Beyond Prompt Engineering: A Systematic Analysis of Prompt Lexical Sensitivity and Its Impacts on Quality

Qipeng Xie[1,5]*, Zi Liang[2]*, Jiafei Wu[3]†, Yufei Chen[3], Weizheng Wang[2], Wenao Ma[4]†, Zhong Ming[1], Haiqin Yang[1]†, Kaishun Wu[5]

[1]Shenzhen Technology University, [2]The Hong Kong Polytechnic University, [3]Zhejiang Lab
[4]The Chinese University of Hong Kong, [5]HKUST (Guangzhou)

[1]{qxieaf@connect.ust.hk, wuks@hkust-gz.edu.cn} [2]{zi1415926.liang@connect.polyu.hk, weizheng.wang@ieee.org}

[3]{wujiafei@zhejianglab.org, cyf200409@gmail.com} [4]wenaoma@gmail.com [5]{yanghaiqin@sztu.edu.cn}

Abstract

Large Language Models (LLMs) often exhibit extreme sensitivity to surface-level prompt variations, where minor lexical perturbations trigger disproportionate performance fluctuations. Moving beyond black-box optimization or coarse-grained templates, we conduct the first n-gram token-level mechanism analysis leveraging a large-scale dataset of 132,000 prompt variants. Our investigation uncovers the Scaling Law of Prompt Performance Stability: higher average performance is inherently associated with lower variance and greater stability. We identify that this robustness is driven by two linguistic pillars: Domain-Specific Terminology, which anchors semantic boundaries, and Explicit Action Directives, which formalize reasoning trajectories. By narrowing the model's interpretative space, these patterns effectively "lock" the generation process. We operationalize these findings into an automated Prompt-Refining Agent that autonomously restructures queries via domain anchoring and operational constraints. Empirical results show a 40.7% reduction in performance variance for code generation, offering a statistically grounded framework for robust prompt engineering.

1 Introduction

With the rapid development of large language models (LLMs), systematically evaluating their robustness and consistency across different prompt formulations has become increasingly critical (Liu et al., 2023a; Dong et al., 2024; Mei et al., 2025). As illustrated in Figure 1 (top), prompt engineering often suffers from a "butterfly effect": even minor lexical variations can lead to substantial performance fluctuations on the same task (Lu et al., 2024). This phenomenon, termed prompt instability, undermines the reliability and reproducibility essential for LLM deployment. Although prompt engineering has emerged as a practical solution to mitigate this issue, it largely relies on heuristic trial-and-error rather than a principled understanding of the underlying mechanisms. Consequently, a fundamental research question remains unanswered: ***What determines whether a prompt can achieve both high average performance and stable behavior under stylistic perturbations?***

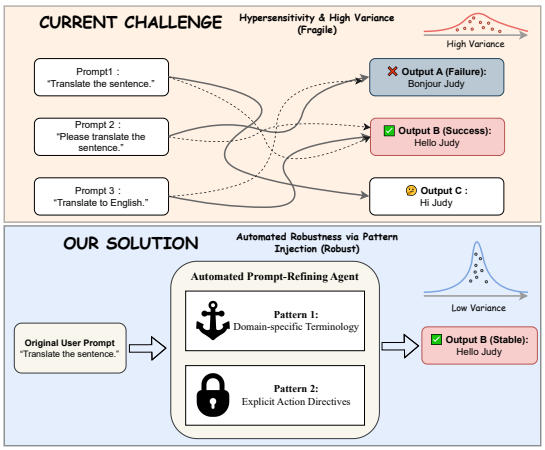


Figure 1: **The Prompt Sensitivity Challenge vs. Our Solution.** Top: Minor lexical variations in user prompts trigger drastic performance fluctuations (High Variance). Bottom: Our Automated Prompt-Refining Agent stabilizes generation by injecting Domain-Specific Terminology and Explicit Action Directives (Low Variance).

Existing studies are divided into two directions. Firstly, optimization-oriented approaches aim to improve overall model performance through techniques such as calibration (Zhao et al., 2021), ensembling (Sclar et al., 2024), regularized training (Liu et al., 2021), and format mixing (Perez et al., 2021). Although empirically effective, these methods focus primarily on performance gains and offer limited insight into the underlying mechanisms that govern prompt effectiveness. The second focuses on template-level (Zhu et al., 2024b; Pezeshkpour and Hruschka, 2024) or decoding-level (Sclar et al.,

*Equal contributions.
†Corresponding authors.

Findings of the Association for Computational Linguistics: ACL 2026, pages 41998–42012

2024) analyses, examining how different prompt templates or decoding strategies affect model behavior. However, these methods are typically conducted at a coarse granularity, making it difficult to capture how specific fine-grained token-level patterns shape model robustness. Moreover, existing sensitivity analyses are either overly abstract or rely on gradient-based saliency methods (Mizrahi et al., 2024), which identify influential tokens for individual model decisions on specific examples. These are often applied to small-scale samples and therefore lack support from large-scale statistical evidence.

To bridge this gap, we take a complementary perspective by conducting systematic mechanism analysis rather than proposing optimization techniques. Specifically, we investigate prompt sensitivity at the **n-gram token level**, an unprecedented fine-grained scale, through large-scale statistical analysis. Unlike recent studies that rely on gradient-based saliency for limited examples (Mizrahi et al., 2024), we provide explicit token-level statistical evidence across 12,000 instructions with 11 variants each (132,000 total evaluations), enabling us to identify interpretable token patterns that characterize high-robustness prompts. Furthermore, we reveal a fundamental scaling law that connects a prompt's performance with its stability, offering a new and mechanism-oriented perspective on prompt engineering, which can be seen in Figure 1 (bottom). Our contributions in this work are summarized as follows:

- We present **the first large-scale, n-gram–level statistical analysis of prompt sensitivity**, grounded in a comprehensive dataset of 132,000 distinct prompt formulations, enabling fine-grained and statistically robust characterization of prompt behavior.
- We empirically identify and validate **a scaling law linking prompt performance and robustness**, showing that prompts with higher average performance consistently exhibit greater stability under stylistic perturbations.
- We formalize **two key linguistic patterns** underlying high-robustness prompts, domain-specific terminology and explicit action directives, into **an automated Prompt-Refining Agent**. By constraining the interpretative space, this agent achieves a 40.7% reduction in performance variance, offering a statistically grounded solution for robust prompt engineering.

