Matryoshka Pilot: Learning to Drive Black-Box LLMs with LLMs

Changhao Li[1†], **Yuchen Zhuang**[1†], **Rushi Qiang**[1], **Haotian Sun**[1],
Hanjun Dai[2,3*], **Chao Zhang**[1], **Bo Dai**[1,3]
[†]Equal Contribution, [1]Georgia Institute of Technology,
[2]Precur AI, [3]Google DeepMind
{cli911, yczhuang, rqiang6, haotian.sun}@gatech.edu
hanjun@precur.ai, chaozhang@gatech.edu, bodai@cc.gatech.edu

Abstract

Despite the impressive generative abilities of black-box large language models (LLMs), their inherent opacity hinders further advancements in capabilities such as reasoning, planning, and personalization. Existing works aim to enhance LLM capabilities via domain-specific adaptation, which require additional training on accessible model parameters, an infeasible option for black-box LLMs. To address this challenge, we introduce `Matryoshka Pilot` (`M-Pilot`), a lightweight white-box LLM controller that guides a large-scale black-box LLM generator by decomposing complex tasks into a series of intermediate outputs. Specifically, we consider the black-box LLM as an environment, with `M-Pilot` serving as a policy to provide intermediate guidance through prompts for driving the black-box LLM. `M-Pilot` is trained to pivot the outputs of the black-box LLM aligning with preferences during iterative interaction, which enables controllable multi-turn generation and self-improvement in optimizing intermediate guidance. Empirical evaluations on diverse tasks demonstrate that our method effectively enhances the capabilities of black-box LLMs in complex, long-horizon tasks.

1 Introduction

Most of the commercial large language models (LLMs) [29, 5, 1, 9, 46, 31] are *black-box* models [42, 63], where the model structure, parameters, or even output logits are not accessible. Although these black-box LLMs have exhibited remarkable efficacy across a diverse array of applications, revolutionizing natural language processing tasks such as text completion [29, 5], translation [62], question-answering [17], etc, the applications of black-box LLMs continue to face significant challenges when faced with tasks that require more advanced cognitive capabilities, particularly in the realms of reasoning [18, 50], planning [47, 64, 21, 26], and personalization problems [34, 43]. Enhancing such capabilities within black-box LLMs presents unique challenges, primarily due to the *lack of direct access to internal model parameters* [20, 42, 63]. This opacity introduces substantial complexity in efforts to refine and augment these advanced cognitive functions within the framework of black-box architectures.

Existing research efforts for improving black-box LLM performance can be largely categorized into two main methodological paradigms (Figure 1): (1) **In-context learning (ICL)-based methods** [41, 44, 63] that are designed to guide LLM in exhibiting specific capabilities or adhering to particular directives. However, these frameworks necessitate *meticulously constructing few-shot demonstrations or prompts* for LLMs to emulate or follow, which relies on heuristic prompts constructions. (2) **Adapter-based methods** [42, 63, 37] that exploit the inherent randomness in LLM generation,

[*]Work performed while at Google DeepMind.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

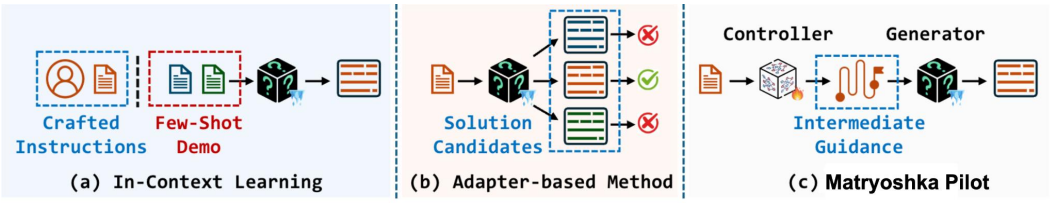


Figure 1: Enhancement in black-box LLMs capabilities. Existing methods either (a) integrate well-crafted instructions or meticulously-picked few-shot demonstrations as guidance or (b) exploit randomness in model generations to identify the most promising solution from candidates. In `M-Pilot`, we present (c) a controller-generator framework that enables white-box LLMs to drive the behavior of black-box LLMs for enhanced capabilities. 🔥 indicates the trainable parameters, whereas ❄️ indicates the inaccessible fixed parameters.

producing multiple candidate outputs and subsequently selecting those that optimally satisfy domain-predetermined criteria. Nevertheless, these approaches are highly dependent on the intrinsic synthetic capabilities or built-in functionalities of the black-box LLM, potentially resulting in the selection of a *suboptimal candidate* when all the generated options are less than ideal. Furthermore, both ICL and adapter-based methodologies exhibit significant limitations when applied to long-horizon tasks (*e.g.*, multi-step reasoning, long-term planning, *etc.*) due to their inherent lack of environmental interaction capabilities. In light of these constraints, we propose to leverage smaller, open-source LLMs as controllers to generate soft prompts as guidance, instead of relying on hard memory in context.

Similar to the scratchpad in `o1-preview`[2] [27], we propose `Matryoshka Pilot` (`M-Pilot`), a modular framework designed to enhance the advanced problem-solving capabilities of black-box LLMs via controllable multi-turn generations. `M-Pilot` consists of a lightweight white-box LLM that functions as a **controller** and a black-box LLM that serves as a **generator** or **solver**. Upon receiving the question description as input, the controller generates intermediate outputs that augment the capabilities of the subsequent black-box LLMs. For example, the controller can decompose the original complex task into high-level subtasks in reasoning or planning scenarios, or summarize profiles from historical records for personalization tasks. By conceptualizing the following **black-box LLM as the environment**, `M-Pilot` generates intermediate guidance alongside the original input to derive the final result through multi-turn interactions with the environment. The feedback for the outputs from the environments distinguishes positive and negative examples of intermediate generations, which can be used for preference optimization. Notably, this optimization process is inherently self-improving through iterative sampling from prior inferences and by considering the policies from earlier iterations as reference policies. `M-Pilot` continually enhances the advanced capabilities of the black-box LLM through controllable multi-turn generations that iteratively interact with environmental feedback.

Extensive experiments conducted on three complex tasks demonstrate the effectiveness and generalizability of `M-Pilot` in improving the advanced problem-solving capabilities of black-box LLMs, with an average improvement of 3.19% in accuracy for reasoning, 7.46% in success rate for planning, and 5.82% in accuracy for personalization. Importantly, `M-Pilot` not only enhances the capabilities of black-box LLMs without requiring access to model parameters, but also facilitates online feedback with environmental interactions. We summarize the main contributions:

- **i),** We introduce `M-Pilot`, one of the first modular frameworks that employ a lightweight white-box LLM to drive the generation of a large-scale black-box LLM for complex problem-solving;
- **ii),** `M-Pilot` intuitively formulates the white-box LLM as a controller and the black-box LLM as a component of the environment, facilitating long-horizon controllable generation with feedback;
- **iii),** `M-Pilot` adopts on-policy learning to iteratively enhance training data quality, inherently self-improving intermediate guidance for the continual enhancement of black-box LLM capabilities.

[2]Scratchpad is a sequence of intermediate chain-of-thoughts generated prior to producing the final answer. In `M-Pilot`, we broaden the definition of intermediate tokens to encompass various forms of guidance that can enhance the capabilities of LLMs, including task decomposition and user history summarization.

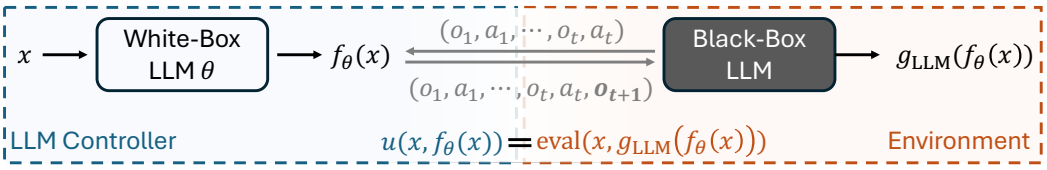


Figure 2: Controller-generator framework in `M-Pilot` comprising a white-box LLM as the controller and a black-box LLM as the generator and part of the environment. Given an input query x, `M-Pilot` leverages the intermediate generation $f_\theta(x)$ from the controller θ to drive the generator's behavior. The final answer is derived from the generation $y \sim g_{\text{LLM}}(f_\theta(x))$.

2 Problem Formulation

Our objective is to enhance the capability of a black-box LLM in solving complex, long-horizon problems by calibrating its output generation to better align with specific tasks. To achieve this, we conceptualize both the original outputs and the optimal solutions as distributions within a joint space, $\mathcal{Y} \sim \mathcal{Y}^{\text{org}} \times \mathcal{Y}^{\text{sol}}$, where $\mathcal{Y}^{\text{org}}$ and $\mathcal{Y}^{\text{sol}}$ represent the original text generations and target solutions, respectively. Specifically, given a set of task descriptions $\mathcal{D} = \{x_i\}_{i=1}^N$, our goal is to adjust the outputs $\hat{y}_i \in \mathcal{Y}^{\text{org}}$ of the black-box LLM toward the hidden target solutions $y_i \in \mathcal{Y}^{\text{sol}}$ that successfully solve the problems. This involves driving the black-box LLM to generate outputs more closely aligned with the desired solutions without access to parameters.

White-Box LLM Drives Black-Box LLMs. To enhance the capabilities of black-box LLMs in solving various tasks, we introduce a lightweight white-box language model as a controller. The process begins by feeding a text-grounded task description x from the task space $\mathcal{X}$ into a smaller language model θ, which acts as the controller. The smaller model generates a sequence of T-step intermediate guidance $\{g_t\}_{t=1}^T \sim f_\theta(x)$ to augment the performance of black-box LLMs on the specific task. These guidances can facilitate various functions, such as chain-of-thoughts for reasoning, task decomposition for planning, and user profile summarization from historical records for personalization. The generated intermediate guidance $\{g_t\}_{t=1}^T$ is then combined with the original problem description x and input $(x, \{g_t\}_{t=1}^T)$ into the black-box LLM to obtain the final prediction $\hat{y} \sim g_{\text{LLM}}(x, \{g_t\}_{t=1}^T)$. To formally characterize this process, we define a trajectory as:

$$\tau := \left(x, \{g_t\}_{t=1}^T, \hat{y}\right), \tag{1}$$

which encapsulates the task description x, the intermediate guidance sequence $\{g_t\}_{t=1}^T$, and the final prediction $\hat{y}$. Assuming an autoregressive generation process for both the intermediate guidance and final output, the conditional probability of the trajectory given the input x factorized as follows:

$$p(\tau|x) = p\left(\{g_t\}_{t=1}^T|x\right) p(\hat{y}|x, \{g_t\}_{t=1}^T) = \left(\prod_{t=1}^T p(g_t|x, \{g_l\}_{l=1}^{t-1})\right) p(\hat{y}|x, \{g_t\}_{t=1}^T), \tag{2}$$

Here, each intermediate step g_t explicitly depends on the task description x and all previously generated guidance steps, while the final prediction $\hat{y}$ is conditioned on the full set of guidance as well as the original input. Given the conditional trajectory distribution $p(\tau|x)$, we aim to maximize the likelihood gap between high-quality trajectories (τ^+) and lower-quality trajectories (τ^-). Formally this motivates the optimization of the contrastive objective:

$$\max_{p(y, \{g_t\}_{t=1}^T|x), \zeta \geq 0} \mathbb{E}_{\tau^+, \tau^-} \left[\log p(\tau_+|x) - \log p(\tau_-|x) - \zeta\right], \tag{3}$$

where τ^+ represents trajectories associated with desired outcomes, and τ^- denotes trajectories yielding suboptimal predictions. The margin term $\zeta \geq 0$ enforces a minimal desired gap between the two trajectory distributions, thus ensuring robust separation and effectively guiding the joint model toward generating improved intermediate guidance and predictions. Thus, we utilize the final correctness of the black-box LLM's output $\hat{y}$ to evaluate the quality of the trajectory $u(\tau)$ as the reward of the intermediate guidance produced by the white-box LLM controller (Figure 2):

$$u(\tau) := \text{eval}(\hat{y}, y), \tag{4}$$

where $\text{eval}(\cdot)$ denotes the oracle evaluation function of the final answer. For example, in question-answering tasks with ground-truth final answer y, the evaluation function measures accuracy by comparing the prediction with ground truth as $\text{eval}(\hat{y}, y) = \mathbb{1}(\hat{y} = y)$, where $\mathbb{1}(\cdot)$ is the indicator function. For planning tasks without a ground-truth solution, the evaluation function assesses the success rate after executing the final solution as $\text{eval}(\hat{y}, y) = \mathbb{1}_{\text{succ}}(\hat{y})$.

Multi-Turn Interaction. The above interaction between the white-box LLM controller and the black-box environment can be repeated for multi-turns for long-horizon tasks.

For initialization, a prompt x is sampled from task space $\mathcal{X}$ and serves as the initial state $s_0 = x$. At each subsequent step $t \in [T]$, the controller generates prompts a_t based on the current s_{t-1}. In response to the controllers action, the environment first initiates an **inner-loop multi-turn interaction**, where the black-box LLM iteratively adjusts execution steps based on the controller's guidance and returns an observation o_t based on the history s_{t-1} and the current action a_t. If the problem remains unsolved, the controller initiates an **outer-loop multi-turn interaction**, refining instructions through feedback to more effectively guide the LLM. Consequently, the state transitions are updated to include the new action and observation:

$$s_t = (s_{t-1}, a_t, o_t) = (x, a_1, o_1, s_1, \cdots, a_t, o_t), \tag{5}$$

and the next step begins. This process repeats for T rounds, resulting in a trajectory:

$$\tau = (x, a_1, o_1, s_1, \cdots, o_T, s_T), \tag{6}$$

and we obtain the reward for the whole trajectories, according to some eval$(\cdot)$.

The framework formulates a Markov Decision Process (MDP), which offers the potential for solving tasks that require long-horizon generations, including long-term planning and multi-step reasoning. By obtaining feedback from eval$(\cdot)$, we can conduct multi-turn optimization over the white-box LLM controller on the intermediate generations. Additionally, the multi-turn interaction with the environment during the data sampling stage can help improve data quality. Although optimizing this guidance presents challenges due to the inaccessibility of the black-box LLM's parameters that preclude backpropagation of gradients during training, the existing reinforcement learning techniques, *e.g.*, [35, 33], can be used for policy optimization.

3 `Matryoshka Pilot` (`M-Pilot`)

In this section, we specialize the white-box LLM controller that generates intermediate guidance to assist in task understanding and problem-solving in Section 3.1 and discuss the data collection procedure by interacting with black-box LLM in Section 3.2, which will be used for `M-Pilot` training to align the outputs of the black-box LLM with preferences in Section 3.3.

3.1 Instantiation of White-Box LLM Controller

We instantiate the white-box LLM as a controller to generate additional guidance that assists the black-box LLM in understanding and solving a diverse range of problems. Given varying complexity and distinct characteristics of different tasks, the controller should be capable of generating guidance in various formats. Examples of reasoning, planning, and personalization tasks are in Figure 3.

Problem Decomposition for Reasoning. For reasoning tasks, generating a sequence of reasoning steps is essential to solve the problem effectively. Existing works [61] have observed that models often perform poorly on tasks that require solving problems more complex than the exemplars provided in the prompts. To enable the model to better reason and overcome the easy-to-hard generalization issue, one strategy is to decompose complex problems into a series of simpler sub-problems and solve them sequentially. Therefore, for reasoning tasks, the white-box LLM controller outputs decomposed sub-tasks to assist the subsequent black-box LLM generator in enhancing its reasoning capabilities.

High-Level Plan for Planning. For planning tasks, LLMs are required to generate a sequence of actions that constitute a plan to solve the given problems. A common strategy [41, 60] is to apply hierarchical planning for complex solutions, where a high-level planner decomposes the task into sub-goals, and a low-level planner generates a sequence of admissible actions corresponding to each specific sub-goal. To enhance the black-box LLM's planning capabilities, we leverage the white-box controller to generate high-level plans as guidance for simplification.

User History Summarization for Personalization. For personalization tasks, LLMs are required to tailor outputs to individual users. Existing work [32] accomplishes this by concatenating the user's input query with a profile summarizing the user's preferences and behavior patterns. To enhance the black-box LLM's personalization capabilities, we utilize the white-box LLM controller to generate summaries of user histories. This approach enables black-box LLMs to better understand users and generate tailored content accordingly.

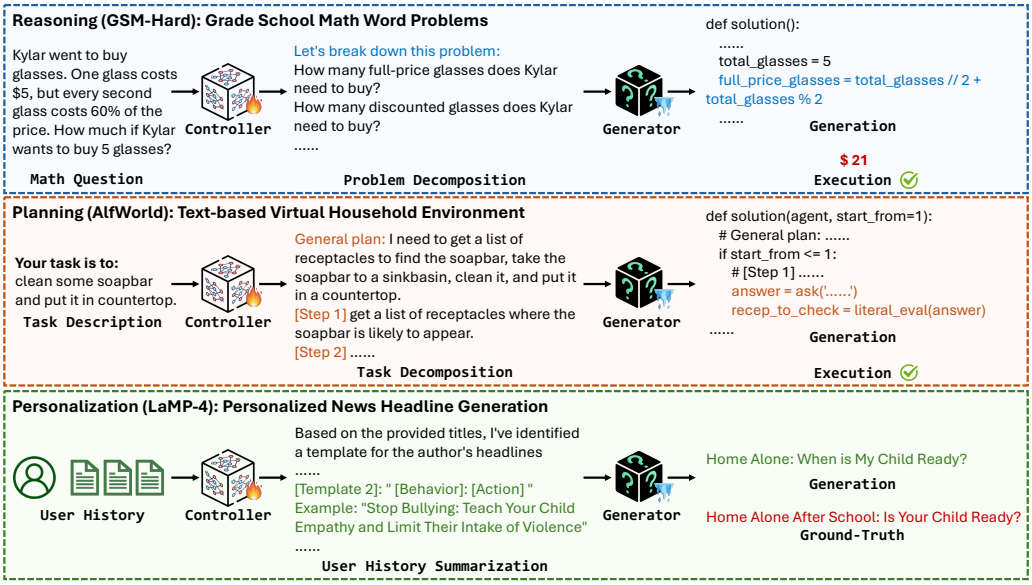


Figure 3: Examples of intermediate guidance generated by `M-Pilot` for complex reasoning, planning, and personalization tasks.

