Beyond Magic Words: Sharpness-Aware Prompt Evolving for Robust Large Language Models with TARE

Guancheng Wan[1]* **Lucheng Fu**[2]* **Mengting Li**[1]
[1]University of California, Los Angeles [2]Georgia Institute of Technology
gcwan3@gmail.com

Abstract

The performance of Large Language Models (LLMs) hinges on carefully engineered prompts. However, prevailing prompt optimization methods, ranging from heuristic edits and reinforcement learning to evolutionary search, primarily target point-wise accuracy. They seldom enforce paraphrase invariance or searching stability, and therefore cannot remedy this brittleness in practice. Automated prompt search remains brittle: small, semantically preserving paraphrases often cause large performance swings. We identify this brittleness as the **textual sharpness** of the **prompt landscape**. In this work, we provide the first formal treatment of textual sharpness in the discrete, semantic space of prompts, together with an operational robustness criterion over a semantic neighborhood; the design is black-box or API-only, requiring no gradients to update the model's parameters. Then we introduce `TARE` (Textual Sharpness-Aware Evolving), a derivative-free framework that alternates between an inner, sampling-based adversarial search that stresses a prompt with hard paraphrases and an outer, robust selection that prefers candidates whose neighborhoods remain strong. We further propose `ATARE`, which learns anisotropic weights to shape the semantic neighborhood and adapts its radius over time to balance exploration and fidelity. Diverse tasks evaluate our methods, whose design for minimizing textual sharpness gap leads to prompts that preserve accuracy under paraphrasing, outperforming accuracy-only prompt search while remaining computationally practical. The code is available for anonymous access at `https://anonymous.4open.science/r/ATARE_TARE/`.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities across a wide array of natural language understanding and generation tasks [1, 29]. The efficacy of these models, however, is critically dependent on the quality of their input prompts. In this vein, prompt engineering aims at manually or automatically discovering optimal prompt structures to guide LLMs toward desired outputs. While automated methods [13, 47] have shown promise, they often produce prompts that are highly sensitive to minor, semantically-equivalent perturbations. An optimized prompt that performs well on a given set of inputs may fail dramatically when faced with slight paraphrasing or rephrasing, a phenomenon we term the "sharpness" of the prompt landscape. This brittleness severely limits the real-world reliability and robustness of LLM-based systems.

The concept of "sharpness" in optimization landscapes is well-studied in the domain of deep neural networks. It has been shown that models converging to flat minima in the loss landscape exhibit

*Equal Contribution.

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

superior generalization performance [15]. Sharpness-Aware Minimization (SAM) [10] and its variants have emerged as powerful techniques to explicitly search these flat minima, thereby improving model robustness and generalization. These methods work by minimizing the loss in a "neighborhood" around the current parameters, effectively smoothing the loss landscape. However, the principles of SAM have been predominantly applied to continuous parameter spaces, such as model weights. Their application to the discrete and combinatorial nature of text-based prompts remains a significant and unexplored challenge.

Therefore, a natural research question arises: **I)** ***How can we formally define and quantify the concept of a "sharpness neighborhood" within the discrete, semantic space of textual prompts?*** Due to the discrete and semantically rich nature of text, traditional notions of local perturbations (e.g., infinitesimal gradient steps) are fundamentally inapplicable. Instead, we must construct neighborhoods that capture semantic similarity—accounting for paraphrasing and rephrasing—so that local sharpness reflects true linguistic and behavioral proximity relevant to LLMs. Defining such neighborhoods and associated metrics is crucial for robust optimization. Building on this, a closely related question arises: **II)** ***How can we design a practical optimization algorithm that navigates this discrete landscape to discover prompts that are both effective and robust to semantic perturbations?*** This requires an algorithm that can efficiently explore the prompt space while incorporating a measure of *landscape flatness* into its search process.

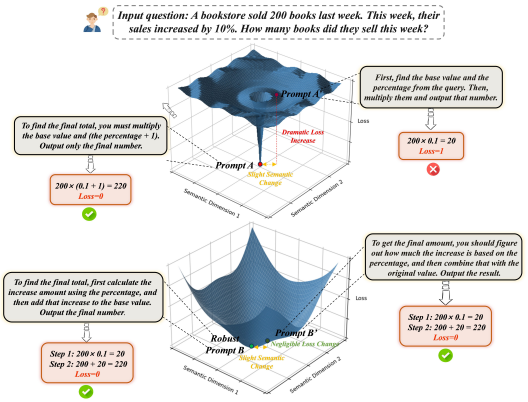


Figure 1: **Problem Illustration.** We illustrate the core challenge in prompt optimization: **I)** conventional methods often find brittle, sharp minima (left), where a **slight semantic change** from an optimal prompt Prompt A to a paraphrase Prompt A' results in a **significant loss increase**. **II)** Our goal is to instead seek flat, stable solutions (right), where a similar **slight semantic change** from a robust prompt Prompt B to its paraphrase Prompt B' only causes the loss to **remain nearly unchanged**, indicating high robustness.

To address these challenges, we introduce ***Textual Sharpness-Aware Evolving*** (TARE), a novel framework inspired by SAM that adapts its core principles for the discrete domain of prompt engineering. Our core contribution is a textual sharpness metric that quantifies prompt robustness by evaluating its performance over a neighborhood of semantically similar variants. We then propose an evolutionary optimization algorithm that iteratively refines prompts, selecting candidates that exhibit both high performance and low sharpness. Furthermore, we introduce an adaptive version, ***Adaptive Textual Sharpness-Aware Evolving*** (ATARE), which dynamically adjusts the neighborhood size during optimization for efficiency and effectiveness. Our contributions are threefold:

❶ **Formalizing Textual Sharpness.** We introduce the first definition of sharpness tailored for the discrete, semantic space of prompts. This is accompanied by a metric to quantify prompt robustness by evaluating performance stability across a semantically coherent neighborhood, bridging the gap between continuous optimization theory and discrete language-based optimization.

❷ **Sharpness-Aware Prompt Evolution.** We propose TARE, a novel algorithm designed to explicitly navigate the discrete prompt landscape. By integrating our textual sharpness metric directly into its fitness function, TARE effectively co-optimizes for both high task performance and low sharpness, yielding prompts that are both effective and robust. We further enhance this with an adaptive variant, ATARE, which dynamically adjusts the neighborhood radius for greater efficiency.

❸ **Superior Robustness and Generalization.** Through extensive experiments on multiple benchmarks, we provide strong empirical evidence that our proposed methods, TARE and ATARE, consistently discover prompts that are significantly more robust and generalize better to unseen data compared to existing state-of-the-art prompt optimization techniques.

2 Problem Formulation

2.1 Problem Setup and Notation

We consider a black-box large language model and a discrete semantic space of textual prompts.

For a supervised task with a training set, the empirical prompt risk is

$$\mathcal{L}_{\mathcal{D}}(p) = \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} \ell\big(\mathcal{M}(p,x),y\big). \tag{1}$$

When the task is generative or judgment based, an evaluator maps model outputs to a numeric loss

$$\ell\big(\mathcal{M}(p,x),y\big) \equiv \mathcal{E}\big(\mathcal{M}(p,x),y\big). \tag{2}$$

The same definitions apply for validation and test. We focus on robust optimization that accounts for semantic neighborhoods of a prompt.

2.2 Semantic Neighborhoods of Prompts

To make this precise, we endow the prompt space with a semantic dissimilarity measure and define the isotropic neighborhood as

$$B\big(p,\rho_{\text{text}}\big) := \big\{\, p' \in \mathcal{P} : d_{\text{text}}(p,p') \le \rho_{\text{text}} \,\big\}. \tag{3}$$

In practice, the dissimilarity can reflect token-level edit distance, paraphrase embedding distance, or membership in transformation families such as rephrasing or style shifts.

Anisotropic neighborhoods. To capture heterogeneous sensitivity across semantic components of a prompt, we use an anisotropic metric

$$d_{\text{ani},\boldsymbol{W}_p}(p,p') := \big\|\, \boldsymbol{W}_p\, \Delta(p,p') \,\big\|_2, \tag{4}$$

and the corresponding ellipsoidal neighborhood

$$B_p\big(p,\rho_{\text{text}}\big) := \big\{\, p' \in \mathcal{P} : d_{\text{ani},\boldsymbol{W}_p}(p,p') \le \rho_{\text{text}} \,\big\}. \tag{5}$$

2.3 Textual Sharpness and Robust Risk

Building on these neighborhoods, the textual sharpness-aware loss is defined as the local worst-case risk over a semantic neighborhood

$$\mathcal{L}_{\text{S}}(p,\rho_{\text{text}}) := \max_{p'\in B(p,\rho_{\text{text}})} \mathcal{L}_{\mathcal{D}}(p'). \tag{6}$$

The corresponding robust optimization problem is

$$\min_{p\in\mathcal{P}} \mathcal{L}_{\text{S}}(p,\rho_{\text{text}}). \tag{7}$$

This mirrors the classical SAM perspective by replacing perturbations in parameter space with textual perturbations in a semantic neighborhood. In the discrete prompt setting, neighborhood exploration during the inner maximization can be operationalized by treating a generator as a sampler.

3 Methodology

3.1 Overview

Motivation. Accuracy-only prompt search is brittle: small paraphrases or rephrasings can flip outcomes, exposing sharp, non-flat regions of the prompt landscape discussed in the introduction. Our aim is to explicitly prefer prompts that remain effective under semantically-preserving perturbations. We operationalize the textual sharpness formalization in Sec. 2 into a robust criterion that penalizes local fragility, so that selected prompts demonstrate stability across their semantic neighborhoods. Equivalently, we aim to shrink the textual sharpness gap $\mathrm{Sharp}_{\rho_{\text{text}}}(p)$ defined in Sec. 2.

