv.org/abs/2303.08774.

OpenAI, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024b.

Richard Yuanzhe Pang, Weizhe Yuan, Kyunghyun Cho, He He, Sainbayar Sukhbaatar, and Jason Weston. Iterative reasoning preference optimization, 2024. URL https://arxiv.org/abs/2404.19733.

Rémy Portelas, Cédric Colas, Katja Hofmann, and Pierre-Yves Oudeyer. Teacher algorithms for curriculum learning of deep rl in continuously parameterized environments. In *Conference on Robot Learning*, pp. 835–853. PMLR, 2020a.

Rémy Portelas, Cédric Colas, Lilian Weng, Katja Hofmann, and Pierre-Yves Oudeyer. Automatic curriculum learning for deep rl: A short survey. *arXiv preprint arXiv:2003.04664*, 2020b.

Rafael Rafailov, Joey Hejna, Ryan Park, and Chelsea Finn. From r to q^*: Your language model is secretly a q-function, 2024a. URL https://arxiv.org/abs/2404.12358.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024b.

Noam Razin, Sadhika Malladi, Adithya Bhaskar, Danqi Chen, Sanjeev Arora, and Boris Hanin. Unintentional unalignment: Likelihood displacement in direct preference optimization, 2024. URL https://arxiv.org/abs/2410.08847.

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

Burrhus F Skinner. Reinforcement today. *American Psychologist*, 13(3):94, 1958.

Aleksandrs Slivkins. Introduction to multi-armed bandits, 2024. URL https://arxiv.org/abs/1904.07272.

Robert Sokal and F. Rohlf. Biometry : the principles and practice of statistics in biological research / robert r. sokal and f. james rohlf, 04 2013.

Yuchong Sun, Che Liu, Jinwen Huang, Ruihua Song, Fuzheng Zhang, Di Zhang, Zhongyuan Wang, and Kun Gai. Parrot: Enhancing multi-turn chat models by learning to ask questions. *arXiv preprint arXiv:2310.07301*, 2023.

Fahim Tajwar, Anikait Singh, Archit Sharma, Rafael Rafailov, Jeff Schneider, Tengyang Xie, Stefano Ermon, Chelsea Finn, and Aviral Kumar. Preference fine-tuning of llms should leverage suboptimal, on-policy data, 2024. URL https://arxiv.org/abs/2404.14367.

Po-An Wang, Ruo-Chun Tzeng, and Alexandre Proutiere. Best arm identification with fixed budget: A large deviation perspective, 2024a. URL https://arxiv.org/abs/2312.12137.

Rui Wang, Joel Lehman, Jeff Clune, and Kenneth O Stanley. Paired open-ended trailblazer (poet): Endlessly generating increasingly complex and diverse learning environments and their solutions. *arXiv preprint arXiv:1901.01753*, 2019.

Xingyao Wang, Zihan Wang, Jiateng Liu, Yangyi Chen, Lifan Yuan, Hao Peng, and Heng Ji. MINT: Evaluating LLMs in multi-turn interaction with tools and language feedback. In *The Twelfth International Conference on Learning Representations*, 2024b. URL https://openreview.net/forum?id=jp3gWrMuIZ.

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

Stephen Wolfram. Statistical mechanics of cellular automata. *Reviews of modern physics*, 55(3): 601, 1983.

Shusheng Xu, Wei Fu, Jiaxuan Gao, Wenjie Ye, Weilin Liu, Zhiyu Mei, Guangju Wang, Chao Yu, and Yi Wu. Is dpo superior to ppo for llm alignment? a comprehensive study, 2024. URL https://arxiv.org/abs/2404.10719.

Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. Wildchat: 1m chatgpt interaction logs in the wild, 2024. URL https://arxiv.org/abs/2405.01470.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023. URL https://arxiv.org/abs/2306.05685.

A IMPACT STATEMENT

Our work can be used to train large language models that have better strategic exploration and decision making capabilities, which can have potential impact in the real world if agentic systems become wide spread. Our experiments are conducted in relatively simple and controlled environments and it is an open question what kind of impacts truly agentic systems will have on society. Other than that, this paper presents work whose goal is to advance the field of Machine Learning. There are many potential overall societal consequences of our work, none of which we feel must be specifically highlighted here.