2 Related Work

Prompt Engineering & Sensitivity Analysis. Prompt engineering aims to unlock LLM capabilities through carefully crafted input. However, research consistently demonstrates that LLMs exhibit hypersensitivity to surface-level prompt variations: even semantically equivalent perturbations can trigger drastic performance volatility. The ubiquity of this phenomenon is systematically quantified, identifying model susceptibility to spurious feature dependence and formatting shifts as primary drivers (Sclar et al., 2024; Voronov et al., 2024). This sensitivity extends beyond lexical phrasing to contextual structure. Empirical evidence (Pezeshkpour and Hruschka, 2024) reveals that the permutation of answer options in multiple-choice tasks significantly biases decision-making. Furthermore, in few-shot learning contexts (Lu et al., 2022), the ordering of demonstrations often outweighs the impact of the example count itself. To quantify such sensitivity, recent works (Zhuo et al., 2024; Lu et al., 2024) propose metric-based frameworks. For instance, the PROSA (Zhuo et al., 2024) is introduced to assess robustness under prompt perturbations, while others (Lu et al., 2024) utilize gradient-based saliency analysis to characterize sensitivity differences. Nevertheless, these approaches predominantly operate at the coarse-grained template level, lacking a systematic statistical attribution of fine-grained lexical units across large-scale instruction datasets.

Prompt Optimization & Calibration. To mitigate instability caused by prompt sensitivity, prior studies (Zhao et al., 2021; Zhou et al., 2024; Yao et al., 2024) investigate inference-time calibration and training-time consistency constraints. During inference, the "calibrate-before-use" (Zhao et al., 2021) minimizes inter-prompt variance by recalibrating output distributions, and another method (Zhou et al., 2024) extends to batch calibration for in-context learning. During training, (Yao et al., 2024) introduces Prompt Perturbation Consistency Learning, which regularizes models to maintain output consistency across paraphrased inputs. Different from those that treat prompt sensitivity as noise to be eliminated, this work posits that sensitivity is a structured statistical phenomenon. Through large-scale, fine-grained analysis, we aim to un-

cover the intrinsic linguistic properties of high-performing prompts, thereby explaining the mechanism behind their inherent robustness.

3 Methodology

This section details a systematic prompt perturbation framework based on orthogonal rewriting strategies. Unlike prior work relying on random noise or single paraphrases, the proposed design enables multidimensional stress testing of prompts under realistic user input variations.

3.1 Systematic Perturbation Strategies

To simulate the diversity of real-world user inputs and examine model sensitivity across multiple dimensions, we define five semantically equivalent yet stylistically distinct rewriting strategies. For each original instruction, the Gemini-2.5-flash model (Comanici et al., 2025) is used to generate a corresponding variant under each strategy, as details in Appendix A.1.

The definitions of the five strategies are as follows: **(1) Semantic Equivalence Paraphrasing (SEP):** Involving extensive syntactic reorganization and synonym substitution, this strategy strictly preserves core task semantics while simulating the natural variations in how different users articulate the same intent (Zhu et al., 2024a; Moore and Shah, 2025). **(2) Adding Contextual Information (ACI):** By injecting substantial unrelated background information—such as historical trivia or fiction—we challenge the model's attention stability, testing its ability to extract core directives from noisy, long-tail contexts (Shi et al., 2023; Yoran et al., 2024). **(3) Changing Format/Style (CFS):** The instruction's tone is transformed into distinct pragmatic registers (*e.g.*, legal, academic, or casual) to evaluate the model's robustness against significant shifts in linguistic style and formality (Ngweta et al., 2025). **(4) Introducing Ambiguity (IA):** Uncertainty is introduced through hedging terms (*e.g.*, "roughly") and conditional clauses, assessing the stability of the model's reasoning processes when facing non-rigid or probabilistic constraints (Kim et al., 2024). **(5) Syntactic Noise (SN):** Simulating low-quality user inputs, this strategy perturbs the surface form with orthographic errors, including typos and irregular spacing, while maintaining basic intelligibility at the character level (Liu et al., 2025).

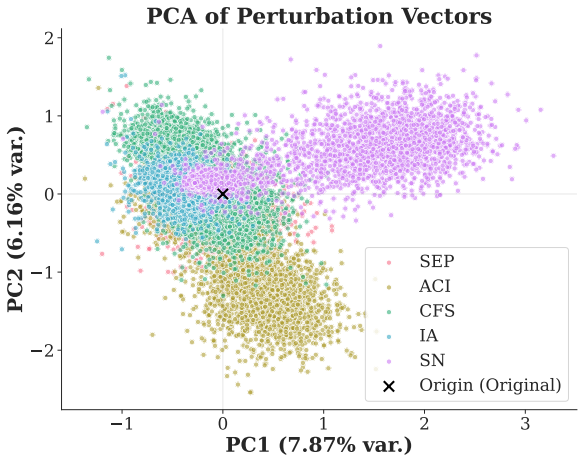


Figure 2: PCA visualization of delta vectors. Each point represents the embedding displacement induced by one rewritten prompt; the origin corresponds to zero displacement.

3.2 Methodological Validation

We validate the proposed strategy set as a measurement tool by examining its orthogonality in embedding space and its coverage of linguistic features.

3.2.1 Orthogonality of Perturbation Directions

It is essential to verify that different strategies probe distinct dimensions of model capability rather than redundantly testing the same feature. To this end, for each rewritten instruction, we compute the delta vector $\Delta\mathbf{v}$ in the embedding space.

$$\Delta\mathbf{v} = \mathbf{v}_{\text{rewrite}} - \mathbf{v}_{\text{original}}, \quad (1)$$

where $\mathbf{v}(\cdot)$ denotes the embedding function used in our experiments.

Cluster Analysis Principal Component Analysis (PCA) (Maćkiewicz and Ratajczak, 1993) on the $\Delta\mathbf{v}$ reveals that samples from different strategies from distinct, well-separated clusters in the project space. As shown in Figure 2, SN and ACI exhibit large displacements from the origin, whereas IA remains clustered near the center.

Cosine Similarity We compute pairwise cosine similarity between $\Delta\mathbf{v}$ of different strategies. The results shown in Figure 3 (a) indicate a low average off-diagonal similarity of approximately 0.22. Notably, the similarity between SN and others is even lower ($\approx$ 0.12).These findings demonstrate the five strategies induce highly independent per-

turbation directions, minimizing redundancy in the evaluation.

3.2.2 Coverage of Perturbation Intensity

Beyond semantic vector analysis, we analyze lexical features to verify the coverage of non-semantic dimensions.

Literal and Informational Coverage Lexical feature analysis in Figure 3 (c) demonstrates that SN yields a high normalized edit distance, effectively covering literal and surface-form perturbations. Meanwhile, ACI results in a length ratio significantly greater than 1.0, achieving systematic information expansion.

