REINFORCEMENT LEARNING IN INFERENCE TIME: A PERSPECTIVE FROM SUCCESSIVE POLICY ITERATIONS

Xinnan Zhang[1], **Chenliang Li**[2], **Siliang Zeng**[1], **Jiaxiang Li**[1], **Zhongruo Wang**[3], **Songtao Lu**[4], **Alfredo Garcia**[2], **Mingyi Hong**[1]

[1]University of Minnesota, Twin Cities, [2]Texas A&M University, College Station, [3]Amazon, [4]The Chinese University of Hong Kong

ABSTRACT

Aligning Large Language Models (LLMs) to human preferences is essential for their effective deployment in real-world applications. Traditional post-training methods, such as Reinforcement Learning with Human Feedback (RLHF), are resource-intensive and time-consuming, especially as model sizes continue to grow. Recently, inference-time alignment methods have gained significant attention, as they can steer the LLM output without direct fine-tuning, and can be integrated with post-training techniques to further enhance performance. Additionally, these methods enable personalization, allowing models to adapt dynamically to user preferences and specific task requirements. However, these approaches operate in a one-shot manner, limiting policy improvement to a single round. To address this limitation, we introduce inference-time Successive Policy Iterations (SPI), a novel algorithm that enables successive policy improvement at inference time. Specifically, inference-time SPI iteratively learns value functions and leverages them to guide the LLM through a search-based optimization process. Theoretically, our algorithm is equivalent to performing multi-iteration policy optimization on the base model, effectively improving its behavior without direct fine-tuning. Experimental results demonstrate that inference-time SPI significantly improves length-control win rates on challenging instruction-following benchmarks, such as AlpacaEval 2.0, achieving a substantial performance boost (e.g., $30.71\% \rightarrow 43.80\%$ for `Llama-3-8B-Instruct` compare against GPT-4 responses). Furthermore, inference-time SPI consistently outperforms existing test-time alignment baselines such as Best-of-N (BoN), weak to strong search, which is effective for inference time scaling on different tasks.

1 INTRODUCTION

As ChatGPT (Achiam et al., 2023) and DeepSeek (Guo et al., 2025) have taken the world by storm, it is clear that AI systems will soon become ubiquitous in our lives. For instance, Large Language Models (LLMs) have been used to solve hard problems including video gaming (Berner et al., 2019), and complex reasoning (Bai et al., 2023). Traditionally, training-time alignment methods such as Reinforcement Learning with Human Feedback (RLHF) (Ouyang et al., 2022), and Direct Preference Optimization (DPO) (Rafailov et al., 2024b) are used to align LLMs with human preferences. While effective, these approaches require extensive computational resources and incur significant costs, particularly for large-scale models. Furthermore, different tasks often require retraining or fine-tuning models for optimal performance. However, in many real-world scenarios, such training-based methods may be impractical due to limited computational resources, the inability to access model weights like black box models, or restrictions on modifying large-scale proprietary models. These limitations have driven significant interest in inference-time alignment methods, such as Snell et al. (2024); Mudgal et al. (2023); Xu et al. (2024b), which keep the LLM frozen while steering its behavior through external guidance without directly fine-tuning the model, further boost the performance.

The guidance in the inference time often uses one or several reward models. A simple yet effective approach, called Best-of-N (BoN) (Nakano et al., 2021; Stiennon et al., 2020), selects the highest-ranking response from N samples using an outcome reward model (ORM). However, BoN operates at the response level, limiting its granularity in guiding generation. To provide finer-grained control, alternative methods attempt to incorporate reward signals at the token or partial-response level. For instance, ARGS (Khanov et al., 2024) simply applies an RM trained on the complete sequence to partially generated responses, introducing substantial estimation errors. Inspired by Rafailov et al. (2024a), several works (Qiu et al., 2024; Zhou et al., 2024; Liu et al., 2024) use an implicit reward, computed as the relative log probability between a base and a fine-tuned model, to provide token-level or partial-response guidance. However, this implicit reward is often inaccurate and can underperform compared to the explicit reward-based method (Liu et al., 2024). Other works (Chakraborty et al., 2024; Huang et al., 2024a) improve reward accuracy by computing rewards on fully generated responses, but this significantly increases inference costs. For complicated tasks, such as mathematical reasoning (Bai et al., 2023) and multi-step decision-making (Shao et al., 2024; Hao et al., 2024), step-level process guidance (Lightman et al., 2023) is important in reducing the search space and improving the final quality.

Another research line (Mudgal et al., 2023; Han et al., 2024; Kong et al., 2024) explores training an external value function to control frozen LLM. Since a value function estimates expected future rewards, it enables more informed and context-aware decision-making during inference. Additionally, some approaches leverage step-level verifiers or process reward models (PRMs) (Li & Li, 2024) for inference-time guidance. However, most of these inference-time alignment work operates in a one-shot manner, allowing for only a single round of policy improvement. To address these limitations, we revisit the constrained policy optimization problem proposed in TRPO (Schulman, 2015) and build a partial Lagrangian reformulation of such a problem. By iteratively solving this optimization problem, we propose incorporating value functions at inference time using a search-based method to simulate training-time policy updates without fine-tuning. Unlike most existing inference-time alignment approaches, which apply a one-shot improvement, our framework enables successive policy improvement by alternating between value training and decoding steps without needing to fine-tune the base model.

Contribution. In this paper, we propose inference-time Successive Policy Iterations (SPI), a novel algorithm that iteratively improves the policy at inference time by learning value models and using value models to guide policy generation. Our contributions can be summarized as follows:

1. We utilize the constrained policy optimization formulation proposed in Peng et al. (2019) and propose an effective algorithm for solving such a problem in inference time. The proposed algorithm successively improves the policy toward the optimal policy by leveraging external guidance from additive value function models without changing the parameters of the base policy. Such process is theoretically equivalent to improving the base policy by policy iteration[1].

2. Empirically, we provide extensive evidence demonstrating that inference-time SPI offers a significant improvement over existing inference-time alignment methods. In the summarization dataset, our method achieves continuous performance improvements based on an SFT model, surpassing both the (inference-time) BoN method and the (train-time) DPO approach. In 8b model experiment, we show that leveraging 7b model as value function can significantly enhance the performance of the 8b base model on the instruction-following benchmark. Additionally, we conduct a detailed ablation study to analyze the impact of key factors, including chunk length, data size, and search strategies, on the performance of inference-time SPI.

The proposed approach demonstrates the effectiveness of inference-time alignment, as a versatile tool for enhancing model performance in the post-training stage.

[1]Our implemented algorithm is still different from the standard policy iteration to ensure efficiency. For example, we did not calculate through the entire action space (i.e., the entire vocabulary) when outputting the response from the reweighted policy model. See Section 3 for details of our implementation.

