Stress Testing LLMs with Prompt Perturbations

Advait Singh, Ishan Kotwani, Prachi Garg
Birla Institute of Technology and Science, Pilani
{f20220636, f20220086, f20230548}@pilani.bits-pilani.ac.in

Abstract

Large Language Models (LLMs) have become foundational tools across natural language processing tasks, demonstrating remarkable performance in reasoning, question answering, and code understanding. However, despite their impressive capabilities, LLMs often display high sensitivity to minor changes in input phrasing—a phenomenon that challenges their reliability in real-world applications. Users rarely phrase prompts in a uniform manner; small variations such as typographical errors, use of synonyms, or reordered instructions can lead to significantly different outputs. This fragility raises critical concerns regarding the stability, fairness, and trustworthiness of LLM-based systems.

Recent research has highlighted this issue of prompt sensitivity as a growing area of investigation. Taking inspiration from recent work on evaluating LLM responses to various prompting techniques, we introduce a red-team/blue-team paradigm for systematically probing model behavior. While current research is largely restricted to handling instruction template perturbations, we shall attempt to focus evaluating natural prompts by simulating human behavior to judge variation in responses. Furthermore, we introduce quantitative metrics for evaluating variation in responses, employing modified variants of tools such as BERTScore to appropriately compare model outputs against a ground-truth reference.

1 Introduction

Large Language Models (LLMs) have become foundational components in a new generation of applications, from sophisticated chatbots to autonomous agents capable of complex tool use. As these models are increasingly integrated into high-stakes environments, their ability to deliver consistent and reliable outputs is of paramount importance. As we enter this new paradigm, it is essential to ensure that minor variations in prompt formatting or user errors do not lead to incorrect responses or cause important information to be omitted without the user's awareness.

Large Language Models operate using billions of parameters and are trained on diverse datasets encompassing text from books, articles, websites, and code. Notable models such as GPT-4, Claude 2, and Mistral 7B build on earlier architectures but have significantly improved reasoning, factual recall, and stylistic flexibility. The ability of these models to learn from few examples, a phenomenon called in-context learning, contributes to their versatility but also reveals a fundamental fragility to input perturbations. When exposed to crafted prompts, these models can be misled into misaligned behavior such as being jailbroken (Pathade, 2025). This input fragility is not limited to malicious attacks; LLMs often display high sensitivity to minor, benign changes in prompt phrasing—a phenomenon that challenges their reliability in real-world applications (He et al., 2024).

Among the most pressing of these systemic vulnerabilities is prompt sensitivity. Users rarely phrase prompts in a uniform manner; small variations such as typographical errors, the use of synonyms, reordered instructions, or subtle changes in context can lead to significantly different, and often incorrect, outputs. This variability raises critical concerns regarding the stability, fairness, and trustworthiness of LLM-based systems. The widespread documentation of this input-output inconsistency highlights an urgent need for systematic methods to evaluate and quantify the impact of natural prompt variations.

In response, a growing area of investigation focuses on evaluating LLM robustness. Recent research has explored the evaluation of LLM responses to a variety of prompting techniques and perturbations in instruction templates (Agrawal et al.,2025). Taking inspiration from these works, we introduce a red-team/blue-team paradigm for systematically probing model behavior. While current research is largely restricted to handling highly structured instruction template perturbations, our focus is on evaluating natural prompts by simulating typical human behavior to judge the resultant variation in responses.

To address this gap, we implemented a structured probing methodology to assess the stability of LLM outputs under realistic variations of user input. Our methodology involved curating a diverse dataset of natural prompt variations, representing real-world user interaction styles that range from simple rephrasing to structural changes. These prompts were tested against a range of proprietary and open-source LLMs.

The results of this simulation demonstrate a quantifiable relationship between input variation and output stability. Furthermore, we introduce novel quantitative metrics for evaluating variation in responses, employing modified variants of tools such as BERTScore to appropriately compare model outputs against a ground-truth reference. These findings highlight the critical importance of evaluating LLM performance not just on fixed test sets, but across a spectrum of possible human inputs to better understand and mitigate prompt sensitivity in practical applications.

2 Related Works

Prompt Sensitivity and Input Perturbations. The vulnerability of Large Language Models (LLMs) to minor changes in input, or prompt sensitivity, is a growing area of research that directly informs our work. This sensitivity manifests in several ways, often resulting in inconsistent or incorrect responses despite the query's underlying intent remaining the same. Early work on the **Primacy Effect of ChatGPT** demonstrated that the ordering of examples or instructions within a prompt can significantly bias the model's output, suggesting a high degree of dependence on the input's structural arrangement (Wang et al., 2023). This finding was reinforced by studies showing that the placement of information in long contexts—specifically, the phenomenon of being **Lost in the Middle** causes models to focus disproportionately on the beginning and end of the input, neglecting central information (Liu et al., 2023). Extending this, research has specifically addressed the **Order Sensitivity of In-Context Demonstration Examples** in causal language models, further highlighting how small structural choices can alter model behavior (Xiang et al., 2024). A more recent study explores the effect prompt formatting on LLM responses, quantifying the significant impact of variations in structure, style, and length on diverse tasks (He et al., 2024). Our work builds upon this by systematically probing the robustness of LLMs against natural, human-simulated prompt variations, rather than just instruction template perturbations.

