The Impact of Prompt Engineering on Code Generation Accuracy and Hallucination Patterns in Language Models

Kadin Matotek [1] Linh B. Ngo [1]

[1]West Chester University, Department of Computer Science

Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities in code generation, yet their reliability remains a critical concern. This study investigates how different prompt engineering strategies affect model accuracy and hallucination patterns across coding tasks.

Research Questions

- How do concise versus verbose instructions impact code generation accuracy?
- What types of errors and hallucinations emerge across different model sizes?
- How do reasoning strategies (Chain-of-Thought, Direct, Program-Aided) affect performance?

Methodology

We evaluated 5 Qwen2.5-Coder model variants (0.5B to 14B parameters) on 323 programming problems using a systematic prompt engineering framework.

Dataset

- **Source:** Kattis competitive programming platform
- Used for rigorous competitions including ICPC
- **Difficulty Range:** 0-2 (on 0-10 Kattis scale)
- Focus on fundamental algorithmic challenges

Experimental Design

- **Base Instructions:** Concise vs. Verbose (2 options)
- **Reasoning Strategies:** Chain-of-Thought, Direct, Program-Aided (3 options)
- **Problem Decomposition:** None vs. Basic (2 options)
- **Output Formats:** Code only, Explanation + Code, Code + Explanation (3 options)
- **Prompt Variations:** $2 \times 3 \times 2 \times 3 = 36$ unique configurations
- **Total Solutions Generated:** 323 problems $\times$ 36 prompts $\times$ 5 models $= 58,140$ code generations

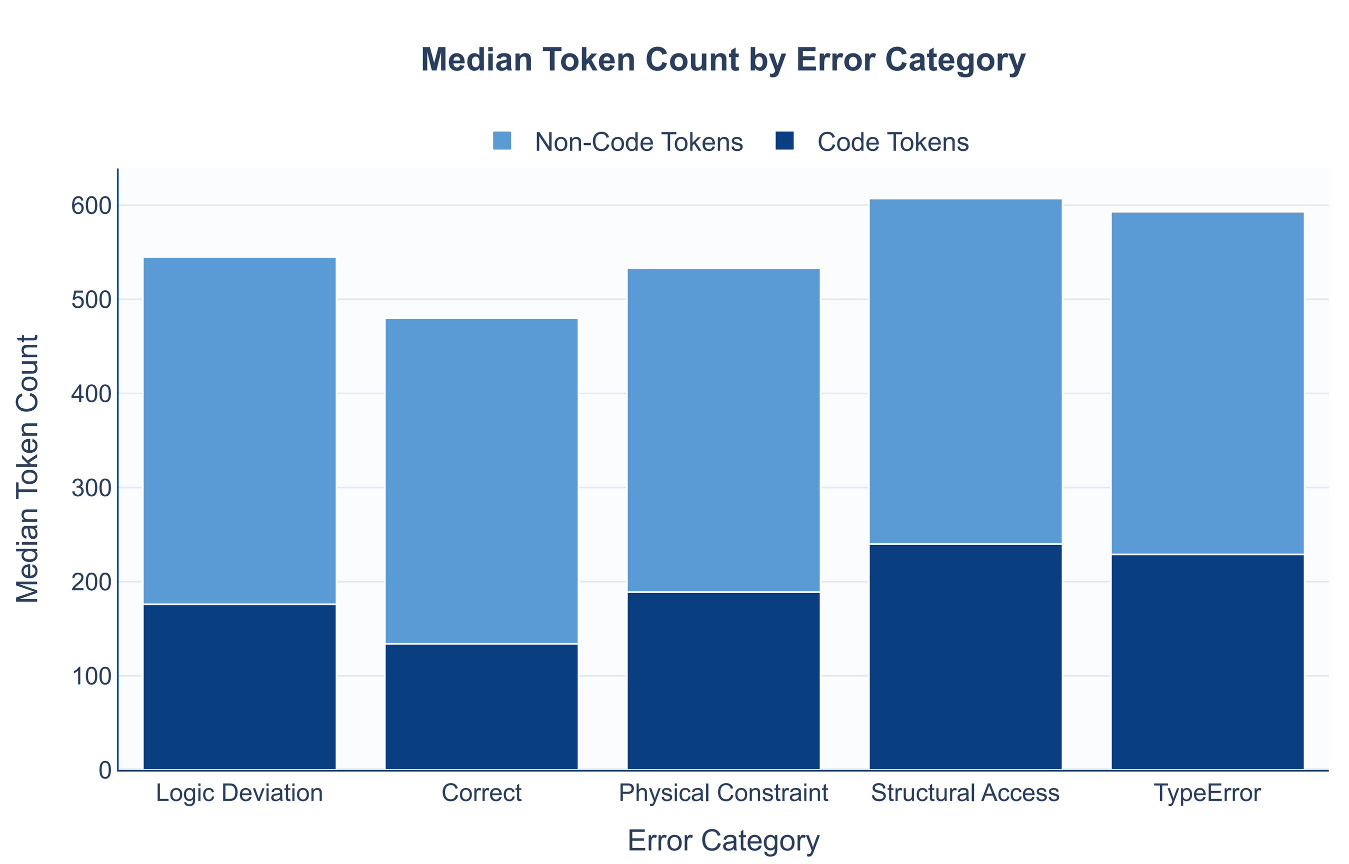

Figure 1. Experimental framework showing prompt construction pipeline and evaluation

Key Finding: Conciseness Advantage

Across all 18 prompt configuration comparisons, **concise instructions outperformed verbose ones in 94.4%** of cases with an average improvement of +1.35%.