2 PRELIMINARIES AND PROBLEM FORMULATION

The Finite-Horizon MDP Model. A Markov decision process (MDP) is defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mu, r, \gamma)$, which consists of the state space $\mathcal{S}$, the action space $\mathcal{A}$, the transition dynamics $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, the initial state distribution $\mu(\cdot)$, the reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and the discounted factor $\gamma \in (0, 1)$. Under a transition dynamics model $\mathcal{P}$ and a policy $\pi : \mathcal{S} \to \Delta_{\mathcal{A}}$ where $\Delta_{\mathcal{A}}$ is the probability simplex on the action space, further define the corresponding state visitation measure as $d_\pi(s) := (1-\gamma)\sum_{t=0}^{T} \gamma^t \mathcal{P}_\pi(s_t = s | s_0 \sim \eta)$ for any state $s \in \mathcal{S}$. Here T is the horizon size.

Token-level MDP Model of LLM. Denote the entire input prompt as x and output continuation as y. The text generation process of LLM can be modeled as an MDP, where generation corresponds to sampling from a learned policy. Specifically, each state at the time step t, denoted as $s_t = [x, y_{1:t-1}]$, includes the prompt x and the sequences of tokens $y_{1:t-1}$ generated up to that point. Each action $a_t = y_t$ represents a token from the vocabulary. The transition kernel $\mathcal{P}$ is deterministic, i.e. given tokens $s_t = [x, y_{1:t-1}]$ and a_t, $s_{t+1} = P(s_t, a_t) = [s_t, a_t]$. This corresponds to adding the newly generated token a_t to the existing sequence, thus forming the updated output.

Formulation of Successive Policy Iterations. Let us consider the most general case where the horizon T is ∞. We inspect the original policy optimization problem, which is:

$$J(\pi) := \mathbb{E}_{\tau \sim \pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right], \tag{1}$$

where $\tau := (s_0, a_0, s_1, a_1, \dots)$ denotes one trajectory, corresponding to one data point with prompt(s) and continuation(s). Under a policy/LLM π, we can define the corresponding value function V_π and the Q-function Q_π as below:

$$\begin{aligned} V_\pi(s) &:= \mathbb{E}_{\tau \sim \pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s\right], \\ Q_\pi(s, a) &:= r(s, a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)}\left[V_\pi(s')\right]. \end{aligned} \tag{2}$$

We can further define the advantage function for each state action pair (s, a) as follows:

$$A_\pi(s, a) = Q_\pi(s, a) - V_\pi(s). \tag{3}$$

Then one can have the following *meta* algorithm that iteratively optimizes $J(\pi)$

$$\pi_{k+1} = \pi_k + \eta_k \sum_{(s,a) \sim \mathcal{D}_k} P(A_{\pi_k}(s, a)), \tag{4}$$

where η_k is the learning rate; $P(\cdot)$ is a stochastic gradient estimate of $J(\pi)$, which is typically a function of the advantage function; $\mathcal{D}_k$ is a data set that contains trajectories. Of course, the key ingredient in any policy optimization algorithm is to properly identify the stochastic gradient direction $P(\cdot)$, which improves the objective function $J(\pi)$.

The fundamental idea of policy improvement is that, suppose there is a reference policy π', we can maximize the performance gap over the reference policy to achieve policy improvement:

$$\eta_{\pi'}(\pi) := J(\pi) - J(\pi'). \tag{5}$$

Therefore, the performance improvement of the policy π over the reference policy π' can be expressed by the advantage function $A_{\pi'}(s, a)$.

Lemma 1 (Lemma 1.16 in Agarwal et al. (2019)) *For any policy π and π', the performance difference can be expressed as below:*

$$\eta_{\pi'}(\pi) = \frac{1}{1-\gamma} \mathbb{E}_{s \sim d_\pi(\cdot), a \sim \pi(\cdot|s)}\left[A_{\pi'}(s, a)\right] \tag{6}$$

where $d_\pi(s) := (1-\gamma)\sum_{t=0}^{\infty} \gamma^t \mathcal{P}_\pi(s_t = s | s_0 \sim \mu)$ denotes the state visitation measure.

In the context of LLM, this performance difference lemma indicates that to align the LLM with the reward model, i.e. to maximize the objective of policy improvement in equation 5, one just needs to seek a policy π that induces a positive expected advantage $\mathbb{E}_{s\sim d_\pi(\cdot),a\sim\pi(\cdot|s)}\big[A_{\pi'}(s,a)\big] > 0$ over the reference policy π'. Therefore, we focus on maximizing equation 6.

However, from a practical point of view, the dependency on sampling data from the visitation measure $d_\pi(\cdot)$ makes it difficult to optimize the performance difference defined in equation 6. Following the trust region policy optimization (TRPO, see Schulman (2015)) framework, we instead consider an approximation to $\eta_{\pi'}(\pi)$ by $\tilde{\eta}_{\pi'}(\pi)$:

$$\tilde{\eta}_{\pi'}(\pi) = \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_{\pi'}(\cdot),a\sim\pi(\cdot|s)}\big[A_{\pi'}(s,a)\big] \tag{7}$$

where $d_{\pi'}(\cdot)$ denotes the state visitation measure under the reference policy π'. According to Theorem 1 in Schulman (2015), $\tilde{\eta}_{\pi'}(\pi)$ serves as a good approximation to $\eta_{\pi'}(\pi)$ when the two policies π and π' are close in terms of the KL-divergence. Thus, when maximizing the surrogate objective $\tilde{\eta}_{\pi'}(\pi)$ defined in equation 7 while penalizing the KL divergence between π and π', we are able to guarantee monotonic performance improvement at each policy iteration step.

The above discussions lead to the following optimization objective, which is also used in the TRPO: at each policy iteration step and given the previous policy π_{old}, one solves the constrained policy optimization problem:

$$\max_{\pi} \quad \tilde{\eta}_{\pi_{\text{old}}}(\pi) \tag{8a}$$

$$s.t. \quad \mathbb{E}_{s\sim d_{\pi_{\text{old}}}(\cdot)}\big[D_{\text{KL}}\big(\pi(\cdot|s)||\pi_{\text{old}}(\cdot|s)\big)\big] \le \epsilon, \tag{8b}$$

$$\sum_{a\in\mathcal{A}} \pi(a|s) = 1, \forall s \in \mathcal{S}, \tag{8c}$$

$$\pi(a|s) \ge 0, \forall s \in \mathcal{S}, a \in \mathcal{A}. \tag{8d}$$

Proposition 1 *Denote the optimal policy of the formulation equation 8a as π_{new}, then we show its expression is given as below (Schulman, 2015; Zhang et al., 2024):*

$$\pi_{\text{new}}(a|s) = \pi_{\text{old}}(a|s)\exp\Big(\frac{1}{\beta}\big(r(s,a) + \gamma\mathbb{E}_{s'\sim\mathcal{P}(\cdot|s,a)}V_{\pi_{\text{old}}}(s')\big)\Big) \tag{9}$$

where $\beta := \frac{1}{(1-\gamma)\alpha}$, α is the dual variables of the KL constraints, $\mathcal{P}$ is the transition kernel of the environment, and $V_{\pi_{\text{old}}}$ is the value function for the previous policy.