Robustness and Adversarial Attacks on LLMs. A parallel line of research focuses on adversarial robustness, where inputs are intentionally manipulated to cause harmful misbehavior, serving as an extreme bound on input sensitivity. **PromptRobust** introduced one of the first comprehensive frameworks for evaluating the robustness of LLMs against adversarial prompts, providing a foundation for systematic testing of input stability (Zhu et al., 2023). While these attacks often involve carefully crafted, non-natural inputs (e.g., adversarial suffixes), they underscore the deep-seated input-output mapping fragility in LLMs. The techniques explored in this area, such as **Greedy Coordinate Gradient-Based Search** (Zou et al., 2023) and its successor **ADV-LLM** (Sun et al., 2024) which optimize adversarial token sequences for jailbreaking, demonstrate how minor token-level changes can completely subvert a model's safety alignment. This literature establishes that LLMs are not inherently stable and that their alignment can be easily compromised by input design.

Evaluation Methodologies and Metrics. Our methodology, which uses simulation of human behavior in prompts and quantitative comparison of responses, is informed by work on general NLP evaluation and text generation metrics. The concept of creating a rigorous evaluation environments for Natural Language Processing tasks was formalized in projects like Robustness Gym, which unifies the evaluation landscape for assessing model performance across various perturbations (Goel et al., 2021). Crucially, our work utilizes and adapts **BERTScore** for comparing model outputs, which provides a robust and context-aware metric for evaluating the semantic similarity of generated text against a ground-truth reference. BERTScore moves beyond traditional n-gram overlaps to use BERT embeddings for a more accurate assessment of qualitative output variation (Zhang et al., 2020). Additionally, while focused on improving reasoning, methods like **Self-Consistency** (Wang et al., 2022) where multiple diverse outputs are generated and a consensus is chosen implicitly acknowledge output variance, which our method seeks to explicitly measure and quantify under varying inputs. Our introduction of quantitative metrics for response variation provides a concrete way to assess the stability observed in previous qualitative studies.

3 Methodology

3.1 Design overview

To systematically assess the variation in prompt responses of Large Language Models (LLMs) to prompt perturbations, we construct an evaluation framework for evaluating the LLM's responses to perturbations in prompts

In this framework, we construct a dataset of adversarial prompts sourced from publicly available datasets on HuggingFace and GitHub. These prompts are passed through our perturbation pipeline, which consists of two variants. In the first variant, each prompt is transformed into five perturbed versions using WordBug and TextFooler, with the perturbation intensity controlled by gradually increasing the percentage of modified words and characters. The original and perturbed prompts are then evaluated using the target LLM to obtain corresponding response sets.

The second variant applies ten distinct perturbation techniques to each prompt in order to assess the effect of individual perturbation types. These techniques include: WordBug, TextFooler, WordBug + TextFooler, random phrase shuffling, stopword removal or insertion, casing perturbation, paraphrasing, punctuation injection, and homophone substitution.

Our evaluation incorporates five leading Large Language Models (LLMs): **Llama 3.2, Qwen 2, Gemma, Mistral, and Phi 3**. To assess robustness, we utilize two distinct variants of our specialized prompt dataset. Passing both variants through the five models generates a comprehensive evaluation corpus comprising 4,420 prompt-response pairs per LLM, resulting in a total dataset of **22,100** pairs.

These responses our then evaluated for perturbations by comparing against the ground truth. (i.e., the response generated by the LLM using the unperturbed prompt). For this task, we employ the **BERTScore-F1** metric (Zhang et al., 2020), which is the standard measure used to assess the semantic concordance between two text sequences.

3.2 Prompts Corpus

The prompt corpus is systematically structured as an evaluation dataset, incorporating the base prompt, the ground truth response, and the corresponding classified prompt-response pairs derived from various perturbation types.

3.2.1 WordBug

Inspired by WordBug(Yang et al., 2018), our implementation generates adversarial text by applying character-level noise. The function random_char_perturb randomly selects one of four operations: **deletion, adjacent character swapping, insertion, or replacement—to modify words.** These transformations introduce subtle, typo-like errors to a portion of the input text, the **apply_wordbug** function controls the intensity of this attack via the **perturb_ratio**, determining the proportion of words subjected to these character-level modifications. This allows us to apply WordBug in varying intensities to judge the allowed range of typos before the LLM responses show significant variation

3.2.2 TextFooler

This implementation employs a semantic-level perturbation strategy, analogous to the methodology in the TextFooler attack (Wallace et al., 2019), by substituting words with their synonyms. The **synonym** function retrieves potential synonyms (using WordNet) for a given word and randomly selects one as a replacement, provided it differs from the original. The **apply_textfooler** function tokenizes the prompt and replaces a specified number of words (**num_replacements**) with their synonyms allowing us to control the extent to which this perturbation is induced. This approach aims to preserve the prompt's overall meaning while attempting to investigate an LLM's response to synonyms.