Intensity Spectrum We further compare strategies by their average semantic distance to the original instruction. Figure 3 (b) shows that the five strategies span a range of perturbation intensities. This ranges from the micro-perturbations of IA (average distance ≈ 0.07), transitioning through the mid-range of SEP and CFS, to the strong perturbations of SN (average distance ≈ 0.36).

4 Experiments

This section validates the systematic perturbation framework via large-scale empirical studies, quantifies the relationship between prompt performance and robustness, and dissects the underlying mechanisms from a microscopic perspective.

4.1 Settings

Dataset To ensure comprehensive coverage of diverse instruction types and complexity levels, we employ the WizardLM_evol_instruct_70k (Luo et al., 2025) dataset, which consists of highly complex instructions generated through multiple rounds of evolutionary optimization, including tasks such as code writing, creative writing, and math calculation. From this dataset, we randomly sample 12,000 instances to form the core experimental benchmark, striking a balance between statistical significance and computational cost.

Pipeline Our experimental framework employs a three-model pipeline. **Rewriter** utilizes Gemini-2.5-flash (Comanici et al., 2025) to generate 10 variants for each original instruction based on the five strategies defined in Sec. 3.1, resulting in a set of 11 prompts per instance (1 original and 10 variants). **Generator** employs `Qwen-plus` [1] (Yang et al., 2025) to synthesize responses for the entire corpus of 132,000 instructions ($12k \times 11$). Finally, to strike a balance among model intelligence, inference speed, and cost efficiency, we employ `Grok-4-fast` [2] as **Evaluator** to perform fully automated quality assessments adopting the LLM-as-a-Judge paradigm (Zheng et al., 2023; Liu et al., 2023b; Gu et al., 2025).

Evaluation Metrics To capture fine-grained quality nuances, we implement a continuous 1-100 scoring mechanism (Appendix A.2) that explicitly counters the central tendency bias prevalent in LLM judges (Wang et al., 2023; Zheng et al., 2023). Adopting a 'Utility Hierarchy' aligned with HELM (Liang et al., 2023) and FLASK (Ye et al., 2023), we prioritize foundational validity before addressing robustness. Specifically, we introduce a composite metric $\mathcal{S}$ based on a *utility hierarchy*. As shown in Eq. (2), the weighting scheme prioritizes *foundational validity* over stylistic features: **Accuracy** (0.30) and **Relevance** (0.25) are assigned the highest weights (cumulatively 0.55) to strictly penalize hallucinations and off-topic responses. **Completeness** (0.20) and **Clarity** (0.15) follow to ensure constraint adherence and readability, while **Practical Value** (0.10) serves as a supplementary metric to reward actionable insights:

$$\begin{aligned}\mathcal{S} = {} & 0.30 \cdot \text{Accuracy} + 0.25 \cdot \text{Relevance} \\ & + 0.20 \cdot \text{Completeness} + 0.15 \cdot \text{Clarity} \\ & + 0.10 \cdot \text{Practical Value}, \end{aligned} \tag{2}$$

Then, we compute respectively the Mean Score (μ) and Standard Deviation (σ) across each group of 11 samples, serving as the core metrics for quantifying prompt performance and sensitivity based on the LLM-as-a-Judge.

4.2 The Scaling Law of Prompt Sensitivity

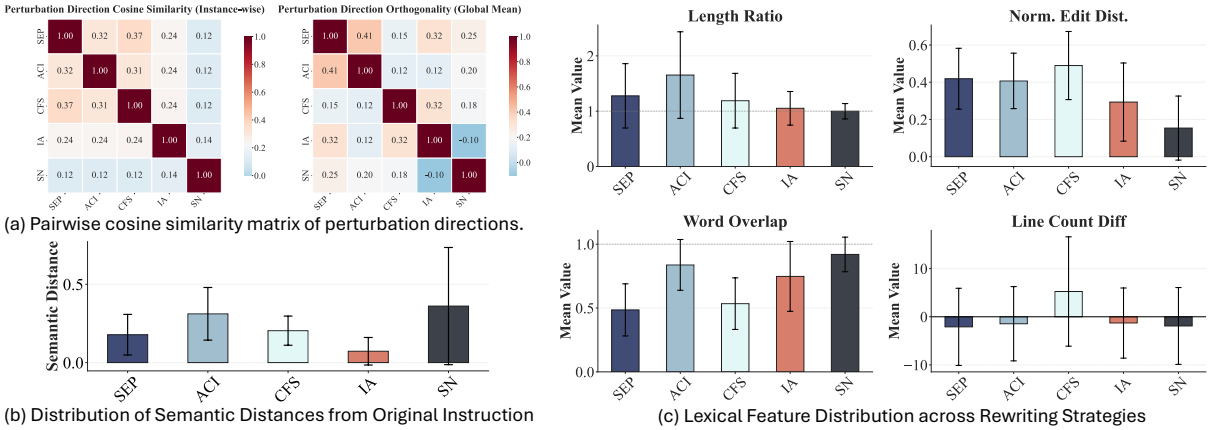
Figure 4 reveals a striking negative correlation between mean performance and response variability. Through Ordinary Least Squares (OLS) regression, we obtain:

$$y = -1.083x + 93.60, \quad R^2 = 0.7006 \tag{3}$$

where y represents the mean performance score and x denotes the standard deviation (prompt sensitivity).

[1] `modelstudio.console.alibabacloud.com`

[2] `https://x.ai/news/grok-4-fast`



(a) Pairwise cosine similarity matrix of perturbation directions.

(b) Distribution of Semantic Distances from Original Instruction

(c) Lexical Feature Distribution across Rewriting Strategies

Figure 3: (a) Pairwise cosine similarity matrix of perturbation directions. (b) Average semantic distance from the original instruction by strategy, illustrating a graded perturbation spectrum. (c) Lexical feature analysis, showing that strategies induce measurable variation in surface form and structure.

This relationship establishes a fundamental scaling law: prompts with higher average performance exhibit systematically greater robustness to stylistic variations. The estimated negative slope of -1.083 quantitatively characterizes this trade-off, indicating that a one-unit increase in performance variability (standard deviation) is associated with an average decrease of approximately 1.08 points in mean performance. Moreover, the coefficient of determination ($R^2 = 0.70$) suggests that robustness to stylistic perturbations accounts for a substantial proportion (70%) of the observed variance in prompt performance.