Design principles. (i) Black-box, derivative-free optimization compatible with LLM APIs and evaluator oracles; (ii) semantic neighborhoods that preserve task intent while revealing local sharpness; (iii) an inner worst-case search to expose adversarial neighbors; (iv) an outer robust update that

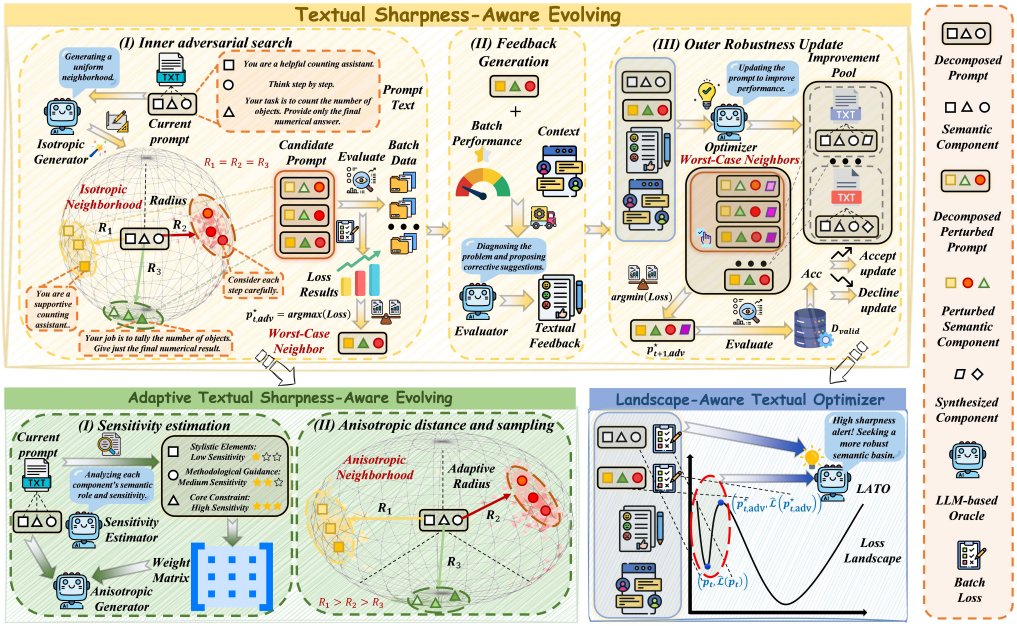


Figure 2: An illustration of our proposed TARE framework. (a) ***The top panel*** shows the main TARE loop, consisting of an Inner Adversarial Search, Feedback Generation, and an Outer Robustness Update. (b) ***The bottom-left panel*** details the ATARE mechanism, which uses Sensitivity Estimation to guide an efficient Anisotropic search. (c) ***The bottom-right panel*** presents the LATO, which perceives the local sharpness to guide updates towards a flatter semantic basin.

chooses candidates improving the max-risk estimate; and (v) lightweight schedules (radius and budget) for stability under limited compute.

Building on Sec. 2, our goal is to minimize the textual sharpness-aware risk by solving

$$\min_{p\in\mathcal{P}} \max_{p'\in B(p,\rho_{\text{text}})} \mathcal{L}(p'), \tag{8}$$

via a derivative-free, LLM-driven procedure. In this discrete, black-box setting, we rely on sampling-based inner maximization and validation-driven outer selection. We describe TARE (isotropic) and ATARE (anisotropic) variants that iteratively (i) search adversarial neighbors and (ii) update the prompt to reduce the robust objective.

Alignment to research questions. The neighborhood-based objective instantiates **Q1** by defining sharpness in semantic prompt space, while our two-stage, derivative-free robust evolution addresses **Q2** by providing a practical algorithm that co-optimizes task accuracy and local flatness under semantically-preserving perturbations.

3.2 TARE: Textual Sharpness-Aware Evolving

Let $p_0 \in \mathcal{P}$ be an initial prompt. At iteration $t = 0, 1, \ldots, T-1$, with radius $\rho_t > 0$ and minibatch $\mathcal{B}_t \subset \mathcal{D}$: To make our algorithm concrete, we trace a running example for a simple text-based object counting task. Let the initial prompt p_0 be:

> **Initial Prompt**: *You are a helpful counting assistant. Your task is to count the number of objects. Think step by step and then provide only the final numerical answer.*

For a given input text, the desired output is a single integer (e.g., "3"), and the loss function $\mathcal{L}$ is 1 if the output deviates from this format and 0 otherwise.

Inner adversarial search. We sample a candidate set inside the isotropic neighborhood using the generator oracle $\mathcal{G}$:

$$\mathcal{C}_{K_t}(p_t) := \{\, p'_1, \ldots, p'_{K_t} \,\} \sim \text{Sample}\big(\mathcal{G},\, p_t,\, \rho_t,\, K_t\big), \qquad p'_k \in B(p_t, \rho_t). \tag{9}$$

We evaluate the empirical loss on $\mathcal{B}_t$ and pick the worst case

$$p^\star_{t,\text{adv}} := \operatorname*{arg\,max}_{p' \in \mathcal{C}_{K_t}(p_t)} \widehat{\mathcal{L}}(p'; \mathcal{B}_t), \qquad \widehat{\mathcal{L}}_{\text{S}}(p_t; \rho_t) := \max_{p' \in \mathcal{C}_{K_t}(p_t)} \widehat{\mathcal{L}}(p'; \mathcal{B}_t). \tag{10}$$

To illustrate this process, consider the isotropic generator $\mathcal{G}$ acting on our initial counting prompt p_t. The generator produces perturbations that are semantically preserving, ensuring all candidates p' remain within the defined neighborhood $B(p_t, \rho_t)$. The goal is to test for fragility without altering the fundamental task. For example, the set of candidates $\mathcal{C}_{K_t}(p_t)$ might include the following paraphrases:

> ❶ **Candidate 1**: *You are a supportive counting assistant. Your job is to tally the number of objects. Consider each step carefully and then give just the final numerical result.*
>
> ❷ **Candidate 2**: *You are a helpful assistant for counting. Your role is to determine the number of objects. Think through each step and then offer only the final number.*
>
> ❸ **Candidate 3**: *You are a useful assistant for counting objects. Your task is to calculate how many objects there are. Reflect on each step and then present only the final numerical answer.*

A close analysis reveals that while the phrasing, synonyms (*"helpful"* $\rightarrow$ *"supportive"*, *"count"* $\rightarrow$ *"tally"*), and sentence structure are altered, the four foundational components of the prompt—persona, task definition, reasoning process, and output format—remain intact across all variations. The algorithm then proceeds to evaluate these candidates to determine if any of these seemingly innocuous rephrasings leads to a performance degradation, thereby revealing the prompt's local sharpness and identifying the adversarial worst-case $p^\star_{t,\text{adv}}$.

Outer robustness update. Using an optimizer oracle $\mathcal{O}$, we produce an improvement pool conditioned on the current and adversarial prompts with a semantic budget $\delta_t > 0$:

$$\mathcal{U}_{M_t}(p_t) := \text{Propose}\big(\mathcal{O},\, p_t,\, p^\star_{t,\text{adv}},\, \delta_t,\, M_t\big) = \{\, \tilde{p}^{(1)}, \ldots, \tilde{p}^{(M_t)} \,\}. \tag{11}$$

In essence, the Propose function is the core step where the optimizer oracle $\mathcal{O}$ suggests M_t potential improvements by analyzing both the current prompt p_t and its worst-performing neighbor $p^\star_{t,\text{adv}}$. The semantic budget δ_t restricts edits to preserve task intent and local coherence. We then select the next prompt by robust validation over the union of current and proposals:

$$p_{t+1} := \operatorname*{arg\,min}_{p' \in \{p_t\} \cup \mathcal{U}_{M_t}(p_t)} \max_{q \in \mathcal{C}_{\tilde{K}}(p')} \widehat{\mathcal{L}}(q; \mathcal{B}_t) \quad \text{with } \mathcal{C}_{\tilde{K}}(p') \sim \text{Sample}\big(\mathcal{G},\, p',\, \rho_t,\, \tilde{K}\big). \tag{12}$$

This greedy selection ensures a non-increasing estimate of the robust risk $\widehat{\mathcal{L}}_{\text{S}}(p_t; \rho_t)$ across iterations when the minimizer is attained. Equivalently, it drives down the minibatch estimate of the textual sharpness gap. **Intuition and relation to SAM:** SAM perturbs weights toward the ascent direction and then takes a descent step optimal under that perturbation. **TARE** mirrors this logic in discrete text: the inner sampling-based maximization uncovers the worst paraphrase within $B(p_t, \rho_t)$, and the outer selection moves p toward regions whose neighborhoods are flatter, co-optimizing task performance and robustness.

Schedules and acceptance. Typical schedules include: (i) *radius* annealing $\rho_{t+1} = \gamma\, \rho_t$ with $\gamma \in (0, 1]$ when progress stalls; (ii) *semantic budget* δ_t constrained to preserve task intent; and (iii) *budgets* $(K_t, M_t, \tilde{K})$ chosen to trade off compute and robustness. An iteration is accepted based on a robust validation criterion that evaluates the generalization performance of the worst-case neighbors. Specifically, the worst neighbors $p^\star_{t,\text{adv}}$ and $p^\star_{t+1,\text{adv}}$ are identified on the previous training batch $\mathcal{B}_{t-1}$ and the current training batch $\mathcal{B}_t$, respectively. Evaluating these worst-case neighbors on a separate validation set is crucial to ensure that any observed robustness is a generalizable property and not merely an artifact of a specific training minibatch. The update is therefore accepted only if the new worst neighbor demonstrates superior performance on $\mathcal{D}_{\text{valid}}$:

$$\widehat{\mathcal{L}}(p^\star_{t+1,\text{adv}}; \mathcal{D}_{\text{valid}}) \le \widehat{\mathcal{L}}(p^\star_{t,\text{adv}}; \mathcal{D}_{\text{valid}}) - \eta, \tag{13}$$

for tolerance $\eta \ge 0$; otherwise we increase search budgets or reduce ρ_t.

From TARE to ATARE. Uniform (isotropic) neighborhoods treat all prompt components equally, yet empirical sensitivity is heterogeneous across a prompt's core constraints, methodological guidance, and stylistic elements. This uniformity is inefficient; an ideal search strategy should apply cautious,

fine-grained perturbations to sensitive components where the landscape is steep, while exploring robust components more broadly where the landscape is flatter. To achieve this nuanced exploration, we introduce an adaptive, anisotropic variant that learns component-wise weights to shape the neighborhood accordingly, while jointly adapting the neighborhood size ρ_t when needed.

3.3 ATARE: Adaptive Textual Sharpness-Aware Evolving

The isotropic ball $B(p, \rho)$ may under/over-explore sensitive components of p. **ATARE** adapts an ellipsoidal neighborhood via a diagonal weight matrix $\boldsymbol{W}_{p_t} = \mathrm{diag}(\boldsymbol{w}_t)$, where $\boldsymbol{w}_t \in \mathbb{R}^m_{\geq 0}$ scores component-wise sensitivity.

Why anisotropy? Different parts of a prompt contribute unequally to its behavior: core constraints, methodological guidance, and stylistic elements exhibit heterogeneous sensitivity. For instance, in our counting-task example, the persona *"You are a helpful..."* is a stylistic element, the instruction to *"think step by step"* provides methodological guidance, and the rule to *"provide only the final numerical answer"* is a core constraint. An isotropic ball may overshoot sensitive tokens or underexplore robust ones. **ATARE** learns component-wise weights to shape an ellipsoidal neighborhood, applying finer, more constrained perturbations where the landscape is steep, while allowing for broader exploration in more stable regions. This accelerates convergence and reduces over-editing of fragile components.