3.2.3 Random Shuffling

The **perturb_phrase_shuffling** function creates a structural adversarial example by reordering components of the input text at the sentence or phrase level. This method primarily shuffles the order of parsed sentences. If only one sentence exists, it defaults to randomly reordering a block of words in the middle of the text. This perturbation targets the

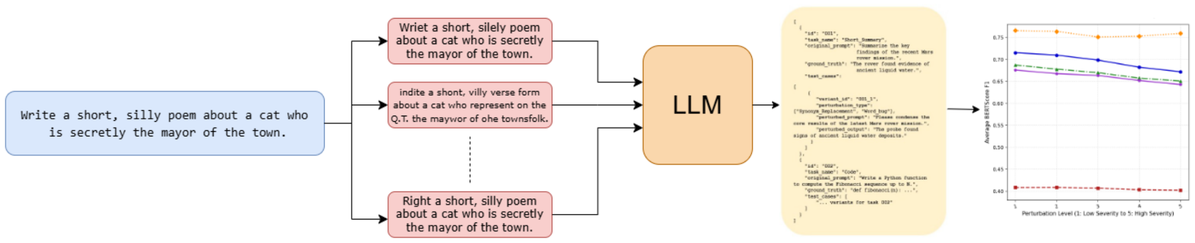

Write a short, silly poem about a cat who is secretly the mayor of the town.
Wriet a short, silely poem about a cat who is secretly the mayor of the town.
indite a shont, villy verse form about a cat who represent on the Q.T. the maywor of ohe townsfolk.
Right a short, silly poem about a cat who is secretly the mayor of the town.
LLM
Average BERTScore F1
Perturbation Level (1: Low Severity to 5: High Severity)

Figure 1: Illustration of the prompt perturbation and evaluation pipeline: multiple perturbed variants of a base prompt are passed to an LLM, and the resulting outputs are analyzed for semantic stability using BERTScore.

reliance of NLP models on sequential dependencies and fixed discourse structure, preserving the original vocabulary while radically altering the textual flow. This technique is used to evaluate model robustness against changes in grammatical or logical order, which is crucial for complex reasoning tasks.

3.2.4 Stopword Removal/Insertion

The **perturb_stopword** function implements a form of lexical perturbation by targeting common English stopwords (e.g., 'the', 'a', 'is'). The attack is executed in two phases: first, it removes a small, random subset of existing stopwords based on the **remove_ratio**; second, it inserts a fixed number **insert_count** of new, short stopwords at random locations throughout the text. This technique generates adversarial examples by slightly altering the frequency and distribution of high-frequency functional words. Since models often rely heavily on content words for semantic understanding, modifying stopwords tests the model's sensitivity to small, non-semantic changes in the text structure, particularly in contexts where precise grammatical features might be important.

3.2.5 Casing Perturbation

The **perturb_casing** function executes a visual-level adversarial perturbation by stochastically altering the capitalization patterns of words within the input prompt. Governed by the **word_casing_ratio**, the algorithm targets alphanumeric tokens to randomly transform them into UPPERCASE, Title Case, or lowercase formats. This method exploits the sensitivity of certain NLP models—particularly those trained on "cased" data—to orthographic variations that do not change the underlying semantic meaning but disrupt the expected visual features of the text. By injecting irregular casing, this perturbation simulates real-world noise such as informal social media text or unintended emphasis, testing a model's reliance on standardized formatting for token recognition and feature extraction.

3.3 Simulated Paraphrasing

The **perturb_paraphrase_simulated** function implements a high-intensity semantic-level perturbation by maximizing lexical substitution to mimic the effects of automated paraphrasing. Unlike lower-magnitude synonym swaps, this approach targets a significantly higher volume of tokens—up to 50% of the input text—replacing them with **WordNet-derived synonyms**. By extensively altering the prompt's lexical surface while attempting to preserve the underlying intent, this method evaluates the model's robustness against paraphrase variance. It tests whether a model is "overfit" to specific word choices or if it can maintain consistent performance when the same concept is expressed through a largely different vocabulary.

3.3.1 Punctuation Injection

The **perturb_punctuation_injection** function introduces non-alphanumeric noise into the input prompt to evaluate model resilience against character-level disturbances. By injecting a specified number of random symbols—ranging from mathematical operators to special characters—at random indices, this method simulates noisy real-world data such as encoding errors. This perturbation disrupts the tokenization process and positional embeddings of Transformer-based models, testing whether the presence of semantically null symbols biases the model's interpretation of the surrounding context.