B REPRODUCIBILITY STATEMENT

We provide sufficient details about our implementation, hyperparameters, environment design and dataset construction in the main paper and the appendix to effectively reproduce the results in this paper. We plan to release our code, training datasets, scripts for evaluation and model weights.

C DETAILS ON ENVIRONMENT DESIGN

Twenty questions: Twenty questions challenges the player to identify a secret topic by asking up to 20 yes-or-no questions. The goal is to guess the topic in as few questions as possible by interpreting previous answers and strategizing to maximize information gained. Twenty questions has been studied in prior benchmarks such as LMRL-Gym (Abdulhai et al., 2023): here we expand upon their environment with a more diverse and difficult set of secret topics. Our secret topics come from a diverse range of scenarios, including famous people, historical events, scientific concepts, locations, etc. Each secret topic corresponds to a subtask, and we have generated a set of 1500 train and 367 test scenarios. In order to generate a diverse set of topics, we use prompting techniques from GenQA (Chen et al., 2024) on GPT-4o-mini. The topics to guess in our training and test sets are distinct from one another and also the set of topics included in LMRL-Gym (159 topics), which use as an additional evaluation set. We use GPT-4o-mini (Hurst et al., 2024; OpenAI et al., 2024a) as the environment to provide yes/no answers at every turn, and also as a judge to make sure the LLMs are only asking yes/no question at every turn, and provide task success label. We also maintain strict train and test set separation to accurately test generalization unlike previous works.

Guess my city: Following LMRL-Gym, this environment requires the player to guess a secret city after asking a maximum of 20 questions. But unlike twenty questions, the questions here can be broader than just yes/no questions, for example, "*What is your city most popular for?*" so long as the answer to the question does not reveal the name of the city directly. We generated a train set of 500 and test set of 185 distinct cities using GPT-4o-mini and GenQA (Chen et al., 2024) prompting techniques. In addition, we also evaluated our models on the list of 91 cities from LMRL-Gym, which does not overlap with our training and test set. We maintain strict train and test set separation.

Customer service: In this environment, we test for efficient directed exploration —- the LLM must act as a support agent who asks maximally informative questions to diagnose problems and minimize the number of interactions needed to resolve the customer's query. To do so, we simulate realistic troubleshooting scenarios ranging from electronic device issues to automobile maintenance. We use GPT-4o-mini to simulate a customer with limited technical expertise, and use another LLM to act as a customer service agent whose role is to listen to the feedback from the customer and suggest a sequence of actions that lead to solving the customer's problem in as few turns as possible.

Murder mystery: Text-based interactive fiction (IF) environments can be a good benchmark to test LLMs' decision making and interaction abilities. Inspired by Hausknecht et al. (2020), we design our murder mystery environment, where an LLM is given a crime scene with a possible list of suspects, witnesses, and clues, and it needs to take actions to uncover more information to successfully determine the culprit. The environments provided in Hausknecht et al. (2020) proved difficult to incorporate directly in our setup, since they have a predefined list of valid actions and uses text-based parsing on the LLM generation to match against the list, making it difficult for LLMs to

play the games. Instead, we use GPT-4o-mini to simulate the environment that can provide dynamic feedback to the agent's actions.

Wordle: Wordle tests an LLM's deductive reasoning abilities. The player must guess a secret 5-letter word within 6 attempts. After each guess, the environment provides feedback for each letter: correct letter in correct position, correct letter in wrong position, or letter not in the word. The player must use this feedback strategically to maximize information gained with each guess. We found that LLMs like GPT-4o-mini cannot generate accurate environment feedback for Wordle, so we use hardcoded rules to generate it instead. We also saw that prompting the LLM agent to do chain-of-thought reasoning before outputting its final guess significantly improves its performance, so we use that here unlike the environments above.

Cellular Automata: A key trait of LLM agents is the ability to code and refine based on interpreter feedback. To model this, we create a cellular automata-based environment. Here, a binary string (e.g., 1010) represents cells, and a transition rule defines a cell's next state based on itself and its neighbors (e.g., 100: 1 means a 0 cell with 1 and 0 neighbors turns into 1). We randomly select a transition rule (one of 256) and up to three input strings and their corresponding outputs generated by the transition rule. The LLM must infer the rule by analyzing input-output pairs. If its guess generates correct outputs, it wins; otherwise, it gets feedback and can refine its guess. The task ends in failure if the correct rule isn't found within six turns. We use chain-of-thought prompting for the agent and a hardcoded program to generate environment feedback.

