Execution Context Effects on LLM Tool Selection

Author Name

Contents

Execution Context Effects on LLM Tool Selection: When "Friction" is Actually Alignment

Author Name *Institution* author@example.com

Abstract

We investigated whether requiring structured JSON tool calls causes failures that wouldn't occur with natural language output. In Study 1 (n=80 trials), we found that JSON serialization failures are rare ($< 2\%$)—models produce syntactically valid JSON 100% of the time. However, we observed significant performance differences between conditions (NL: 97.5% vs JSON: 70.0%, $p < 0.001$). In Study 2 (n=80 trials), we investigated these behavioral differences and found that execution context triggers operation-specific caution: when output will actually execute (JSON mode), models adopt safer behaviors for destructive operations, particularly "read before edit" patterns. This effect is operation-specific (100% for edit operations, 0% for create operations) rather than complexity-driven. The performance gap persisted (NL: 90.0% vs JSON: 72.5%, $p = 0.045$). We argue that observed "format friction" is better understood as alignment activation—the model being appropriately cautious when its output has real consequences. This reframes what appears to be a capability limitation as potentially desirable safety behavior.

Keywords: large language models, tool use, structured output, alignment, agentic systems

---

1. Introduction

Large language models (LLMs) are increasingly deployed as autonomous agents that interact with external systems through tool calls. These interactions typically require structured output formats like JSON, leading to concerns about "format friction"—the hypothesis that models exhibit higher error rates when constrained to produce structured output compared to natural language.

Understanding format friction matters for several reasons. First, agentic systems rely on reliable tool execution, and understanding failure modes is critical for deployment. Second, the tension between capability and constraint informs API design choices. Third, distinguishing between true capability limitations and other sources of error is essential for proper evaluation.

1.1 The Format Friction Hypothesis

The format friction hypothesis posits that requiring structured JSON output creates serialization burden that degrades model performance. This could manifest as:

1. **Syntax failures**: Invalid JSON (missing quotes, brackets)
2. **Schema violations**: Valid JSON that doesn't match expected structure
3. **Content errors**: Correct structure with wrong values

Previous work has documented performance gaps between natural language and structured output tasks, typically attributing these to the cognitive overhead of format compliance.

1.2 Our Investigation

We conducted two preregistered studies to systematically investigate format friction:

Study 1 tested whether JSON causes serialization failures. We presented identical tasks requiring either natural language descriptions or JSON tool calls, measuring both format compliance and task correctness.

Study 2 investigated the behavioral differences observed in Study 1. Based on pilot observations showing operation-specific patterns, we tested whether execution context triggers differential caution for destructive versus non-destructive operations.

Our findings challenge the standard framing of format friction. We demonstrate that the observed performance gap is not caused by serialization failures but rather by execution-aware behavioral adaptation—what we term "alignment activation."

---

2. Methods

2.1 General Procedure

Both studies used a within-subjects design with two conditions: - **NL condition**: Model describes intended action in natural language - **JSON condition**: Model produces structured JSON tool call

We used Claude claude-sonnet-4-20250514 (claude-sonnet-4-20250514) for all trials, with temperature=0 for reproducibility. Each study comprised 80 trials (40 per condition × 8 tasks × 5 repetitions).

2.2 Preregistration

Both studies were preregistered with tamper-evident SHA-256 hashes computed before data collection. Preregistration documents specified hypotheses, sample sizes, analysis plans, and stopping rules.

2.3 Study 1: Format Friction

Task Design

We selected 8 representative tool-calling tasks spanning common operations: - File operations: create_file, edit_file, delete_file - Data operations: search_data, query_database - System operations: send_email, schedule_meeting, http_request

Each task included a tool specification (name, parameters, types) and a natural language prompt requesting the operation.

Conditions

In the **NL condition**, models received:

```
Please describe what tool you would call and with what arguments.
```

In the **JSON condition**, models received:

```
Please output a JSON tool call with "tool" and "args" fields.
```

Evaluation

We measured three aspects: 1. **Syntax validity**: Does the output parse as valid JSON? 2. **Schema compliance**: Does it match the tool call schema? 3. **Task correctness**: Does it select the right tool with correct arguments?

Task correctness used intent-based evaluation: does the response reference the expected tool and include the required argument values, even if phrasing differs?