Sensitivity estimation. Given $\mathcal{C}_{K_t}(p_t)$, define the per-component adversarial gain

$$s_{t,j} := \max_{p' \in \mathcal{C}_{K_t}(p_t) \,:\, \Delta_j(p_t, p') \neq 0} \big[\, \widehat{\mathcal{L}}(p'; \mathcal{B}_t) - \widehat{\mathcal{L}}(p_t; \mathcal{B}_t) \,\big]_+, \tag{14}$$

where $[u]_+ := \max\{u, 0\}$ and Δ_j extracts the change in component j. We update normalized weights with momentum

$$\tilde{w}_{t+1,j} = (1 - \alpha)\, \tilde{w}_{t,j} + \alpha\, s_{t,j}, \qquad w_{t+1,j} = \frac{\tilde{w}_{t+1,j}}{\epsilon + \sum_{k=1}^m \tilde{w}_{t+1,k}}, \;\; \alpha \in (0, 1], \; \epsilon > 0. \tag{15}$$

This normalization keeps $\sum_{j=1}^m w_{t+1,j} \approx 1$, and ϵ prevents division-by-zero while stabilizing early iterations. For our counting prompt, if a minor paraphrase of the output format rule consistently leads to a high loss, this component's weight $w_{t,j}$ would increase, marking it as highly sensitive.

Anisotropic distance and sampling. Using $\boldsymbol{W}_{p_t} = \mathrm{diag}(\boldsymbol{w}_t)$, define $d_{\text{ani}}(p_t, p'; \boldsymbol{W}_{p_t}) = \|\boldsymbol{W}_{p_t}\, \Delta(p_t, p')\|_2$ and the ellipsoid

$$B_{p_t}(p_t, \rho_t) = \big\{\, p' : d_{\text{ani}}(p_t, p'; \boldsymbol{W}_{p_t}) \leq \rho_t \,\big\}. \tag{16}$$

Here, a high weight $w_{t,j}$ for a sensitive component penalizes large edits, thus requiring smaller perturbations to stay within the neighborhood. Candidate generation is therefore biased *away* from sensitive components by sampling edit indices with a probability inversely proportional to their sensitivity:

$$\Pr\{\text{edit component } j\} \propto (1/w_{t,j})^\beta, \qquad \beta \geq 1. \tag{17}$$

This ensures robust components are explored broadly while fragile ones are perturbed cautiously. The inner/outer steps then mirror **TARE** with B replaced by B_{p_t}.

This anisotropic sampling process culminates in the generation of complete, holistic prompts where the degree of variation in each component reflects its learned sensitivity. For instance, in our counting-task example's candidates below, the low-sensitivity persona is creatively reimagined—from a *"helpful counting assistant"* to a *"cheerful counter"*. In contrast, the high-sensitivity constraint on the output format is meticulously preserved; although its phrasing is subtly varied (e.g., *"give just the final number"* or *"present only the final digit answer"*), the core directive to output only a number remains unchanged:

Table 1: **Main results across different backbone engines.** We report accuracy (%) and the relative improvement over TextGrad. The best and second-best results are highlighted with **bold** and underline, respectively.

Dataset	Model	BACKBONE: GPT-4o					BACKBONE: Claude 3.5 Sonnet				
		COT	TEXTGRAD	REVOLVE	TARE	ATARE	COT	TEXTGRAD	REVOLVE	TARE	ATARE
Object Counting	GPT-3.5	77.9$_{\downarrow 10.1}$	88.0	89.8$_{\uparrow 1.8}$	90.2$_{\uparrow 2.2}$	**91.0**$_{\uparrow 3.0}$	77.9$_{\downarrow 5.4}$	83.3	87.5$_{\uparrow 4.2}$	**90.4**$_{\uparrow 7.1}$	88.0$_{\uparrow 4.7}$
	Gemini 1.5 Flash 8B	82.0$_{\downarrow 1.3}$	83.3	83.5$_{\uparrow 0.2}$	84.7$_{\uparrow 1.4}$	**85.7**$_{\uparrow 2.4}$	82.0$_{\downarrow 6.5}$	88.5	90.0$_{\uparrow 1.5}$	**94.4**$_{\uparrow 5.9}$	91.0$_{\uparrow 2.5}$
	Gemini 1.5 Pro	94.0$_{\uparrow 0.0}$	94.0	94.0$_{\uparrow 0.0}$	**97.3**$_{\uparrow 3.3}$	**97.3**$_{\uparrow 3.3}$	94.0$_{\downarrow 3.0}$	97.0	97.7$_{\uparrow 0.7}$	98.0$_{\uparrow 1.0}$	**98.3**$_{\uparrow 1.3}$
	Llama 3.1 8B Instruct	86.0$_{\downarrow 2.6}$	88.6	88.2$_{\downarrow 0.4}$	91.0$_{\uparrow 2.4}$	**92.4**$_{\uparrow 3.8}$	86.0$_{\downarrow 3.5}$	89.5	88.0$_{\downarrow 1.5}$	90.6$_{\uparrow 1.1}$	**93.5**$_{\uparrow 4.0}$
	Llama 3 8B Instruct	80.0$_{\downarrow 5.8}$	85.8	86.8$_{\uparrow 1.0}$	88.7$_{\uparrow 2.9}$	**90.3**$_{\uparrow 4.5}$	80.0$_{\downarrow 2.0}$	82.0	84.3$_{\uparrow 2.3}$	89.5$_{\uparrow 7.5}$	**93.6**$_{\uparrow 11.6}$
Temporal Sequences	GPT-3.5	79.0$_{\downarrow 2.0}$	81.0	84.0$_{\uparrow 3.0}$	87.5$_{\uparrow 6.5}$	**88.0**$_{\uparrow 7.0}$	79.0$_{\downarrow 7.7}$	86.7	84.4$_{\downarrow 2.3}$	88.0$_{\uparrow 1.3}$	**89.0**$_{\uparrow 2.3}$
	Gemini 1.5 Flash 8B	92.0$_{\downarrow 0.5}$	92.5	93.0$_{\uparrow 0.5}$	94.3$_{\uparrow 1.8}$	**95.2**$_{\uparrow 2.7}$	92.0$_{\downarrow 2.0}$	94.0	94.7$_{\uparrow 0.7}$	95.3$_{\uparrow 1.3}$	**95.7**$_{\uparrow 1.7}$
	Gemini 1.5 Pro	96.0$_{\downarrow 1.7}$	97.7	97.7$_{\uparrow 0.0}$	**98.0**$_{\uparrow 0.3}$	**98.0**$_{\uparrow 0.3}$	96.0$_{\downarrow 1.0}$	97.0	98.0$_{\uparrow 1.0}$	98.5$_{\uparrow 1.5}$	**98.8**$_{\uparrow 1.8}$
	Llama 3.1 8B Instruct	86.0$_{\downarrow 2.3}$	88.3	88.3$_{\uparrow 0.0}$	90.0$_{\uparrow 1.7}$	**91.0**$_{\uparrow 2.7}$	86.0$_{\downarrow 7.0}$	93.0	89.6$_{\downarrow 3.4}$	93.7$_{\uparrow 0.7}$	**94.3**$_{\uparrow 1.3}$
	Llama 3 8B Instruct	84.0$_{\uparrow 0.0}$	84.0	84.5$_{\uparrow 0.5}$	85.0$_{\uparrow 1.0}$	**88.7**$_{\uparrow 4.7}$	84.0$_{\downarrow 7.5}$	91.5	89.2$_{\downarrow 2.3}$	**94.0**$_{\uparrow 2.5}$	**94.0**$_{\uparrow 2.5}$
Tracking Shuffled Objects	GPT-3.5	62.0$_{\downarrow 4.3}$	66.3	65.7$_{\downarrow 0.6}$	**72.0**$_{\uparrow 5.7}$	69.0$_{\uparrow 2.7}$	62.0$_{\downarrow 13.0}$	75.0	72.2$_{\downarrow 2.8}$	**77.0**$_{\uparrow 2.0}$	**77.0**$_{\uparrow 2.0}$
	Gemini 1.5 Flash 8B	82.0$_{\downarrow 1.0}$	83.0	82.5$_{\downarrow 0.5}$	88.6$_{\uparrow 5.6}$	**93.7**$_{\uparrow 10.7}$	82.0$_{\downarrow 5.3}$	87.3	89.8$_{\uparrow 2.5}$	91.3$_{\uparrow 4.0}$	**94.0**$_{\uparrow 6.7}$
	Gemini 1.5 Pro	99.0$_{\uparrow 0.0}$	99.0	99.0$_{\uparrow 0.0}$	99.0$_{\uparrow 0.0}$	99.0$_{\uparrow 0.0}$	99.0$_{\downarrow 0.0}$	99.0	99.0$_{\uparrow 0.0}$	99.0$_{\uparrow 0.0}$	99.0$_{\uparrow 0.0}$
	Llama 3.1 8B Instruct	82.0$_{\downarrow 4.3}$	86.3	83.7$_{\downarrow 2.6}$	90.0$_{\uparrow 3.7}$	**93.5**$_{\uparrow 7.2}$	82.0$_{\uparrow 0.8}$	81.2	79.2$_{\downarrow 2.0}$	91.2$_{\uparrow 10.0}$	**93.0**$_{\uparrow 11.8}$
	Llama 3 8B Instruct	50.0$_{\downarrow 5.5}$	55.5	52.7$_{\downarrow 2.8}$	57.5$_{\uparrow 2.0}$	**67.7**$_{\uparrow 12.2}$	50.0$_{\downarrow 14.5}$	64.5	66.8$_{\uparrow 2.3}$	72.3$_{\uparrow 7.8}$	**78.5**$_{\uparrow 14.0}$
GSM8K	GPT-3.5	72.9$_{\downarrow 8.0}$	80.9	82.1$_{\uparrow 1.2}$	**83.0**$_{\uparrow 2.1}$	82.3$_{\uparrow 1.4}$	72.9$_{\downarrow 8.2}$	81.1	80.1$_{\downarrow 1.0}$	**83.7**$_{\uparrow 2.6}$	**83.7**$_{\uparrow 2.6}$
	Gemini 1.5 Flash 8B	88.6$_{\downarrow 1.0}$	89.6	89.4$_{\downarrow 0.2}$	**90.1**$_{\uparrow 0.5}$	89.7$_{\uparrow 0.1}$	88.6$_{\downarrow 0.1}$	88.7	88.9$_{\uparrow 0.2}$	89.6$_{\uparrow 0.9}$	**89.7**$_{\uparrow 1.0}$
	Gemini 1.5 Pro	92.9$_{\downarrow 0.4}$	93.3	93.0$_{\downarrow 0.3}$	**95.5**$_{\uparrow 2.2}$	94.7$_{\uparrow 1.4}$	92.9$_{\downarrow 2.4}$	95.3	95.3$_{\uparrow 0.0}$	**96.1**$_{\uparrow 0.8}$	95.5$_{\uparrow 0.2}$
	Llama 3.1 8B Instruct	84.9$_{\uparrow 0.0}$	84.9	84.9$_{\uparrow 0.0}$	86.2$_{\uparrow 1.3}$	**86.4**$_{\uparrow 1.5}$	84.9$_{\downarrow 1.3}$	86.2	86.4$_{\uparrow 0.2}$	86.9$_{\uparrow 0.7}$	**87.7**$_{\uparrow 1.5}$
	Llama 3 8B Instruct	81.8$_{\uparrow 0.0}$	81.8	81.8$_{\uparrow 0.0}$	81.8$_{\uparrow 0.0}$	81.8$_{\uparrow 0.0}$	81.8$_{\uparrow 0.0}$	81.8	81.8$_{\uparrow 0.0}$	81.8$_{\uparrow 0.0}$	81.8$_{\uparrow 0.0}$

❶ **Candidate 1**: *You are a friendly counting helper. Your task is to count the objects. Work through the process step by step and then give just the final number.*

❷ **Candidate 2**: *You are an assistant designed to count things. First reason through the counting carefully, then respond with the single final numeric result.*

❸ **Candidate 3**: *As a cheerful counter, your role is to determine how many items there are. Go through your reasoning in order, but at the end present only the final digit answer.*

Adaptive radius schedule. To realize the adaptive design, we adjust the radius using validation outcomes: if robust validation improves for s consecutive iterations, set $\rho_{t+1} = \min\{\kappa\,\rho_t,\ \rho_{\max}\}$ with $\kappa > 1$; if an iteration is rejected, set $\rho_{t+1} = \max\{\gamma\,\rho_t,\ \rho_{\min}\}$ with $\gamma \in (0, 1)$. This expands exploration when stable and contracts it to preserve semantics.