Mastermind: Similar to Wordle, Mastermind challenges players to deduce a 4-digit secret code within 12 turns. After each guess, environment feedback indicates two values: the number of digits that are correct and in the right position (exact matches), and the number of digits that appear in the code but in wrong positions (partial matches). Players must use this feedback to iteratively refine subsequent guesses. We use chain-of-thought prompting for the agent and a hardcoded program to generate environment feedback.

Battleship: Battleship tests an LLM's ability to balance exploration and exploitation. The environment features a 2D square grid where three ships are hidden: a carrier (5 cells), a battleship (4 cells), and a destroyer (2 cells). Ships are placed horizontally or vertically. At each turn, the player targets one cell with a missile. The environment environment reports either a hit (including the ship type) or a miss. A ship sinks when all its cells are hit. The player must sink all ships within 20 turns. This environment environment requires grid exploration to locate ships and once located, exploitation in the form of targeted attacks to sink them. We use chain-of-thought prompting for the agent and a hardcoded program to generate environment feedback.

Minesweeper: We include minesweeper to test an LLM's sequential logical reasoning ability. The player interacts with a 2D rectangular grid containing hidden mines. At each turn, the player reveals one cell. The first move is always safe since mines are placed afterward. If a mine is revealed, the environment ends in failure. To win, the player must reveal all mine-free cells within 20 turns. When a cell is revealed, it displays a number indicating how many mines are in adjacent cells. If a revealed cell has no adjacent mines (shown as '0'), all neighboring mine-free cells are automatically revealed. We use chain-of-thought prompting for the agent and a hardcoded program to generate environment feedback.

Bandit Best Arm Selection: Multi-arm bandits are a classic test for an agent's ability to perform sequential decision making, which has been studied in prior works such as Krishnamurthy et al. (2024); Nie et al. (2024). In this environment, an LLM is presented with a hypothetical scenario where it can select arms at every turn and observe the reward chosen from a Bernoulli distribution with a fixed but unknown mean attached to that arm. We created a modified version of their environment with three key distinctions: 1) prior works operated on bandits in a single-turn fashion: at each turn, LLMs were given the problem setup and summarized history of past interactions within a single system prompt and asked to choose the next arm. In other words, LLMs had to generate a response given a single user prompt. Instead, our design employs multi-turn interactions, where the task description is given in the first turn, and later turns only provide rewards for the selected arm. 2) Prior works required the LLM to output only the chosen arm, whereas we employ chain-of-thought

prompting to let the LLM think before it chooses an arm. 3) Instead of minimizing regret over a long time horizon, we instead work on the bandit best arm selection problem, where the LLM gets to choose arms and observe rewards for 20 turns, and then is prompted to choose what it thinks is the arm with the highest mean reward. This is done mainly to control for context length, as we could not run inference for more than 20 turns without running into computational issues, and the observed regret between multiple models is too small if horizon length is 20. We randomize the arm rewards at every iteration. For evaluation, we use the same setup as Krishnamurthy et al. (2024), for training, we use GPT-4o-mini to generate 81 diverse scenarios that are similar to it but has randomly chosen arm names and hypothetical scenarios. We also note that if the two best arms have very close mean reward (for example, 0.7 and 0.65), then it can be very difficult to identify the best arm within 20 turns. Following Krishnamurthy et al. (2024); Nie et al. (2024), we set the mean reward of the best arm to be above a certain threshold over the mean rewards of the other arms. We randomly choose $\epsilon \in [0.1, 0.2]$, and set the mean reward of the best arm to be $0.5 + \epsilon$. For all other arms, we randomly choose their mean rewards from $[0, 0.5 - \epsilon]$.

D More Details on Curriculum Learning

To calculate Coefficient of variation on task t (in this case, a single secret topic in twenty questions), we generate $n = 20$ trajectories for this task. Let these trajectories be $\tau_1, \ldots, \tau_n$. Let $|\tau_i|$ be the number of turns it takes for the agent to succeed in the i-th trajectory — if the agent fails in the i-th trajectory, we set $\tau_i| = 20$, which is also the maximum number of turns in this environment. We use number of turns it takes the agent to solve the task as a proxy for reward, and measure the coefficient of variation on number of turns to compare different tasks.

