Lost in Execution: On the Multilingual Robustness of Tool Calling in Large Language Models

Zheng Luo[1] **T Pranav Kutralingam**[2] **Ogochukwu N. Okoani**[2]
Wanpeng Xu[2] **Hua Wei**[2]* **Xiyang Hu**[2]*
[1]University of Southern California [2]Arizona State University
luozheng@usc.edu, {tknolast,ookoani,wanpeng.xu,hua.wei,xiyanghu}@asu.edu

Abstract

Large Language Models (LLMs) are increasingly deployed as agents that invoke external tools through structured function calls. While recent work reports strong tool-calling performance under standard English-centric evaluations, the robustness of tool calling under multilingual user interactions remains underexplored. In this work, we introduce MLCL, a diagnostic benchmark, and conduct a systematic evaluation of multilingual tool calling across Chinese, Hindi, and the low-resource language Igbo. Through fine-grained error analysis, we show that many failures occur despite correct intent understanding and tool selection. We identify *parameter value language mismatch* as a dominant failure mode, where models generate semantically appropriate parameter values in the user's language, violating language-invariant execution conventions. We further evaluate several inference-time system strategies and find that while these strategies substantially reduce language-induced execution errors, none of them can fully recover English-level performance.

1 Introduction

Large Language Models (LLMs) are increasingly deployed as agents that interact with external tools and services, rather than as standalone conversational systems (Parisi et al., 2022). Through tool calling, an LLM can invoke structured APIs to retrieve information, perform computations, or trigger downstream actions, enabling reliable execution beyond free-form text generation (Schick et al., 2023; Liu et al., 2025). Accordingly, recent work has focused on improving tool-calling performance through supervised fine-tuning (Tang et al., 2023) and reinforcement learning (Qian et al., 2025), and existing benchmarks report strong performance under standard evaluation settings (Chen et al., 2024b).

Figure 1: Multilingual tool-calling failures stem from execution-level parameter mismatches. The same user intent expressed in English, Chinese, Hindi, and Igbo yields semantically appropriate tool calls that become non-executable when parameter values violate English-only execution conventions, a failure mode we refer to as *parameter value language mismatch*.

However, most evaluations implicitly assume that user queries are expressed *only* in English. In practice, LLM-based agents are often exposed to multilingual user queries while relying on shared, language-invariant tool interfaces. In such settings, robustness is not only about understanding user intent, but also about maintaining executable behavior at the language–tool boundary. Despite its practical importance, the impact of linguistic context on tool-calling reliability remains largely unexamined in existing tool-calling benchmarks (Wang et al., 2023).

*Corresponding authors.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 44059–44077

In this work, we show that crossing linguistic boundaries exposes systematic failure patterns that are not captured by standard accuracy metrics. When user queries are expressed in non-English languages, models frequently generate tool calls that are semantically appropriate yet operationally invalid, as illustrated in Figure 1. In a typical tool-calling setup, an LLM produces a structured function call consisting of a function name (e.g., `get_weather()`) and a set of parameter keys (e.g., *"location"* and *"days"*) with values (e.g., *"New York" and "7"*) that are passed as arguments to the execution interface. While the predicted function and the underlying semantics of the arguments are often correct, parameter values are expected to conform to execution-level conventions, such as using English string identifiers. We refer to failures where models directly copy non-English expressions from the user query into parameter values as *parameter value language mismatch*. Such mismatches render correct tool calls non-executable, exposing failures that arise at the language–execution boundary rather than from intent misunderstanding.

To systematically study this phenomenon, we introduce a diagnostic multilingual benchmark for tool calling, MLCL, by extending a commonly-used English dataset, the Berkeley Function Calling Leaderboard (BFCL) (Patil et al., 2025). We focus on isolating the effect of query language on tool-calling behavior through controlled query language composition and semantic perturbations, coupled with a fine-grained error taxonomy that distinguishes execution-level violations from semantic errors. Our evaluation covers Chinese, Hindi, and Igbo, enabling analysis across high-resource and low-resource language settings.

Based on this diagnostic framework, we further examine whether simple inference-time system strategies, including partial translation, explicit prompting, and pre- or post-translation, can mitigate language-induced execution errors. Across models and languages, these strategies reduce certain error types but fail to consistently recover English-level performance, suggesting that multilingual tool-calling robustness is primarily a system- and interface-level challenge rather than a limitation of intent understanding alone. Together, our findings highlight the need to better align natural-language interaction with execution conventions in globally deployed LLM-based agents.

In summary, the contributions of this paper are as follows:

- We introduce MLCL, a diagnostic benchmark for tool-calling robustness under multilingual user queries, covering Chinese, Hindi, and the low-resource language Igbo. The benchmarking dataset characterizes systematic robustness in multilingual tool calling under controlled and interpretable settings.
- Through detailed error analysis, we identify *parameter value language mismatch* as a dominant failure mode in multilingual tool calling, despite correct intent understanding and tool selection.
- We conduct a fine-grained error analysis that separates execution-level violations from semantic errors, revealing systematic differences in error distributions across high-resource and low-resource languages. For high-resource languages like Chinese and Hindi, the major cause of the errors is the tool calling convention and information loss during translation, rather than difficulty in user query comprehension; while for the low-resource Igbo language, the confusion in the user query semantics takes up a larger proportion of errors.
- We empirically evaluate several simple inference-time system strategies and show that while they substantially reduce language-induced errors, they cannot fully restore English-level performance, highlighting the role of system- and interface-level conventions in multilingual tool calling.

2 Related Work

Tool Calling and Tool Learning Recent work has made substantial progress in enabling large language models to interact with external tools through structured APIs. Early studies, such as Gorilla (Patil et al., 2024), highlighted issues such as API hallucination, where models produce syntactically valid but operationally incorrect calls. To improve reliability, supervised fine-tuning (SFT) approaches (Chen et al., 2023; Zeng et al., 2024; Chen et al., 2024c; Acikgoz et al., 2025) and reinforcement learning methods (Qian et al., 2025; Feng et al., 2025; Wei et al., 2025; Wu et al., 2025; Zhang et al., 2026; Chen et al., 2026) have been proposed to align model outputs with tool-calling formats and execution constraints.

Evaluation in this line of work is typically conducted under standardized, predominantly English-language settings (Qin et al., 2024; Patil et al., 2025). Benchmarks such as BFCL (Patil et al., 2025) assess tool selection and parameter accuracy assuming language-consistent execution environ-

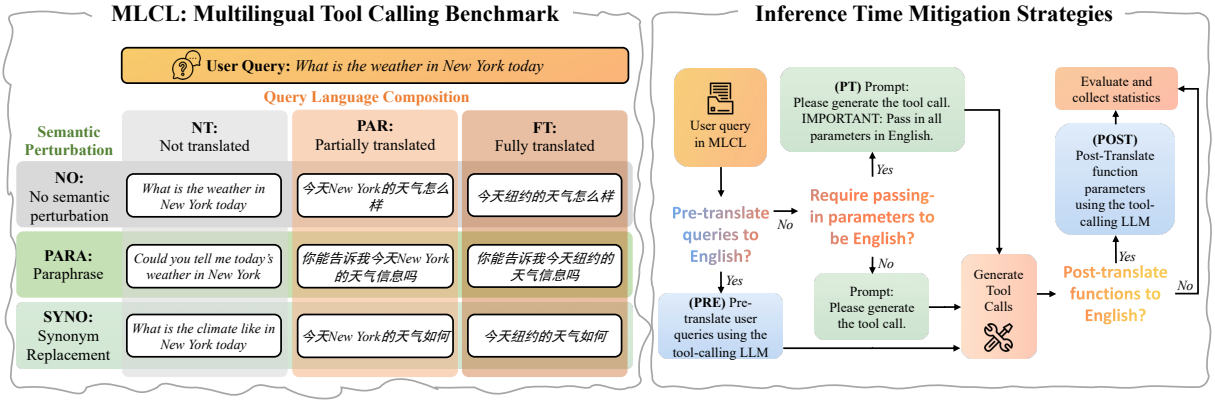


Figure 2: **Diagnostic design of the Multilingual Tool-Calling (MLCL) Benchmark and inference-time mitigation strategies.** The benchmark systematically varies *query language composition* (NT, PAR, FT) and *semantic perturbations* (NO, PARA, SYNO) to isolate multilingual execution failures under a fixed, language-invariant tool interface. This design exposes cases where tool calls are semantically correct but operationally invalid due to execution-level violations, such as parameter value language mismatch. The right panel summarizes inference-time mitigation strategies (PT, PRE, POST) evaluated in this work, which reduce some multilingual errors but do not fully eliminate execution-level gaps.

ments. While these studies demonstrate strong performance under standard conditions, they largely abstract away linguistic variation in user queries. As a result, how language differences interact with execution-level conventions in tool calling remains underexplored. Our work complements this literature by focusing on multilingual robustness and by analyzing execution failures induced by language shifts, rather than proposing new training objectives or architectures.

