FRI Natural language processing course 2025
Project report

University of Ljubljana
Faculty of Computer and Information Science

Improving Prompt Sensitivity of LLMs

Gonçalo Cardoso, Gopika Krishnan, Muhammad Ali

Abstract

Large Language Models (LLMs) often display prompt sensitivity, where semantically equivalent prompts yield inconsistent outputs. This project explores both inference-time and training-time strategies to improve prompt robustness. At inference time, we evaluate several prompting techniques including Chain of Thought, Self-Consistency, and Self-Refinement to examine how they reduce variability across prompt variants. At training time, we apply Parameter-Efficient Fine-Tuning (PEFT) using Low-Rank Adaptation (LoRA) on grouped prompt variants that share the same intended output. To evaluate prompt sensitivity, we use two complementary methods: the Prompt Sensitivity Index (POSIX), which quantifies output variation across prompt perturbations, and AlpacaEval, which uses a Language Model (LLM) as a judge to assess consistency and response quality. Our work provides a systematic comparison of methods for improving prompt sensitivity, aiming to make language model outputs more stable and reliable across different prompt formulations.

Keywords
Large Language Models (LLMs), Prompt Sensitivity, Chain of Thought, Low-Rank Adaptation (LoRA), Parameter-Efficient Fine-Tuning (PEFT), Instruction Tuning

Advisors: Slavko Žitnik, Marko Robnik-Šikonja

Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities across various natural language processing tasks. However, a significant challenge in their practical application is their sensitivity to subtle variations in user prompts. Even minor changes in phrasing, such as word order or the inclusion of seemingly innocuous words, can lead to drastically different outputs in terms of content, quality, and even correctness. This phenomenon, known as prompt sensitivity, undermines the reliability and consistency of LLMs, hindering their widespread adoption in critical applications. Prior work highlights this instability: Mizrahi et al. document inconsistent LLM behavior under slight prompt perturbations [1], while Cao et al. report performance drops of up to 45.48% between paraphrased prompts [2]. Prompt sensitivity has therefore become a focal point in recent research, motivating both diagnostic metrics and mitigation strategies. Researchers have proposed approaches ranging from log-likelihood-based sensitivity scores to robustness-enhancing prompting techniques and fine-tuning methods.

To systematically assess prompt sensitivity, several evaluation metrics have been proposed. Chatterjee et al. introduced POSIX, a metric that quantifies the invariance of model outputs across paraphrased prompts [3]. ROBUSTALPACAEVAL by Cao et al. benchmarks instruction-following models under adversarial prompt variations by measuring performance drops across rephrasings [2]. Zhuo et al. proposed PromptSensiScore, which evaluates how much a model's output changes under minor prompt modifications[4]. Mizrahi et al. also introduced metrics such as MaxP, AvgP, and CPS, capturing both peak and consistency of performance across variations [1]. Collectively, these tools enable fine-grained analysis and comparison of prompt robustness across models and prompting strategies. Researchers have proposed a range of strategies to mitigate prompt sensitivity in LLMs. Voting-based methods, which aggregate outputs from multiple perturbed prompts, have shown stable performance, although self-refinement techniques, where the model revisits and revises its own outputs, have yielded limited improvements [2]. Prompting strategies such as Few-Shot Learning [4], Chain-of-Thought prompting, and General Knowledge prompting [5] encourage the model to reason or ground its responses, leading to more consistent outputs. While these methods are promising, their effectiveness often varies across models, limiting general applicability.

In this work, we systematically evaluate and compare multiple prompting strategies, including Chain-of-Thought, Iterative Refinement, and Meta-Evaluation prompting, using both metric-based and model-based robustness evaluations. We also fine-tune instruction-following models on grouped paraphrased prompts to reduce sensitivity and improve consistency. Our study combines quantitative metrics like POSIX with LLM-based similarity scoring to provide a multifaceted view of model robustness.

