Quirx: A Mutation-Based Framework for Evaluating Prompt Robustness in LLM-based Software

Souhaila Serbout
University of Zurich
souhaila.serbout@uzh.ch

***Abstract*—Large Language Models (LLMs) increasingly power critical business processes, yet prompt robustness remains under-explored. Small variations—such as synonym changes or instruction reordering—can cause significant output shifts, undermining reliability in domains like customer service and finance. Existing evaluations rely on ad-hoc manual testing, limiting scalability in production environments.**

We present *Quirx*, a mutation-based fuzzing framework for systematically evaluating prompt robustness across LLM providers. *Quirx* applies tri-dimensional mutations (lexical, semantic, structural), executes them against target models, and measures response consistency via multi-level similarity analysis. It produces robustness scores, reveals failure patterns, and supports informed model selection.

We evaluate *Quirx* on four models (GPT-3.5-turbo, GPT-4o-mini, Claude-3.5-Sonnet, Claude-Sonnet-4) across three tasks. Results show sentiment classification is uniformly robust (1.00), summarization is highly provider-sensitive (0.23–0.58) with Claude models 2.5× more robust than OpenAI, and SQL generation is consistently strong (0.80–1.00). Structural mutations cause 50–67% of summarization failures but have minimal effect on other tasks.

Demo video: https://youtu.be/Sm3Gk2X2-vk

***Index Terms*—Large Language Models, Prompt Engineering, Software Testing, Fuzzing, Robustness Evaluation**

I. Introduction

The integration of Large Language Models (LLMs) into production systems has transformed natural language processing but exposed a critical reliability challenge: prompt brittleness [1, 9]. Unlike traditional software with predictable APIs, LLM applications depend on natural language prompts that exhibit unpredictable sensitivity to minor variations.

Consider real deployment scenarios: A customer service chatbot responds differently to “How do I cancel my subscription?” versus “Cancel my subscription how?”, potentially escalating or resolving issues inconsistently. Financial analysis systems may generate contradictory investment recommendations when prompts change from “Analyze Q3 revenue trends” to “Q3 revenue trends analysis”. Text summarization tools may emphasize different key points when instructions are reordered, affecting content extraction. In healthcare, diagnostic assistants might focus on different symptoms when instruction order varies, affecting clinical decision support.

This brittleness manifests beyond surface-level typos. Synonym substitutions (“customers” $\rightarrow$ “clients”), instruction reordering, and punctuation changes can trigger fundamentally different reasoning paths in LLMs. A SQL generation prompt may produce syntactically different queries with altered JOIN conditions when instructions are restructured, while summarization prompts may highlight different aspects of the same content based on minor phrasing changes, despite identical semantic intent. Current evaluation approaches are predominantly manual and insufficient for production-scale deployment [3, 4].

Quirx addresses this challenge through systematic mutation-based testing that automatically generates diverse prompt variations across lexical, semantic, and structural dimensions. The framework tests each mutation against multiple LLM providers, quantifies response consistency using multi-dimensional similarity analysis, and produces actionable robustness reports. This enables developers to identify prompt vulnerabilities before deployment and make evidence-based decisions about model selection and prompt engineering.

Our contributions include: (1) a tri-dimensional mutation engine, (2) multi-provider LLM integration, (3) multi-dimensional similarity analysis [6], (4) systematic robustness quantification, and (5) empirical evaluation providing insights into prompt reliability patterns.

Quirx is open-source and available at https://github.com/souhailaS/Quirx. In addition to the code, the repository contains overall documentation, usage examples, and the raw evaluation results. All experimental configurations and prompts are reproducible using the evaluation scripts and prompt files available in the *Quirx* repository.

II. Related Work

Recent work in prompt engineering has focused primarily on optimization techniques for improving task performance [1], with limited attention to robustness evaluation. Wei et al. [8] introduced chain-of-thought prompting, while Reynolds et al. [7] explored prompt programming paradigms. However, these approaches do not address the fundamental challenge of prompt brittleness.

In the software testing domain, fuzzing techniques have proven effective for finding vulnerabilities and edge cases [2]. Recent adaptations to NLP include semantic-preserving transformations for adversarial testing [5], but existing tools focus on model robustness rather than prompt engineering scenarios. *Quirx* bridges this gap by providing systematic prompt-level fuzzing specifically designed for production LLM-based applications.

III. System Design

A. Architecture Overview

Quirx implements a four-tier modular architecture designed for extensibility and reproducibility across diverse LLM providers and evaluation scenarios. Fig. 1 illustrates an overview of the system architecture and data flow.