3.3.2 Homophone Substitution

The **perturb_homophone** function implements a phonetic-level adversarial attack by substituting target words with their homophones or commonly confused near-homophones (e.g., swapping "their" for "there"). This rule-based approach utilizes a curated dictionary of linguistic "confused word" pairs to induce subtle errors that are often overlooked by human readers due to their identical auditory properties. From a computational perspective, these perturbations test the robustness of NLP models—particularly those utilizing sub-word tokenization like BPE. By maintaining the phonetic structure while altering the semantic meaning of individual tokens, this method evaluates whether a model relies on local features or deeper contextual understanding to disambiguate meaning.

3.3.3 Typographic Swap

The **perturb_typographic_swap** function implements a specialized character-level adversarial perturbation that simulates "fat finger" keyboard entry errors. Unlike purely random character substitutions, this method utilizes a mapping of physically adjacent keys on a standard QWERTY layout to replace a character with its neighboring counterpart. By specifically targeting the **KEYBOARD_ADJACENT** set, the algorithm mimics highly probable human typographic errors. This approach evaluates the robustness of NLP models—particularly their tokenizers and embedding layers—against realistic noise that preserves the approximate visual and structural integrity of the word while inducing a misaligned sub-word representation or an out-of-vocabulary ($[UNK]$) state.

4 Evaluation and Experimental Setup

4.1 Data Preparation and Perturbation Pipeline

We curated a specialized prompt corpus by aggregating samples from diverse open-source datasets hosted on **Hugging Face** and **GitHub**. A custom dataset builder was employed to standardize these prompts into the specific schema illustrated in Figure 1.

Our experimental methodology evaluates model robustness through three distinct testing protocols:

1. **Severity Scaling:** We generate five incremental levels of *WordBug* character-level perturbations. In the perturbation framework, severity is controlled by increasing the intensity of modifications applied to the input prompts. For WordBug, severity levels correspond to the proportion of words in the prompt that are randomly altered ranging from 5% to 25%.
2. **Semantic Scaling:** We apply five incremental levels of *TextFooler* synonym substitutions. For TextFooler, severity increases by the number of words replaced with synonyms, from 1 to 5 replacements per prompt.
3. **Algorithmic Variants:** We implement ten unique perturbation types to compare the effect of different perturbations on LLMs.

To manage the evaluation, these variants are structured into two datasets per model. The first dataset contains the severity-scaled variants (WordBug and TextFooler), while the second stores the ten distinct algorithmic perturbations. These base prompt versions are passed through the target models to generate corresponding response datasets, enabling a quantitative comparison between original and perturbed outputs.

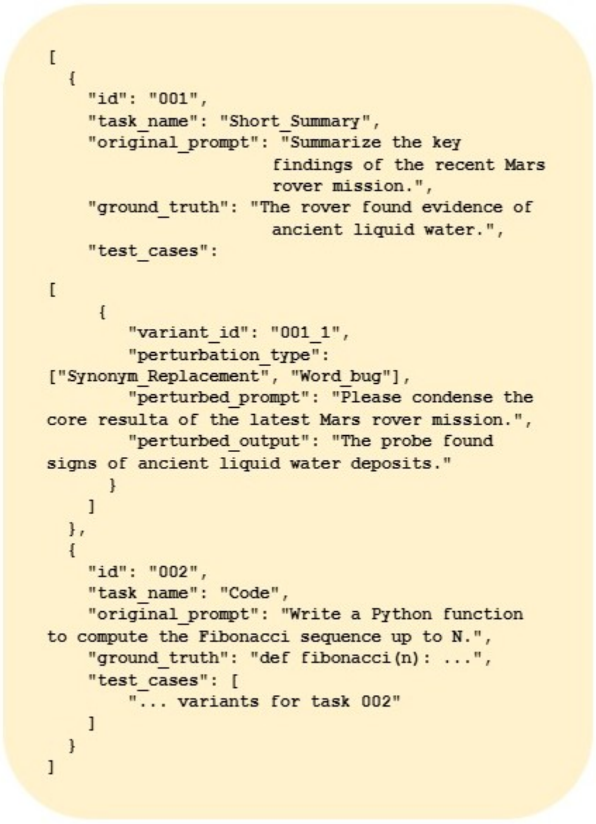
4.2 Experimental Setup

To evaluate the robustness of Large Language Models (LLMs) under prompt perturbations, we conduct a controlled experimental study using the curated prompt corpus described in Section 3.2. Each base prompt is evaluated in both its original (unperturbed) form and under multiple perturbed variants generated by our perturbation pipeline.