3.2 Data Collection by Interacting with Black-Box LLM Environment

Optimizing the intermediate guidance generated by the controller presents significant challenges for two main reasons: (1) *Lack of ground-truth guidance*: There are no ground-truth intermediate generations available to serve as supervision signals for the controller's outputs. (2) *Uncertainty in performance improvement*: It is difficult to determine which guidance will reliably enhance the downstream performance of the black-box LLM. To address these challenges, we formulate the black-box LLM as an environment system and employ multi-turn interactions with environmental feedback during data sampling.

In the MDP formulation, we consider the *action space* as the set of possible guidance that can enhance the capabilities of black-box LLMs. The *observation space* is determined by the oracle evaluation function for each task, defined as $\text{eval}(\cdot)$, where the sampled supervision signal is denoted as z, with $z = 1$ indicating that $f_\theta(x)$ is positive guidance while $z = 0$ indicating $f_\theta(x)$ negative guidance. During the multi-turn interactions, if the observation o_t at the t-th step returns a negative signal, the next action step a_{t+1} involves modifying the intermediate guidance based on the feedback. The interactions continue until a positive signal is observed or the maximum number of turns T is reached.

For each input x_i, we perform T-step multi-turn interactions with the black-box LLM-based environment to obtain the trajectories $(a_{i,1}, o_{i,1}, a_{i,2}, o_{i,2}, \cdots, a_{i,T}, o_{i,T})$. To increase the diversity of intermediate generations, we introduce randomness into the policy and repeat the entire interaction process K times. This results in K trajectories, yielding intermediate generations $\{\tau_{i,1}, \tau_{i,2}, \cdots, \tau_{i,K\times T}\}$ along with their corresponding observations $\{o_{i,1}, o_{i,2}, \cdots, o_{i,K\times T}\}$, which serve as sampling signals. We then sample the positive trajectory τ_i^+ from the set of guidance with positive observations, $\tau_i^+ \sim \{\tau_{i,j} | o_{i,j} = 1\}$ and the negative trajectory from the remaining generations, $\tau_i^- \sim \{\tau_{i,j} | o_{i,j} = 0\}$.

3.3 Iterative Direct Preference Optimization

As white-box LLMs like LLaMA are pre- and post-trained for general purposes, they may struggle to fulfill the specific tasks required by the controller. Additionally, there may be discrepancies between what the controller considers "good" guidance and what the generator interprets as "good" guidance. To this end, the guidance generated by the white-box LLM controller needs further optimization to enhance the performance of the black-box LLM generator.

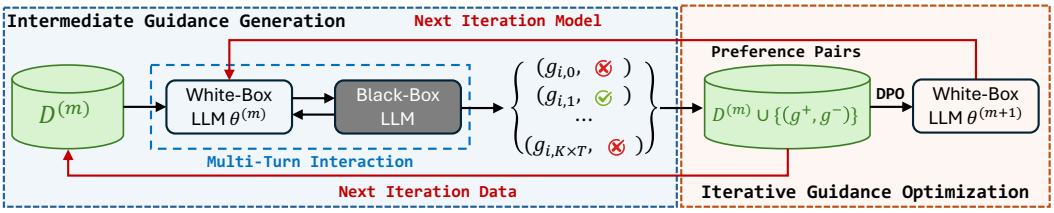


Figure 4: Overview of iterative guidance optimization. By iteratively updating both the model and the reference policy, `M-Pilot` progressively refines its intermediate guidance.

Supervised Fine-Tuning for Behavior Cloning. To quickly initialize the controller's policy, we adopt the concept of behavior cloning (BC) from reinforcement learning, which involves learning an initial policy by imitating the actions of an expert agent. This is typically achieved through supervised learning on a set of curated instruction-completion pairs for LLMs. We leverage the capabilities of more advanced models, such as `GPT-3.5` [36], to generate the desired guidance for the black-box LLMs on a **small set of** samples. This data is then used to perform supervised fine-tuning (SFT) on the white-box LLM controller as an initial warm-up step:

$$\mathcal{L}_{\text{SFT}} = -\mathbb{E}_{\tau}\big[\log p(\tau \mid x)\big] \quad = -\mathbb{E}_{\tau}\Big[\textstyle\sum_{t=1}^{T} \log p\big(g_t \mid x, \{g_\ell\}_{\ell=1}^{t-1}\big) \underbrace{+\log p\big(y \mid x, \{g_t\}_{t=1}^{T}\big)}_{\text{Non-learnable}}\Big]. \tag{7}$$

Through this SFT process, the white-box LLM controller begins to acquire the capability to effectively guide the subsequent black-box LLM. It can then be utilized to generate high-quality guidance for further optimization steps. Alternatively, if the initial white-box controller already demonstrates strong performance, we can directly skip the SFT process.

Iterative Preference Pair Collection. By allowing the warmed-up white-box LLM controller to interact with the black-box LLM environment over multiple turns, we can curate a dataset containing both "good" and "bad" guidance pairs from `M-Pilot`'s intermediate generations. However, an imbalance between positive and negative samples may arise, leading to overfitting on simplistic patterns and hindering the self-improvement of the white-box LLM controller. To address this issue, we propose an iterative guidance optimization method (Figure 4) that interleaves data sampling and training steps. We begin by initializing the model with parameters $\theta^{(0)} = \theta$ without any prior training and sample an initial dataset $\mathcal{D}^{(0)}$ as introduced in Section 3.2. At the m-th iteration, we have the optimized model $\theta^{(m)}$. Following STaR [57], we enhance the model's generation for the next iteration by bootstrapping the dataset. This involves combining the previous datasets with new trajectories $\{\tau_{i,0}^{(m)}, \tau_{i,1}^{(m)}, \cdots, \tau_{i,K\times T}^{(m)}\}$ generated by the current model $\theta^{(m)}$:

$$\begin{cases} \mathcal{D}_{+}^{(m)} = \{\tau_{i,j}^{(m)} | o_{i,j}^{(m)} = 1\} \cup \mathcal{D}_{+}^{(m-1)}, \\ \mathcal{D}_{-}^{(m)} = \{\tau_{i,j}^{(m)} | o_{i,j}^{(m)} = 0\} \cup \mathcal{D}_{-}^{(m-1)}. \end{cases} \tag{8}$$

In the m-th iteration, following reinforcement learning with human feedback (RLHF) [3, 28, 65], we construct the training dataset $\mathcal{D}^{(m)}$ by sampling positive and negative generated guidance that share the same prompt.

Iterative DPO. When training the model for the next iteration $\theta^{(m+1)}$, the preference signal is modeled using the Bradley-Terry model [4]. Given an input x and a generated guidance pair (g^+, g^-), the model specifies the probability of g^+ being chosen over g^- as:

$$p(\tau^+ \succ \tau^- | x) = \frac{\exp\big(u(\tau^+)\big)}{\exp(u(\tau^+)) + \exp(u(\tau^-))} = \sigma\left(u\left(\tau^+\right) - u\left(\tau^-\right)\right), \tag{9}$$

where $\sigma(x) = \frac{e^x}{(e^x+1)}$ is the logistic function. This formulation allows us to access sequence-level preferences to optimize the intermediate guidance generated by the white-box LLM controller. Following [30], we establish a connection between the white-box LLM controller and its associated optimal policy. Specifically, we consider the following KL-regularized planning problem with respect to a reference policy π_{ref}:

$$\max_{\theta} \mathbb{E}_x \mathbb{E}_\tau \left[u\left(\tau\right) - \eta^{-1} \mathbb{D}_{\text{KL}}\left[\pi_\theta\left(\tau|x\right) || \pi_{\text{ref}}\left(\tau|x\right)\right]\right].$$

The optimization problem above has a closed-form solution. For any guidance g, the optimal policy π^* is given by $\pi^*(g|x) \propto \pi_{\text{ref}}(g|x) \exp(\eta u(x, g))$. To enable the white-box LLM to self-improve,

Table 1: Main experimental results on LaMP benchmark. We utilize `gpt-4o-mini` as the black-box LLM generator for baselines and `M-Pilot`. R-1 and R-L refer to ROUGE-1 and ROUGE-L, respectively. k denotes the number of items retrieved. ↑ indicates that higher values are preferred, whereas (↓) signifies that lower values are better. The best score and second-best score for each task are emphasized in **bold** and underlined, respectively. IDPO represents Iterative Direct Preference Optimization. Notations are consistent across tables.

Dataset (→)	**LaMP-1**		**LaMP-2N**		**LaMP-2M**		**LaMP-3**		**LaMP-4**		
Method (↓)	Acc. ↑	F-1 ↑	Acc. ↑	F-1 ↑	Acc. ↑	F-1 ↑	MAE ↓	RMSE ↓	R-1 ↑	R-L ↑	BLEU ↑
`gpt-4o-mini`	0.514	0.513	0.655	0.473	0.413	0.325	0.371	0.673	0.132	0.116	0.992
RAG (k=1) [34]	0.626	0.624	0.733	0.539	0.444	0.378	0.311	0.631	0.141	0.126	1.296
RAG (k=4) [34]	0.632	0.632	0.792	0.611	0.502	0.430	**0.272**	**0.579**	0.161	0.146	2.953
PAG [32]	0.624	0.624	0.775	0.559	0.496	0.443	0.316	0.645	0.143	0.130	1.968
`M-Pilot`	**0.640**	**0.639**	**0.823**	**0.607**	**0.527**	**0.465**	0.277	0.581	**0.174**	**0.160**	**4.298**
w/o IDPO	0.611	0.611	0.807	0.575	0.496	0.432	0.311	0.636	0.131	0.120	1.341

we update the reference policy to be the model from the previous iteration, $\pi_{\text{ref}} = \pi_\theta^{(m)}(g|x)$. Consequently, the training objective for iterative guidance optimization of the white-box LLM controller becomes:

$$\mathcal{L}_{\text{IDPO}} := \mathbb{E}_{(x,\tau^+,\tau^-)\sim\mathcal{D}}\Big[-\log\sigma\Big(\eta^{-1}\Big(\log\Big(\frac{\pi_\theta^{(m+1)}(\tau^+|x)}{\pi_\theta^{(m)}(\tau^+|x)}\Big) - \log\Big(\frac{\pi_\theta^{(m+1)}(\tau^-|x)}{\pi_\theta^{(m)}(\tau^-|x)}\Big)\Big)\Big)\Big]. \tag{10}$$

Taking Eq. 2 into Eq. 10 leads to:

$$\frac{\pi_{\theta^{(m+1)}}(\tau^+|x)}{\pi_{\theta^{(m)}}(\tau^+|x)} = \frac{p_{\theta^{(m+1)}}\big(\{g_t^+\}_{t=1}^T|x\big)\cancel{p(\hat{y}^+|x,\{g_t^+\}_{t=1}^T)}}{p_{\theta^{(m)}}\big(\{g_t^+\}_{t=1}^T|x\big)\cancel{p(\hat{y}^+|x,\{g_t^+\}_{t=1}^T)}}. \tag{11}$$

Similar conclusions can be achieved for $\frac{\pi_{\theta^{(m+1)}}(\tau^-|x)}{\pi_{\theta^{(m)}}(\tau^-|x)}$. Thus, we can rewrite Eq. 10 as:

$$\mathcal{L}_{\text{IDPO}} := \mathbb{E}_{(x,\tau^+,\tau^-)\sim\mathcal{D}}\Big[-\log\sigma\Big(\eta^{-1}\Big(\log\Big(\frac{p_{\theta^{(m+1)}}\big(\{g_t^+\}_{t=1}^T|x\big)}{p_{\theta^{(m)}}\big(\{g_t^+\}_{t=1}^T|x\big)}\Big) - \log\Big(\frac{p_{\theta^{(m+1)}}\big(\{g_t^-\}_{t=1}^T|x\big)}{p_{\theta^{(m)}}\big(\{g_t^-\}_{t=1}^T|x\big)}\Big)\Big)\Big)\Big].$$

4 Experiments

4.1 Experimental Setup

Tasks and Datasets. We consider three types of tasks in experiments, each targeting a distinct capability of black-box LLMs: (1) *LaMP* [34] for personalization capabilities, (2) *GSM8K* [10] for reasoning capabilities, and (3) *ALFWorld* [39] for planning capabilities. For reasoning, We also test on more challenging *MATH* [18] dataset in Appendix B.2, with ablations on outer-loop multi-turn interactions. More tasks details are in Appendix D.

Baselines. We consider the following baselines: (1) *Baselines in personalization*, we consider both one-stage and two-stage personalization models, including Profile-Augmented Generation (PAG) [32] and Retrieval-Augmented Generation (RAG) [34]. (2) *Baselines in reasoning*, we include Chain-of-Thoughts (CoT) [51], Least-to-Most [61], Program-Aided Language Models (PAL) [15], and $\text{PAL}_{\text{Self-Debug}}$ [6]. (3) *Baselines in planning*, we mainly compare `M-Pilot` with BUTLER [39], ReAct [55], Reflextion [38], and AdaPlanner [41]. Baseline details can be found in Appendix E. Furthermore, we also include comparison with several other baselines in Appendix B.3

Evaluation Metrics. For personalization tasks, consistent with the evaluation metrics specified in LaMP [34], we use accuracy (*Acc*) and F1 score (*F1*) for the classification tasks in LaMP-2N and LaMP-2M. For the ordinal multi-class classification task, LaMP-3, we employ mean absolute error (MAE) and root mean squared error (*RMSE*). To comprehensively evaluate the personalized text generation tasks in LaMP-4 and LaMP-5, we report ROUGE-1 (*R-1*), ROUGE-L (*R-L*), and *BLEU* scores. For the math reasoning task, we assess the models based on the *accuracy* of obtaining the final correct answer. For the planning task, consistent with previous works [41], we evaluate performance using the *success rate* (%). The success rate is calculated as the number of successful episodes divided by the total number of episodes. In ALFWorld, an episode is considered a failure if the task remains unsolved after executing 50 actions, which is the maximum allowed actions.

Table 2: Plug-and-Play results for `gpt-3.5-turbo` and `gemini-1.5-flash` across the LaMP benchmark. We employ `M-Pilot` pre-trained on `gpt-4o-mini` as the white-box LLM controller.

Dataset (→)	**LaMP-1**		**LaMP-2N**		**LaMP-2M**		**LaMP-3**		**LaMP-4**		
Method (↓)	Acc. ↑	F-1 ↑	Acc. ↑	F-1 ↑	Acc. ↑	F-1 ↑	MAE ↓	RMSE ↓	R-1 ↑	R-L ↑	BLEU ↑
`M-Pilot (4o-mini)`	**0.640**	**0.639**	**0.823**	**0.607**	**0.527**	**0.465**	**0.277**	**0.581**	**0.174**	**0.160**	**4.298**
`gpt-3.5-turbo`	0.590	0.589	0.790	0.594	0.399	0.325	0.357	0.693	0.166	0.150	3.433
Plug-and-play (`gpt-3.5`)	**0.594**	**0.593**	**0.798**	**0.609**	0.469	0.412	**0.286**	**0.599**	**0.176**	**0.161**	**4.222**
w/o IDPO (`gpt-3.5`)	0.585	0.585	0.790	0.608	**0.472**	**0.425**	0.334	0.670	0.160	0.147	3.015
`gemini-1.5-flash`	0.518	0.510	0.700	0.498	0.368	0.279	0.546	0.825	0.135	0.113	1.494
Plug-and-play (`gemini`)	**0.573**	**0.565**	**0.825**	**0.615**	0.504	**0.418**	**0.298**	**0.614**	**0.183**	**0.170**	**5.002**
w/o IDPO (`gemini`)	0.568	0.561	0.811	0.602	**0.505**	0.411	0.365	0.715	0.164	0.150	3.439

Implementations. For the white-box LLM controller, we utilize `LLaMA-3-8B-Instruct` as the backbone language model, we also consider `Qwen2.5-7B-Instruct` as the backbone in Appendix B.5. In the black-box LLM environment, our experiments employ `gpt-4o-mini` for personalization tasks in LaMP, and `gpt-3.5-turbo` for reasoning and planning tasks in GSM8K and ALFWorld, respectively. Please refer to Appendix F for implementation details.

4.2 Personalization: LaMP

Main Results. Table 1 summarizes the primary experimental results on the LaMP dataset. Our proposed method, `M-Pilot`, consistently outperforms or matches other state-of-the-art baselines, highlighting its efficacy of advancing black-box LLMs in personalization. For classification tasks, `M-Pilot` achieves an accuracy of 0.832 on LaMP-2N and 0.535 on LaMP-2M, surpassing other baselines by a significant margin. For generation tasks, `M-Pilot` also attains over a 25% improvement in BLEU score on LaMP-4. These results demonstrate the effectiveness of `M-Pilot` in both classification and generative personalization tasks. Furthermore, `M-Pilot` has the potential to be enhanced with RAG, combining with the retrieved user history data to improve performance.