From equation 9, it is clear that the distribution of π_{new} is modified from the previous policy by incorporating an advantage-weighted adjustment. The parameter β serves as a critical hyperparameter that regulates the trade-off between exploration and exploitation. When β is large, the updated policy remains closer to the previous policy, resulting in more conservative updates. Conversely, when β is small, the updated policy places greater emphasis on the reward and value estimates, leading to more aggressive adjustments.

Note that once we obtain the new policy as equation 9, we can plug it back to equation 2 to obtain the corresponding new value functions, and compute the advantage, then follow the policy iteration equation 4 to further improve the policy, which is known as a policy improvement iteration in the literature (Agarwal et al., 2019).

Summarizing the above discussion, it is possible to design an iterative algorithm approximating the policy iteration that solves, which iteratively enhances the policy without fine-tuning the parameters. At each iteration, the policy π_k is updated in inference time based on the previously improved policy π_{k-1} by using the value function trained in the last iteration. According to the proposition 1, the optimal solution at the kth iteration is given by:

$$\pi_k(a|s) \propto \pi_{k-1}(a|s)\exp\Big(\frac{1}{\beta}\big(r(s,a) + \gamma\mathbb{E}_{s'\sim\mathcal{P}(\cdot|s,a)}V_{\pi_k}(s')\big)\Big) \tag{10}$$

$$\propto \pi_{\text{base}}(a|s)\exp\Big(\frac{k}{\beta}r(s,a) + \sum_{i=1}^{k}\frac{\gamma}{\beta}\mathbb{E}_{s'\sim\mathcal{P}(\cdot|s,a)}V_{\pi_i}(s')\Big), \tag{11}$$

where π_{base} is the initial base policy. This formulation demonstrates that by iteratively incorporating the base model, reward model, and previously trained value functions, we can achieve successive policy improvement at inference time without directly fine-tuning the base model.

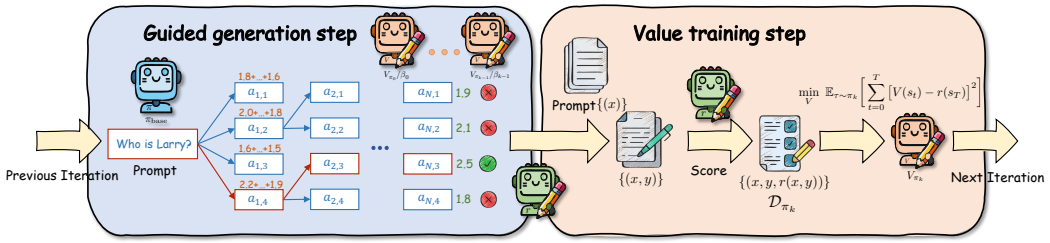

Figure 1: Illustration of our algorithm: Test-Time Successive Policy Iterations (SPI). The framework consists of two stages: (1) a guided generation step that produces optimized policies based on current value estimates and (2) a value training step that trains a new value function based on a current new policy.

3 Algorithm Design

In this section, we design the practical implementation for the proposed inference time SPI formulation. While the theoretical formulation provides a framework for iterative policy improvement, directly applying the previous algorithm is not feasible. Specifically, in language models, the action space consists of an extensive vocabulary (for example, GPT-4 has a vocabulary size of 100256), making it computationally infeasible to compute policy updates by evaluating all possible actions. Therefore, we introduce a practical approach that addresses this challenge and ensures efficient implementation. On a high level, the proposed algorithm alternates between two key steps: the value training step and the guided generation step. In the **value training step**, the algorithm updates the value function under a fixed policy π. In the **guided generation step**, a search-based method is employed to optimize the generation process by leveraging the learned value function. Below, we provide detailed descriptions of these components, which are also shown in Fig. 1.

Value training step. In the k-th iteration, given the continuations generated by the guided generation step (in the first iteration, the base model generates the continuation directly), this step aims to train a corresponding value function V_{π_k}, enabling a policy improvement step according to Proposition 1. Specifically, we leverage the set of previously trained value functions $\{V_{\pi_i}\}_{i=0}^{k-1}$ to approximate the improved policy π_k by equation 11. This updated policy is then used to generate continuations, with a reward model providing scores to collect a new dataset $\mathcal{D}_k = \{(x, y, r(x, y))\}$ to train the value function V_{π_k} for the next round of policy updates.

The objective of the value training step is to minimize the discrepancy between the predicted values and the actual return values. There are several approaches to achieve this. For example, Schulman et al. (2017) employs temporal difference learning and generalized advantage estimation (Schulman et al., 2015) to learn the value function, while Farebrother et al. (2024) uses a cross-entropy loss. In this work, we adopt a direct regression approach, where the predicted values are regressed to the observed returns, following Yang & Klein (2021); Mudgal et al. (2023). We choose this method for its simplicity and stability, and it is given by:

$$V_\pi := \arg\min_V \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{T}\left[V(s_t) - r(s_T)\right]^2\right]. \tag{12}$$

Guided generation step. In this step, our objective is to sample the generation by the improved policy. Specifically, during the inference stage at the k-th iteration, the log probability of generating an action a given state s is determined by both the summation of previously trained value functions and the base policy. This can be expressed as:

$$\log \pi_k(a|s) \propto \log \pi_{\text{base}}(a|s) + \sum_{i=0}^{k-1}\frac{1}{\beta_i}r(s,a) + \sum_{i=0}^{k-1}\frac{\gamma}{\beta_i}\mathbb{E}_{s'\sim\mathcal{P}(\cdot|s,a)}V_{\pi_i}(s'), \tag{13}$$

where $\pi_0 = \pi_{\text{base}}$. In the standard RLHF training pipeline (Ouyang et al., 2022; Bai et al., 2022), the outcome-based reward model is commonly adopted, which means that the rewards are sparse

and typically assigned only at the terminal state, e.g., the EOS token (Ouyang et al., 2022). In addition, to ensure the policy satisfies the simplex constraint, we adopt a softmax to normalize the distribution. As a result, this simplifies the equation 13 to:

$$\log \pi_k(a|s) \propto \text{Softmax}\left(\log \pi_{\text{base}}(a|s) + \sum_{i=0}^{k-2} \frac{\gamma}{\beta_i} \mathbb{E}_{s' \sim \mathcal{P}(\cdot|s,a)} V_{\pi_i}(s')\right) + \frac{\gamma}{\beta_{k-1}} \mathbb{E}_{s' \sim \mathcal{P}(\cdot|s,a)} V_{\pi_{k-1}}(s'). \tag{14}$$

In practice, exploring the entire action space (i.e., the full vocabulary) at each step is computationally infeasible, especially in large-scale language models. On the other hand, Tree search methods like Monte Carlo Tree Search offer thorough exploration but are prohibitively expensive for real-time or large-scale applications (Browne et al., 2012)

To address this, we adopt a beam search variant inspired by Zhou et al. (2024), which offers a more computationally efficient alternative. In this approach, multiple candidate nodes (tokens) are expanded according to the original policy, and the top k nodes (tokens) are selected based on evaluations from the value function. In addition, following Zhou et al. (2024), we incorporate *chunked sampling*, where multiple tokens are generated as a single node rather than expanding token-by-token. This significantly reduces the number of value function queries, improving efficiency without compromising the quality of generated sequences.