We evaluate five widely-used open-source LLMs: **Llama 3.2, Qwen 2, Gemma, Mistral, and Phi 3**. All models are queried independently using identical prompt variants to ensure comparability across architectures. For each model, the unperturbed prompt is first passed to the LLM to obtain a reference response, which serves as the **ground truth** for subsequent comparisons. The same model is then queried using each perturbed prompt variant, producing a corresponding perturbed response.

As mentioned, our experimental setup consists of two complementary evaluation variants. In the first variant, perturbation *intensity* is studied by applying WordBug and TextFooler at progressively increasing levels, allowing us to analyze how response stability degrades as the proportion of modified characters or words increases. In the second variant, individual perturbation *types* are evaluated independently by applying ten distinct perturbation techniques to each prompt, enabling a fine-grained comparison of their respective impact on model outputs.

Across both variants, each LLM is evaluated on a total of **4,420 prompt-response pairs**, resulting in an aggregate evaluation corpus of **22,100 prompt-response pairs**. This setup ensures consistent coverage across perturbation strategies and models while maintaining identical experimental conditions.



```
[
  {
    "id": "001",
    "task_name": "Short_Summary",
    "original_prompt": "Summarize the key
                        findings of the recent Mars
                        rover mission.",
    "ground_truth": "The rover found evidence of
                        ancient liquid water.",
    "test_cases":

[
      {
        "variant_id": "001_1",
        "perturbation_type":
["Synonym_Replacement", "Word_bug"],
        "perturbed_prompt": "Please condense the
core resulta of the latest Mars rover mission.",
        "perturbed_output": "The probe found
signs of ancient liquid water deposits."
      }
    ]
  },
  {
    "id": "002",
    "task_name": "Code",
    "original_prompt": "Write a Python function
to compute the Fibonacci sequence up to N.",
    "ground_truth": "def fibonacci(n): ...",
    "test_cases": [
        "... variants for task 002"
    ]
  }
]
```

Figure 2: Format of the dataset used for evaluation

4.3 Evaluation Metric

To quantitatively measure the variation between responses generated from unperturbed and perturbed prompts, we employ the **BERTScore-F1** metric (Zhang et al., 2020). BERTScore computes semantic similarity by aligning token-level contextual embeddings from a pretrained BERT model, providing a robust alternative to surface-level n-gram overlap metrics such as BLEU or ROUGE.

For each perturbed prompt-response pair, the perturbed response is treated as the **candidate**, while the response generated from the corresponding unperturbed prompt is treated as the **reference**. The resulting BERTScore-F1 value captures the degree of semantic concordance between the two responses, with lower scores indicating greater divergence induced by the prompt perturbation.

We report average BERTScore-F1 values aggregated by perturbation type and perturbation intensity to assess overall response stability. This evaluation framework enables a task-agnostic and model-independent comparison of robustness, focusing specifically on semantic variation rather than strict lexical equivalence.

4.4 Analysis Protocol

All evaluation results are computed independently for each model and perturbation category. Scores are averaged across prompts to mitigate the influence of individual outliers and to highlight systematic trends. No additional task-specific correctness labels or human annotations are used; instead, semantic similarity to the model's own unperturbed response serves as the primary robustness signal.

This protocol allows us to isolate the effect of prompt perturbations on model behavior while controlling for prompt content, model architecture, and evaluation metric. The resulting analysis provides a quantitative basis for comparing robustness across both perturbation strategies and LLMs.

5 Results

We present the results of our robustness evaluation across five Large Language Models (LLMs): **LLaMA 3.2, Qwen 2, Gemma, Mistral, and Phi-3**. Model robustness is quantified using average **BERTScore-F1** between responses generated from unperturbed prompts and their perturbed counterparts. Higher scores indicate greater semantic stability under prompt perturbations.

Our evaluation consists of two complementary analyses: (1) *severity-scaled perturbations* using incremental levels of WordBug and TextFooler, and (2) *algorithmic perturbation variants* spanning ten distinct lexical, syntactic, and semantic transformations.

5.1 Robustness under Severity-Scaled Perturbations

Tables 1 and 2 report BERTScore-F1 values under progressively increasing levels of TextFooler (TF1–TF5) and WordBug (WB1–WB5) perturbations, respectively.

Across all models, we observe a consistent degradation in semantic similarity as perturbation severity increases, indicating heightened response instability under more aggressive prompt modifications. This trend is particularly pronounced for TextFooler perturbations, where synonym substitutions increasingly alter the semantic surface of the prompt.

