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 Qwen model variants (0.5B to 14B parameters) on 3,230 programming problems using a systematic prompt engineering framework.

Experimental Design

- **Base Instructions:** Concise vs. Verbose
- **Reasoning Strategies:** Chain-of-Thought, Direct, Program-Aided
- **Problem Decomposition:** None vs. Basic
- **Output Formats:** Code only, Explanation + Code, Code + Explanation
- **Total Configurations:** 36 prompt variants per model
- **Dataset:** 116,280 total test cases

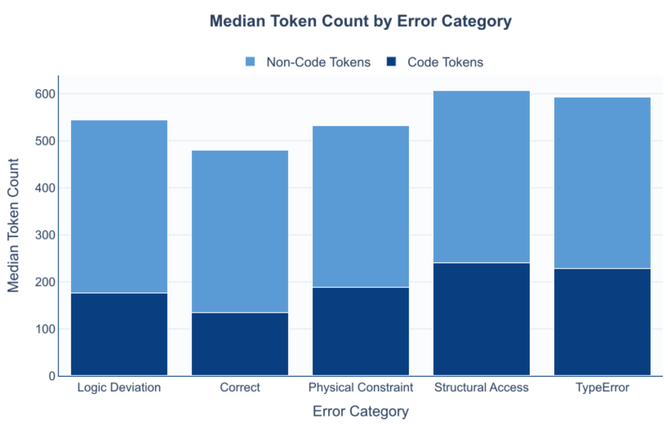

Figure 1. Experimental framework showing prompt construction pipeline and evaluation methodology.

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 in-

Results: Prompt Strategy Performance

Top Performing Configurations

Base	Reasoning	Decomp	Output	Acc %
Concise	CoT	None	Exp + Code	32.07
Concise	Direct	None	Code only	31.95
Concise	Direct	None	Exp + Code	31.83
Concise	CoT	None	Code only	31.67
Verbose	CoT	None	Code only	30.71
Verbose	Direct	Basic	Code only	30.19

Table 1. Accuracy of top prompt configurations (n=3,230 per config).

Biggest Performance Gaps

- Direct reasoning, no decomposition, code-only output: **+2.88%**
- Chain-of-Thought with explanation + code: **+2.26%**

Model Size Effects

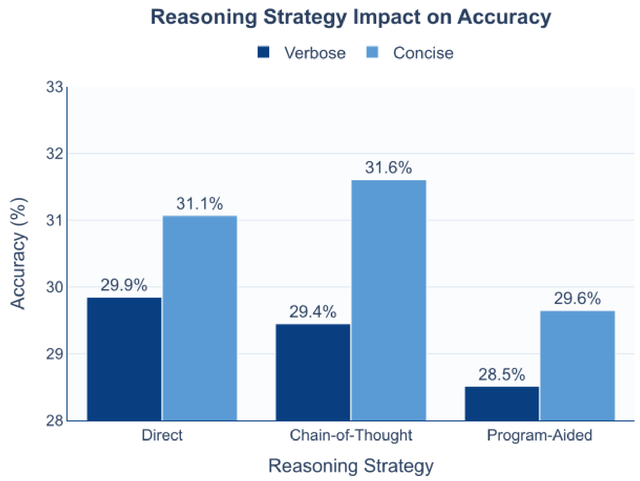

Figure 2. Accuracy comparison showing concise prompts consistently outperform verbose across all model sizes. The advantage diminishes as models grow larger.

Smaller models showed greater sensitivity to prompt verbosity, with relative improvements from **+21.0%** (0.5B) to **+0.4%** (14B).

Error and Hallucination Analysis

We categorized 28 distinct error types, revealing systematic failure patterns.

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

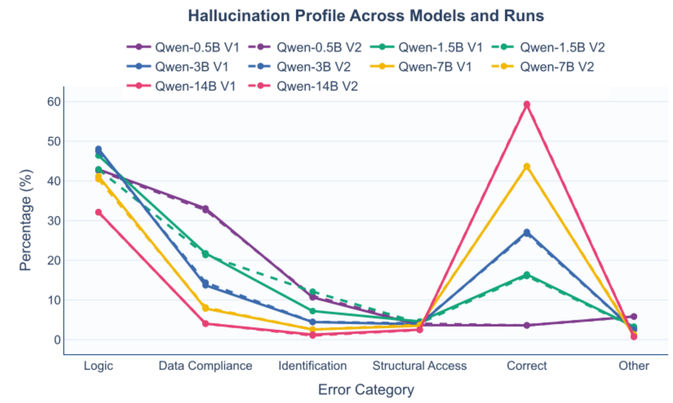

Figure 3. Distribution of error types across model sizes. Logic deviations dominate but decrease with model capacity.

Token Usage Patterns

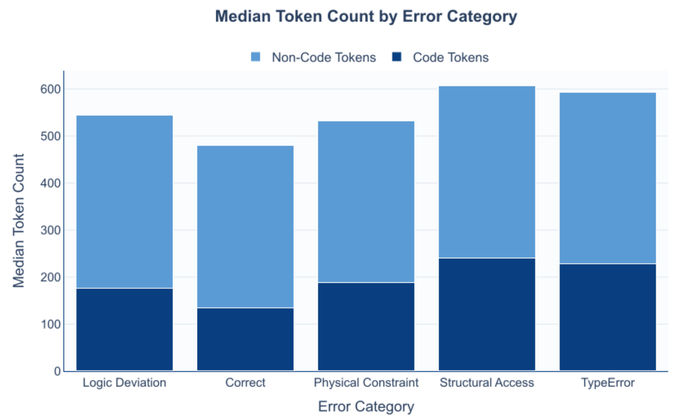

Figure 4. Median token counts for correct vs. incorrect responses.

Correct solutions used 343–501 median tokens depending on model size. Logic deviation errors showed similar token usage, indicating verbosity alone is not pre-