3.4 Landscape-Aware Textual Optimizer

The Outer robustness update step relies on an optimizer oracle $\mathcal{O}$ to instantiate the Propose function. We operationalize this oracle with a potent, landscape-aware implementation, which we term the **Landscape-Aware Textual Optimizer (LATO)**. LATO realizes this step as a principled, landscape-guided update, formally defining the Propose function with its full set of inputs:

$$\text{Propose} := \left\{ \mathcal{O}_{\text{LATO}}^{(i)} \left(p_t, p_{t,\text{adv}}^\star, \widehat{\mathcal{L}}(p_t; \mathcal{B}_t), \widehat{\mathcal{L}}(p_{t,\text{adv}}^\star; \mathcal{B}_t), \text{Feedback}\big(\widehat{\mathcal{L}}(p_{t,\text{adv}}^\star; \mathcal{B}_t)\big), \delta_t \right) \right\}_{i=1}^{M_t}. \tag{18}$$

The update mechanism of LATO is designed to enhance robustness directly. Instead of merely correcting errors at its current position p_t, LATO analyzes the textual feedback, $\text{Feedback}\big(\widehat{\mathcal{L}}(p_{t,\text{adv}}^\star; \mathcal{B}_t)\big)$, which is derived from the point of highest local loss. It then applies this insight to refine p_t. This process preemptively addresses the sharpest vulnerabilities in the prompt's immediate semantic neighborhood. By learning from the failure modes of its neighbors, the optimizer guides p_t to become inherently more robust against similar types of semantic perturbations in the future.

This approach is powerful because LATO is, by construction, landscape-aware. By processing the two distinct prompt-loss pairs, $(p_t, \widehat{\mathcal{L}}(p_t; \mathcal{B}_t))$ and $(p_{t,\text{adv}}^\star, \widehat{\mathcal{L}}(p_{t,\text{adv}}^\star; \mathcal{B}_t))$, it directly perceives the local

sharpness of the semantic landscape. This awareness of the landscape's geometry—the steepness of the loss increase from p_t to $p^\star_{t,\text{adv}}$ and this empirically found worst-case direction—allows LATO to modulate its optimization strategy. It makes more informed decisions about both the direction and magnitude of the required edits, steering the prompt trajectory towards a demonstrably "flatter" and more stable semantic basin.

Operationally, LATO is instantiated using a powerful LLM as the core of the optimizer oracle $\mathcal{O}_{\text{LATO}}$. The update process can be expressed as the LLM generating a new prompt based on a structured meta-prompt, Π_{LATO}, which contains all the landscape information:

$$\tilde{p}^{(i)} := \text{LLM}\left(\Pi_{\text{LATO}}\left(p_t, p^\star_{t,\text{adv}}, \widehat{\mathcal{L}}(p_t;\mathcal{B}_t), \widehat{\mathcal{L}}(p^\star_{t,\text{adv}};\mathcal{B}_t), \text{Feedback}\big(\widehat{\mathcal{L}}(p^\star_{t,\text{adv}};\mathcal{B}_t)\big), \delta_t\right)\right). \tag{19}$$

Here, Π_{LATO} represents a meta-prompt template that synthesizes all the landscape-aware inputs from Equation (18) into a coherent, actionable instruction. The semantic budget δ_t acts as a crucial constraint, ensuring that the edits proposed by the LLM remain coherent and preserve the core intent of the task. The LLM then executes this instruction to generate an improved candidate prompt $\tilde{p}^{(i)}$, which forms an element of the proposal set Propose, effectively acting as a reasoning engine that performs a landscape-guided optimization step.

4 Experiments

We comprehensively evaluate our proposed methods, `TARE` and `ATARE`, through four axes: **Q1** (Superiority), **Q2** (Effectiveness), **Q3** (Resilience), and **Q4** (Sensitivity). The answers of **Q1-Q3** are illustrated in Sec. 4.2-Sec. 4.4, and sensitivity analysis (**Q4**) can be found in the Sec. D.

4.1 Experimental Setup

Tasks and Datasets. We evaluate our methods on four challenging reasoning tasks: three from the Big Bench Hard benchmark [36, 35]—**Object Counting**, **Temporal Sequences**, and **Tracking Shuffled Objects (Five Objects)**—and the **GSM8K** dataset [6]. For evaluation, our primary metric is Accuracy (Acc), measured by a strict string-based exact match on the final numerical answer [45]. Further details on datasets and implementation are provided in Appendix B.

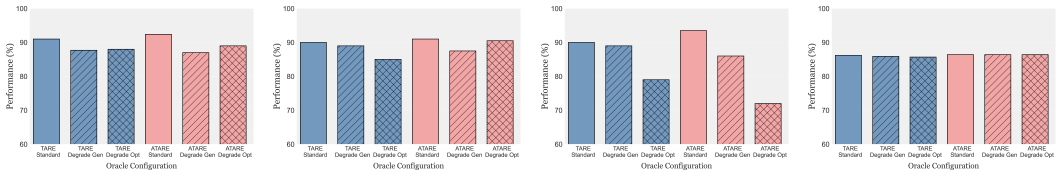
LLM Backends. Our experiments are conducted on a diverse set of five LLM backends: **GPT-3.5-turbo-0125**, **Gemini 1.5 Flash 8B**, **Gemini 1.5 Pro**, **Llama 3.1 8B Instruct**, and **Llama 3 8B Instruct**. To ensure a fair and controlled comparison, the optimizer and evaluator oracles for all methods are powered by two universal backbones: **GPT-4o** and **Claude 3.5 Sonnet**.

Counterparts. We compare our methods, `TARE` and `ATARE`, against three key baselines: Zero-shot Chain-of-Thought (CoT) [23, 41], TextGrad [45], and Revolve [46].

4.2 Superiority

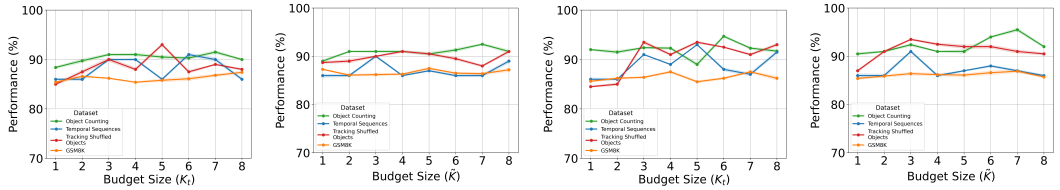
To answer **Q1**, we present the main prompt optimization results in Tab. 1. We summarize our key observations as follows (**Obs.**): **Obs. ❶** Our proposed methods, `TARE` and `ATARE`, consistently achieve state-of-the-art performance, outperforming all baselines, including TextGrad and Revolve, across nearly all evaluated tasks and LLM backbones. This significant performance gap stems from a fundamental difference in optimization objectives. While baselines are designed to maximize point-wise accuracy, our framework explicitly seeks robust solutions by optimizing for the worst-case performance within a semantic neighborhood, leading to more generalizable and effective prompts. **Obs. ❷** `ATARE` consistently demonstrates a performance advantage over `TARE` in most scenarios. This underscores the benefit of its adaptive, anisotropic search mechanism, which intelligently perturbs prompt components based on their learned sensitivity. This more nuanced search strategy consistently discovers superior solutions within the prompt landscape. **Obs. ❸** The framework's superiority shows **broad universality**, with substantial performance gains observed across diverse architectures, including proprietary models like GPT-3.5 and Gemini 1.5 Pro, as well as open-source models like the Llama 3 Instruct series. This confirms that our sharpness-aware approach is a model-agnostic and widely applicable solution for robust prompt optimization.

4.3 Effectiveness



(a) Object Counting (b) Temporal Sequences (c) Tracking Shuffled Objects (d) GSM8K

Figure 4: Resilience analysis of **TARE** and **ATARE** under oracle degradation, where the powerful GPT-4o oracles are replaced with a weaker Llama 3.1 8B model. For an in-depth analysis, please refer to Sec. 4.4.



(a) **TARE** vs. K_t (b) **TARE** vs. $\tilde{K}$ (c) **ATARE** vs. K_t (d) **ATARE** vs. $\tilde{K}$

Figure 5: Sensitivity analysis of **TARE** and **ATARE** with respect to the inner adversarial search budget (K_t) and the outer robust validation budget ($\tilde{K}$). For an in-depth analysis, please refer to Sec. D.

To address **Q2**, we conducted an ablation study on the key mechanisms of our framework using the Llama 3.1 8B Instruct model, with results shown in Figure 3. The chart clearly shows that the full **TARE** and **ATARE** frameworks perform best, and removing any of their core components leads to a significant drop in performance. Specifically, the **Inner Adversarial Search** is essential for finding challenging perturbations, the **LATO** optimizer uses landscape information to make smarter updates, and the **Robust Validation** criterion ensures that improvements generalize well. Finally, the consistent superiority of **ATARE** over **TARE** (detailed in Tab. 1) serves as a direct ablation for the **Anisotropic Search**, confirming the benefits of an adaptive strategy. When these components work together, the framework reaches its peak effectiveness, validating our design choices.

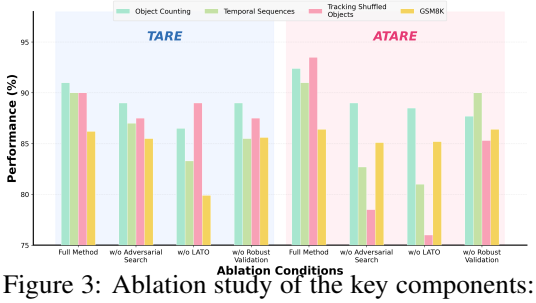


Figure 3: Ablation study of the key components: the Inner Adversarial Search, the LATO, and the Robust Validation. For an in-depth analysis, please refer to Sec. 4.3.