However, we observe that this method often results in redundant or identical chunks, which limits the diversity of the search space and may hinder the exploration of potentially better sequences. To mitigate this issue, we incorporate a *diversity-first* principle into our selection process. Specifically, we prioritize maintaining a diverse set of candidate nodes by first clustering the same nodes and ensuring that each cluster contributes at least one representative to the next beam. Finally, we use the reward model to choose the best one. We summarize the proposed algorithm in Algorithm 1 and 2, corresponding to the value training and guided generation steps.

Remark 3.1 *Although the continuations generated during the guided generation step follow the same distribution as the explicit optimal policy, approximation errors may arise in equation 12. Specifically, after collecting generations from the decoding step, solving equation 12 can be viewed as a supervised learning problem that can be solved by stochastic gradient descent. From Vershynin (2018), we know that the generalization error in such empirical risk minimization is bounded by the bound $\mathcal{O}(\sqrt{d/n})$ (d is the VC Dimension), which implies that V_{π_k} can be effectively approximated given a sufficiently large dataset generated by π_k. Further, as shown in Table 1, this approximation error could be significantly reduced as the size of the sampled generation increases. In this case, the proposed method becomes equivalent to multi-iteration policy optimization.*

Remark 3.2 *As the number of iterations increases, an increasing number of value functions are required to guide the base policy in obtaining the latest policy. This can introduce a significant external memory burden. A potential solution to mitigate this issue is to keep the backbone of the value function fixed and fine-tune only the value head, reducing memory overhead while maintaining adaptability. Additionally, another approach is to distill a single model to approximate the aggregated scores from all previous value function models, effectively compressing the information and reducing memory cost.*

4 EXPERIMENT

In this section, we demonstrate the effectiveness of our proposed method on TL;DR summarization dataset (Stiennon et al., 2020) in Section 4.1, and its use in UltraFeedback dataset in Section 4.2.

4.1 ALIGNING LLM WITH TL;DR DATASET

Model and Datasets. We use the SFT model based on `EleutherAI/pythia` family with size 1b and 6.9b, fine-tuned on TL;DR text-summarization dataset following the methodology outlined in Huang et al. (2024b). We utilize a public reward model[2] trained from `Pythia-6.9b` with `TL;DR`

[2]see https://huggingface.co/vwxyzjn/EleutherAI_pythia-6.9b-deduped__reward__tldr

Algorithm 1: Inference-time successive policy iterations

Input: Preference datasets $\mathcal{P} := \{(x, y_l, y_w)\}$, prompt datasets $\mathcal{D} := \{(x)\}$, a base policy π_{base}, reward model r.
Output: Updated policy in the inference time.
Train a reward model r using preference datasets $\mathcal{P} := \{(x, y_l, y_w)\}$.
Initialize $\pi_0 = \pi_{\text{base}}$.
for $k = 0$ to K **do**
    Roll out by using current policy π_k on prompt dataset $\mathcal{D}$ to generate continuations y to collect a dataset.
    Using the trained reward model r to evaluate the prompt-continuation pairs to create $\mathcal{D}_{\pi_k} := \{(x, y, r(x, y))\}$.
    Train a value function V_{π_k} by minimizing the loss (12) on $\mathcal{D}_{\pi_k}$.
    Update policy π_{k+1} by using the guided generation step (Algorithm. 2) with $\{V_{\pi_i}\}_{i=0}^{k}$.
end for
return π_{K+1}

Algorithm 2: Guided generation step at the kth iteration

Input: Prompt x, beam width W, successors per state K, chunk length L, previous value function list $\{V_i\}_{i=1}^{k}$, base policy π_{base}, reward model r
Output: Response y^* with the highest reward.
Generate N partial responses with length L to initialize the candidate set $\mathcal{C} = \{y_1, y_2, ..., y_N\}$
while $\exists y' \in \mathcal{C}$ such that y' is incomplete **do**
    Query the value function list $\{V_i\}_{i=0}^{k-1}$ to compute the value scores v_j for each candidate y_j as $v_j = \sum_{i=1}^{k} \frac{1}{\beta_i} V_i(x, y_j)$.
    Select the top K candidates following the diversity-first principle to form a parent set.
    Generate the W child node with maximum length L for each parent node to update the candidate set $\mathcal{C}$.
end while
return $y^* = \arg\max_{y' \in \mathcal{C}} r(x, y')$

dataset as the evaluator and `Pythia-1b` reward model for the scorer. For training details of the value function, we initialize the backbone with the parameters of the policy model and train it on a scored dataset comprising 8192 samples per iteration.

Baselines. Our test-time alignment baselines include Best-of-N (BoN), which generates $N = 16$ candidate full responses, evaluates them using the outcome reward model, and selects the response with the highest reward. Additionally, we compare against the training time alignment method DPO.

Generation and Evaluation. During generation, we use $T = 0.7$, top-k $= 50$, top-p $= 1.0$, with a maximum length 53 and chunk length of $L = 8$. For evaluation, we sample a subset of 300 instances from the test dataset. We also record the win rate of the model generated summary against the reference summary (human-written reference), evaluated by GPT-4o-mini (the GPT prompt can be found in Appendix C). In our algorithm, we use $\gamma = 1$ and $\beta_k = 1$ for all iterations.

Result. As shown in Fig. 2, our method outperforms both the BoN method and training time method DPO, even in the first iteration when using 1b value functions to guide 1b policy. Additionally, inference-time SPI demonstrates consistent improvements in both reward score and win rate compared to the ground truth summary across multiple iterations. For the weak to strong guidance, like 1b value functions guiding the 6.9b policy, we observe that our method is still better than the BoN. While our method achieves comparable performance to the training-based DPO method on reward scores, it shows a lower win rate against DPO. This discrepancy is likely attributable to the size disparity between models - our method uses only a 1B model trained for test-time guidance, whereas DPO directly trains the larger 6.9B model, allowing it to better internalize the preference information.

