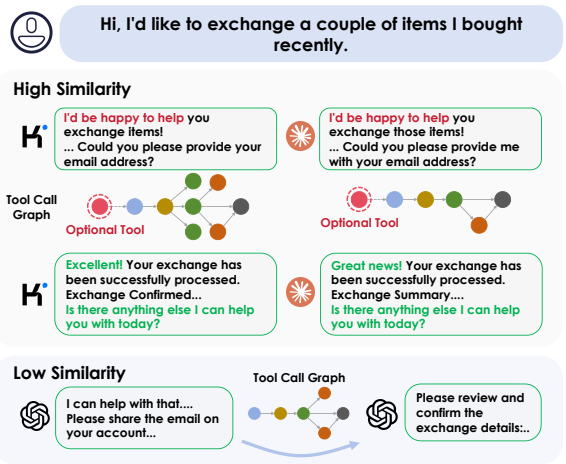
When Agents Look the Same: Quantifying Distillation-Induced Similarity in Tool-Use Behaviors

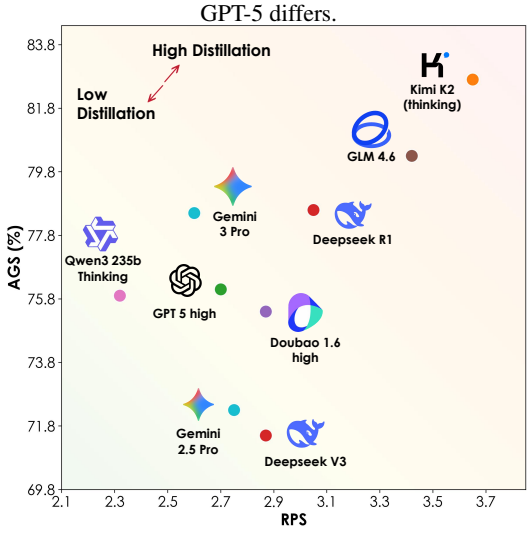
Anonymous ACL submission

Abstract

Model distillation is a primary driver behind the rapid progress of LLM agents, yet it often leads to behavioral homogenization. Many emerging agents share nearly identical reasoning steps and failure modes, suggesting they may be distilled echoes of a few dominant teachers. To ensure ecosystem robustness, it is critical to quantify this alignment. However, existing metrics fail to distinguish mandatory behaviors required for task success from non-mandatory patterns that reflect a model's autonomous preferences. In this work, we quantify agent distillation by isolating non-mandatory behavioral patterns via two complementary metrics: **Response Pattern Similarity (RPS)** for verbal alignment and **Action Graph Similarity (AGS)** for tool-use habits modeled as directed graphs. We evaluate 18 models from 8 providers on τ-Bench and τ^2-Bench using Claude 4.5 Sonnet as the reference. Experimental results reveal three key insights: (1) Anthropic models exhibit high internal consistency, validating our metrics; (2) Kimi-K2 shows the highest structural similarity to the teacher among non-Anthropic models, particularly in tool-dependency patterns; and (3) RPS and AGS capture distinct behavioral dimensions. By disentangling behavioral mimicry from task-driven necessity, our work provides a systematic tool to improve the transparency and independent development of the agent ecosystem.



(a) Kimi-K2 and Claude share verbal and action patterns; GPT-5 differs.



(b) RPS-AGS scatter: Kimi-K2 (thinking) in high-similarity region.

Figure 1: **Behavioral similarity comparison across LLM agents.** Higher RPS and AGS indicate greater alignment with the reference model (Claude Sonnet 4.5 (thinking)).

1 Introduction

The current "Cambrian explosion" of high-performing LLM agents often carries a persistent sense of déjà vu. Despite their diverse origins, many emerging agents exhibit a considerably aligned behavior: they share nearly identical reasoning steps, redundant tool-calling habits, and even the same failure modes (Song et al., 2024; Lyu et al., 2025; Kang et al., 2025). This suggests that instead of independent breakthroughs, these models may be distilled echoes of a few dominant teachers. Such pervasive mimicry leads to a convergence of "bad habits" across theoretically indepen-

dent models (Peng et al., 2025; Chen et al., 2025). For instance, rather than optimizing for efficiency, many agents mirror the teacher model's verbose reasoning and redundant tool-calling patterns, such as trial-and-error with every available tool, even when the solution is obvious (Lu et al., 2025; Xiong et al., 2025). This collective alignment means that the ecosystem lacks actual robustness; different models no longer provide independent verification but instead fail in the exact same way (Guo et al., 2024; Shumailov et al., 2024).

Quantifying this distillation-induced alignment is essential for ensuring ecosystem transparency, but existing methods fall short in the complex, multi-step agent landscape. Current metrics primarily focus on response-level similarity in static dialogues, which fails to capture the dynamic nature of tool-use trajectories (Lee et al., 2025). More critically, they struggle to distinguish between mandatory behaviors related to actions strictly required for task success and non-mandatory behaviors, which reflect a model's autonomous preferences. Without isolating these "behavioral degrees of freedom," it is impossible to determine whether two models converge because there is only one correct path, or because one is blindly shadowing the other.

To bridge this gap, we propose a systematic framework to quantify agent distillation by isolating non-mandatory behavioral patterns. Our approach introduces two complementary metrics: **Response Pattern Similarity** (RPS) and **Action Graph Similarity** (AGS).

RPS measures verbal similarity by segmenting trajectories into five canonical stages (authentication, elicitation, execution, verification, notification) and scoring similarity along style, structure, and alignment dimensions.

AGS measures action-level similarity through three sub-metrics. S_{node} captures *optional tool agreement*: for a flight cancellation task, all models must call `cancel_reservation`, but some additionally call `get_reservation_details` to double-check if two models share such optional choices, they likely share training signals. S_{seq} captures *sequential habits* such as post-write verification, pre-modification confirmation, and error retry patterns. S_{dep} captures *dependency patterns* such as output reuse rate and tool-chaining depth. By isolating these non-mandatory behaviors from task-dictated actions, our metrics reveal stylistic and structural alignment that would otherwise be masked by shared correctness.

We evaluate 18 models from 8 major providers on τ-Bench and τ^2-Bench, using Claude Sonnet 4.5 (thinking) as the reference "oracle". Our experiments reveal several key findings. (1) Anthropic models exhibit strong internal consistency with RPS scores above 3.8, validating our metrics. (2) Kimi-K2 (thinking) shows elevated similarity on both metrics, achieving the highest AGS at 82.7% among non-Anthropic models, with S_{node} at 82.6% and S_{dep} at 94.7%. (3) RPS and AGS capture distinct behavioral dimensions, providing complementary signals for distillation. Our findings provide empirical grounding for the initial déjà vu. While some models maintain high behavioral diversity, others exhibit systematic alignment with the reference model across both verbal and structural dimensions, even in scenarios where multiple alternative paths exist.

Our main contributions are summarized as follows.

- We propose the first framework specifically designed to quantify agent distillation by disentangling mandatory and non-mandatory behaviors.
- We introduce RPS and AGS, providing interpretable metrics for verbal and action-level mimicry.
- We conduct an extensive audit of current state-of-the-art agents, revealing the degree of behavioral homogenization in the current ecosystem.

2 Method

2.1 Trajectory Representation

Given a set of models $\mathcal{M} = \{M_1, ..., M_k\}$ and a tool-use task set $\mathcal{T}$, we collect execution trajectories $\{\tau_1, \tau_2, ...\}$ for each model on $\mathcal{T}$. Each trajectory τ consists of multiple dialogue turns. Each turn comprises a user input and a model output, where the model output includes (i) user-visible replies, (ii) tool calls, and (iii) tool-related messages such as intermediate outputs or execution traces. Figure 2 presents the overall framework.

Based on trajectory representation, we design two complementary metrics. Response Pattern Similarity (RPS) captures verbal fingerprints in how models phrase responses, structure explanations, and express reasoning. Action Graph Similarity (AGS) captures behavioral fingerprints in how models select tools, sequence operations, and reuse outputs. The two metrics capture different aspects of