4.4 Resilience

To assess the resilience of our framework (**Q3**), we evaluate its performance under two challenging conditions: (i) degrading its powerful GPT-4o oracles by separately replacing the **Generator** and the **Optimizer** with a weaker Llama 3.1 8B model, and (ii) drastically reducing the search budgets for the inner adversarial search K_t and outer robust validation $\tilde{K}$. The results, illustrated in Figure 4 and Figure 5, demonstrate the framework's remarkable stability. As shown in Figure 4, even when individual core oracles are weakened, the performance degradation is remarkably graceful. With the exception of the Optimizer degradation on the Tracking Shuffled Objects task, the accuracy drop across all other conditions is consistently maintained within a 5% margin. Similarly, as shown in our sensitivity analysis (Figure 5), when the perturbation budgets ($K_t, \tilde{K}$) are reduced to a minimal value of 1 or 2, the framework's performance remains highly stable, exhibiting only a minor decrease relative to its performance at a budget of 3. This dual resilience proves that our sharpness-aware approach is robust to component degradation and computationally efficient, maintaining strong performance even under such challenging conditions.

5 Conclusion

Reliable prompt optimization begins with naming the right failure mode. Our work identifies and formalizes the overlooked problem of textual sharpness—the tendency of a prompt to collapse under semantically equivalent paraphrases—and reframes prompt optimization from chasing point-wise accuracy to seeking neighborhood-stable solutions. We instantiate this perspective with **TARE**, a

black-box, derivative-free procedure that adversarially probes a semantic neighborhood and selects candidates by their worst-case minibatch performance, and with **ATARE**, which learns anisotropic weights and adaptively schedules the neighborhood radius to balance exploration and fidelity. Both variants are API-only and gradient-free; the adaptive version adds only linear overhead in the number of semantic components while enforcing a fixed-margin decrease per accepted step. Across diverse tasks, **TARE** and **ATARE** consistently reduce the textual sharpness gap and preserve accuracy under paraphrasing, surpassing accuracy-only baselines while remaining computationally practical. Looking ahead, we see opportunities to extend textual sharpness-aware evolution to multi-turn and tool-augmented settings, to design task-aware semantic neighborhoods and edit families, and to deepen theory connecting textual sharpness with generalization in real-world LLM systems.

References

[1] Tom B. Brown, Benjamin Mann, Nick Ryder, et al. Language models are few-shot learners, 2020.

[2] Pratik Chaudhari et al. Entropy-sgd: Biasing gradient descent into wide valleys, 2017.

[3] Wenhu Chen et al. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks, 2023.

[4] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways, 2022.

[5] Hyung Won Chung et al. Scaling instruction-finetuned language models, 2022.

[6] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021.

[7] Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric P. Xing, and Zhiting Hu. Rlprompt: Optimizing discrete text prompts with reinforcement learning, 2022.

[8] Jiawei Du, Hanshu Yan, Jiashi Feng, Joey Tianyi Zhou, Liangli Zhen, Rick Siow Mong Goh, and Vincent Y. F. Tan. Efficient sharpness-aware minimization for improved training of neural networks, 2022.

[9] Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rockt"aschel. Promptbreeder: Self-referential self-improvement via prompt evolution, 2023.

[10] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization, 2021.

[11] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization, 2021.

[12] Aaron Grattafiori et al. The llama 3 herd of models, 2024.

[13] Qingyan Guo et al. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers, 2023.

[14] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021.

[15] Sepp Hochreiter and J"urgen Schmidhuber. Flat minima. *Neural Computation*, 9(1):1–42, 1997.

[16] Or Honovich et al. Instruction induction: From few examples to natural language task descriptions, 2022.

[17] Pavel Izmailov et al. Averaging weights leads to wider optima and better generalization, 2019.

[18] Albert Q. Jiang et al. Mistral 7b, 2023.

[19] Albert Q. Jiang et al. Mixtral of experts, 2024.

[20] Nitish Shirish Keskar et al. On large-batch training for deep learning: Generalization gap and sharp minima, 2017.

[21] Omar Khattab et al. Dspy: Compiling declarative language model calls into self-improving pipelines, 2023.

[22] Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. Dspy: Compiling declarative language model calls into self-improving pipelines, 2023.

[23] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners, 2023.

[24] Jungmin Kwon, Jeongseop Kim, Hyunseo Park, and In Kwon Choi. Asam: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5905–5914. PMLR, 18–24 Jul 2021.

[25] H.Y. Leong and Y. Wu. Why should next-gen llm multi-agent systems move beyond fixed architectures to dynamic, input-driven graphs? *SSRN Electronic Journal*, 2024.

[26] Tao Li et al. Friendly sharpness-aware minimization, 2024.

[27] Yong Liu et al. Towards efficient and scalable sharpness-aware minimization, 2022.

[28] Sejoon Oh, Yiqiao Jin, Megha Sharma, Donghyun Kim, Eric Ma, Gaurav Verma, and Srijan Kumar. Uniguard: Towards universal safety guardrails for jailbreak attacks on multimodal large language models. *arXiv:2411.01703*, 2024.

[29] OpenAI. Gpt-4 technical report, 2023.

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

[31] Reid Pryzant et al. Automatic prompt optimization with "gradient descent" and beam search, 2023.

[32] Rafael Rafailov et al. Direct preference optimization: Your language model is secretly a reward model, 2024.

[33] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof qa benchmark, 2023.

[34] Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts, 2020.

[35] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, Agnieszka Kluska, Aitor Lewkowycz, Akshat Agarwal, Alethea Power, Alex Ray, Alex Warstadt, Alexander W. Kocurek, Ali Safaya, Ali Tazarv, Alice Xiang, Alicia Parrish, Allen Nie, Aman Hussain, Amanda Askell, Amanda Dsouza, Ambrose Slone, Ameet Rahane, Anantharaman S. Iyer, Anders Andreassen, Andrea Madotto, Andrea Santilli, Andreas Stuhlmüller, Andrew Dai, Andrew La, Andrew Lampinen, Andy Zou, Angela Jiang, Angelica Chen, Anh Vuong, Animesh Gupta, Anna Gottardi, Antonio Norelli, Anu Venkatesh, Arash Gholamidavoodi, Arfa Tabassum, Arul Menezes, Arun Kirubarajan, Asher Mullokandov, Ashish Sabharwal, Austin Herrick, Avia Efrat, Aykut Erdem, Ayla Karakaş, B. Ryan Roberts, Bao Sheng Loe, Barret Zoph, Bartłomiej Bojanowski, Batuhan Özyurt, Behnam Hedayatnia, Behnam Neyshabur, Benjamin Inden, Benno Stein, Berk Ekmekci, Bill Yuchen Lin, Blake Howald, Bryan Orinion, Cameron Diao, Cameron Dour, Catherine Stinson, Cedrick Argueta, César Ferri Ramírez, Chandan Singh, Charles Rathkopf, Chenlin Meng, Chitta Baral, Chiyu Wu, Chris Callison-Burch, Chris Waites, Christian Voigt, Christopher D. Manning, Christopher Potts, Cindy Ramirez, Clara E. Rivera, Clemencia Siro, Colin Raffel, Courtney Ashcraft, Cristina Garbacea, Damien Sileo, Dan Garrette, Dan Hendrycks, Dan Kilman, Dan Roth, Daniel Freeman, Daniel Khashabi, Daniel Levy, Daniel Moseguí González, Danielle Perszyk, Danny Hernandez, Danqi Chen, Daphne Ippolito, Dar Gilboa, David Dohan, David Drakard, David Jurgens, Debajyoti Datta, Deep Ganguli, Denis Emelin, Denis Kleyko, Deniz Yuret, Derek Chen, Derek Tam, Dieuwke Hupkes, Diganta Misra, Dilyar Buzan, Dimitri Coelho Mollo, Diyi Yang, Dong-Ho Lee, Dylan Schrader, Ekaterina Shutova, Ekin Dogus Cubuk, Elad Segal, Eleanor Hagerman, Elizabeth Barnes, Elizabeth Donoway, Ellie Pavlick, Emanuele Rodola, Emma Lam, Eric Chu, Eric Tang, Erkut Erdem, Ernie Chang, Ethan A. Chi, Ethan Dyer, Ethan Jerzak, Ethan Kim, Eunice Engefu Manyasi, Evgenii Zheltonozhskii, Fanyue Xia, Fatemeh Siar, Fernando Martínez-Plumed, Francesca Happé, Francois Chollet, Frieda Rong, Gaurav Mishra, Genta Indra Winata, Gerard de Melo, Germán Kruszewski, Giambattista Parascandolo, Giorgio Mariani, Gloria Wang, Gonzalo Jaimovitch-López, Gregor Betz, Guy Gur-Ari, Hana Galijasevic, Hannah Kim, Hannah Rashkin, Hannaneh Hajishirzi, Harsh Mehta, Hayden Bogar, Henry Shevlin, Hinrich Schütze, Hiromu Yakura, Hongming Zhang, Hugh Mee Wong, Ian Ng, Isaac Noble, Jaap Jumelet, Jack Geissinger, Jackson Kernion, Jacob Hilton, Jaehoon Lee, Jaime Fernández Fisac, James B. Simon, James Koppel, James Zheng, James Zou, Jan Kocoń, Jana Thompson, Janelle Wingfield, Jared Kaplan, Jarema Radom, Jascha Sohl-Dickstein, Jason Phang, Jason Wei, Jason Yosinski, Jekaterina Novikova, Jelle Bosscher, Jennifer Marsh, Jeremy Kim, Jeroen Taal, Jesse Engel, Jesujoba Alabi, Jiacheng Xu, Jiaming Song, Jillian Tang, Joan Waweru, John Burden, John Miller, John U. Balis, Jonathan Batchelder, Jonathan Berant, Jörg Frohberg, Jos Rozen, Jose Hernandez-Orallo, Joseph Boudeman, Joseph Guerr, Joseph Jones, Joshua B. Tenenbaum, Joshua S. Rule, Joyce Chua, Kamil Kanclerz, Karen Livescu, Karl Krauth, Karthik Gopalakrishnan, Katerina Ignatyeva, Katja Markert, Kaustubh D. Dhole, Kevin Gimpel, Kevin Omondi, Kory Mathewson, Kristen Chiafullo, Ksenia Shkaruta, Kumar Shridhar, Kyle McDonell, Kyle Richardson, Laria Reynolds, Leo Gao, Li Zhang, Liam Dugan, Lianhui Qin, Lidia Contreras-Ochando, Louis-Philippe Morency, Luca Moschella, Lucas Lam, Lucy Noble, Ludwig Schmidt, Luheng He, Luis Oliveros Colón, Luke Metz, Lütfi Kerem Şenel, Maarten Bosma, Maarten Sap, Maartje ter Hoeve, Maheen Farooqi, Manaal Faruqui, Mantas Mazeika, Marco Baturan, Marco Marelli, Marco Maru, Maria Jose Ramírez Quintana, Marie Tolkiehn, Mario Giulianelli, Martha Lewis, Martin Potthast, Matthew L. Leavitt, Matthias Hagen, Mátyás Schubert, Medina Orduna Baitemirova, Melody Arnaud, Melvin McElrath, Michael A. Yee, Michael Cohen, Michael Gu, Michael Ivanitskiy, Michael Starritt, Michael Strube, Michał Swędrowski, Michele Bevilacqua, Michihiro Yasunaga, Mihir Kale, Mike Cain, Mimee Xu, Mirac Suzgun, Mitch Walker, Mo Tiwari, Mohit Bansal, Moin Aminnaseri, Mor Geva, Mozhdeh Gheini, Mukund Varma T, Nanyun Peng, Nathan A. Chi, Nayeon Lee, Neta Gur-Ari Krakover, Nicholas Cameron, Nicholas Roberts, Nick Doiron, Nicole Martinez, Nikita Nangia, Niklas Deckers, Niklas Muennighoff, Nitish Shirish Keskar, Niveditha S. Iyer, Noah Constant, Noah Fiedel, Nuan Wen, Oliver Zhang, Omar Agha, Omar Elbaghdadi, Omer Levy, Owain Evans, Pablo Antonio Moreno Casares, Parth Doshi, Pascale Fung, Paul Pu Liang, Paul Vicol, Pegah Alipoormolabashi, Peiyuan Liao, Percy Liang, Peter Chang, Peter Eckersley, Phu Mon Htut, Pinyu Hwang, Piotr Miłkowski, Piyush Patil, Pouya Pezeshkpour, Priti Oli, Qiaozhu Mei, Qing Lyu, Qinlang Chen, Rabin Banjade, Rachel Etta Rudolph, Raefer Gabriel, Rahel Habacker, Ramon Risco, Raphaël Millière, Rhythm Garg, Richard Barnes, Rif A. Saurous, Riku Arakawa, Robbe Raymaekers, Robert Frank, Rohan