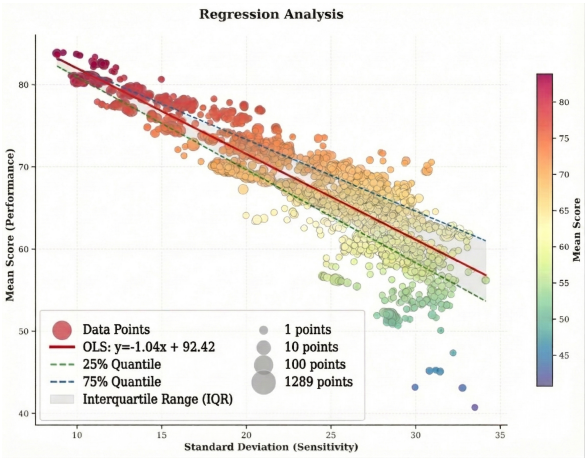


Figure 4: **The Scaling Law of Prompt Sensitivity.** Scatter plot showing mean performance vs. standard deviation across 12,000 instructions with 11 variants each (132,000 total evaluations). The red line shows OLS regression ($y = -1.083x + 93.60$, $R^2 = 0.7006$). Color gradient indicates point density.

4.2.1 Implications

The identified scaling law carries profound implications across three critical dimensions: **(1) Methodologically**, it necessitates re-conceptualizing prompt sensitivity—not as transient measurement noise to be averaged out, but as a fundamental dimension of model capability that warrants explicit reporting. **(2) Practically**, it guides production deployment strategies to prioritize prompt designs situated on the *high-performance, low-variability* frontier of this scaling relationship; and **(3) Mechanistically**, the intrinsic correlation suggests that robustness and performance stem from shared underlying linguistic factors, a hypothesis we dissect in the subsequent section.

5 Token-Level Analysis

In this section, we further investigate token-level factors that contribute to prompt robustness and design a controlled experiment to examine whether explicitly injecting these patterns into standard prompts can effectively enhance robustness across diverse tasks.

5.1 Stratification Strategy

To disentangle token-level linguistic mechanisms of robustness, we map the 12,000 core instructions onto a "Performance-Sensitivity" coordinate system. By applying the median Mean Score (μ) and median Standard Deviation (σ) as orthogonal thresholds, we partition the dataset into four mutually exclusive quadrants, see Table 1 for details. This strategy specifically isolates the High-Performance, High-Robustness (HP-HR) group, which we designate as the "Ideal State" of prompt engineering for its ability to simultaneously achieve

Quadrant	Criteria	Behavioral Archetype	Utility Assessment
HP-HR	$\mu > \text{Med}, \sigma < \text{Med}$	*The Ideal State*	**Optimal**: Consistently high quality; resilient to perturbations.
HP-LR	$\mu > \text{Med}, \sigma > \text{Med}$	*High Risk, High Return*	**Unreliable**: High potential but statistically unstable output.
LP-HR	$\mu < \text{Med}, \sigma < \text{Med}$	*Consistent Mediocrity*	**Limited**: Stable behavior but persistently fails to meet quality standards.
LP-LR	$\mu < \text{Med}, \sigma > \text{Med}$	*Chaotic Failure*	**Detrimental**: Both low-quality and highly unpredictable.

Table 1: **Quadrant Stratification Taxonomy.** We partition prompts into four distinct categories based on median thresholds of μ and σ. The **HP-HR** group represents the ideal target for robust prompt engineering.

high quality and stability. We then conduct a contrastive extraction of distinctive n-grams ($n = 1, 2, 3$) using Log Odds Ratio with add-k smoothing ($k = 0.01$)(Monroe et al., 2008). This statistical approach rigorously identifies tokens that are *statistically overrepresented* in robust prompts compared to the general population.

5.2 Key Findings: Interpretable Token Patterns

Figure 5 presents the top-20 most distinctive n-grams for the HP-HR quadrant. Our analysis reveals two prominent patterns: Domain-Specific Terminology and Explicit Action Directives.

Pattern 1: Domain-Specific Terminology (Semantic Anchors) Distinctive 1-grams are dominated by specialized terms (e.g., "algorithm", "protocol", "specification") that drastically narrow the semantic search space.
Example (Code Writing):

- × **Vague:** "Check the **inputs**" → Ambiguous (data types vs. logical validity?), leading to high variance.
- ✓ **Robust:** "Validate **function parameters**" → Locks the model into a standard programming context, acting as a semantic anchor.

Pattern 2: Explicit Action Directives (Operational Anchors) While domain terms define what to discuss, distinctive n-grams like "compare and contrast" and "step-by-step explanation" define how to structure the response.
Example (Analytical Reasoning):

- × **Open-ended:** "Tell me about X" → High structural degrees of freedom.
- ✓ **Robust:** "**Compare and contrast** X using **bullet points**" → Constrains the execution path and output scaffolding, eliminating the "butterfly effect" of lexical perturbations.

High-potential prompts thus enhance robustness by minimizing the interpretation space. simultaneously specifying the semantic domain and the structural template.

5.3 Automated Optimization via Prompt-Refining Agent

To validate the casual utility of our findings and provide a scalable solution, we design an automated Prompt-Refining Agent. Instead of treating prompt engineering as a manual heuristic, this agent autonomously transforms fragile user queries into robust instructions by strictly adhering to the Pattern Injection principles identified in Sec. 5.2.

5.3.1 Pattern Injection

The agent functions through a two-stage injection pipeline, explicitly designed to minimize the interpretation space:

- Domain Anchoring: Replacing generic descriptions with Domain-Specific Terminology to narrow the semantic search space (e.g., refining "fix the code" into "debug the implementation based on syntax standards").
- Constraint Refinement: Injecting clear formatting and operational directives to lock the execution path (e.g., appending "provide a step-by-step derivation" or "format as bullet points"), the agent injects Explicit Action Directives.

We then conducted a comparative evaluation (Base vs. Improved) across 2,000 samples per task. Our primary metric for robustness is the change in Standard Deviation (Std), while Mean Score (Mean) serves as the indicator for performance quality.

5.3.2 Quantitative Analysis

The results of this intervention are summarized in Table 2. The data provides strong empirical evidence that our pattern injection strategy significantly mitigates prompt sensitivity across six diverse task domains.