Since we use a small number of trajectories, instead of using $\nu = \frac{s}{\bar{x}}$, where s and $\bar{x}$ is the sample mean and standard deviation of $|\tau_i|$ respectively, we assume the unbiased estimator for coefficient of variation for normally distributed data instead (Sokal & Rohlf, 2013):

$$\nu = \left(1 + \frac{1}{4n}\right) \frac{s}{\bar{x}}$$

E More Empirical Results

E.1 Ablations

Paprika does not hurt LLMs' regular capabilities. We have demonstrated the efficacy of Paprika and the curriculum learning algorithm in instilling decision making capabilities into LLMs efficiently. However, to scale up Paprika, one would potentially use online reinforcement learning on such decision making tasks, and an important question is whether Paprika fine-tuning would hurt the LLM's regular capabilities which might hinder scaling it up. To study this, we run standard MT-Bench (Zheng et al., 2023) evaluation on our fine-tuned model. 2 shows our findings: instead of degrading the instruct model's capabilities, Paprika results in MT-Bench score improvement.

Finetuning on regular multiturn data does not help. Another hypothesis behind Paprika's success is that the instruct model has seen comparatively fewer multiturn trajectories during training, and finetuning on such trajectories may naturally lead to performance improvement in sequential decision making tasks, making our complex data generation process unnecessary. To test this, we finetune a Llama-3.1-8B-Instruct model on 100,000 English language trajectories randomly sampled from WildChat (Zhao et al., 2024), which contains multiturn interactions between GPT-4 and human users. Our results in 2 show that this finetuning results in minor performance degradation on the MT-Bench task, but much larger performance degradation on our decision making tasks (5,6).

E.2 Success Rate Comparison with More Baselines

5 and 6 demonstrates that Paprika improve Llama-3.1-8B-Instruct model's performance on both success rate and task efficiency.

Table 2: **(Evaluation of PAPRIKA on regular tasks)** Evaluation of PAPRIKA vs the instruct model on MT-Bench. PAPRIKA improves performance on MT-Bench instead of degrading the model.

Model	MT-Bench Average Score
Llama-3.1-8B-Instruct	7.85
+ WildChat Finetuned	7.60
+ PAPRIKA	**8.16**