Sikand, Roman Novak, Roman Sitelew, Ronan LeBras, Rosanne Liu, Rowan Jacobs, Rui Zhang, Ruslan Salakhutdinov, Ryan Chi, Ryan Lee, Ryan Stovall, Ryan Teehan, Rylan Yang, Sahib Singh, Saif M. Mohammad, Sajant Anand, Sam Dillavou, Sam Shleifer, Sam Wiseman, Samuel Gruetter, Samuel R. Bowman, Samuel S. Schoenholz, Sanghyun Han, Sanjeev Kwatra, Sarah A. Rous, Sarik Ghazarian, Sayan Ghosh, Sean Casey, Sebastian Bischoff, Sebastian Gehrmann, Sebastian Schuster, Sepideh Sadeghi, Shadi Hamdan, Sharon Zhou, Shashank Srivastava, Sherry Shi, Shikhar Singh, Shima Asaadi, Shixiang Shane Gu, Shubh Pachchigar, Shubham Toshniwal, Shyam Upadhyay, Shyamolima, Debnath, Siamak Shakeri, Simon Thormeyer, Simone Melzi, Siva Reddy, Sneha Priscilla Makini, Soo-Hwan Lee, Spencer Torene, Sriharsha Hatwar, Stanislas Dehaene, Stefan Divic, Stefano Ermon, Stella Biderman, Stephanie Lin, Stephen Prasad, Steven T. Piantadosi, Stuart M. Shieber, Summer Misherghi, Svetlana Kiritchenko, Swaroop Mishra, Tal Linzen, Tal Schuster, Tao Li, Tao Yu, Tariq Ali, Tatsu Hashimoto, Te-Lin Wu, Théo Desbordes, Theodore Rothschild, Thomas Phan, Tianle Wang, Tiberius Nkinyili, Timo Schick, Timofei Kornev, Titus Tunduny, Tobias Gerstenberg, Trenton Chang, Trishala Neeraj, Tushar Khot, Tyler Shultz, Uri Shaham, Vedant Misra, Vera Demberg, Victoria Nyamai, Vikas Raunak, Vinay Ramasesh, Vinay Uday Prabhu, Vishakh Padmakumar, Vivek Srikumar, William Fedus, William Saunders, William Zhang, Wout Vossen, Xiang Ren, Xiaoyu Tong, Xinran Zhao, Xinyi Wu, Xudong Shen, Yadollah Yaghoobzadeh, Yair Lakretz, Yangqiu Song, Yasaman Bahri, Yejin Choi, Yichi Yang, Yiding Hao, Yifu Chen, Yonatan Belinkov, Yu Hou, Yufang Hou, Yuntao Bai, Zachary Seid, Zhuoye Zhao, Zijian Wang, Zijie J. Wang, Zirui Wang, and Ziyi Wu. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models, 2023.

[36] Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. Challenging big-bench tasks and whether chain-of-thought can solve them, 2022.

[37] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

[38] Xuezhi Wang et al. Self-consistency improves chain of thought reasoning in language models, 2023.

[39] Yizhong Wang et al. Self-instruct: Aligning language models with self-generated instructions, 2023.

[40] Jason Wei et al. Chain-of-thought prompting elicits reasoning in large language models, 2023.

[41] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023.

[42] Dongxian Wu et al. Adversarial weight perturbation helps robust generalization, 2020.

[43] Shunyu Yao et al. React: Synergizing reasoning and acting in language models, 2023.

[44] Shunyu Yao et al. Tree of thoughts: Deliberate problem solving with large language models, 2023.

[45] Mert Yuksekgonul et al. Textgrad: Automatic "differentiation" via text, 2024.

[46] Peiyan Zhang et al. Revolve: Optimizing ai systems by tracking response evolution in textual optimization, 2025.

[47] Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers, 2022.

[48] Juntang Zhuang, Boqing Gong, Liangzhe Yuan, Yin Cui, Hartwig Adam, Nicha Dvornek, Sekhar Tatikonda, James Duncan, and Ting Liu. Surrogate gap minimization improves sharpness-aware training, 2022.

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 abstract and introduction clearly state the paper's contributions and scope.

   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 discussed limitations and future work in the Conclusion section.

   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: [NA]

Justification: The paper does not include formal theoretical results with proofs.

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: Experiment settings and implementation details are described in Section 4 and Appendix.

   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: All datasets used are public with proper citations. The code is available at `https://anonymous.4open.science/r/ATARE_TARE/`.

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: Experimental settings are described in Section 4 and Appendix.

   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 the main accuracy scores obtained from multiple experimental runs.

   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: [No]

   Justification: Our experiments were conducted by querying LLM APIs. Therefore, local compute resource details are not applicable.

   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 this paper conforms 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: [NA]

    Justification: This is foundational research on prompt optimization with no direct societal impact.

    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. This work focuses on prompt optimization algorithms without releasing high-risk models or datasets.

    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: All datasets and base models are properly cited with original papers. Public datasets are used with appropriate citations.

    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: [NA]

    Justification: The paper does not release new assets.

    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: [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.
    - 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. LLMs were used only for writing assistance.

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 Related work.

A.1 Large Language Models

Large Language Models (LLMs) have rapidly advanced in scale and capability, from early few-shot systems such as GPT-3 [1] and GPT-4 [29] to general-purpose foundation and open models including PaLM [4], Llama 2 [37], Llama 3 [12], Mistral 7B [18], and Mixtral [19]. Instruction finetuning and alignment further steer model behavior toward user intents [5, 30, 32, 25]. Our problem setting assumes black-box (API-only) access to an LLM (and optionally an evaluator), and focuses on optimizing prompts rather than modifying model parameters, making our approach complementary to parameter-finetuning and alignment.

A.2 Prompt Optimization

Prompt engineering has evolved from manual design to automated optimization. Early automated approaches include gradient-free token editing and trigger search (AutoPrompt) [34], reinforcement-learning-based optimization (RLPrompt) [7], and search-based schemes such as APO [31]. Recent work connects evolutionary algorithms with LLMs or leverages LLMs as optimizers to iteratively propose and select candidates [13, 47, 9, 28]; programmatic frameworks like DSPy compile declarative pipelines into self-improving prompt graphs [21]. In parallel, instruction induction and self-instruction curate high-coverage supervision for prompt/task design [16, 39]; and reasoning-oriented prompting (CoT, self-consistency, ToT, ReAct, PoT) improves average-case reasoning performance [40, 38, 44, 43, 3]. Nevertheless, most of these methods primarily optimize point-wise metrics on static validation sets and seldom enforce robustness under semantically preserving paraphrases. Our work explicitly targets this failure mode by formalizing textual sharpness in semantic prompt space and optimizing worst-case performance over a neighborhood.

A.3 Sharpness-Aware Minimization

Generalization and robustness in deep learning have been linked to the geometry of the loss landscape, where flat minima often correlate with better generalization [15, 20]. Sharpness-Aware Minimization (SAM) biases solutions toward flatter regions by minimizing loss under worst-case local perturbations [10, 11]. Subsequent variants extend this idea with scale-invariant updates (ASAM) [24], efficiency-focused or surrogate-gap formulations [8, 48, 27], and friendly/trustworthy adaptations [26]. Related techniques encourage wide valleys via entropy or averaging [2, 17] and adversarial weight perturbations [42]. Unlike these methods that operate in continuous parameter space with gradient access, we instantiate an *analogous* principle in discrete text: we define and measure sharpness over a semantic neighborhood of prompts and develop a black-box, derivative-free algorithm that co-optimizes task performance and local flatness.

B Experimental Details

B.1 Dataset Details

To assess the effectiveness of our framework, we conduct experiments on four diverse and challenging reasoning tasks. Consistent with prior work [45], the evaluation metric is string-based exact match accuracy. A detailed description is provided below:

- **BIG-Bench Hard Tasks [36, 35].** BIG-Bench Hard is a suite of 23 challenging tasks from the BIG-Bench benchmark, specifically selected because prior language models had failed to outperform the average human-rater on them. These tasks often require multi-step reasoning, making them suitable for evaluating advanced model capabilities. From this benchmark, we select three distinct tasks:
  - **Object Counting:** Given a list of items and their quantities, the task is to determine the total number of items belonging to a specific category.
  - **Temporal Sequences:** Given a series of events and activities a person has completed, the task is to determine a time they might have been free for another activity.
  - **Tracking Shuffled Objects (Five Objects):** Given the initial positions of several objects and a series of pairwise swaps, the task is to determine the final position of each object.

For the **Object Counting** task, we adopt the data split of 50 training, 100 validation, and 100 test samples from TextGrad [45]. For **Temporal Sequences** and **Tracking Shuffled Objects (Five Objects)**, we follow an identical splitting methodology.

- **GSM8K [6].** To further assess mathematical reasoning, we use this widely-used benchmark consisting of grade-school math word problems that require multi-step reasoning. For this task, we adopt the dataset splits provided by DSPy [22], which include 200 training, 300 validation, and 1319 test samples.

B.2 Counterpart Details

This section provides an overview of the baseline approaches employed in our study for comparison.

- **Zero-shot Chain-of-Thought (CoT) [23, 41].** A foundational baseline that elicits multi-step reasoning by prompting the model with instructions like "Think step-by-step" before it provides a final answer.
- **TextGrad [45].** A first-order optimization method that treats natural language feedback from an evaluator LLM as a "textual gradient" to iteratively refine variables based on immediate, local feedback.
- **Revolve [46].** An optimization method that extends first-order techniques by tracking how system responses evolve across iterations. By incorporating this historical context, Revolve aims for more stable optimization and to escape the local optima that can trap methods relying on single-step feedback.

