Prompt Drift as a Reliability Risk: Measuring Stability of LLM-Based SE Pipelines

Ethan Chen (918004305), Khoi Nguyen (919833517), Pei-Yu Lin (925570136), Pablo Rodriguez (925562115), Shuang Ma (924922662)

1 Introduction & Motivation

Large Language Models have rapidly transformed software engineering with applications spanning code generation, test synthesis, and documentation, as evidenced by recent surveys cataloging over 900 LLM-based SE studies [1, 2]. However unlike traditional deterministic software components, LLM-based systems can produce different outputs for the same inputs, which poses significant reliability risks for production deployments. Recent empirical studies reveal that even with deterministic settings like temperature set to zero and fixed seeds, LLMs show accuracy variations up to 15% across identical runs with performance gaps between best and worst runs reaching 70% [3].

While the SE community has studied LLM performance on various tasks, investigation of stability remains limited. Researchers have proposed metrics for measuring prompt sensitivity and output consistency [4], but these insights have not been brought together into practical guidance for SE practitioners building production systems. Our study addresses this gap by measuring prompt drift across representative SE tasks to identify contributing factors and provide practitioners with recommendations for deploying more reliable LLM-based systems.

2 Research questions

Prior work shows LLMs exhibit non-determinism even at temperature=0, with accuracy varying up to 15% across identical runs [3]. Additionally, LLMs are sensitive to prompt rephrasings, with metrics proposed to measure this instability [4]. However, these studies have not systematically examined which SE tasks are most affected or what factors drive instability in production pipelines. Our research questions address these gaps:

RQ1: How consistent are open-source code LLMs across repeated identical prompts?

RQ2: How sensitive are models to semantically equivalent prompt paraphrasing?

RQ3: Which models exhibit the highest stability for SE tasks, and how can we visualize these differences to aid practitioner decision-making?

RQ4: What task characteristics and prompt features contribute most to instability, and is there a relationship between consistency and correctness?

3 Research Approach

This research adopts an empirical measurement-based approach to investigate prompt drift as a reliability risk in LLM-based software engineering pipelines. We will test 3–4 large language models (mix of commercial like GPT-4 and open-source like CodeLlama/Llama-3, depending on cost and availability) on four representative SE tasks: code generation, code summarization, bug detection, and test generation. We use standard benchmark datasets like HumanEval and CodeSearchNet which contain code snippets for reproducibility and comparability with prior work. For each task, we construct five semantically equivalent prompt variations that differ only in wording or structure. For each prompt, we perform 10 repeated runs at temperature=0. This design yields approximately 600–800 total model queries, enabling analysis of both run-to-run consistency and sensitivity to prompt paraphrasing.

We measure output stability using four complementary metrics: (1) consistency rate (percentage of identical outputs for the same prompt), (2) paraphrase sensitivity (similarity across prompt variations), (3) output validity (whether code compiles and follows expected format), and (4) correctness (using

test cases from HumanEval). Beyond aggregate metrics, we will develop visual analysis tools including similarity heatmaps, consistency timelines, and side-by-side output comparisons to help reveal how and when outputs diverge. Potential challenges include API rate limiting (mitigated through batched calls), unparseable model outputs (addressed with robust parsing heuristics and manual inspection), and statistical analysis complications if models show near-perfect or highly variable consistency (addressed through qualitative analysis when quantitative differences are small).

4 Timeline and Milestones

Weeks 2–3 (Jan 19–26): Literature review and background research. Finalize model selection and dataset preparation. Deliver background presentation on Jan 26.

Weeks 4–5 (Jan 27–Feb 9): Infrastructure setup: implement model inference pipeline, create prompt variations for all tasks, and build metrics evaluation code.

Weeks 6–7 (Feb 10–17): Run experiments across all models and tasks. Collect and organize output data. Submit progress report on Feb 17 showing preliminary results from at least 2 tasks.

Weeks 8–9 (Feb 18–Mar 3): Complete remaining experiments, implement visualizations (heatmaps, consistency charts), and perform statistical analysis.

Weeks 10–11 (Mar 4–18): Write final report, create presentation materials, and finalize all figures. Submit final report on Mar 18.

Key Deliverables: Background presentation (Jan 26), progress report with preliminary data (Feb 17), final report with complete analysis and visualizations (Mar 18).

5 Team Membership and Attestation

Team Members:

- Pablo Rodriguez (925562115)
- Pei Yu Lin (925570136)
- Ethan Chen (918004305)
- Khoi Nguyen (919833617)
- Shuang Ma (924922662)

All team members have read, contributed to, and agree with the contents of this proposal.

References

[1] Zhang, Q., et al. "A Survey on Large Language Models for Software Engineering." arXiv:2312.15223 (2023). https://arxiv.org/abs/2312.15223

[2] Liu, J., et al. "Large Language Model-Based Agents for Software Engineering: A Survey." arXiv:2409.02977 (2024). https://arxiv.org/abs/2409.02977

[3] Atil, B., et al. "Non-Determinism of 'Deterministic' LLM Settings." arXiv:2408.04667 (2024). https://arxiv.org/abs/2408.04667

[4] Errica, F., et al. "What Did I Do Wrong? Quantifying LLMs' Sensitivity and Consistency to Prompt Engineering." arXiv:2406.12334 (2024). https://arxiv.org/abs/2406.12334