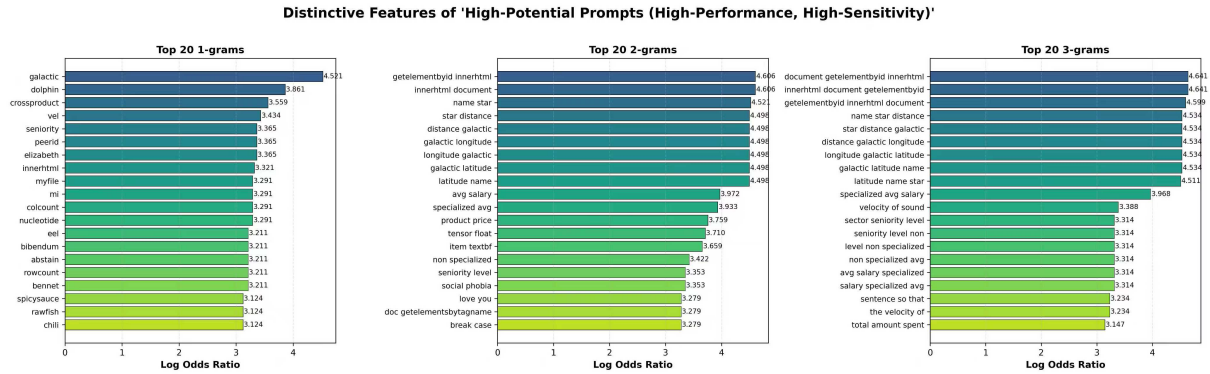


Figure 5: **Distinctive Token Patterns in High-Robustness Prompts.** Top-20 n-grams ranked by Log Odds Ratio for the high-performance, high-robustness quadrant (N=3,247 instructions) compared to the union of other quadrants (N=8,753). Left: 1-grams (domain terminology). Center: 2-grams (action phrases). Right: 3-grams (structured directives). Error bars show 95% confidence intervals via bootstrap resampling (1,000 iterations).

Task Domain	**Metric**	**Base Prompt**		**Improved Prompt (Ours)**		**Improvement**	
		Mean (↑)	Std (↓)	Mean (↑)	Std (↓)	Δ Mean	Δ Std
Code Writing	Rouge-L	16.77	3.24	**17.40**	**1.92**	+3.8%	-40.7%
	LLM-as-a-Judge	77.17	12.96	**83.22**	**7.73**	+7.9%	-40.4%
Creative Writing	Rouge-L	**17.01**	3.61	16.44	**1.92**	-3.4%	-46.9%
	LLM-as-a-Judge	**78.85**	9.13	72.55	**4.28**	-8.0%	-53.1%
Math Calculation	Rouge-L	18.69	3.71	**19.64**	**2.32**	+5.1%	-37.3%
	LLM-as-a-Judge	75.70	13.08	**78.48**	**10.09**	+3.7%	-22.8%
Format Conversion	Rouge-L	18.76	3.98	**19.09**	**2.00**	+1.8%	-49.7%
	LLM-as-a-Judge	80.76	11.82	**84.65**	**8.41**	+4.8%	-28.8%
Health Medical	Rouge-L	**16.66**	3.18	15.96	**1.58**	-4.2%	-50.1%
	LLM-as-a-Judge	**83.49**	8.71	81.47	**4.33**	-2.4%	-50.2%
Logic Puzzles	Rouge-L	15.34	3.18	**16.43**	**1.96**	+7.2%	-38.4%
	LLM-as-a-Judge	73.50	14.15	**80.32**	**9.53**	+9.3%	-32.6%

Table 2: **Performance and Stability Comparison.** Comparison between the Base prompts and our Improved prompts (incorporating identified robust patterns) across six diverse tasks. Results are averaged over 2,000 samples. **Mean** denotes average performance (higher is better), and **Std** denotes standard deviation (lower indicates better stability). The best results are highlighted in **bold**.

The most striking empirical finding is the universal decline in Standard Deviation across all evaluated tasks and metrics, underscoring a significant gain in stability. Specifically, in Code Writing, which is a domain highly sensitive to syntactic precision, the Improved Prompts achieved a remarkable 40.7% reduction in Rouge-L Std and a 40.4% reduction in LLM-as-a-Judge Std. This result confirms that introducing "Explicit Action Directives" effectively constrains the code generation space, preventing the model from drifting into non-functional stylistic variations. The LLM-as-a-Judge metric showed particularly dramatic improvement, with both Mean Score increasing by 7.9% and Std decreasing from 12.96 to 7.73, demonstrating enhanced quality alongside stability.

Similar trends were observed in Math Calculation where Rouge-L Std decreased by 37.3% and LLM-as-a-Judge Std by 22.8%, demonstrating that "Domain Anchoring" successfully ensures model adherence to standard mathematical notations and reasoning steps. This pattern extends to other technical domains such as Format Conversion (49.7% Rouge-L Std reduction) and Logic Puzzles (38.4% Rouge-L Std reduction), confirming the broad applicability of our approach.

Beyond stability, our results refute the common concern that enforcing robustness must come at the expense of performance. Instead, constraining the interpretation space frequently enhances qual-

ity. Across the six tasks, four domains exhibited simultaneous improvements in both Mean performance and stability. Logic Puzzles achieved the most dramatic Mean Score gains (+7.2% Rouge-L, +9.3% LLM-as-a-Judge), while Code Writing and Format Conversion also showed consistent positive lifts across both metrics.

A nuanced trade-off appears in the open-ended domains of Creative Writing and Health Medical, where we observed slight reductions in Mean Scores across both subjective and objective metrics (Creative Writing: −3.4% Rouge-L, −8.0% LLM-Judge; Health Medical: −4.2% Rouge-L, −2.4% LLM-Judge). We attribute this phenomenon to the *constriction of the solution space*. In these open-ended tasks, unconstrained prompts allow the model to traverse a vast execution topology, occasionally reaching high-scoring local maxima through high variance ("creative but unstable" outputs). Our pattern injection imposes necessary structural constraints that prune these diverse but unpredictable execution paths. While this regularization slightly lowers the ceiling for stylistic diversity—reflected in the dipped mean scores—it massively raises the floor for reliability. This is confirmed by drastic reductions in variance across both metrics (Creative Writing: 46.9% Rouge-L Std, 53.1% LLM-Judge Std; Health Medical: 50.1% Rouge-L Std, 50.2% LLM-Judge Std). For safety-critical domains like Health Medical, exchanging minor stylistic fluidity for predictable, hallucination-resistant stability represents a crucial optimization for production readiness.

5.4 Case Study

Figure 6 visualizes the mechanistic impact of pattern injection in Code Writing. The Base Prompt ("Write a python function to sort a list") leaves the *Interpretation Space* unconstrained, resulting in stochastic divergence ranging from built-in methods to inefficient Bubble Sorts. Conversely, the Improved Prompt narrows this space via *Domain Terms* (e.g., "QuickSort") and *Explicit Directives*. Acting as locking mechanisms, these features eliminate ambiguity and force reasoning to converge on a standardized path, demonstrating that robustness is achieved by rigorously defining output boundaries rather than suppressing creativity.