B.3 Implementation Details

Our experiments are conducted on a diverse set of five LLM backends: **GPT-3.5-turbo-0125**, **Gemini 1.5 Flash 8B**, **Gemini 1.5 Pro**, **Llama 3.1 8B Instruct**, and **Llama 3 8B Instruct**. To ensure a fair and controlled comparison, our setup relies on a universal backbone engine for three key roles: generators, optimizers, and evaluators. For these backbone roles, we employ two powerful models: **GPT-4o** and **Claude 3.5 Sonnet**.

For all iterative methods, we follow the experimental setup in Revolve [46], using a batch size of 3 across 12 optimization iterations, processing a total of 36 training examples. For our sharpness-aware methods, we set the key search budgets to $K_t = 3$ (inner adversarial search), $M_t = 1$ (proposal pool size), and $\tilde{K} = 3$ (outer robust validation). For LLM generation, our configuration largely mirrors that of Revolve [46]: we allow a maximum of 2000 new tokens and use a top-p value of 0.99. To ensure maximum reproducibility, we set the decoding temperature to 0 for all models.

C Complexity and Practical Notes

Beyond asymptotic cost, the design rationale is that the inner loop diagnoses textual sharpness while the outer loop enforces progress on the robust objective; radius annealing preserves semantics, and acceptance tests prevent regressions. The framework is modular: $\mathcal{G}$ (candidate generators), $\mathcal{O}$ (optimizers), and evaluators $\mathcal{E}$ can be swapped without changing the principle. The choice of ρ_t can be guided by paraphrase detection or embedding-similarity thresholds to maintain semantic fidelity.

Each iteration evaluates K_t adversarial and $(M_t + 1)\tilde{K}$ robust losses on a minibatch, totaling $O\big((K_t + (M_t + 1)\tilde{K})\,|\mathcal{B}_t|\big)$ calls to $\mathcal{M}$ and $\mathcal{E}$. `ATARE` adds an $O(m)$ overhead for weight updates and negligible cost for adaptive radius updates. In practice: (i) reuse evaluations across inner/outer loops; (ii) maintain a replay buffer of high-loss neighbors to warm-start future inner maximizations; and (iii) set ρ_t to preserve semantic intent while revealing sharp regions.

D Sensitivity

To address **Q4**, we perform a sensitivity analysis on the two key search budget hyperparameters of our framework: the inner adversarial search budget K_t and the outer robust validation budget $\tilde{K}$. As illustrated in Figure 5, we evaluate the performance of `TARE` and `ATARE` on the Llama 3.1 8B model,

using GPT-4o as the backbone engine, by systematically varying one budget within the range of [1, 8] while keeping the other fixed at a moderate value of 3. The results indicate that our framework is not highly sensitive to the precise choice of these parameters. Performance generally improves as the budgets increase from 1 to 3 and then stabilizes, exhibiting only minor fluctuations for values up to 8. This demonstrates that our methods can achieve strong, robust performance without requiring extensive hyperparameter tuning, as a relatively small budget (e.g., $K_t = \tilde{K} = 3$) is sufficient to capture the benefits of our sharpness-aware approach.

E Solution Optimization

While the core of our work focuses on prompt optimization, the principles of our framework can be extended to other complex textual domains. A critical application is **Solution Optimization**, which involves the iterative refinement of multi-step reasoning chains. Unlike prompts, solutions are highly structured and logically interlocked, presenting unique challenges that require a tailored approach.

E.1 Applying ATARE to Fragile Reasoning Chains

A key characteristic of a solution is its inherent fragility. A solution is not merely a collection of sentences; it is a delicate, logically-interlocked chain of reasoning where each step builds upon the previous one. A minor alteration to an early, correct step can invalidate the entire downstream logic. This fragility renders isotropic perturbation methods like **TARE** ineffective, as uniform paraphrasing would inevitably disrupt the "correct reasoning backbone," creating a noisy and uninformative loss landscape.

This very structure—a stable, correct reasoning backbone combined with a specific, identifiable flaw—makes the problem of solution optimization an ideal application for the **ATARE** framework. **ATARE** is fundamentally designed to handle textual components with varying degrees of sensitivity. The logical chain of a solution presents a natural, clear-cut case of anisotropic sensitivity, making **ATARE** not just a possible tool, but a perfectly suited one.

Our approach applies the core components of the **ATARE** lifecycle to this problem as follows:

1. Semantic Sensitivity Analysis and Anisotropic Neighborhood Definition. The first step in the **ATARE** lifecycle is sensitivity analysis. We implement this by performing a comprehensive semantic diagnosis of the entire incorrect solution to identify the single, core "cognitive trap." This diagnosis effectively partitions the solution into two distinct regions of sensitivity:

- **Low-Sensitivity Region:** The identified core logical flaw itself. This part is considered the primary target for perturbation. The rationale is that a single type of logical error can manifest in many different, deceptive forms. All solutions that commit the same conceptual error, regardless of phrasing, are considered to be within the same **anisotropic semantic neighborhood**.
- **High-Sensitivity Region:** The entire chain of correct reasoning that precedes the flaw. This logical backbone is treated as immutable to maintain the solution's structural integrity.

2. Inner Adversarial Search. With the sensitivity regions defined, we conduct an inner adversarial search within this highly constrained neighborhood. To formalize this, let s_t be the original incorrect solution at a given iteration t. We represent s_t as a composition of two parts: its high-sensitivity correct backbone, denoted $s_{t,\text{correct}}$, and its low-sensitivity flaw, denoted $s_{t,\text{flaw}}$. The set of candidate solutions, $\mathcal{C}_{K_t}(s_t)$, is then generated by keeping the backbone fixed while using a generator to perturb only the flaw component. This process creates a set of K_t candidates, defined as:

$$\mathcal{C}_{K_t}(s_t) := \left\{ s_{t,\text{correct}} \oplus \mathcal{G}^{(i)}(s_{t,\text{flaw}}) \right\}_{i=1}^{K_t}, \tag{20}$$

where the correct backbone $s_{t,\text{correct}}$ is held fixed, $\mathcal{G}_{\text{flaw}}$ is the generator responsible for creating variations of the flaw, the superscript (i) indexes each of the K_t unique generation events, and the symbol $\oplus$ denotes the composition of the text segments. This process explores the defined semantic neighborhood to find the variations that perform the worst on the task.

3. Outer Robustness Update. From the generated set of flaw variations, we identify the "worst-case" neighbor $s^*_{t,adv}$. The textual feedback derived from critiquing this worst-case scenario is then

Table 2: **Results of solution optimization.** We report accuracy (%) and the relative improvement over Textgrad. The best and second-best results are highlighted with **bold** and underline, respectively.

Dataset	Model	COT	TEXTGRAD	REVOLVE	ATARE
GPQA	GPT-4o	$48.5_{\downarrow 4.0}$	<u>52.5</u>	$49.5_{\downarrow 3.0}$	$\mathbf{53.0}_{\uparrow \mathbf{0.5}}$
	Llama 3.1 8B Instruct	$27.8_{\downarrow 3.0}$	<u>30.8</u>	$\underline{30.8}_{\uparrow 0.0}$	$\mathbf{33.8}_{\uparrow \mathbf{3.0}}$
	Qwen 2.5 7B Instruct	$35.9_{\downarrow 1.5}$	<u>37.4</u>	$\underline{37.4}_{\uparrow 0.0}$	$\mathbf{39.9}_{\uparrow \mathbf{2.5}}$
MMLU (College Physics)	GPT-4o	$91.0_{\downarrow 2.5}$	93.5	$\underline{94.1}_{\uparrow 0.6}$	$\mathbf{96.1}_{\uparrow \mathbf{2.6}}$
	Llama 3.1 8B Instruct	$69.6_{\uparrow 0.0}$	69.6	$\underline{70.6}_{\uparrow 1.0}$	$\mathbf{72.5}_{\uparrow \mathbf{2.9}}$
	Qwen 2.5 7B Instruct	$\underline{78.4}_{\uparrow 0.0}$	<u>78.4</u>	$\underline{78.4}_{\uparrow 0.0}$	$\mathbf{79.4}_{\uparrow \mathbf{1.0}}$

used to update the original solution s_t. By learning from the most challenging manifestation of its own core error, the solution is guided to patch this specific cognitive vulnerability. This directly implements the outer robustness update step of **ATARE**, optimizing for worst-case performance within the semantic neighborhood to guide the solution toward a flatter, more robust basin in the semantic landscape.

E.2 Experiments

Tasks and Datasets. We evaluate our solution optimization approach on two challenging benchmarks where model performance has not yet saturated.

- **GPQA** [33]: The Google-proof Question Answering benchmark consists of expert-level multiple-choice questions in physics, biology, and chemistry. Its difficulty is highlighted by the performance gap between experts (81% accuracy) and skilled non-experts (22%).
- **MMLU** [14]: We use the challenging **College Physics** subset from the Massive Multitask Language Understanding benchmark, which is designed to measure human-level performance.

We follow the experimental setup of Revolve [46] for iterative methods: we perform three iterations of optimization for each question and determine the final answer by majority voting. Consistent with prior work [45], the evaluation metric is string-based exact match accuracy.

LLM Backends and Counterparts. We apply all methods on three distinct LLMs: **GPT-4o**, **Llama 3.1 8B Instruct**, and **Qwen 2.5 7B Instruct**. We compare our **ATARE**-based method against three primary baselines: **Zero-shot Chain-of-Thought (CoT)** [23], **TextGrad** [45], and **Revolve** [46].

Results and Analysis. The performance of our method against the baselines is presented in Table 2. As shown, our **ATARE**-based approach consistently outperforms all baselines—CoT, TextGrad, and Revolve—across both datasets and all evaluated LLM backends. This strong and universal improvement validates our central hypothesis: for fragile, logically-interlocked reasoning chains, an anisotropic optimization strategy is superior. While first-order methods like TextGrad can sometimes struggle with the delicate structure of solutions, and even advanced methods like Revolve may not always escape local optima, our approach demonstrates a more robust path to improvement. By precisely targeting only the low-sensitivity "flaw region" for perturbation while preserving the high-sensitivity correct reasoning, our approach provides a stable and effective optimization signal, consistently guiding the solution toward a more correct state.

F Prompt Details

This section provides the detailed architecture of the core prompts that power our optimization framework. We present the prompts for the main components of our framework—the perturbation generators for **TARE** and **ATARE**, and the LATO—as well as the prompts used for the Solution Optimization task.

F.1 TARE Perturbation Generator Prompt

This prompt directs the TARE perturbation generator to conduct the isotropic neighborhood search required by our framework. It instructs a powerful LLM to create a set of minimally-altered, semantically-equivalent variations of a given text. The key to this process is the explicit goal of finding weaknesses; the prompt directs the generator to explore potentially worse-performing variations. This serves as the inner adversarial search, designed to identify sharp cliffs in the semantic landscape where the system's performance is brittle.