Plug-and-Play. `M-Pilot` can seamlessly apply the optimized white-box controller to other black-box models in a plug-and-play manner without additional training costs. We further utilize this well-tuned white-box controller as a plug-in to integrate with black-box models such as `gpt-3.5-turbo` and `gemini-1.5-flash`. Table 2 presents the plug-and-play results. The experimental results show that our well-tuned controller consistently outperforms other baselines. Specifically, on LaMP-3 and LaMP-4, our plug-in surpasses other baselines by a large margin, demonstrating effectiveness across both classification and generation tasks. The effectiveness of `M-Pilot` in plug-and-play scenarios arises from the generalization capability of intermediate guidance, which can benefit different black-box LLMs.

Ablation Studies. For ablation studies on LaMP, we compare our proposed method, `M-Pilot`, with a baseline lacking Iterative Direct Preference Optimization (IDPO) in Table 1. Using the same black-box model (`gpt-4o-mini`), our optimized white-box controller consistently and significantly outperformed the original `LLaMA-3-8B-Instruct`. These results demonstrate the effectiveness of IDPO in enhancing the white-box controller to generate more informative and higher-quality intermediate outputs, thereby guiding the black-box model toward better final answers. Further ablation studies on LaMP are provided in Appendix B.1.

4.3 Reasoning: GSM8K

Table 3 presents the main results on the GSM8K dataset. We employ a three-shot prompt design across all baselines, including ours. $\text{PAL}_{\text{Self-Debug}}$ refers to the addition of close-loop refinement to PAL during the inference stage. Our method consistently outperforms all baselines across the dataset, surpassing the strongest baseline, $\text{PAL}_{\text{Self-Debug}}$, by a margin of 6.7% when using the base LLM.

Table 3: Accuracy on the mathematical reasoning task using the GSM8K dataset.

Dataset (→)	**GSM8K**		**GSM-HARD**	
Method (↓)	`gpt-3.5`	`4o-mini`	`gpt-3.5`	`4o-mini`
CoT	0.809	0.932	0.406	0.500
Least-to-Most	0.811	0.908	0.425	0.498
PAL	0.802	0.920	0.638	0.748
$\text{PAL}_{\text{Self-Debug}}$	0.864	0.943	0.701	0.774
`M-Pilot`	**0.931**	**0.964**	**0.761**	**0.801**
w/o IDPO	0.896	0.954	0.729	0.780

This improvement stems from the optimized intermediate guidance generated by `M-Pilot`. Conditioned on this guidance, `M-Pilot` enables the black-box LLM to generate long-horizon solutions to solve the tasks. Similar to LaMP, `M-Pilot` trained with `gpt-3.5-turbo` can be seamlessly applied to other black-box models for solving mathematical problems on GSM8K without additional training costs. Notably, `M-Pilot` learns high-level planning abilities without focusing on specific details.

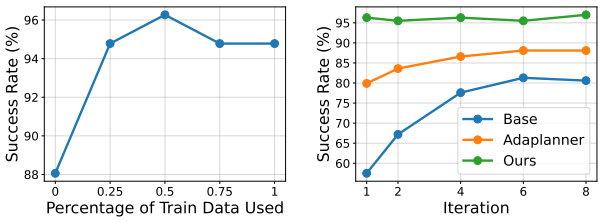
4.4 Planning: ALFWorld

Main Results. `M-Pilot` consistently outperforms existing baselines, achieving state-of-the-art performance with an overall success rate of 96.27% on ALFWorld tasks (Table 4). This superior performance indicates that `M-Pilot` effectively generates plans to guide the task execution of the black-box model, enhancing its ability to interact with the environment. Furthermore, we observe that `M-Pilot` exhibits superior performance compared to both the untuned white-box model (w/o Guidance Optimization) and the white-box models trained with fewer rounds of Iterative Direct Preference Optimization (w/o $1^{st}/2^{nd}$-round IDPO). As the number of IDPO training rounds increases, `M-Pilot`'s performance on ALFWorld correspondingly improves, ultimately raising the success rate from 81.34% to 96.27%. These results underscore the efficacy of the IDPO in `M-Pilot`.

Table 4: Success rate (%) across six planning tasks from AlfWorld. For all baselines, including `M-Pilot`, we utilize `gpt-3.5-turbo` as the black-box LLM.

Methods (↓) Tasks (→)	Pick	Clean	Heat	Cool	Exam	Pick2	All
BUTLER [39]	46.00	39.00	74.00	**100.00**	22.00	24.00	37.00
ReAct [55]	37.50	64.52	69.57	42.86	38.89	17.65	47.76
Reflexion [38]	50.00	41.94	65.22	52.38	66.67	47.06	52.99
AdaPlanner [41]	**100.00**	**93.55**	78.26	95.24	66.67	**88.24**	88.06
`M-Pilot`	**100.00**	**93.55**	**100.00**	95.24	**100.00**	**88.24**	**96.27**
w/o 2^{nd}-round IDPO	100.00	93.55	100.00	100.00	83.33	88.24	94.78
w/o 1^{st}, 2^{nd}-round IDPO	100.00	93.55	86.96	95.24	55.56	88.24	88.06
w/o Guidance Optimization	100.00	93.55	91.30	85.71	11.11	88.24	81.34

Ablation Studies on Sample Efficiency. Figure 5(a) illustrates the relationship between success rate (%) and the proportion of training data used to optimize the controller. In the ALFWorld environment, `M-Pilot` achieves an accuracy of 94.78% using only one-quarter of the training data, surpassing the best-performing baseline, AdaPlanner, by 6.7%. This demonstrates the sample efficiency of `M-Pilot` in achieving high performance with limited training data. This study demonstrates that `M-Pilot` significantly reduces the reliance on high-quality task planning samples and expert trajectories, improving the resource efficiency.



(a) Sample Efficiency. (b) Number of Turns M.

Figure 5: Success rate (%) *w.r.t* number of (a) training samples and (b) inner-loop interaction turns.

As illustrated in Figure 5(b), following DPO training, `M-Pilot` achieves an accuracy exceeding 95% in the open-loop inference setting (inner loop M=1), significantly surpassing both AdaPlanner and `LLaMA3-8B-Instruct`. Furthermore, during an 8-iteration closed-loop inference, `M-Pilot` maintains the highest accuracy of 97%. These findings indicate that `M-Pilot` is capable of generating exceptionally high-quality plans, enabling the GPT model serving as the executor to interact with the environment and complete tasks successfully without closed-loop refinement.

5 Related Works

Black-Box LLMs Generation Enhancement. Existing approaches aiming to enhance the generation capabilities of black-box LLMs can be broadly categorized into two groups: (1) *ICL-* and (2) *adapter-based* methods. ICL-based methods [41, 43, 63] are designed to augment the original query with well-crafted instructions or well-constructed few-shot demonstrations to guide the model. While this enables the black-box LLM to exhibit specific capabilities, these methods require significant

human effort in prompt engineering and result in prompts that are rigid and static. Adapter-based methods [42, 37, 63] follow a best-of-N selection evaluation paradigm [23], which evaluate N candidate solutions with a lightweight adapter and identify the highest-scoring solution as the final answer. However, such methods are heavily dependent on the generative capabilities of the black-box LLM, which may result in selecting a suboptimal candidate as *the best of a bad bunch*.

Reinforcement Learning for Prompt Optimization. As LLMs scale, new capabilities emerge, enabling models to learn tasks efficiently through a few in-context demonstrations. To harness these capabilities, several approaches have been proposed to leverage reinforcement learning for improved prompt generation, enhancing LLM performance. RLPrompt [11] introduces an RL-based framework for generating optimal prompts via black-box optimization. Similarly, TEMPERA [59] formulates prompt optimization as test-time prompt editing, using RL to efficiently explore the editing space. BDPL [12] further advances this by proposing a variance-reduced policy gradient algorithm to estimate gradients of parameters in the categorical distribution of each discrete prompt. However, these methods primarily focus on classification tasks, where gradient estimation is straightforward, limiting their applicability to more complex generation tasks requiring long-horizon solutions.

6 Conclusion

We introduced `Matryoshka Pilot` (`M-Pilot`), a lightweight white-box LLM controller designed to augment the capabilities of large-scale black-box LLMs across a wide range of complex tasks, including reasoning, planning, and personalization. By leveraging a controller-generator framework with environmental feedback, `M-Pilot` effectively decomposes complex tasks and guides black-box LLMs through intermediate guidance. Through policy gradient optimization, `M-Pilot` exhibits a self-improving nature that continually enhances LLM capabilities via multi-turn guidance optimization. Extensive experiments on three diverse datasets demonstrate its effectiveness in steering black-box LLMs for long-horizon tasks without requiring access to model parameters or output probabilities. Compared to the best-performing state-of-the-art baselines, `M-Pilot` achieves average improvements of 3.19% in reasoning tasks, 7.46% in planning tasks, and 5.82% in personalization tasks. These results underscore the potential `M-Pilot` as a transparent and scalable solution, enabling white-box LLMs to drive black-box LLMs in complex problem-solving.

7 Acknowledgments and Disclosure of Funding

We thank the anonymous reviewers and area chairs for their valuable feedback. This work was supported in part by the ONR under grant N000142512173, and by the NSF under grants ECCS: 2401391 and IIS: 2403240.

References

[1] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] M. G. Azar, Z. D. Guo, B. Piot, R. Munos, M. Rowland, M. Valko, and D. Calandriello. A general theoretical paradigm to understand learning from human preferences. In *International Conference on Artificial Intelligence and Statistics*, pages 4447–4455. PMLR, 2024.

[3] Y. Bai, A. Jones, K. Ndousse, A. Askell, A. Chen, N. DasSarma, D. Drain, S. Fort, D. Ganguli, T. Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

[4] R. A. Bradley and M. E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

[5] T. B. Brown. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[6] X. Chen, M. Lin, N. Schärli, and D. Zhou. Teaching large language models to self-debug. *arXiv preprint arXiv:2304.05128*, 2023.

[7] Z. Chen, Y. Deng, H. Yuan, K. Ji, and Q. Gu. Self-play fine-tuning converts weak language models to strong language models, 2024.

[8] L. Choshen, L. Fox, Z. Aizenbud, and O. Abend. On the weaknesses of reinforcement learning for neural machine translation. In *International Conference on Learning Representations*, 2019.

[9] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

[10] 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.

[11] M. Deng, J. Wang, C.-P. Hsieh, Y. Wang, H. Guo, T. Shu, M. Song, E. Xing, and Z. Hu. Rlprompt: Optimizing discrete text prompts with reinforcement learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3369–3391, 2022.

[12] S. Diao, Z. Huang, R. Xu, X. Li, L. Yong, X. Zhou, and T. Zhang. Black-box prompt learning for pre-trained language models. *Transactions on Machine Learning Research*, 2023.

[13] L. Engstrom, A. Ilyas, S. Santurkar, D. Tsipras, F. Janoos, L. Rudolph, and A. Madry. Implementation matters in deep policy gradients: A case study on ppo and trpo. *arXiv preprint arXiv:2005.12729*, 2020.

[14] K. Ethayarajh, W. Xu, N. Muennighoff, D. Jurafsky, and D. Kiela. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*, 2024.

[15] L. Gao, A. Madaan, S. Zhou, U. Alon, P. Liu, Y. Yang, J. Callan, and G. Neubig. Pal: Program-aided language models. In *International Conference on Machine Learning*, pages 10764–10799. PMLR, 2023.

[16] C. Gulcehre, T. L. Paine, S. Srinivasan, K. Konyushkova, L. Weerts, A. Sharma, A. Siddhant, A. Ahern, M. Wang, C. Gu, et al. Reinforced self-training (rest) for language modeling. *arXiv preprint arXiv:2308.08998*, 2023.

[17] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

[18] 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.

[19] A. Hosseini, X. Yuan, N. Malkin, A. Courville, A. Sordoni, and R. Agarwal. V-star: Training verifiers for self-taught reasoners, 2024.

[20] Y. Huang, D. Liu, Z. Zhong, W. Shi, and Y. T. Lee. k nn-adapter: Efficient domain adaptation for black-box language models. *arXiv preprint arXiv:2302.10879*, 2023.

[21] C. E. Jimenez, J. Yang, A. Wettig, S. Yao, K. Pei, O. Press, and K. Narasimhan. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*, 2023.

[22] J. Kim and Y. Yang. Few-shot personalization of llms with mis-aligned responses. *arXiv preprint arXiv:2406.18678*, 2024.

[23] H. Lightman, V. Kosaraju, Y. Burda, H. Edwards, B. Baker, T. Lee, J. Leike, J. Schulman, I. Sutskever, and K. Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

[24] T. Q. Luong, X. Zhang, Z. Jie, P. Sun, X. Jin, and H. Li. Reft: Reasoning with reinforced fine-tuning. *arXiv preprint arXiv:2401.08967*, 2024.

[25] Y. Meng, M. Xia, and D. Chen. Simpo: Simple preference optimization with a reference-free reward. *arXiv preprint arXiv:2405.14734*, 2024.

[26] G. Mialon, C. Fourrier, C. Swift, T. Wolf, Y. LeCun, and T. Scialom. Gaia: a benchmark for general ai assistants. *arXiv preprint arXiv:2311.12983*, 2023.

[27] OpenAI. Introducing openai o1-preview. *OpenAI Blog*, 2024.

[28] 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.

[29] 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.

[30] R. Rafailov, A. Sharma, E. Mitchell, C. D. Manning, S. Ermon, and C. Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024.

[31] M. Reid, N. Savinov, D. Teplyashin, D. Lepikhin, T. Lillicrap, J.-b. Alayrac, R. Soricut, A. Lazaridou, O. Firat, J. Schrittwieser, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

[32] C. Richardson, Y. Zhang, K. Gillespie, S. Kar, A. Singh, Z. Raeesy, O. Z. Khan, and A. Sethy. Integrating summarization and retrieval for enhanced personalization via large language models. *arXiv preprint arXiv:2310.20081*, 2023.

[33] C. Rosset, C.-A. Cheng, A. Mitra, M. Santacroce, A. Awadallah, and T. Xie. Direct nash optimization: Teaching language models to self-improve with general preferences. *arXiv preprint arXiv:2404.03715*, 2024.

[34] A. Salemi, S. Mysore, M. Bendersky, and H. Zamani. Lamp: When large language models meet personalization. *arXiv preprint arXiv:2304.11406*, 2023.

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

[36] J. Schulman, B. Zoph, C. Kim, J. Hilton, J. Menick, J. Weng, J. F. C. Uribe, L. Fedus, L. Metz, M. Pokorny, et al. Introducing chatgpt. *OpenAI Blog*, 2022.

[37] W. Shi, R. Xu, Y. Zhuang, Y. Yu, H. Wu, C. Yang, and M. D. Wang. Medadapter: Efficient test-time adaptation of large language models towards medical reasoning. *arXiv preprint arXiv:2405.03000*, 2024.

[38] N. Shinn, B. Labash, and A. Gopinath. Reflexion: an autonomous agent with dynamic memory and self-reflection. *arXiv preprint arXiv:2303.11366*, 2023.

[39] M. Shridhar, X. Yuan, M.-A. Côté, Y. Bisk, A. Trischler, and M. Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020.

[40] A. Singh, J. D. Co-Reyes, R. Agarwal, A. Anand, P. Patil, P. J. Liu, J. Harrison, J. Lee, K. Xu, A. Parisi, et al. Beyond human data: Scaling self-training for problem-solving with language models. *arXiv preprint arXiv:2312.06585*, 2023.

[41] H. Sun, Y. Zhuang, L. Kong, B. Dai, and C. Zhang. Adaplanner: Adaptive planning from feedback with language models. *Advances in Neural Information Processing Systems*, 36, 2024.

[42] H. Sun, Y. Zhuang, W. Wei, C. Zhang, and B. Dai. Bbox-adapter: Lightweight adapting for black-box large language models. In *Forty-first International Conference on Machine Learning*, 2024.

[43] Z. Tan, Q. Zeng, Y. Tian, Z. Liu, B. Yin, and M. Jiang. Democratizing large language models via personalized parameter-efficient fine-tuning. *arXiv preprint arXiv:2402.04401*, 2024.

[44] Z. Tan, Q. Zeng, Y. Tian, Z. Liu, B. Yin, and M. Jiang. Democratizing large language models via personalized parameter-efficient fine-tuning, 2024.

[45] Y. Tang, Z. D. Guo, Z. Zheng, D. Calandriello, R. Munos, M. Rowland, P. H. Richemond, M. Valko, B. A. Pires, and B. Piot. Generalized preference optimization: A unified approach to offline alignment. In *Forty-first International Conference on Machine Learning*, 2024.

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

[47] K. Valmeekam, A. Olmo, S. Sreedharan, and S. Kambhampati. Large language models still can't plan (a benchmark for llms on planning and reasoning about change). In *NeurIPS 2022 Foundation Models for Decision Making Workshop*, 2022.

[48] P. Wang, L. Li, L. Chen, F. Song, B. Lin, Y. Cao, T. Liu, and Z. Sui. Making large language models better reasoners with alignment. *arXiv preprint arXiv:2309.02144*, 2023.

[49] 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.

[50] Y. Wang, X. Ma, G. Zhang, Y. Ni, A. Chandra, S. Guo, W. Ren, A. Arulraj, X. He, Z. Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *arXiv preprint arXiv:2406.01574*, 2024.

[51] 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.

[52] Y. Wu, Z. Sun, H. Yuan, K. Ji, Y. Yang, and Q. Gu. Self-play preference optimization for language model alignment. *arXiv preprint arXiv:2405.00675*, 2024.

[53] C. Xu, Y. Xu, S. Wang, Y. Liu, C. Zhu, and J. McAuley. Small models are valuable plug-ins for large language models. *arXiv preprint arXiv:2305.08848*, 2023.

[54] C. Yang, X. Wang, Y. Lu, H. Liu, Q. V. Le, D. Zhou, and X. Chen. Large language models as optimizers. *arXiv preprint arXiv:2309.03409*, 2023.