This effect was most pronounced in smaller models, suggesting that verbose instructions may introduce noise that degrades performance.

Results: Prompt Strategy Performance (averaged across all models)

Top 3 Configurations

Base	Reasoning	Decomp	Output	Acc.
Concise	Direct	None	Code only	**33.10%**
Concise	Direct	None	Exp + Code	32.79%
Concise	CoT	None	Exp + Code	32.04%

Bottom 3 Configurations (all verbose!)

Base	Reasoning	Decomp	Output	Acc.
Verbose	CoT	Basic	Code + Exp	28.17%
Verbose	PAL	Basic	Code + Exp	27.80%
Verbose	PAL	Basic	Exp + Code	26.78%

Max gap: ***+6.32%*** *(best concise vs worst verbose)*

Model Size Effects

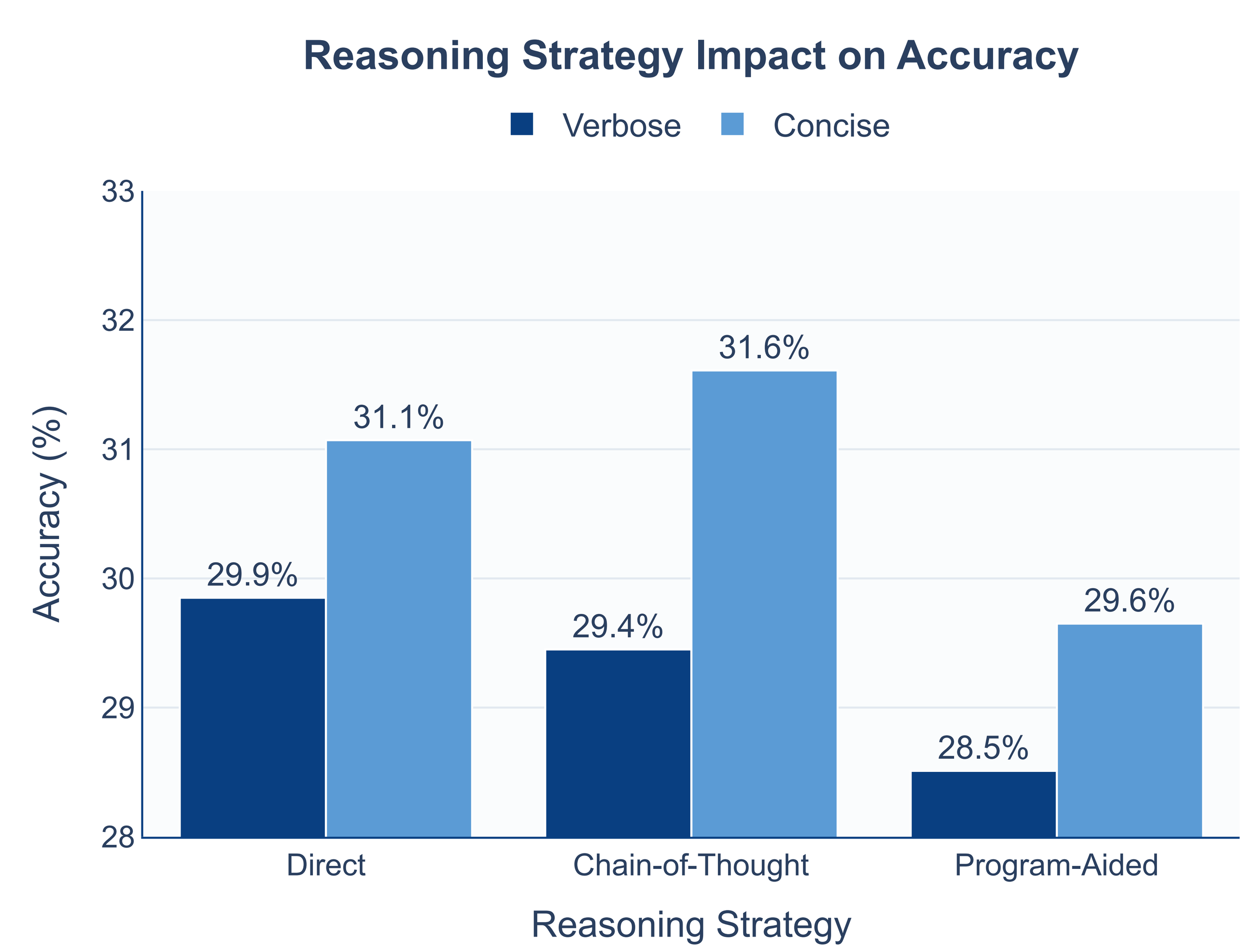

Figure 2. Accuracy comparison showing concise prompts consistently outperform verbose across all reasoning strategies.