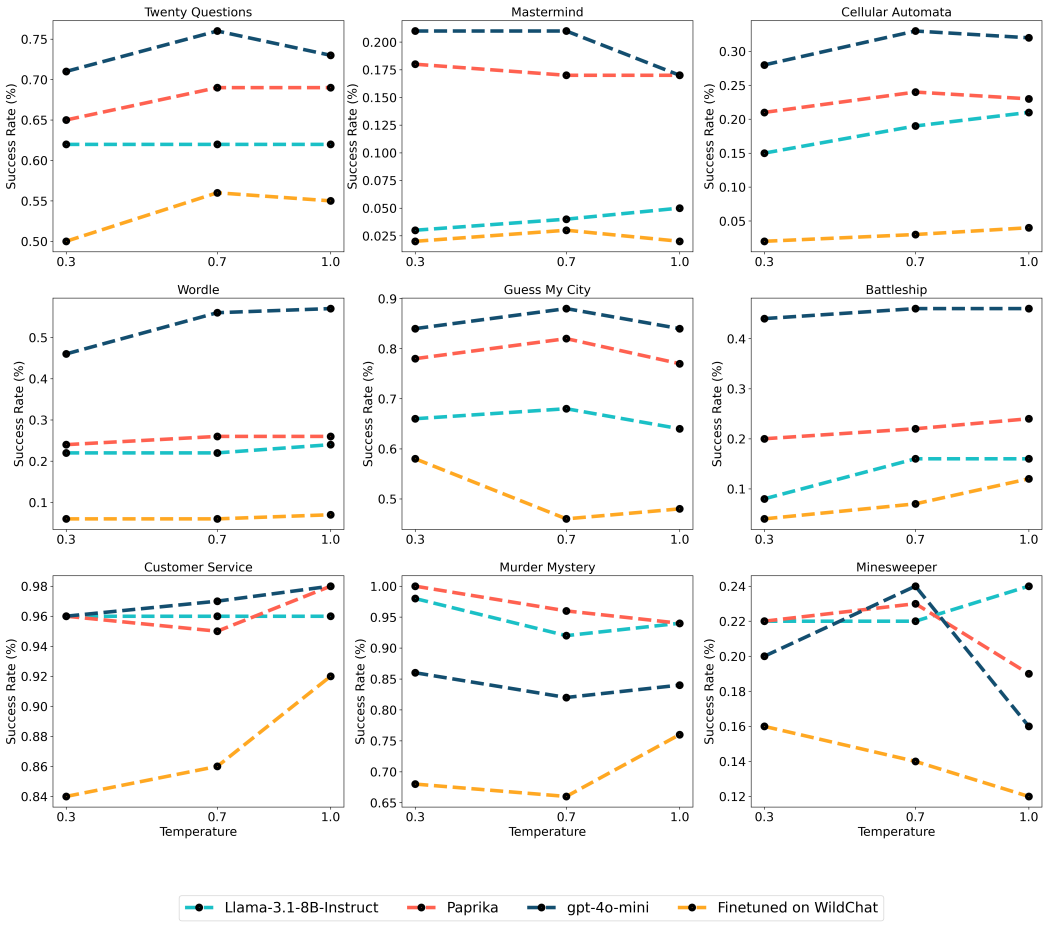

Figure 5: **(PAPRIKA improve success rate (best-of-4))** Success rate of PAPRIKA vs other models evaluated across temperatures 0.3, 0.7 and 1.0. See that PAPRIKA, when trained on trajectories from all environments, shows significant improvement across all of them. We also compare against a Llama-3.1-8B-Instruct model finetuned on 100,000 trajectories randomly sampled from the WildChat dataset. This model performs poorly on all tasks, which might be a result of model collapse.

E.3 EVALUATION ON LMRL-GYM SPLIT

In our paper, we construct a larger set of secret topics for twenty questions and guess my city, compared to LMRL-Gym (Abdulhai et al., 2023). Our training and evaluation sets are filtered to not have any overlap with their dataset. However, for the sake of fair comparison, we also report the performance of PAPRIKA on this dataset. 7 and 8 shows the performance of PAPRIKA on the LMRL-Gym split of guess my city and twenty questions, respectively.