[55] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. R. Narasimhan, and Y. Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

[56] Z. Yuan, H. Yuan, C. Li, G. Dong, K. Lu, C. Tan, C. Zhou, and J. Zhou. Scaling relationship on learning mathematical reasoning with large language models, 2023.

[57] E. Zelikman, Y. Wu, J. Mu, and N. Goodman. STar: Bootstrapping reasoning with reasoning. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[58] 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.

[59] T. Zhang, X. Wang, D. Zhou, D. Schuurmans, and J. E. Gonzalez. TEMPERA: Test-time prompt editing via reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023.

[60] Q. Zhao, H. Fu, C. Sun, and G. Konidaris. Epo: Hierarchical llm agents with environment preference optimization. *arXiv preprint arXiv:2408.16090*, 2024.

[61] D. Zhou, N. Schärli, L. Hou, J. Wei, N. Scales, X. Wang, D. Schuurmans, C. Cui, O. Bousquet, Q. V. Le, et al. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*, 2023.

[62] W. Zhu, H. Liu, Q. Dong, J. Xu, S. Huang, L. Kong, J. Chen, and L. Li. Multilingual machine translation with large language models: Empirical results and analysis. *arXiv preprint arXiv:2304.04675*, 2023.

[63] Y. Zhuang, H. Sun, Y. Yu, Q. Wang, C. Zhang, and B. Dai. Hydra: Model factorization framework for black-box llm personalization. *arXiv preprint arXiv:2406.02888*, 2024.

[64] Y. Zhuang, Y. Yu, K. Wang, H. Sun, and C. Zhang. Toolqa: A dataset for llm question answering with external tools. *Advances in Neural Information Processing Systems*, 36:50117–50143, 2023.

[65] D. M. Ziegler, N. Stiennon, J. Wu, T. B. Brown, A. Radford, D. Amodei, P. Christiano, and G. Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims are supported by our methodology 3 and experiments 4.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations in Appendix A.3.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: We provide the assumptions and proof in Section 3.3.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide experimental settings and configurations in Section 4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide the code in supplementary materials.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so No is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide experimental setting/details both in Section 4 and Appendix F.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We report results without error bars or confidence intervals.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide experiments compute resources detailed in Appendix F.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper discuss both potential positive societal impacts and negative societal impacts in Appendix A.2.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We provide citations to all datasets and models used in the paper.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: New assets introduced in the paper (mainly code) is well documented together.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [No]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Impact Statement

A.1 Ethical Aspects

We strictly followed the data usage guidelines for interactions with Gemini API and ChatGPT API service. Although our research relied solely on publicly available datasets, we took extra precautions to minimize any potential risk of information leakage. Specifically, we opted out of the human review process by completing and submitting the Additional Use Case Form[3]. This proactive measure highlights our commitment to maintaining the highest data privacy standards and ethical research practices, especially concerning personalization tasks.

A.2 Future Societal Consequences

Potential Positive Societal Impacts. The proposed `M-Pilot` framework addresses a critical challenge in consistently enhancing the capabilities of black-box LLMs for long-horizon tasks with broad scopes. By improving reasoning, planning, and personalization, `M-Pilot` can deliver significant benefits across various domains. For instance, it can provide insights into complex theorems, advance industrial automation, and offer more personalized interactions for end users. Overall, `M-Pilot` has the potential to facilitate more useful, relevant, and satisfying interactions, thereby improving productivity, decision-making, and quality of life. Moreover, `M-Pilot` operates without requiring access to the model weights of black-box LLMs, making the technology accessible to a wide range of off-the-shelf LLM APIs and enabling seamless integration into diverse use cases. By leveraging existing LLMs, `M-Pilot` can be readily adopted by researchers, developers, and organizations, accelerating the development and deployment of advanced language models in real-world applications.

Potential Negative Societal Impacts. Enhancing black-box LLMs through a small-scale white-box LLM introduces potential risks. One significant concern is the possibility of using the white-box model to jailbreak black-box LLMs, injecting malicious instructions or producing harmful content. This could lead to the spread of misinformation, hate speech, or other offensive materials, with severe consequences for individuals and society. Additionally, this approach poses a threat to user data privacy. Training the white-box model requires collecting and storing interaction data between the black-box LLM and the environment, which could be improperly handled or misused, potentially compromising sensitive information.

A.3 Limitations

In this study, we propose a modular framework, `M-Pilot`, that leverages a lightweight white-box LLM controller to enhance the capabilities of black-box LLMs. Despite its effectiveness, we have identified several potential limitations of `M-Pilot`:

Malign Usage. Since `M-Pilot` employs a white-box LLM controller to augment black-box LLMs, there are notable risks to consider. Malicious actors could exploit this approach to engineer harmful capabilities or generate toxic content for training purposes. While black-box LLMs are designed to resist producing such content, our controller could be misused to manipulate these models into generating undesirable outputs. Furthermore, there is a risk that the intermediate guidance produced by our controller could be exploited to extract sensitive information from black-box LLMs, potentially facilitating jailbreaking or other targeted attacks.

Data Privacy. `M-Pilot` preserves the confidentiality of training data by avoiding third-party API sharing, thereby safeguarding the integrity of training samples during the enhancement process of black-box LLMs. However, when applied to personalization tasks, it is important to recognize that retrieved historical records or the queries themselves may inadvertently contain sensitive information, potentially risking unintended disclosure of private data.

B Additional Experiments

B.1 Further Ablation Studies on LaMP

[3] `https://aka.ms/oai/additionalusecase`

To further investigate the effect of user profile count on the generation of intermediate outputs, we analyze performance across different numbers of profiles per user. Figure 6 presents the accuracy and ROUGE-L curves separately for LaMP-2M and LaMP-4, with the x-axis representing the total number of profiles per user (*e.g.*, "0-20" indicates users with 0 to 20 profiles). We compared the results of our proposed method, `M-Pilot`, and PAG, utilizing the white-box controller `Llama-3-8B-Instruct` and the black-box model `gpt-4o-mini`.

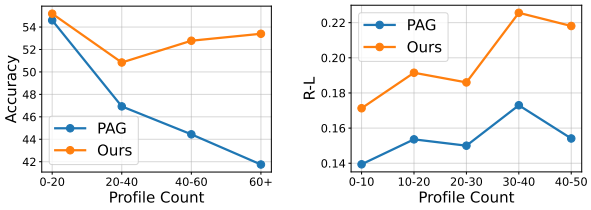


Figure 6: Effect of # history per user in LaMP-2M and -4.

On LaMP-2M, as the profile count increases, PAG's performance significantly deteriorates, whereas `M-Pilot` maintains stable performance and surpasses PAG by an increasing margin. For LaMP-4, both `M-Pilot` and PAG exhibit similar trends, but `M-Pilot` consistently outperforms PAG by a substantial and steady margin. These results demonstrate the efficacy of IDPO in enhancing the summarization capabilities of the black-box controller, especially when dealing with varying and plenty of profiles.

B.2 Experiments on challenging MATH500

In this section, we show that `M-Pilot` can generalize well to other more challenging tasks, MATH500 [17] for instance. As shown in Figure. 7, even with single-turn outer-loop interactions, our method already significantly outperforms the black-box only approach. This is because the controller model provides high-quality intermediate guidance, directing the black-box LLM to adjust its execution steps more effectively toward the correct direction. Moreover, when employing multi-turn outer-loop interactions, performance is further enhanced, demonstrating the controller's ability to refine its instructions through feedback for more efficient guidance of the black-box LLM.

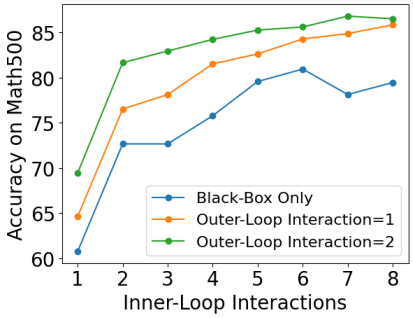


Figure 7: Examples of `M-Pilot` performance on MATH500 dataset.

B.3 Comparison with other baselines

We additionally conduct a comprehensive comparison with several state-of-the-art baselines across various domains to demonstrate the effectiveness and generalizability of our proposed framework. In personalization tasks, ORPO [54] optimizes prompts by iteratively generating and evaluating solutions using LLMs without training, while Fermi [22] personalizes LLMs by iteratively refining user-specific prompts based on profiles and misaligned responses; For reasoning tasks, LEAP [58] improves prompts by learning from previous mistakes, whereas Bbox-Adapter [42] employs Noise Contrastive Estimation(NCE) to train an adapter that guides the policy more effectively.

Table 5: Comparison with baselines on LaMP.

Tasks (↓) **Methods** (→)	**Fermi** [22]	**ORPO** [54]	`M-Pilot` **(Ours)**
LaMP-2M ↑	37.8	34.3	**47.0**
LaMP-3 ↓	34.0	57.0	**29.0**

Table 6: Comparison with baselines on GSM8K.

Task (↓) **Methods** (→)	**LEAP(low-level)** [58]	**LEAP(high-level)** [58]	**Bbox-Adapter** [42]	`M-Pilot` **(Ours)**
GSM8K	77.4	76.6	74.94	**93.1**

We show the comparison results in Table 5 and Table 6, where all baseline results are reported as in their original papers, and `gpt-3.5-turbo` is used consistently across all experiments. Our method outperforms the strongest baseline on the LaMP benchmark by 9.2% and 5.0% for LaMP-2M and LaMP-3, respectively. On the GSM8K dataset, our approach achieves a 15.7% improvement over the best baseline, demonstrating the broad effectiveness and versatility of `M-Pilot`.

From an algorithmic standpoint, ORPO and LEAP both share the core idea of prompt optimization using LLMs, aligning with our motivation. However, they both rely on training-free, in-context learning approaches. In contrast, BBox-Adapter focuses on constraining the black-box LLMs tree search process rather than providing high-level task guidance. `M-Pilot` distinguishes itself by introducing a trainable white-box controller for prompt optimization, offering a more powerful and flexible mechanism to steer the black-box LLM and further enhance its performance.

B.4 `M-Pilot` with other Iterative Training Method

We want to emphasize that the core contribution of our work lies in adapting the pioneering controller-generator framework. This framework is highly flexible and enables controllable multi-turn generation, significantly enhancing the ability of black-box large language models to handle complex, long-horizon tasks. Although we previously use IDPO to demonstrate our approach, we want to clarify that the specific RL method used to train the controller model is merely a design choice and not the primary focus of our work. To demonstrate the adaptability of our framework, we conducted experiments on all three previously mentioned datasets using SimPo [25] as the RL method. Specifically, given input x and generated guidance pair (g^+, g^-), the SimPo loss for optimizing the white-box LLM controller can be expressed as:

$$\mathcal{L}_{\text{SimPO}} := \mathbb{E}_{(x,g^+,g^-)\sim\mathcal{D}}\left[-\log\sigma\left(\eta^{-1}\left(\frac{1}{|g^+|}\log\pi_\theta(g^+|x) - \frac{1}{|g^-|}\log\pi_\theta(g^-|x) - \gamma\right)\right)\right]. \tag{12}$$

Remaining consistent with the previous notation, we can then easily represent the training objective for iterative guidance optimization with the SimPo loss as follows:

$$\mathcal{L}_{\text{ISimPO}} := \mathbb{E}_{(x,g^+,g^-)\sim\mathcal{D}}\left[-\log\sigma\left(\eta^{-1}\left(\frac{1}{|g^+|}\log\pi_\theta^{(m+1)}(g^+|x) - \frac{1}{|g^-|}\log\pi_\theta^{(m+1)}(g^-|x) - \gamma\right)\right)\right]. \tag{13}$$

Table 7: Additional experimental results on the personalization task using the LaMP benchmark. All baseline settings and notations remain consistent with main experiment. ISimPo represents Iterative SimPo loss.

Dataset (→)	**LaMP-1**		**LaMP-2N**		**LaMP-2M**		**LaMP-3**		**LaMP-4**		
Method (↓)	Acc. ↑	F-1 ↑	Acc. ↑	F-1 ↑	Acc. ↑	F-1 ↑	MAE ↓	RMSE ↓	R-1 ↑	R-L ↑	BLEU ↑
`gpt-4o-mini`	0.514	0.513	0.655	0.473	0.413	0.325	0.371	0.673	0.132	0.116	0.992
RAG (k=1) [34]	0.626	0.624	0.733	0.539	0.444	0.378	0.311	0.631	0.141	0.126	1.296
RAG (k=4) [34]	0.632	0.632	0.792	**0.611**	0.502	0.430	**0.272**	**0.579**	0.161	0.146	2.953
PAG [32]	0.624	0.624	0.775	0.559	0.496	0.443	0.316	0.645	0.143	0.130	1.968
`M-Pilot` (IDPO)	**0.640**	**0.639**	0.823	0.607	**0.527**	**0.465**	0.277	0.581	0.174	0.160	4.298
`M-Pilot` (ISimPo)	0.628	0.628	**0.826**	0.598	0.522	0.461	0.294	0.614	**0.180**	**0.167**	**4.997**

The results presented in Table 7, Table 8, and Table 9, despite being obtained with sub-optimal hyperparameters, still demonstrate that iterative guidance optimization can be seamlessly incorporated into various reinforcement learning methods. Moreover, it consistently outperforms other state-of-the-art baselines, further underscoring the effectiveness and versatility of the framework.

Table 8: Additional experimental results on GSM8K dataset.

Dataset (→)	**GSM8K**		**GSM-HARD**	
Method (↓)	`gpt-3.5`	`4o-mini`	`gpt-3.5`	`4o-mini`
CoT	0.809	0.932	0.406	0.500
Least-to-Most	0.811	0.908	0.425	0.498
PAL	0.802	0.920	0.638	0.748
$\text{PAL}_{\text{Self-Debug}}$	0.864	0.943	0.701	0.774
`M-Pilot` (IDPO)	**0.931**	**0.964**	**0.761**	**0.801**
`M-Pilot` (ISimPo)	0.908	0.950	0.731	0.789

Table 9: Additional experimental results across six planning tasks from AlfWorld.

Methods (↓) Tasks (→)	**Pick**	**Clean**	**Heat**	**Cool**	**Examine**	**Pick Two**	**All (134 tasks)**
BUTLER [39]	46.00	39.00	74.00	**100.00**	22.00	24.00	37.00
ReAct [55]	37.50	64.52	69.57	42.86	38.89	17.65	47.76
Reflexion [38]	50.00	41.94	65.22	52.38	66.67	47.06	52.99
AdaPlanner [41]	**100.00**	**93.55**	78.26	95.24	66.67	**88.24**	88.06
`M-Pilot` (IDPO)	**100.00**	**93.55**	**100.00**	95.24	**100.00**	**88.24**	**96.27**
`M-Pilot` (ISimPo)	**100.00**	**93.55**	95.65	95.24	77.78	**88.24**	92.54

B.5 `M-Pilot` with other controller model

Moreover, from a modular perspective, each component in the pipeline can be flexibly substituted without disrupting the overall framework. In this section, we replace the controller model from `LLaMA3-8B-Instruct` with `Qwen2.5-7B-Instruct` and replicate experiments across all domains as presented in the main paper. As shown in Tables 10, 11, and 12, `M-Pilot` continues to deliver strong results, achieving up to 1.6%, 7.2%, and 0.8% improvements over the second-best baseline across the respective benchmarks. These results further underscore its broad effectiveness and robustness across different controller models.

B.6 Comparison with Black-Box LLMs Controllers

We further compare our controller-generator framework with directly using a black-box LLM to guide another black-box LLM. As shown in Tables 13, 14, and 15, our framework consistently delivers either comparable or significantly better results across tasks in planning, reasoning, and personalization. We attribute this to **Effective Problem Decomposition with Feedback**. As highlighted in our abstract, we treat the black-box LLM within our framework as an "environment" and the white-box LLM as a "controller". The white-box LLM decomposes the problem and provides it as input to the black-box LLM. The black-box LLM either interacts with the environment or compares its output against ground truth, returning feedback as a supervisory signal. This feedback helps filter high-quality problem decompositions to train the white-box LLM. As a result, the trained white-box LLM generates problem decompositions that more effectively guide the black-box LLM to solve tasks. In contrast, a black-box LLM alone lacks environment feedback and cannot achieve equally effective problem decomposition.

B.7 Reduced Token Usage

For the AlfWorld task, we follow the settings used in Adaplanner [41], allowing the black-box LLM to reflect up to eight times. The process terminates either upon success or when the maximum number of reflections is reached. We report the black-box API cost and task performance with and without the controller in the table below. All experiments are conducted using `gpt-3.5-turbo` as the black-box LLM. As shown in Table 16, our method reduces API cost by 30% while still achieving a 9% improvement in task performance. This is attributed to the controller models well-structured instructions, which enable the black-box LLM to interact with the environment fewer times while attaining a higher success rate.

Table 10: `M-Pilot` compared with baselines on LaMP benchmark. The Controller model utilizes `Qwen2.5-7B-Instruct` as the base model, other settings remain consistent with main experiment.