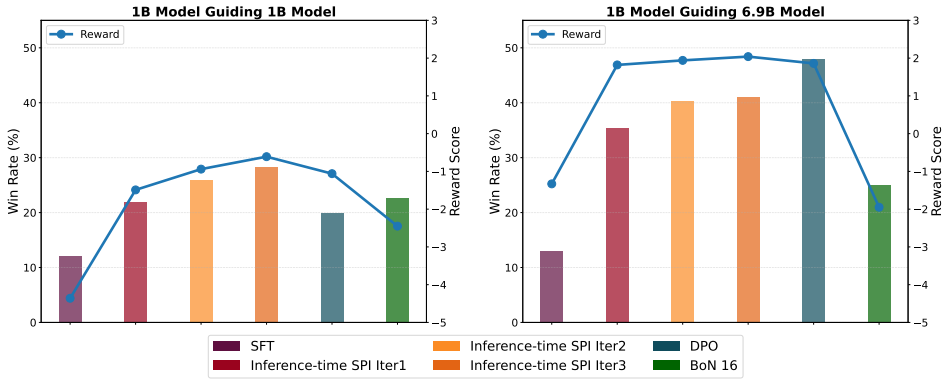

Figure 2: Reward score (line) and win rates (bar) evaluated by GPT-4o-mini across multiple iterations on 300 samples from the test dataset. The decoding process maintains a beam width of 4 with 4 candidates preserved per state (following the methodology in Zhou et al. (2024)).

4.2 Aligning LLM with UltraFeedback Dataset

In this section, we evaluate the effectiveness of our method in the weak-to-strong guidance setting, where a value function trained on smaller, weaker LLM is used to guide larger, more capable base LLM.

Model and Datasets. We use `Meta-Llama-3-8B-Instruct`[3] as our base policy. To get one explicit reward model, we initialize the reward model from `mistralai/Mistral-7B-v0.1`, and then train it on the UltraFeedback dataset (Cui et al., 2023)[4]. The value function is initialized from the reward model and further trained through using the model's rollouts / generations.

Baseline. We consider the following baselines: Best-of-N with explicit reward (BoN-E) and implicit reward (BoN-I), and weak-to-strong search (Zhou et al., 2024). For the implicit reward, we use the relative log probability between `zephyr-7b-beta` and `mistral-7b-sft-beta`, where `zephyr-7b-beta` is fine-tuned based on the `mistral-7b-sft-beta` using DPO loss. For fairness, all models are trained on the UltraFeedback dataset, ensuring a consistent comparison across methods.

Generation and Evaluation. For generation, we use the $T = 0.6$, top-k $= 50$ and top-p $= 0.9$ with a maximum length 2048. We evaluate the performance on a standard single-turn instruction-following benchmark, AlpacaEval 2.0 (Li et al., 2023), which consists of 805 prompts from various open-source datasets. Our evaluation reports both the raw win rate and the length-controlled (LC) win rate (Dubois et al., 2024), the latter being a metric specifically designed to mitigate biases arising from model verbosity. We also conduct an ablation study on a 200-sample subset of the AlpacaEval 2.0 dataset (see Section 4.2.1). We also use a held-out 8B reward model (Xiong et al., 2024; Dong et al., 2023), `sfairXC/FsfairX-LLaMA3-RM-v0.1`[5] as the ground-truth reference for evaluating with the formula win rate + $\frac{1}{2}$ tie rate to measure generation quality compared against the BoN.

Result. The evaluation result is shown in Table 2. Inference-time SPI demonstrates consistent and substantial improvements over the baseline compared against the response generated by GPT-4. Notably, BoN-E significantly outperforms other baseline algorithms, implying that BoN with an explicit reward serves as an efficient and effective inference-time alignment method. Furthermore, by leveraging the value function to conduct a fine-grained search during decoding in multiple iterations, our method achieves successive performance gains that substantially exceed those of BoN-E. This confirms our intuition that rather than requiring additional computational resources to fine-tune the model during training, significant improvements can be achieved successively through value-guided exploration at inference time. Here, we adopt an increasing parameter β as iterations proceed. This

[3]https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct
[4]https://huggingface.co/datasets/HuggingFaceH4/ultrafeedback_binarized
[5]https://huggingface.co/sfairXC/FsfairX-LLaMA3-RM-v0.1

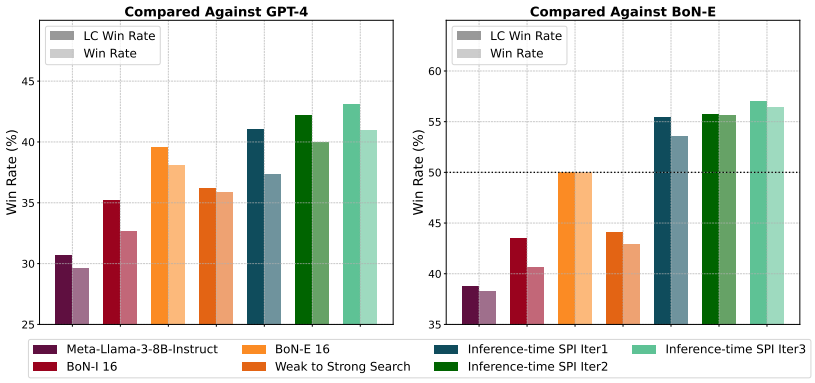

Figure 3: AlpacaEval 2 length-controlled win rate and raw win rate compared against GPT-4 (left) and BoN-E (right), evaluated by GPT-4. We use $\beta_1 = 1, \beta_2 = 2, \beta_3 = 2.5$ while neglecting the $\log \pi(a|s)$, which places more emphasis on the learned value function during generation. The decoding process maintains a beam width of 4 with 4 candidates preserved per state, and $L = 16$ tokens as a state. For fairness, we use $N = 16$ for BoN. The numerical result is shown in Appendix D.

more conservative strategy becomes necessary to ensure stability and prevent excessive deviation from the previous policy.

4.2.1 Ablation Study

Explore different sampling strategies for inference-time SPI. We investigate the impact of different sampling strategies during the guided generation step on the performance. We compare four sampling approaches: beam search variants (Zhou et al., 2024), with and without the diversity-first principle, and two stochastic sampling methods. For stochastic sampling, we sample the next text chunks using a softmax distribution weighted by the value function, i.e., $a \sim \frac{\exp(V_i/T)}{\sum_j \exp V_j/T}$ with different temperature T.

As shown in Figure 4, incorporating the diversity-first principle into beam search significantly improves performance, making the search more effective. While stochastic sampling methods also perform well, their effectiveness is highly sensitive to the temperature parameter T.

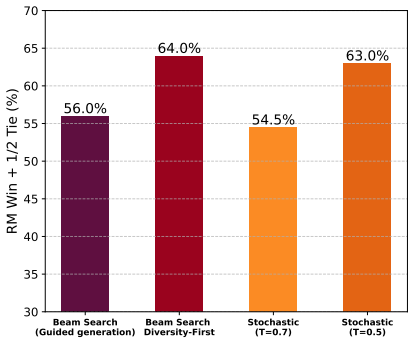

Figure 4: Comparison of different sampling strategies against BoN method under different sampling strategies during guided generation step on the 200 sub dataset.