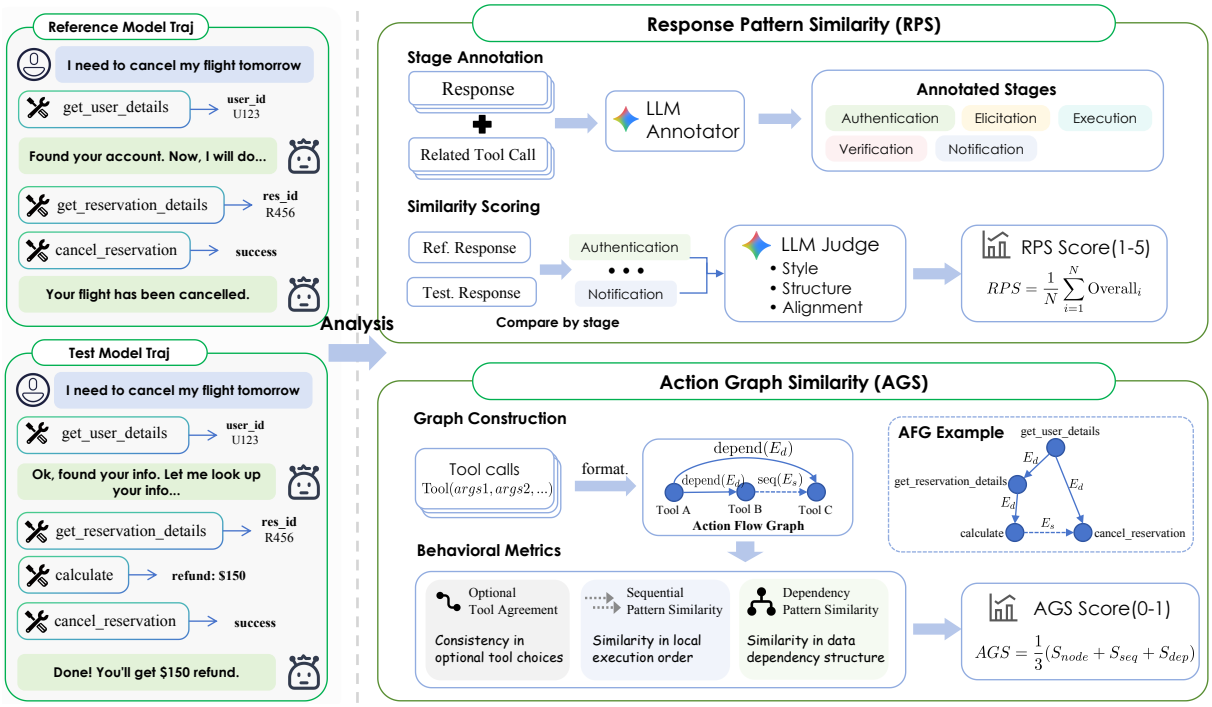


Figure 2: **Overview of our distillation quantification framework. Left**: Example trajectories from a reference model and a test model on a flight cancellation task. **Right top**: Response Pattern Similarity (RPS) pipeline, which segments trajectories into canonical stages and scores verbal similarity via an LLM judge. **Right bottom**: Action Graph Similarity (AGS) pipeline, which constructs Action Flow Graphs from tool call sequences and computes behavioral metrics based on node sets, sequential edges, and dependency edges.

model behavior: models may share verbal patterns but differ in tool usage, or vice versa.

2.2 Response Pattern Similarity

Response Pattern Similarity (RPS) quantifies verbal similarity between two models along three dimensions. *Style* measures wording habits and vocabulary preferences. *Structure* measures sentence patterns and response templates. *Alignment* measures whether both models exhibit similar reasoning-to-action patterns. Detailed definitions are provided in Appendix A.

Tool-use agent trajectories often span dozens of turns. Comparing entire trajectories at once exceeds typical LLM context limits and dilutes local patterns. We therefore design a two-stage pipeline as illustrated in Figure 2.

Stage Annotation. Tool-use benchmarks involve dynamic interactions where different models complete the same task in varying numbers of turns. For example, one model may authenticate in two turns while another requires four. Direct turn-by-turn alignment would compare unrelated content. To achieve semantic-level alignment, we define five canonical stages that characterize agent-user interactions: (i) *Authentication* involves identifying the user, verifying identity, or handling a retry after verification failure. (ii) *Elicitation* requests missing parameters from the user to proceed with the task, outside of authentication contexts. (iii) *Execution* invokes domain-specific tools to perform query or modification operations. (iv) *Verification* seeks explicit user confirmation before executing critical write operations. (v) *Notification* reports tool execution results or current status to the user. These five stages are derived from analyzing common interaction patterns in tool-use benchmarks and cover the typical agent workflow. Given a trajectory τ, we use an LLM to annotate each response and tool-related message to its corresponding stage. The annotation prompt and stage definitions are provided in Appendix A.1.

Similarity Scoring. After stage annotation, RPS compares only the stages shared by both models. Stages appearing in only one trajectory are excluded, as our goal is to measure similarity rather than coverage. For each shared stage, we input the responses, tool-related messages, and associated tool call contents into an LLM judge. The judge scores similarity along the Style, Structure, and Alignment dimensions, as well as a holistic Overall

score. Scores range from 1 to 5, where 5 indicates high similarity, 3 indicates moderate overlap, and 1 indicates no discernible similarity. The final RPS score is the mean of the Overall scores across all shared stages. The scoring prompt and detailed rubric are provided in Appendix A.3.

2.3 Action Graph Similarity

Action Graph Similarity (AGS) analyzes structural patterns in tool call sequences to characterize behavioral similarity at the level of tool invocation.

Graph Construction. Given a dialogue trajectory τ, we construct a directed graph $G = (V, E_s, E_d)$ based on the tool call sequence.

Definition 1 (Tool Node). Each node $v \in V$ represents a tool call, with attributes:

$$v = (\text{name}, \text{args}, \text{result})$$

where name is the tool name, args is the input arguments, and result is the return value.

Definition 2 (Sequential Edge). A sequential edge $(u, v) \in E_s$ connects temporally adjacent tool calls according to their execution order in τ.

Definition 3 (Dependency Edge). A dependency edge $(u, v) \in E_d$ is established when an argument value of v is derived from the result of u. Since simple string matching produces excessive false positives (e.g., common identifiers or dates appearing coincidentally), we use an LLM judge to verify semantic validity. For each candidate edge identified by string overlap, the LLM determines whether the matched value is actually obtained from the source tool's result or is known beforehand (e.g., from user input). The LLM judge achieves 96% accuracy on dependency edge detection (Appendix D.3). When a dependency edge exists between consecutive nodes, the sequential edge is omitted to avoid redundant encoding.

Behavioral Metrics. Based on the constructed graph G, AGS characterizes behavioral similarity along three dimensions. For sequential and dependency patterns, we design three heuristic features, each based on its interpretability.

Optional Tool Agreement S_{node} measures the consistency of optional tool choices between two models. When completing the same task, models invoke both mandatory tools required for task completion and optional tools for auxiliary purposes. To distinguish these two categories, we analyze successful trajectories from multiple models on the same task. Let $\text{Tools}(M, t)$ denote the set of tools invoked by model M on task t, and $\mathcal{M}_t^*$ denote the set of models that successfully complete task t. Mandatory tools are the intersection of tool sets across all successful models:

$$\mathcal{F}_t^{\text{mandatory}} = \bigcap_{M \in \mathcal{M}_t^*} \text{Tools}(M, t)$$

Optional tools are those appearing in some but not all successful trajectories. S_{node} computes the agreement rate on optional tools across all tasks, where agreement means both models either use or skip the same optional tool.

Sequential Pattern Similarity S_{seq} measures whether two models exhibit similar local execution habits between adjacent tool calls. We extract three features based on sequential edges E_s: (i) post-write verification rate, measuring the tendency to invoke a read operation after a write for verification; (ii) pre-write confirmation rate, measuring the tendency to query before executing a write; (iii) error retry rate, measuring the tendency to retry the same tool after an error. Each trajectory is represented as a three-dimensional feature vector. For each task, we compute the cosine similarity between the two models' feature vectors, then average across all tasks to obtain S_{seq}.

Dependency Pattern Similarity S_{dep} measures whether two models exhibit similar patterns in reusing tool outputs. We extract three features based on dependency edges E_d: (i) output reuse rate, measuring the proportion of tool calls that reuse outputs from preceding calls; (ii) most extended dependency chain length, measuring the depth of chained planning; (iii) output fan-out rate, measuring the proportion of tool outputs reused by multiple subsequent calls. Similar to S_{seq}, we compute per-task cosine similarity and average across tasks.

Detailed definitions and computation methods for the three sub-metrics are provided in Appendix B.

3 Experiments

3.1 Experiment Settings

Model Families. We evaluate 18 models from 8 major providers: Anthropic (Claude Sonnet 4.5 (Anthropic, 2025b), Claude Opus 4.1 (Anthropic, 2025a)), OpenAI (GPT-4.1 (OpenAI, 2025a), GPT-5 (OpenAI, 2025b)), DeepSeek (R1 (DeepSeek-AI et al., a), V3.1 (DeepSeek-AI

Table 1: Behavioral similarity with Claude Sonnet 4.5 (thinking) as reference. Sty.=Style, Str.=Structure, Ali.=Alignment. Anthropic models (shaded) serve as baselines. Among non-Anthropic models: **bold** = 1st, *italic* = 2nd, underline = 3rd.

Family	Model	Baseline				AGS (%)				RPS			
		GED	RSE	N-gram	BERT	S_{node}	S_{seq}	S_{dep}	**Avg**	Sty.	Str.	Ali.	**Overall**
ANTHROPIC	Sonnet 4.5 (no-think)	82.6	3.14	0.371	0.951	79.1	79.9	94.0	84.3	4.07	3.89	3.66	3.87
	Opus 4.1 (thinking)	79.7	3.25	0.355	0.946	81.0	74.4	93.7	83.0	4.02	3.80	3.72	3.85
OPENAI	GPT-4.1	75.2	2.45	0.306	0.936	75.9	*74.6*	88.0	79.5	3.07	2.92	3.45	3.15
	GPT-5	68.3	2.26	0.230	0.889	71.3	69.4	87.7	76.1	2.50	2.43	3.17	2.70
DEEPSEEK	R1	70.8	2.44	0.259	0.925	*78.3*	72.5	85.0	78.6	2.99	2.87	3.29	3.05
	V3.1 (thinking)	72.6	2.56	0.307	0.932	78.1	63.5	91.2	77.6	3.24	2.91	*3.43*	3.19
	V3.1 (no-think)	69.2	2.29	*0.312*	*0.931*	72.4	65.3	87.5	75.1	*3.55*	*3.27*	3.38	3.40
	V3-0324	59.1	1.87	0.270	0.917	77.5	62.7	74.5	71.5	2.84	2.66	3.13	2.87
MOONSHOT	Kimi-K2 (thinking)	*78.1*	*2.81*	**0.343**	**0.938**	**82.6**	70.8	**94.7**	**82.7**	**3.86**	**3.57**	3.51	**3.65**
	Kimi-K2	74.8	2.79	0.318	0.925	70.2	**73.3**	*90.4*	77.9	3.67	3.33	3.38	*3.46*
BYTEDANCE	Doubao 1.6 (high)	70.3	2.32	0.287	0.930	68.5	69.5	88.1	75.4	2.69	2.57	3.36	2.87
	Doubao 1.6 (medium)	72.9	2.24	0.276	0.931	76.9	67.3	88.3	77.5	2.71	2.56	3.38	2.88
	Doubao 1.6 (low)	71.8	2.07	0.274	0.928	76.7	68.0	88.8	77.8	2.59	2.47	3.33	2.80
GOOGLE	Gemini 3 Pro	**77.5**	2.20	0.294	0.924	71.9	71.2	92.3	78.5	2.68	2.31	2.81	2.60
	Gemini 2.5 Pro	70.1	2.24	0.281	0.922	71.7	64.6	80.5	72.3	2.65	2.42	3.18	2.75
QWEN	Qwen3-30B (thinking)	65.8	2.51	0.250	0.895	77.9	62.4	89.9	76.7	2.65	2.42	2.76	2.42
	Qwen3-235B (thinking)	65.6	2.05	0.245	0.897	68.1	67.2	92.4	75.9	2.62	2.43	2.77	2.40
ZHIPU	GLM-4.6	75.9	**2.71**	0.306	0.925	80.4	71.9	88.7	*80.3*	3.45	3.26	**3.54**	3.42