Methods

Datasets

We build on the dataset introduced by [3], which extends the Alpaca dataset [6] with grouped prompt variants to measure model sensitivity to surface-level changes. Each group contains semantically equivalent instructions paired with a shared response. We use a publicly released subset comprising 500 such groups, which we find sufficient for fine-tuning and evaluation.

Each prompt variant is labeled with one of four perturbation types: PARAPHRASE, TEMPLATE SHIFT, SPELLING ERROR, and NOISE INJECTION. We introduce the last category to simulate real-world user inputs with character-level corruption. Prior work shows such perturbations (including emojis, punctuation noise, and Unicode exploits) can degrade model outputs [7, 8, 9]. Our setup tests whether prompting and fine-tuning strategies enhance robustness to both benign and adversarial-style input variations. All prompts follow a consistent format, with a system message prepended and a "Q:" followed by the instruction. Definitions and examples for each perturbation type are provided in Appendix A.

Techniques

We implement several prompting-based interventions, including Chain-of-Thought, Meta-Evaluation, and Iterative Refinement, each designed to encourage more stable model behavior. In addition, we explore parameter-efficient fine-tuning using LoRA. To evaluate the robustness of each strategy, we use a combination of POSIX, model-based scoring, and small-scale human judgments.

Inference-Time Techniques

These techniques modify the input prompt at inference time without updating model weights.

Chain-of-Thought (CoT) Appends "Let's think step by step" to the instruction to encourage intermediate reasoning. This often results in more deliberate and consistent outputs.

Self-Refinement (SR) Performs a two-step process: (1) the model rewrites the instruction for clarity, and (2) generates an answer to the rewritten prompt. This reduces phrasing sensitivity by aligning the surface form of inputs.

Self-Consistency (SC) Generates multiple answers using the CoT prompt and uses a meta-evaluator to select the best response. This addresses instability by aggregating over diverse generations.

Iterative Refinement (IR) The model first responds to the instruction. Then, in two additional rounds, it refines its own output using the conversation history. This process helps the model detect and correct earlier mistakes.

Fine-Tuning with Prompt Variants

To further reduce sensitivity, we fine-tune the base model using sets of semantically aligned prompts paired with a shared response. Each group includes perturbations such as paraphrasing, reordering, and noise injection. We apply parameter-efficient fine-tuning via Low-Rank Adaptation (LoRA) [10], injecting trainable rank-reduced matrices into attention and feedforward layers while keeping the rest of the model frozen. The dataset comprises 500 prompt groups annotated with perturbation types. All inputs are tokenized and padded to a fixed sequence length.

Models

We use four open-access language models across fine-tuning and inference. We selected these models because they were released before the evaluation datasets, ensuring that the datasets could not have been part of the models' training data, preventing data leakage and supports fair evaluation.

- **LLaMA 2 7B** [11]: A pretrained base model (not instruction-tuned), fine-tuned and used for inference.
- **Mistral 7B Instruct** [12]: Instruction-tuned; used for fine-tuning, inference, and automatic evaluation.
- **Falcon 7B Instruct** [13]: Instruction-tuned; used for fine-tuning and inference.
- **Falcon 1B Instruct** [13]: Instruction-tuned; used only for fine-tuning.

Evaluation Strategies

We evaluate model robustness to prompt perturbations using two complementary strategies: POSIX and LLM-as-a-Judge based scoring. We also include a small-scale human judgment study as a sanity check.

POSIX Metric

We adopt the Prompt Sensitivity Index (POSIX) [3] which quantifies how much a model's response likelihood changes when the prompt is rephrased. A lower POSIX indicates higher robustness. Full formulation is provided in Appendix C .

LLM-as-a-Judge Scoring

To complement POSIX, which evaluates likelihood sensitivity, we use Mistral-7B-Instruct as an automatic judge to assess surface-level similarity between outputs. Given a reference and a model output, it assigns a similarity score from 0.0 to 1.0. This helps quantify semantic consistency under prompt variation without relying on token-level likelihoods. We selected Mistral-7B-Instruct for its strong instruction-following capabilities and low latency. The prompt template is provided in Appendix D.