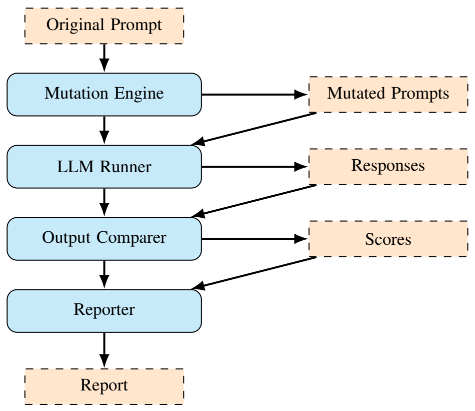

Fig. 1. Quirx System Architecture

The system follows a pipeline architecture where each component processes data sequentially while maintaining independence, enabling parallel processing and modular replacement of individual components without affecting system integrity. The *Mutation Engine* generates systematic prompt variations, the *LLM Runner* provides unified API interfaces across providers, the *Output Comparer* performs multi-dimensional similarity analysis, and the *Reporter* produces detailed robustness documentation.

B. Mutation Engine

The mutation engine implements a stratified sampling approach across three orthogonal dimensions derived from software testing mutation operators [2] and adversarial NLP techniques [5]: lexical (case changes, punctuation, whitespace), semantic (synonyms, paraphrasing), and structural (sentence reordering, instruction reorganization), ensuring systematic coverage of potential prompt variations encountered in production environments.

This tri-dimensional approach captures real-world prompt variations: accidental typos (lexical), alternative phrasings (semantic), and instruction reordering (structural). (1) *Lexical Mutations* apply surface-level changes (case, punctuation, whitespace) with severity scores 0.1-0.4. (2) *Semantic Mutations* use WordNet-based synonym substitution and paraphrasing with scores 0.3-0.5. (3) *Structural Mutations* reorder sentences and instructions with highest severity scores 0.5-0.7.

For example, “Generate SQL for customers” becomes: Lexical: “GENERATE sql FOR customers”, Semantic: “Create query for clients”, Structural: “For customers, generate SQL”.

C. Multi-Provider LLM Integration

The LLM Runner abstracts provider APIs (OpenAI GPT, Anthropic Claude, mock providers) with unified interfaces, handling authentication, rate limiting, error recovery, and response normalization. The system includes retry mechanisms, error logging, and detailed performance tracking.

D. Multi-Dimensional Output Analysis

The Output Comparer implements three-tier similarity analysis: **Token-Level** using Ratcliff-Obershelp sequence matching, **Semantic** using Sentence-BERT embeddings with cosine similarity, and **Structural** analyzing format preservation and logical organization. Similarity thresholds were empirically determined through iterative testing. The framework uses adaptive thresholds (0.95 for classification, 0.85 for generation) based on task complexity and handles API failures through exponential backoff retry mechanisms.

E. Robustness Quantification

The system computes robustness scores through weighted aggregation of classification outcomes:

$$R = \frac{|E| \times 1.0 + |M| \times 0.7 + |D| \times 0.0}{|T|} \qquad (1)$$

where $|E|$, $|M|$, and $|D|$ represent counts of Equivalent, Minor variation, and Deviation responses, and $|T|$ is the total mutation count. The weights (1.0, 0.7, 0.0) are heuristic choices reflecting production requirements: equivalent responses receive full credit (1.0), minor variations receive partial credit (0.7) as they represent acceptable semantic drift, and behavioral deviations receive zero credit (0.0) as they signal critical failures requiring immediate attention.

Additionally, the system computes mutation-type-specific robustness scores:

$$R_{type} = \frac{\sum_{i \in type} w_i \times s_i}{\sum_{i \in type} w_i} \qquad (2)$$

where w_i represents the severity weight of mutation i and s_i is its similarity score, enabling granular analysis of robustness patterns across different mutation categories.

IV. Experimental Evaluation

A. Experimental Design

Experimental Design. Using *Quirx*, we evaluate four LLM models: (M_1) OpenAI GPT-3.5-turbo, (M_2) GPT-4o-mini, (M_3) Anthropic Claude-3.5-Sonnet, and (M_4) Claude-Sonnet-4 across three task categories. *Sentiment Classification* tests constrained tasks with discrete outputs (Positive/Negative/Neutral), representing classification scenarios with well-defined answer spaces. *Text Summarization* evaluates content extraction and preservation under prompt variations, representing intermediate complexity tasks requiring content understanding. *SQL Generation* tests complex generative tasks requiring structured reasoning and domain expertise, representing high-complexity generation scenarios.

Experimental Protocol. Experiments use standardized protocols with 4-8 mutations per prompt (40% lexical, 30% semantic, 30% structural), rate limiting (0.3-1.0s delays), metadata collection, and three-tier similarity analysis with adaptive thresholds. The details of the combinations are available in the evaluation scripts.