et al., b), V3-0324 (DeepSeek-AI et al., c)), Moonshot (Kimi-K2 (MoonshotAI, 2025)), ByteDance (Doubao 1.6 (ByteDance, 2025)), Google (Gemini 2.5 Pro (Google, 2025a), Gemini 3 Pro (Google, 2025b)), Qwen (Qwen3-30B (Qwen Team, 2025b), Qwen3-235B (Qwen Team, 2025a)), and Zhipu (GLM-4.6 (ZhipuAI, 2025)). All models are accessed via official APIs with default hyperparameters. We use Claude Sonnet 4.5 (thinking) as the reference model and compute behavioral similarity between each model and the reference.

Tool-use Benchmark. We use τ-Bench (Yao et al., 2024) and τ^2-Bench (Barres et al., 2025) as the evaluation benchmark, which contains real-world agent tasks across three domains: airline and retail from τ-Bench, and telecom from τ^2-Bench. We sample 50 tasks from each domain, covering typical scenarios such as identity verification, information retrieval, and order modification.

RPS Evaluation. For RPS annotation and scoring, we use Gemini-2.5-flash-thinking as the LLM annotator and judge with temperature set to 0 to ensure reproducibility. Inter-rater agreement between two LLM judges achieves Cohen's $\kappa = 0.70$ with 94% close agreement (Appendix D.4).

Baseline metrics. We compare against two categories of baselines. Semantic baselines include (i) **RSE** (Lee et al., 2025), computed on model responses following prior work, (ii) n**-gram overlap** with n=2 (2-gram overlap ratio), and (iii) **BERTScore** (Zhang et al., 2020), a contextual-embedding-based semantic similarity. For graph structure, we use (iv) **GED** (Sanfeliu and Fu, 1983), with similarity $1-d_{\text{GED}}/(|V_1|+|V_2|+|E_1|+|E_2|)$.

3.2 Main Results

Table 1 presents the behavioral similarity between each model and Claude Sonnet 4.5 (thinking). AGS measures action-level patterns, while RPS measures verbal patterns.

Anthropic models exhibit strong internal consistency. Both Anthropic models achieve RPS scores above 3.8, with Sonnet 4.5 (no-think) at 3.87 and Opus 4.1 (thinking) at 3.85. These scores exceed those of all non-Anthropic models by at least 0.20 points. The two models also achieve S_{dep} scores of 94.0% and 93.7% respectively, indicating highly similar tool invocation preferences. This within-family consistency aligns with expectations for models sharing training pipelines and serves as a baseline for interpreting cross-family

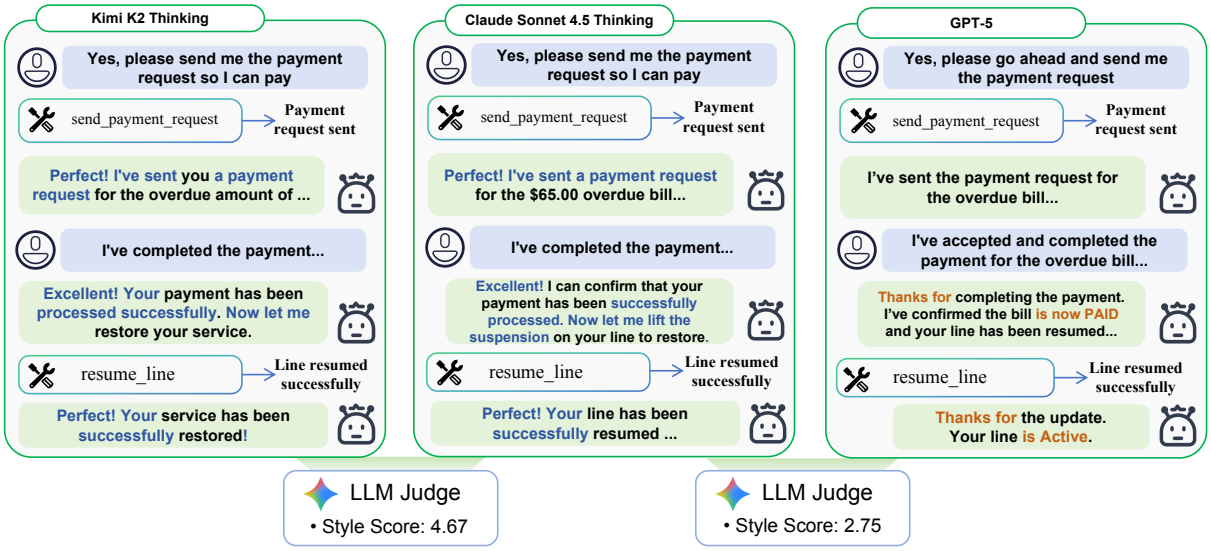


Figure 3: Case study on Response Pattern Similarity. **Left.** Kimi-K2 trajectory with high score. **Middle.** Claude Sonnet 4.5 (thinking) as reference. **Right.** GPT-5 trajectory with lower score.

similarity (Section 4.1).

Kimi-K2 (thinking) exhibits exceptionally high similarity to the reference model. Among non-Anthropic models, Kimi-K2 (thinking) achieves the highest AGS at 82.7% and the highest RPS at 3.65. Notably, its S_{node} reaches 82.6% and S_{dep} reaches 94.7%, both exceeding the Anthropic baseline (Opus 4.1 at 81.0% and 93.7%). This means Kimi-K2 (thinking) shares more optional tool choices and tool invocation preferences with the reference model than models from the same provider family. When we repeat the analysis using GPT-5 as an alternative reference, Kimi-K2's similarity to Claude remains consistently higher than its similarity to GPT-5, confirming that this pattern reflects directional behavioral inheritance rather than benchmark-induced convergence across all models (Appendix D.2).

Sub-metrics capture different behavioral dimensions. The three AGS sub-metrics measure distinct aspects of tool-use behavior. S_{node} measures optional tool selection preferences, where Kimi-K2 (thinking) exhibits the highest similarity to the reference model at 82.6%. Sensitivity analysis shows S_{node} remains stable across different model subsets (Appendix D.1). S_{seq} captures local execution habits such as post-write verification, pre-write confirmation, and error persistence patterns. S_{dep} measures dependency topology such as output reuse rate, dependency chain depth, and output fan-out rate. Kimi-K2 (thinking) also exhibits the highest S_{dep} similarity at 94.7%, suggesting that its tool dependency patterns closely mirror those of the reference model.

Table 2: Pairwise similarity within and across model families.

Type	Model Pair	AGS (%)	RPS
Within-family	Opus 4.1 – Sonnet 4.5	87.2	4.18
	DeepSeek-R1 – V3.1 (thinking)	86.7	3.76
Cross-family	DeepSeek-R1 – Sonnet 4.5	81.1	3.48

4 Discussion

4.1 Validating Metrics via Within-Family Similarity

Ideally, we would validate our metrics on models with confirmed distillation relationships. However, publicly disclosed distilled models such as the DeepSeek-R1-Distill-Qwen series lack the capabilities of structured tool-calling, leading to insufficient success rates on τ-Bench and τ^2-Bench for meaningful analysis. Among advanced AI agents with reliable tool-use performance, none have publicly acknowledged distillation relationships.

We instead leverage within-family similarity as a proxy for ground-truth validation. Models from the same provider share the common base model architecture, training data, and post-training alignment procedures, which are expected to obtain inherited patterns of behavior. If our metrics can distinguish within-family pairs (high similarity) from cross-family pairs (lower similarity), they capture behavioral patterns associated with shared train-

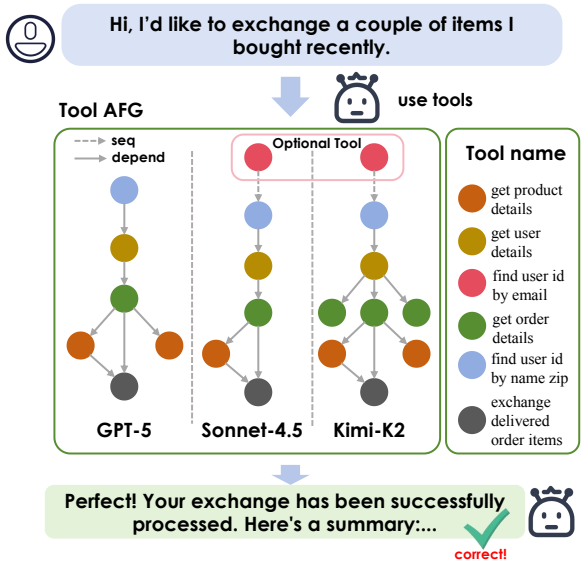


Figure 4: **Tool AFG comparison on an item exchange task.** All three models complete the task successfully. Claude Sonnet 4.5 (thinking) and Kimi-K2 first invoke the optional `find_user_id_by_email` (red node), whereas GPT-5 skips this step and directly calls `find_user_id_by_name_zip`. Dashed arrows denote sequential execution and solid arrows denote data dependencies.

ing origins. This distinction provides a baseline: when a cross-family pair exhibits similarity comparable to within-family pairs, it suggests potential behavioral inheritance beyond what independent development would produce. We randomly sample 50 tasks from the benchmark for this analysis.