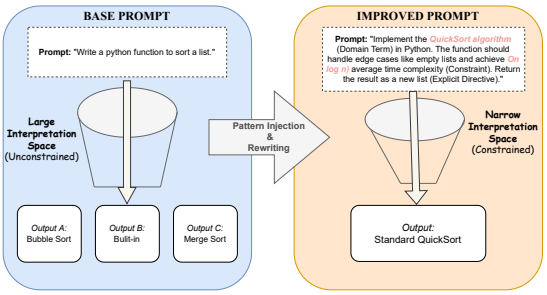


Figure 6: **Case Study of Prompt Robustness.** The left is the base prompt, while the right is our improved prompt.

5.5 Practical Implications for Prompt Engineering

Our findings offer concrete, actionable guidance for designing robust prompts by (1) incorporating domain-specific terminology to clearly signal the task context instead of relying on generic language, and (2) providing explicit operational instructions that specify concrete actions, such as analyze" or compare," along with clear formats and constraints rather than vague directives. Furthermore, (3) combining domain anchoring with execution specification is essential, as the most effective prompts demonstrate a synergy of these patterns to unambiguously define both *what* is relevant and *how* the task should be performed. Crucially, these guidelines are derived not from anecdotal intuition but from the systematic statistical analysis of token patterns established in this study, providing a solid empirical foundation for prompt engineering best practices.

6 Conclusion

This work addresses the challenge of prompt sensitivity in large language models by advancing from anecdotal observations to a systematic, mechanistic analysis. Based on a massive-scale investigation of 132,000 systematic prompt variants, we identify a fundamental scaling law of prompt sensitivity, showing that high-performing prompts are intrinsically associated with low output variance. From a mechanistic perspective, we uncover two linguistic anchors underlying this robustness: domain-specific terminology, which delineates semantic boundaries, and explicit action directives, which constrain reasoning trajectories. We operationalize these findings by designing an automated Prompt-Refining Agent. Experimental deployment confirms that by strictly adhering to Pattern Injection,

our agent reduces performance variance by up to 40.7% in code generation tasks. Overall, this work bridges empirical prompt heuristics and principled prompt engineering, demonstrating that robustness emerges from precisely specifying the model's interpretation space to minimize ambiguity.

Acknowledgments

This work is supported partly by Guangdong Provincial Key Lab of Integrated Communication, Sensing and Computation for Ubiquitous Internet of Things (No.2023B1212010007), China NSFC Grant (No.62472366), 111 Center (No.D25008), the Project of DEGP (No.2024GCZX003, 2023KCXTD042), Shenzhen Science and Technology Foundation (ZDSYS20190902092853047), the China NSFC Grant (No. 62372307, No. U2001207), Guangdong NSF (No. 2024A1515011691), Shenzhen Science and Technology Program (No. RCYX20231211090129039), Shenzhen Science and Technology Foundation (No. JCYJ20230808105906014).

Limitations

While our findings offer significant insights, we acknowledge several limitations that chart the course for future research. First, our experiments primarily relied on the Qwen and Gemini model families; while representative of state-of-the-art architectures, the identified scaling laws and token patterns may exhibit different characteristics in smaller-scale models or distinct architectures such as Mixture-of-Experts. Second, the analysis focused on English prompts within the WizardLM benchmark covering code, math, and creative writing, so the generalizability of "Domain Anchoring" patterns to low-resource languages or highly abstract reasoning tasks remains to be verified. Finally, we employed automated evaluation to scale our experiments, and although we implemented a multi-dimensional scoring rubric to mitigate bias, automated judges may still possess inherent preferences that differ from human evaluation, particularly in subjective domains like creative writing.

References

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, and 1 others. 2024. A survey on in-context learning. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pages 1107–1128.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Saizhuo Wang, Kun Zhang, Yuanzhuo Wang, Wen Gao, Lionel Ni, and Jian Guo. 2025. A survey on llm-as-a-judge. *Preprint*, arXiv:2411.15594.

Hyuhng Joon Kim, Youna Kim, Cheonbok Park, Junyeob Kim, Choonghyun Park, Kang Min Yoo, Sang goo Lee, and Taeuk Kim. 2024. Aligning language models to explicitly handle ambiguity. *Preprint*, arXiv:2404.11972.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Cosgrove, Christopher D. Manning, Christopher Ré, Diana Acosta-Navas, Drew A. Hudson, and 31 others. 2023. Holistic evaluation of language models. *Preprint*, arXiv:2211.09110.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023a. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM computing surveys*, 55(9):1–35.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021. Gpt understands, too. *arXiv:2103.10385*.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023b. G-eval: Nlg evaluation using GPT-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2511–2522.

Yihong Liu, Raoyuan Zhao, Lena Altinger, Hinrich Schütze, and Michael A. Hedderich. 2025. Evaluating robustness of large language models against multilingual typographical errors. *Preprint*, arXiv:2510.09536.

Sheng Lu, Hendrik Schuff, and Iryna Gurevych. 2024. How are prompts different in terms of sensitivity? In *Proceedings of the North American Chapter of the ACL: Human Language Technologies (NAACL-HLT) 2024*, volume 1, pages 5833–5856.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the*

60th Annual Meeting of the Association for Computational Linguistics (ACL 2022), volume 1, pages 8086–8098.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, Yansong Tang, and Dongmei Zhang. 2025. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *Preprint*, arXiv:2308.09583.

Andrzej Maćkiewicz and Waldemar Ratajczak. 1993. Principal components analysis (pca). *Computers & Geosciences*, 19(3):303–342.

Lingrui Mei, Jiayu Yao, Yuyao Ge, Yiwei Wang, Baolong Bi, Yujun Cai, Jiazhi Liu, Mingyu Li, Zhong-Zhi Li, Duzhen Zhang, Chenlin Zhou, Jiayi Mao, Tianze Xia, Jiafeng Guo, and Shenghua Liu. 2025. A survey of context engineering for large language models. *Preprint*, arXiv:2507.13334.

Moran Mizrahi, Guy Kaplan, Dan Malkin, Rotem Dror, Dafna Shahaf, and Gabriel Stanovsky. 2024. State of what art? a call for multi-prompt llm evaluation. *Preprint*, arXiv:2401.00595.

Burt L Monroe, Michael P Colaresi, and Kevin M Quinn. 2008. Fightin'words: Lexical feature selection and evaluation for identifying the content of political conflict. *Political Analysis*, 16(4):372–403.

Hayden Moore and Asfahan Shah. 2025. Evaluating autoformalization robustness via semantically similar paraphrasing. *Preprint*, arXiv:2511.12784.