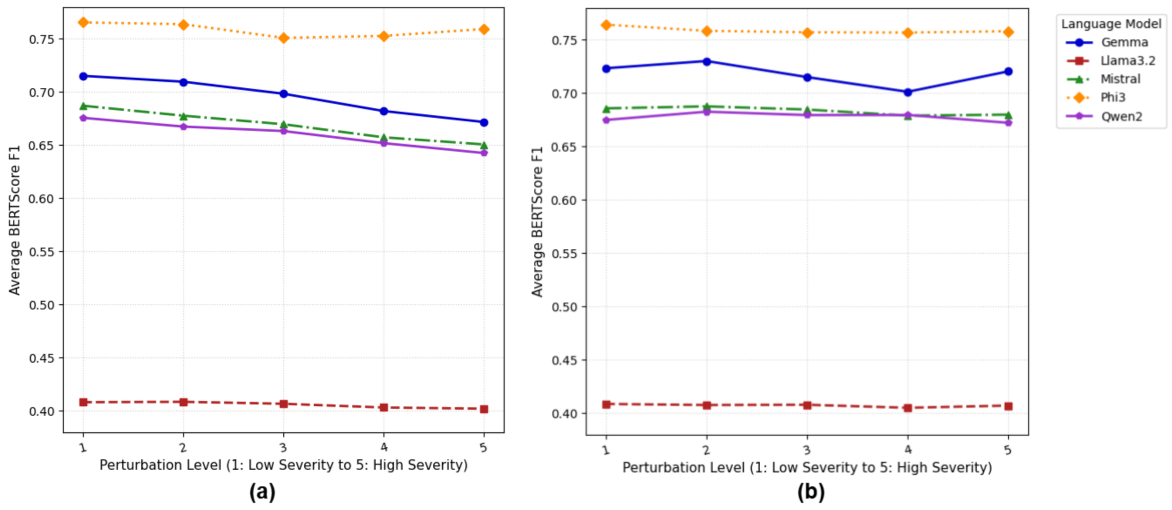


Figure 3: Graph (a) shows Model responses to elevating severity of TextFooler (Word level Perturbation) while graph (b) shows Model responses to WordBug (Character Level Perturbation)
These are the most common forms of typos even in well-engineered prompt and hence it necessitates study of the effect of varying severity of these perturbations on prompt responses

Among the evaluated models, **Phi-3** demonstrates the highest robustness under both perturbation families, maintaining BERTScore-F1 values above 0.75 even at higher perturbation levels. **Gemma** and **Mistral** exhibit moderate robustness, with gradual declines as perturbation intensity increases. In contrast, **LLaMA 3.2** shows substantially lower stability, with BERTScore-F1 values consistently near 0.40 across both WordBug and TextFooler variants, suggesting a heightened sensitivity to even minor lexical and character-level noise.

Interestingly, WordBug perturbations result in comparatively smaller drops in BERTScore-F1 than TextFooler for most models, implying that character-level noise such as typographical errors is less disruptive to semantic consistency than word-level synonym substitutions.

5.2 Impact of Algorithmic Perturbation Variants

Table 3 summarizes model robustness across ten distinct perturbation strategies, encompassing orthographic, lexical, structural, and semantic transformations.

Across all models, **simulated paraphrasing** and **combined WordBug+TextFooler attacks** consistently yield the lowest BERTScore-F1 values, indicating that large-scale lexical substitution and compound perturbations are the most disruptive to model outputs. This suggests that while models may tolerate isolated noise sources, compounded or high-coverage semantic changes substantially alter response behavior.

Conversely, perturbations such as **casing variation**, **punctuation injection**, and **homophone substitution** tend to preserve higher semantic similarity, reflecting greater robustness to surface-level and phonetic noise. This trend holds consistently across all evaluated LLMs.

Model-wise, **Phi-3** again demonstrates the strongest overall robustness, achieving the highest average BERTScore-F1 across all perturbation types. **Gemma**, **Mistral**, and **Qwen 2** form a middle tier with comparable stability profiles, while **LLaMA 3.2** exhibits the lowest resilience across nearly all perturbation categories.

5.3 Overall Trends

Aggregating results across both evaluation variants reveals a clear robustness hierarchy among the evaluated models, with Phi-3 consistently outperforming other LLMs under both severity-scaled and algorithmic perturbations. Our findings further indicate that semantic-level perturbations particularly paraphrasing and synonym substitution pose a significantly greater challenge to output stability than character-level noise or formatting changes.

These results underscore the importance of evaluating LLM robustness beyond fixed benchmarks,