Dataset (→)	**LaMP-1**		**LaMP-2N**		**LaMP-2M**		**LaMP-3**		**LaMP-4**		
Method (↓)	Acc. ↑	F-1 ↑	Acc. ↑	F-1 ↑	Acc. ↑	F-1 ↑	MAE ↓	RMSE ↓	R-1 ↑	R-L ↑	BLEU ↑
`gpt-4o-mini`	0.514	0.513	0.655	0.473	0.413	0.325	0.371	0.673	0.132	0.116	0.992
RAG (k=1) [34]	0.626	0.624	0.733	0.539	0.444	0.378	0.311	0.631	0.141	0.126	1.296
RAG (k=4) [34]	0.632	0.632	0.792	**0.611**	0.502	0.430	**0.272**	**0.579**	0.161	0.146	2.953
PAG [32]	0.624	0.624	0.775	0.559	0.496	**0.443**	0.316	0.645	0.143	0.130	1.968
`M-Pilot` **(Qwen)**	**0.640**	**0.640**	**0.808**	0.579	**0.509**	**0.443**	0.301	0.619	**0.167**	**0.153**	**4.466**
w/o IDPO	0.608	0.608	0.777	0.573	0.499	0.432	0.313	0.637	0.156	0.139	2.296

Table 11: `M-Pilot` compared with baselines on GSM8K dataset using `Qwen2.5-7B-Instruct` as the controller model.

Dataset (→)	**GSM8K**		**GSM-HARD**	
Method (↓)	`gpt-3.5`	`4o-mini`	`gpt-3.5`	`4o-mini`
CoT	0.809	0.932	0.406	0.500
Least-to-Most	0.811	0.908	0.425	0.498
PAL	0.802	0.920	0.638	0.748
$\text{PAL}_{\text{Self-Debug}}$	0.864	0.943	0.701	0.774
`M-Pilot` **(Qwen)**	**0.936**	**0.964**	**0.773**	**0.800**
w/o IDPO	0.932	0.961	0.767	0.799

For the GSM8K task, we compare our method with the self-consistency [49] approach, where the black-box model generates 8 responses per question and selects the final answer via majority voting. As shown in Table 17, compared to self-consistency, our method reduces API cost by 65% while improving performance by over 12% , thanks to the high-quality guidance provided by the controller model.

C Additional Related Works

Small LMs Drive LLMs Generation. SuperICL [53] incorporates outputs from smaller language models (LMs) as complementary information for input queries, integrating them into the context provided to black-box LLMs. However, these smaller LMs are fixed and can only support classification tasks that rely on label predictions with associated confidence scores. HYDRA [63] is a retrieval-augmented generation framework that trains a BERT-sized reranker to reorder retrieved passages to better cater to user-specific requirements. Nevertheless, these methods apply only discrete optimization on the prompt through reranking and selection of few-shot demonstrations, which limits the potential improvements achievable via prompt engineering.

RLHF. Proximal policy optimization (PPO) [35] is the predominant deep reinforcement learning method used in RLHF, leading to significant successes in models like InstructGPT [28], ChatGPT [1], and Gemini [31]. However, applying PPO requires extensive effort and resources [8, 13, 45], often beyond the scope of open-source capabilities. To simplify implementation and streamline the training process, recent works [2, 14] have proposed direct preference learning algorithms following the DPO framework [30]. These algorithms bypass the reward modeling step and directly optimize carefully designed loss objectives on the preference dataset, hence the term direct preference learning.

Self-Improvement Training. Recent advances in self-improvement methods for language models fall broadly into two categories: (1) online fine-tuning approaches and (2) bootstrapping methods. Fine-tuning approaches aim to enhance models by adjusting their parameters based on additional data or objectives. Notable methods include Rejection Fine-Tuning (RFT) [56], which augments the training set with correct completions; Alignment Fine-Tuning (AFT) [48], which introduces an alignment loss to increase the probabilities of correct chain-of-thoughts; Reinforced Fine-Tuning (ReFT) [24], which applies reinforcement learning to token prediction; and self-play [7], which iteratively refines the model using its own previous outputs. Bootstrapping methods, on the other hand, leverage the model's own generations to create new training data. Notable examples include

Table 12: `M-Pilot` compared with baselines on Alfworld dataset using `Qwen2.5-7B-Instruct` as the controller model.

Methods (↓) Tasks (→)	Pick	Clean	Heat	Cool	Exam	Pick2	All
BUTLER [39]	46.00	39.00	74.00	**100.00**	22.00	24.00	37.00
ReAct [55]	37.50	64.52	69.57	42.86	38.89	17.65	47.76
Reflexion [38]	50.00	41.94	65.22	52.38	**66.67**	47.06	52.99
AdaPlanner [41]	**100.00**	**93.55**	78.26	95.24	**66.67**	88.24	88.06
`M-Pilot` **(Qwen)**	**100.00**	**93.55**	**100.00**	80.95	55.56	**94.12**	**88.81**
w/o IDPO	100.00	90.32	82.61	80.95	44.44	94.12	83.58

Table 13: `M-Pilot` compared to Black-Box LLM Controllers on AlfWorld. For tasks in ALFWorld, we adhered to the setup used in Adaplanner [41].

Methods (↓) Tasks (→)	Pick	Clean	Heat	Cool	Examine	Pick Two	All (134 tasks)
`gpt-3.5` + `gpt-3.5`	**100.00**	41.94	**100.00**	76.19	88.89	88.24	79.85
`gpt-4o-mini` + `gpt-3.5`	95.83	45.16	56.52	52.38	5.56	88.24	57.46
`M-Pilot` + `gpt-3.5`	**100.00**	**93.55**	**100.00**	**95.24**	**100.00**	**88.24**	**96.27**

Self-Taught Reasoner (STaR) [52], which iteratively samples high-quality data; Reinforcement and Self-Training (ReST) [16] and its simplified version ReST$^{\text{EM}}$ [40], which alternate between data generation and reward-based optimization; and Verified Self-Taught Reasoner (V-STaR) [19], which combines self-training with outcome-based verification. Collectively, these approaches offer diverse strategies for enhancing model performance through targeted training and iterative refinement, highlighting the potential for self-improvement in language models.

D Dataset and Task Details

D.1 LaMP: Personalization

We employ the Language Model Personalization (LaMP) benchmark [34], an open-source benchmark specifically designed to train and evaluate the capability of language models in generating personalized content. LaMP encompasses a diverse set of tasks (with LaMP-2 comprising two tasks, LaMP-2N, and LaMP-2M), covering both personalized text classification and generation tasks. The dataset statistics are presented in Table 18 for a clear overview of its structure. Below are detailed descriptions of each task:

- **Task 1: Personalized Citation Identification (LaMP-1)**: A binary text classification task aimed at citation recommendation. The task assesses the language model's ability to identify a user's citation preferences. Given a user and their authored paper, the model predicts which of two candidate papers the user is more likely to cite. The user's profile contains titles and abstracts of their authored papers.
- **Task 2: Personalized News Categorization (LaMP-2N)**: A categorical text classification task that involves categorizing news articles into one of 15 categories based on a journalist's profile. Given an article written by a user, the model predicts its category using the user's history of articles and their categories.
- **Task 3: Personalized Movie Tagging (LaMP-2M)**: An ordinal text classification task focused on predicting one of 15 tags for a movie based on a user's tagging history. The task evaluates the model's ability to assign tags to a movie description using historical user-specific movie-tag pairs.
- **Task 4: Personalized Product Rating (LaMP-3)**: A text classification task that involves predicting product ratings, framed as a five-class problem. The model must predict a rating between one and five for a product review, using the user's past review and rating history. This task tests the model's ability to capture user-specific rating patterns.
- **Task 5: Personalized News Headline Generation (LaMP-4)**: A text generation task in which the model generates personalized news headlines for articles based on the author's

Table 14: M-Pilot compared to Black-Box LLM Controllers on GSM8K dataset.

Method (↓) Dataset (→)	GSM8K	GSM-HARD
gpt-3.5 + gpt-3.5	0.896	0.734
gpt-4o-mini + gpt-4o-mini	0.948	0.791
M-Pilot + gpt-4o-mini	**0.964**	**0.801**
M-Pilot + gpt-3.5	0.931	0.761

Table 15: M-Pilot compared to Black-Box LLM Controllers on the LaMP benchmark.

Dataset (→)	LaMP-1		LaMP-2N		LaMP-2M		LaMP-3		LaMP-4		
Method (↓)	Acc. ↑	F-1 ↑	Acc. ↑	F-1 ↑	Acc. ↑	F-1 ↑	MAE ↓	RMSE ↓	R-1 ↑	R-L ↑	BLEU ↑
gpt-3.5 + gpt-3.5	0.590	0.589	0.790	0.594	0.399	0.325	0.357	0.693	0.166	0.150	3.433
gpt-4o-mini + gpt-4o-mini	0.624	0.624	0.775	0.559	0.496	0.443	0.316	0.645	0.143	0.130	1.968
M-Pilot + gpt-4o-mini	**0.640**	**0.639**	**0.823**	0.607	**0.527**	**0.465**	**0.277**	**0.581**	0.174	0.160	**4.298**
M-Pilot + gpt-3.5	0.594	0.593	0.798	**0.609**	0.469	0.412	0.286	0.599	**0.176**	**0.161**	4.222

past article-title pairs. The task assesses the model's ability to replicate the author's stylistic preferences when creating headlines.

LaMP-6 has been excluded because the dataset is not publicly available. Furthermore, Tasks 1, 2, and 3 above cover personalization classification tasks, Task 4 covers personalization rating tasks, and Task 5 covers personalization generation tasks. Therefore, the tasks we selected encompass all categories of tasks in the LaMP benchmark.

D.2 Reasoning: GSM8K

GSM8K [10] is a dataset focused on high school-level mathematical reasoning. The numerical reasoning tasks within this dataset typically consist of a descriptive scenario followed by a culminating question. Answering these questions requires performing multi-step mathematical calculations based on the context provided in the description.

D.3 Planning: ALFWorld

AlfWorld [39] is a comprehensive suite of synthetic, text-based environments set within a virtual household, featuring six distinct task types: *Pick*, *Clean*, *Heat*, *Cool*, *Examine*, and *Pick Two*. Each task presents a unique high-level objective (e.g., "put a vase in the safe") that requires the agent to navigate and interact with various objects or receptacles (e.g., *go to shelf 6*, *clean apple*). To accomplish the assigned task, the agent must execute a series of actions to achieve the specified goal. However, the challenge lies in the object's potential location - it could be in any of over 50 possible places within a given task instance - necessitating sequential exploration of each location by the agent. Consequently, the complete action sequence may encompass more than 50 discrete actions, posing a considerable challenge to the agent's capabilities and efficiency.

E Baseline Details

E.1 LaMP: Personalization

We compare our proposed M-Pilot with several competitive baselines, encompassing both one-stage and two-stage methods. For all baseline approaches, we employ a consistent prompt template and utilize BM25 as the default retrieval mechanism across all experiments.

- gpt-4o-mini follows a zero-shot approach, directly answering the user query without leveraging the user's profile data.
- **RAG** combines the user's top retrieved history data with the input question as prompts for gpt-4o-mini to generate the final answer.

Table 16: API cost on AlfWorld.

Method	API cost ($)	Task Performance
M-Pilot	0.818	97.8
Adaplanner [41] (w/o controller)	1.151	88.8

Table 17: API cost on GSM8K.

Method	API cost ($)	Task Performance
M-Pilot	1.740	93.1
Self-Consistency [49] (w/o controller)	4.946	81.3

- **PAG** utilizes `gpt-4o-mini` to first generate a summary of the user's retrieved history data and then combines the summary with the input question as prompts for `gpt-4o-mini` to produce the final answer.

For our ablation study, we primarily compare `M-Pilot` with the following ablated baseline:

- `M-Pilot` **w/o IDPO** utilizes the controller model `Llama-3-8B-Instruct` to first generate a summary of the user's retrieved history data. It then combines this summary with the input question as prompts for the environment model `gpt-4o-mini` to generate the final answer.

E.2 GSM: Reasoning

For all baselines, we employ `gpt-3.5-turbo` as the black-box model to facilitate the description of their processes with 3-shot prompt template. The ablated baselines primarily focus on problem decomposition, including `M-Pilot` **w/o IDPO**. The remaining baselines for mathematical reasoning consist of **CoT** [51], **Least-to-Most** [61], **PaL** [15], and **PAL$_{\text{Self-Debug}}$** [6].

- `M-Pilot` **w/o IDPO** first utilizes a vanilla `LLaMA3-8B-Instruct` to break down the problem into sub-questions, and then `gpt-3.5-turbo` provide solutions based on both the main problem and the decomposed sub-questions.
- **CoT** uses `gpt-3.5-turbo` to break the problem down into a series of intermediate reasoning steps that ultimately lead to the final answer.
- **PaL** utilizes `gpt-3.5-turbo` to interpret natural language problems and generate programs as intermediate reasoning steps, delegating the solution process to a runtime environment like a Python interpreter.
- **PAL$_{\text{Self-Debug}}$** builds upon PaL by introducing a close-loop refinement during the inference phase. Specifically, if the code generated by PaL encounters issues during execution, `gpt-3.5-turbo` is instructed to reflect on the error and regenerate the code. The maximum number of reflections is set to 6.

E.3 Alfworld: Planning

We compare `M-Pilot` with several strong baselines in the planning task, encompassing both one-stage and two-stage approaches. For all baselines, we employ `gpt-3.5-turbo` as the black-box model for task execution. The ablated baselines (two-stage) include **w/o Guidance Optimization**, **w/o 1^{st}, 2^{nd}-round IDPO**, and **w/o 2^{nd}-round IDPO**. Additional baselines (one-stage) include **BUTLER** [39], **ReAct** [55], **Reflexion** [38], and **AdaPlanner** [41].

- **Ablated baselines.** These approaches utilize a white-box model to provide a high-level plan for the task, while `gpt-3.5-turbo` generates the specific solution based on this plan. Specifically:
  - **w/o Guidance Optimization** refers to an untuned `LLaMA3-8B-Instruct`.
  - **w/o 1^{st}, 2^{nd}-round IDPO** indicates a `LLaMA3-8B-Instruct` model that has undergone supervised fine-tuning on a limited amount of training data.

Table 18: Dataset statistics of five different personalization tasks (LaMP-1, 2N, 2M, 3, and 4) from the LaMP benchmark [34].

Task	Type	# Train	# Validation	# Test	Input Length	Output Length	# Profiles	# Classes
LaMP-1	Classification	9682	2500	2500	51.40 ± 5.72	-	90.61 ± 53.87	2
LaMP-2N	Classification	5914	1052	1274	65.40 ± 12.29	-	306.42 ± 286.65	15
LaMP-2M	Classification	5073	1410	1557	92.39 ± 21.95	-	86.76 ± 189.52	15
LaMP-3	Classification	20000	2500	2500	145.14 ± 157.96	-	188.10 ± 129.42	5
LaMP-4	Generation	12527	1925	2376	30.53 ± 12.67	9.78 ± 3.10	287.16 ± 360.62	-

  - **w/o 2^{nd}-round IDPO** denotes the `LLaMA3-8B-Instruct` model further trained using DPO on $\{$positive, negative$\}$ pairs from the training set, building upon the supervised fine-tuned model.
- **BUTLER** [39] is an agent that initially learns to perform abstract tasks in TextWorld through Imitation Learning (IL) and subsequently transfers the acquired policies to embodied tasks in ALFWorld.
- **ReAct** [55] is a general paradigm that combines reasoning and acting with language models to solve diverse language reasoning and decision-making tasks.
- **Reflexion** [38] employs verbal reinforcement to help agents learn from prior failures.
- **AdaPlanner** [41] is a closed-loop planning method where the LLM plays two roles, planner and refiner. It leverages code-based prompting for precise planning and refinement.

F Implementation Details

F.1 Hardware and Software

We conduct all black-box LLM enhancement experiments on CPU: AMD(R) EPYC(R) 7702 64-Core Processor@1.50GHz and GPU: NVIDIA A100-SXM4-80GB using Python 3.10.13.

F.2 Training Cost

Table 19: Detailed API cost per 1 million tokens.

Backbone Model	Input cost ($) / 1M tokens	Output cost ($) / 1M tokens
`gpt-3.5-turbo`	3.0	6.0
`gpt-4o-mini`	0.15	0.6

The data generation cost was calculated by aggregating the total token consumption statistics provided by Azure API and subsequently applying the cost per token (`gpt-3.5-turbo-0125`, `gpt-4o-mini`) as specified in official documentation [4]. The cost for processing 1M tokens, as detailed in Table 19, served as the basis for this calculation.

For the AlfWorld dataset, the entire training set consists of 8,808 samples. On average, using `gpt-3.5-turbo` to sample 100 examples costs approximately $3.20, making the estimated cost for complete data collection $282. For the GSM8K dataset, the full training set comprises 7,473 samples. The average cost for sampling 100 examples using `gpt-3.5-turbo` is $1.215, resulting in an estimated total cost for data collection of $90.80. In comparison, fine-tuning the `gpt-3.5-turbo` costs $216.50, and requires hourly payment in deployment for inference. For LaMP-1, LaMP-2M, LaMP-2N, LaMP-3, LaMP-4, we use gpt-4o-mini for data generation. The total costs are separately $6.144, $1.882, $2.348, $8.111, $10.022, with 5252, 2719, 2369, 8506, 12518 generated data samples.

During the training phase, we used four H100 GPUs for two rounds of DPO training. The process took approximately 1.5 hours for AlfWorld and GSM8K, resulting in a total training cost of 6 GPU

[4] https://openai.com/api/pricing/

hours. It took separately 4 gpu hours, 4 gpu hours, 4 gpu hours, 8 gpu hours, 12 gpu hours for the training process of LaMP-1, -2M, -2N, -3, -4.

F.3 LaMP: Personalization

F.3.1 Algorithm Details

We formalize the personalization problem within the context of our proposed `M-Pilot` framework. Specifically, we employ the controller model `Llama-3-8B-Instruct` to analyze the user's retrieved history data and generate an informative and clear intermediate summary. This summary is then combined with the input question as prompts for the environment model `gpt-4o-mini` to derive the final answer. To enhance control capabilities, we utilize online DPO to optimize the controller model `Llama-3-8B-Instruct`.

