Do LLMs Know Tool Irrelevance? Demystifying Structural Alignment Bias in Tool Invocations

Yilong Liu[1,2], Xixun Lin[1,2], Pengfei Cao[3], Ge Zhang[4], Fang Fang[1,2], Yanan Cao[1,2]*

[1]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[3]Institute of Automation, Chinese Academy of Sciences, Beijing, China
[4]School of Computer Science and Technology, Donghua University, Shanghai, China
{liuyilong, caoyanan}@iie.ac.cn

Abstract

Large language models (LLMs) have demonstrated impressive capabilities in utilizing external tools. In practice, however, LLMs are often exposed to tools that are irrelevant to the user's query, in which case the desired behavior is to refrain from invocations. In this work, we identify a widespread yet overlooked mechanistic flaw in tool refusal, which we term structural alignment bias: Even when a tool fails to serve the user's goal, LLMs still tend to invoke it whenever query attributes can be validly assigned to tool parameters. To systematically study this bias, we introduce SABEval, a new dataset that decouples structural alignment from semantic relevance. Our analysis shows that structural alignment bias induces severe tool-invocation errors in LLMs, yet remains largely unaccounted for in existing evaluations. To investigate the internal mechanisms underlying this bias, we propose Contrastive Attention Attribution, which reveals two competing pathways for semantic checking and structural matching. The relative strength of these pathways drives LLMs' tool invocation decisions. Based on these findings, we further introduce a rebalancing strategy that effectively mitigates structural alignment bias, as demonstrated by extensive experiments, without degrading general tool-use capabilities. Our code and dataset are available at https://github.com/along-l/irrelevant-tool.

1 Introduction

Driven by the significant breakthroughs in model architectures, computational capabilities, and large-scale data sources, large language models (LLMs) have advanced rapidly in recent years (Naveed et al., 2025; Xi et al., 2025; Mohammadi et al., 2025). A key capability of LLMs is the utilization of external tools, which bridges the gap between text generation and real-world environmental interaction (Gao et al., 2024; Houliston et al., 2025; Qu et al., 2025).

However, in many practical scenarios, LLMs are often exposed to irrelevant tools that are useless for addressing user queries (Zhang et al., 2024; Ross et al., 2025; Liu et al., 2025a; Lin et al., 2025). For example, a user may request to refund a transaction, but the LLM only has access to a tool for transferring funds. Invoking such irrelevant tools may introduce system latency and misleading outputs, and more critically, for high-risk applications, such as financial transactions or database management, can lead to irreversible and catastrophic consequences, including financial loss or data corruption (Xia et al., 2025; Subramani et al., 2025). As a result, the desired behavior of LLMs is to recognize the limitation of the available tools and refrain from invocations.

Although recent studies have explored the ability of LLMs to reject irrelevant tools (Xu et al., 2024; Patil et al., 2025), we find that there exists a widespread yet overlooked mechanistic flaw in LLMs when refusing tool invocations, which we term **Structural Alignment Bias**. As illustrated in Fig. 1, while a tool cannot help realize the user's goal (e.g., user queries a game on *PlayStation*, but the provided tool is designed to check *Nintendo Switch* games), the model still shows a strong tendency to invoke this tool as there exists a valid one-to-one assignment between query attributes and tool parameters (e.g., the attribute "Spider-Man" can be validly assigned to "`game_title`", and "Canadian" to "`region`").

In this paper, we define the inconsistency between the tool's function and the user's goal as **Semantic Irrelevance**, and refer to the existence of such one-to-one assignment as **Structural Alignment**. Building upon these definitions, we characterize structural alignment bias as a systematic tendency of LLMs to prioritize this structural alignment as a shortcut for tool invocations, thereby

*Corresponding author.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 31934–31958
July 2-7, 2026

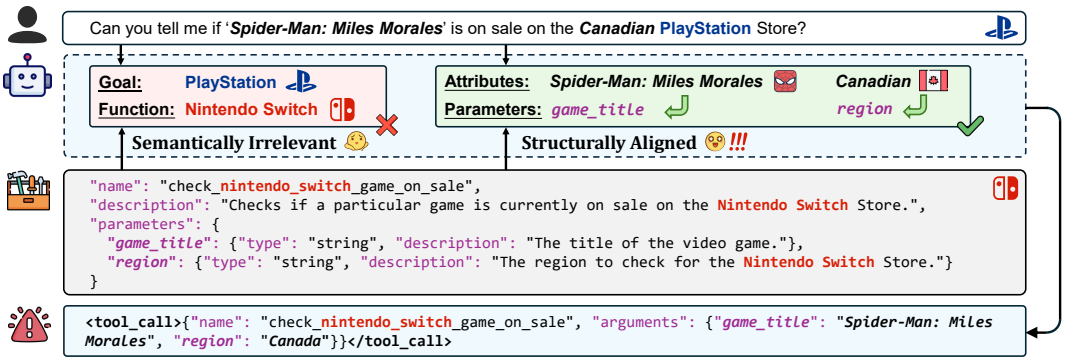

Figure 1: Illustration of erroneous tool invocation driven by the model's strong reliance on structural alignment.

bypassing the verification of semantic relevance.

To further verify and analyze the proposed structural alignment bias, we construct **SABEval**, a dataset designed to decouple structural alignment from semantic relevance. SABEval mirrors real-world scenarios where distinct services share unified interfaces, avoiding synthetic corner cases. We conduct comprehensive experiments on five widely used tool-augmented LLMs. Our behavioral analysis uncovers a striking contrast: while the error rate is negligible ($< 0.2\%$) on structurally misaligned samples created by previous studies (See § 2 for details.), it rises sharply to 41.9% under structural alignment and further escalates to 90.4% as the alignment degree increases. Furthermore, we perform the counterfactual analysis which confirms a strong causal relationship between structural alignment and erroneous invocations, highlighting the critical impact of structural alignment bias on model behaviors.

To investigate the internal mechanisms underlying structural alignment bias, we introduce **Contrastive Attention Attribution** (CAA), a novel interpretability method that traces the information flow without requiring the strict token-wise alignment between counterfactual pairs. Our method reveals two distinct pathways responsible for semantic checking and structural matching, respectively. Moreover, we find that LLMs' tool invocation decisions are driven by the relative strength of these two pathways. Based on these insights, we propose a new strategy that rebalances the two pathways to mitigate structural alignment bias. Extensive experiments demonstrate that our method can significantly alleviate structural alignment bias across diverse models and tools, while maintaining their general tool-use capabilities. Our main contributions are summarized as follows:

- **Problem Identification.** We are the first to identify and formalize structural alignment bias, a mechanistic flaw in which models over-rely on structural alignment when making tool invocation decisions.
- **Empirical Evidence.** We construct SABEval, a dataset that decouples structural alignment from semantic relevance and provides comprehensive evidence of the prevalence, severity, and causality of structural alignment bias.
- **Interpretability Method.** We propose CAA, a novel interpretability method that reveals models' internal computations without relying on the strict token correspondence of counterfactuals, enabling the discovery of competitive information flow for tool invocations.
- **Bias Mitigation.** We introduce a precise intervention to mitigate structural alignment bias. Extensive experiments show its effectiveness, achieving an 80% relative error reduction while preserving general tool-use capabilities.

2 Problem Formulation

In this section, we first introduce the irrelevance setting in tool invocations. We then propose a formal concept of structural alignment with our core question.

Tool Invocations with Irrelevant Tools. A critical challenge in tool-augmented LLMs is whether LLMs can correctly refuse to invoke tools that cannot serve the user's goal, a scenario often referred to as the **Irrelevance Setting** (Patil et al., 2025). Our work specifically addresses this problem. Consider an input (t, q), where t is a candidate tool and q is the user query. We denote the tool's function by f_t and the user's goal by g_q. In the irrelevance setting, t is *semantically irrelevant* to q, which we formalize as $f_t \not\equiv g_q$. This implies that f_t does not help realize g_q, and the model should reject

t (Chen et al., 2025). For example, as illustrated in Fig. 1, the tool is designed to *check Nintendo Switch games* (i.e., f_t), which fails to address the user's goal of *checking PlayStation games* (i.e., g_q), and LLMs should refuse this invocation.

Structural Alignment. Besides semantic relevance, we hypothesize that an LLM's decision to invoke tools is strongly influenced by another factor: whether the attributes expressed in q can be validly mapped to the parameters of t. We refer to this factor as structural alignment as described in the Introduction.

Formally, let $\mathcal{P}_t = \{p_1, \ldots, p_n\}$ be the parameters of t (e.g., "`game_title`", "`region`"), and $\mathcal{A}_q = \{a_1, \ldots, a_m\}$ be the attributes extracted from q (e.g., "Spider-Man", "Canadian"). We introduce a binary indicator:

$$b_{t,q} = \mathbb{I}\big(\mathcal{A}_q \cong \mathcal{P}_t\big), \tag{1}$$

where $\mathbb{I}(\cdot)$ is the indicator function and $\mathcal{A}_q \cong \mathcal{P}_t$ denotes that there exists a *bijection*[1] between $\mathcal{A}_q$ and $\mathcal{P}_t$ such that every $a_j \in \mathcal{A}_q$ can be validly assigned to a parameter $p_i \in \mathcal{P}_t$, and vice versa (e.g., "Spider-Man" for "`game_title`" and "Canadian" for "`region`" in Fig. 1, yielding $b_{t,q} = 1$). For a given input (t, q), we suppose that structural alignment holds when $b_{t,q} = 1$, and that structural misalignment exists when $b_{t,q} = 0$.

The phenomenon of structural alignment has been largely ignored by previous evaluations of models' ability to reject irrelevant tools (Patil et al., 2025; Chen et al., 2025; Huang et al., 2023; Liu et al., 2025c). They typically construct evaluation samples by randomly pairing a query with tools sampled from a pool. While such constructions ensure semantic irrelevance, they predominantly induce structural misalignment, thereby confounding evaluation results. This observation raises the core question of our work: *Do LLMs genuinely recognize that semantic relevance is required for a valid tool invocation, or do they merely rely on structural alignment to make decisions?*

3 Experimental Setup

To answer the above question, we first establish the experimental setup for our empirical study. Specifically, we construct a new dataset, SABEval, and describe its extensions, as well as the evaluated models used in our experiments.

[1]That is, $\mathcal{A}_q$ and $\mathcal{P}_t$ are in a one-to-one correspondence, implying $|\mathcal{A}_q| = |\mathcal{P}_t|$.

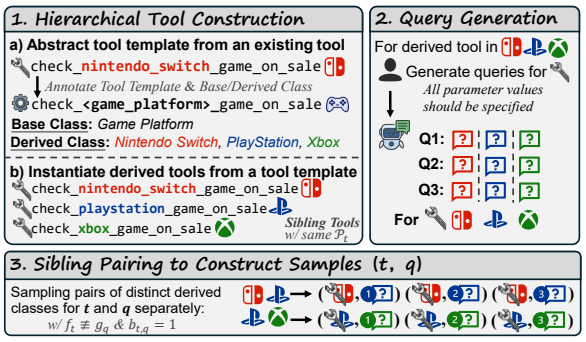

Figure 2: SABEval construction with three steps. Samples are semantically irrelevant but structurally aligned.

Dataset. Existing datasets entangle semantic irrelevance with structural misalignment. To strictly isolate the former (i.e., $f_t \not\equiv g_q$ & $b_{t,q} = 1$), we construct a new dataset **SABEval**, based on the *polymorphism* principle of Object-Oriented Programming (Meyer, 1997). As illustrated in Fig. 2, the construction pipeline comprises three steps: hierarchical tool construction, query generation, and sibling pairing. Crucially, no valid tool is available for any query in SABEval, meaning any tool invocation constitutes an error. Construction details are provided in Appendix A. This design mirrors real-world scenarios where distinct services share unified interfaces, rather than creating synthetic corner cases. Finally, we construct *101* tool templates, *5* queries per derived tool, and *10* sibling combinations per tool template. This process yields our base dataset $\mathcal{D}_0$, comprising *5,050* instances.