TARE Perturbation Generator Prompt

You are an expert in semantics and creative writing. Your task is to generate `{k}` slightly different versions of the following text.
These perturbed versions must adhere to these rules:

1. **Maintain Core Intent:** The core intent and theme of the original text must be preserved.
2. **Small Degree of Perturbation:** The changes should be minor. For example, you can replace a few non-essential words, make small adjustments to sentence structure, or add/remove a few descriptive words.
3. **Preserve Factual Correctness:** Do not introduce irrelevant information or factual errors.
4. **Explore Vulnerabilities:** The goal is to explore closely related, but potentially worse-performing, variations of the text.

Input Text ``system_prompt_text''

Output Format The output should be a Python-style list of strings, with each string being one perturbed version.

Now, provide the `{k}` perturbed versions for the original text.

F.2 ATARE Perturbation Prompt

The ATARE prompt architecture implements our framework's anisotropic search. It operates as a two-step "analyze-then-generate" chain. The first prompt performs Semantic Sensitivity Estimation, directing an analyst LLM to decompose a prompt into three tiers of sensitivity (Constraint, Method, Style). The second prompt then performs Anisotropic Perturbation, using this analysis to guide a generator LLM in applying targeted, differentiated edits to each tier.

Step 1: Sensitivity Analysis Prompt

You are a specialized Prompt Architecture Analyst. Your task is to analyze a system prompt and decompose it into a three-tier hierarchy of components based on their sensitivity.

Definition of Tiers

- **Tier 1 (Constraint Layer - High Sensitivity):** Non-negotiable rules that define success or failure. Changing these will likely break the prompt's core function or format. (e.g., format rules, core task definition, absolute prohibitions).
- **Tier 2 (Method Layer - Medium Sensitivity):** Guidelines on "how" to perform the task. Changing these affects the quality and reasoning path, but not task completion itself. (e.g., "think step by step", process instructions).
- **Tier 3 (Style Layer - Low Sensitivity):** Persona, tone, and other stylistic elements. Changing these affects the prompt's personality, not its logic. (e.g., 'You are a helpful assistant', politeness).

Input Prompt ``system_prompt_text''

Output Format Your output MUST be a single, valid JSON object with three keys: ``constraint_layer'', ``method_layer'', and ``style_layer''. Each key must have a list of strings as its value.

Now, provide the three-tier JSON analysis for the original prompt.

Step 2: `ATARE` Perturbation Generator Prompt

You are an expert in semantics and creative writing. The goal is to explore closely related, but potentially worse-performing, variations of the text.

Inputs

1. **Original Prompt:** ‘‘`system_prompt_text`’’
2. **Three-Tier Sensitivity Analysis:** `{analysis_text}`

YOUR TASK & RULES Your generated versions MUST adhere to these rules:

1. **Maintain Core Intent (Global Constraint):** All perturbed versions MUST maintain the core intent of the original prompt. The goal is to create semantically close variations to find weaknesses, not to write a new prompt.
2. **Targeted & Differentiated Perturbation:** Your changes should be targeted, and their degree must be based on the component's sensitivity from the analysis:
   - For **Tier 1 (Constraint Layer - High Sensitivity)** components, apply only MINIMAL and SUBTLE changes (e.g., synonym swaps like "only" to "just", slight rephrasing). These are fragile and require careful stress-testing.
   - For **Tier 2 (Method Layer - Medium Sensitivity)** components, you can apply MODERATE changes (e.g., rephrasing the reasoning process, altering the sequence of steps).
   - For **Tier 3 (Style Layer - Low Sensitivity)** components, you have the most freedom. Apply CREATIVE and DIVERSE changes (e.g., completely changing the persona, tone, or conversational style).
3. **No Invalid Information:** Do not introduce irrelevant information, contradictions, or factual errors.

Now, provide the `{k}` targeted, perturbed versions for the original text, strictly following all the rules above.

F.3 LATO Optimizer Prompt

The LATO prompt is the engine for the Outer Robustness Update step. It is composed of three main parts: a glossary that defines the structured tags, a system prompt that outlines the optimizer's core task, and an instantiated user message that provides the specific context for an optimization step.

Glossary

To ensure the optimizer LLM correctly interprets the structured inputs, we first provide it with a glossary defining all the tags used in the prompt.

LATO Prompt Glossary

- `<ORIGINAL_VARIABLE>`: The original variable that you need to improve.
- `<PERTURBED_VARIABLE>`: A slightly perturbed version of the original variable that resulted in the feedback.
- `<ORIGINAL_VARIABLE_LOSS>`: The performance of the original variable on the current batch.
- `<PERTURBED_VARIABLE_LOSS>`: The performance of the perturbed variable on the current batch.
- `<LM_SYSTEM_PROMPT>`: The system prompt for the language model.
- `<LM_INPUT>`: The input to the language model.
- `<LM_OUTPUT>`: The output of the language model.
- `<FEEDBACK>`: The feedback to the variable.
- `<CONVERSATION>`: The conversation history.
- `<FOCUS>`: The focus of the optimization.
- `<ROLE>`: The role description of the variable.

LATO System Prompt

The LATO system prompt is designed to make the optimizer LLM explicitly landscape-aware. By providing a rich, contextual view of the local semantic landscape and the nature of a performance failure, it enables a more informed update than first-order methods, steering the variable towards a flatter, more robust semantic basin.

LATO System Prompt

You are an expert optimizer and a creative critic within an advanced AI system. You will be asked to creatively and critically improve text-based variables (prompts, solutions, code, etc.) to make them more effective and robust.

THE PROCESS To do this, you will be given an `<ORIGINAL_VARIABLE>`. This variable was perturbed into a `<PERTURBED_VARIABLE>`, and the system's performance using this perturbed version resulted in critical `<FEEDBACK>`.

YOUR TASK & OBJECTIVES Based on all available information, your goal is to generate a new, improved version of the `ORIGINAL_VARIABLE`. The new version must achieve the following objectives:

1. **Address the Failure:** It must resolve the specific issues pointed out in the provided `<FEEDBACK>`.
2. **Preserve Performance:** It must maintain or improve upon the original variable's good performance.
3. **Enhance Robustness:** It must be more resilient to similar small perturbations in the future.

GUIDING PRINCIPLES

- **Preserve Core Meaning:** Whatever the edit, you must strictly preserve the core task intent and local coherence of the original text.
- **Analyze Noisy Feedback:** The provided `<FEEDBACK>` may be noisy. Critically evaluate it to identify what is important and correct.
- **Consider Full Context:** Always pay attention to the variable's `<ROLE>` and the full context in which it is used to ensure your improvements are relevant.

IMPORTANT - OUTPUT FORMAT You MUST give your response by sending the improved variable between `{new_variable_start_tag}{improved variable}{new_variable_end_tag}` tags. The text you send between the tags will directly replace the variable. GLOSSARYTEXT

Example of an Instantiated LATO Prompt

The following box shows an example of a complete prompt constructed and sent to the optimizer LLM, combining the system prompt with the specific variables for a single optimization step.

Example of an Instantiated LATO Prompt

Here is the role of the variable you will improve: `<ROLE>structured system prompt to a language model</ROLE>`.
The optimizer is provided with two key variables that define the local semantic landscape, along with their performance (loss) on the current batch:

- **The ORIGINAL variable that we are optimizing is:**

```
<ORIGINAL_VARIABLE>
You will answer a reasoning question. Think step by step. The
last line of your response should be of the following format:
‘‘Answer: $VALUE’’ where VALUE is a numerical value.
</ORIGINAL_VARIABLE>
<ORIGINAL_VARIABLE_LOSS> 1, 1, 1 </ORIGINAL_VARIABLE_LOSS>
```

- **When this variable was slightly perturbed into the following version:**

```
<PERTURBED_VARIABLE>
You are to solve a reasoning question. The final line of your
response should be in the format: ‘‘Answer: $VALUE’’ where
VALUE is a numerical value.
</PERTURBED_VARIABLE>
<PERTURBED_VARIABLE_LOSS> 1, 1, 1 </PERTURBED_VARIABLE_LOSS>
```

The system received the following feedback based on the PERTURBED version's performance:

```
<CONTEXT>
Here is a conversation:
<CONVERSATION>
...
<LM_INPUT>
I have three oranges, a pig, a frog, a cow, three bananas,
a nectarine, and a snail. How many animals do I have?
</LM_INPUT>
<LM_OUTPUT>
To find the total number of animals, we need to identify the
animals...So, the total number of animals is 4.
Answer: 4
</LM_OUTPUT>
</CONVERSATION>
Here is the feedback we got in the conversation:
<FEEDBACK>
To improve the adaptively perturbed prompt variable in order
to enhance the objective function, consider the following
strategies:
...
</FEEDBACK>
</CONTEXT>
```

Based on the feedback from the perturbed version, improve the ORIGINAL variable to make it more robust.

F.4 Solution Optimization Prompt Details

This section details the prompt architecture for our **ATARE**-based solution optimization pipeline. The process is divided into two main stages, each with a corresponding prompt.

The first stage is Flaw Diagnosis, which operationalizes the Semantic Sensitivity Analysis step. The prompt below instructs an expert LLM to identify the core "cognitive trap" in an incorrect solution, thereby defining the low- and high-sensitivity regions.

Flaw Diagnosis Prompt

You are a world-class expert in logic and science. The following solution is INCORRECT. Your task is to deeply analyze its reasoning and clearly describe the core "cognitive trap" or "flawed reasoning path" that led to the wrong conclusion.

Incorrect Solution to Diagnose (s_t) `{solution_var.value}`

Please provide your detailed analysis of its flawed reasoning path.

The second stage is the Anisotropic Adversarial Search, executed by the **ATARE** Perturbation Generator. This prompt takes the flaw analysis from Stage 1 as input and directs the LLM to generate diverse variations of only the identified flaw, while strictly preserving the correct reasoning backbone.

ATARE Perturbation Generator Prompt

You are a creative AI that can mimic different thinking styles. You will receive an incorrect solution and an analysis of its core flaw. Your task is to generate `{k}` new, distinct, but equally flawed solutions.

Guiding Principle Treat the solution as a reasoning chain. The correct reasoning *before* the identified flaw is the high-sensitivity backbone that **MUST** be preserved. Your task is to vary the expression of the flaw itself (the low-sensitivity target).

Rules All new solutions **MUST**:

1. Commit the same type of core error as described in the "Flaw Analysis".
2. Use different phrasing, examples, or intermediate steps to express this error. The goal is to explore different deceptive manifestations of this single cognitive trap.
3. Ensure that the final conclusion and the chosen answer letter **LOGICALLY FOLLOW** from your flawed reasoning.
4. Choose your final answer from the available options in the "Problem Context". **Do not** invent new options.

Inputs

- **Problem Context:** `{question}`
- **Original Incorrect Solution (s_t):** `{solution_var.value}`
- **Flaw Analysis Report:** `{flaw_analysis}`

Output Format The output **MUST** be a valid Python list of strings.

NeurIPS Paper Checklist