During the interaction stage, we follow the aforementioned pipeline, leveraging the controller model `Llama-3-8B-Instruct` to generate various intermediate outputs. By interacting with the environment model `gpt-4o-mini`, we obtain intermediate generations paired with ground truth answers as corresponding observations. We then sample both positive and negative intermediate generations based on the quality of the final answer. For classification tasks such as LaMP-1 and LaMP-2, an intermediate generation is labeled as positive if the final answer exactly matches the ground truth, and vice versa. For generation tasks like LaMP-4, we rank the generations by their metric scores and select the top ones as positive and the bottom ones as negative.

To prevent overfitting and reward hacking, the interaction stage processes the entire training dataset once for all personalization tasks. We sample at most two contrastive pairs for each training data point. We employ LoRA (Low-Rank Adaptation), a parameter-efficient method, to update the controller model `Llama-3-8B-Instruct`. LoRA is well-suited for personalization tasks, allowing efficient and effective optimization of the controller model. We utilize DPO for optimization.

F.3.2 Hyperparameter Configurations

We set the maximum sequence length for generated solutions to 512 tokens across all tasks and scenarios. The controller model is `Llama-3-8B-Instruct`, while the environment model is `gpt-4o-mini` for the primary tasks and `gpt-3.5-turbo` for specific ablation studies. For each user, we retrieve the minimum of k and the total number of user profiles as historical profiles. The value of k varies by dataset: all profiles for LaMP-1, 120 for LaMP-2N, 150 for LaMP-2M, 30 for LaMP-3, and 50 for LaMP-4. These retrieved profiles are utilized in generating intermediate solutions. Comprehensive prompt templates and additional details are provided in Appendix H.

To prevent overfitting and reward hacking, we iterate through the entire training dataset only once. For each data point, we perform ten interactions, generating ten distinct intermediate solutions with a temperature setting of 1.0. Consequently, each data point results in at least ten intermediate generations. To further mitigate overfitting and reward hacking, we sample a maximum of two contrastive pairs per data point. The total number of contrastive pairs sampled during the interaction stage is as follows: 5410 for LaMP 1, 2850 for LaMP-2M, 2548 for LaMP-2N, 4320 for LaMP-3, and 12518 for LaMP-4, respectively.

During optimization, we train for two epochs per task using the following hyperparameters: LoRA rank to 8, LoRA α to 16, LoRA dropout to 0.05, learning rate to 1e-5, float type to bf16, max length to 8192, and label smoothing to 0.1. We utilize all the contrastive pairs sampled from the interaction stage for optimization. For all the experiments, we set all the random seeds to 42 for reproducibility consideration.

F.4 GSM8K: Reasoning

Following the PAL framework [15], we employ code-style LLM prompts to facilitate the conversion of mathematical problems into executable code, thereby augmenting the model's problem-solving capabilities. Unlike PAL, which directly translates mathematical problems into code, `M-Pilot` first assists GPT in decomposing the problem into more manageable sub-problems. This decomposition allows GPT to more effectively convert these simpler sub-problems into code, enhancing both the correctness and stability of the generated code. Additionally, since `M-Pilot` is responsible solely for high-level planning without engaging in low-level execution, we can train on the GSM8K dataset and

evaluate on the GSM-Hard dataset. Both datasets comprise similar problem types, with GSM-Hard featuring more intricate numerical calculations.

In our experimental setup, we begin by randomly sampling 216 code-based solutions to mathematical problems from the GSM8K training set using `gpt-3.5-turbo-0125`. We then extract the planning components from these code blocks to perform supervised fine-tuning (SFT) on the LLaMA model, thereby equipping LLaMA with foundational planning capabilities for solving mathematical problems. The SFT training configuration mirrors that used for ALFWorld. Subsequently, LLaMA functions as the planner, generating breakdowns and planning solutions for each of the 7,473 problems in the GSM8K training set. Concurrently, GPT serves as the executor, producing executable code based on each problem and the corresponding plan provided by LLaMA.

During inference, consistent with our experiments on ALFWorld, we implement closed-loop refinement to enhance model performance. `M-Pilot` initially decomposes the mathematical problem into simpler sub-problems. The black-box model then generates corresponding code blocks for each sub-problem. If the execution of the generated code does not produce the expected answer or if execution issues arise, the error information is relayed back to the black-box model for reflection and iterative improvement. We restrict the number of reflection attempts to six; `M-Pilot` is given an additional opportunity to re-decompose the task if the problem remains unsolved in the first iteration. Any problem that remains unresolved after all these attempts is deemed beyond the reasoning capabilities of the black-box model..

F.5 Alfworld: Planning

Following AdaPlanner [41], we employ a closed-loop planning approach for inference on ALFWorld. The primary distinction lies in `M-Pilot`'s responsibility for generating the high-level plan, while a black-box model, such as `gpt-3.5-turbo-0125`, handles low-level execution after comprehending both the problem and the high-level plan. Similar to AdaPlanner, we utilize code-style LLM prompts to enhance the black-box model's planning and interaction capabilities with the environment.

Our initial objective is to enhance LLaMA's planning ability on ALFWorld. To achieve this, we enable GPT to perform closed-loop high-level planning and low-level execution on 400 samples from ALFWorld's training set. From these runs, we selected 277 examples that successfully reached the goal state and extracted the planning components to fine-tune LLaMA using supervised learning. For the SFT, we set the learning rate to 2×10^{-5}, with a batch size of 64, and trained a LoRA module with a rank of 8, an alpha of 16, and a dropout rate of 0.05 over 3 epochs. After LLaMA acquires a foundational level of planning ability, we designate it as the planner and assign GPT as the executor. The two models then perform closed-loop inference on the ALFWorld training set, comprising 8,810 samples. Each sample is executed eight times, with successful runs labeled as positive samples and unsuccessful ones as negative samples. This process yields 4,844 unique $\{$positive, negative$\}$ pairs, which are utilized for the first epoch of DPO training on LLaMA.

Subsequently, we repeat the data collection process on the ALFWorld training set using the DPO-trained model, gathering 1,586 samples. This reduction in samples occurs because, as `M-Pilot` becomes more capable post-DPO training, it generates a higher proportion of positive outcomes, resulting in fewer $\{$positive, negative$\}$ pairs. By aggregating all collected samples, we obtain a total of 6,430 pairs, which are then used to conduct the second epoch of DPO training on `M-Pilot`. This further enhances its planning capabilities and aligns them more closely with GPT's execution proficiency. Through this iterative DPO training approach, we observe that the high-level plans generated by LLaMA more effectively guide GPT's execution, leading to a higher success rate in ALFWorld tasks.

Additionally, during the inference stage, we maintain a closed-loop approach to bolster the model's performance. Specifically, the black-box model first generates a corresponding trajectory based on the task and the prompt provided by `M-Pilot`. If an error occurs during execution or the task remains incomplete after a predetermined number of steps, the black-box model reflects on its generated trajectory and the encountered error, subsequently regenerating a new trajectory. The model is allowed up to six reflection attempts, if it fails after these attempts, `M-Pilot` is prompted to re-decompose the task and instruct the black-box model to attempt solving the problem again. If the black-box model still fails after six additional reflection attempts, the task is considered beyond the models

capabilities. This approach significantly improves the model's ability to interact with the environment and increases the likelihood of successfully completing tasks.

G Case Studies

G.1 Alfworld

We compare `M-Pilot` with both the `LLaMA3-8B-Instruct` model used as the controller and AdaPlanner [41] in terms of performance across various tasks and environments. In `M-Pilot`, the controller model is trained based on `LLaMA3-8B-Instruct`, while the black-box executor employed is `gpt-3.5-turbo-0125`. Our results demonstrate that `M-Pilot` enhances the black-box executor's ability to interact with the environment, enabling it to successfully accomplish target tasks. In contrast, both AdaPlanner and the `LLaMA3-8B-Instruct` model commonly encounter issues such as initiating actions that the agent cannot complete and misjudging the agent's state. `M-Pilot` effectively mitigates these problems, resulting in more reliable task execution.

Task Definition

```
You are in the middle of a room. Looking quickly around you, you
see a bed 1, a desk 2, a desk 1, a drawer 6, a drawer 5, a drawer
4, a drawer 3, a drawer 2, a drawer 1, a garbagecan 1, a
laundryhamper 1, a safe 1, a shelf 6, a shelf 5, a shelf 4, a shelf
3, a shelf 2, and a shelf 1.

Your task is to: examine the book with the desklamp.
```

M-Pilot

```
# Decomposition:
General plan: I need to get a list of receptacles to find the book
and take the book with me, then I get another list of receptacles
to find the desklamp and turn it on.
[Step 1] get a list of receptacles where a book is likely to appear
    .
[Step 2] go to each receptacle in the list until seeing a book.
[Step 3] take the book from the receptacle.
[Step 4] get a list of receptacles where a desklamp is likely to
    appear.
[Step 5] go to each receptacle in the list until seeing a desklamp.
[Step 6] turn on the desklamp.
[Step 7] go to the receptacle where the book is and turn on the
    desklamp.
[Step 8] examine the book with the desklamp.

# GPT Solution:
def solution(agent, start_from=1):
    # General plan: I need to get a list of receptacles to find the
    book and take the book with me, then I get another list of
    receptacles to
    find the desklamp and turn it on.
    if start_from <= 1:
    print("[Step 1] get a list of receptacles where a book is
    likely to appear.")
    # Ask the assistant to sort the receptacles based on the
    likelihood
    of finding a book.
    answer = ask(f'Given a list of receptacles, please sort them in
    descending order based on the likelihood
    of finding a book in each of them. The list of receptacles is:
```

```
    {agent.receptacles}. You should directly return a Python list
        .')
    recep_to_check = literal_eval(answer)
    # Check if the returned list is not empty.
    assert recep_to_check, f'Error in [Step 1]: recep_to_check
    should not be empty. {agent.report()}'

if start_from <= 2:
    print("[Step 2] go to each receptacle in the list until seeing
    a book.")
    for receptacle in recep_to_check:
        observation = agent.goto(receptacle)
        # Check if the receptacle is closed. If so, open it.
        if 'closed' in observation:
            observation = agent.open_receptacle(receptacle)
        # Check if a book is in/on the receptacle.
        if 'book' in observation:
            break
    # Check if a book is found in one of the receptacles.
    assert 'book' in observation, f'Error in [Step 2]: There is no
    book in/on {recep_to_check}. {agent.report()}'

if start_from <= 3:
    print("[Step 3] take the book from the receptacle.")
    # Get the identifier of the book.
    answer = ask(f'From the observation, get the identifier of an
    object. For example, On the cabinet 1, you see a cloth 2, and a
    book 1. The identifier of cloth is 2. Now, {observation} The
    identifier of the book? Only Output a single number without any
    other words.')
    found_book = f'book {answer}'
    observation = agent.take(found_book, receptacle)
    # Check if the agent is holding the book.
    assert agent.holding == found_book, f'Error in [Step 3]: I
    cannot take {found_book} from the {receptacle}.
    {agent.report()}'

if start_from <= 4:
    print("[Step 4] get a list of receptacles where a desklamp is
    likely to appear.")
    # Ask the assistant to sort the receptacles based on the
    likelihood of finding a desklamp.
    answer = ask(f'Given a list of receptacles, please sort them in
    descending order based on the likelihood of finding a desklamp
    in each of them. The list of receptacles is:
    {agent.receptacles}. You should directly return a Python list
        .')
    recep_to_check = literal_eval(answer)
    # Check if the returned list is not empty.
    assert recep_to_check, f'Error in [Step 4]: recep_to_check
    should not be empty. {agent.report()}'

if start_from <= 5:
    print("[Step 5] go to each receptacle in the list until seeing
    a desklamp.")
    for receptacle in recep_to_check:
        observation = agent.goto(receptacle)
        # Check if the receptacle is closed. If so, open it.
        if 'closed' in observation:
            observation = agent.open_receptacle(receptacle)
        # Check if a desklamp is in/on the receptacle.
        if 'desklamp' in observation:
            break
```

```
    # Check if a desklamp is found in one of the receptacles.
    assert 'desklamp' in observation, f'Error in [Step 5]: There is
    no desklamp in/on {recep_to_check}. {agent.report()}'

if start_from <= 6:
    print("[Step 6] turn on the desklamp.")
    # Get the identifier of the desklamp.
    answer = ask(f'From the observation, get the identifier of an
    object. For example, On the cabinet 1, you see a cloth 2, and a
    desklamp 1. The identifier of cloth is 2. Now, {observation}
    The identifier of the desklamp? Only Output a single number
    without any other words.')
    found_desklamp = f'desklamp {answer}'
    # Turn on the desklamp.
    observation = agent.turn_on(found_desklamp)
    # Check if the desklamp is turned on.
    assert 'turn on' in observation, f'Error in [Step 6]: I cannot
    turn on {found_desklamp} in/on {receptacle}. {agent.report()}'

if start_from <= 7:
    print("[Step 7] go to the receptacle where the book is and turn
    on the desklamp.")
    observation = agent.goto(receptacle)
    # Check if the receptacle is closed. If so, open it.
    if 'closed' in observation:
        observation = agent.open_receptacle(receptacle)
    # Turn on the desklamp.
    observation = agent.turn_on(found_desklamp)
    # Check if the desklamp is turned on.
    assert 'turn on' in observation, f'Error in [Step 7]: I cannot
    turn on {found_desklamp} in/on {receptacle}. {agent.report()}'

if start_from <= 8:
    print("[Step 8] examine the book with the desklamp.")
    # Examine the book with the desklamp.
    observation = f'You examine the {found_book} with the
    {found_desklamp}.'
    # Check if the book is examined with the desklamp.
    assert 'examine' in observation, f'Error in [Step 8]: I cannot
    examine the {found_book} with the {found_desklamp}.
    {agent.report()}'

Success
```

Vanilla LLaMA3-8B-Instruct

```
# Decomposition:
General plan: I need to get a list of receptacles to find the book
and take the book with me, then I get another list of receptacles
to find the desklamp and examine the book with it.
[Step 1] Get a list of receptacles where a book is likely to appear
    .
[Step 2] Go to each receptacle in the list until seeing a book.
[Step 3] Take the book from the receptacle.
[Step 4] Get a list of receptacles where a desklamp is likely to
appear.
[Step 5] Go to each receptacle in the list until seeing a desklamp.
[Step 6] Examine the book with the desklamp.

# GPT Solution:
def solution(agent, start_from=1):
```

```
# General plan: I need to get a list of receptacles to find the
book and take the book with me, then I get another list of
receptacles to find the desklamp and examine the book with it.

if start_from <= 1:
    print("[Step 1] Get a list of receptacles where a book is
    likely to appear.")
    # Ask the assistant to sort the receptacles based on the
    likelihood of finding a book.
    answer = ask(f'Given a list of receptacles, please sort
    them in descending order based on the likelihood of finding
    a book in each of them. The list of receptacles is:
    {agent.receptacles}. You should directly return a Python
    list.')
    recep_to_check = literal_eval(answer)
    # Check if the returned list is not empty.
    assert recep_to_check, f'Error in [Step 1]: recep_to_check
    should not be empty. {agent.report()}'

if start_from <= 2:
    print("[Step 2] Go to each receptacle in the list until
    seeing a book.")
    for receptacle in recep_to_check:
        observation = agent.goto(receptacle)
        # Check if the receptacle is closed. If so, open it.
        if 'closed' in observation:
            observation = agent.open_receptacle(receptacle)
        # Check if a book is in/on the receptacle.
        if 'book' in observation:
            break
    # Check if a book is found in one of the receptacles.
    assert 'book' in observation, f'Error in [Step 2]: There is
    no book in/on {recep_to_check}. {agent.report()}'

if start_from <= 3:
    print("[Step 3] Take the book from the receptacle.")
    # Get the identifier of the book from the observation.
    answer = ask(f'From the observation, get the identifier of
    an object. For example, On the cabinet 1, you see a cloth
    2, and a book 1. The identifier of cloth is 2. Now,
    {observation} The identifier of the book? Only Output a
    single number without any other words.')
    found_book = f'book {answer}'
    observation = agent.take(found_book, receptacle)
    # Check if the agent is holding the book.
    assert agent.holding == found_book, f'Error in [Step 3]: I
    cannot take {found_book} from the {receptacle}.
    {agent.report()}'

if start_from <= 4:
    print("[Step 4] Get a list of receptacles where a desklamp
    is likely to appear.")
    # Ask the assistant to sort the receptacles based on the
    likelihood of finding a desklamp.
    answer = ask(f'Given a list of receptacles, please sort
    them in descending order based on the likelihood of finding
    a desklamp in each of them. The list of receptacles is:
    {agent.receptacles}. You should directly return a Python
    list.')
    recep_to_check = literal_eval(answer)
    # Check if the returned list is not empty.
    assert recep_to_check, f'Error in [Step 4]: recep_to_check
    should not be empty. {agent.report()}'
```

```
    if start_from <= 5:
        print("[Step 5] Go to each receptacle in the list until
        seeing a desklamp.")
        for receptacle in recep_to_check:
            observation = agent.goto(receptacle)
            # Check if the receptacle is closed. If so, open it.
            if 'closed' in observation:
                observation = agent.open_receptacle(receptacle)
            # Check if a desklamp is in/on the receptacle.
            if 'desklamp' in observation:
                break
        # Check if a desklamp is found in one of the receptacles.
        assert 'desklamp' in observation, f'Error in [Step 5]:
        There is no desklamp in/on {recep_to_check}.
        {agent.report()}'

    if start_from <= 6:
        print("[Step 6] Examine the book with the desklamp.")
        # Examine the book with the desklamp.
        observation = agent.examine(found_book,
        found_desklamp
        # Check if the book is being examined with the desklamp.
        assert 'examine' in observation, f'Error in [Step 6]: I
        cannot examine the book with the desklamp. {agent.report()
            }'

AttributeError: 'Agent' object has no attribute 'examine'}, in
step6: observation = agent.examine(found_book, found_desklamp)
```