Evaluation Metrics. Evaluation metrics include overall robustness scores (Equation (1)), mutation-type analysis, response consistency percentages, performance metrics (response time, token usage), failure analysis, and cost implications.

V. QUIRX IN ACTION: TOOL DEMONSTRATION

A. Discovering Task-Dependent Vulnerability Patterns

We used *Quirx* to evaluate prompt robustness across four prominent models on three task categories.

For *sentiment classification* prompts, the evaluation using *Quirx* showed that all four tested models achieve perfect robustness scores (1.00/1.00), with 100% of mutations producing equivalent responses. *Text summarization* tasks revealed significant provider variation in robustness: GPT-3.5-turbo achieved 0.35 robustness with 50% minor variations and 50% behavioral deviations, GPT-4o-mini showed lower robustness (0.23) with 33% minor variations and 67% deviations, while both Claude models demonstrated superior performance (0.58 robustness) with 83% minor variations and only 17% behavioral deviations.

For *SQL generation* tasks, the evaluation revealed consistently strong performance across most models. GPT-3.5-turbo, GPT-4o-mini, and Claude-3.5-Sonnet all achieved identical robustness scores (0.80) with 33% equivalent responses, 67% minor variations, and zero behavioral deviations. Claude-Sonnet-4 demonstrated exceptional stability with perfect robustness (1.00 score) and 100% equivalent responses, representing the most robust performance across all task categories.

B. Identifying Critical Mutation Types and Failure Patterns

Quirx's tri-dimensional mutation analysis revealed distinct vulnerability patterns across task categories. In text summarization, structural mutations (sentence reordering) were primary drivers of behavioral deviations, particularly affecting OpenAI models more severely than Claude models. SQL generation tasks showed remarkable resilience to structural mutations, with most variations resulting in minor formatting differences rather than functional changes. Classification tasks demonstrated complete robustness to all mutation types across all models. This insight enables developers to prioritize structural prompt consistency for summarization tasks while having confidence in SQL generation robustness.

For example, *Quirx* identified that a SQL generation prompt became unreliable when instructions were reordered, causing GPT-3.5-turbo to generate different JOIN conditions and date functions (CURDATE() vs CURRENT_DATE) across mutations. Similarly, text summarization prompts showed varying emphasis when structural mutations altered guideline ordering, affecting which aspects were prioritized in summaries.

The tool's multi-dimensional similarity analysis provides actionable insights by classifying response differences into three categories: Equivalent, Minor Variation, and Deviation, enabling developers to understand not just whether prompts fail, but how they fail. This granular analysis supports iterative prompt refinement by identifying specific mutation types that cause problems and quantifying improvement across optimization cycles.

C. Cross-Provider Analysis and Strategic Insights

The cross-provider evaluation using *Quirx* revealed significant performance variations across task categories. While all models achieved perfect classification robustness (1.00), text summarization showed substantial provider differences: Claude models consistently outperformed OpenAI models (0.58 vs. 0.23-0.35 robustness). SQL generation demonstrated strong overall performance (0.80-1.00), with Claude-Sonnet-4 achieving perfect robustness (1.00) and all other models showing identical performance (0.80). Performance characteristics varied significantly: GPT-3.5-turbo provided fastest response times (0.69s average), while Claude models showed more variable latency but superior summarization robustness.

The evaluation showed that structural mutations pose the greatest threat across all providers, enabling focused prompt engineering efforts on high-risk mutation categories.

D. Tool Usage and Availability

Quirx provides both command-line and programmatic interfaces that enable systematic prompt robustness evaluation with minimal setup. The tool supports multiple LLM providers and generates both markdown and JSON reports automatically, making it suitable for diverse development workflows and CI/CD integration. Evaluation of 8 mutations across 4 models completes in 2-3 minutes with API costs under $0.50 per prompt. Fig. 2 shows usage examples of the command line interface.

The complete evaluation scripts used in this paper are available in the repository: `run_evaluation.py` for comprehensive testing and `final_evaluation.py` for multi-provider comparison. All experimental prompts are located in the `examples/` directory: `prompt_classifier.txt` for sentiment classification, `prompt_summarizer.txt` for text summarization, and `prompt_sql.txt` for SQL generation tasks.

```
# Cross-provider summarization comparison
quirx --prompt "examples/prompt_summarizer.txt"
        --provider openai,claude
        --mutations 8
        --output summarization_report.json

# SQL generation robustness testing
quirx --prompt "examples/prompt_sql.txt"
        --provider anthropic
        --model claude-sonnet-4-20250514
        --mutations 8
        --format html
```