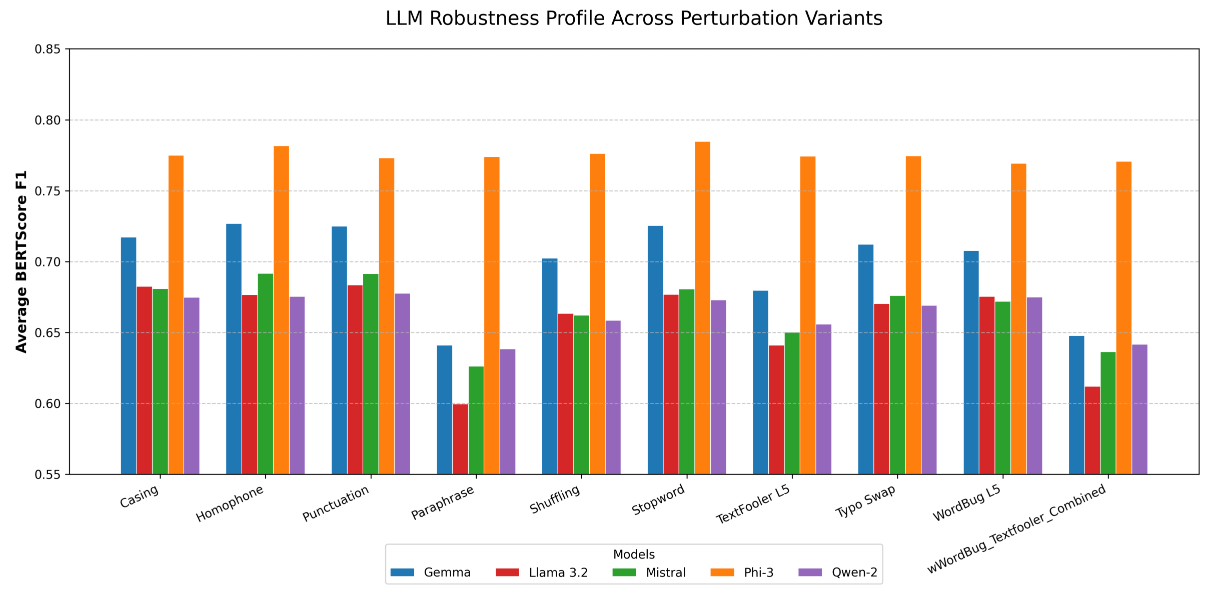


Figure 3: Average BERTScore F1 performance of five LLMs across various linguistic perturbation variants. Each bar represents the model's resilience to a specific attack type, with higher values indicating greater robustness.

highlighting how natural, human-like prompt variations can induce substantial and model-dependent response divergence. The observed disparities suggest that architectural choices, training data composition, and tokenization strategies play a critical role in determining model resilience to prompt perturbations.

6 Conclusion

In this work, we presented a systematic evaluation of the robustness of five large language models under a diverse set of natural and adversarial prompt perturbations. Using BERTScore-F1 as a measure of semantic consistency, we quantified how model responses change under both severity-scaled attacks (TextFooler and WordBug) and a broad range of algorithmic perturbation variants.

Our results demonstrate clear and consistent differences in robustness across models. Phi-3 exhibits the strongest overall resilience, maintaining high semantic similarity across both character-level and word-level perturbations. Gemma, Mistral, and Qwen 2 show moderate robustness, with gradual degradation as perturbation severity increases. In contrast, LLaMA 3.2 displays substantially lower stability across all evaluated settings, indicating heightened sensitivity to even minor prompt variations.

Across perturbation types, we observe that character-level noise and surface formatting changes, such as casing variation and punctuation injection, tend to preserve semantic consistency more effectively than semantic-level perturbations. In particular, simulated paraphrasing and combined WordBug–TextFooler attacks consistently induce the largest drops in BERTScore-F1 across all models, highlighting the vulnerability of current LLMs to high-coverage lexical and semantic transformations.

Taken together, these findings underscore the importance of evaluating robustness under realistic prompt variability rather than relying solely on fixed or clean benchmarks. The observed disparities suggest that architectural design choices, training data diversity, and tokenization strategies play a significant role in determining model resilience to perturbations. Future work may explore targeted robustness training, perturbation-aware evaluation metrics, and broader model families to better understand and improve the stability of large language models under real-world usage conditions.

The source code is available at `https://github.com/Testady21/Prompt-Perturbation-Simulator`.

7 Future Works

Building on the findings of this study, a natural extension of our work lies in the development and evaluation of a layered defense pipeline designed to improve robustness to prompt perturbations. Inspired by the initial proposition in our original framework, this pipeline would incorporate two complementary strategies:

1. **Self-Denoised Smoothing:** Each perturbed prompt would first be passed through a denoising stage, where the model attempts to normalize noisy or inconsistent inputs before generating a response. This mechanism could reduce sensitivity to character-level, lexical, and structural perturbations by providing a cleaner representation of the user intent.

2. **Consensus-based Aggregation:** After generating responses from multiple denoised or perturbed prompt variants, a consensus mechanism such as majority voting or self-consistency could be applied to produce a final robust output. This approach leverages redundancy across perturbation variants to mitigate the impact of individual perturbations that might mislead the model.

Future research could explore several directions within this defense pipeline: evaluating its effectiveness across different LLM architectures, determining the optimal balance between denoising strength and response fidelity, and investigating adaptive aggregation strategies that weight model outputs based on their estimated reliability. Additionally, integrating the pipeline with automated adversarial prompt generation frameworks may provide a continuous feedback loop for stress testing and improving model resilience. Overall, this approach represents a promising avenue for enhancing the stability and trustworthiness of LLM outputs under realistic, noisy user inputs.