Adaplanner

```
def solution(agent, start_from=1):
    # General plan: I need to find the book and examine it with the
    desklamp.

    if start_from <= 1:
        print("[Step 1] get a list of receptacles where a book is
        likely to appear.")
        # Ask the assistant to sort the receptacles based on the
        likelihood of finding a book.
        answer = ask(f'Given a list of receptacles, please sort
        them in descending order based on the likelihood of finding
        a book in each of them. The list of receptacles is:
        {agent.receptacles}. You should directly return a Python
        list.')
        recep_to_check = literal_eval(answer)
        # Check if the returned list is not empty.
        assert recep_to_check, f'Error in [Step 1]: recep_to_check
        should not be empty. {agent.report()}'

    if start_from <= 2:
        print("[Step 2] go to each receptacle in the list until
        finding a book")
        for receptacle in recep_to_check:
            observation = agent.goto(receptacle)
            # Check if the receptacle is closed. If so, open it.
            if 'closed' in observation:
                observation = agent.open_receptacle(receptacle)
            # Check if a book is in/on the receptacle.
            if 'book' in observation:
```

```
                break
        # Check if a book is found in the receptacle.
        assert 'book' in observation, f'Error in [Step 2]: There is
        no book in/on {recep_to_check}. {agent.report()}'

    if start_from <= 3:
        print("[Step 3] examine the book with the desklamp")
        # Find the desklamp to use for examining the book.
        answer = ask(f'Given the observation, find the identifier
        of the desklamp. For example, On the desk 1, you see a lamp
        1 and a book 2. The identifier of the lamp is 1. Now,
        {observation} What is the identifier of the desklamp? Only
        output a single number without any other words.')
        found_desklamp = f'desklamp {answer}'
        # Turn on the desklamp to examine the book.
        observation = agent.turn_on(found_desklamp)
        # Check if the desklamp is turned on.
        assert 'turn on' in observation, f'Error in [Step
        3]: Unable to turn on {found_desklamp} in/on {receptacle}.
        {agent.report()}'

AssertionError: Error in [Step 3]: Unable to turn on desklamp 0
in/on bed 1, in step3: assert 'turn on' in observation, f'Error in
[Step 3]: Unable to turn on {found_desklamp} in/on {receptacle}.
{agent.report()}'
```

G.2 GSM-Hard

On the mathematical reasoning task, we compare `M-Pilot` with several strong baselines, including `LLaMA3-8B-Instruct` as the controller model, Chain-of-Thought (CoT) [51], and PaL [15]. In `M-Pilot`, the controller model is trained based on `LLaMA3-8B-Instruct`, and the black-box model employed is `gpt-3.5-turbo-0125`. Our results indicate that common issues with the baselines include incorrect code execution order and difficulties in translating questions into code. In contrast, `M-Pilot` significantly enhances the black-box model's ability to solve mathematical problems through code generation.

Task Definition

```
There are 5915126 students in Marissa s class. Each student
    started the
year with 10 pencils. After two months, 1/5 of the total pencils in
class were used. At the end of the year, only 1/3 of the remaining
pencils were left. How many pencils were left?

True Answer: 15773669.333333334
```

M-Pilot

```
# Decomposition:
Let's break down this problem:
1. How many pencils were used in total?
2. How many pencils are left after using the total amount and then
losing 2/3 of the remaining pencils?


# GPT Solution:
def solution():
```

```
    """There are 5915126 students in Marissa s class. Each
        student
    started the year with 10 pencils. After two months, 1/5 of the
    total pencils in class were used. At the end of the year, only
        1/3
    of the remaining pencils were left. How many pencils were left
        ?"""
    # Initialization of variables
    num_students = 5915126
    pencils_per_student = 10
    pencils_used_fraction = 1/5
    pencils_left_fraction = 1/3

    # How many pencils were used in total?
    total_pencils_initial = num_students * pencils_per_student
    pencils_used = total_pencils_initial * pencils_used_fraction

    # How many pencils are left after using the total amount and
        then
    losing 2/3 of the remaining pencils?
    pencils_remaining = total_pencils_initial - pencils_used
    pencils_left = pencils_remaining * (1 - pencils_left_fraction)
    result = pencils_left
    return result

Answer: 31547338.66666667, Wrong Answer

# Second-Turn Decomposition:
Let's break down this problem:
1. How many pencils were used in total?
2. How many pencils are left after using the total amount?
Note: We don't need to consider losing 2/3 of the remaining pencils
     in
the decomposition, as it's a part of the calculation for the total
pencils used.

# Second-Turn GPT Solution:
def solution():
    """There are 5915126 students in Marissa s class. Each
        student
    started the year with 10 pencils. After two months, 1/5 of the
    total pencils in class were used. At the end of the year, only
        1/3
    of the remaining pencils were left. How many pencils were left
        ?"""
    # Initialization of variables
    num_students = 5915126
    pencils_per_student = 10
    pencils_used_fraction = 1/5
    pencils_left_fraction = 1/3

    # How many pencils were used in total?
    total_pencils_initial = num_students * pencils_per_student
    pencils_used = total_pencils_initial * pencils_used_fraction

    # How many pencils are left after using the total amount and
        then
    losing 2/3 of the remaining pencils?
    pencils_remaining = total_pencils_initial - pencils_used
    pencils_left = pencils_remaining * (1 - pencils_left_fraction)
    result = pencils_left
    return result
```

```
Answer: 15773669.333333332, Correct Answer
```

Vanilla LLaMA3-8B-Instruct

```
# Decomposition:
Let's break down this problem:
1. How many pencils were used in total?
2. How many pencils were left after the first two months?
3. How many pencils were left at the end of the year?


# GPT Solution:
def solution():
    """There are 5915126 students in Marissa s class. Each
        student
    started the year with 10 pencils. After two months, 1/5 of the
    total pencils in class were used. At the end of the year, only
        1/3
    of the remaining pencils were left. How many pencils were left
        ?"""
    # Initialization of variables
    num_students = 5915126
    pencils_per_student = 10
    pencils_used_fraction = 2/5
    pencils_left_fraction = 1/3

    # How many pencils were used in total?
    total_pencils_initial = num_students * pencils_per_student
    pencils_used_total = total_pencils_initial *
        pencils_used_fraction

    # How many pencils were left after the first two months?
    pencils_remaining_after_two_months = total_pencils_initial -
    pencils_used_total

    # How many pencils were left at the end of the year?
    pencils_left = pencils_remaining_after_two_months *
    pencils_left_fraction
    result = pencils_left
    return result

Answer: 11830252.0, Wrong Answer
```

PaL

```
def solution():
    """There are 5915126 students in Marissa s class. Each
        student started
    the year with 10 pencils. After two months, 1/5 of the total
        pencils in
    class were used. At the end of the year, only 1/3 of the
        remaining
    pencils were left. How many pencils were left?"""
    num_students = 5915126
    pencils_per_student = 10
    total_pencils_initial = num_students * pencils_per_student

    pencils_used = total_pencils_initial // 5
    pencils_remaining = total_pencils_initial - pencils_used
```

```
    pencils_left_end_of_year = pencils_remaining // 3
    result = pencils_left_end_of_year
    return result

Answer: 15773669.0, Wrong Answer
```

CoT

```
First, we need to find out how many pencils were used after two
    months.
Since 1/5 of the total pencils were used, the number of pencils
    used is
1/5 * (5915126 * 10) = 1183025.2 pencils.

Next, we need to find out how many pencils were left at the end of
    the
year. Since only 1/3 of the remaining pencils were left, the number
     of
pencils left is 2/3 * (5915126 * 10 - 1183025.2) = 3156750.8
    pencils.

Since we can't have a fraction of a pencil, we need to round down
    to
the nearest whole number. Therefore, the number of pencils left is
3156750 pencils. So the answer is 3156750.

Answer: 3156750, Wrong Answer
```

G.3 LaMP

We utilize case study tables to illustrate the performance improvements achieved through the intermediate generations of the controller model. Specifically, we present detailed case studies for LaMP-2M, LaMP-2N, and LaMP-3 in Table 20. In these analyses, the controller model is `Llama-3-8B-Instruct`, and the environment model is `gpt-4o-mini`. As demonstrated in Table 20, `M-Pilot` enhances the summarization capabilities of the controller model. Common errors in PAG, such as inaccurate summaries or improper utilization of retrieved user history, are mitigated by `M-Pilot`, which enables the controller model to produce more accurate generations through interaction and optimization.

H Prompt Templates

H.1 ALFWorld

Following Adaplanner [41], we implement a code-style prompt for `M-Pilot`, which can be divided into the following sections:

High-level Planning. The `<high_level_planning>` prompt is used to instruct `M-Pilot` to break the current task down into multiple subtasks, where `<decompose>` is replaced by a standard task decomposition process, and `<receptacle_list>` is substituted by the list of interactive receptacles provided by the task environment. Finally, `<task>` is replaced by the task description, expressed in natural language.

<high_level_planning> Prompt

Table 20: Case Study for LaMP. In the "Target" column, we present the ground-truth categorization. The "Gen" column displays the final answer generated by the black-box model corresponding to each intermediate generation. The "Score" column indicates whether the generated answer ("Gen") matches the target categorization ("Target"). In the "Method" column, "Ours" refers to `M-Pilot`, while "PAG" stands for Profile Augmented Generation.

Task	Input Question	Intermediate Generation	Method	Target	Gen	Score
2M	Which tag does this movie relate to among the following tags? A ticking-time-bomb insomniac and a slippery soap salesman channel...	most popular tag: "dystopia", "fantasy", "comedy", "violence"	Ours	violence	violence	1
		Here are the most popular tags for the user: dystopia, fantasy, comedy	PAG	violence	dystopia	0
2N	Which category does this article relate to among the following categories? The suspect, Akayed Ullah, was the most seriously hurt in the rush-hour blast...	most popular category: politics, crime, entertainment, women, business, sports.	Ours	crime	crime	1
		Based on the articles you provided, the most popular category written by this journalist is: politics.	PAG	crime	politics	0
3	What is the score of the following review on a scale of 1 to 5? After almost 20 years in and around MIT, I've encountered only two great MIT books: (1) A.R. Gurney's out-of-print novel The Snow Ball (correction: it is Entertaining Strangers); (2) Pepper White's book...	Based on this user's past reviews, the most common positive score is: 4, with 4 reviews out of 8 receiving a score of 4. The most common negative score is: 1, with 4 reviews out of 8 receiving a score of 1.	Ours	4	4	1
		Based on the reviews, the most common positive score is 5, and the most common negative score is 1.	PAG	4	5	0

```
# Decompose the task into steps. First give a general plan of how
    you
would solve the task, then for each step you plan to take, mark
    with
'[Step xx]'.

# Here is an example of a decomposition to the task:
# define environment
receptacles = ['diningtable 1','drawer 2', 'drawer 1', 'sinkbasin
    1',
'toilet 1', 'sidetable 2', 'sidetable 1', 'cabinet 1', 'countertop
    1',
'microwave 1', 'fridge 1']

<decompose>

# Here is the actual task.
# define environment
receptacles = <receptacle_list>

# <task>
# here is a decomposition:
```

Multi-Turn Planning. The `<multi_turn_planning>` prompt is used to guide `M-Pilot` to reflect on errors in the previous decomposition and re-break the current task into multiple subtasks. Here, `<predecompose>` is replaced with the high-level plan from the previous turn, while all other elements retain the same meaning as before.

<multi_turn_planning> Prompt

```
# Decompose the task into steps. First give a general plan of how
    you
would solve the task, then for each step you plan to take, mark
    with
'[Step xx]'.

# Here is a successful example of a decomposition to the task:
# define environment
receptacles = ['diningtable 1','drawer 2', 'drawer 1', 'sinkbasin
    1',
'toilet 1', 'sidetable 2', 'sidetable 1', 'cabinet 1', 'countertop
1', 'microwave 1', 'fridge 1']

<decompose>

# Here is the actual task.
# define environment
receptacles = <receptacle_list>

# <task>

# Here are the decomposition steps you previously generated for the
task.
<predecompose>

# However, you made a mistake in the decomposition above because of
lack of understanding of the task.

Referring to the successful example, please correct the error, if
any, and rewrite the decomposition.
# here is a decomposition:
```

Low-level Execution. The <low_level_execution> prompt is used to instruct the black box model to generate a specific solution based on the problem and the plan provided by M-Pilot. <basic_info> defines the agent and admissible actions on Alfworld and can be found in [41]. The <example> is replaced with a combination of planning and expert trajectory, while the meanings of <receptacle_list> and <task> remain consistent with the previous description. <decomposition> represents the high-level plan provided by M-Pilot for the current task.

<low_level_execution> Prompt

```
<basic_info>

# Now complete the function solution() below to solve the task by
composing the agent's methods to interact with the environment.
# First give a general plan of how you would solve the task, mark
    with
' # General Plan'. Then for each step you plan to take, 1) mark
    with
'[Step xx]', 2) give a reason why you think it is a good step to
    take
3) write an assertion to check if the step is successful.

# Here is an example of a solution to the task:
# define environment and agent
receptacles = ['diningtable 1','drawer 2', 'drawer 1', 'sinkbasin
    1',
```

```
'toilet 1', 'sidetable 2', 'sidetable 1', 'cabinet 1', 'countertop
    1',
'microwave 1', 'fridge 1']
agent = Agent(receptacles)

<example>

# Here is the actual task.
# define environment and agent
receptacles = <receptacle_list>
agent = Agent(receptacles)

# <task>
# here is a decomposition:
<decomposition>

# here is a solution:
```

Planning Samples. In ALFWorld, there are six types of tasks: `Pick`, `Clean`, `Heat`, `Cool`, `Examine`, and `Pick two`. For each type, we collect a reasonable high-level planning approach, allowing `M-Pilot` to reference them. These six planning samples are presented as follows:

Planning Sample for the task `Pick`:

<planning_sample_pick> Prompt

```
# Your task is to: put soapbar on countertop.
# here is a decomposition:
# General Plan: I need to get a list of receptacles where the
    soapbar
is likely to appear, and then go to each receptacle in the list
    until
seeing a soapbar. Then I can put get the identifier of the soapbar
    and
take it. Finally I can go to the countertop and put the soapbar.
# [Step 1] get a list of receptacles where the soapbar is likely to
appear.
# [Step 2] go to each receptacle in the list until seeing a soapbar
    .
# [Step 3] identify the soapbar I juts found and take it.
# [Step 4] go to a countertop and put the soapbar on it.
```

Planning Sample for `Clean`:

<planning_sample_clean> Prompt

```
# Your task is to: put a clean lettuce in diningtable / clean a
lettuce and put it in diningtable.
# here is a decomposition:
# General plan: I need to get a list of receptacles to find the
lettuce, take the lettuce to the sinkbasin, clean it and put it in
    a
diningtable.
# [Step 1] get a list of receptacles where the lettuce is likely to
appear.
# [Step 2] go to each receptacle in the list until seeing a lettuce
    .
# [Step 3] identify the lettuce I just found and take it.
# [Step 4] go to a sinkbasin to clean the lettuce.
```

```
# [Step 5] go to a diningtable and put the lettuce on it.
```

Planning Sample for `Heat`:

<planning_sample_heat> Prompt

```
# Your task is to: put a hot lettuce in diningtable / heat some
lettuce and put it in diningtable.
# here is a decomposition:
# General plan: I need to get a list of receptacles to find the
lettuce, take the lettuce to the microwave, heat it and put it in a
diningtable.
# [Step 1] get a list of receptacles where the lettuce is likely to
appear.
# [Step 2] go to each receptacle in the list until seeing a lettuce
    .
# [Step 3] identify the lettuce I juts found and take it.
# [Step 4] go to a microwave to heat the lettuce.
# [Step 5] go to a diningtable and put the lettuce on it.
```

Planning Sample for `Cool`:

<planning_sample_cool> Prompt

```
# Your task is to: put a cold lettuce in diningtable / cool some
lettuce and put it in diningtable.
# here is a decomposition:
# General plan: I need to get a list of receptacles to find the
lettuce, take the lettuce to the fridge, cool it and put it in a
diningtable.
# [Step 1] get a list of receptacles where the lettuce is likely to
appear.
# [Step 2] go to each receptacle in the list until seeing a lettuce
    .
# [Step 3] identify the lettuce I juts found and take it.
# [Step 4] go to a fridge to cool the lettuce.
# [Step 5] go to a diningtable and put the lettuce on it.
```

Planning Sample for `Examine`:

<planning_sample_examine> Prompt

```
# Your task is to: look at the bowl under the desklamp / examine
    the
bowl with the desklamp
# here is a decomposition:
# General plan: I need to get a list of receptacles to find the
    bowl
and take the bowl with me, then I get another list of receptacles
    to
find the desklamp and turn it on.
# [Step 1] get a list of receptacles where a bowl is likely to
    appear.
# [Step 2] go to each receptacle in the list until seeing a pen.
# [Step 3] take the bowl from the receptacle.
# [Step 4] get a list of receptacles where a desklamp is likely to
appear.
```

```
# [Step 5] go to each receptacle in the list until seeing a
    desklamp.
# [Step 6] turn on desklamp.
```

Planning Sample for `Pick Two`:

<planning_sample_picktwo> Prompt

```
# Your task is to: put two cellphone in cabinet / find two
    cellphone
and put them in cabinet
# here is a decomposition:
# General plan: I need to get a list of receptacles to find the two
cellphones, find and take the first cellphone and put it in a
    cabinet,
then find and take the second cellphone and put it in the cabinet.
# [Step 1] get a list of receptacles where a cellphone is likely to
appear.
# [Step 2] go to each receptacle in the list until seeing a
    cellphone.
# [Step 3] identify the first cellphone found and take it.
# [Step 4] go to a cabinet and put the first cellphone found on it.
# [Step 5] go to each of the remaining receptacle in the list until
seeing a second cellphone.
# [Step 6] identify the second cellphone I just found and take it.
# [Step 7] go to a cabinet and put the second cellphone found on it
    .
```

Execution Samples. Our execution sample is based on the prompt structure from [41], with the key distinction being the incorporation of the planning component. In this setup, `<decompose>` is substituted with the task-specific planning sample, `<execution>` is replaced by the expert samples from [41], and the definition of `<task>` remains unchanged from the previous description.