Human Evaluation

To assess real-world performance, we conducted a small-scale human evaluation of 19 model–strategy pairs. Each was tested on approximately five perturbed prompts, drawn from five core questions with a mix of spelling errors, paraphrasing, template shifts, and noise injection. This yielded around 100 outputs. Three annotators independently rated subsets of the responses to ensure balanced coverage and manageable workload. Each response was scored from 1 to 5 on four axes: Correctness (factual accuracy and relevance), Coherence (fluency and structure), Robustness (stability across prompt variants), and No Hallucination (absence of fabricated content).

Results

POSIX Scores

We computed POSIX scores for all models and prompting/fine-tuning methods. To analyze performance in detail, scores were broken down by perturbation type. We report both a comprehensive global POSIX score (across all perturbation types) and a restricted score that excludes noise perturbations, following [3]. Below are the key observations from the POSIX sensitivity scores reported in Table 1:

Table 1. Performance of prompting techniques across models and perturbation types. Green values indicate the lowest (best) global scores for each model, representing reduced sensitivity. Red values mark the highest (worst) scores. Bolded values highlight the overall best and worst global scores across all models.

Model	**Technique**	Global	Global (NoN)	Noise Injection	Para-phrase	Spelling Error	Template Shift
Falcon 7B	CoT	3.95	3.65	3.19	3.69	3.84	3.41
	IR	4.42	3.98	3.39	4.09	4.07	3.81
	NA	4.47	3.85	3.27	3.91	3.97	3.66
	SC	2.76	2.78	2.97	2.64	2.89	2.64
	SR	4.26	4.29	5.24	5.55	4.68	5.52
Falcon 7B (FT)	CoT	6.34	6.85	6.34	7.24	6.75	6.55
	IR	6.28	6.96	6.28	7.72	6.42	6.73
	NA	4.77	4.61	3.14	5.73	4.62	3.47
	SC	3.58	4.11	4.74	4.27	4.01	4.04
	SR	**9.54**	6.60	9.90	8.40	6.60	10.60
Falcon 1B (FT)	CoT	4.91	5.28	6.20	5.40	5.29	5.17
	IR	4.22	4.70	4.22	4.93	4.69	4.47
	NA	2.53	3.33	2.66	4.21	3.25	2.53
	SC	3.19	3.80	4.67	3.94	3.78	3.68
	SR	6.41	7.71	9.41	9.01	8.59	6.34
LLaMA	CoT	**1.26**	1.62	1.95	1.26	2.02	1.58
	IR	1.27	1.39	1.75	1.27	1.44	1.48
	NA	1.32	1.63	2.05	1.32	1.82	1.74
	SC	2.41	2.67	2.96	2.41	2.97	2.66
	SR	2.47	2.56	2.75	2.68	2.47	2.53
LLaMA (FT)	CoT	5.60	6.20	5.60	6.71	5.87	6.02
	IR	7.86	8.39	8.62	8.89	8.05	8.23
	NA	3.72	4.59	3.78	6.01	4.03	3.72
	SR	7.51	9.14	10.37	9.23	8.26	10.82
Mistral	CoT	2.21	2.53	2.48	2.81	2.55	2.21
	IR	3.67	4.03	3.82	4.40	3.98	3.67
	NA	3.01	3.41	3.42	3.74	3.46	3.01
	SC	4.87	5.24	5.67	5.58	5.29	4.87
	SR	2.96	2.64	2.86	2.77	2.92	2.10
Mistral (FT)	CoT	4.73	5.08	6.33	4.73	4.89	5.62
	IR	6.79	7.64	8.64	8.10	7.40	7.41
	NA	6.21	6.27	8.72	6.28	6.21	6.30
	SC	2.78	3.04	3.94	2.78	3.01	3.31
	SR	6.71	7.55	7.95	7.78	7.06	7.82

Techniques: CoT = Chain of Thought, IR = Iterative Refinement, NA = None, SC = Self-Consistency, SR = Self-Refinement.