References

Z. Yang, D. Yang, Y. He, J. Gao, L. Deng, and A. J. Smola. Black-box Generation of Adversarial Text Sequences to Evade Deep Learning Classifiers. *arXiv preprint arXiv:1801.04354*, 2018.

A. Agrawal et al. Enhancing LLM Robustness to Perturbed Instructions: An Empirical Study. In *Proceedings of the International Conference on Learning Representations (ICLR)*. 2025.

J. He, M. Rungta, D. Koleczek, A. Sekhon, F. X. Wang, and S. Hasan. Does Prompt Formatting Have Any Impact on LLM Performance?. In *arXiv preprint arXiv:2411.10541*, 2024.

N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang. Lost in the Middle: How Language Models Use Long Contexts. In *Transactions of the Association for Computational Linguistics (TACL)*. 2023.

Y. Wang, Y. Cai, M. Chen, Y. Liang, and B. Hooi. Primacy Effect of ChatGPT. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2023.

X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou. SELF-CONSISTENCY IMPROVES CHAIN OF THOUGHT REASONING IN LANGUAGE MODELS. In *Proceedings of the International Conference on Learning Representations (ICLR)*. arXiv preprint arXiv:2203.11171, 2022.

Y. Xiang, H. Yan, L. Gui, and Y. He. Addressing Order Sensitivity of In-Context Demonstration Examples in Causal Language Models. In *arXiv preprint arXiv:2402.15637*, 2024.

K. Zhu, J. Wang, J. Zhou, Z. Wang, H. Chen, Y. Wang, L. Yang, W. Ye, N. Z. Gong, Y. Zhang, and X. Xie. PromptRobust: Towards Evaluating the Robustness of Large Language Models on Adversarial Prompts. In *arXiv preprint arXiv:2306.04528*, 2023.

K. Goel, N. Rajani, J. Vig, S. Tan, J. Wu, S. Zheng, C. Xiong, M. Bansal, and C. Ré. Robustness Gym: Unifying the NLP Evaluation Landscape. In *arXiv preprint arXiv:2101.04840*, 2021.

T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi. BERTScore: Evaluating Text Generation with BERT. In *Proceedings of the International Conference on Learning Representations (ICLR)*. arXiv preprint arXiv:1904.09675, 2020.

C. Pathade. Red Teaming the Mind of the Machine: A Systematic Evaluation of Prompt Injection and Jailbreak Vulnerabilities in LLMs. *arXiv preprint arXiv:2505.04806*, 2025.

E. Wallace, J. Kaufmann, A. Clark, N. Goyal, and P. Liang. Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment. *arXiv preprint arXiv:1907.11932*, 2019.

A Appendix

A.1 BERTScore F1 under TextFooler perturbations (TF1–TF5)

Table 1

Model	**TF1**	**TF2**	**TF3**	**TF4**	**TF5**
Gemma	0.7153	0.7098	0.6984	0.6822	0.6718
LLaMA 3.2	0.4083	0.4086	0.4068	0.4033	0.4022
Mistral	0.6871	0.6778	0.6697	0.6574	0.6508
Phi-3	0.7655	0.7638	0.7510	0.7528	0.7593
Qwen2	0.6757	0.6675	0.6633	0.6520	0.6427

A.2 BERTScore F1 under WordBug perturbations (WB1–WB5).

Table 2

Model	**WB1**	**WB2**	**WB3**	**WB4**	**WB5**
Gemma	0.7235	0.7302	0.7152	0.7014	0.7205
LLaMA 3.2	0.4088	0.4078	0.4081	0.4052	0.4073
Mistral	0.6859	0.6877	0.6847	0.6790	0.6800
Phi-3	0.7643	0.7585	0.7572	0.7569	0.7581
Qwen2	0.6750	0.6827	0.6796	0.6797	0.6723

A.3 BERTScore F1 across semantic and lexical perturbation variants.

Table 3

Model	**Casing**	**Homophone**	**Noise+Punct**	**Paraphrase**	**Phrase**	**Stopword**	**TF-L5**	**Typo**	**WB-L5**	**WB+TF**
Gemma	0.7172	0.7268	0.7250	0.6410	0.7025	0.7254	0.6796	0.7121	0.7078	0.6479
LLaMA 3.2	0.6825	0.6768	0.6837	0.5997	0.6634	0.6770	0.6412	0.6704	0.6756	0.6121
Mistral	0.6810	0.6916	0.6914	0.6262	0.6622	0.6808	0.6504	0.6760	0.6721	0.6366
Phi-3	0.7749	0.7816	0.7731	0.7739	0.7762	0.7846	0.7744	0.7746	0.7693	0.7706
Qwen2	0.6747	0.6755	0.6778	0.6385	0.6586	0.6730	0.6560	0.6692	0.6750	0.6417