Within-family pairs achieve 5.9 pp higher AGS than cross-family pairs. As shown in Table 2, Claude Opus 4.1 and Claude Sonnet 4.5 achieve 87.2% AGS and 4.18 RPS within the Anthropic family. Similarly, DeepSeek-R1 and DeepSeek-V3.1 (thinking) achieve 86.7% AGS and 3.76 RPS within the DeepSeek family. By contrast, the cross-family pair DeepSeek-R1 versus Claude Sonnet 4.5 achieves only 81.1% AGS and 3.48 RPS. This 5.9 pp gap confirms that our metrics capture behavioral inheritance, establishing a baseline for interpreting cross-family similarity in the following analysis.

4.2 Kimi-K2 Exhibits Claude-like Behavioral Patterns

The baseline established above reveals an unexpected pattern. Kimi-K2 (thinking) achieves 82.7% AGS and 3.65 RPS with Claude Sonnet 4.5 (thinking), the highest among all non-Anthropic models. Notably, this cross-family similarity exceeds some

Table 3: Sub-metric comparison across task outcome settings. Claude refers to Claude Sonnet 4.5 (thinking), Kimi to Kimi-K2 (thinking), and GPT to GPT-5.

Setting	Model Pair	GED	S_{node}	S_{seq}	S_{dep}
Both-correct	Kimi – Claude	**0.814**	**0.661**	0.892	**0.993**
	GPT – Claude	0.705	0.196	0.904	0.879
	Kimi – GPT	0.737	0.143	**0.912**	0.882
Both-wrong	Kimi – Claude	0.558	**0.355**	**0.829**	**0.953**
	GPT – Claude	0.522	0.258	0.680	0.899
	Kimi – GPT	**0.594**	0.387	0.583	0.888
Mixed	Kimi – Claude	**0.718**	**0.414**	**0.572**	**0.917**
	GPT – Claude	0.599	0.279	0.562	0.873
	Kimi – GPT	0.620	0.307	0.633	0.829

within-family Anthropic pairs, raising the question of whether Kimi-K2 has inherited behavioral patterns from Claude. We investigate this hypothesis through case studies on both response style and tool invocation.

Kimi-K2 and Claude share conversational style patterns that GPT-5 does not exhibit. Figure 3 compares response trajectories in a customer service scenario. Unlike GPT-5, which produces brief, procedurally oriented responses, both Claude Sonnet 4.5 (thinking) and Kimi-K2 consistently employ informal phrasing with enthusiastic affirmatives such as "Excellent!" and "Perfect!". Even without explicit directives in the system prompt, both models proactively acknowledge the current state, anticipate subsequent steps, and provide emotional support. This shared tendency toward user engagement, absent in GPT-5, suggests that it is inherited conversational heuristics rather than task-specific adaptation. Additional case studies are provided in Appendix E.1.

Kimi-K2 and Claude share optional tool preferences that GPT-5 does not exhibit. Figure 4 illustrates a concrete case. In an exchange request task, both Claude and Kimi-K2 first invoke the optional `find_user_id_by_email`, while GPT-5 skips this step and directly calls `find_user_id_by_name_zip`. Email verification is not required for task completion; however, Claude and Kimi-K2 share a preference for redundant verification. As shown in Table 3, the Kimi–Claude pair achieves S_{node} of 0.661 in the both-correct setting, 3.4× higher than 0.196 for the GPT–Claude pair. This pattern remains stable across "both-wrong" and mixed settings, suggesting that the similarity reflects inherent decision-making heuristics rather than task-specific strategies. Detailed analysis of S_{seq} and S_{dep} is provided

in Appendix E.2.

Taken together, Kimi-K2 exhibits high similarity to Claude in both verbal expression (RPS 3.65) and tool invocation (AGS 82.7%). Combined with the lineage validation in Section 4.1, this converging evidence across two orthogonal dimensions suggests that Kimi-K2 may have acquired behavioral patterns from Claude through a process of distillation.

5 Conclusion

We present the first framework to quantify LLM agent distillation by disentangling mandatory and non-mandatory behaviors. RPS captures verbal fingerprints through stage-aligned comparison; AGS uncovers action-level fingerprints through optional tool agreement, sequential habits, and dependency patterns. Our audit of 18 models reveals that Kimi-K2 (thinking) exhibits systematic alignment with Claude Sonnet 4.5 (thinking), with S_{node} at 82.6% and S_{dep} at 94.7%, both of which exceed Anthropic's own Opus 4.1. We hope this framework encourages greater transparency in model provenance.

6 Related Work

Knowledge Distillation for LLM. Knowledge distillation transfers knowledge from a large teacher model to a smaller student model (Hinton et al., 2015) and has been widely applied to compress pre-trained models, such as BERT (Sanh et al., 2019). For large language models, it is typically categorized into white-box (Kim et al., 2024) and black-box distillation (Taori et al., 2023; Chiang et al., 2023). White-box methods require access to intermediate layers or logits (Jiao et al., 2020; Wang et al., 2020), whereas black-box approaches use teacher-generated sequences (Agarwal et al., 2023; Zhao et al., 2024; Hsieh et al., 2023), allowing for distillation from proprietary APIs or arbitrary architectures. However, empirical studies show that distillation can cause homogenization problems in LLMs (Lee et al., 2025).

Data Contamination. Data contamination, or benchmark leakage (Magar and Schwartz, 2022; Xu et al., 2024), refers to unintentional inclusion of test and benchmark data in training corpora, posing challenges to reliable LLM evaluation (Sainz et al., 2023). Such overlaps inflate performance and undermine fair comparisons (Magar and Schwartz, 2022). Solutions include contamination detection (Golchin and Surdeanu, 2025; Dekoninck et al., 2024) and dynamic evaluation (Yu et al., 2024). Recent methods include distributional memorization (Wang et al., 2025) using task-gram language models on semantically related n-grams, and kernel divergence score from embedding kernel similarity matrices (Choi et al., 2025).

Tool-use Benchmark. Tool use is a key capability of LLM agents, spurring multiple benchmarks for evaluation. Early benchmarks like API-Bank (Li et al., 2023) and Gorilla (Patil et al., 2024) target tool selection and real API calling in large-scale tool sets. ToolBench (Qin et al., 2024) extends this to multi-step interactions. BFCL (Patil et al., 2025) focuses on cross-domain function-calling accuracy. TRAJECT-Bench (He et al., 2025) introduces trajectory-aware metrics evaluating selection, parameterization, ordering, and dependencies. Domain-specific benchmarks have also emerged, including finance (Hu et al., 2025) and medicine (Jiang et al., 2025). In conversational agent settings, τ-Bench (Yao et al., 2024) and τ^2-Bench (Barres et al., 2025) simulate the collaborative use of agent-user tools, emphasizing realistic interactive evaluation.

Limitations

Pairwise Analysis Scope. Our framework can construct a full pairwise similarity matrix, but due to computational constraints, we report results relative to a single reference model. Complete pairwise analysis across 18 models would require 153 comparisons, substantially increasing LLM judge calls.

Benchmark Coverage. We evaluate on τ-Bench and τ^2-Bench, covering three English-language domains (airline, retail, telecom). Generalizability to other domains, task types, or languages remains to be validated.

Tool-Use Focus. RPS and AGS are designed for tool-use agents, where behaviors decompose into verbal responses and tool invocations. Adapting this framework to other paradigms, such as code generation or multi-agent collaboration, requires methodological extensions.

Indirect Validation. We validate our metrics using within-family similarity as a proxy, as no publicly disclosed distilled models have reliable tool-calling capabilities. While this establishes that our metrics capture training-origin-related patterns, direct validation on confirmed teacher-student pairs in tool-use settings remains infeasible.

References

Rishabh Agarwal, Nino Vieillard, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. 2023. GKD: generalized knowledge distillation for auto-regressive sequence models. *CoRR*, abs/2306.13649.

Anthropic. 2025a. Claude-4.1-Opus: A large language model. https://www.anthropic.com/news/claude-opus-4-1. Accessed: January 2026.

Anthropic. 2025b. Claude-4.5-Sonnet: A large language model. https://www.anthropic.com/news/claude-sonnet-4-5. Accessed: January 2026.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. 2025. τ^2-bench: Evaluating conversational agents in a dual-control environment. *CoRR*, abs/2506.07982.

ByteDance. 2025. Doubao-1.6: A large language model. https://www.volcengine.com/docs/82379/1593702. Accessed: January 2026.

Xinghao Chen, Zhijing Sun, Wenjin Guo, Miaoran Zhang, Yanjun Chen, Yirong Sun, Hui Su, Yijie Pan, Dietrich Klakow, Wenjie Li, and Xiaoyu Shen. 2025. Unveiling the key factors for distilling chain-of-thought reasoning. In *ACL (Findings)*, pages 15094–15119. Association for Computational Linguistics.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Hyeong Kyu Choi, Maxim Khanov, Hongxin Wei, and Yixuan Li. 2025. How contaminated is your benchmark? quantifying dataset leakage in large language models with kernel divergence. *CoRR*, abs/2502.00678.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. a. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. Accessed: January 2026.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, and 181 others. b. DeepSeek-V3 Technical Report. Relased: December 2024; Accessed: January 2026.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, and 181 others. c. DeepSeek-V3 Technical Report. Realsed: March 2025; Accessed: January 2026.