Explore different chunk lengths for inference-time SPI. We further explore the effect of chunk length on the performance of our method compared against the BoN method with $N = 16$ (see Figure 5). We can see that the performance of inference-time SPI increases and then decreases as the chunk length increases. Intuitively, when the number of provided tokens is very small, the value function is insufficient to distinguish the quality of different candidates. As the token length increases to a certain extent, the value function can make more accurate judgments between different candidates. As the chunk length increases toward the max length, the algorithm gradually converges to the BoN method, causing the win rate relative to BoN to approach 50%.

Explore different data sizes on training a value function for inference-time SPI. Table 1 shows the impact of data size for training a value function on the performance under different chunk length settings. When the data size is small (e.g., 1024 or 2048), larger chunk lengths (L = 32, 64) yield better results. This is likely because the value function, trained on limited data, lacks the precision to provide accurate estimates for smaller token sequences. In contrast, as data size increases (e.g.,

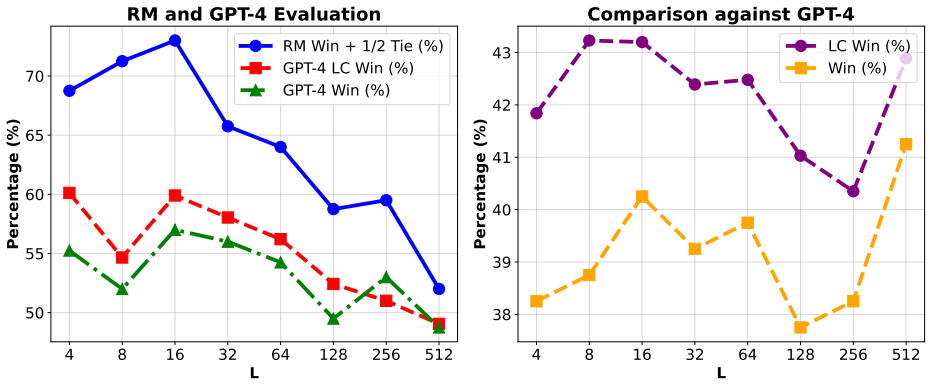

Figure 5: The effect of the chunk length on inference-time SPI compared against BoN with $N = 16$ on a 200-sample sub-dataset. Left: The reward and GPT-4 evaluation on the inference-time SPI compared against BoN. Right: The comparison against GPT-4. Numerical result are in Appendix D

4096 or 8192), smaller chunks (L = 16) perform better, suggesting that finer-grained search leads to a more accurate inference-time policy update.

Chunk Length	Data Size	RM Win +$\frac{1}{2}$Tie (%)	GPT-4 Evaluation (%)	
			LC Win	Win
16	1024	49.75%	48.61%	46.66%
	2048	49%	51.26%	49.68%
	4096	60%	56.45%	53.25%
	8192	73%	59.9%	57%
32	1024	55%	48.52%	45.51%
	2048	57.75%	50.92%	51.85%
	4096	63%	48.96%	48.75%
	8192	66%	58.04%	56%
64	1024	51%	53.64%	50.24%
	2048	48%	54.63%	54.5%
	4096	61.25%	53.52%	51.73%
	8192	64%	56.21%	54.25%

Table 1: Comparison of inference-time SPI and BoN-16, evaluated by the reward model and GPT-4, across different chunk lengths and data sizes used for training the value function.

5 Conclusion

We introduced Inference-Time Successive Policy Iterations (SPI), a novel algorithm that enables multi-round policy improvement at inference time and is theoretically equivalent to fine-tuning the policy model through reinforcement learning. Empirically, our method outperforms existing inference-time alignment approaches, demonstrating the ability to successively enhance LLM's performance in reasoning and planning without requiring direct fine-tuning.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Alekh Agarwal, Nan Jiang, Sham M Kakade, and Wen Sun. Reinforcement learning: Theory and algorithms. *CS Dept., UW Seattle, Seattle, WA, USA, Tech. Rep*, 32, 2019.

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms. *arXiv preprint arXiv:2402.14740*, 2024.

Mohammad Gheshlaghi Azar, Mark Rowland, Bilal Piot, Daniel Guo, Daniele Calandriello, Michal Valko, and Rémi Munos. A general theoretical paradigm to understand learning from human preferences. *arXiv preprint arXiv:2310.12036*, 2023.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

Souradip Chakraborty, Soumya Suvra Ghosal, Ming Yin, Dinesh Manocha, Mengdi Wang, Amrit Singh Bedi, and Furong Huang. Transfer q star: Principled decoding for llm alignment. *arXiv preprint arXiv:2405.20495*, 2024.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Wei Zhu, Yuan Ni, Guotong Xie, Zhiyuan Liu, and Maosong Sun. Ultrafeedback: Boosting language models with high-quality feedback. 2023.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. Raft: Reward ranked finetuning for generative foundation model alignment. *arXiv preprint arXiv:2304.06767*, 2023.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*, 2024.

Jesse Farebrother, Jordi Orbay, Quan Vuong, Adrien Ali Taïga, Yevgen Chebotar, Ted Xiao, Alex Irpan, Sergey Levine, Pablo Samuel Castro, Aleksandra Faust, et al. Stop regressing: Training value functions via classification for scalable deep rl. *arXiv preprint arXiv:2403.03950*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Seungwook Han, Idan Shenfeld, Akash Srivastava, Yoon Kim, and Pulkit Agrawal. Value augmented sampling for language model alignment and personalization. *arXiv preprint arXiv:2405.06639*, 2024.

Shibo Hao, Yi Gu, Haotian Luo, Tianyang Liu, Xiyan Shao, Xinyuan Wang, Shuhua Xie, Haodi Ma, Adithya Samavedhi, Qiyue Gao, et al. Llm reasoners: New evaluation, library, and analysis of step-by-step reasoning with large language models. *arXiv preprint arXiv:2404.05221*, 2024.

Joey Hejna and Dorsa Sadigh. Inverse preference learning: Preference-based rl without a reward function. *Advances in Neural Information Processing Systems*, 36, 2024.

James Y Huang, Sailik Sengupta, Daniele Bonadiman, Yi-an Lai, Arshit Gupta, Nikolaos Pappas, Saab Mansour, Katrin Kirchhoff, and Dan Roth. Deal: Decoding-time alignment for large language models. *arXiv preprint arXiv:2402.06147*, 2024a.

Shengyi Huang, Michael Noukhovitch, Arian Hosseini, Kashif Rasul, Weixun Wang, and Lewis Tunstall. The n+ implementation details of rlhf with ppo: A case study on tl; dr summarization. *arXiv preprint arXiv:2403.17031*, 2024b.

Maxim Khanov, Jirayu Burapacheep, and Yixuan Li. Args: Alignment as reward-guided search. *arXiv preprint arXiv:2402.01694*, 2024.

Lingkai Kong, Haorui Wang, Wenhao Mu, Yuanqi Du, Yuchen Zhuang, Yifei Zhou, Yue Song, Rongzhi Zhang, Kai Wang, and Chao Zhang. Aligning large language models with representation editing: A control perspective. *arXiv preprint arXiv:2406.05954*, 2024.