Fig. 2. Quirx Command-Line Usage Examples for Evaluation Reproduction

TABLE I
MODEL PERFORMANCE AGAINST MUTATED PROMPTS FOR EACH TASK USING *Quirx*

		Robustness Scores			Equivalent (%)			Minor Var. (%)			Deviation (%)			Performance		
ID	**Model**	**Class**	**Summ**	**SQL**	**Class**	**Summ**	**SQL**	**Class**	**Summ**	**SQL**	**Class**	**Summ**	**SQL**	**Time**	**Tokens**	**Cost**
M_1	GPT-3.5-turbo	1.00	0.35	0.80	100	0	33	0	50	67	0	50	0	0.75	165	1.0x
M_2	GPT-4o-mini	1.00	0.23	0.80	100	0	33	0	33	67	0	67	0	1.19	174	1.5x
M_3	Claude-3.5-Sonnet	1.00	0.58	0.80	100	0	33	0	83	67	0	17	0	4.61	181	1.2x
M_4	Claude-Sonnet-4	1.00	0.58	**1.00**	100	0	**100**	0	83	**0**	0	17	**0**	2.00	184	1.8x

VI. DISCUSSION AND IMPLICATIONS

A. Task-Dependent Robustness Patterns

The evaluation revealed fundamental differences across task categories and providers. Classification tasks showed exceptional stability (1.00 robustness) with well-defined output spaces across all models. Text summarization tasks demonstrated significant provider variation (0.23-0.58 robustness), with Claude models showing superior resilience to prompt variations compared to OpenAI models. SQL generation tasks exhibited consistently strong performance (0.80-1.00 robustness), with structural mutations causing minor formatting variations rather than behavioral deviations.

B. Architectural and Production Insights

Model architecture significantly influences robustness patterns, particularly in summarization tasks where Claude models consistently outperformed OpenAI models (0.58 vs. 0.23-0.35 robustness). While all models achieved identical performance in classification tasks (1.00), text summarization revealed substantial provider differences in handling prompt variations. SQL generation demonstrated remarkable consistency across models (0.80 for three models, 1.00 for Claude-Sonnet-4), suggesting that structured generation tasks may be inherently more robust to prompt perturbations. *Quirx* addresses critical gaps in LLM development by providing automated vulnerability discovery, evidence-based model selection, and systematic robustness evaluation that enables transition from ad-hoc testing to structured quality assurance practices.

C. Limitations and Future Directions

Our evaluation covers English-language prompts and three representative task categories. Future work should expand to multilingual scenarios, domain-specific applications, and multi-turn conversations where context history influences robustness. The semantic similarity metrics may not capture all domain-specific quality measures, suggesting need for task-specific evaluation criteria. Evaluation time scales linearly with mutation count, handling prompts up to 2000 tokens effectively.

VII. CONCLUSION

This paper presents *Quirx*, a mutation-based framework addressing systematic prompt robustness evaluation in LLM development. Through tri-dimensional mutation strategies and multi-provider evaluation capabilities, *Quirx* enables comprehensive robustness assessment across diverse task categories and provider architectures.

The evaluation revealed distinct robustness patterns: universal stability in classification tasks (1.00), significant provider variation in summarization (0.23-0.58), and consistent strong performance in SQL generation (0.80-1.00). These findings demonstrate that prompt robustness is both task-dependent and provider-specific, emphasizing the necessity of systematic evaluation. *Quirx* addresses the need for automated prompts robustness testing in production LLM-based systems.

REFERENCES

[1] Tom Brown et al. "Language models are few-shot learners". In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[2] Jun Li et al. "Fuzzing: a survey". In: *Cybersecurity* 1.1 (2018), p. 6.

[3] Pengfei Liu et al. "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing". In: *ACM computing surveys* 55.9 (2023), pp. 1–35.

[4] Felipe Maia Polo et al. "Efficient multi-prompt evaluation of LLMs". In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 22483–22512.

[5] John Morris et al. "TextAttack: A Framework for Adversarial Attacks, Data Augmentation, and Adversarial Training in NLP". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. 2020, pp. 119–126.

[6] Nils Reimers et al. "Sentence-bert: Sentence embeddings using siamese bert-networks". In: *arXiv preprint arXiv:1908.10084* (2019).

[7] Laria Reynolds et al. "Prompt programming for large language models: Beyond the few-shot paradigm". In: *Extended abstracts of the 2021 CHI conference on human factors in computing systems*. 2021, pp. 1–7.

[8] Jason Wei et al. "Chain-of-thought prompting elicits reasoning in large language models". In: *Advances in neural information processing systems* 35 (2022), pp. 24824–24837.

[9] Zihao Zhao et al. "Calibrate before use: Improving few-shot performance of language models". In: *International conference on machine learning*. PMLR. 2021, pp. 12697–12706.