Robustness in Large Language Models Robustness of large language models has been extensively studied under surface-level perturbations such as paraphrasing, synonym substitution, and distributional shifts (Feng et al., 2021; Zhou et al., 2024; Kumar and Mishra, 2025; Rabinovich and Anaby Tavor, 2025). These studies show that even minor linguistic variations can significantly affect model behavior, motivating robustness benchmarks beyond aggregate accuracy metrics. However, most robustness analyses focus on free-form generation or classification tasks, where failures are primarily semantic. In contrast, tool calling introduces execution-level constraints: outputs must satisfy strict formatting and parameter conventions to be operationally valid. Our work extends robustness analysis to this setting by introducing a fine-grained error taxonomy that distinguishes semantic understanding errors from execution-level violations, revealing multilingual failure modes that are obscured by standard accuracy metrics.

Multilingual Evaluation of Language Models Multilingual evaluation benchmarks assess cross-lingual understanding, reasoning, and instruction following across diverse languages (Ruder et al., 2021; Ahuja et al., 2023; Shi et al., 2023; Chen et al., 2024a; Xu and Hu, 2026; Zhou et al., 2026). While these studies show that large language models can preserve semantic reasoning across languages, prior work on multilingual settings with structured outputs suggests that maintaining cross-lingual consistency remains challenging even when translation quality is high (Moradshahi et al., 2023). However, existing multilingual evaluations rarely consider scenarios where model outputs must interface with external, language-sensitive systems such as tools or APIs. Our work addresses this gap by studying multilingual robustness in tool calling and identifying *parameter value language mismatch* as a distinct execution-level failure mode.

3 Multilingual Tool-Calling Benchmark

To characterize multilingual tool-calling failures under controlled, interpretable settings, rather than treating multilinguality as a simple dataset extension, we introduce a diagnostic benchmark focusing on failure modes that arise at the interface between natural-language input and a language-invariant execution environment (Figure 2).

3.1 Design Goals and Diagnostic Scope

Our design goal is to separate errors caused by *query-language variation* from those caused by *execution-interface conventions*. To this end, the

benchmark dataset is structured along two orthogonal diagnostic dimensions: *Query Language Composition* and *Semantic Perturbation Design*. Query Language Composition controls how non-English content is introduced in the user query, while Semantic Perturbation Design applies surface-form variations that preserve intent. Together, these dimensions define a compact diagnostic space for attributing multilingual tool-calling failures to language understanding, execution-interface mismatches, or parameter realization.

3.2 Dataset Construction

Our benchmark is constructed by extending an existing English tool-calling dataset into a multilingual diagnostic suite, while keeping the execution interface fixed. This design ensures that any observed performance degradation can be attributed to changes in the natural-language input, rather than differences in tool schemas, APIs, or evaluation criteria.

3.2.1 Base Tasks and Execution Interface

We formulate tool calling as a structured generation problem at the interface between natural-language understanding and programmatic execution. Each task consists of a user query and a set of candidate tools, where each tool is specified by a function name and a fixed parameter schema.

Given a query and tool descriptions, the model generates a tool call by selecting a function name and producing concrete parameter values. Correct execution requires adherence to execution-level constraints such as exact parameter keys (e.g., *"location"* in Figure 1) and surface-form conventions for parameter values (e.g., *"New York"* in Figure 1). The execution interface is kept language-invariant and English-only throughout this work. The original English queries define the **NT** (Not Translated) reference setting. All the later multilingual variants modify only the natural-language query while keeping the execution interface fixed.

3.2.2 Query Language Composition

To model multilingual user interactions, we construct translated variants of each query while preserving the execution interface. Only the natural-language query text is modified; function names, parameter keys, and tool descriptions are not translated. This isolates the effect of query language on tool selection and parameter realization.

In the **FT** (Fully Translated) setting, the entire user query is translated into a target language. In the **PAR** (Partially Translated) setting, ground-truth parameter values remain in English while the surrounding context is translated, yielding mixed-language queries that reduce parameter-language mismatch while preserving multilingual context. Translations are generated by GPT-5 and manually verified to ensure semantic equivalence with the original English queries, as detailed in Appendix A.4. Verification focuses on intent preservation rather than literal word-level correspondence, reflecting realistic multilingual usage.

3.2.3 Semantic Perturbation Design

To assess whether multilingual execution failures are sensitive to benign surface-form variation, we introduce semantic perturbations that preserve intent while modifying wording. These perturbations are applied consistently across the English and translated datasets. We consider two perturbation types: **PARA** generates paraphrased versions of each query with altered phrasing but unchanged meaning. **SYNO** applies synonym substitutions to individual words where appropriate. Both perturbations are generated using GPT-5 and manually reviewed to maintain semantic consistency. For partially translated queries, perturbations are applied without explicitly protecting English parameter strings, allowing semantic variation to naturally interact with mixed-language inputs.

3.3 Experimental Protocol

This section describes how the benchmark defined above is instantiated for evaluation. It varies only the query language composition and semantic perturbations, enabling controlled comparison across languages, models, and settings.

Base Dataset We adopt the BFCL V4 (Patil et al., 2025) as the base benchmark in English since it provides tool-calling tasks with strict execution-level ground truth, making it well-suited for analyzing failures that arise from parameter realization and interface compliance. To avoid confounding factors such as dialogue context or live execution, we use the `BFCL_v4_multiple.json` subset, which contains single-turn queries paired with multiple candidate functions.

Language Evaluated Besides English, we evaluate three languages with distinct linguistic properties and resource availability: Chinese, Hindi,

Table 1: Models evaluated in the multilingual tool-calling benchmark. We include proprietary and open-source models from multiple families and scales to ensure that observed multilingual tool-calling failures are evaluated across diverse model architectures.

Model Family	Model Name
GPT-5	GPT-5, GPT-5 mini, GPT-5 nano
DeepSeek	DeepSeek V3.2
Llama 3.1	meta-llama/Llama-3.1-8B-Instruct
	meta-llama/Llama-3.1-70B-Instruct
Qwen 3	Qwen/Qwen3-8B
	Qwen/Qwen3-14B
	Qwen/Qwen3-30B-A3B
	Qwen/Qwen3-32B
	Qwen/Qwen3-Next-80B-A3B-Instruct
Granite 4	ibm-granite/granite-4.0-h-tiny
	ibm-granite/granite-4.0-h-small

and Igbo. Chinese represents a high-resource language with logographic writing, Hindi introduces richer morphology and more flexible word order, and Igbo serves as a low-resource language with limited representation in tool-calling training data. This selection is intended to test whether observed multilingual failure patterns generalize across typologically diverse languages.

Models Evaluated We evaluate a diverse set of large language models with explicit tool-calling capabilities, including both proprietary and open-source systems. The selected models span multiple model families and scales, allowing us to examine whether multilingual tool-calling robustness varies with architecture and capacity. All models are evaluated using their officially supported tool-calling interfaces and recommended decoding configurations. Model-specific input and output formats are summarized in Appendix A.10.

Evaluation Metrics and Error Attribution Evaluation follows the BFCL protocol (Patil et al., 2025), which requires exact matching of function names, parameter keys, and parameter values. While strict surface-form matching penalizes benign variation, it directly reflects whether a generated tool call can be executed without additional system intervention.

To support multilingual analysis, we extend BFCL's evaluation by explicitly separating semantic correctness from language conformity in parameter values. As illustrated in Figure 3, errors are organized by severity. At the most severe level are syntax and function-level errors, which prevent execution due to malformed outputs or schema violations. A central focus of this work is *parameter value language mismatch*, where parameter values are generated in a non-English language despite correct intent understanding and tool selection. We further distinguish these cases by semantic correctness, separating execution failures caused purely by language mismatch from those involving incorrect values. This allows us to distinguish errors caused by incorrect intent understanding or argument selection from those arising solely due to violations of execution-level language conventions. Results are reported using **overall error rate** together with a structured **error breakdown**, enabling fine-grained analysis across query language composition and semantic perturbation settings. Detailed definitions and examples for each category are provided in Appendix A.2.

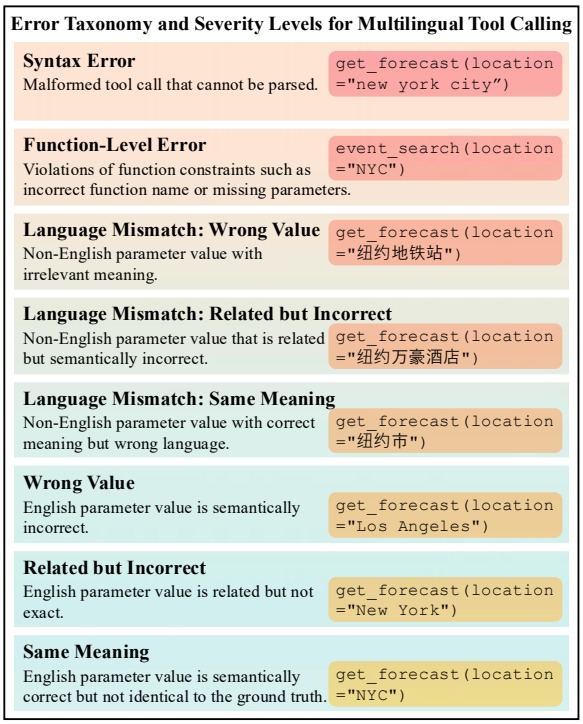


Figure 3: Error taxonomy and severity levels used for evaluating multilingual tool calling. Error categories are ordered from most severe (top) to least severe (bottom), with illustrative examples for each category.