<execution_sample_template> Prompt

```
# <task>
# <decompose>
# <execution>
```

Close-loop Refinement. To implement close-loop refinement during the inference stage, we follow the approach from [41] and introduce several prompts: a `<code_check>` prompt to identify and fix any syntax errors during execution generation, a `<refinement>` prompt to address refinement in case of assertion errors, and a `<start_from>` prompt to determine the starting point for the new solution after revising the plan. Detailed descriptions of these prompts can be found in [41].

H.2 GSM-Hard

Following PAL framework [15], we implement a code-based framework to solve mathematical problems on GSM-Hard, which is primarily divided into two steps: `M-Pilot` breaks down the mathematical problem into sub-problems, and the black-box model converts each sub-problem into a code block.

Problem Decomposition. For `M-Pilot`, we employ a three-shot prompt to guide the decomposition steps, where `<question>` represents the current problem.

<problem_decomposition> Prompt

```
# System Message: You will decompose a math problem into smaller
parts. Follow the prompt instruction and do not generate redundant
information.

Q: Olivia has $23. She bought five bagels for $3 each. How much
    money
does she have left?
A: Let's break down this problem:\nHow much does Olivia spend on
bagels?\nHow much money does Olivia have left after the purchase?

Q: Michael had 58 golf balls. On tuesday, he lost 23 golf balls. On
wednesday, he lost 2 more. How many golf balls did he have at the
    end
of wednesday?
A: Let's break down this problem:\nHow many golf balls did Michael
lose in total by the end of Wednesday?\nHow many golf balls does
Michael have left after losing the total amount?

Q: There were nine computers in the server room. Five more
    computers
were installed each day, from monday to thursday. How many
    computers
are now in the server room?
A: Let's break down this problem:\nHow many computers were added in
total from Monday to Thursday?\nHow many computers are now in the
server room after adding the new ones?

Q: <question>
A:
```

Multi-turn Decomposition. If the Black-box LLM encounters an error while solving the problem, `M-Pilot` is required to reflect on the previous decomposition `<decompose>` and re-decompose the problem.

<multi_turn_decomposition> Prompt

```
# System Message: You will decompose a math problem into smaller
parts. Follow the prompt instruction and do not generate redundant
information.

# Here are some examples on how to decompose the question into
    smaller
parts.
Q: Olivia has $23. She bought five bagels for $3 each. How much
    money
does she have left?
A: Let's break down this problem:\nHow much does Olivia spend on
bagels?\nHow much money does Olivia have left after the purchase?

Q: Michael had 58 golf balls. On tuesday, he lost 23 golf balls. On
wednesday, he lost 2 more. How many golf balls did he have at the
    end
of wednesday?
A: Let's break down this problem:\nHow many golf balls did Michael
lose in total by the end of Wednesday?\nHow many golf balls does
Michael have left after losing the total amount?

Q: There were nine computers in the server room. Five more
    computers
```

```
were installed each day, from monday to thursday. How many
    computers
are now in the server room?
A: Let's break down this problem:\nHow many computers were added in
total from Monday to Thursday?\nHow many computers are now in the
server room after adding the new ones?

# Here is the actual question.
Q: <question>
You have decomposed the problem into smaller parts:
<decompose>
However, you made a mistake in the decomposition above because of
    lack
of understanding of the question.

Referring to the examples, please correct the error, if any, and
rewrite the decomposition.
A:
```

Code Generation. Given the problem and the decomposition provided by `M-Pilot`, the Black-box model generates the corresponding code block for each sub-problem. We continue to use a three-shot prompt to instruct the Black-box model on how to translate the sub-problems into code, where `<question>` represents the current problem and `<decompose>` represents the decomposition provided by `M-Pilot`.

<code_generation> Prompt

```
# System Message: You will write python program to solve math
problems. You will write annotations and code blocks following
instructions. Annotations should be written in the form of a
    question.

Let's use python to solve math problems. Here are three examples
    how
to do it,
Q: Olivia has $23. She bought five bagels for $3 each. How much
    money
does she have left?
Let's break down this problem:\nHow much does Olivia spend on
    bagels?
\nHow much money does Olivia have left after the purchase?
```
```
def solution():
    """Olivia has $23. She bought five bagels for
    $3 each. How much money does she have left?"""
    # Initialization of variables
    money_initial = 23
    bagels = 5
    bagel_cost = 3

    # How much does Olivia spend on bagels?
    money_spent = bagels * bagel_cost

    # How much money does Olivia have left after the purchase?
    money_left = money_initial - money_spent
    result = money_left
    return result
```
```

Q: Michael had 58 golf balls. On tuesday, he lost 23 golf balls. On
```

```
wednesday, he lost 2 more. How many golf balls did he have at the
    end
of wednesday?
Let's break down this problem:\nHow many golf balls did Michael
    lose
in total by the end of Wednesday?\nHow many golf balls does Michael
have left after losing the total amount?
'''
def solution():
    """Michael had 58 golf balls. On tuesday, he lost 23 golf balls
        .
    On wednesday, he lost 2 more. How many golf balls did he have
        at
    the end of wednesday?"""
    # Initialization of variables
    golf_balls_initial = 58
    golf_balls_lost_tuesday = 23
    golf_balls_lost_wednesday = 2

    # How many golf balls did Michael lose in total by the end of
    Wednesday?
    golf_balls_left = golf_balls_initial - golf_balls_lost_tuesday
        -
    golf_balls_lost_wednesday

    # How many golf balls does Michael have left after losing the
    total amount?
    result = golf_balls_left
    return result
'''

Q: There were nine computers in the server room. Five more
    computers
were installed each day, from monday to thursday. How many
    computers
are now in the server room?
Let's break down this problem:\nHow many computers were added in
    total
from Monday to Thursday?\nHow many computers are now in the server
room after adding the new ones?
'''
def solution():
    """There were nine computers in the server room. Five more
    computers were installed each day, from monday to thursday. How
    many computers are now in the server room?"""
    # Initialization of variables
    computers_initial = 9
    computers_per_day = 5
    num_days = 4  # 4 days between monday and thursday

    # How many computers were added in total from Monday to
        Thursday?
    computers_added = computers_per_day * num_days

    # How many computers are now in the server room after adding
        the
    new ones?
    computers_total = computers_initial + computers_added
    result = computers_total
    return result
'''

How about this question?
```

```
Q: <question>
<decompose>
```

Close-loop Refinement. `<refinement>` prompt is employed to encourage the model to reflect and fix issues in its own solution, wherein `<error_msg>` is replaced by the error message returned by the solution function.

`<refinement>` Prompt

```
Let's use python to solve math problems. Here are three successful
cases on how to do it,
Q: Olivia has $23. She bought five bagels for $3 each. How much
    money
does she have left?
'''
def solution():
    """Olivia has $23. She bought five bagels for $3 each. How much
    money does she have left?"""
    money_initial = 23
    bagels = 5
    bagel_cost = 3
    money_spent = bagels * bagel_cost
    money_left = money_initial - money_spent
    result = money_left
    return result
'''

Q: Michael had 58 golf balls. On tuesday, he lost 23 golf balls. On
wednesday, he lost 2 more. How many golf balls did he have at the
    end
of wednesday?
'''
def solution():
    """Michael had 58 golf balls. On tuesday, he lost 23 golf balls
        .
    On wednesday, he lost 2 more. How many golf balls did he have
        at
    the end of wednesday?"""
    golf_balls_initial = 58
    golf_balls_lost_tuesday = 23
    golf_balls_lost_wednesday = 2
    golf_balls_left = golf_balls_initial - golf_balls_lost_tuesday
        -
    golf_balls_lost_wednesday
    result = golf_balls_left
    return result
'''

Q: There were nine computers in the server room. Five more
    computers
were installed each day, from monday to thursday. How many
    computers
are now in the server room?
'''
def solution():
    """There were nine computers in the server room. Five more
    computers were installed each day, from monday to thursday. How
    many computers are now in the server room?"""
    computers_initial = 9
    computers_per_day = 5
```

```
    num_days = 4  # 4 days between monday and thursday
    computers_added = computers_per_day * num_days
    computers_total = computers_initial + computers_added
    result = computers_total
    return result
'''

# Here is the actual question.
Q: <question>
You have generated code of solution() to solve the task. However,
    you
executed the solution() function and get an error message:
<error_msg>

Referring to the successful case and the error message, you should
complete the solution function with the correct code.
```

H.3 LaMP

Following the RAG-based framework [34] and the PAG-based framework [32], we implement prompt designs for both `M-Pilot` and the baseline methods. The prompt design for RAG is presented in Table 21, while the prompts for the two-stage PAG and `M-Pilot` are shown in Table 22. We create prompts for the controller model using the templates from Table 22 and subsequently combine the intermediate generations with the input question to form prompts for the environment model. Since LaMP-3 prompts are particularly lengthy, we provide additional examples of our PAG prompts for LaMP-1, LaMP-2N, LaMP-2M, and LaMP-4 as follows.

Table 21: RAG prompt design for five LaMP tasks. Concat($\cdot$) concatenates the input strings in order, and PPEP($\cdot$) composes the prompt for each retrieved item from the profile. [INPUT] represents the task's input.

Task	Per Profile Entry Prompt (PPEP)	Aggregated Input Prompt (AIP)
LaMP-1	"P_i[title]"	concat([PPEP(P_1), ..., PPEP(P_n)], ", and "). [INPUT]
LaMP-2N	"the category for the article: "P_i[text]" is ""P_i[category]""	concat([PPEP(P_1), ..., PPEP(P_n)], ", and "). [INPUT]
LaMP-2M	"the tag for the movie: "P_i[description]" is "P_i[tag]"	concat([PPEP(P_1), ..., PPEP(P_n)], ", and "). [INPUT]
LaMP-3	P_i[score] is the score for "P_i[text]"	concat([PPEP(P_1), ..., PPEP(P_n)], ", and "). [INPUT]
LaMP-4	"P_i[title]" is the title for "P_i[text]"	concat([PPEP(P_1), ..., PPEP(P_n)], ", and "). [INPUT]

PAG Prompt Demo for LaMP-1

```
Write a summary, in English, of the research interests and topics
of a researcher who has published the following papers.
Only generate the summary, no other text.
The published papers are:
    \"Efficient Evaluation of Continuous Text Search Queries\",
    and \"Continuous Monitoring of Spatial Queries in Wireless
    Broadcast Environments\", and \"Spatial queries in wireless
    broadcast environments\", and \"Maximum Rank Query\", and
    \"Anonymous Query Processing in Road Networks\",
    and \"An Incremental Threshold Method for Continuous Text
    Search Queries\", and \"Continuous Top-k Monitoring on
    Document Streams.\", and \"Best upgrade plans for large road
    networks\", and \"Scalable verification for outsourced dynamic
    databases\", and \"Heuristic algorithms for balanced multi-way
    number partitioning\", and \"Aggregate nearest neighbor
```

Table 22: Summarization prompt design for the five LaMP tasks. [INPUT] represents the task's input.

Task	Prompt
LaMP-1	Write a summary, in English, of the research interests and topics of a researcher who has published the following papers. Only generate the summary, no other text.
LaMP-2N	Look at the following past articles this journalist has written and determine the most popular category they write in. Answer in the following format: most popular category: ¡category top1¿, ¡category top2¿, ..., ¡category topn¿
LaMP-2M	Look at the following past movies this user has watched and determine the mostpopular tag they labeled. Answer in the following form: most popular tag: ¡tag top1¿, ¡tag top2¿, ..., ¡tag topn¿
LaMP-3	Based on this user's past reviews, what are the most common scores they give for positive and negative reviews? Answer in the following form: most common positive score: ¡most common positive score¿, most common negative score: ¡most common negative score¿
LaMP-4	Given this author's previous articles, try to describe a template for their headlines. I want to be able to accurately predict the headline gives one of their articles. Be specific about their style and wording, don't tell me anything generic. Use the following format: The template is: '[template 1]', '[template 2]', '[template 3]', '[template 4]'

```
    queries in spatial databases\", and \"Partially materialized
    digest scheme: an efficient verification method for outsourced
    databases\", and \"Best upgrade plans for single and multiple
    source-destination pairs.\", and \"Tree-based partition
    querying: a methodology for computing medoids in large spatial
    datasets\", and \"A Threshold-Based Algorithm for Continuous
    Monitoring of k Nearest Neighbors\", and \"Computing immutable
    regions for subspace top-k queries\", and \"Historical traffic-
    tolerant paths in road networks\"...
```

PAG Prompt Demo for LaMP-2M

```
Look at the following past movies this user has watched and
    determine
the most popular tag they labeled. Answer in the following form:
most popular tag: <tag top1>, <tag top2>, ..., <tag topn>.
The movies and tags are:
    the tag for the movie: \"Young hobbit Frodo Baggins,
    after inheriting a mysterious ring from his uncle Bilbo,
    must leave his home in order to keep it from falling into
    the hands of its evil creator. Along the way,
    a fellowship is formed to protect the ringbearer
    and make sure that the ring arrives at its final destination:
    Mt. Doom, the only place where it can be destroyed.\" is
    \"fantasy\" , and the tag for the movie: \"Set in the 22nd
    century, The Matrix tells the story of a computer hacker
    who joins a group of underground insurgents fighting the vast
    and powerful computers who now rule the earth.\" is \"sci-fi\"
        ,
    and the tag for the movie: \"Batman raises the stakes in his
    war on crime. With the help of Lt. Jim Gordon and District
    Attorney Harvey Dent, Batman sets out to dismantle the
    remaining criminal organizations that plague the streets. The
    partnership proves to be effective, but they soon find
    themselves prey to a reign of chaos unleashed by a rising
    criminal mastermind known to the terrified citizens of Gotham
```

as the Joker.\" is \"psychology\" , and the tag for the movie: \"An unsuspecting, disenchanted man finds himself working as a spy in the dangerous, high-stakes world of corporate espionage. Quickly getting way over-his-head, he teams up with a mysterious femme fatale.\" is \"twist ending\"...

PAG Prompt Demo for LaMP-2N

Look at the following past articles this journalist has written and determine the most popular category they write in. Answer in the following format: most popular category: <category top1>, <category top2>, ..., <category topn>.
The articles and categories are:
the category for the article: \"Champions like Tiger Woods are always charting and changing their course to be certain everything is on track. Tiger didn't just come to Augusta because it was the popular thing to do. He wouldn't have showed up if he wasn't ready to win. He came to win and he's prepared to win.\" is \"sports\" , and the category for the article: \"In 2011, in an interview with The Golf Channel, I predicted a Tiger Woods comeback while many others said he was done. I was right that time and I am right again, and I'll say it right now and on the record: Tiger Woods will be back again and dominate the game of golf like the Tiger of old.\" is \"sports\" , and the category for the article: \"What do you teach your kids about money, prosperity and how to get rich? If you\u2019re like most parents, the answer is probably\" is \"business\" , and the category for the article: \"With a little bit of planning and a lot of discipline, accomplishing your goals in the New Year can become a reality. Imagine the immense satisfaction you'll feel at this same time next year when you can look back and look at how far you've come and all that you have accomplished.\" is \"healthy living\" , and the category for the article: \"This whole argument boils down to a simple premise: who is in charge of our lives? Doctors? Politicians? Religious leaders? Or Us? Are we so feeble minded that we cannot be trusted to be responsible for our own existence?\" is \"politics\"...

PAG Prompt Demo for LaMP-4

Given this author's previous articles, try to describe a template for their headlines. I want to be able to accurately predict the headline gives one of their articles. Be specific about their style and wording, don't tell me anything generic. Use the following format: The template is: '[template 1]', '[template 2]', '[template 3]', '[template 4]'.
Previous articles and titles are:
\"Selling a House to Buy a House\" is the title for \"
Homeowners
sell their homes and buy other homes for a variety of reasons including a need to live closer to a place of employment, to be closer to family, to enjoy a better climate, or simply to upgrade. This article is about finding the best sequence of steps in the process.\", and \"Investing In a Larger Down Payment: High Yields and No Risk\" is the title for \"Consumers looking to purchase a home within the near future face many decisions, including how

large a down payment to make. The down payment is the sale price (confirmed by a appraisal) less the loan amount. In most cases, home purchasers must have financial assets at least as large as the down payment they make.\", and \"Why and How to Eliminate Mortgage Charges by Third Parties\" is the title for \"Third-party settlement costs could be eliminated by implementation of one simple rule: any service required by lenders as a condition for the granting of a home mortgage must be purchased and paid for by the lender.\", and \"Do Home Buyers Need a Pre-Approval?\" is the title for \"With bargaining power shifting from home buyers to sellers in an increasing number of local markets, buyers in competition with other buyers are looking for any edge they can get. One possible edge is a pre-approval letter (henceforth PAL) from a lender.\", and \"A New Challenge to the HECM Reverse Mortgage Program\" is the title for \"The United States today faces a retirement funds crisis: a rapidly growing number of persons who are retiring without the financial capacity to support themselves during ever-increasing life spans.\"...