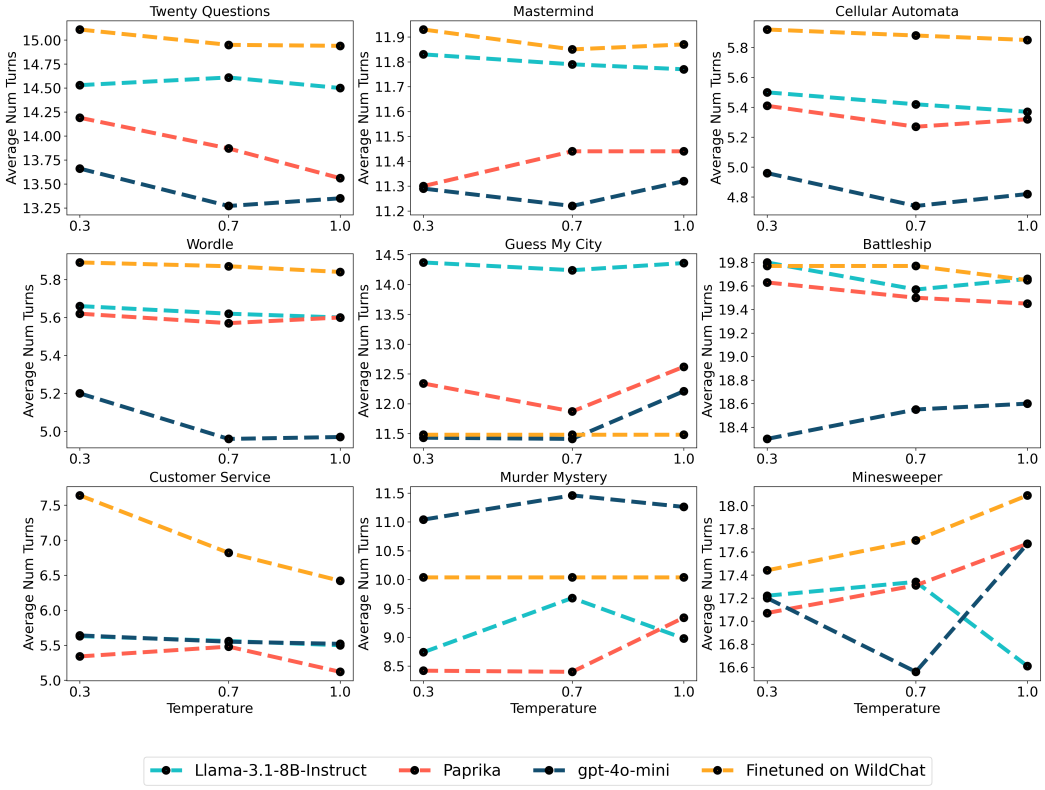

Figure 6: **(PAPRIKA improve task efficiency on all environments)** Average number of turns (on successful trajectories) of PAPRIKA vs other models, evaluated across temperatures 0.3, 0.7 and 1.0. Note that we do not measure number of turns on the bandit best arm identification task, since it is fixed to be 20. PAPRIKA reduce the average number of turns it takes an LLM to solve subtasks in all environments.

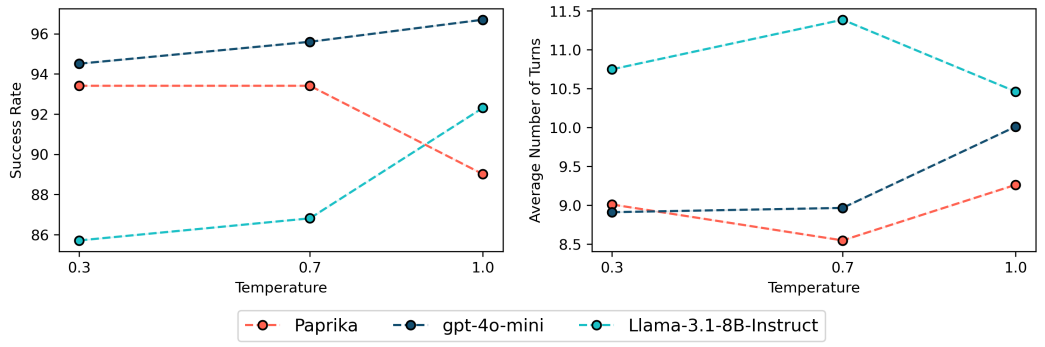

Figure 7: (PAPRIKA evaluated on guess my city, LMRL-Gym split) We evaluate our method on the LMRL-Gym split (disjoint from our training and test sets) for guess my city. We see that the gains we saw on our test set mostly translates to this dataset as well.

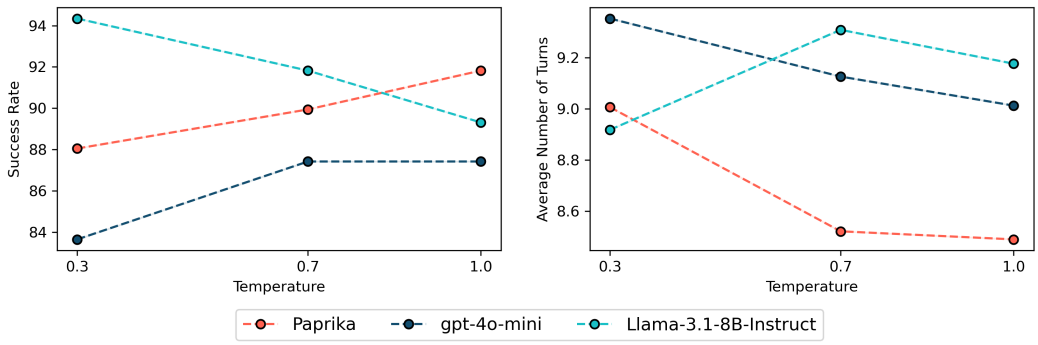

Figure 8: (PAPRIKA evaluated on twenty questions, LMRL-Gym split) We evaluate our method on the LMRL-Gym split (disjoint from our training and test sets) for twenty questions.