3.4 Results

We analyze multilingual tool-calling behavior across query language composition and semantic perturbation settings, using the execution-oriented error taxonomy illustrated in Figure 3. Results are reported in terms of error composition and severity rather than overall accuracy, reflecting whether failures arise from execution-interface violations or semantic misunderstanding.

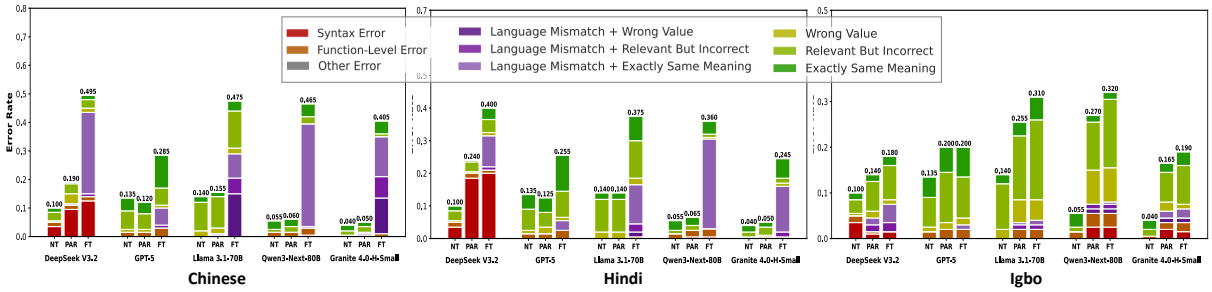


Figure 4: Error distributions for five representative models under English (NT), partially translated (PAR), and fully translated (FT) queries in Chinese, Hindi, and Igbo. Across languages and model families, moving from NT to FT systematically increases execution-level errors, driven primarily by *parameter value language mismatch* (in purple color), while PAR substantially reduces these violations. The consistency of this trend across models indicates a shared failure mechanism at the language–execution interface rather than model-specific weaknesses.

Introducing fully non-English queries substantially increases execution-level errors. Figure 4 shows the error breakdown for five representative models when moving from the English reference setting (NT) to fully translated queries (FT). Across all three languages, the FT setting leads to a pronounced increase in execution failures, dominated by parameter value language mismatch. In the FT setting, models frequently copy non-English tokens from the user query directly into parameter values, violating the English-only execution interface. In most cases, these values remain semantically correct, indicating that intent understanding and tool selection have succeeded and that failures arise primarily at the language–execution boundary.

The composition of multilingual tool-calling errors differs systematically across languages. Although the increase in execution-level errors under FT is consistent across languages, the dominant error types vary. As shown in Figure 4, parameter value language mismatch is most prevalent for Chinese, followed by Hindi, and is least frequent for Igbo. This pattern suggests that models are more likely to reuse query tokens from high-resource languages as parameter values, while avoiding doing so for lower-resource languages. As a result, lower-resource languages exhibit fewer language-mismatch errors but a higher proportion of errors related to semantic misunderstanding, indicating that execution-level mismatch and language comprehension contribute differently across languages.

Partial translation isolates execution-interface violations from language understanding errors. For several models, including GPT-5 and Llama 3.1–70B, the partially translated (PAR) setting exhibits fewer execution-level errors than FT setting, and in some cases matches or outperforms the English reference. By preserving English parameter strings while translating the surrounding context, PAR removes parameter value language mismatch without substantially altering the semantic content of the query. These results indicate that, once execution-interface violations are controlled for, many models can interpret non-English queries with comparable reliability to English ones.

Semantic perturbations exacerbate execution errors when strict surface-form matching is required. Figure 5 summarizes the impact of paraphrasing (PARA) and synonym substitution (SYNO) across query language composition settings. Additional results can be found in Appendix B.3. In the English reference setting, semantic perturbations substantially increase execution failures because altered surface forms no longer match expected parameter values. In contrast, semantic perturbations have limited additional effect in fully translated settings, where execution errors are already dominated by language mismatch. Partially translated settings exhibit intermediate sensitivity: perturbations can replace preserved English parameter strings with non-English equivalents, reintroducing execution-level violations. Overall, semantic noise amplifies multilingual tool-calling failures primarily when strict surface-form conformity is required.

4 Inference Time Mitigation Strategies

4.1 Motivation and Scope

As shown in Section 3.4, most multilingual tool-calling failures originate from execution-level language mismatch rather than semantic misunderstanding: Models often select the correct tool and generate semantically appropriate arguments, but fail to conform to the English-only execution inter-

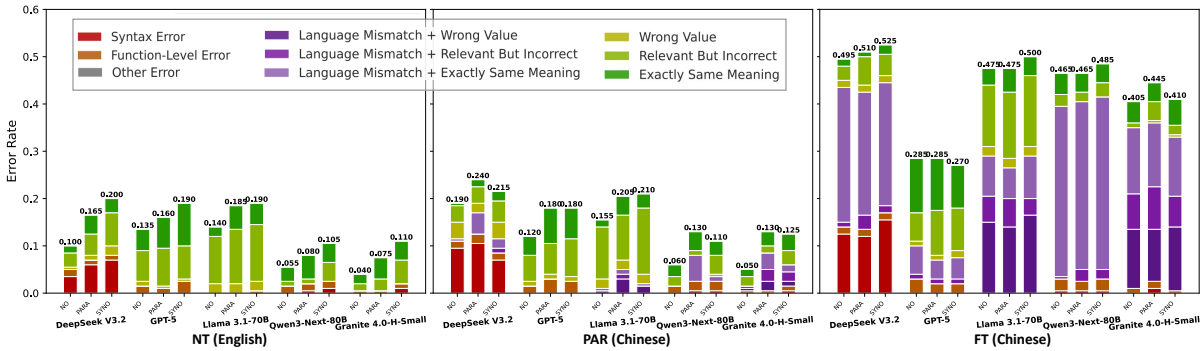


Figure 5: Evaluation of semantic perturbations for five representative models on the Chinese dataset under paraphrasing (PARA) and synonym substitution (SYNO) across NT, PAR, and FT settings. Semantic perturbations substantially increase errors when exact English parameter surface forms are required (NT), but have limited additional impact in fully translated (FT) queries dominated by parameter value language mismatch (in purple color). The partially translated (PAR) setting exhibits intermediate sensitivity, indicating an interaction between semantic variation and execution-level language constraints.

face when queries are non-English. This raises a practical question: *can simple inference-time strategies reduce such failures without retraining or fine-tuning models?* We focus on lightweight interventions that are compatible with deployed systems and do not require changes to model weights. All mitigation strategies are evaluated under the same task abstraction and error taxonomy as in the main benchmark, enabling direct comparison with the NT, PAR, and FT settings.

4.2 Mitigation Methods

We consider three lightweight inference-time mitigation strategies that target execution-level language mismatch at different stages of the tool-calling pipeline, as shown in Figure 2. All strategies operate without modifying model parameters and are compatible with deployed systems. Implementation details, prompts, and translation procedures for all mitigation strategies are provided in Appendix A.3.

• **Prompt-Level Instruction (PT)** PT fully translates the user query and explicitly instructs the model to output parameter values in English. This strategy tests whether execution-interface conventions can be enforced through natural-language instructions alone.

• **Pre-Translation of User Queries (PRE)** PRE translates non-English user queries into English before tool calling. By normalizing the input, this strategy removes multilingual variation from the tool-calling step and serves as an upper bound on mitigation through input preprocessing.

• **Post-Translation of Parameter Values (POST)** POST translates generated parameter values into English after tool calling but before execution. This strategy directly targets parameter value language mismatch while preserving the model's original tool selection and argument structure.

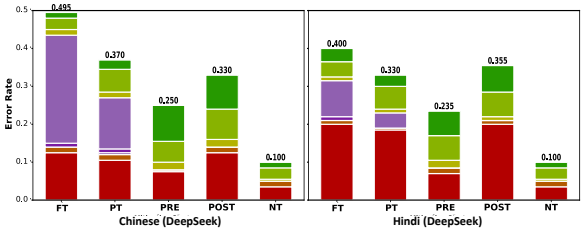


Figure 6: Evaluation of inference-time mitigation strategies for DeepSeek V3.2 on Chinese and Hindi datasets, including explicit prompting (PT), pre-translation (PRE), and post-translation (POST). Across both languages, these strategies reduce parameter value language mismatch errors but fail to recover English-only (NT) performance. PRE generally outperforms POST due to access to the full query context, while PT exhibits inconsistent compliance, indicating persistent execution-level gaps beyond simple translation fixes.