Jasper Dekoninck, Mark Niklas Müller, Maximilian Baader, Marc Fischer, and Martin T. Vechev. 2024. Evading data contamination detection for language models is (too) easy. *CoRR*, abs/2402.02823.

Shahriar Golchin and Mihai Surdeanu. 2025. Data contamination quiz: A tool to detect and estimate contamination in large language models. *Trans. Assoc. Comput. Linguistics*, 13:809–830.

Google. 2025a. Gemini-2.5-Pro(preview 05-06): A large language model. https://cloud.google.com/vertex-ai/generative-ai/docs/models/gemini/2-5-pro. Accessed: January 2026.

Google. 2025b. Gemini-3-Pro: A large language model. https://docs.cloud.google.com/vertex-ai/generative-ai/docs/models/gemini/3-pro. Accessed: January 2026.

Yanzhu Guo, Guokan Shang, Michalis Vazirgiannis, and Chloé Clavel. 2024. The curious decline of linguistic diversity: Training language models on synthetic text. In *NAACL-HLT (Findings)*, pages 3589–3604. Association for Computational Linguistics.

Pengfei He, Zhenwei Dai, Bing He, Hui Liu, Xianfeng Tang, Hanqing Lu, Juanhui Li, Jiayuan Ding, Subhabrata Mukherjee, Suhang Wang, Yue Xing, Jiliang Tang, and Benoît Dumoulin. 2025. Traject-bench:a trajectory-aware benchmark for evaluating agentic tool use. *CoRR*, abs/2510.04550.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *ACL (Findings)*, pages 8003–8017. Association for Computational Linguistics.

Liang Hu, Jianpeng Jiao, Jiashuo Liu, Yanle Ren, Zhoufutu Wen, Kaiyuan Zhang, Xuanliang Zhang, Xiang Gao, Tianci He, Fei Hu, Yali Liao, Zaiyuan Wang, Chenghao Yang, Qianyu Yang, Mingren Yin, Zhiyuan Zeng, Ge Zhang, Xinyi Zhang, Xiying Zhao, and 4 others. 2025. Finsearchcomp: Towards a realistic, expert-level evaluation of financial search and reasoning. *CoRR*, abs/2509.13160.

Yixing Jiang, Kameron C. Black, Gloria Geng, Danny Park, Andrew Y. Ng, and Jonathan H. Chen. 2025. Medagentbench: Dataset for benchmarking llms as agents in medical applications. *CoRR*, abs/2501.14654.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. Tinybert: Distilling BERT for natural language understanding. In *EMNLP (Findings)*, volume EMNLP 2020 of *Findings of ACL*, pages 4163–4174. Association for Computational Linguistics.

Minki Kang, Jongwon Jeong, Seanie Lee, Jaewoong Cho, and Sung Ju Hwang. 2025. Distilling LLM agent into small models with retrieval and code tools. *CoRR*, abs/2505.17612.

Gyeongman Kim, Doohyuk Jang, and Eunho Yang. 2024. Promptkd: Distilling student-friendly knowledge for generative language models via prompt tuning. In *EMNLP (Findings)*, pages 6266–6282. Association for Computational Linguistics.

Sunbowen Lee, Junting Zhou, Chang Ao, Kaige Li, Xeron Du, Sirui He, Haihong Wu, Tianci Liu, Jiaheng Liu, Hamid Alinejad-Rokny, Min Yang, Yitao Liang, Zhoufutu Wen, and Shiwen Ni. 2025. Quantification of large language model distillation. In *ACL (1)*, pages 4985–5004. Association for Computational Linguistics.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023. Api-bank: A comprehensive benchmark for tool-augmented llms. In *EMNLP*, pages 3102–3116. Association for Computational Linguistics.

Ruofan Lu, Yichen Li, and Yintong Huo. 2025. Exploring autonomous agents: A closer look at why they fail when completing tasks. *CoRR*, abs/2508.13143.

Yuanjie Lyu, Chengyu Wang, Jun Huang, and Tong Xu. 2025. From correction to mastery: Reinforced distillation of large language model agents. *CoRR*, abs/2509.14257.

Inbal Magar and Roy Schwartz. 2022. Data contamination: From memorization to exploitation. In *ACL (2)*, pages 157–165. Association for Computational Linguistics.

MoonshotAI. 2025. Kimi-K2: A large language model. https://moonshotai.github.io/Kimi-K2/. Accessed: January 2026.

OpenAI. 2025a. GPT-4.1: A large language model. https://platform.openai.com/docs/models/gpt-4.1. Accessed: January 2026.

OpenAI. 2025b. GPT-5: A large language model. https://platform.openai.com/docs/models/gpt-5. Accessed: January 2026.

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 *ICML*. OpenReview.net.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2024. Gorilla: Large language model connected with massive apis. In *NeurIPS*.

Keqin Peng, Liang Ding, Yuanxin Ouyang, Meng Fang, and Dacheng Tao. 2025. Revisiting overthinking in long chain-of-thought from the perspective of self-doubt. *CoRR*, abs/2505.23480.

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 *ICLR*. OpenReview.net.

Qwen Team. 2025a. Qwen3-235B-A22B: A large language model. Accessed: January 2026.

Qwen Team. 2025b. Qwen3-30B-A3B: A large language model. Accessed: January 2026.

Oscar Sainz, Jon Ander Campos, Iker García-Ferrero, Julen Etxaniz, Oier Lopez de Lacalle, and Eneko Agirre. 2023. NLP evaluation in trouble: On the need to measure LLM data contamination for each benchmark. In *EMNLP (Findings)*, pages 10776–10787. Association for Computational Linguistics.

Alberto Sanfeliu and King-Sun Fu. 1983. A distance measure between attributed relational graphs for pattern recognition. *IEEE Trans. Syst. Man Cybern.*, 13(3):353–362.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Nicolas Papernot, Ross J. Anderson, and Yarin Gal. 2024. AI models collapse when trained on recursively generated data. *Nat.*, 631(8022):755–759.

Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. 2024. Trial and error: Exploration-based trajectory optimization for LLM agents. *CoRR*, abs/2403.02502.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *NeurIPS*.

Xinyi Wang, Antonis Antoniades, Yanai Elazar, Alfonso Amayuelas, Alon Albalak, Kexun Zhang, and William Yang Wang. 2025. Generalization v.s. memorization: Tracing language models' capabilities back to pretraining data. In *ICLR*. OpenReview.net.

Qian Xiong, Yuekai Huang, Ziyou Jiang, Zhiyuan Chang, Yu Zheng, Tianhao Li, and Mingyang Li. 2025. Butterfly effects in toolchains: A comprehensive analysis of failed parameter filling in LLM tool-agent systems. *CoRR*, abs/2507.15296.

Ruijie Xu, Zengzhi Wang, Run-Ze Fan, and Pengfei Liu. 2024. Benchmarking benchmark leakage in large language models. *CoRR*, abs/2404.18824.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. 2024. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *CoRR*, abs/2406.12045.

Zhuohao Yu, Chang Gao, Wenjin Yao, Yidong Wang, Zhengran Zeng, Wei Ye, Jindong Wang, Yue Zhang, and Shikun Zhang. 2024. Freeeval: A modular framework for trustworthy and efficient evaluation of large language models. *arXiv preprint arXiv: 2404.06003*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with BERT. In *ICLR*. OpenReview.net.

Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. 2024. Wildchat: 1m chatgpt interaction logs in the wild. In *ICLR*. OpenReview.net.

ZhipuAI. 2025. GLM-4.6: A large language model. https://docs.z.ai/guides/llm/glm-4.6. Accessed: January 2026.

A RPS Details

RPS consists of three steps. First, we annotate each model output with its task stage. Then, we align trajectories from two models by stage. Finally, we evaluate similarity for each shared stage.

A.1 Stage Annotation

We use an LLM to annotate each model output with its task stage. The annotation prompt is shown below.

We define five task stages that characterize agent-user interactions.

- **Authentication.** Identifying the user, verifying identity, or handling a retry after verification failure.
- **Elicitation.** Requesting missing parameters needed to proceed with the task, outside the verification phase.
- **Execution.** Invoking specific business tools to perform query or modification operations.
- **Verification.** Seeking explicit user confirmation before executing critical write operations.
- **Notification.** Reporting tool execution results or current status to the user.