- **LLaMA's Robustness:** The base LLaMA model consistently shows the lowest POSIX sensitivity across all prompting methods and perturbations, more stable than Falcon or Mistral and their fine-tuned (FT) counterparts.
- **Fine-Tuning Increases POSIX Score:** FT generally raises the score, particularly in LLaMA (FT) and Falcon 7B (FT). This is visible across perturbations and techniques, indicating that fine-tuning may amplify prompt sensitivity rather than reduce it.
- **Size vs. Sensitivity Trade-Off.** Comparing Falcon 1B and 7B (base), the larger 7B has slightly lower global sensitivity (3.95 vs. 4.91). Yet after FT, 7B's sensitivity jumps much more (to 6.34) than 1B's (to 4.91), suggesting that larger models may be more vulnerable to over-specialization during fine-tuning.
- **Self-Refinement (SR) Often Harmful:** SR yields some of the highest sensitivity scores in Falcon and LLaMA FT models (e.g., 9.54 for Falcon 7B FT and 10.37 for LLaMA FT). While designed to refine outputs, SR appears to make models more fragile to input variation.
- **Technique Performance Varies by Model:** Techniques like CoT, IR, and NA perform well on LLaMA base but poorly on other models. For instance, CoT yields only 1.26 sensitivity on LLaMA but over 6.0 on Falcon 7B (FT), highlighting that technique efficacy is model-dependent.
- **Perturbation Patterns:** While no perturbation is uniformly worst, 'Template Shift' often spikes sensitivity, especially with SR and FT models. Some exceptions exist: base Falcon 7B shows relatively low values under 'Noise Injection' with NA and SC.

LLM-as-a-Judge Scores

Table 2 presents the mean similarity scores across models, techniques, and perturbation types. Below are key observations from the LLM-as-a-Judge scores:

Table 2. Mean similarity scores from the judge model for each model, technique, and perturbation.

Model	**Technique**	Original	Spelling Error	Template Shift	Noise Injection	Paraphrase
Llama-2-7b	CoT	0.39	0.33	0.38	0.39	0.44
Falcon-7b	CoT	0.64	0.58	0.64	0.58	0.63
Llama-2-7b	IR	0.43	0.42	0.45	0.48	0.48
Falcon-7b	IR	0.61	0.58	0.60	0.57	0.62
Llama-2-7b	SC	0.48	0.42	0.45	0.40	0.46
Falcon-7b	SC	0.63	0.60	0.61	0.59	0.62
Llama-2-7b	NA	0.65	0.60	0.64	0.60	0.65
Falcon-7b	NA	0.67	0.62	0.65	0.61	0.66
Llama-2-7b	SR	—	0.36	0.73	—	—
Falcon-7b	SR	0.69	0.73	0.71	0.72	0.83

- **Model Comparison:** Falcon-7b consistently achieved higher mean scores than Llama-2-7b across all techniques, indicating stronger alignment with the gold references according to the judge model.
- **Technique Trends:** For LLaMA, SR had limited data but showed high performance on template shift (0.72). Falcon-7b exhibited strong performance in SR across all perturbations.

- **Perturbation Impact:** Spelling errors and noise injection posed consistent challenges for both models, with lower scores across all techniques. Template shifts showed moderate difficulty, while original and paraphrase cases typically yielded higher similarity scores.
- **Model Trends:** The judge-based scoring demonstrated Falcon-7b's relative robustness across perturbations and techniques, while highlighting areas for potential improvement in LLaMA, particularly under perturbations like spelling error and noise injection.