Lilian Ngweta, Kiran Kate, Jason Tsay, and Yara Rizk. 2025. Towards llms robustness to changes in prompt format styles. *Preprint*, arXiv:2504.06969.

Ethan Perez, Douwe Kiela, and Kyunghyun Cho. 2021. True few-shot learning with language models. *Preprint*, arXiv:2105.11447.

Pouya Pezeshkpour and Estevam Hruschka. 2024. Large language models sensitivity to the order of options in multiple-choice questions. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2006–2017.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. 2024. Quantifying language models' sensitivity to spurious features in prompt design. In *The Twelfth Int. Conf. on Learning Representations, The Twelfth Int. Conf. on Learning Representations*.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. *Preprint*, arXiv:2302.00093.

Anton Voronov, Lena Wolf, and Max Ryabinin. 2024. Mind your format: Towards consistent evaluation of in-context learning improvements. In *Findings of the Association for Computational Linguistics: ACL 2024*.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. 2023. Large language models are not fair evaluators. *Preprint*, arXiv:2305.17926.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Qiang Yao, Subhrangshu Nandi, Ninareh Mehrabi, Greg Ver Steeg, Anoop Kumar, Anna Rumshisky, and Aram Galstyan. 2024. Prompt perturbation consistency learning for robust language models. In *Findings of the Association for Computational Linguistics: EACL 2024*.

Seonghyeon Ye, Doyoung Kim, Sungdong Kim, Hyeonbin Hwang, Seungone Kim, Yongrae Jo, James Thorne, Juho Kim, and Minjoon Seo. 2023. Flask: Fine-grained language model evaluation based on alignment skill sets. *arXiv preprint arXiv:2307.10928*.

Ori Yoran, Tomer Wolfson, Ori Ram, and Jonathan Berant. 2024. Making retrieval-augmented language models robust to irrelevant context. *Preprint*, arXiv:2310.01558.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *International conference on machine learning*, pages 12697–12706. PMLR.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yanping Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36.

Han Zhou, Xingchen Wan, Lev Proleev, Diana Mincu, Jilin Chen, Katherine Heller, and Subhrajit Roy. 2024. Batch calibration: Rethinking calibration for in-context learning and prompt engineering. In *International Conference on Learning Representations (ICLR)*.

Kaijie Zhu, Jindong Wang, Jiaheng Zhou, Zichen Wang, Hao Chen, Yidong Wang, Linyi Yang, Wei Ye, Yue Zhang, Neil Zhenqiang Gong, and Xing Xie. 2024a. Promptrobust: Towards evaluating the robustness of large language models on adversarial prompts. *Preprint*, arXiv:2306.04528.

Kaijie Zhu, Qinlin Zhao, Hao Chen, Jindong Wang, and Xing Xie. 2024b. Promptbench: A unified library for evaluation of large language models. *Preprint*, arXiv:2312.07910.

Jingming Zhuo, Songyang Zhang, Xinyu Fang, Haodong Duan, Dahua Lin, and Kai Chen. 2024. Prosa: Assessing and understanding the prompt sensitivity of llms. In *Findings of the Association for Computational Linguistics: EMNLP 2024*.

A Appendix

A.1 Rewrite Prompt

To construct the "Sensitivity Manifold" of 132,000 prompt variants, we employed Gemini-2.5-flash as the rewriting engine. The model was instructed to generate variations across five orthogonal strategies while strictly preserving the core task semantics and any few-shot examples contained in the original instruction. The full system prompt used for this generation process is provided in Figure 7.

A.2 Evaluation Prompt

For the automated quality assessment, we utilized Grok-4-fast as the evaluator. To mitigate the "clustering effect" often observed in LLM judges (where scores cluster around 7-10 on a 10-point scale), we designed a continuous 1-100 scoring mechanism with weighted dimensions. The full evaluation prompt is detailed in Figure 8.

A.3 Qualitative Comparison of Prompt Variants

Below, we present six representative examples from our dataset, contrasting the original instructions with the robust versions generated by our Prompt-Refining Agent.

You are an expert in language model evaluation and instruction design. Your task is to generate multiple variations of a given task instruction using **five enhanced rewriting strategies**.
A key requirement is that if the original instruction contains any examples (e.g., input-output pairs, formatting samples), these examples MUST be preserved in all generated variations. However, the **position of the examples** in the instruction may be changed (e.g., moved to the beginning, middle, or end).

The underlying task must remain unchanged in all variations. Do not replace the original task with a different one under any circumstance.
--

Strategies (Enhanced Version for Greater Difference)
1. **Semantic Restructuring**
- Completely reorganize sentence structures, change phrasing order, and rewrite the wording extensively.
- Ensure the same task meaning, but make the text look significantly different.
2. **Rich Irrelevant Context**
- Insert multiple sentences of unrelated background information (e.g., historical facts, fictional stories, scientific trivia).
- Keep the actual task intact, but surround it with distracting or decorative content.

3. **Extreme Style Shifting**
- Transform the tone into a very different style (e.g., academic, marketing, casual chat, legal contract, technical manual).
- Keep the task the same, but make the instruction *feel* very different from the original.

4. **Vagueness + Conditional Framing**
- Introduce uncertainty with words like "about," "roughly," "in some cases," or conditional phrases like "if applicable" / "when possible."
- Keep the core task intact, but make it appear less strict and more open-ended.
5. **Structural Noise**
- Add noticeable typos, grammar mistakes, misplaced punctuation, random capitalization, inconsistent spacing, or awkward line breaks.
- The instruction should still be understandable but look much messier

Global Requirements
- Generate **exactly 2 variations for each strategy** (10 total).
- Each variation must be **noticeably different** from the original instruction in at least **two aspects** (e.g., structure, tone, length, or example placement).
- The length of each rewritten instruction can vary by **±30%** compared to the original.
- **All examples must remain intact** in content, but their position in the instruction may shift.
- Output strictly in JSON format as shown below:

```
[{
"strategy": "Semantic Restructuring",
"instruction": "..."
},{
"strategy": "Semantic Restructuring",
"instruction": "..."
},{
"strategy": "Rich Irrelevant Context",
"instruction": "..."
},{
"strategy": "Rich Irrelevant Context",
"instruction": "..."
},
...{
"strategy": "Structural Noise",
"instruction": "..."
}]
```

Figure 7: The rewrite prompt to define five strategies.