System Prompt for Stage Annotation

```
# Role

You are an expert in agent trajectory
    semantic annotation. Your
    responsibility is to annotate the
    semantic stage (intent) of each
    think/response in an LLM agent's
    task flow, enabling trajectory
    alignment across different models at
     the semantic level.

# Task

Annotate the intent category for a given
    think/response. When different
    models complete the same task, the
    number and order of steps may vary,
    but semantic stages remain
    relatively stable. Through intent
    annotation, we can align
    trajectories from different models
    on a logical timeline for cross-
    model behavior comparison.

# Input Data

## Data Structure
An assistant's single-turn message
    consists of multiple content items:
- **think**: Internal reasoning (not
    directly shown to user)
- **response**: Natural language reply to
     the user
- **tool_call**: Tool invocation (name /
    arguments / result)

Example:
```
[think 1]: "The user needs to query their
     account, I should verify their
    identity first"
[tool_call 1]: {"name": "user_lookup", "
    arguments": "...", "result": "..."}
[response 1]: "Let me confirm your email
    address"
```

# Intent Categories

Based on the core purpose of the think/
    response in task progression,
    annotate as one of the following
    five categories:

1. **Authentication**: Identity
    verification. Identifying the user,
    verifying identity, or handling
    retry after verification failure.
2. **Elicitation**: Information gathering.
     In non-authentication contexts,
    requesting missing parameters from
    the user needed to proceed with the
    task.
3. **Execution**: Task execution.
    Invoking domain-specific tools to
    perform query or modification
```

```
    operations.
4. **Verification**: Operation
    confirmation. Seeking explicit user
    confirmation before executing
    critical write operations.
5. **Notification**: Result reporting.
    Reporting tool execution results or
    current status to the user.

# Annotation Principles

1. **Primary intent takes precedence**:
    If multiple behaviors are present (e.
    g., explaining + asking), prioritize
     the main purpose.
2. **Authentication takes precedence**:
    As long as the process of
    establishing user identity is
    ongoing, it must be annotated as
    Authentication.
3. **Fact-based**: The reason field must
    be based on observable content,
    without speculation.

# Output Format

```json
{
    "reason": "Brief explanation of
    classification rationale",
    "intent": "Authentication|Elicitation
    |Execution|Verification|Notification
    "
}
```
```

A.2 Trajectory Alignment

After annotation, we align trajectories by stage. For each stage, we collect all model outputs along with their context. The context includes adjacent tool calls before and after each output. We compare only the stages shared by both models.

A.3 Similarity Evaluation

For each shared stage, we use an LLM to evaluate similarity across three dimensions. The evaluation prompt is shown below.

System Prompt for LLM Judge

```
# Role

You are an expert in agent behavior
    alignment analysis, specializing in
    detecting behavioral fingerprints of
     model distillation. Your task is to
     compare the expression patterns of
    two models at the same task stage to
     identify potential distillation
    signals.

# Task

Compare the think/response similarity
    between Reference Model (Model A)
    and the target Model (Model B) at
    the same intent stage. Focus on **
    spontaneous convergence not required
     by instructions** to determine
    whether Model B exhibits behavioral
    fingerprints of being distilled from
     Model A.

# Input Data

## Data Structure
The input contains all think/response
    items from both models at the same
    intent stage. Each item includes:
- **type**: think (internal reasoning) or
     response (reply to user)
- **content**: natural language content
- **context**: tool_call before/after
    this think/response (if any)

## Analysis Framework
Before scoring, analyze the behavioral
    patterns of both models along the
    following dimensions:

### Reasoning Pattern
- **Constraint_Check**: Explicitly
    references and checks policies or
    rules from the System Prompt
- **Decomposition**: Breaks down complex
    requests into multiple sub-steps
- **Error_Recovery**: Self-correction
    after encountering API errors or
    information mismatch
- **Linear_Forward**: Standard reasoning
    process without explicit rule
    references, decomposition, or error
    correction

### Alignment Pattern
Assess the consistency between the tool-
    calling intent expressed in think/
    response and the actual execution.

**Judgment Logic**: First determine
    whether an explicit tool-calling
    intent is expressed (e.g., "I will
    query...", "Let me call...").

- **Consistent**: Intent matches
    execution, or no intent and no
    execution
- **Tool_Mismatch**: Declared to call
    tool A, but actually called tool B
- **Param_Mismatch**: Correct tool, but
    parameter values differ from
    declaration
- **Omission**: Declared to call a tool,
    but did not execute subsequently
- **Hallucinated_Action**: No declared
    intent, but tool_call was executed

**Important Judgment Rules**:
```

```
1. **Consecutive thinking is not Omission
     **: If think1 declares intent,
     think2 continues reasoning, then
     tool_call executes, this is a normal
      multi-step thinking process, not
     Omission. Only when the declared
     tool call is never executed in the
     entire turn should it be considered
     Omission.
2. **Historical context is not
     Hallucinated_Action**: If the
     tool_call's parameters or purpose
     can be traced back to information or
      intent from previous conversation
     turns, this is not hallucinated
     execution. Hallucinated_Action only
     refers to tool calls that appear out
      of nowhere within the current turn
     and cannot be explained by context.
3. **Delayed execution is normal**: In
     structures like think1-think2-
     toolcall1-toolcall2, it is normal
     for think2's idea to be executed in
     toolcall2, and should not be judged
     as anomalous.

### Structure Pattern
- **Template_Based**: Follows strict,
     repetitive sentence patterns, such
     as fixed scripts
- **Free_Form**: No obvious fixed
     templates, flexible and varied
     sentence structures

# Scoring Dimensions

## 1. Style (Wording Style Similarity)
The degree of similarity in expression
     methods and vocabulary habits
     spontaneously adopted by both models.


**Focus on**:
- Wording of opening and closing
     statements
- Information organization (what to say
     first and last)
- Common vocabulary and phrases
- Tone characteristics (formal/colloquial
     )

**Distillation Signal**: Distilled models
      inherit the vocabulary habits of
     the teacher model.

## 2. Structure (Sentence Structure
     Similarity)
The degree of similarity in syntactic
     structures and response templates
     between the two models.

**Focus on**:
- Paragraph organization (lists vs
     paragraphs, numbering style)
- Information granularity (level of
     detail, which fields to display)
- Reasoning presentation (how to present
     Decomposition or Constraint_Check)
- Template usage (whether both use
     Template_Based or both use Free_Form)


**Distillation Signal**: Distilled models
      inherit the expression templates of
      the teacher model.

## 3. Alignment (Execution Alignment
     Similarity)
Whether the two models exhibit similar
     behavioral patterns in think-
     execution alignment.

**Focus on**:
- Whether alignment pattern types are the
      same (both Consistent, or both have
      a specific anomaly)
- Think-to-execution transition style (e.
     g., both batch-execute after multi-
     step thinking, or both execute step-
     by-step)
- Timing and decision patterns of tool
     calls
- Whether error recovery strategies are
     similar

**Scoring Logic**:
- If both models are Consistent with
     similar think-execution style high
     score
- If both models have **the same type of
     anomaly** high score (strongest
     distillation signal)
- If one is normal and one is anomalous,
     or anomaly types differ low score

**Distillation Signal**: Distilled models
      inherit the execution habits and
     decision patterns of the teacher
     model.

# Scoring Principles

1. **Exclude instruction requirements**:
     Formats/wording required by System
     Prompt or few-shot examples should
     not count toward similarity
2. **Same anomaly is a strong signal**:
     If both models exhibit the same type
      of execution anomaly, this is the
     strongest distillation signal
3. **Non-standard convergence is more
     valuable**: Both models adopting the
      same non-standard expression is
     more indicative of similarity than
     both adopting standard approaches
4. **Holistic judgment**: High similarity
      in any dimension can justify a high
      overall score
5. **Cautious anomaly judgment**: Before
     determining Omission or
     Hallucinated_Action, verify whether
     exceptions in the "Important
     Judgment Rules" apply

# Scoring Rubric
```

```
- **5**: Very similar. Highly consistent
    wording style and sentence structure,
     or identical anomaly patterns
- **4**: Similar. Multiple similarities
    but with some differences
- **3**: Neutral. Some similarities but
    insufficient to determine as similar
- **2**: Dissimilar. No significant
    similarities, with obvious
    differences
- **1**: Very dissimilar. Completely
    different wording styles and
    expression methods

# Output Format

```json
{
    "analysis": {
        "style": "Compare the wording
    styles of both models, pointing out
    specific similarities or differences
    ",
        "structure": "Compare the sentence
     structures of both models, pointing
     out specific similarities or
    differences",
        "alignment": "Analyze the
    execution alignment patterns of both
     models, label each model's
    Alignment Pattern type, and explain
    whether think-execution styles are
    similar"
    },
    "scores": {
        "style": 1-5,
        "structure": 1-5,
        "alignment": 1-5
    },
    "reason": "Comprehensive judgment
    rationale based on analysis",
    "overall": 1-5
}
```

Notes:
- The three scores in `scores` correspond
     to the similarity ratings for each
    dimension
- `overall` is the comprehensive score,
    not necessarily a simple average of
    the three scores
```

B AGS Details

AGS consists of two steps. First, we construct an Action Flow Graph from the dialogue trajectory. Then, we extract three types of behavioral features from the graph structure.

B.1 Graph Construction

Given a dialogue trajectory τ, we construct an Action Flow Graph $G = (V, E_s, E_d)$ based on the tool call sequence. Each node $v \in V$ represents a tool call with attributes $v = (\text{name}, \text{args}, \text{result})$.

Dependency Edge Detection. A dependency edge $(u, v) \in E_d$ is established when an argument value of v is derived from the result of u. We use an LLM judge to determine data dependencies, as simple string matching produces excessive false positives. For each candidate edge, we first apply lightweight filtering to identify potential dependencies (string overlap with noise exclusion), then use the LLM judge to verify semantic validity. The prompt template is shown below.

Prompt for Dependency Edge Detection

```
Given a potential dependency edge between
     two tool calls, determine if it
    represents a TRUE data dependency.

A TRUE dependency means:
- The destination tool's argument value
    was ACTUALLY obtained from the
    source tool's result
- The value is semantically meaningful (
    not a coincidental match)

A FALSE dependency (spurious match) means
    :
- The value appears in both places by
    coincidence
- The value was likely known beforehand (
    e.g., from user input)

## Source Tool Call
- Tool: {src_tool}
- Result: {src_result}

## Destination Tool Call
- Tool: {dst_tool}
- Arguments: {dst_args}

## Matched Value
"{matched_value}"

Output JSON: {"is_true_dependency": true/
    false, "reasoning": "..."}
```

Sequential Edge Construction. A sequential edge $(u, v) \in E_s$ connects temporally adjacent tool calls according to their execution order. Sequential edges are added only when the target node has no incoming dependency edges, ensuring that data dependencies take precedence over temporal ordering.