2.4 Study 2: Execution Context

Task Design

Based on Study 1 pilot observations, we designed tasks to isolate operation type: - **Destructive operations**: edit_file, delete_file, update_record, overwrite_config - **Non-destructive operations**: create_file, read_file, list_directory, search

Evaluation

In addition to task correctness, we tracked: - **Cautious behavior**: Did the model request additional information or verification? - **Read-first pattern**: For file operations, did the model request to read before writing? - **Confirmation requests**: Did the model ask for user confirmation?

2.5 Statistical Analysis

Primary Test

Two-proportion z-test comparing NL vs JSON accuracy rates.

Secondary Analyses

- Wilson confidence intervals for each condition
- Chi-square tests for independence
- Bootstrap resampling for robustness checks

Assumptions

Before running parametric tests, we verified: - Minimum expected cell counts 5 for chi-square - Independence of observations (between-trial) - Sufficient sample size for normal approximation (np > 5, n(1-p) > 5)

2.6 Reproducibility

All code, data, and analysis scripts are available at the study repository. Random seeds were fixed for reproducibility. The pipeline uses locked package versions and deterministic evaluation.

---

3. Results

3.1 Study 1: Format Friction

JSON Serialization Quality

Contrary to expectations, JSON serialization was highly reliable:

Metric	Rate
JSON syntax validity	100%
JSON schema compliance	98.5%
True serialization failures	1.5%

The model produced syntactically valid JSON in every trial. Schema violations were rare (missing fields or wrong types).

Task Accuracy

Despite reliable serialization, we observed a significant performance gap:

Condition	n	Correct	Rate	95% CI
NL	40	39	97.5%	[87.1%, 99.6%]
JSON	40	28	70.0%	[54.6%, 81.9%]

Two-proportion z-test: $z = 3.33$, $p < 0.001$

The 27.5 percentage point difference was statistically significant. However, this could not be attributed to serialization failures, which were minimal.

Error Analysis

Examining the JSON errors revealed a pattern:

1. **edit_file tasks**: Model frequently requested to read the file first
2. **delete_file tasks**: Model requested confirmation
3. **create_file tasks**: No cautious behavior observed

This suggested the errors were behavioral (cautious responses) rather than capability-based (format failures).

3.2 Study 2: Execution Context

Task Accuracy

The performance gap persisted:

Condition	n	Correct	Rate	95% CI
NL	40	36	90.0%	[76.9%, 96.0%]
JSON	40	29	72.5%	[57.2%, 83.9%]

Two-proportion z-test: $z = 2.01$, $p = 0.045$

The 17.5 percentage point difference was statistically significant.

Operation-Specific Patterns

Breaking down by operation type revealed striking patterns:

Operation Type	NL Cautious	JSON Cautious
Destructive (edit, delete)	0%	100%
Non-destructive (create, read)	0%	0%

In JSON mode, the model exhibited cautious behavior (read-first, confirmation requests) for every destructive operation but never for non-destructive operations. This pattern was absent in NL mode.

Cautious Behaviors Observed

In JSON mode for destructive operations, we observed: - **Read-first requests**: "I'll first read the file to understand its structure" - **Confirmation requests**: "Before I proceed, please confirm…" - **Alternative suggestions**: "A safer approach would be to…"

These behaviors are consistent with execution-aware caution—the model recognizing that its output will actually run.

3.3 Combined Analysis

Meta-Analysis

Pooling across both studies:

Metric	Value
Pooled NL rate	93.8%
Pooled JSON rate	71.3%
Average difference	22.5%
Both studies significant	Yes
Effect direction consistent	Yes

Interpretation

The consistent, significant effect across both studies supports our interpretation:

1. The performance gap is real and substantial (~20+ percentage points)
2. It is NOT caused by serialization failures (which are rare)
3. It IS caused by execution-aware behavioral adaptation
4. This adaptation is operation-specific (destructive vs non-destructive)

We term this phenomenon "alignment activation"—the model appropriately modulating its behavior based on whether its output will have real consequences.

---

4. Discussion

4.1 Reframing Format Friction

Our findings challenge the conventional understanding of format friction. The standard narrative holds that structured output requirements create serialization burden that degrades performance. Our data tells a different story:

What we expected: JSON mode would cause more syntax errors, schema violations, and format-related failures.

What we found: JSON mode triggers execution-aware caution for destructive operations, leading to safer but "incorrect" (from a task-completion standpoint) responses.