Dataset Extensions. To support the analysis in §4 and §5, we construct four extensions $\mathcal{D}_1, \ldots, \mathcal{D}_4$ by adding tool templates of $\mathcal{D}_0$ with $k \in \{1, 2, 3, 4\}$ additional parameters[2]. We follow the pipeline in Fig. 2 to generate derived tools and queries from new tool templates, while reusing the original sibling pairings of $\mathcal{D}_0$ to construct samples. This guarantees that $\mathcal{D}_{k+1}$ differs from $\mathcal{D}_k$ by one additional attribute-parameter assignment.

Evaluated Models. We investigate five models that natively support tool invocations: the Qwen3 series (4B, 8B, and 14B) (Yang et al., 2025)[3], ToolACE-2.5-8B (Liu et al., 2025c), and Watt-Tool-8B (Watt-AI, 2024). ToolACE-2.5-8B and

[2]We employ GPT-4o (Hurst et al., 2024) to synthesize additional parameters.

[3]For the Qwen3 series, we disable the "thinking mode" to maintain consistency with the other models. Nevertheless, we report results with thinking mode enabled in Appendix M.1. We also evaluate Qwen3-32B to study the impact of model size in Appendix M.2.

Watt-Tool-8B are fine-tuned from Llama-3.1-8B-Instruct (Dubey et al., 2024)[4]. We adopt the default system prompts for all evaluated models, and more details are given in Appendix B.

4 Behavior Analysis

To address the research question proposed in § 2, we tackle the following two progressive inquiries:

1. How do LLMs perform with structurally aligned but semantically irrelevant tools? (§ 4.1)
2. To what extent does structural alignment contribute to erroneous tool invocations? (§ 4.2)

4.1 Performance under Structural Alignment

This section evaluates the model's capability to reject semantically irrelevant but structurally aligned tools. We first introduce the used metric: **Tool Invocation Rate** (TIR). Consider a dataset $\mathcal{D}$ consisting of N samples. Let $P(w|q,t)$ denote the model's predicted probability for the next token w. TIR is defined as the proportion of samples where the special tool-call token w_{tool} is the most likely prediction:

$$\text{TIR} = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}\left(\underset{w\in\mathcal{V}}{\text{argmax}}\, P(w|q,t) = w_{\text{tool}}\right), \tag{2}$$

where $\mathcal{V}$ is the vocabulary. w_{tool} for all models are shown in Appendix F. This formulation enables efficient evaluation via a single forward pass without full response generation. While it theoretically counts all tool-call format outputs, including hallucinated tool invocations, our premise is that models faithfully invoke only the provided tools. We verify this in Appendix F. For models prone to tool hallucination, a fine-grained metric based on full response parsing would be necessary. In our setting, a lower TIR indicates a stronger capability to reject irrelevant tools. We then report two key empirical observations.

High Rates of Erroneous Invocations. We conduct experiments on $\mathcal{D}_0$ of SABEval and compare the results with the random pairing setting (uniformly sampling a tool for every query). As shown in Table 1, across all models, TIR of random pairing remains extremely low (max 0.16%), while they surge dramatically (max 41.86%) on SABEval. This substantial gap indicates that LLMs hardly reject irrelevant tools when they are structurally aligned, suggesting that their decisions could be strongly biased toward structural alignment. More importantly, this gap indicates that the random pairing setting substantially overestimates LLMs' ability to reject irrelevant tools.

[4]We exclude the original Llama models due to their inability to consistently adhere to the tool invocation format.

Model	Random	SABEval	Δ
Qwen3-4B	0.16	**40.04**	+39.88
Qwen3-8B	0.04	**34.26**	+34.22
Qwen3-14B	0.04	**41.86**	+41.82
ToolACE-2.5-8B	0.12	**40.67**	+40.55
Watt-Tool-8B	0.00	**10.51**	+10.51

Table 1: Tool invocation rate (%) on randomly paired samples and SABEval ($\mathcal{D}_0$).

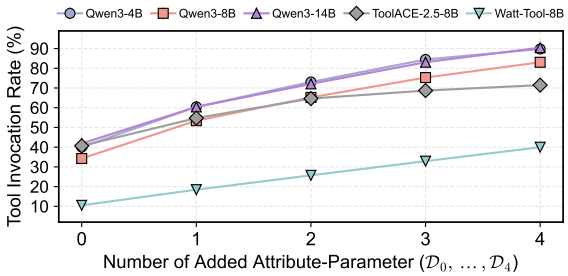

Figure 3: Tool invocation rate (%) on SABEval subsets $\mathcal{D}_0, \ldots, \mathcal{D}_4$ with increasing structural alignment degree.

Error Scales with Alignment Degree. Intuitively, more fillable tool parameters may make models more likely to invoke the tool. We therefore examine whether TIR increases systematically with the number of attribute–parameter assignments between q and t, which we term the **structural alignment degree**. To this end, we evaluate on $\mathcal{D}_0, \ldots, \mathcal{D}_4$ of SABEval[5]. As illustrated in Fig. 3, we find the error rates of all models consistently and significantly increase as the alignment degree grows (up to 90.44%). Results grouped by tool template show the same trend (See Appendix C for details.). This further underscores the severity of the performance collapse on structurally aligned samples and reveals the fundamental flaw in the models.

4.2 Causal Role of Structural Alignment

In this section, we answer the question of to what extent structural alignment contributes to the erroneous invocations observed in § 4.1. To do so, we

[5]We do not simply group samples by the number of parameters in $\mathcal{D}_0$, as parameter counts are unevenly distributed across tool templates, and the template itself acts as a confounding factor between TIR and alignment degree.

User Query: Can you tell me if 'Spider-Man: Miles Morales' is on sale on the Canadian **PlayStation** Store?
Original Tool: `check_nintendo_switch_game_on_sale` **Params:** `game_title`, `region`

Counterfactual	Strategy	Tool Name	Parameters
Structural	*Param Substitution*	`check_nintendo_switch_game_on_sale`	`game_title`, `include_dlc`
	Param Removal	`check_nintendo_switch_game_on_sale`	`game_title`
	Param Addition	`check_nintendo_switch_game_on_sale`	`game_title`, `region`, `include_dlc`
Semantic	*Target Tool*	`check_playstation_game_on_sale`	`game_title`, `region`

Table 2: Example of counterfactuals. We construct **structural counterfactuals** by intervening on the original tool parameters via three strategies (*substitution*, *removal*, and *addition*) while retaining the original query. The **semantic counterfactual** includes a valid tool that serves the user's goal, which is utilized for pathway discovery in § 5.1.

analyze structural counterfactuals of SABEval by intervening on the structural alignment status.

Intervention Design. Given a sample (t, q) with $\mathcal{P}_t = \{p_1, \ldots, p_n\}$ and $b_{t,q} = 1$, we intervene on t, i.e., $t \to t^*$, and pair it with q to construct the counterfactual (t^*, q), for which $b_{t^*,q} = 0$. Specifically, we implement three intervention strategies, as illustrated in Table 2:

- **Parameter Substitution.** We substitute l original parameters, such that $\mathcal{P}_{t^*} = \{p_1, \ldots, p_{n-l}, p_1^*, \ldots, p_l^*\}$, where the new parameters p_i^* are mutually distinct and do not appear in $\mathcal{P}_t$. In this case, there exist both unassignable user attributes and unspecified parameters.
- **Parameter Removal.** We remove l original parameters, such that $\mathcal{P}_{t^*} = \{p_1, \ldots, p_{n-l}\}$. In this case, $|\mathcal{A}_q| > |\mathcal{P}_{t^*}|$, representing a scenario where user attributes cannot be fully assigned.
- **Parameter Addition.** We append l new parameters, such that $\mathcal{P}_{t^*} = \{p_1, \ldots, p_n, p_1^*, \ldots, p_l^*\}$. In this case, $|\mathcal{A}_q| < |\mathcal{P}_{t^*}|$, representing a scenario where some parameters are unspecified.

In practice, we set the perturbation size $l = 1$. We conduct experiments on the $\mathcal{D}_1$ subset of SABEval and utilize the additional parameters generated in § 3, thereby ensuring the plausibility of the counterfactual tools.

Empirical Results. As shown in Fig. 4, all intervention strategies lead to a pronounced reduction in TIR. Notably, parameter substitution exhibits the most significant and consistent impact across all models (relative declines of 58% for Qwen3-8B and 83% for Watt-Tool), aligning with its effect of introducing unassignable attributes and unspecified parameters. We further investigate the probability shifts of w_{tool} in Appendix D, where over 70% and 90% of samples exhibit clear probability decreases for the Qwen3 family and other models, respectively. Overall, these results underscore a strong causal effect of structural alignment on erroneous tool invocations.

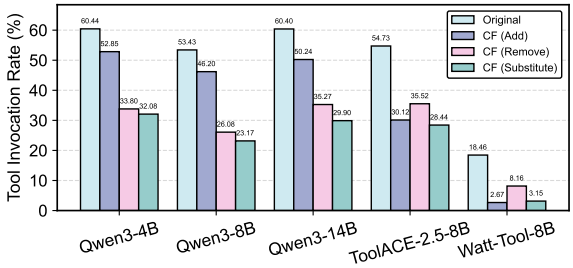

Figure 4: Tool invocation rate (%) on original samples and their structural counterfactuals from $\mathcal{D}_1$.

Structural Alignment Bias. We previously show that LLMs are prone to invoking irrelevant tools under structural alignment (§ 4.1). Here, we further establish a strong causal link between such behavior and structural alignment. Based on these analyses, we can now answer the core question posed in § 2: *Instead of genuinely recognizing the necessity of semantic relevance, LLMs exhibit a systematic bias toward structural alignment in tool invocations.* We term this important problem structural alignment bias[6]. Such bias prevents LLMs from reliably rejecting irrelevant tools, leading to erroneous tool invocations and potentially irreversible consequences.

5 Mechanistic Analysis

In this section, we investigate the internal mechanism driving structural alignment bias. We consider semantic irrelevance and structural alignment as competing signals and propose a novel method

[6]We provide further analysis in Appendix E to distinguish this bias from naive textual similarity, confirming it originates from the structural properties of the data.

CAA (§ 5.1), which reveals the specific information flow responsible for processing them: *semantic pathways* (checking the semantic irrelevance, then suppressing invocations) and *structural pathways* (matching attributes to parameters, then promoting invocations). Our analysis suggests that their relative strength drives models' decisions, and targeted interventions offer an effective avenue for mitigating this bias (§ 5.2).

5.1 Contrastive Attention Attribution

Attention heads serve as channels for information flow between positions (Elhage et al., 2021). For a head h, the attention weight $\alpha^h_{i,j}$ acts as a modulation coefficient controlling the intensity of information flow from the source position j to the target position i. Given this, we focus on measuring the importance of attention weights to identify semantic and structural pathways. Unlike traditional causal mediation analysis (Wang et al., 2022; Hanna et al., 2023), CAA introduces counterfactual contrast at the group level, eliminating the reliance on strict token correspondence of original and counterfactual inputs. CAA includes three steps: estimating the sample-wise importance; deriving the group-wise importance via span aggregation; introducing the contrastive importance score to identify pathways.

Sample-wise Importance. Given the input $x = (t, q) \in \mathcal{D}$ and a task metric m, we measure the importance of $\alpha^h_{i,j}$ via the indirect effect (Pearl, 2001; Vig et al., 2020):

$$I(m, \alpha^h_{i,j}, x) = m(x) - m(x \mid \mathrm{do}(\alpha^h_{i,j} = 0)), \tag{3}$$

where $\mathrm{do}(\alpha^h_{i,j} = 0)$ zeros out $\alpha^h_{i,j}$. Eq. (3) describes the induced change for m under zero ablation. Based on it, we define two new metrics for semantic checking m_{sem} and structural matching m_{str}, respectively:

$$m_{\text{sem}}(x) = \max_{w \in \mathcal{R}} \log P(w|x) - \log P(w_{\text{tool}}|x), \tag{4}$$

$$m_{\text{str}}(x) = \log P(w_{\text{tool}}|x) - \max_{w \in \mathcal{R}} \log P(w|x). \tag{5}$$

Here, $\mathcal{R} = \{w_{r_1}, \ldots, w_{r_k}\} \subset \mathcal{V}$ is a set of refusal tokens that the model typically starts with when refusing invocations (See Appendix F for a discussion on refusal token selection.). Due to the large number of attention weights, we further use attribution patching (Nanda, 2023) to efficiently approximate Eq. (3) via a first-order Taylor expansion:

$$I(m, \alpha^h_{i,j}, x) \approx \alpha^h_{i,j} \cdot \frac{\partial m(x)}{\partial \alpha^h_{i,j}}. \tag{6}$$

This enables parallel estimation for all α using only one forward and one backward pass.