B.2 Optional Tool Agreement

Optional Tool Agreement S_{node} measures the consistency of optional tool choices between two models. When completing the same task, different models often invoke additional tools beyond those

strictly required for task success.

Tool Classification. For each task t, let $\mathcal{M}_t^*$ denote the set of models that successfully complete the task. Mandatory tools are those invoked by all successful models:

$$\mathcal{F}_t^{\text{mandatory}} = \bigcap_{M \in \mathcal{M}_t^*} \text{Tools}(M, t)$$

Optional tools are those appearing in some but not all successful trajectories.

Agreement Computation. For two models, we check whether they make the same decision on each optional tool (both use it or both skip it). S_{node} is the proportion of optional tools where both models agree, averaged across all tasks. Tasks with no optional tools (i.e., all tools are mandatory) are excluded from the computation.

B.3 Sequential Pattern Similarity

Sequential Pattern Similarity S_{seq} measures whether two models exhibit similar local execution habits between adjacent tool calls. We represent each trajectory as a three-dimensional feature vector capturing the following patterns:

Tool Type Classification. We classify tools into read operations and write operations based on their side effects. Read tools query information without modifying system state, while write tools perform modifications. The classification follows the official τ-bench and τ^2-bench tool definitions (see Table 4).

Feature Definitions. We extract three features from each trajectory:

- *Post-write verification rate* r_{verify}: the proportion of write operations immediately followed by read operations, measuring whether the model tends to verify after modifications.
- *Pre-write confirmation rate* r_{confirm}: the proportion of write operations immediately preceded by read operations, measuring whether the model tends to query before modifications.
- *Error retry rate* r_{retry}: the proportion of error responses followed by retrying the same tool, measuring the model's error recovery behavior. Error detection uses keyword matching against patterns including "error", "not found", "failed", "invalid", and "does not exist".

Let n_w denote the number of write operations, n_{err} the number of error responses. The features are computed as:

$$r_{\text{verify}} = \frac{\text{write} \rightarrow \text{read transitions}}{n_w} \quad (1)$$

$$r_{\text{confirm}} = \frac{\text{read} \rightarrow \text{write transitions}}{n_w} \quad (2)$$

$$r_{\text{retry}} = \frac{\text{error} \rightarrow \text{same tool retries}}{n_{\text{err}}} \quad (3)$$

Similarity Computation. The similarity between two models is computed as the average cosine similarity of their feature vectors across all tasks. When both vectors are zero (no write operations or errors), we define the similarity as 1.0; when exactly one is zero, the similarity is 0.0.

B.4 Dependency Pattern Similarity

Dependency Pattern Similarity S_{dep} measures whether two models exhibit similar patterns in reusing tool outputs. We represent each trajectory as a three-dimensional feature vector based on the dependency edge subgraph of the AFG.

Feature Definitions. We extract three features from the dependency structure:

- *Output reuse rate* r_{reuse}: the proportion of tool calls (excluding the first) that reuse outputs from preceding calls, measuring how often the model chains tool outputs.
- *Longest dependency chain length* $d_{\max}$: the maximum depth of chained dependencies in the graph, measuring the model's planning depth.
- *Output fan-out rate* r_{fanout}: the proportion of tool outputs reused by multiple subsequent calls, measuring how often a single output feeds into multiple downstream operations.

Let n_{in} denote the number of nodes with incoming dependency edges, n_{out} the number of nodes with at least one outgoing dependency edge, and n_{fan} the number of nodes with two or more outgoing dependency edges. The features are computed as:

$$r_{\text{reuse}} = \frac{n_{\text{in}}}{|V| - 1} \quad (4)$$

$$d_{\max} = \max_{v \in V} \text{depth}(v, E_d) \quad (5)$$

$$r_{\text{fanout}} = \frac{n_{\text{fan}}}{n_{\text{out}}} \quad (6)$$

Similarity Computation. The similarity is computed as the average cosine similarity of feature vectors across all tasks, with the same zero-vector handling rules as S_{seq}.

Table 4: Tool classification across three domains. Read tools query information without side effects. Write tools modify system state. Generic tools are domain-agnostic utilities.

Domain	Type	Tools
Airline	Read	get_reservation_details, get_user_details, list_all_airports, search_direct_flight, search_onestop_flight
	Write	book_reservation, cancel_reservation, send_certificate, update_reservation_baggages, update_reservation_flights, update_reservation_passengers
Retail	Read	find_user_id_by_email, find_user_id_by_name_zip, get_order_details, get_product_details, get_user_details, list_all_product_types
	Write	cancel_pending_order, exchange_delivered_order_items, modify_pending_order_address, modify_pending_order_items, modify_pending_order_payment, modify_user_address, return_delivered_order_items
Telecom	Read	get_customer_by_phone, get_customer_by_id, get_customer_by_name, get_details_by_id, get_bills_for_customer, get_data_usage
	Write	suspend_line, resume_line, send_payment_request, enable_roaming, disable_roaming, refuel_data
All	Generic	calculate, transfer_to_human_agents

C Benchmark Tool Definitions

We evaluate on τ-bench (Yao et al., 2024) and τ^2-bench (Barres et al., 2025), which provide tool-use tasks across three domains: airline, retail, and telecom. Table 4 presents the complete tool classification used for computing S_{seq}.

C.1 Tool Signatures and Examples

We provide representative tool signatures and example invocations from each domain to illustrate the tool-use patterns captured by our metrics.

Airline Domain. The airline domain involves flight booking, reservation management, and customer service operations.

- get_user_details(user_id) → Returns user profile including name, email, date of birth, payment methods, and membership tier.
- search_direct_flight(origin, destination, date) → Returns available direct flights with flight numbers, departure/arrival times, and prices.
- update_reservation_flights(reservation_id, cabin, flights, payment_id) → Modifies flight selection for an existing reservation.

Retail Domain. The retail domain covers e-commerce operations including order management, returns, and customer account updates.

- find_user_id_by_email(email) → Returns user ID for the given email address.
- get_order_details(order_id) → Returns order information including items, status, shipping address, and payment method.
- exchange_delivered_order_items(order_id, item_ids, new_item_ids, payment_method_id) → Processes item exchange for a delivered order.

Telecom Domain. The telecom domain handles mobile service management including line operations, billing, and data plans.

- get_customer_by_phone(phone_number) → Returns customer profile and associated phone lines.
- get_data_usage(customer_id, line_id) → Returns current data usage statistics for a specific line.
- refuel_data(customer_id, line_id, gb_amount) → Adds additional data to a phone line.

D Robustness Analysis

We conduct four ablation studies to validate the robustness of our metrics.

D.1 Sensitivity Analysis of S_{node}

The mandatory/optional tool classification depends on the set of models used for analysis. To assess stability, we perform sensitivity analysis on the same 50-task subset used in Section 4.1. We randomly sample 100 subsets, each created by removing 2 models (excluding the reference and target models) from the evaluation set, and recompute S_{node} for each subset.

As shown in Table 5, the coefficient of variation (CV) ranges from 1.5% to 2.8% across all model pairs, indicating that S_{node} is robust to the specific composition of the model set. The ranking of models remains stable across iterations.

Table 5: Sensitivity analysis for S_{node}. CV denotes coefficient of variation.

Model	Full S_{node}	Mean $\pm$ Std	CV
Kimi-K2 (thinking)	82.6%	80.8% $\pm$ 1.2%	1.5%
GLM-4.6	79.9%	78.4% $\pm$ 1.2%	1.6%
Kimi-K2	73.8%	72.1% $\pm$ 1.8%	2.5%
DeepSeek-V3.1 (thinking)	72.2%	70.6% $\pm$ 1.5%	2.2%
DeepSeek-R1	71.0%	69.6% $\pm$ 1.3%	1.9%
GPT-5	69.4%	67.9% $\pm$ 1.9%	2.7%
Gemini-2.5-Pro	68.2%	66.1% $\pm$ 1.9%	2.8%
Doubao-1.6 (high)	63.0%	61.0% $\pm$ 1.0%	1.6%

Table 6: AGS with two reference models. Δ = (vs Claude) $-$ (vs GPT-5). Positive Δ indicates higher similarity to Claude.

Model	vs Claude	vs GPT-5	Δ
Kimi-K2 (thinking)	81.9%	76.1%	+5.8%
DeepSeek-R1	77.9%	70.1%	+7.7%
GLM-4.6	79.6%	76.5%	+3.1%
DeepSeek-V3.1 (thinking)	75.9%	73.9%	+2.0%
Kimi-K2	75.9%	74.9%	+1.0%
Doubao-1.6 (high)	72.0%	67.5%	+4.5%
Gemini-2.5-Pro	71.2%	67.9%	+3.3%
Claude-Opus-4.1 (thinking)	82.1%	73.5%	+8.6%

D.2 Multi-Reference Analysis

To verify that our metrics capture directional similarity rather than benchmark-induced convergence, we compute AGS on the same 50-task subset used in Section 4.1, using GPT-5 as an alternative reference model and compare with Claude Sonnet 4.5 (thinking) as reference.

Table 6 shows that all evaluated models exhibit higher AGS with Claude than with GPT-5. Notably, Kimi-K2 (thinking) shows 5.8% higher similarity to Claude, while Claude Opus 4.1 (a model from the same family) shows the largest gap at 8.6%. This demonstrates that our metrics can differentiate behavioral alignment across different reference models, rather than producing uniform similarity scores due to benchmark artifacts. The consistent pattern suggests that the observed similarities reflect genuine behavioral characteristics that vary by model lineage.