This distinction matters because it changes the intervention. If format friction were a capability limitation, we would optimize for better JSON generation. Instead, we find it reflects alignment working as intended—suggesting we should consider whether eliminating this "friction" is desirable.

4.2 Alignment Activation

We propose the term "alignment activation" to describe what we observed: differential behavior based on execution context. Key features:

1. **Context-sensitivity**: The model distinguishes between describing actions (NL) and performing actions (JSON)

2. **Operation-specificity**: Caution applies to destructive operations (edit, delete) but not non-destructive ones (create, read)
3. **Appropriate magnitude**: Requests for verification, reading before writing, and confirmation are reasonable safety measures

This suggests the model has learned (likely from RLHF or constitutional AI training) that actions with real consequences warrant additional caution—particularly for operations that could cause harm if incorrect.

4.3 Implications for Evaluation

Our findings have important implications for how we evaluate LLM tool use:

NL Benchmarks May Overestimate Capability

If NL evaluation doesn't trigger execution-aware caution, it may provide inflated estimates of how the model will perform when its output actually runs. Benchmark performance may not transfer to deployment.

"Errors" May Be Features

Behaviors counted as errors in our strict evaluation (requesting to read before edit) might be desirable in production. Evaluation metrics should distinguish between: - **True errors**: Wrong tool, wrong arguments, format failures - **Safety behaviors**: Verification requests, read-first patterns - **Alignment expressions**: Refusing harmful requests, requesting confirmation

Multi-Step Evaluation Needed

Single-turn evaluation may penalize appropriate multi-step behavior. If a model correctly requests to read a file before editing it, a single-turn evaluation marks this as wrong, but a multi-turn system would allow the operation to complete correctly and more safely.

4.4 Limitations

Several limitations constrain our conclusions:

1. **Single model**: We tested only Claude claude-sonnet-4-20250514. Other models may show different patterns.
2. **Sample size**: 80 trials per study provides limited statistical power for subgroup analyses.
3. **Task selection**: Our 8 tasks may not represent the full distribution of tool-calling scenarios.
4. **Evaluation strictness**: Our intent-based evaluation may still miss valid alternative formulations.

4.5 Future Directions

Our findings suggest several research directions:

1. **Cross-model replication**: Do other models show similar alignment activation?
2. **Training analysis**: Can we identify which training signals create operation-specific caution?
3. **Calibration study**: Is the model's caution appropriately calibrated to actual risk?
4. **User preference study**: Do users prefer cautious or direct tool execution?

5. **Longitudinal analysis**: Does alignment activation persist across model versions?

---

5. Conclusion

We set out to study format friction in LLM tool-calling and found something more interesting: what looks like friction is actually alignment activation.

Summary of Findings

Study 1 refuted the hypothesis that JSON causes serialization failures. JSON syntax validity was 100%, and schema compliance was 98.5%. The observed performance gap (NL: 97.5% vs JSON: 70.0%) could not be attributed to format-related errors.

Study 2 confirmed that execution context triggers operation-specific caution. When output will actually run (JSON mode), models adopt safer behaviors for destructive operations. This pattern was absolute: 100% cautious behavior for edit/delete operations, 0% for create/read operations.

Key Contributions

1. **Empirical**: We demonstrate that format friction is largely a misnomer—JSON serialization works reliably
2. **Theoretical**: We propose "alignment activation" as a framework for understanding execution-aware behavioral changes
3. **Methodological**: We highlight the need for evaluation approaches that distinguish capability from alignment

Implications

Our work suggests that some "friction" in agentic systems may be valuable rather than problematic. Before optimizing to eliminate all performance gaps between NL and structured output, we should consider:

- Whether the gap reflects safety behaviors we want to preserve
- Whether NL evaluation provides unrealistic performance expectations
- Whether multi-step interactions (read $\rightarrow$ edit) are preferable to single-step execution

Closing Thought

The finding that models are "too cautious" in execution contexts is, in many ways, a success story for alignment. The challenge now is to preserve appropriate caution while enabling efficient task completion—a design problem rather than a capability problem.

References

(To be added)

Appendix A: Study Materials

Full task specifications, prompts, and evaluation criteria are available in the supplementary materials.

Appendix B: Raw Data

Complete trial-level data including all model responses are available at the study repository.

Generated by Research Pipeline v0.4.0