Group-wise Importance. Since sequence lengths vary across samples, the importance of attention weights is not directly comparable. To address this, we adopt a span-level aggregation approach (Haklay et al., 2025) by partitioning the input text into ordered spans $\mathcal{S} = \langle s_i \rangle_{i=1}^N$, where the tool definition and the query are treated as separate spans to ensure consistency across samples. Details of the span partitioning are provided in Appendix G. Consequently, the group-wise importance from a source span s_j to a target span s_i, measured over $\mathcal{D}$, is defined as:

$$\hat{I}^h_{s_i,s_j}(m, \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \sum_{l \in s_j} \sum_{k \in s_i} I(m, \alpha^h_{k,l}, x). \tag{7}$$

Contrastive Importance Score. Based on Eq. (3), we can identify attention weights that contribute significantly to the metric m. However, this manner captures the aggregated contributions of all features originating from x, making it difficult to disentangle the effects of specific factors. In our setting, we are specifically interested in the effects induced by semantic irrelevance and structural alignment. To isolate these factors, we construct counterfactual samples and introduce **Contrastive Importance Score** (CIS). Let $\mathcal{D}^* = \{x^* \mid x \in \mathcal{D}\}$ be the set of counterfactual samples corresponding to $\mathcal{D}$. We define CIS as:

$$\mathrm{CIS}^h_{s_i,s_j}(m, \mathcal{D}, \mathcal{D}^*) = Z^h_{s_i,s_j}(m, \mathcal{D}) - Z^h_{s_i,s_j}(m, \mathcal{D}^*), \tag{8}$$

$$Z^h_{s_i,s_j}(m, \mathcal{D}') = \frac{\max\left(\hat{I}^h_{s_i,s_j}(m, \mathcal{D}'), 0\right)}{\max_{h',s_p,s_q} \hat{I}^{h'}_{s_p,s_q}(m, \mathcal{D}')}. \tag{9}$$

Here, Eq. (9) is a normalized form of group-wise importance, where $\mathcal{D}'$ denotes either $\mathcal{D}$ or $\mathcal{D}^*$. In Eq. (8), by differencing two normalized terms, we remove the effects of features shared in x.

CAA constructs two types of counterfactuals. As shown in Table 2, we construct structural counterfactuals $\mathcal{D}^{*,\text{str}}$ by parameter substitution, and

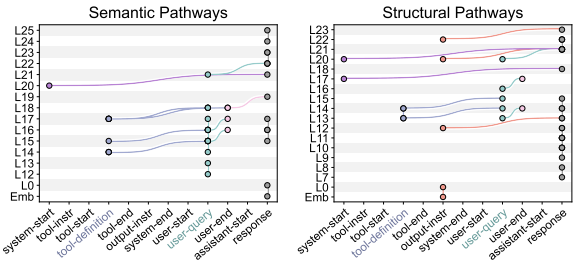

Figure 5: Semantic (left) and structural (right) pathways in Qwen3-8B from top-k attention weights ($k = 23$, corresponding to 2% of the attention heads).

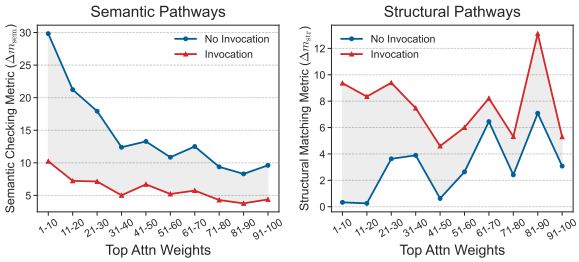

Figure 6: Comparison of pathway strengths between invocation and non-invocation cases for Qwen3-8B.

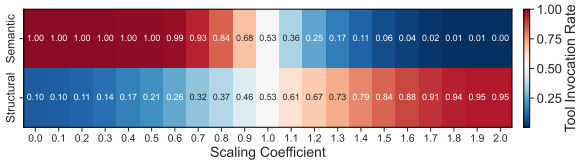

Figure 7: TIR under different scaling coefficients for Qwen3-8B ($k = 23$).

semantic counterfactuals $\mathcal{D}^{*,\text{sem}}$ by pairing a query with its target tool:

$$\text{CIS}^{h,\text{sem}}_{s_i,s_j} := \text{CIS}^{h}_{s_i,s_j}(m_{\text{sem}}, \mathcal{D}_{\text{nocall}}, \mathcal{D}^{*,\text{sem}}_{\text{nocall}}), \quad (10)$$

$$\text{CIS}^{h,\text{str}}_{s_i,s_j} := \text{CIS}^{h}_{s_i,s_j}(m_{\text{str}}, \mathcal{D}_{\text{call}}, \mathcal{D}^{*,\text{str}}_{\text{call}}). \quad (11)$$

Here, $\mathcal{D}_{\text{nocall}}$ and $\mathcal{D}_{\text{call}}$ consist of cases where the model originally refuses or performs tool invocations, and decisions on their respective counterfactuals, $\mathcal{D}^{*,\text{sem}}_{\text{nocall}}$ and $\mathcal{D}^{*,\text{str}}_{\text{call}}$, are swapped[7]. We compute Eq. (10) and Eq. (11), and identify the top-k attention weights between spans as semantic and structural pathways, respectively.

5.2 Experiment & Analysis

We split SABEval into training/validation/test sets (40%, 20%, 40%), with disjoint tool templates. We compute CIS by sampling 500 instances per group ($\mathcal{D}_{\text{nocall}}$ and $\mathcal{D}_{\text{call}}$) from the training split of the $\mathcal{D}_1$ subset. All analyses are performed on the held-out test set. Unless otherwise specified, we select the top-k attention weights where k corresponds to 2% of the number of attention heads.

Characteristics of Pathways. Fig. 5 illustrates the semantic and structural pathways identified in Qwen3-8B. The x-axis represents the ordered input spans (e.g., tool definition, user query), and the y-axis represents the model layer. Edges denote top-k attention weights aggregated across heads between source and target spans at different layers. While both pathways involve similar layers, we observe clear differences in their information flow patterns. Specifically, semantic pathways are denser at the query positions, indicating that these positions play a key role during semantic checking. In contrast, during structural matching, the model exhibits strong attention at the final span toward the output instruction, which specifies the tool invocation format. This pattern suggests that excessive focus on output formatting biases Qwen3 toward structural alignment. More discussion and results for other models are provided in Appendix H.

Relative Pathway Strength Drives Model Decisions. We find that tool invocation decisions are governed by the relative strength of the two pathways. To quantify this, we partition the top-100 attention weights into 10 groups and iteratively patch each group via Eq. (3) to assess their impacts on m_{sem} and m_{str}, using 500 invocation and 500 non-invocation cases from $\mathcal{D}_{\text{call}}$ and $\mathcal{D}_{\text{nocall}}$ of $\mathcal{D}_1$, respectively. As shown in Fig. 6, pathway strengths vary significantly: erroneous invocation cases exhibit weaker semantic pathways and stronger structural pathways, opposite to non-invocation cases. We further show in Appendix I that as structural alignment degree increases, structural pathways strengthen while semantic pathways weaken, consistent with the higher TIR noted in § 4.1. These results indicate that the relative dominance of the semantic and structural pathways determines LLMs' tool invocation decisions.

Mitigating the Bias via Attention Scaling. If these pathways indeed govern LLMs' decision process, manipulating their strength should causally influence models' behavior. To validate their causal roles, we intervene on them by scaling the attention

[7]For example, $\mathcal{D}_{\text{nocall}}$ is constructed by selecting samples where the model refuses the original (irrelevant) tool and invokes the tool after replacing it with the target tool.

Model	Method	SABEval ↓					ACEBench ↑	
		$\mathcal{D}_0$	$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_4$	Single	Parallel
Qwen3-4B	Base	37.61	57.85	73.41	85.56	89.76	65	60
	Prompt	27.12	42.20	56.83	69.71	75.95	**66**	57
	CAA (Ours)	**8.29**	**14.29**	**20.29**	**27.76**	**36.00**	63	**61**
Qwen3-8B	Base	30.39	51.61	66.29	75.56	82.44	71	62
	Prompt	19.90	34.29	48.73	57.66	66.20	**72**	**66**
	CAA (Ours)	**4.29**	**6.49**	**7.27**	**8.98**	**11.12**	67	**66**
Qwen3-14B	Base	41.37	59.46	74.73	83.90	91.02	68	62
	Prompt	24.29	33.41	43.56	56.78	65.66	**70**	62
	CAA (Ours)	**5.56**	**7.27**	**9.31**	**15.51**	**21.22**	68	**63**
ToolACE-2.5	Base	37.95	51.12	63.02	67.41	68.00	**90**	**81**
	Prompt	39.12	53.66	62.63	67.71	67.71	87	**81**
	CAA (Ours)	**3.41**	**5.27**	**5.76**	**6.88**	**11.46**	89	79
Watt-Tool	Base	8.68	15.46	23.31	31.51	39.51	**84**	76
	Prompt	8.59	15.37	22.34	29.61	38.05	83	**77**
	CAA (Ours)	**2.54**	**2.98**	**5.46**	**8.10**	**12.68**	83	75

Table 3: Evaluation results on all subsets of SABEval and ACEBench. We report the tool invocation rate (↓) on SABEval and accuracy (↑) on ACEBench. Base denotes the performance of the original model.

weights of head h between span s_i and s_j:

$$\alpha^h_{k,l} \leftarrow \rho \cdot \alpha^h_{k,l}, \quad \forall k \in s_i, l \in s_j, \qquad (12)$$

where ρ is a scaling coefficient. We first examine how TIR varies with different ρ on 500 test cases. As shown in Fig. 7, the two pathways exert causal effects on the tool invocation rate (TIR) in opposite directions, which is consistent with their hypothesized roles and demonstrates the expected causal effect. Results for other models and different top-k thresholds are reported in Appendix J, which show similar trends.

Motivated by these findings, we propose a targeted intervention strategy to mitigate structural alignment bias. Specifically, we amplify semantic pathways ($\rho_{\text{sem}} > 1$) while suppressing structural pathways ($\rho_{\text{str}} < 1$). The coefficients ρ_{sem} and ρ_{str} are selected on the validation set (See Appendix K for details.). We evaluate efficacy on SABEval and general tool-use performance on ACEBench (single and parallel settings[8]) (Chen et al., 2025), and compare our method against the prompt baseline[9].

The results are shown in Table 3, highlighting three key observations: (1) Our method substantially reduces structural alignment bias, achieving a 45.55% average reduction in tool invocation rate, whereas the prompt baseline shows more limited improvements (11.19% average reduction). (2) With disjoint tool templates, rebalancing the pathways discovered from tools in the training set remains effective in mitigating bias on samples generated from unseen tools. This indicates that the pathways are not tool-specific but generalize across tools. (3) Our method has negligible impact on general tool-use capabilities. Results show that our method introduces only minimal accuracy changes on ACEBench, which are comparable to those introduced by prompting.

6 Related Work

Irrelevant Tools. Prior studies have explored scenarios involving irrelevant tools. Xu et al. (2024) frame tool irrelevance as a form of task insolvability and introduce a dedicated "ChangeTools action", while Huang et al. (2023); Ross et al. (2025); Patil et al. (2025) require LLMs to explicitly acknowledge when no tool is applicable. Other works further expect models to clarify irrelevance or address it through user interaction (Chen et al., 2025; Liu et al., 2025b; Wang et al., 2025). However, these works ignore the factor of structural alignment, thereby obscuring the underlying bias.

Irrelevant Context. Although irrelevant tools appear as part of an LLM's context, they differ fundamentally from irrelevant context studied in prior works (Cuconasu et al., 2024; Wu et al., 2024a; Niu et al., 2025; Cheng et al., 2025). First, irrelevant tools change the model's expected behavior (tool invocation to refusal), whereas irrelevant context does not. Moreover, despite prior work identifying the competition between irrelevant context and internal knowledge (Cheng et al., 2025), the com-

[8]The model must correctly select one or simultaneously multiple tools, and then fulfill the desired parameters. We report the accuracy metric.