Wendi Li and Yixuan Li. Process reward model with q-value rankings. *arXiv preprint arXiv:2410.11287*, 2024.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Alpacaeval: An automatic evaluator of instruction-following models, 2023.

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

Zhixuan Liu, Zhanhui Zhou, Yuanfu Wang, Chao Yang, and Yu Qiao. Inference-time language model alignment via integrated value guidance. *arXiv preprint arXiv:2409.17819*, 2024.

Sidharth Mudgal, Jong Lee, Harish Ganapathy, YaGuang Li, Tao Wang, Yanping Huang, Zhifeng Chen, Heng-Tze Cheng, Michael Collins, Trevor Strohman, et al. Controlled decoding from language models. *arXiv preprint arXiv:2310.17022*, 2023.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35: 27730–27744, 2022.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Jiahao Qiu, Yifu Lu, Yifan Zeng, Jiacheng Guo, Jiayi Geng, Huazheng Wang, Kaixuan Huang, Yue Wu, and Mengdi Wang. Treebon: Enhancing inference-time alignment with speculative tree-search and best-of-n sampling. *arXiv preprint arXiv:2410.16033*, 2024.

Rafael Rafailov, Joey Hejna, Ryan Park, and Chelsea Finn. From r to q^*: Your language model is secretly a q-function. *arXiv preprint arXiv:2404.12358*, 2024a.

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

John Schulman. Trust region policy optimization. *arXiv preprint arXiv:1502.05477*, 2015.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021, 2020.

Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

Wei Xiong, Hanze Dong, Chenlu Ye, Ziqi Wang, Han Zhong, Heng Ji, Nan Jiang, and Tong Zhang. Iterative preference learning from human feedback: Bridging theory and practice for rlhf under kl-constraint. In *Forty-first International Conference on Machine Learning*, 2024.

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

Yuancheng Xu, Udari Madhushani Sehwag, Alec Koppel, Sicheng Zhu, Bang An, Furong Huang, and Sumitra Ganesh. Genarm: Reward guided generation with autoregressive reward model for test-time alignment. *arXiv preprint arXiv:2410.08193*, 2024b.

Kevin Yang and Dan Klein. Fudge: Controlled text generation with future discriminators. *arXiv preprint arXiv:2104.05218*, 2021.

Xinnan Zhang, Siliang Zeng, Jiaxiang Li, Kaixiang Lin, and Mingyi Hong. Llm alignment through successive policy re-weighting (spr). In *NeurIPS 2024 Workshop on Fine-Tuning in Modern Machine Learning: Principles and Scalability*, 2024.

Zhanhui Zhou, Zhixuan Liu, Jie Liu, Zhichen Dong, Chao Yang, and Yu Qiao. Weak-to-strong search: Align large language models via searching over small language models. *arXiv preprint arXiv:2405.19262*, 2024.

A Related work

A.1 Reinforcement learning with Human Feedback

Reinforcement learning from human feedback (RLHF) (Stiennon et al., 2020; Ouyang et al., 2022) is a widely adopted technique for fine-tuning AI systems to align with human preferences and values. Current RLHF approaches typically involve training a reward model using human preference feedback and then fine-tuning the language model via proximal policy optimization (PPO) (Schulman et al., 2017). In addition to PPO, other reinforcement learning solvers such as RLOO (Ahmadian et al., 2024) and GRPO (Shao et al., 2024) have also demonstrated effectiveness in advanced foundation language models. However, optimizing these algorithms for peak performance requires substantial effort and resources, which are often beyond the reach of the open-source community.

A.2 Training-time Alignment

In an effort to reduce computational overhead in reinforcement learning, alternative alignment strategies such as Direct Preference Optimization (DPO) (Rafailov et al., 2024b) and Inverse Preference Learning (IPL) (Hejna & Sadigh, 2024) remove the need for explicit reward modeling by extracting policies directly from preference data. While this substantially lowers training complexity, these algorithms can be unstable during training (Azar et al., 2023; Xu et al., 2024a), and once the preference dataset is fixed, they offer limited opportunities for further policy improvement.

A.3 INFERENCE-TIME ALIGNMENT

Although training-time alignment methods are effective, they demand substantial computational and engineering resources. To mitigate these costs, decoding-time alignment approaches typically freeze pre-trained models and adjust their outputs during a decoding phase, which is handled by smaller, specialized models (Zhou et al., 2024; Liu et al., 2024).

The conceptual framework for aligning language models at decoding time is rooted in the use of value function (Mudgal et al., 2023) or reward function (Khanov et al., 2024). Khanov et al. (2024) treat alignment as a Reward-Guided framework that integrates alignment into the decoding process without considering a long-term return in token-level MDP. Liu et al. (2024) and Mudgal et al. (2023) proposed methods leveraging explicit and implicit value functions trained via a KL-regularized reinforcement learning objective. However, these approaches rely solely on the value function derived from the pre-trained model for guided generation, which can result in limited policy improvements.

Informed by these insights, we propose a method that integrates sequentially trained value functions to enhance alignment with human preferences during the model's decoding phase. Our empirical findings, presented in Section 4, demonstrate that the proposed method captures richer contextual understanding and yields superior results.

B PROOF

In this section, we show the solution to the constrained policy optimization problem defined in equation 8a-equation 8d.

B.1 PROOF OF THEOREM 1

In this proof, we will first write down the *partial* Lagrangian function, which only considers the constraints equation 8b-equation 8c. After solving the partial Lagrangian function, we will show that the constraint equation 8d is satisfied.

Let α and $\zeta := \{\zeta_s | s \in \mathcal{S}\}$ denote the dual variables of the constraints equation 8b and equation 8c, respectively. Then the partial Lagrangian function can be expressed as below:

$$\mathcal{L}(\pi, \alpha, \zeta) := \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_{\pi_{\text{base}}}(\cdot), a\sim\pi(\cdot|s)}\big[A_{\pi_{\text{base}}}(s,a)\big] + \alpha\Big(\epsilon - \mathbb{E}_{s\sim d_{\pi_{\text{base}}}(\cdot)}\big[D_{\text{KL}}\big(\pi(\cdot|s)||\pi_{\text{base}}(\cdot|s)\big)\big]\Big) + \sum_{s\in\mathcal{S}}\zeta_s\big(1 - \sum_{a\in\mathcal{A}}\pi(a|s)\big).$$

Through taking partial derivative of $\mathcal{L}(\pi, \alpha, \zeta)$ w.r.t. $\pi(a|s)$, we can obtain the following equation:

$$\frac{\partial}{\partial\pi(a|s)}\mathcal{L}(\pi,\alpha,\zeta) = \frac{1}{1-\gamma}d_{\pi_{\text{base}}}(s)A_{\pi_{\text{base}}}(s,a) - \alpha d_{\pi_{\text{base}}}(s)\Big(-\log\pi_{\text{base}}(a|s) + \log\pi(a|s) + 1\Big) - \zeta_s.$$