Human Evaluation
Figure 1 shows average human ratings across systems. Amongst the prompts examined, Mistral with self-refinement, Falcon 1B fine-tuned, and Mistral with iterative refinement performed best, producing fluent and consistent outputs. Mistral SR converged well across variants, and Falcon 1B FT handled rephrasings with minimal drift. Mid-tier systems (Falcon IR, Mistral (no prompting), and LLaMA 2 FT) were less stable, with occasional overgeneration, input sensitivity, or misinterpretation. LLaMA base and LLaMA SR performed worst, often repeating inputs, refining prompts instead of answers, or failing under perturbation. Fine-tuned models hallucinated more under paraphrasing and spelling errors, often changing the question itself. However, they remained relatively robust under template shifts and noise injection.

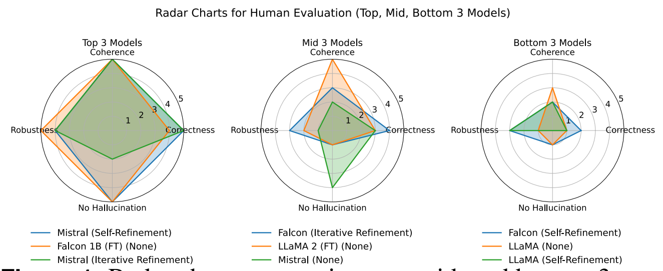

Figure 1. Radar charts comparing top, mid, and bottom 3 models based on human evaluation scores.

Discussion

Beyond POSIX: The Limits of Sensitivity Metrics
Our findings reveal that POSIX offers valuable but incomplete insight into robustness. Critically, POSIX does *not* evaluate answer quality. For instance, LLaMA-2-7B's low POSIX scores reflect low token-level confidence and verbose, hallucinated outputs. Such models appear "stable" not because they are robust, but because they are uncertain.

Moreover, POSIX can be inflated by fine-tuning and instruction tuning, which is because such models answer with higher confidence. Thus, inconsistency in high-confidence answers can reflect a poor POSIX, which may not be fairly evaluated against the less-confident answers. POSIX also overlooks meaning: semantically divergent responses can score low if confidence is stable, and vice versa. These caveats highlight the risk of over-relying on POSIX to rank models.

Despite these limitations, we found partial alignment between POSIX and human feedback. For example, SR improved Mistral's consistency and output quality, reflected in both reduced POSIX sensitivity and higher human ratings. In contrast, SR failed for Falcon-7B, producing repetitive or incomplete answers, which is a breakdown captured by both metrics.

LLM-as-a-Judge: Strengths and Shortcomings
LLM-as-a-Judge approach yielded informative trends but also surfaced critical pitfalls. First, we observed a notable mismatch with human judgments. Falcon-7B with SR received high judge scores despite producing empty or trivial outputs. This inflation likely stems from the judge over-rewarding short, repeated, or structurally similar responses, regardless of substance.

Second, identical generations (especially under SR) were often rated as "high similarity" by the judge, despite lacking semantic richness. This exposes a key limitation: overlap-based metrics may conflate repetition with correctness.

Third, techniques like CoT and IR scored moderately on LLaMA but higher on Falcon, mirroring human feedback that Falcon produced more verbose, if sometimes off-target, outputs. Yet even when models hallucinated or misunderstood the prompt, the judge often failed to penalize them due to surface-level similarity. Perturbation types also played a role: spelling errors and noise injection consistently reduced judge scores and were flagged by humans as destabilizing. LLaMA, for instance, often repeated prompts when perturbed—an issue missed by POSIX but captured by the judge. Interestingly, SR improved Mistral's robustness across all metrics, while for Falcon it led to high judge scores despite low semantic quality, underscoring again the need for human oversight.