Error and Hallucination Analysis

We manually defined and annotated 28 distinct error types. Crucially, we observe:

Highly reproducible hallucination profiles: Independent runs (v1 vs v2) produce nearly identical error distributions (overlapping solid/dashed lines).

Systematic shift with model scale: Larger models dramatically reduce the dominant error mode while maintaining the same overall failure pattern.

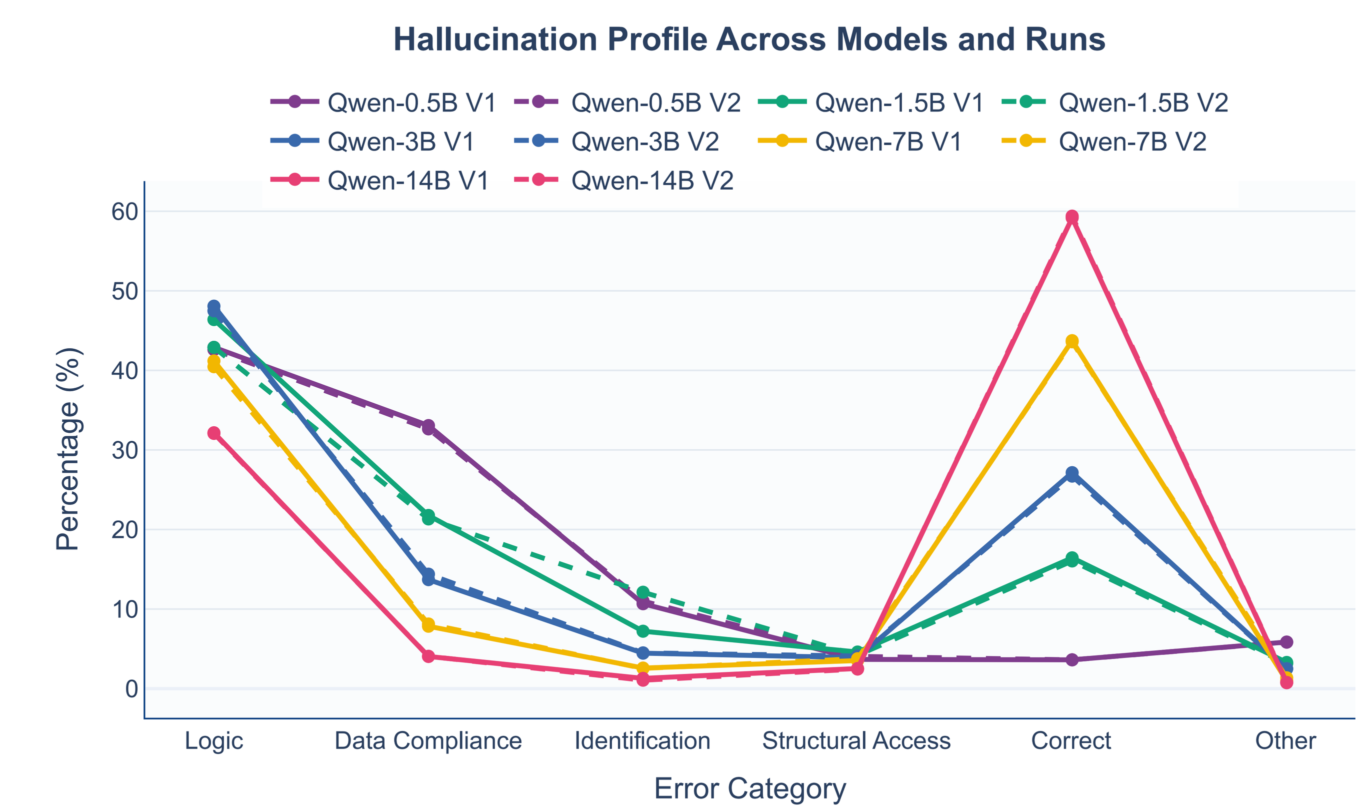

Figure 3. **Hallucination profiles are remarkably stable across independent runs** (solid vs dashed lines nearly overlap) but **shift predictably with model size**.

Most Common Error Types

- **Logic Deviation** (41.6%): Incorrect algorithmic approach
- **ValueError** (12.9%): Invalid input handling
- **TypeError** (2.9%): Type mismatches
- **IndexError** (3.1%): Array boundary violations
- **NameError** (2.8%): Undefined variables

Conclusions

Key Takeaways

- Concise prompts consistently outperform verbose alternatives
- Smaller models are more sensitive to prompt engineering
- Logic deviations are the dominant failure mode
- Reasoning strategy and output format significantly affect accuracy

References

[1] Claude E. Shannon.
A mathematical theory of communication.
Bell System Technical Journal, 27(3):379–423, 1948.