[9]Prompting the model to refuse the invocation when no suitable tool is available. See Appendix L for details. We also compare with one-shot prompting and supervised fine-tuning in Appendix M.3.

peting pathways found in our work arise entirely from irrelevant tools themselves, reflecting two distinct query–tool matching mechanisms.

Mechanistic Interpretability. Mechanistic interpretability aims to reverse-engineer the computations of LLMs to uncover the causal mechanisms behind their behaviors (Olah, 2022; Wang et al., 2022). Recent studies indicate that contextual information is often utilized by specific attention heads (Wu et al., 2024b; Jin et al., 2024; Crosbie and Shutova, 2025), employing causal mediation analysis that relies on counterfactual interventions on model components (Vig et al., 2020; Meng et al., 2022; Syed et al., 2024; Haklay et al., 2025). By extending this framework, we gain an internal mechanistic understanding of the causes of structural alignment bias and provide the mitigation method.

7 Conclusion

In this work, we identify structural alignment bias, which can lead LLMs to erroneously invoke irrelevant tools, potentially resulting in serious and irreversible consequences. To study this problem, we construct a new dataset and conduct a comprehensive empirical analysis. Our results indicate that structural alignment bias is prevalent, can dominate tool-refusal behavior, and substantially increases erroneous invocations as structural alignment strengthens. Moreover, we introduce CAA to link this behavioral pattern to competing internal signals. Building on these insights, we further design a rebalancing strategy that mitigates this bias without sacrificing general tool-use capabilities.

Limitations

To rigorously isolate structural alignment from semantic relevance, we focus on the irrelevance setting in tool invocations and therefore do not extend our analysis to multi-turn agentic workflows. Nevertheless, understanding this fundamental bias in isolation provides a critical prerequisite for addressing it in more complex settings. Moreover, although we systematically analyze structural alignment bias, we do not distinguish whether it originates from general pre-training or is specifically introduced during tool-use fine-tuning. We leave the precise pinpointing of its source to future work.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No.62402491, No.U2336202), the China Postdoctoral Science Foundation (No.2025M771524), and the Shanghai Pujiang Program (24PJA004).

References

Andy Arditi, Oscar Obeso, Aaquib Syed, Daniel Paleka, Nina Panickssery, Wes Gurnee, and Neel Nanda. 2024. Refusal in language models is mediated by a single direction. *Advances in Neural Information Processing Systems*, 37:136037–136083.

Chen Chen, Xinlong Hao, Weiwen Liu, Xu Huang, Xingshan Zeng, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Yuefeng Huang, and 1 others. 2025. Acebench: Who wins the match point in tool usage? *arXiv preprint arXiv:2501.12851*.

Ziling Cheng, Meng Cao, Marc-Antoine Rondeau, and Jackie CK Cheung. 2025. Stochastic chameleons: Irrelevant context hallucinations reveal class-based (mis)generalization in LLMs. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 30187–30214, Vienna, Austria. Association for Computational Linguistics.

Joy Crosbie and Ekaterina Shutova. 2025. Induction heads as an essential mechanism for pattern matching in in-context learning. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 5034–5096, Albuquerque, New Mexico. Association for Computational Linguistics.

Florin Cuconasu, Giovanni Trappolini, Federico Siciliano, Simone Filice, Cesare Campagnano, Yoelle Maarek, Nicola Tonellotto, and Fabrizio Silvestri. 2024. The power of noise: Redefining retrieval for rag systems. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 719–729.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, and 6 others. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2021/framework/index.html.

Shen Gao, Zhengliang Shi, Minghang Zhu, Bowen Fang, Xin Xin, Pengjie Ren, Zhumin Chen, Jun Ma, and Zhaochun Ren. 2024. Confucius: Iterative tool learning from introspection feedback by easy-to-difficult curriculum. In *Proceedings of the AAAI conference*

on artificial intelligence, volume 38, pages 18030–18038.

Tal Haklay, Hadas Orgad, David Bau, Aaron Mueller, and Yonatan Belinkov. 2025. Position-aware automatic circuit discovery. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2792–2817, Vienna, Austria. Association for Computational Linguistics.

Michael Hanna, Ollie Liu, and Alexandre Variengien. 2023. How does gpt-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. *Advances in Neural Information Processing Systems*, 36:76033–76060.

Sam Houliston, Ambroise Odonnat, Charles Arnal, and Vivien Cabannes. 2025. Provable benefits of in-tool learning for large language models. *arXiv preprint arXiv:2508.20755*.

Yue Huang, Jiawen Shi, Yuan Li, Chenrui Fan, Siyuan Wu, Qihui Zhang, Yixin Liu, Pan Zhou, Yao Wan, Neil Zhenqiang Gong, and 1 others. 2023. Metatool benchmark for large language models: Deciding whether to use tools and which to use. *arXiv preprint arXiv:2310.03128*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Zhuoran Jin, Pengfei Cao, Hongbang Yuan, Yubo Chen, Jiexin Xu, Huaijun Li, Xiaojian Jiang, Kang Liu, and Jun Zhao. 2024. Cutting off the head ends the conflict: A mechanism for interpreting and mitigating knowledge conflicts in language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 1193–1215, Bangkok, Thailand. Association for Computational Linguistics.

Xixun Lin, Yucheng Ning, Jingwen Zhang, Yan Dong, Yilong Liu, Yongxuan Wu, Xiaohua Qi, Nan Sun, Yanmin Shang, Kun Wang, and 1 others. 2025. Llm-based agents suffer from hallucinations: A survey of taxonomy, methods, and directions. *arXiv preprint arXiv:2509.18970*.

Jian Liu, Kangyun Ning, Yisong Su, Wenjuan Han, Jinan Xu, and Yuanzhe Zhang. 2025a. Wtu-eval: A whether-or-not tool usage evaluation benchmark for large language models. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.

Jingyu Liu, JingquanPeng JingquanPeng, Xiaopeng Wu, Xubin Li, Tiezheng Ge, Bo Zheng, and Yong Liu. 2025b. Do not abstain! identify and solve the uncertainty. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 17177–17197, Vienna, Austria. Association for Computational Linguistics.

Weiwen Liu, Xu Huang, Xingshan Zeng, xinlong hao, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, Zezhong WANG, Yuxian Wang, Wu Ning, Yutai Hou, Bin Wang, Chuhan Wu, Wang Xinzhi, Yong Liu, Yasheng Wang, and 8 others. 2025c. ToolACE: Winning the points of LLM function calling. In *The Thirteenth International Conference on Learning Representations*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372.

Bertrand Meyer. 1997. *Object-oriented software construction*, volume 2. Prentice hall Englewood Cliffs.

Mahmoud Mohammadi, Yipeng Li, Jane Lo, and Wendy Yip. 2025. Evaluation and benchmarking of llm agents: A survey. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 6129–6139.

Neel Nanda. 2023. Attribution patching: Activation patching at industrial scale.

Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. 2025. A comprehensive overview of large language models. *ACM Transactions on Intelligent Systems and Technology*, 16(5):1–72.

Jingcheng Niu, Xingdi Yuan, Tong Wang, Hamidreza Saghir, and Amir H. Abdi. 2025. Llama see, llama do: A mechanistic perspective on contextual entrainment and distraction in LLMs. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16218–16239, Vienna, Austria. Association for Computational Linguistics.

Chris Olah. 2022. Mechanistic interpretability, variables, and the importance of interpretable bases.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, 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*.

Judea Pearl. 2001. Direct and indirect effects. In *Proc. of the Conference on Uncertainty in Artificial Intelligence*.

Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-Rong Wen. 2025. Tool learning with large language models: A survey. *Frontiers of Computer Science*, 19(8):198343.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*

and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Hayley Ross, Ameya Sunil Mahabaleshwarkar, and Yoshi Suhara. 2025. When2Call: When (not) to call tools. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3391–3409, Albuquerque, New Mexico. Association for Computational Linguistics.

Nishant Subramani, Jason Eisner, Justin Svegliato, Benjamin Van Durme, Yu Su, and Sam Thomson. 2025. MICE for CATs: Model-internal confidence estimation for calibrating agents with tools. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 12362–12375, Albuquerque, New Mexico. Association for Computational Linguistics.

Aaquib Syed, Can Rager, and Arthur Conmy. 2024. Attribution patching outperforms automated circuit discovery. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 407–416.

Henrique Schechter Vera, Sahil Dua, Biao Zhang, Daniel Salz, Ryan Mullins, Sindhu Raghuram Panyam, Sara Smoot, Iftekhar Naim, Joe Zou, Feiyang Chen, and 1 others. 2025. Embeddinggemma: Powerful and lightweight text representations. *arXiv preprint arXiv:2509.20354*.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. 2020. Investigating gender bias in language models using causal mediation analysis. *Advances in neural information processing systems*, 33:12388–12401.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. 2022. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. *arXiv preprint arXiv:2211.00593*.

Wenxuan Wang, Shi Juluan, Zixuan Ling, Yuk-Kit Chan, Chaozheng Wang, Cheryl Lee, Youliang Yuan, Jentse Huang, Wenxiang Jiao, and Michael R. Lyu. 2025. Learning to ask: When LLM agents meet unclear instruction. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 21784–21795, Suzhou, China. Association for Computational Linguistics.

Watt-AI. 2024. Watt-tool 8b. Accessed: 2025-12-08.

Siye Wu, Jian Xie, Jiangjie Chen, Tinghui Zhu, Kai Zhang, and Yanghua Xiao. 2024a. How easily do irrelevant inputs skew the responses of large language models? *arXiv preprint arXiv:2404.03302*.

Wenhao Wu, Yizhong Wang, Guangxuan Xiao, Hao Peng, and Yao Fu. 2024b. Retrieval head mechanistically explains long-context factuality. *arXiv preprint arXiv:2404.15574*.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, and 1 others. 2025. The rise and potential of large language model based agents: A survey. *Science China Information Sciences*, 68(2):121101.

Hongfei Xia, Hongru Wang, Zeming Liu, Qian Yu, Yuhang Guo, and Haifeng Wang. 2025. Safetoolbench: Pioneering a prospective benchmark to evaluating tool utilization safety in llms. *arXiv preprint arXiv:2509.07315*.

Hongshen Xu, Zichen Zhu, Lei Pan, Zihan Wang, Su Zhu, Da Ma, Ruisheng Cao, Lu Chen, and Kai Yu. 2024. Reducing tool hallucination via reliability alignment. *arXiv preprint arXiv:2412.04141*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Yuxiang Zhang, Jing Chen, Junjie Wang, Yaxin Liu, Cheng Yang, Chufan Shi, Xinyu Zhu, Zihao Lin, Hanwen Wan, Yujiu Yang, Tetsuya Sakai, Tian Feng, and Hayato Yamana. 2024. ToolBeHonest: A multi-level hallucination diagnostic benchmark for tool-augmented large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11388–11422, Miami, Florida, USA. Association for Computational Linguistics.

A Dataset Construction

By leveraging the polymorphism principle of Object-Oriented Programming (Meyer, 1997), we construct groups of sibling tools that share an identical interface (i.e., parameters) but possess mutually exclusive functionalities (i.e., they belong to different derived classes), enabling us to create samples that are structurally aligned yet semantically irrelevant. In what follows, we detail the construction procedure in three steps:

Hierarchical Tool Construction. We select seed tools from two mainstream benchmarks: BFCL (Patil et al., 2025) and ACEBench (Chen et al., 2025). First, we sanitize these tools and convert them into a unified format. To minimize parsing noise and focus on decision logic, we filter out seed tools that contain nested (i.e., object type) parameters and mandate all parameters as required. Subsequently, we manually annotate a *base class* for each seed tool. These base classes are derived typically from the entities targeted by the seed tools (e.g., mapping "Basketball Player" to "Athlete"). Then, we annotate *tool templates* by replacing specific entity references in the seed tools with a generic placeholder, denoted as "`<class>`".

Based on tool functionality, we then annotate *derived classes* for each base class. We enforce two main constraints during this process:

(1) The seed tool interface must be directly applicable to every derived class.

(2) The derived classes must be semantically mutually exclusive, such that a tool instantiated for one derived class cannot resolve queries intended for another.

Finally, to instantiate specific derived tools, we simply replace the "`<class>`" placeholder with the target derived class name via string substitution. This process yields tools that share identical parameters but possess mutually exclusive functionalities. We provide an example in Fig. 24.