Conclusion
Our analysis shows that prompt sensitivity varies widely by model and technique. Fine-tuning does not guarantee robustness, and prompting strategies like SR may succeed in one model but fail in another. This variability underscores that no single metric suffices. Robustness must be evaluated through a multi-faceted lens: POSIX captures confidence stability, judge-based scores capture semantic overlap, and human evaluations capture nuance and factuality. Only by triangulating these perspectives can we make meaningful claims about a model's real-world reliability. Future work should explore hybrid scoring pipelines that incorporate content-aware judges or adversarial perturbation detection, and better couple robustness with answer quality. Ultimately, robustness should not just mean consistency—it should mean consistent correctness under variation.

References
[1] Moran Mizrahi, Guy Kaplan, Dan Malkin, Rotem Dror, Dafna Shahaf, and Gabriel Stanovsky. State of what art?

a call for multi-prompt LLM evaluation. *Transactions of the Association for Computational Linguistics*, 12:933–949, 2024.

[2] Bowen Cao, Deng Cai, Zhisong Zhang, Yuexian Zou, and Wai Lam. On the worst prompt performance of large language models, 2024.

[3] Anwoy Chatterjee, H S V N S Kowndinya Renduchintala, Sumit Bhatia, and Tanmoy Chakraborty. POSIX: A prompt sensitivity index for large language models. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 14550–14565, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

[4] Jingming Zhuo, Songyang Zhang, Xinyu Fang, Haodong Duan, Dahua Lin, and Kai Chen. Prosa: Assessing and understanding the prompt sensitivity of llms. *arXiv preprint arXiv:2410.12405*, 2024.

[5] Sheng Lu, Hendrik Schuff, and Iryna Gurevych. How are prompts different in terms of sensitivity?, 2024.

[6] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models. https://crfm. stanford. edu/2023/03/13/alpaca. html*, 3(6):7, 2023.

[7] Karan Goel, Nazneen Rajani, Jesse Vig, Samson Tan, Jason Wu, Stephan Zheng, Caiming Xiong, Mohit Bansal, and Christopher Ré. Robustness gym: Unifying the nlp evaluation landscape. *arXiv preprint arXiv:2101.04840*, 2021.

[8] Zhipeng Wei, Yuqi Liu, and N Benjamin Erichson. Emoji attack: A method for misleading judge llms in safety risk detection. *arXiv preprint arXiv:2411.01077*, 2024.

[9] Yangshijie Zhang. Emoti-attack: Zero-perturbation adversarial attacks on nlp systems via emoji sequences. *arXiv preprint arXiv:2502.17392*, 2025.

[10] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021.

[11] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[12] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023.

[13] Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Merouane Debbah, Etienne Goffinet, Daniel Heslow, Julien Launay, Quentin Malartic, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. Falcon-40B: an open large language model with state-of-the-art performance. 2023.

Appendix A: Perturbation Types

We group prompt variations into four categories that introduce surface-level changes while preserving semantic intent. Each type is illustrated below using stylized examples.

Paraphrase

Prompts are reworded using alternative phrasing or synonyms without changing their meaning.

```
Q: Explain how photosynthesis works.
A:
Q: Describe the process by which
plants convert light into energy.  A:
```

Template Shift

Prompt structure or formatting is altered. Layout or section labels may vary.

```
Q: Describe the role of mitochondria.
A:
Question || Describe the role of
mitochondria.  Answer||
```

Spelling Error

Minor spelling mistakes or typos are introduced, simulating user error.

```
Q: What is the capital of France?  A:
Q: What s the capittal of Frnace?  A:
```

Noise Injection

Character-level noise such as symbols, emojis, or spacing artifacts is added.

```
Q: How do airplanes fly?
Q: How do air#planes fly?  hjbk A:
```

Appendix B: Prompt Templates

Chain-of-Thought Prompt

This prompt encourages the model to reason step-by-step before producing a final answer. It is appended to the default instruction format to guide structured thinking.

```
Your role is to meticulously
execute instructions and provide
concise, relevant output.
[I] Let's think step by step.
```

Self-Refinement Prompts

This two-step method begins by prompting the model to rewrite the instruction for clarity, followed by generating an answer to the rewritten version.