4.3 Evaluation of Mitigation Strategies

Inference-time mitigation strategies reduce execution-interface violations but do not recover English-level performance. Figure 6 shows representative results on Chinese queries with different mitigation strategies. Similar trends are also observed across different models and languages. Prompt-level instruction (PT) reduces parameter value language mismatch relative to the fully translated setting, but compliance is inconsistent, and residual mismatches remain common. Pre-translation (PRE) and post-translation (POST) further reduce language mismatch errors, with PRE generally more effective due to its access to the full query context. However, none of these strate-

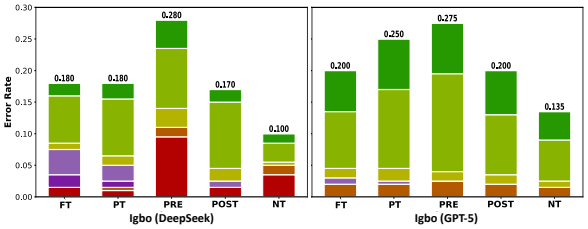


Figure 7: Inference-time mitigation effects on a low-resource language (Igbo) for DeepSeek V3.2 and GPT-5. Unlike Chinese and Hindi, translation-based mitigations (PT, PRE, POST) provide limited benefit and can increase errors, as parameter value language mismatch is rare even without explicit prompting. Instead, remaining failures are dominated by query understanding errors, indicating that mitigation strategies targeting execution-level language mismatches are less effective for low-resource languages.

gies eliminates execution failures. Translation introduces semantic drift and surface-form normalization, replacing language mismatch with new execution-level errors under strict matching. We include a case study in Appendix B.1 for better understanding.

Mitigation behavior differs systematically for low-resource languages. Low-resource languages such as Igbo exhibit mitigation behavior that differs from that of higher-resource languages like Chinese and Hindi. As shown in Figure 7, translation-based mitigation strategies often provide limited benefit for Igbo and can even degrade performance. Unlike higher-resource languages, parameter value language mismatch is rare for Igbo even without explicit prompting. This suggests that models are less likely to directly copy low-resource language tokens from the query into parameter values, thereby reducing execution-interface violations. As a result, mitigation strategies that primarily target language mismatch, such as PT and POST, have limited room to improve performance. The remaining failures are instead dominated by query understanding errors, including unstable semantic grounding and imprecise mapping between natural-language expressions and the target parameter schema.

5 Discussion

This work analyzes multilingual tool-calling failures under controlled execution settings and demonstrates that the multilingual degradation is primarily driven by execution-interface violations rather than semantic misunderstandings.

Execution interfaces are a central bottleneck. Across models and languages, non-English queries increase execution-level errors, most notably parameter value language mismatch. In many cases, models select the correct tool and generate semantically appropriate arguments, yet fail to satisfy strict surface-form requirements imposed by the execution interface. This indicates a systematic mismatch between flexible natural-language generation and rigid programmatic constraints.

Inference-time mitigation is helpful but insufficient. Prompting and translation-based strategies can reduce language mismatch errors, but none recover English-level performance. While these methods alleviate direct language violations, they introduce semantic drift and normalization effects that lead to new execution failures. This suggests that inference-time patching can mitigate symptoms but cannot fully resolve the underlying issue.

Implications for deployment with global users. For real-world systems, these results imply that multilingual users may experience reliability gaps even when intent understanding succeeds. Without explicit handling of execution-interface constraints, tool-calling LLMs risk failures for non-English queries, limiting global deployment robustness.

Overall, our findings suggest that improving multilingual tool calling requires not only stronger language understanding, but also execution-aware system design that aligns natural-language variability with programmatic interfaces.

6 Conclusion

This paper investigates multilingual tool calling under controlled execution settings and shows that performance degradation is driven primarily by execution-interface violations rather than failures in intent understanding. Across models and languages, fully translated queries substantially increase execution errors, with parameter value language mismatch emerging as a dominant failure mode. We further show that inference-time mitigation strategies can reduce specific classes of errors but do not recover English-level performance, often introducing new execution failures through semantic drift or surface-form variation. These findings highlight multilingual tool calling as a system-level challenge that cannot be addressed by inference-time interventions alone and motivate more careful alignment between language generation and execution interfaces.

Acknowledgment

The work was partially supported by NSF award #2442477 and #2550203. We thank Amazon Research Awards, Cisco Faculty Research Awards, and Toyota Faculty Research Awards. The authors acknowledge Google and OpenAI for providing us with API credits and Research Computing at Arizona State University for providing computing resources. The views and conclusions in this paper are those of the authors and should not be interpreted as representing any funding agencies.

Limitations

This study has several limitations that should be considered when interpreting the results. First, our evaluation is based on the vanilla subset of the Berkeley Function Call Leaderboard to focus on single-turn tool-calling scenarios with predefined function interfaces. This choice allows us to isolate execution-level effects under controlled conditions, but the results should be interpreted within this single-turn setting. Second, although we examine three typologically diverse languages, including a low-resource language, the set of languages remains limited and does not cover all linguistic families or writing systems. Our goal is not exhaustive multilingual coverage, but to identify systematic patterns that emerge across representative high-resource and low-resource languages. Third, our definition of execution correctness assumes that tool interfaces expect parameter values in English. This reflects a common design choice in current tool-calling systems and benchmarks, and our analysis focuses on understanding model behavior under this convention rather than advocating a particular interface design. Fourth, the inference-time strategies studied in this paper are intentionally simple. They are used as diagnostic probes to help identify the sources of multilingual failures, rather than as optimized mitigation methods. Finally, although we analyze model scale and architecture effects across several model families, our conclusions are constrained by the specific models evaluated and should not be interpreted as claims about all future model designs.

Ethical Considerations

This work evaluates multilingual tool calling by LLMs under a controlled, English-only execution interface, using translated and perturbed versions of existing benchmark queries (Chinese, Hindi, Igbo) without collecting user data or conducting human-subject studies. The primary risk is that tool-calling failures in non-English settings can cause silent non-executable calls or incorrect downstream actions, which may disproportionately affect non-English users; our contribution is a diagnostic taxonomy and mitigation analysis intended to help system builders detect and reduce such execution-level errors, although stronger tool-calling can also increase the effectiveness of automation if connected to high-impact tools. Limitations include partial language coverage and possible translation artifacts that can introduce semantic drift; we report language-specific error patterns to avoid overgeneralization. AI-assisted tools were used to improve the writing of this paper.

References

Emre Can Acikgoz, Jeremiah Greer, Akul Datta, Ze Yang, William Zeng, Oussama Elachqar, Emmanouil Koukoumidis, Dilek Hakkani-Tür, and Gokhan Tur. 2025. Can a single model master both multi-turn conversations and tool use? coalm: A unified conversational agentic language model. *Preprint*, arXiv:2502.08820.

Kabir Ahuja, Harshita Diddee, Rishav Hada, Millicent Ochieng, Krithika Ramesh, Prachi Jain, Akshay Nambi, Tanuja Ganu, Sameer Segal, Mohamed Ahmed, Kalika Bali, and Sunayana Sitaram. 2023. MEGA: Multilingual evaluation of generative AI. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4232–4267, Singapore. Association for Computational Linguistics.

Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. 2023. Fireact: Toward language agent fine-tuning. *Preprint*, arXiv:2310.05915.

Nuo Chen, Zinan Zheng, Ning Wu, Ming Gong, Dongmei Zhang, and Jia Li. 2024a. Breaking language barriers in multilingual mathematical reasoning: Insights and observations. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 7001–7016, Miami, Florida, USA. Association for Computational Linguistics.

Tiejin Chen, Xiaoou Liu, Vishnu Nandam, Kuan-Ru Liou, and Hua Wei. 2026. Conformal feedback alignment: Quantifying answer-level reliability for robust llm alignment. *arXiv preprint arXiv:2601.17329*.

Zehui Chen, Weihua Du, Wenwei Zhang, Kuikun Liu, Jiangning Liu, Miao Zheng, Jingming Zhuo, Songyang Zhang, Dahua Lin, Kai Chen, and 1 others. 2024b. T-eval: Evaluating the tool utilization capability of large language models step by step. In

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 9510–9529.

Zehui Chen, Kuikun Liu, Qiuchen Wang, Wenwei Zhang, Jiangning Liu, Dahua Lin, Kai Chen, and Feng Zhao. 2024c. Agent-FLAN: Designing data and methods of effective agent tuning for large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 9354–9366, Bangkok, Thailand. Association for Computational Linguistics.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. 2025. Retool: Reinforcement learning for strategic tool use in llms. *Preprint*, arXiv:2504.11536.

Steven Y. Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. 2021. A survey of data augmentation approaches for nlp. *Preprint*, arXiv:2105.03075.

Pankaj Kumar and Subhankar Mishra. 2025. Robustness in large language models: A survey of mitigation strategies and evaluation metrics. *Preprint*, arXiv:2505.18658.

Xiaoou Liu, Tiejin Chen, Longchao Da, Chacha Chen, Zhen Lin, and Hua Wei. 2025. Uncertainty quantification and confidence calibration in large language models: A survey. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 6107–6117.