Query Generation. For each instantiated derived tool, we leverage GPT-4o (Hurst et al., 2024) to generate diverse user queries. To ensure the quality and validity of the data, we mainly enforce three constraints during the generation process:

(1) The query must explicitly incorporate information corresponding to every parameter defined in the tool.

(2) The query must explicitly mention the name of the derived class associated with the tool.

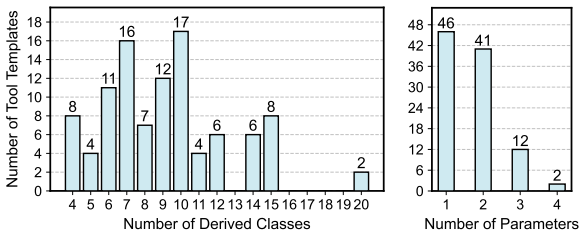

Figure 8: Distribution of derived class counts (left) and parameter counts (right) across tool templates.

(3) The query must be unambiguous and fully self-contained, requiring no further clarification to be understood.

The specific prompt template used for this generation is illustrated in Fig. 25.

Sibling Pairing. Sibling tools are those derived from the same tool template (i.e., sharing the same base class) but instantiated with distinct derived classes. Finally, we construct samples (t, q) by pairing a query, originally generated from a specific tool, with one of that tool's sibling tools. Through this pairing strategy, the pair (t, q) exhibits structural alignment because the tool accommodates every attribute present in the query, yet it remains semantically irrelevant.

Data Statistics. Since the seed tools are sourced from benchmarks with broad coverage, SABEval naturally inherits the domain diversity. Specifically, our dataset spans a wide spectrum of topics, including the seven domains explicitly identified in ACEBench: finance, society, health, entertainment, technology, environment, and culture (Chen et al., 2025).

In total, SABEval ($\mathcal{D}_0$) comprises 101 tool templates, each defined with 1 to 4 parameters. Across the base classes corresponding to these templates, we annotated a total of 729 distinct derived classes. On average, each tool template is instantiated into 9 derived tools, with at least 5 queries generated for each derived tool. We sample 10 distinct sibling pairs per tool template to construct the final samples. Fig. 8 shows the distributions of derived class counts and parameter counts across tool templates.

Dataset Extensions. As described in § 3, we construct four dataset extensions, $\mathcal{D}_1, \ldots, \mathcal{D}_4$, by augmenting the tool templates from $\mathcal{D}_0$ with $k \in \{1, 2, 3, 4\}$ additional parameters, respectively. We employ GPT-4o to synthesize these supplementary parameters, followed by a manual verification pro-

cess to ensure their validity. The specific prompt used for this process is provided in Fig. 26.

B System Prompts

The default system prompts for all models are shown in Figs. 27, 28.

C TIR Analysis Across Tool Templates

We randomly sample 10 tool templates from SABEval and compute the average TIR on the samples generated by each template, as well as the overall TIR aggregated across all 10 templates. As shown in Fig. 9, the TIR exhibits a highly consistent increasing trend with respect to the degree of structural alignment across different tool templates.

D Probability Decrease in Structural Counterfactuals

We investigate the probability shifts of w_{tool} in structural counterfactuals. A decrease in $P(w_{\text{tool}})$ directly reflects a reduced tendency for tool invocation. Specifically, we count samples for which the probability drops by more than 5% relative to the original probability, i.e., $P(w_{\text{tool}} \mid q, t^*) < 0.95 \cdot P(w_{\text{tool}} \mid q, t)$. As shown in Table 4, over 70% of counterfactual samples exhibit such decreases for the Qwen3 series. This proportion further increases to 90.65% and 96.45% for ToolACE-2.5-8B and Watt-Tool-8B, respectively. These results further substantiate the strong causal effect of structural alignment on tool invocations.

Model	Add	Remove	Substitute
Qwen3-4B	38.81	68.57	**70.30**
Qwen3-8B	42.65	72.67	**76.71**
Qwen3-14B	42.43	66.48	**73.76**
ToolACE-2.5-8B	81.92	80.55	**90.65**
Watt-Tool-8B	86.63	91.05	**96.45**

Table 4: Percentage of samples in which the probability of w_{tool} drops by more than 5% relative to its original value under structural counterfactuals.

E Discussion about Textual Similarity

Compared to the structural counterfactual samples obtained in § 4.2, the original structurally aligned samples exhibit more shared concepts and fewer semantic conflicts. Consequently, they naturally yield higher semantic textual similarity[10]. This observation is intuitive, as textual similarity can be viewed as a downstream consequence of structural alignment.

Models	Original	Instantiate Param Count			
		1	2	3	4
$\text{sim}(t, q)$	52.26	54.72	57.48	60.57	63.09
Qwen3-4B	89.76	87.39	81.56	73.50	65.11
Qwen3-8B	83.05	77.64	71.60	64.30	57.07
Qwen3-14B	90.44	88.10	82.14	77.96	70.99
ToolACE-2.5	71.47	69.66	70.00	66.71	63.09
Watt-Tool	39.98	38.93	34.42	31.39	24.87

Table 5: TIR for cases flattened with different numbers of parameters (from $\mathcal{D}_4$ of SABEval). The similarity score $\text{sim}(t, q)$ is also reported.

However, we argue that structural alignment implies more than just elevated textual similarity; specifically, textual similarity alone is insufficient to explain the observed bias. We demonstrate this through a *structural flattening* experiment, where we move target parameters from the tool definition to the tool description by instantiating them with query attributes (see Table. 6). This operation increases the naive textual similarity between the tool and the query, as the tool description now contains exact mentions of the query attributes. We then measure the cosine similarity between the tool and the query embeddings[11] and evaluate TIR.

As shown in Table 5, instantiating more parameters improves textual similarity; however, TIR declines as LLMs need to predict fewer parameters, suggesting that merely increasing textual similarity does not consistently drive up TIR. This indicates that the structural alignment bias is systematically induced by the structured nature of the data rather than naive textual similarity.

F Special Tokens

Tool-call Tokens. When invoking a tool, LLMs are instructed to start their responses with a special token w_{tool}, which enables us to analyze the tool-invocation behavior. The concrete w_{tool} for each model and their output formats are shown in Table 7.

We also inspect the full generated texts on $\mathcal{D}_0$ of SABEval and find no cases where LLMs emit

[10] Commonly measured by cosine similarity of dense vector representations, e.g., Sentence-BERT (Reimers and Gurevych, 2019).

[11] We use EmbeddingGemma-300M (Vera et al., 2025).

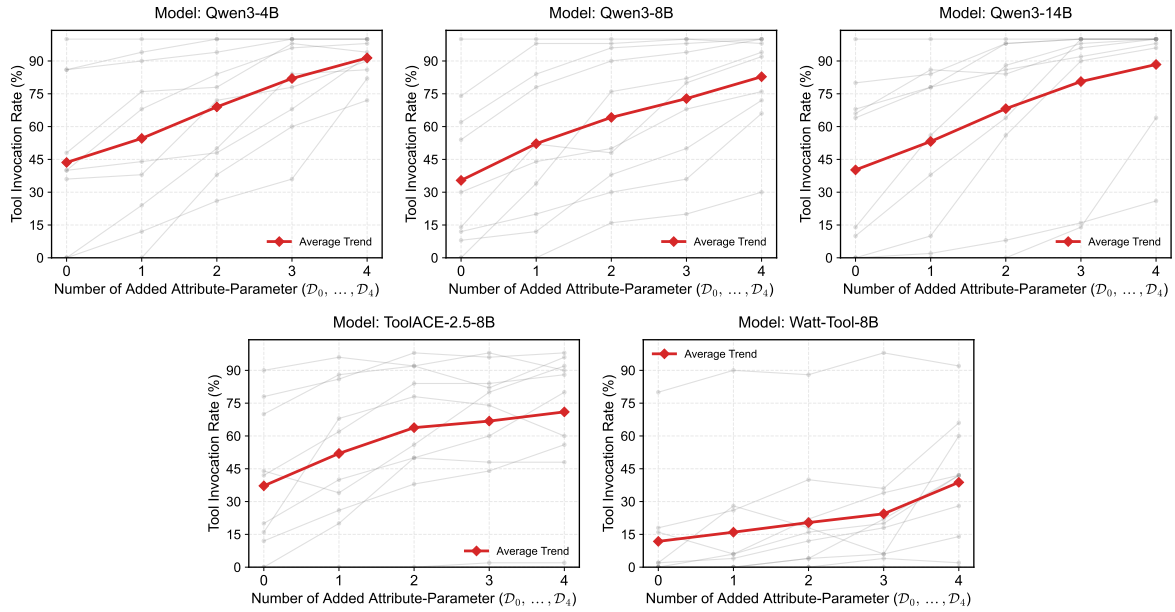

Figure 9: Tool invocation rate (%) across 10 randomly sampled tool templates. The red line denotes the overall TIR aggregated over these 10 groups, while the gray lines represent individual tool template groups.

User Query: I need **hiking shoes** recommendations for males dealing with overpronation. **Tool Name: sandals**_prep	
Original	**Tool Description:** Provides a specific list of **sandals** recommendations for common foot conditions. **Tool Parameters:** `foot_condition`, `gender`
Flattened	**Tool Description:** Provides a specific list of male **sandals** recommendations for common foot conditions. **Tool Parameters:** `foot_condition`

Table 6: An illustration of the structurally flattened case.

this token without invoking a provided tool (i.e., hallucinating a non-existent tool).

Refusal Tokens. Consistent with observations of Arditi et al. (2024), we find that model generations typically start with specific phrases when rejecting a tool. Therefore, for each model, we define a set of refusal tokens $\mathcal{R} = \{w_{r_1}, \ldots, w_{r_k}\} \subset \mathcal{V}$, and then define the semantic checking metric m_{sem} and structural matching metric m_{str} in Eq. (4) and Eq. (5). The concrete refusal tokens we choose for each model are shown in Table 8. These tokens cover over 98% of refusal cases.

We use the log-probability difference between the tool-call token and refusal tokens, rather than the former alone, for two reasons. First, when a model is highly confident, absolute probabilities saturate near 0 or 1. In this Softmax region, gradients approach zero, causing attribution patching (Eq. 3) to severely underestimate the causal impact of attention heads and potentially leading to numerical instability. Second, the difference metric isolates the competition between invocation and refusal by canceling out the Softmax denominator, eliminating confounding effects from other tokens. For example, if the model assigns high probability to a malformed tool invocation, the probability of the tool-call token would decrease due to the unit-sum constraint, conflating formatting errors with genuine refusal.

G Span Partitioning

We partition the input text into contiguous and non-overlapping spans to measure group-wise importance. The specific partitions for different models are shown in Tables 12, 13.

H Semantic and Structural Pathways

We visualize the identified pathways for other models in Figs. 10, 11, 12, 13.

Across all models, tool information primarily flows to the query positions and subsequently to the last span (either directly or via the end position of user message). The Qwen3-4B and 14B models exhibit characteristics consistent with the 8B version.

Model	Tool-call token	Tool-call token string	Tool-call format
Qwen3 family	151657	"<tool_call>"	<tool_call> {"name": <func-name>, "arguments": <args-json-object>} </tool_call>
ToolACE-2.5-8B	58	"["	[func_1(param_a=value_a, param_b=value_b...),func_2(params)]
Watt-Tool-8B	58	"["	[func_1(param_a=value_a, param_b=value_b...),func_2(params)]

Table 7: Tool-call tokens w_{tool}, their string representations, and concrete tool-call formats

Model	Refusal tokens	Refusal tokens string	Example refusal
Qwen3 family	{785, 40, 4064, 2132}	{"The", "I", "None", "It"}	"It seems there might be a misunderstanding. The provided function is for comparing VR headsets, not drones. If you'd like to compare VR headsets....."
ToolACE-2.5-8B	{791, 40}	{"The", "I"}	"I'm sorry, but I can only provide the current price for a game on the PlayStation platform. I do not have access to GOG prices."
Watt-Tool-8B	{791, 40}	{"The", "I"}	"The given question lacks the parameters required by the function. The available function is for basketball, not lacrosse."

Table 8: Refusal tokens for each model, corresponding string representations, and an example refusal. These tokens cover over 98% of refusal cases.