Prompt Rewriting Stage

```
Your role is to rewrite prompts
concisely for clarity.
Rewrite this prompt for clarity:
[I]
Rewritten prompt:
```

Answer Generation Stage

```
Your role is to meticulously
execute instructions and provide
concise, relevant output.
[I_rewritten]
```

Self-Consistency Prompt

This method prompts the model to generate multiple candidate answers and then evaluate them to select the best one.

Meta-Evaluation Prompt

```
You are a meta-evaluator.
Evaluate the following candidates
and select the best answer for the
question:
[I]
Candidates:
1.  [A1]
2.  [A2]
3.  [A3]
Your choice:
```

Iterative Refinement Prompt

The model iteratively improves its response by reflecting on the conversation history across multiple turns.

Refinement Turn Prompt

```
Your role is to meticulously
refine previous answers based on
the conversation history.
[Conversation History (User and
Assistant turns)]
Refinement Request:  Please refine
the last assistant answer to be
more precise.
Refined Answer:
```

Appendix C: Prompt Sensitivity Index (POSIX)

The Prompt Sensitivity Index (POSIX) [3] quantifies the variability in response likelihoods under different, semantically equivalent prompts. Given a model $\mathscr{M}$ and a prompt set $\mathscr{X} = \{x_1, x_2, \ldots, x_N\}$ with aligned intent and corresponding outputs $\{y_1, y_2, \ldots, y_N\}$, the sensitivity score is defined as:

$$\psi_{\mathscr{M},\S} = \frac{1}{N(N-1)} \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{1}{L_{y_j}} \left| \log \frac{\mathbb{P}_{\mathscr{M}}(y_j \mid x_i)}{\mathbb{P}_{\mathscr{M}}(y_j \mid x_j)} \right|$$

where L_{y_j} is the number of tokens in y_j and $\mathbb{P}_{\mathscr{M}}(y_j \mid x)$ is the likelihood of output y_j given prompt x.

The overall POSIX score for dataset $\mathscr{D} = \{\mathscr{X}_1, \ldots, \mathscr{X}_M\}$ is the average over all sets:

$$\text{POSIX}_{\mathscr{D},\mathscr{M}} = \frac{1}{M} \sum_{i=1}^{M} \psi_{\mathscr{M},\mathscr{X}_i}$$

This measures how much the model's confidence in its own answer changes when the prompt is replaced with another semantically aligned variant.

Appendix D: LLM-as-a-Judge Prompt

In addition to computing POSIX scores, we used a large language model (LLM) based evaluation approach, where Mistral-7B-Instruct served as an automatic judge to assess semantic similarity between gold references and candidate outputs. This method complements the POSIX metric by directly measuring semantic alignment rather than relying on token likelihoods.

Similarity Judging For each prompt-response pair, we constructed a prompt instructing the judge to compare the gold reference and candidate answer, and to rate their similarity on a scale from 0.0 (completely dissimilar) to 1.0 (perfectly identical). This numeric score was then extracted from the judge's output.

Prompt Template We used the following prompt structure to elicit similarity scores from the judge model:

```
Compare the following two texts
and rate their similarity on a
scale from 0.0 to 1.0.
Return only the number, as a
decimal with at least one digit
after the point.

Reference:
{gold}

Candidate:
{model output}

Similarity Score:
```

Usage and Output This prompt was incorporated into our evaluation pipeline using the `transformers` library. The Mistral-7B-Instruct model generated a similarity score for each pair of reference and candidate, which was parsed and saved in a CSV file. Each row of the CSV file contains the group ID, perturbation type, technique, model, and the assigned similarity score. This setup allowed us to compare performance across models and perturbation types in a consistent and automated fashion.Similarity scores closer to 1.0 indicate that the candidate answer closely matches the gold reference in meaning and phrasing, while scores near 0.0 indicate significant semantic drift. This judge-based scoring approach offers a complementary perspective to POSIX, focusing on content similarity rather than confidence shifts.