Mehrad Moradshahi, Tianhao Shen, Kalika Bali, Monojit Choudhury, Gael de Chalendar, Anmol Goel, Sungkyun Kim, Prashant Kodali, Ponnurangam Kumaraguru, Nasredine Semmar, Sina Semnani, Jiwon Seo, Vivek Seshadri, Manish Shrivastava, Michael Sun, Aditya Yadavalli, Chaobin You, Deyi Xiong, and Monica Lam. 2023. X-RiSAWOZ: High-quality end-to-end multilingual dialogue datasets and few-shot agents. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 2773–2794, Toronto, Canada. Association for Computational Linguistics.

Aaron Parisi, Yao Zhao, and Noah Fiedel. 2022. Talm: Tool augmented language models. *Preprint*, arXiv:2205.12255.

Shishir G. Patil, Huanzhi Mao, Charlie Cheng-Jie Ji, Fanjia Yan, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. 2025. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2024. Gorilla: Large language model connected with massive apis. In *Advances in Neural Information Processing Systems*, volume 37, pages 126544–126565. Curran Associates, Inc.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. 2025. Toolrl: Reward is all tool learning needs. *Preprint*, arXiv:2504.13958.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, dahai li, Zhiyuan Liu, and Maosong Sun. 2024. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *The Twelfth International Conference on Learning Representations*.

Ella Rabinovich and Ateret Anaby Tavor. 2025. On the robustness of agentic function calling. In *Proceedings of the 5th Workshop on Trustworthy NLP (TrustNLP 2025)*, pages 298–304, Albuquerque, New Mexico. Association for Computational Linguistics.

Sebastian Ruder, Noah Constant, Jan Botha, Aditya Siddhant, Orhan Firat, Jinlan Fu, Pengfei Liu, Junjie Hu, Dan Garrette, Graham Neubig, and Melvin Johnson. 2021. XTREME-R: Towards more challenging and nuanced multilingual evaluation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10215–10245, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551.

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, Dipanjan Das, and Jason Wei. 2023. Language models are multilingual chain-of-thought reasoners. In *The Eleventh International Conference on Learning Representations*.

Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, Boxi Cao, and Le Sun. 2023. Toolalpaca: Generalized tool learning for language models with 3000 simulated cases, 2023. *URL https://arxiv.org/abs/2306.05301*.

Xingyao Wang, Zihan Wang, Jiateng Liu, Yangyi Chen, Lifan Yuan, Hao Peng, and Heng Ji. 2023. Mint: Evaluating llms in multi-turn interaction with tools and language feedback. *arXiv preprint arXiv:2309.10691*.

Yifan Wei, Xiaoyan Yu, Yixuan Weng, Tengfei Pan, Angsheng Li, and Li Du. 2025. Autotir: Autonomous tools integrated reasoning via reinforcement learning. *Preprint*, arXiv:2507.21836.

Mingyuan Wu, Jingcheng Yang, Jize Jiang, Meitang Li, Kaizhuo Yan, Hanchao Yu, Minjia Zhang, Chengxiang Zhai, and Klara Nahrstedt. 2025. Vtool-r1: Vlms learn to think with images via reinforcement learning on multimodal tool use. *Preprint*, arXiv:2505.19255.

Jiayi Xu and Xiyang Hu. 2026. Language shapes mental health evaluations in large language models. *Preprint*, arXiv:2603.06910.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. 2024. AgentTuning: Enabling generalized agent abilities for LLMs. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3053–3077, Bangkok, Thailand. Association for Computational Linguistics.

Dengjia Zhang, Xiaoou Liu, Lu Cheng, Yaqing Wang, Kenton Murray, and Hua Wei. 2026. Selaur: Self evolving llm agent via uncertainty-aware rewards. *arXiv preprint arXiv:2602.21158*.

Xiaolin Zhou, Zheng Luo, Yicheng Gao, Qixuan Chen, Xiyang Hu, Yue Zhao, and Ruishan Liu. 2026. Fairness or fluency? an investigation into language bias of pairwise llm-as-a-judge. *Preprint*, arXiv:2601.13649.

Yue Zhou, Chenlu Guo, Xu Wang, Yi Chang, and Yuan Wu. 2024. A survey on data augmentation in large model era. *Preprint*, arXiv:2401.15422.

A Appendix

A.1 Code and Dataset

All code and datasets used in this paper are available at `https://anonymous.4open.science/r/multilingual_robustness_tool_calling-CA44`.

A.2 Detailed Error Taxonomy

To characterize multilingual tool-calling failures beyond binary correctness, we organize errors according to their impact on execution rather than semantic adequacy alone. As illustrated in Figure 3, error categories are ordered by severity, reflecting whether a generated tool call can be parsed, executed, or reliably recovered in a realistic tool-calling system.

At the most severe end are **syntax errors**, where malformed outputs cannot be parsed into valid function invocations. These failures prevent any execution attempt and reflect violations of the output format rather than language understanding. **Function-level errors** are syntactically valid but violate the tool schema, such as incorrect function names or missing parameters. While closer to execution, they still result in non-executable tool calls.

A central focus of this work is *parameter value language mismatch*, which we further decompose by semantic correctness. (a) **Language Mismatch + Wrong Value:** In the most severe cases, parameter values are expressed in a non-English language and are semantically unrelated to the ground truth. (b) **Language Mismatch + Relevant but Incorrect:** Less severe variants preserve partial relevance but do not precisely match the intended meaning. They typically arise when models mirror the user's language while producing underspecified values, reflecting combined language and semantic imprecision. (c) **Language Mismatch + Same Meaning:** Another less severe variant shows non-English parameter values that are conceptually related but do not precisely match the intended meaning. They typically arise when models mirror the user's language while producing underspecified values, reflecting combined language and semantic imprecision. Together, these errors isolate failures at the language–execution interface, as intent understanding and tool selection have already succeeded.

Finally, we consider parameter value errors without language mismatch, ranging from semantically incorrect and irrelevant values (**Wrong**

Value) to semantically relevant but incorrect values (**Relevant but Incorrect**), values that are semantically the same but differ in surface form (**Exactly Same Meaning**), thus fail strict BFCL surface-form matching.

A.3 Details of Inference-Time Mitigation Strategies

A.3.1 Prompt-Level Instruction (PT)

Prompt-Level Instruction (PT) fully translates the user query into the target language and explicitly instructs the model to output parameter values in English. This strategy probes whether models can follow execution-level conventions through natural-language instructions alone, without altering the input query or post-processing the output.

A.3.2 Pre-Translation (PRE)

Pre-Translation of User Queries (PRE) translates non-English user queries into English before tool calling. To avoid confounding the effect of translation quality with model capability, the same language model is used for both translation and tool calling. PRE removes multilingual input entirely from the tool-calling step, serving as an upper bound on what can be achieved through input normalization.

A.3.3 Post-Translation (POST)

Post-Translation of Parameter Values (POST) translates generated parameter values into English after tool calling but before execution. This strategy directly targets parameter value language mismatch while preserving the model's original tool selection and argument structure. As with PRE, translation is performed by the same model to control for language proficiency.

A.4 Manual Verification Process

We have group members who speak Chinese, Hindi, and Igbo.

For each entry of the fully translated dataset, we verify that the translated user queries have the same meaning as the original ones, but we do not enforce the wordings of the queries so that a tool-calling LLM intelligent enough always has a chance to hit the ground truth. For example, "queen-size bed" may not have a strict equivalence in many languages; therefore, we may judge any translation that is semantically equivalent to "a large bed" as appropriate. As a consequence, we do not expect the tool-calling LLM to recover exactly "queen-sized bed", but if the LLM fails to do so, we still count it as a failure to meet the ground truth.

For the partially translated dataset, we verify both the semantic invariance and the property that keywords acting as parameter values in the ground truth are kept in English. It is noteworthy that this kind of keyword preserving may not stop LLMs from passing in parameter values that are not in English. For example, if only "A" in "Company A" appears in the ground truth parameter values, we do not also keep the word "Company" in English, although LLMs may pass in "Company A" in the user query's language, causing a language mismatch.

A.5 Sample Tool Call Generation Prompt

We use BFCL's prompt for tool generation, and add slight modifications to it to adapt to the properties of different models (for example, GPT-5 uses "developer" message instead of "system" message). A sample prompt can be found below. Note that the "IMPORTANT: Pass all parameter values in English" prompt only appears in the **PT** setting, as illustrated in Figure 2.

System Prompt:
You are an expert in composing functions. You are given a question and a set of possible functions. Based on the question, you will need to make one or more function/tool calls to achieve the purpose. If none of the functions can be used, point it out. If the given question lacks the parameters required by the function, also point it out.
You should ONLY return function calls in your response. You MUST NOT include any other text, explanations, or direct answers. If you decide to invoke any function(s), you MUST use the provided tools. Do NOT attempt to answer the question directly without using the available functions.
[IMPORTANT: Pass all parameter values in English.]
User Prompt:
[user query]

A.6 Fully Translated Dataset Generation Prompt

Prompt:
Translate the following English question to [target_language]. Provide a natural, fluent translation that maintains the meaning and intent of the original question.