You are an expert evaluator with extensive experience in assessing response quality. Your task is to evaluate multiple candidate answers on a **continuous 1-100 scale** and provide output in valid JSON format. Your primary goal is to ensure **full-range score utilization** while maintaining **fine-grained differentiation**.
[Original Prompt] {ori_instruction}
[Candidate Answers]
Answer 1:
{answer_1_content}
.....
Answer 11:
{answer_11_content}
[Scoring Principles]
1. **Full-Range Utilization (1–100)**: You MUST use the entire spectrum. Avoid clustering scores (e.g., 70, 80, 90). A score of 71 should be as likely as70 or 72.
2. **Continuous Scale Mindset**: Treat the scale as a slider. Even subtle differences between two answers (e.g., slightly better clarity) MUST yield different scores (e.g., 88 vs. 89).
3. **Anchored Interpretation of Score Ranges** (for calibration, not strict bins):
• 95–100: Outstanding — Exceeds expectations, innovative or exceptionally comprehensive
• 90–94: Excellent — High quality across all criteria, only minor enhancements possible
• 85–89: Very Good — Strong performance with notable strengths, minimal weaknesses
• 80–84: Good — Solid response, meets requirements with some room for improvement
• 75–79: Above Average — Adequate response with several positive aspects but clear limitations
• 70–74: Average — Meets basic requirements but lacks depth or has moderate issues
• 65–69: Below Average — Partial coverage with significant gaps or errors
• 60–64: Poor — Major deficiencies, limited usefulness
• 50–59: Very Poor — Substantial problems, barely addresses prompt
• 40–49: Severely Inadequate — Misunderstanding or major inaccuracies
• 30–39: Completely Off-topic — Irrelevant or incorrect response
• 20–29: Nonsensical — Incoherent or meaningless content
• 10–19: Minimal Effort — Extremely brief or unhelpful
• 1–9: No Value — Empty, gibberish, or inappropriate

4. **Weighted Judgment** (apply holistically):
- Accuracy: 30%
- Relevance: 25%
- Completeness: 20%
- Clarity: 15%
- Practical Value: 10%
5. **Justification Requirement**: Provide evidence-based reasoning in the `criteria_breakdown`.
[Evaluation Criteria Definitions]
- **Relevance (25%)**: Directness in addressing the prompt.
- **Accuracy (30%)**: Factual correctness and reliability.
- **Completeness (20%)**: Thoroughness in covering all aspects

- **Clarity (15%)**: Organization, readability, coherence.
- **Practical Value (10%)**: Usefulness and applicability of response.
[JSON Output Structure]
{"evaluations": [{
"candidate_id": "Answer 1",
"score": integer 1-100,
"overall_reason": "concise evaluation summary reflecting holistic weighted judgment",
"criteria_breakdown": {
"relevance": "brief assessment",
"accuracy": "brief assessment",
"completeness": "brief assessment",
"clarity": "brief assessment",
"practical_value": "brief assessment"
}}],
"final_ranking": ["Answer X", "Answer Y", "Answer Z", "..."]}

Figure 8: The evaluation prompt of fine-grained quality variations.

Example 1: Logic Puzzles (Trace Interpretation)

Original Prompt (Fragile)

How can the traveler determine the correct number of open hands after the 2021st gesture based on the pattern provided by the villagers?
Here is a possible solution in Ruby: [...]
Can you explain how this Ruby code works to solve the puzzle posed by the villagers?

Improved Prompt (Robust)

As an **expert in Ruby programming and algorithmic sequence computation**, explain how the following Ruby code implements an iterative solution...
Decompose your explanation into these **explicit steps**:
1. Initialize variables and describe the loop structure...
2. Trace the conditional branches...
3. Analyze the final output computation...
Output format: Numbered steps with inline code references and a brief trace...

Figure 9: Comparison for Logic Puzzles. The robust prompt anchors the domain to "algorithmic sequence computation" and provides a structural template for explanation.

Example 2: Health & Medical (Safety Guardrails)

Original Prompt (Fragile)

We have some data about a health condition. Summarize the data in a few sentences.
Felicity has anxiety. She often experiences physical symptoms, such as a racing heart, shortness of breath, and shakes. [...]

Improved Prompt (Robust)

In the **domain of clinical psychology**, synthesize a concise summary... incorporating key concepts such as somatic manifestations...
Output a 3-4 sentence paragraph:
1. Introduce the primary condition...
2. Describe cognitive and fear-related aspects.
3. Conclude with the affective experience.
Do not add diagnoses, interpretations, or external information...

Figure 10: Comparison for Health Medical. The robust prompt enforces strict safety boundaries and structured synthesis.

Example 3: Format Conversion (Grounding Simulations)

Original Prompt (Fragile)

How can I use PHP to parse a JSON object [...] and then retrieve the information for each item to **take a photograph** of each item...

Improved Prompt (Robust)

As a PHP developer... refactor the provided PHP code... to generate and "save" images with descriptive filenames.
Key requirements: ...
- **Simulate image capture** and save by using file_put_contents() to write a placeholder byte string...
- Do not implement actual photography; focus solely on filename generation and file I/O.

Figure 11: Comparison for Format Conversion. The robust prompt translates abstract/impossible requests into concrete engineering simulations.

Example 4: Math Calculation (Algorithmic Constraints)

Original Prompt (Fragile)

```
Can you solve this mountain peak puzzle? [...] the code should not use any built-in Python
functions to sort the data or find the maximum value.
```

Improved Prompt (Robust)

```
As a Python developer specializing in algorithmic constraints and manual data traversal...
Procedural Directives:
1. Define the mountain_peaks list...
2. Initialize variables: max_elevation = -1...
3. Iterate through the list using a for loop...
```

Figure 12: Comparison for Math Calculation. The robust prompt enforces constraints by defining the exact implementation procedure.

Example 5: Creative Writing (Structured Generation)

Original Prompt (Fragile)

```
How can I use C# code to generate a random plot twist for the given short story using the
following Markdown syntax? [...]
```

Improved Prompt (Robust)

```
As a C# developer specializing in procedural content generation...
Requirements:
1. Initialize a Random instance...
2. Define exactly 3 distinct plot twists...
3. Output the selected twist... formatted to insert seamlessly...
Do not alter the original story; append only the twist.
```

Figure 13: Comparison for Creative Writing. The robust prompt isolates the generation scope to prevent content drift.

Example 6: Code Writing (Refactoring Stability)

Original Prompt (Fragile)

```
How can I use Ruby to write a web scraping script... Please let me know if there's anything
I can add or modify to improve the script.
```

Improved Prompt (Robust)

```
As a Ruby web scraping specialist, refactor the following incomplete Ruby script... to
robustly extract prices...
Ensure the script handles exceptions... cleans extracted text...
Do not add new features like authentication or multi-threading; preserve the core array output.
```

Figure 14: Comparison for Code Writing. The robust prompt focuses the model on engineering robustness rather than open-ended feature expansion.