Through setting the partial derivative $\frac{\partial}{\partial\pi(a|s)}\mathcal{L}(\pi,\alpha,\zeta)$ to 0, we obtain

$$\frac{1}{1-\gamma}d_{\pi_{\text{base}}}(s)A_{\pi_{\text{base}}}(s,a) - \alpha d_{\pi_{\text{base}}}(s)\Big(-\log\pi_{\text{base}}(a|s) + \log\pi(a|s) + 1\Big) - \zeta_s = 0.$$

Then we obtain the closed-form expression of the optimal policy π^* as below:

$$\log\pi^*(a|s) = \frac{A_{\pi_{\text{base}}}(s,a)}{(1-\gamma)\alpha} + \log\pi_{\text{base}}(a|s) - 1 - \frac{\zeta_s}{\alpha d_{\pi_{\text{base}}}(s)}, \tag{15a}$$

$$\pi^*(a|s) = \pi_{\text{base}}(a|s)\exp\Big(\frac{A_{\pi_{\text{base}}}(s,a)}{(1-\gamma)\alpha}\Big)\exp\Big(-1 - \frac{\zeta_s}{\alpha d_{\pi_{\text{base}}}(s)}\Big). \tag{15b}$$

Here we can denote $\beta := (1-\gamma)\alpha$. Then according to the expression of $\pi(a|s)$ in equation 15b, we obtain the following relation:

$$\pi(a|s) \propto \pi_{\text{base}}(a|s)\exp\Big(\frac{1}{\beta}A_{\pi_{\text{base}}}(s,a)\Big). \tag{16}$$

Based on the constraint equation 8c, we know that $\pi(\cdot|s)$ is a distribution so that $\sum_{a\in\mathcal{A}}\pi(a|s)=1$. Therefore, according to the expressions in equation 15b and equation 16, we can obtain the following expression of the optimal policy π^* as below:

$$\pi^*(a|s)=\frac{\pi_{\text{base}}(a|s)\exp\left(\frac{1}{\beta}A_{\pi_{\text{base}}}(s,a)\right)}{\sum_{a'\in\mathcal{A}}\pi_{\text{base}}(a'|s)\exp\left(\frac{1}{\beta}A_{\pi_{\text{base}}}(s,a')\right)}.$$

Recall that $A_{\pi_{\text{base}}}(s,a):=Q_{\pi_{\text{base}}}(s,a)-V_{\pi_{\text{base}}}(s)$ has been defined in equation 3, then we can rewrite the expression of $\pi^*(a|s)$:

$$\begin{aligned}\pi^*(a|s)&=\frac{\pi_{\text{base}}(a|s)\exp\left(\frac{1}{\beta}\left(Q_{\pi_{\text{base}}}(s,a)-V_{\pi_{\text{base}}}(s)\right)\right)}{\sum_{a'\in\mathcal{A}}\pi_{\text{base}}(a'|s)\exp\left(\frac{1}{\beta}\left(Q_{\pi_{\text{base}}}(s,a')-V_{\pi_{\text{base}}}(s)\right)\right)}\\&=\frac{\pi_{\text{base}}(a|s)\exp\left(\frac{1}{\beta}Q_{\pi_{\text{base}}}(s,a)\right)}{\sum_{a'\in\mathcal{A}}\pi_{\text{base}}(a'|s)\exp\left(\frac{1}{\beta}Q_{\pi_{\text{base}}}(s,a')\right)}. \quad (17)\\&=\frac{\pi_{\text{base}}(a|s)\exp\left(\frac{1}{\beta}(\gamma\mathbf{E}_{s'\sim\mathcal{P}(\cdot|s,a)}V_{\pi_{\text{base}}}(s')+r(s,a))\right)}{\sum_{a'\in\mathcal{A}}\pi_{\text{base}}(a'|s)\exp\left(\frac{1}{\beta}(\gamma\mathbf{E}_{s'\sim\mathcal{P}(\cdot|s,a')}V_{\pi_{\text{base}}}(s')+r(s,a'))\right)}. \quad (18)\end{aligned}$$

C GPT AS A JUDGE IN TL;DR TASK

System Prompt in TL;DR

Which of the following summaries does a better job of summarizing the most important points in the given forum post, without including unimportant or irrelevant details? Judge based on accuracy, coverage, and coherence.
Post:
<post>
Summary A:
<Summary A>
Summary B:
<Summary B>
FIRST provide a one-sentence comparison of the two summaries, explaining which you prefer and why. SECOND, on a new line, state only "A" or "B" to indicate your choice. Your response should use the format:
Comparison: <one-sentence comparison and explanation>
Preferred: <"A" or "B">

D EXTENDED EXPERIMENTAL RESULTS

Method	Compared Against GPT-4		Compared Against BoN-E	
	LC Win Rate	Win Rate	LC Win Rate	Win Rate
Meta-Llama-3-8B-Instruct	30.71	29.63	38.77	38.32
BoN-I 16	35.25	32.67	43.52	40.68
BoN-E 16	39.61	38.14	50.00	50.00
weak to strong search	36.20	35.90	44.13	42.92
Inference-time SPI Iter1	41.06	37.32	55.42	53.60
Inference-time SPI Iter2	42.20	40.00	55.75	55.65
Inference-time SPI Iter3	43.11	41.00	57.00	56.46

Table 2: AlpacaEval 2 length-controlled win rate and raw win rate compared against GPT-4. We use $\beta_1 = 1, \beta_2 = 2, \beta_3 = 2.5$, which places more emphasis on the learned value function during generation. The decoding process maintains a beam width of 4 with 4 candidates preserved per state, and $l = 16$ tokens as a state. For fairness, we use $N = 4 * 4$ for BoN.

D.1 CHUNK LENGTH ABLATION STUDY

L	RM Win (%)	RM Tie (%)	RM Win + $\frac{1}{2}$Tie (%)	GPT-4 Evaluation (%)		Compared with GPT-4	
				LC Win	Win	LC Win	Win
4	67.5%	2.5%	68.75%	60.12%	55.25%	41.84%	38.25%
8	69%	4.5%	71.25 %	54.66%	52.0%	43.23%	38.75%
16	**70.5%**	**5%**	**73%**	**59.9%**	**57.00%**	43.2 %	40.25%
32	63.5%	5%	65.75%	58.04%	56.00%	42.39 %	39.25%
64	61%	6%	64%	56.21%	54.25%	42.48%	39.75%
128	56.5%	4.5%	58.75%	52.42%	49.5%	41.03%	37.75%
256	57%	5%	59.5%	51.0%	53.00%	40.35%	38.25%
512	49%	6%	52%	49.05%	48.75%	42.89%	41.25%

Table 3: The effect of the chunk length on value guided compared against BoN with $N = 16$ on a 200-sample sub-dataset. GPT-4 LC Win means that two responses are judged by the GPT-4. The last column means the LC win rate compared with the result generated by GPT-4.