A.7 Partially Translated Dataset Generation Prompt

A sample prompt can be found below. Note that the keywords are extracted from the parameter values in the ground truth.

Prompt:
Translate the following English question to [target_language]. Keep the keywords listed below unchanged (do not translate them).
Question: [question_content]
Keywords to preserve (keep in English): [keywords_str]
Provide only the [target_language] translation.

A.8 Paraphrased Dataset Generation Prompt

Prompt:
You are a helpful assistant helping rephrase user requests, while accurately preserving their meaning, including numbers and names if they exist. Do not answer the requirement; just produce another one that is identical in meaning but is phrased differently. Produce ONLY the rephrased requirement, without further thoughts or explanations.

A.9 Synonym Dataset Generation Prompt

Prompt:
You are a helpful assistant that replaces words with synonyms of similar meaning while maintaining semantic correctness. Your task is to process word by word and replace each word with a synonym if possible.
IMPORTANT RULES: 1. Replace words with appropriate synonyms 2. Maintain the semantic meaning and grammatical structure 3. Do NOT perform general paraphrasing, only synonym replacement 4. Process word by word, not phrase by phrase 5. If a word has no suitable synonym or is a proper noun, keep it unchanged
Produce ONLY the modified text with synonyms, without further thoughts or explanations. Consider the example below:
USER: Can I find the dimensions and properties of a triangle, if it is known that its three sides are 5 units, 4 units and 3 units long?
ASSISTANT: Can I discover the measurements and characteristics of a triangle, if it is known that its three sides are 5 units, 4 units and 3 units long?

A.10 Tool Calling IO Protocol for Tested Model Families

We do not use the official BFCL repository's implementation of tool calling protocol handling, but implement it ourselves to stick to the latest official documentation for each model and make the code base clean and extensible. The following are the documentations we reference to implement the tool, calling format conversion, and output parsing. The detailed implementation can be viewed at Appendix A.1.

GPT-5 Family The official documentation for the GPT-5 tool calling IO protocol can be found at https://platform.openai.com/docs/guides/function-calling?strict-mode=enabled#defining-functions.

DeepSeek V3.2 The official documentation for the DeepSeek V3.2 tool calling IO protocol can be found at https://api-docs.deepseek.com/guides/function_calling.

Qwen 3 Family The official documentation for the Qwen 3 tool calling IO protocol can be found

at https://qwen.readthedocs.io/en/latest/framework/function_call.html.

Llama 3.1 Family The official documentation for the Llama 3.1 tool calling IO protocol can be found at https://huggingface.co/meta-llama/Llama-3.1-70B-Instruct#tool-use-with-transformers.

Granite 4 Family The official documentation for Granite 4 tool calling IO protocol can be found at https://huggingface.co/ibm-granite/granite-4.0-micro.

B Additional Results

B.1 Case Study of Translation-Induced Semantic Drift.

Figure 8 presents a representative example of pre-translation (PRE), where translating the user query alters parameter surface forms through normalization or synonym substitution. Although the translated query preserves the original intent, the resulting tool call fails under strict surface-form matching, illustrating why PRE cannot fully recover English-level performance.

Original User Query:
I want to book a suite with queen size bed for 3 nights in Hilton New York. Can you find the pricing for me?
Ground Truth:
hotel_room_pricing.get(
hotelName= "Hilton New York", roomType= "suite with queen size bed", nights=3)

Fully Translate Query to Chinese:
我想在纽约希尔顿酒店预订一间带**大号床**的套房，住3晚。你能帮我查一下价格吗？
Translated back to English by Qwen3-Next-80B-A3B:
I would like to book a suite with a king-size bed at the Hilton Hotel in New York for three nights. Could you help me check the price?

Output Tool Call:
hotel_room_pricing.get(
hotelName= "Hilton New York", roomType= "suite with a king-size bed", nights=3)
Verdict:
Invalid parameter values, relevant but incorrect.

Figure 8: Case study: semantic drift introduced by query-level translation. An example where pre-translating a user query alters the realized parameter surface forms, leading to an execution failure despite correct intent understanding.

B.2 Effects of Model Scale and Architecture

Within the GPT-5 family (Figures 9(a), 9(b), 9(c)), performance is consistent across scales, with the standard model slightly outperforming distilled variants. This indicates that tool-calling accuracy depends on the model's style at comprehending the user queries and its understanding of the best tool-calling convention to determine things like how many functions to call, what parameter values to choose, etc, under limited constraint or instruction, to align with the input convention of the downstream task.

For Llama 3.1 (Figures 10(a), 10(b), 10(c)), larger models show fewer syntax errors and stronger instruction following, including implicit awareness of English parameter conventions. Larger models also have a much better instruction following capability when being prompted to pass in parameter values in English, as shown in the PT experiment. In short, for the Llama 3.1 family models, a large model size does improve the overall robustness under user queries of different languages. The improvement is composed of the stability of generating a valid tool call under confusion and uncertainty, a better awareness of the parameter value passing convention, and a better instruction following capability.

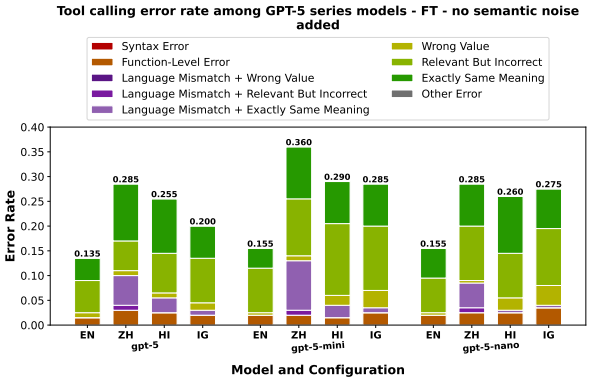
We also notice that in the partially translated experiment, the Llama 3.1 70B model has slightly worse performance than the Llama 3.1 8B model, although it has fewer syntax errors. After investigating the evaluation results, we find that for all numeric values, Llama 3.1 70B appears to only be able to output integers, while the ground truths are decimal numbers. We exclude the possibility of bugs in the parsing logic, and the framework's soundness is further backed up by the fact that Llama 3.1 8B uses the same framework while not having the same problem. This phenomenon remains unexplained.

Qwen3 MoE models (Figures 11(a), 11(b), 11(c)) do not show monotonic gains with scale. Qwen3-30B-A3B underperforms Qwen3-14B, and Qwen3-Next-80B-A3B shows limited gains over Qwen3-32B. This is mainly because they are Mixture of Experts (MoE) models, where only a part of the neurons of the models are activated during inference, selected depending on the category of the task. This may cause the models to not have the expertise in both multilingual understanding and tool calling formation at the same time, thus having worse performance than smaller non-MoE models. Notably, Qwen3-Next-80B-A3B performs best on Igbo, showing that it has more knowledge about the Igbo language than the smaller models.

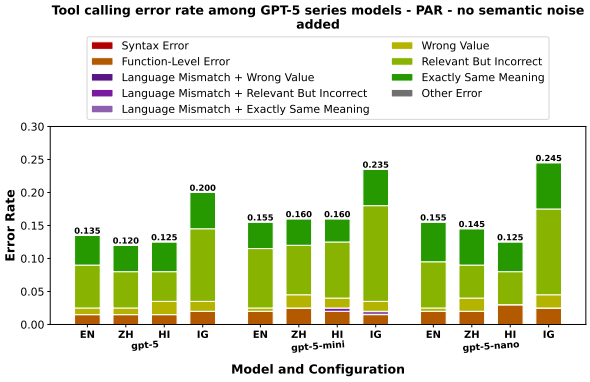
Granite 4 models (Figures 12(a), 12(b), and 12(c)) show an inverse trend: larger models perform worse on Chinese but better on Igbo. This suggests that Granite 4's tool-calling behavior is less responsive to instruction-based mitigation, likely reflecting tighter coupling between tool execution patterns and training-time conventions.

B.3 Additional Results of Semantic Perturbations

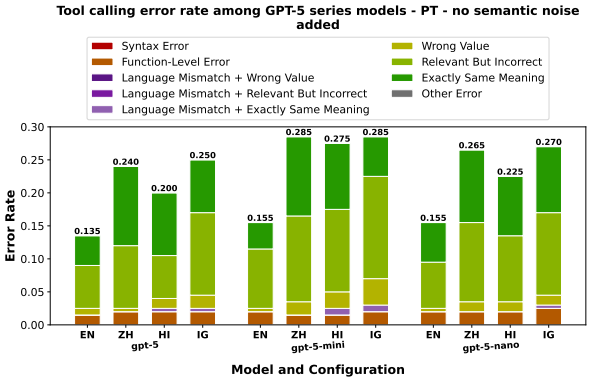
We provide additional results on the interaction between semantic perturbations and execution settings in Appendix Figures 13 and 14. These figures complement the main-text analysis by illustrating how paraphrasing and synonym substitution affect error composition under different language–execution regimes. Consistent with the trends discussed in Section 3.4, semantic perturbations have a limited impact in fully translated settings, where execution failures are dominated by parameter value language mismatch. In contrast, under partially translated queries or explicit English-parameter prompting, semantic perturbations introduce additional errors by disrupting the recovery of exact English parameter surface forms. These results further support our conclusion that semantic noise primarily amplifies multilingual tool-calling failures when strict execution-level surface-form constraints are enforced.