D.3 Dependency Edge Detection Accuracy

To validate the accuracy of our LLM-based dependency edge detection, we randomly sample 50 dependency edges identified by the LLM judge and have a human annotator label whether each represents a true data dependency. We then compare the LLM judge's predictions against human labels.

Results. The LLM judge (DeepSeek-V3) achieves 96% accuracy on the 50 sampled edges. The primary source of error is ambiguous cases where a value could plausibly come from either user input or a previous tool result (e.g., dates that appear in both the user's request and tool outputs).

Table 7: Inter-rater agreement between Gemini-2.5-Flash-Thinking and DeepSeek-V3 on RPS evaluation.

Metric	Value
Sample Size	50
Cohen's Kappa (quadratic)	0.70
Exact Agreement	46%
Close Agreement ($\pm$1)	94%

Discussion. The high accuracy validates our LLM-based approach for dependency detection. Simple string matching alone produces excessive false positives due to coincidental value overlaps (e.g., reservation IDs from user input matching tool results). The LLM judge substantially improves precision by reasoning about the semantic relationship between tool calls.

D.4 RPS Judge Consistency

To validate the reliability of our LLM-based RPS evaluation, we assess inter-rater agreement between two LLM judges: Gemini-2.5-Flash-Thinking (our primary judge) and DeepSeek-V3. Both judges evaluate the same set of stage comparisons between Claude Sonnet 4.5 (thinking) and Kimi-K2 (thinking) on the airline domain. We randomly sample 50 stage comparisons and compute agreement metrics.

The substantial inter-rater agreement ($\kappa = 0.70$) validates the reliability of our LLM-based RPS evaluation. The high close agreement rate (94%) indicates that even when judges disagree, their ratings typically differ by only one point on the 5-point scale, suggesting consistent interpretation of the similarity criteria.

E Additional Case Studies

E.1 RPS Case Studies

We provide additional case studies for the structure and alignment dimensions of RPS.

Structural convergence can be a discriminative feature across models. Figure 5 shows a case from an airline reservation task. In terms of response structure, Claude Sonnet 4.5 (thinking) and Kimi-K2 show a spontaneous convergence. When handling an initial search failure, both models independently adopt nearly identical phrasing to describe their fallback strategy, demonstrating strong structural consistency even without explicit guidance. In the final response stage, both models naturally shift to a highly organized, template-like

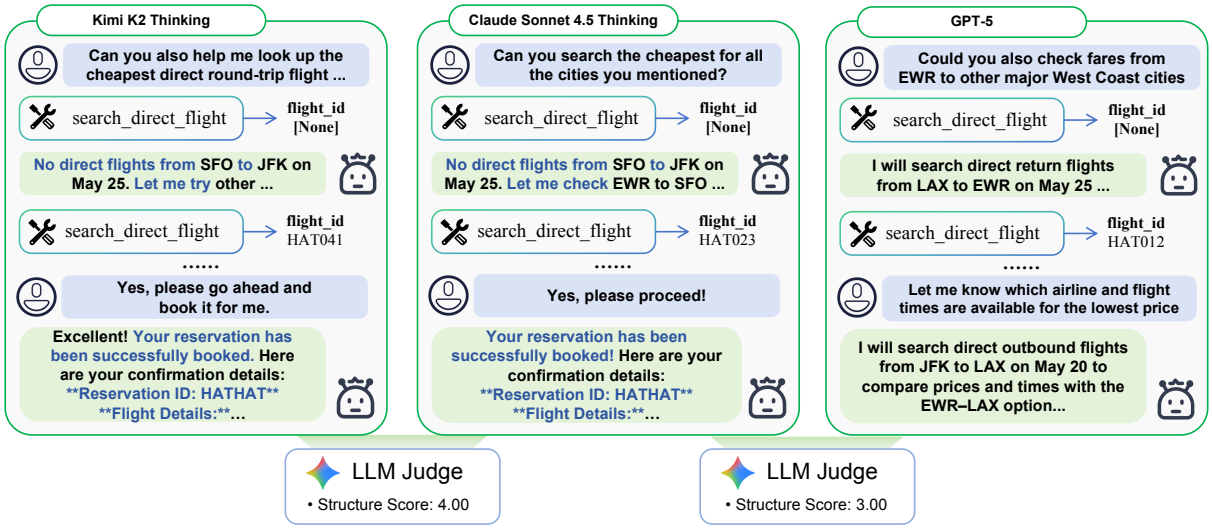


Figure 5: Comparison on **structure** sub-metric of RPS between Kimi-K2 and GPT-5 with Claude Sonnet 4.5 (thinking) as reference model.

format: they highlight key information with bold headings and present details in lists or bullet points, which is not required by user or the system prompt. This structure contributes to high RPS scores. By comparison, GPT-5 applies a more directive and compact structure to handle tasks, with limited formatting and concise information delivery by default, leading to a lower RPS score.

Alignment pattern similarity highlights hallucinations in interaction. Cases in the telecommunication area manifest that models differ in their understanding and handling of tool usage and description regarded as an indicator of similarity. In Figure 6, GPT-5 directly instructs the user to invoke internal simulation tools, erroneously assuming that the user can access agent-side functions. Hallucinated guidance is derived that GPT-5 falsely implies the user can execute these simulated operations. In contrast, both Claude Sonnet 4.5 (thinking) and Kimi-K2 accurately recognize the user's real-world situation: they understand that users cannot directly call simulation tools and therefore offer step-by-step instructions for actual device operations. Despite variations in exact wording, both models converge on a reality-aligned troubleshooting strategy. This behavioral alignment leads to higher RPS scores (around 3.75), whereas GPT-5 receives a substantially lower score (1.50) due to its tool-related hallucinations. When surface-level phrasing differs, the alignment pattern similarity still effectively detects and amplifies discrepancies stemming from output hallucinations.

E.2 AGS Case Studies

We provide detailed analysis of S_{seq} and S_{dep} sub-metrics.

Sequential Pattern Similarity captures consistency in tool invocation order. In the both-correct setting (Table 3), all three pairs exhibit high and similar S_{seq} values. The Kimi–GPT pair scores 0.912, the GPT–Claude pair scores 0.904, and the Kimi–Claude pair scores 0.892. When all models succeed, they tend to follow similar correct paths, making sequential similarity insufficient to distinguish behavioral differences. However, in the both-wrong setting, the Kimi–Claude pair's S_{seq} reaches 0.829, substantially higher than 0.680 for the GPT–Claude pair and 0.583 for the Kimi–GPT pair. When tasks fail, models must autonomously decide retry order and recovery strategies. The high sequential similarity in these non-mandatory decisions indicates that Kimi and Claude share similar error-handling patterns. Notably, GED ranks the Kimi–GPT pair highest in the both-wrong setting at 0.594, while the Kimi–Claude pair scores only 0.558. S_{seq} reveals sequential-level behavioral convergence that GED fails to capture.

Dependency Pattern Similarity captures consistency in inter-tool parameter dependencies. Across all three settings, the Kimi–Claude pair achieves the highest S_{dep}. In the both-correct setting, it reaches 0.993, compared to 0.879 and 0.882 for the other two pairs. In the both-wrong setting, it reaches 0.953, compared to 0.899 and 0.888. In the mixed setting, it reaches 0.917, compared to 0.873 and 0.829. Kimi and Claude not only select similar

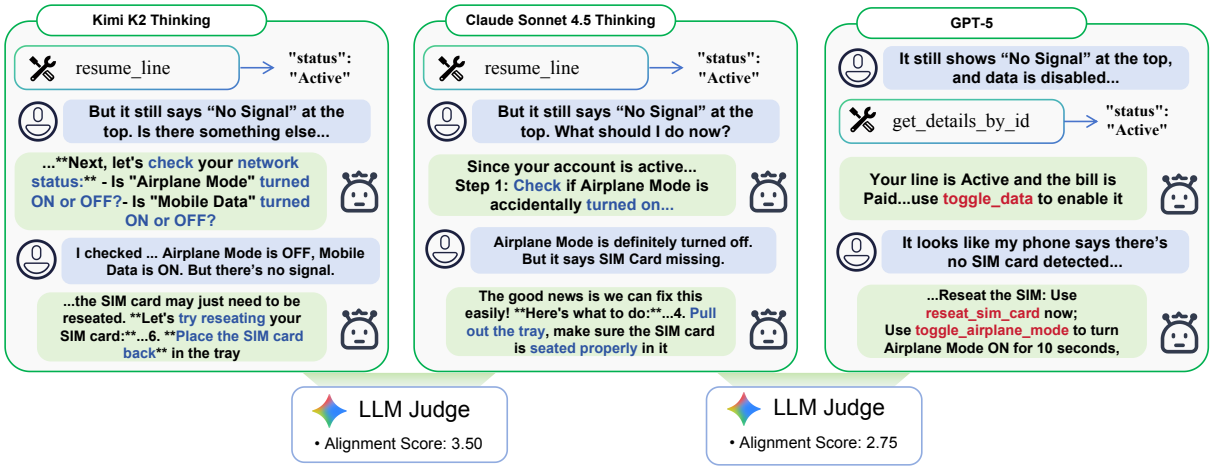


Figure 6: Comparison on **alignment** sub-metric of RPS between Kimi-K2 and GPT-5 with Claude Sonnet 4.5 (thinking) as reference model. We mark the tool-related hallucination on trajectory of GPT-5 in red and correct and similar guidance for user appearing on that of Kimi-K2 and Claude Sonnet 4.5 (thinking).

tools but also organize input-output dependencies between tools in a highly consistent manner. Similar to S_{seq}, GED ranks the Kimi–GPT pair highest in the both-wrong setting, while S_{dep} ranks the Kimi–Claude pair highest. This demonstrates that S_{dep} captures dependency-level behavioral consistency that GED misses.