However, ToolACE-2.5-8B and Watt-Tool-8B diverge in their pathway distributions. Specifically, ToolACE-2.5-8B shows denser semantic pathways at the tool positions, whereas Watt-Tool-8B exhibits more focused structural pathways on these positions, and neither model concentrates its structural pathways on the tool output format. Moreover, for ToolACE-2.5-8B, the semantic pathways heavily attend to the tool instruction, which explicitly outlines tool rejection, suggesting a dependency on this prompt during semantic checking.

I Pathway Strengths Across Degrees of Structural Alignment

We present the comparison of pathway strengths between invocation and non-invocation cases for other models in Figs. 14, 15, 16, 17.

Here, we further investigate how semantic and structural pathways vary with increasing structural alignment. Across $\mathcal{D}_0, \ldots, \mathcal{D}_4$, we zero-ablate the top-k attention weights of the semantic and structural pathways, where k equals the 2% of the LLM attention heads, and report the results in Fig. 18. These results further indicate that the relative strength of the semantic and structural pathways drives LLMs' tool invocation decisions.

J Pathways Intervention

We further examine other models and alternative top-k thresholds, and report the variation in TIR under different scaling factors, as shown in Figs. 19, 20, 21, 22, 23.

K Selection of Scaling Coefficient

We select the optimal ρ_{sem} and ρ_{str} on the validation set by grid search: $\rho_{\text{sem}} \in \{1.1, 1.2, 1.3, 1.4, 1.5\}$ and $\rho_{\text{str}} \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$. We retain combinations that achieve at least a 60% relative reduction in TIR on the original samples, while the increase in TIR on semantic counterfactuals is no more than 3%[12]. Among the remaining candidates, we select the combination that minimizes the sum of TIR on the original samples and the non-invocation rate on semantic counterfactual samples.

L Prompt Baseline

We additionally append the following instruction to the end of the system prompt: "*If none of the tools can be used, point it out.*" We then evaluate LLMs under this setting.

[12]We relax the 3% threshold to 6% since no combination is feasible for Qwen3-4B.

M Additional Analysis on Qwen3 Models

M.1 Impact of Thinking Mode

We evaluate the impact of enabling the thinking mode on Qwen3-8B and 14B models. Specifically, we measure TIR on a subset of $\mathcal{D}_1$ (randomly selecting 10 tool templates and 5 sibling pairs per template) under the "Substitute" intervention strategy, and compare the results against the default non-thinking setting on the same data points. As shown in Table 9, test-time scaling via the thinking mode reduces erroneous tool invocations, particularly for the 14B model. However, directly comparing reasoning and non-reasoning modes is not entirely fair, as the former consumes significantly more compute and response time. Moreover, the error rates under the thinking mode remain substantial (30.8% and 29.2%), further highlighting the prevalence of structural alignment bias.

Strategy	Qwen3-8B		Qwen3-14B	
	Disabled	Enabled	Disabled	Enabled
Original	47.2	**30.8**	50.8	**29.2**
Substitute	21.6	**15.6**	26.8	**14.0**

Table 9: TIR (%) with thinking mode disabled and enabled on Qwen3-8B and 14B models, evaluated on a subset of $\mathcal{D}_1$.

M.2 Impact of Model Size

We further evaluate Qwen3-32B on $\mathcal{D}_1$ using the same settings as Fig. 4. The results are shown in Table 10. While Qwen3-8B exhibits a notably lower TIR, this is not observed in the larger Qwen3-32B, which instead yields the highest error rate (62.22%) across all model sizes. Overall, TIR does not change monotonically with model size: mid-sized models (8B, 14B) show lower rates, whereas both the smaller (4B) and larger (32B) models exhibit higher rates.

M.3 Additional Baselines

We compare our method with one-shot prompting and supervised fine-tuning on Qwen3-8B. The results are shown in Table 11.

One-shot Prompting. We randomly select one training sample to demonstrate the correct refusal behavior, and evaluate two formats: (1) *One-shot (raw)*, which includes the complete tool definition, the user query, and the expected refusal response; and (2) *One-shot (summarized)*, which provides a natural language description of the refusal logic (i.e., "If a tool is designed to retrieve the most visited zoo, but the user asks for the most visited museum, you should not call the tool and instead explicitly state it is unsuitable."). Note that the "Prompt" method in Table 3 serves as the zero-shot baseline. Our results show that one-shot prompting fails to improve performance. Surprisingly, the raw format significantly degrades it, suggesting that exposing the model to additional complete structural examples exacerbates structural alignment bias rather than mitigating it.

Strategy	Qwen3-4B	Qwen3-8B	Qwen3-14B	Qwen3-32B
Original	60.44	53.43	60.40	**62.22**
Add	52.85	46.20	50.24	**55.41**
Remove	33.80	26.08	35.27	**41.52**
Substitute	32.08	23.17	29.90	**34.47**

Table 10: TIR (%) across different Qwen3 model sizes on $\mathcal{D}_1$.

Method	$\mathcal{D}_0$	$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_4$
Base	30.39	51.61	66.29	75.56	82.44
Prompt	19.90	34.29	48.73	57.66	66.20
CAA (Ours)	4.29	6.49	7.27	8.98	11.12
One-shot (r)	63.71	84.05	91.22	94.78	96.98
One-shot (s)	30.54	49.37	60.14	69.80	79.95
Fine-tuning	**1.41**	**2.48**	**2.10**	**3.51**	**4.39**

Table 11: TIR (%) of additional baselines on Qwen3-8B across SABEval. (r) and (s) denote raw and summarized one-shot formats, respectively.

Supervised Fine-Tuning. We fine-tune Qwen3-8B using LoRA (rank=8, α=8, dropout=0.05) on all parameters and select the best checkpoint over 2 epochs. While fine-tuning achieves strong in-distribution performance on SABEval, it fundamentally acts as a black box that forces the model to fit refusal behaviors. Our method approaches the efficacy of fine-tuning without requiring any parameter updates, and allows flexible control over intervention strength at test time. Moreover, our method is interpretable. It serves not merely as a mitigation strategy, but as a direct causal validation of the competing semantic and structural pathways. This mechanistic understanding may also lay the groundwork for more effective future mitigation strategies, such as targeted fine-tuning on the attention heads within the identified pathways.

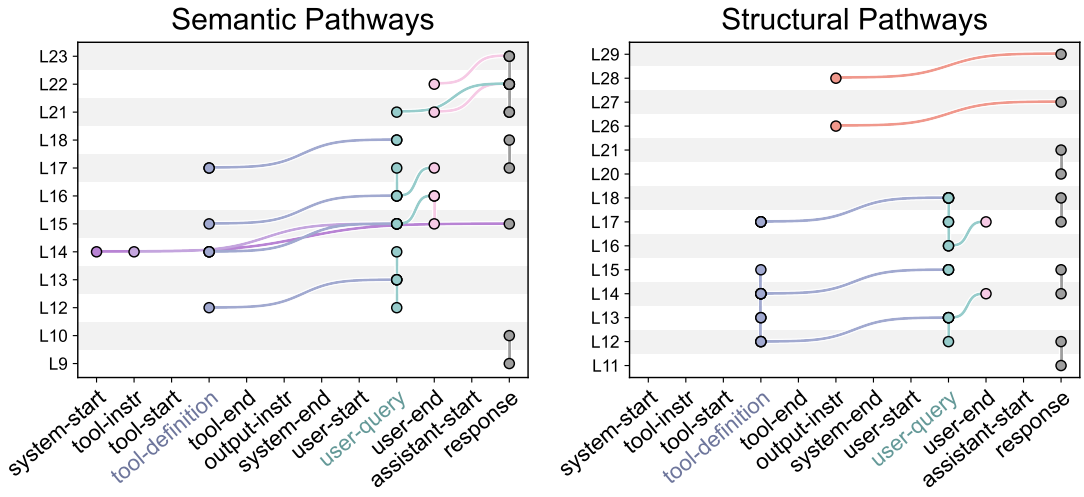

Figure 10: Pathways in Qwen3-4B.

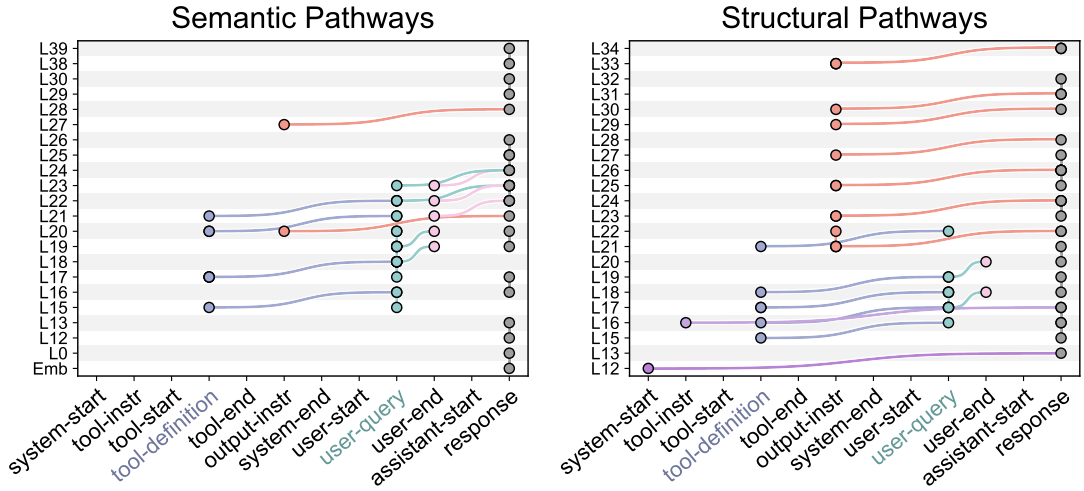

Figure 11: Pathways in Qwen3-14B.

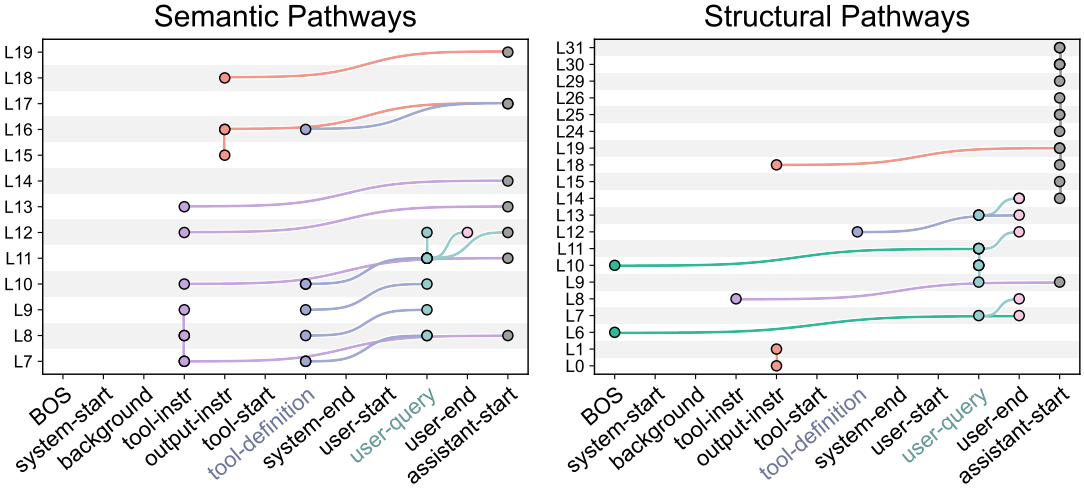

Figure 12: Pathways in ToolACE-2.5-8B.