(a) **Fully translated queries (FT).**

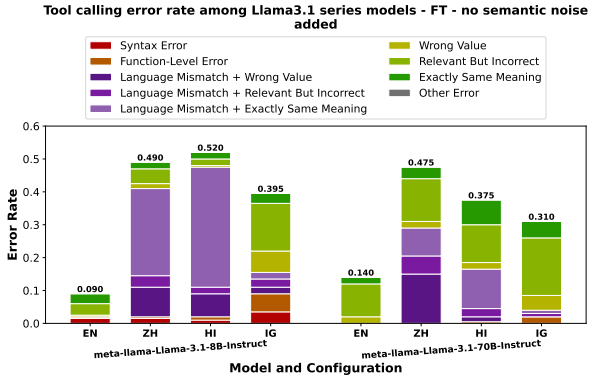


(b) **Partially translated queries (PAR).**

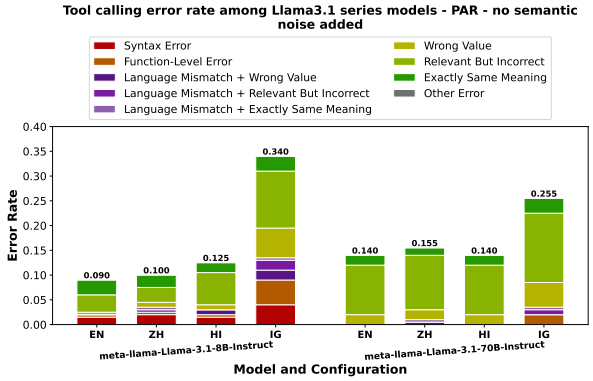


(c) **Fully translated with English-parameter prompting (PT).**

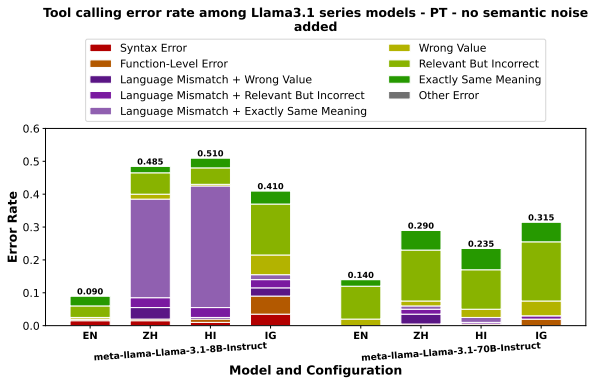
Figure 9: **GPT-5 series behavior across query language compositions and inference-time prompting.** We compare GPT-5, GPT-5 mini, and GPT-5 nano under fully translated (FT), partially translated (PAR), and fully translated queries with explicit English-parameter prompting (PT). Across model sizes, GPT-5 variants exhibit low semantic error rates, indicating strong multilingual intent understanding. Errors in the FT setting are primarily driven by parameter value language mismatch, which is substantially reduced in PAR and PT, demonstrating that execution-level language conventions, rather than semantic reasoning, dominate GPT-5 failures under multilingual tool calling.



(a) **Fully translated queries (FT).**

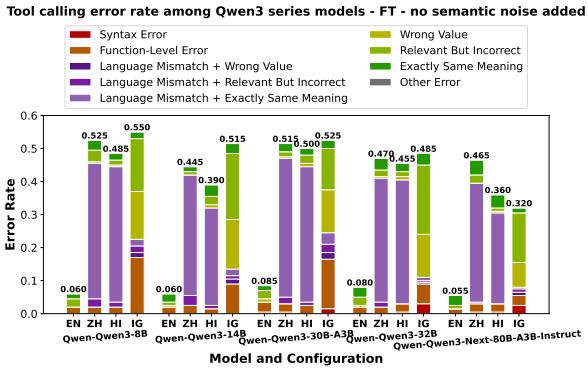


(b) **Partially translated queries (PAR).**

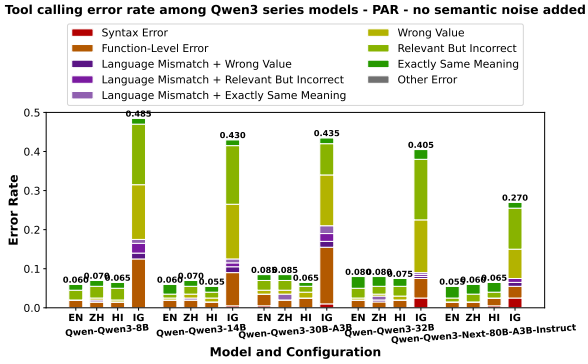


(c) **Fully translated with English-parameter prompting (PT).**

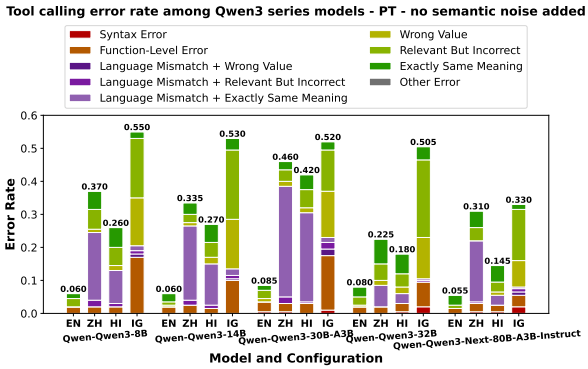
Figure 10: **Llama 3.1 series exhibits limited robustness to multilingual execution constraints.** We compare Llama 3.1–8B and Llama 3.1–70B under fully translated (FT), partially translated (PAR), and fully translated queries with explicit English-parameter prompting (PT). In contrast to GPT-5, Llama 3.1 shows substantial execution failures in the FT setting that combine parameter value language mismatch with semantic errors. While PAR and PT reduce some mismatch errors, a non-trivial portion of failures persists, indicating that multilingual degradation for Llama 3.1 reflects both execution-interface violations and reduced robustness in cross-lingual semantic interpretation.



(a) **Fully translated queries (FT).**

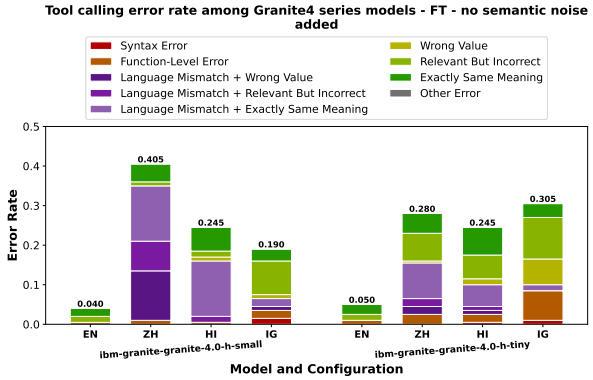


(b) **Partially translated queries (PAR).**

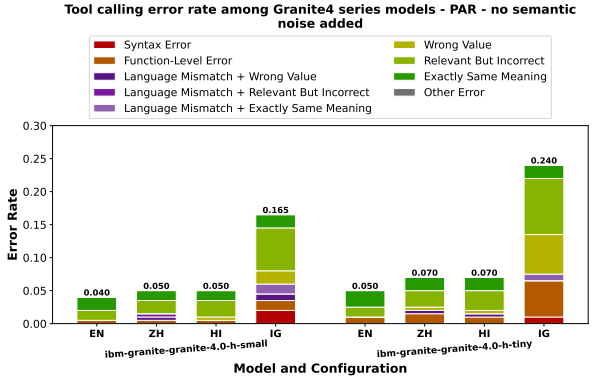


(c) **Fully translated with English-parameter prompting (PT).**

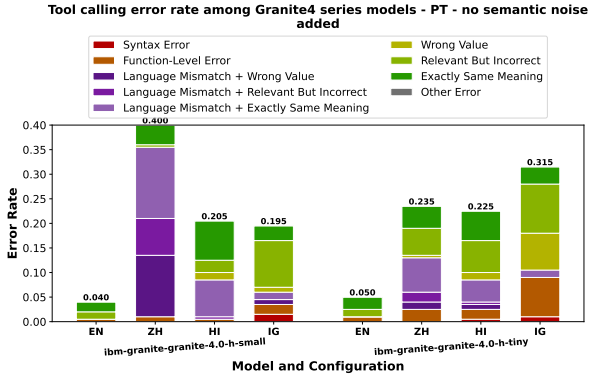
Figure 11: **Qwen 3 models exhibit strong language alignment but inconsistent adherence to execution conventions.** We compare Qwen 3 models of different sizes under fully translated (FT), partially translated (PAR), and fully translated queries with explicit English-parameter prompting (PT). In the FT setting, Qwen 3 frequently preserves non-English parameter values, leading to prominent execution-level language mismatch errors. While PAR substantially reduces these errors by preserving English parameter strings, explicit prompting (PT) yields mixed improvements, indicating that strong language alignment can conflict with strict execution-interface requirements.