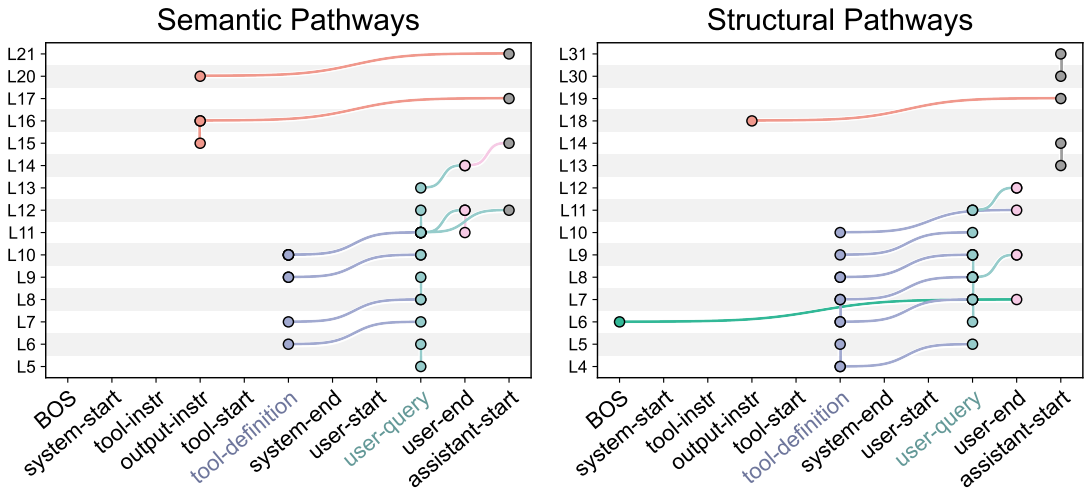

Figure 13: Pathways in Watt-Tool-8b.

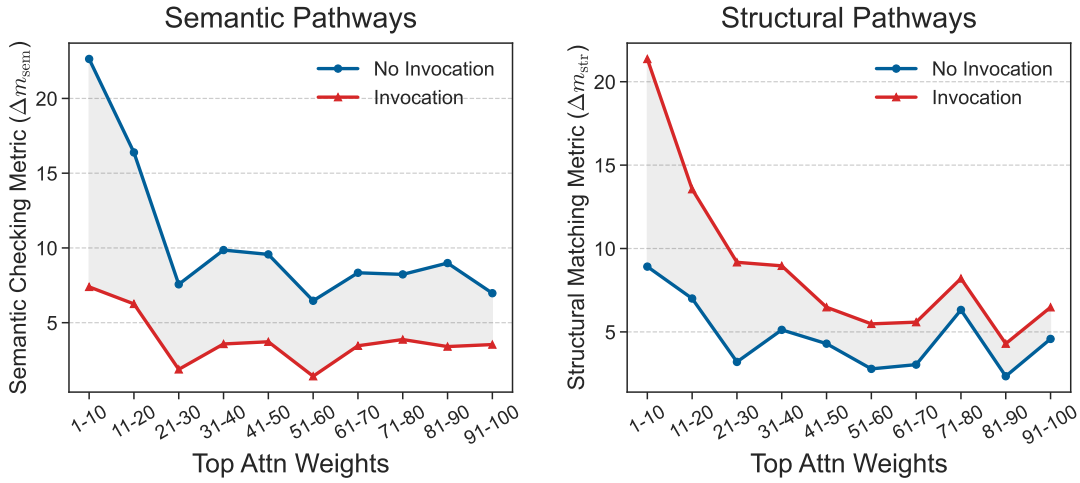

Figure 14: Comparison of pathway strengths between invocation and non-invocation cases for Qwen3-4B.

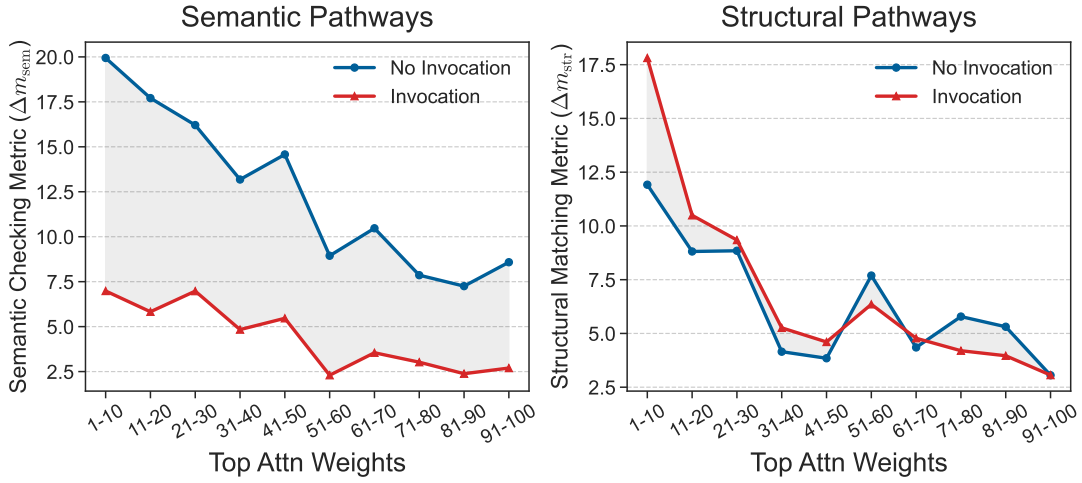

Figure 15: Comparison of pathway strengths between invocation and non-invocation cases for Qwen3-14B.

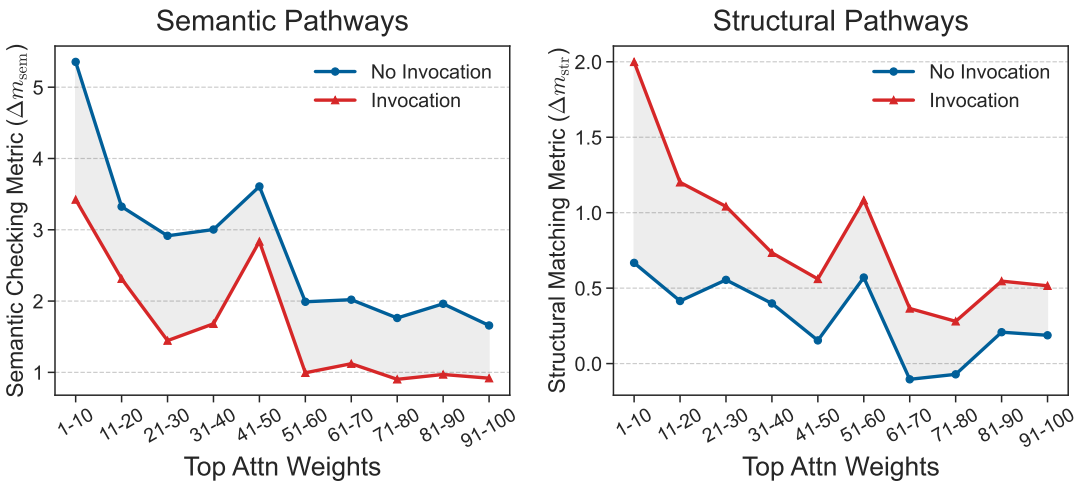

Figure 16: Comparison of pathway strengths between invocation and non-invocation cases for ToolACE-2.5-8B.

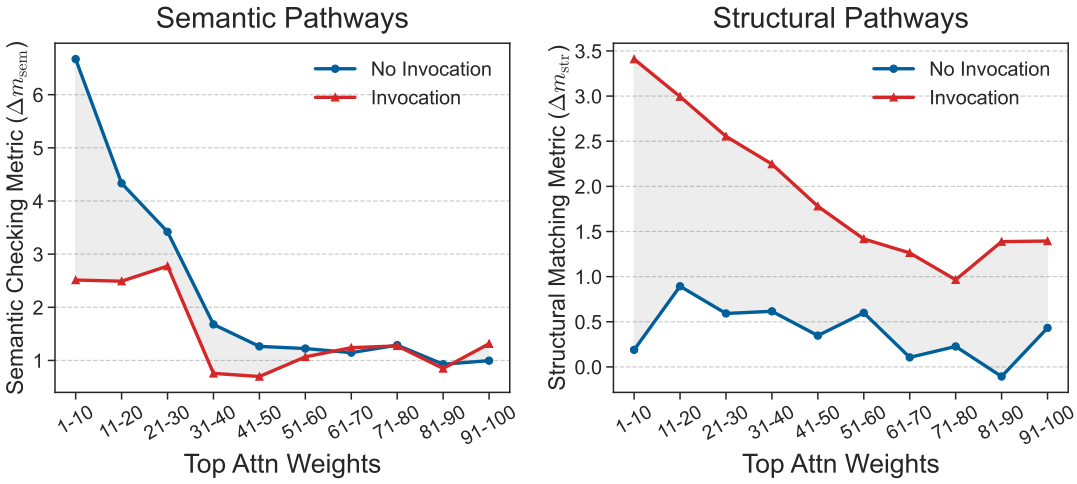

Figure 17: Comparison of pathway strengths between invocation and non-invocation cases for Watt-Tool-8B.

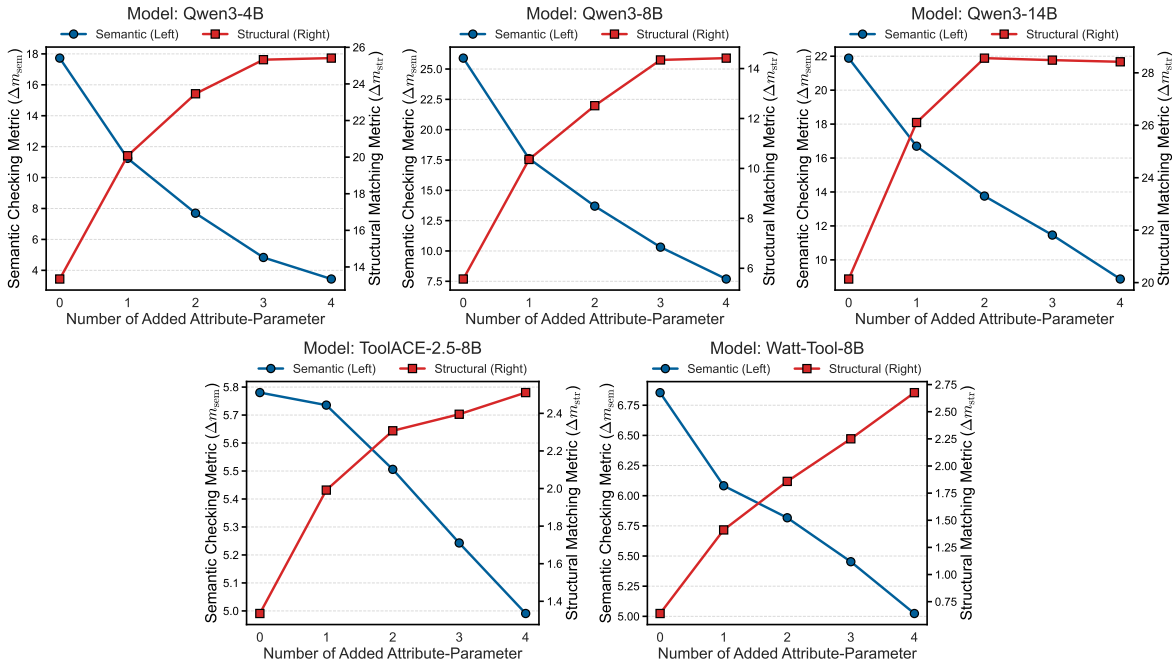

Figure 18: Pathway strengths across degrees of structural alignment for Qwen3-8B and ToolACE-2.5-8B.

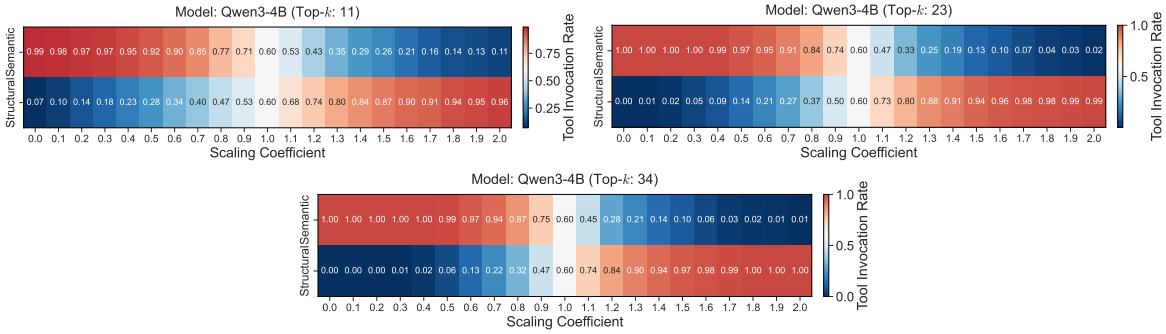

Figure 19: TIR at different top-k thresholds under varying scaling coefficients for Qwen3-4B.

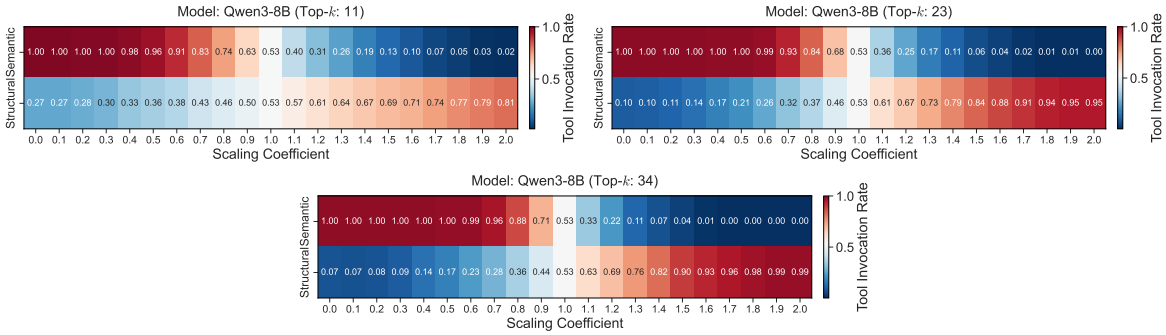

Figure 20: TIR at different top-k thresholds under varying scaling coefficients for Qwen3-8B.

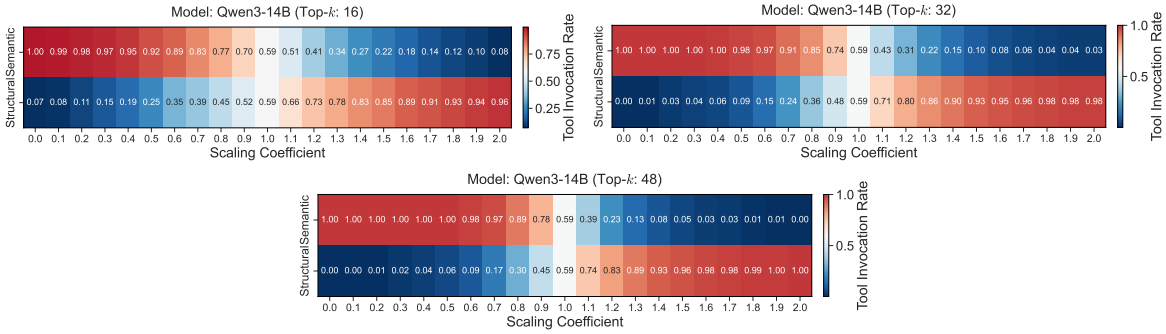

Figure 21: TIR at different top-k thresholds under varying scaling coefficients for Qwen3-14B.

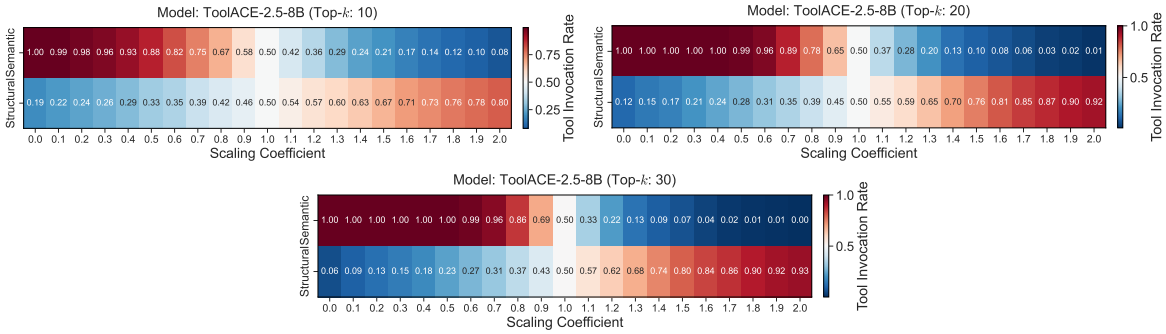

Figure 22: TIR at different top-k thresholds under varying scaling coefficients for ToolACE-2.5-8B.

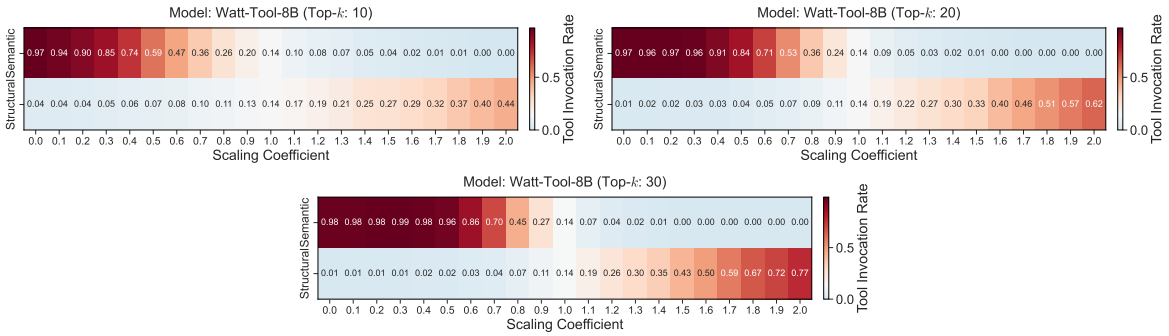

Figure 23: TIR at different top-k thresholds under varying scaling coefficients for Watt-Tool-8B.

Span Name	Span Content
system-start	"<lim_start|>system\n"
tool-instruction	"# Tools\n\nYou may call one or more functions to assist with the user query.\n\nYou are provided with function signatures within <tools></tools> XML tags:\n"
tool-start	"<tools>\n"
tool-definition	tool definition
tool-end	"</tools>\n\n"
output-instruction	For each function call, return a json object with function name and arguments within <tool_call></tool_call> XML tags:\n<tool_call>\n"name": <function-name>, "arguments": <args-json-object>\n</tool_call>
system-end	"<lim_end|>\n"
user-start	"<lim_start|>user\n"
user-query	user query
user-end	"<lim_end|>\n"
assistant-start	"<lim_start|>assistant\n"
response	"<think>\n\n</think>\n\n"

Table 12: Spans for Qwen3 models (4B, 8B, and 14B).

Span Name	Span Content
BOS	"<|begin_of_text|>"
system-start	"<|start_header_id|>system<|end_ header_id|>\n\n"
background	"Cutting Knowledge Date: December 2023\nToday Date: ...\n\n"
tool-instruction	"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.\nIf none of the function can be used, point it out. If the given question lacks the parameters required by the function, also point it out.\nYou should only return the function call in tools call sections.\n\n"
output-instruction	"If you decide to invoke any of the function(s), you MUST put it in the format of [func_name1(params_name1=params_value1, params_name2=params_value2...), func_name2(params)]\nYou SHOULD NOT include any other text in the response.\n"
tool-start	"Here is a list of functions in JSON format that you can invoke.\n"
tool-definition	tool definition
system-end	"<|eot_id|>"
user-start	"<|start_header_id|>user<|end_header_id|>\n\n"
user-query	user query
user-end	"<|eot_id|>"
assistant-start	"<|start_header_id|>assistant<|end_header_id|>\n\n"

Table 13: Spans for ToolACE-2.5-8B and Watt-Tool-8B models (identical partitioning, except Watt-Tool-8B has no background span).

Annotation Example: Base Class, Derived Class, Tool Template

```
{
    "base_class": "athlete",
    "derived_class": [
        "basketball",
        "soccer",
        "baseball",
        "american football",
        "ice hockey",
        "volleyball",
        "esports",
        "cricket",
        "rugby union",
        "handball",
        "futsal",
        "field hockey"
    ],
    "tool_template": {
        "name": "find_<class>_athlete",
        "description": "Find the profile information of a <class>
                        athlete based on their full name.",
        "parameters": {
            "type": "object",
            "properties": {
                "name": {
                    "type": "string",
                    "description": "The full name of the <class> athlete."
                }
            },
            "required": [
              "name"
            ]
        }
    }
}
```

Figure 24: An example annotation illustrating a base class, derived classes, and a tool template.

Prompt for Query Generation During Dataset Construction

```
Your task is to generate a set of diverse user queries for the given tool.
Note that the queries you generate represent real user queries directed at an LLM assistant.
The LLM assistant will need to call the provided tool to answer these queries.

Rules for Query Generation:
1. Derived Class Inclusion: Every generated query must contain the exact derived class name
provided.
2. Parameter Constraints:
  - Each query must explicitly contain information for all and only the parameters of the
    tool.
  - The expected parameter values must be specific and realistic. Avoid vague values.
3. No Attachments: Do not assume or pretend that files, images, audio clips, videos, or any
other attachments are being provided.
4. Quality: Generated queries must be solvable with the tool without requiring further
clarification.
5. Diversity:
  - You should generate at least 5 distinct queries.
  - The queries should have varied sentence structures (e.g., imperative commands,
    interrogative queries).
  - The parameter values across different queries should also be diverse, covering a wide
    range of realistic scenarios, if applicable.

Output Format:
Return a single JSON array as follows:
[ {{ "query": "First generated query..." }}, {{ "query": "Second generated query..." }}, ... ]

[EXAMPLE START]
{few-Shot examples}
[EXAMPLE END]

Now, generate queries for the following tool.

Tool Definition:
{tool}

Derived Class Name:
{subclass}
```

Figure 25: Query-generation prompt used during dataset construction.

Prompt for Adding Additional Parameters

```
Your task is to propose a set of new, additional parameters for the given tool template.
These parameters should enrich the tools's functionality while remaining applicable across all
derived classes.

You are Given:
1. Tool Template: An tool schema that uses '<class>' as a placeholder for a specific derived
class.
2. List of Derived Classes: A list of the specific derived class names that will eventually
replace '<class>'.
3. Base Class Description: A brief explanation of the base class corresponding to the tool
template.

Design Principles:
- The tool's purpose must remain clear and unambiguous after adding new parameters.
- All parameter values would be provided by the user when invoke the tool, not generated or
  assumed by the LLM assistant. The LLM acts as a bridge to execute the tool with user-
  provided information, not as a param value generator.

Rules:
1. Universally Applicable: Each proposed parameter must be universally applicable and make
sense for all derived classes provided.
2. Uniqueness: The proposed parameters must be entirely new. They cannot duplicate the
functionality or name of any parameters already present in the tool template.
3. Placeholder Usage: Parameter names must be generic and must not contain the '<class>'
placeholder. Parameter description, however, should contain the '<class>' placeholder if it is
contextually appropriate when replaced with a specific subclass.
4. Quantity: Generate at least four distinct and meaningful parameters.
5. Type: New parameters should be simple types: string, integer, number, boolean or array of
simple types. Do not propose complex nested structures.

Output Format:
Return a single JSON object as follows:
{{"parameter_name_1": {{"type": "string", "description": "Description for param 1."}}, "
parameter_name_2": {{"type": "integer", "description": "Description for param 2."}}, ...}}

[EXAMPLE START]
{few-Shot examples}
[EXAMPLE END]

Now, generate new parameters based on the following inputs.

Tool Template:
{tool template}

List of Derived Classes:
{derived class list}

Base Class Description:
{base class description}
```

Figure 26: Prompt for adding additional parameters during dataset extension.

System Prompt (Qwen3-4B, Qwen3-8B and Qwen3-14B)

```
# Tools

You may call one or more functions to assist with the user query.

You are provided with function signatures within <tools></tools> XML tags:
<tools>
{tool_definition}
</tools>

For each function call, return a json object with function name and arguments
within <tool_call></tool_call> XML tags:
<tool_call>
{"name": <function-name>, "arguments": <args-json-object>}
</tool_call>
```

Figure 27: Default system prompt for Qwen3-4B, Qwen3-8B and Qwen3-14B

System Prompt (ToolACE-8B and Watt-Tool-8B)

```
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 function 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 the function call in tools call sections.

If you decide to invoke any of the function(s), you MUST put it in the format
of [func_name1(params_name1=params_value1, params_name2=params_value2...),
func_name2(params)]
You SHOULD NOT include any other text in the response.
Here is a list of functions in JSON format that you can invoke.
{tool_definition}
```

Figure 28: Default system prompt for ToolACE-8B and Watt-Tool-8B