(a) **Fully translated queries (FT).**



(b) **Partially translated queries (PAR).**



(c) **Fully translated with English-parameter prompting (PT).**

Figure 12: **Granite 4 models show strong English tool-calling performance but limited gains from inference-time mitigation.** We evaluate Granite 4 models under fully translated (FT), partially translated (PAR), and fully translated queries with explicit English-parameter prompting (PT). While PAR reduces execution-level language mismatch by preserving English parameter strings, explicit prompting (PT) yields limited additional improvement. This suggests that Granite 4's tool-calling behavior is less responsive to instruction-based mitigation, likely reflecting tighter coupling between tool execution patterns and training-time conventions.

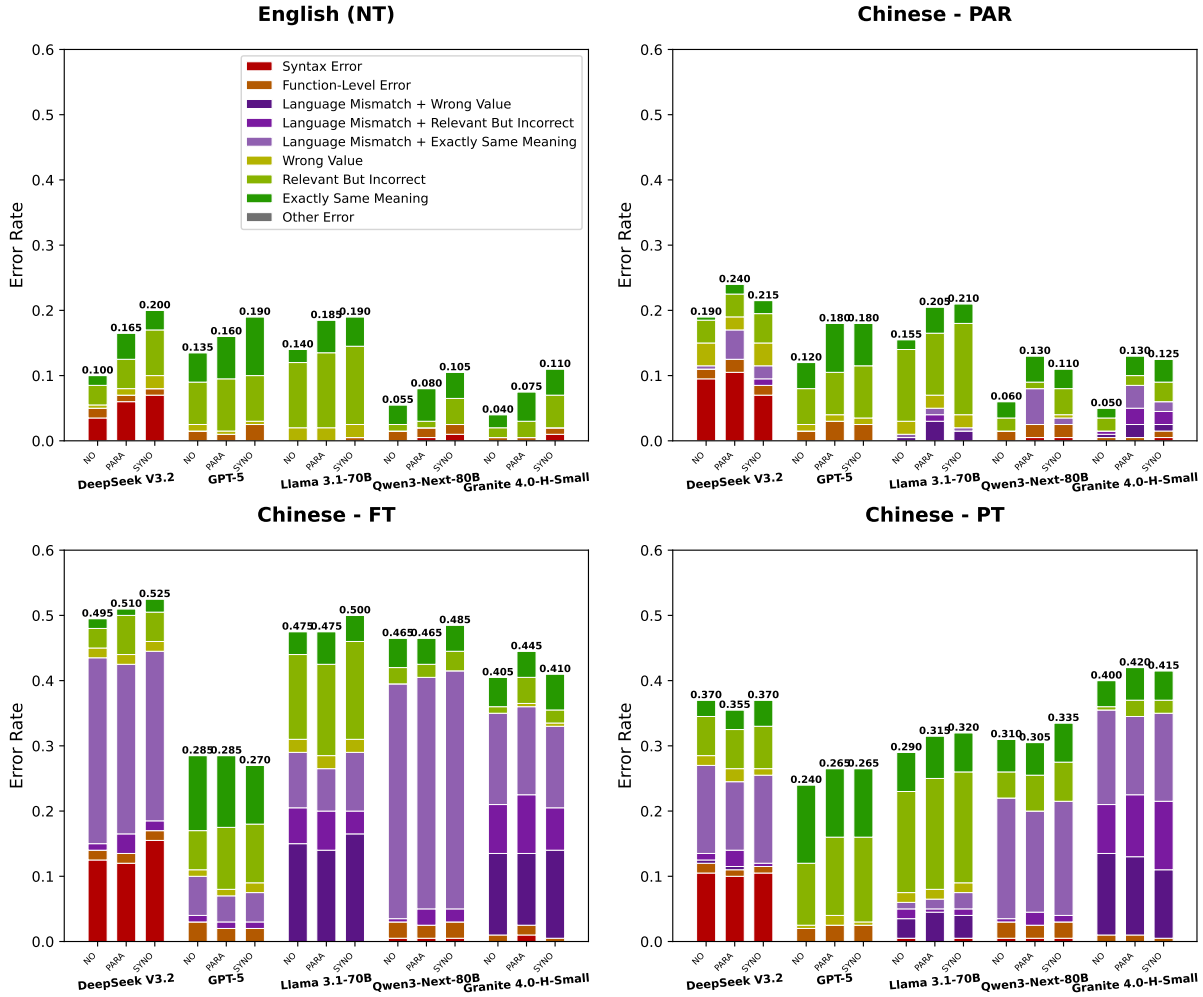


Figure 13: **Semantic perturbations primarily affect Chinese tool calling when English parameter surface forms are required.** We compare five representative models under semantic perturbations across three execution settings on Chinese queries: fully translated (FT), partially translated (PAR), and fully translated with explicit English-parameter prompting (PT).

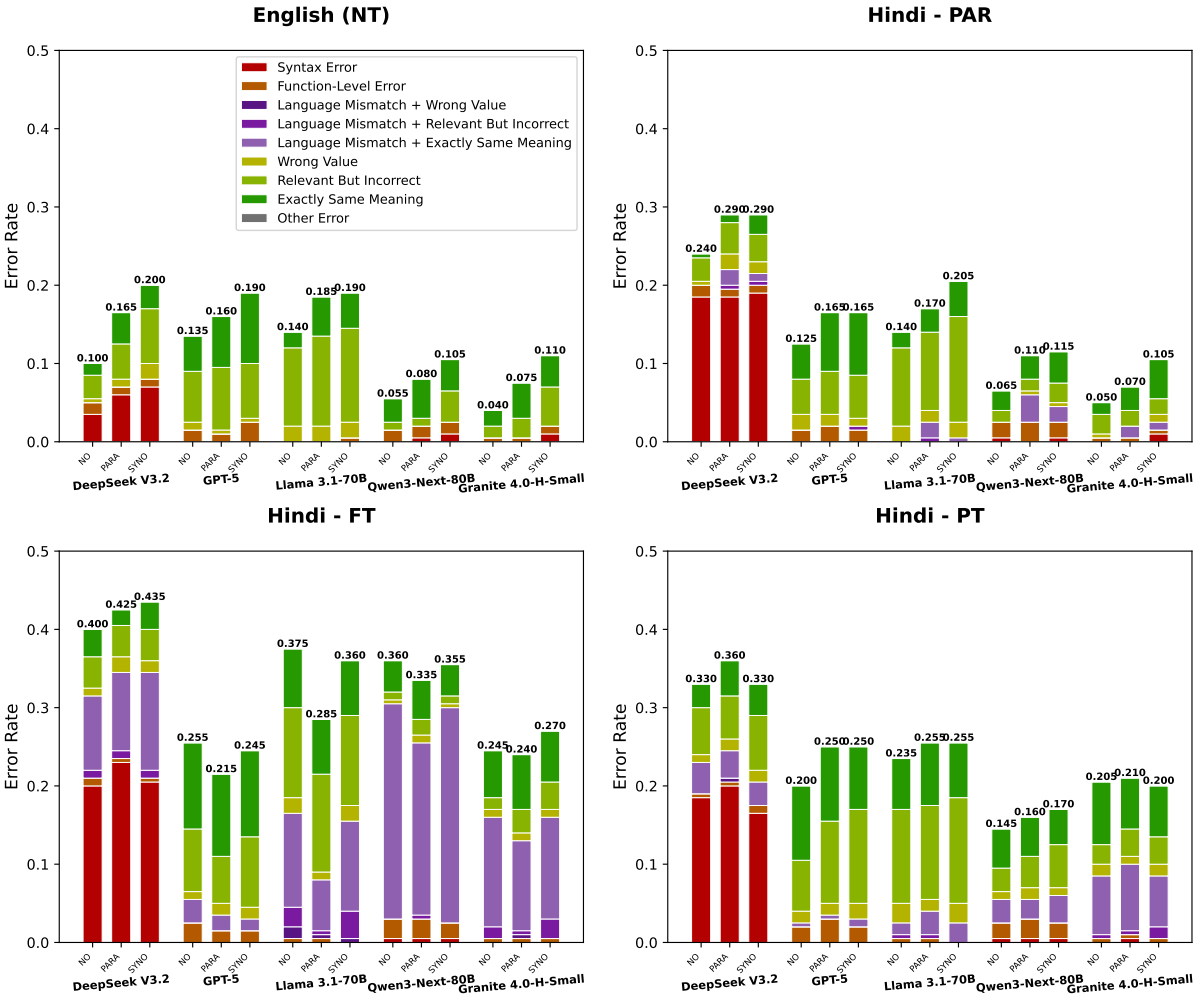


Figure 14: **Semantic perturbations primarily affect Hindi tool calling when English parameter surface forms are required.** We compare five representative models under semantic perturbations across three execution settings: fully translated (FT), partially translated (PAR), and fully translated with English-parameter prompting (PT).