

One Instruction Is a Benchmark: End-to-End Instruction-Following Evaluation with FlexBench

Yuchen Ge¹, Yizhao Sun¹, Wei He¹, Jinpeng Ou¹, Wenzhe Niu¹,
Chengshun Shi¹, Huichuan Fan¹, Biao Ma¹, Shuo Huang²,
Xin Tian³, Jiuchong Gao¹, Jinghua Hao¹, Renqing He¹
¹Meituan ²University of Southern California ³Zhejiang University
{geyuchen, hewei19}@meituan.com

Abstract

Real-world deployments of large language models (LLMs) increasingly rely on user-provided instructions that are long, precise, and tailored to specific workflows. In such settings, aggregate leaderboard performance is often a weak proxy for a model’s ability to faithfully execute the particular instruction that a practitioner intends to deploy. To address this gap, we introduce **FlexBench/FlexEval**, an instruction-conditioned evaluation framework that instantiates a specialized test suite from a user-provided target instruction. Given a target instruction, FlexBench (i) derives a set of verifiable evaluation dimensions and (ii) generates an instruction-conditioned conversation corpus. FlexEval then aggregates dimension-level judgments into two complementary metrics: **COVERAGE**, which measures workflow progress, and **ACHIEVEMENT**, which measures conditional correctness. We evaluate the framework on nine multi-turn target instructions spanning diverse scenario categories and workflow topologies, across nine contemporary LLMs over 3,888 generated dialogues. The results show that model rankings and failure modes vary substantially across target instructions, highlighting the need for instruction-conditioned evaluation beyond fixed aggregate benchmarks. Our source code is publicly available at <https://github.com/Coryyy/FlexBench>.

1 Introduction

As LLMs are increasingly deployed across diverse scenarios, their behavior is often governed by concrete target instructions, such as standard operating procedures, state-machine-like policies, and multi-step workflows. Recent evidence suggests that such workflows become especially challenging when procedures involve conditional control flow and interdependent steps (Nandi et al., 2025). For such applications, aggregate leaderboard performance is often a weak proxy for whether a model

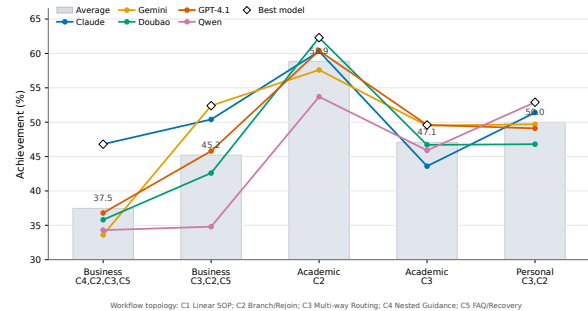


Figure 1: Achievement across representative target instructions spanning different scenario categories and workflow topologies. Bars denote average scores across models, lines show model-specific scores, and diamonds mark the best-performing model for each instruction. The leading model changes across instructions.

can faithfully execute the particular instruction that a practitioner intends to deploy. In practice, the central question is not only “which model is best overall?”, but more concretely, “which model best follows this target instruction?”

Figure 1 illustrates this gap. We compare representative LLMs on several concrete target instructions instantiated from different application categories and workflow topologies. The average difficulty differs across instructions, but more importantly, the relative strengths of models are not uniform: the best-performing model changes from one target instruction to another. This pattern indicates that model suitability is *instruction-conditioned*; a single global ranking cannot reliably identify the best model for a specific deployment instruction.

Existing instruction-following benchmarks provide valuable global signals for model comparison, including increasingly fine-grained evaluation of constraints, compositional requirements, and structured interaction settings (Zhou et al., 2023; Qin et al., 2024; Jiang et al., 2024; Wen et al., 2024; Zhang et al., 2025; Li et al., 2025; Qi et al., 2025). Recent work has also moved beyond static

evaluation sets by dynamically constructing benchmarks (Shashidhar et al., 2025; Zhou et al., 2026; Prem et al., 2026). However, these efforts typically generate evaluation items from documents, interactive environments, or model-generated questions, and are not designed to evaluate whether a model can faithfully execute a concrete target instruction that a practitioner intends to deploy. They are therefore not designed to answer a complementary, deployment-oriented question: given a new target instruction, how can we automatically construct a faithful, fine-grained, and reproducible benchmark tailored to that instruction itself?

We address this problem with **FlexBench** and **FlexEval**, an instruction-conditioned evaluation framework that instantiates a specialized benchmark from a user-provided target instruction. Given a target instruction I , FlexBench automatically derives a set of verifiable evaluation dimensions $D(I)$ and synthesizes a dialogue corpus $C(I)$ that elicits the behaviors required by that instruction. The resulting test suite is therefore specialized to I rather than selected from a fixed task bank. FlexEval evaluates model traces against these dimensions and aggregates the results into two complementary metrics: **Coverage**, which measures whether required workflow steps are exercised, and **Achievement**, which measures whether conditionally applicable requirements are satisfied once those steps are exercised. By using preconditions and mutually exclusive branches during aggregation, FlexEval separates omission failures from execution failures: a model may fail to progress through the workflow, or it may attempt the workflow but violate applicable conditional requirements.

We evaluate FlexBench/FlexEval on nine multi-turn target instructions organized along two complementary axes: *scenario category* and *workflow topology*. We further validate FlexBench against human-curated decompositions from established instruction-following benchmarks (Qin et al., 2024; Wen et al., 2024), assess FlexEval with calibrated human annotations, and examine robustness to dialogue rollout randomness, instruction reformulation, and pipeline-backend changes. Together, these experiments show that instruction-conditioned evaluation is both practically meaningful and empirically reliable.

In summary, our contributions are threefold:

- We formulate **instruction-conditioned eval-**

uation as a complementary setting to fixed instruction-following benchmarks: instead of only asking which model performs best on average, we ask which model most reliably follows a target instruction.

- We propose **FlexBench/FlexEval**, an end-to-end framework that instantiates an instruction-specialized benchmark from a user-provided target instruction by automatically deriving verifiable dimensions, generating instruction-aware dialogue corpora, and aggregating tri-valued judgments with logic-aware Coverage and Achievement metrics.
- We provide a comprehensive empirical validation across diverse multi-turn target instructions, external human-curated decompositions, calibrated human judgments, and robustness analyses.

2 Related Work

Instruction-following evaluation has been studied from increasingly fine-grained perspectives. IFEval (Zhou et al., 2023) introduced verifiable instruction checks, while FollowBench (Jiang et al., 2024) further organized instruction constraints into multiple fine-grained categories. ComplexBench (Wen et al., 2024) focused on instructions composed of multiple constraints and incorporated dependency structures into evaluation, and CFBench (Zhang et al., 2025) constructed a large-scale benchmark of realistic user constraints. A related line of work broadens instruction-following evaluation to richer interaction structures and specialized settings. MT-Bench-101 (Bai et al., 2024) and StructFlowBench (Li et al., 2025) evaluates multi-turn dialogue instruction following, while AgentIF (Qi et al., 2025) targets long and complex instructions in agentic scenarios. Complementary efforts examine correctness versus faithfulness in factual QA (Adlakha et al., 2024), long-context instruction robustness (Wu et al., 2025), and reasoning over structure-rich text (Gu et al., 2025). Domain-specific benchmarks further extend instruction-following evaluation to interactive code generation (Wang et al., 2025), compositional mathematical reasoning (Fu et al., 2025), and expert-domain retrieval (Zhou et al., 2025; Song et al., 2025).

Together, these works substantially improve the granularity, structural coverage, and domain

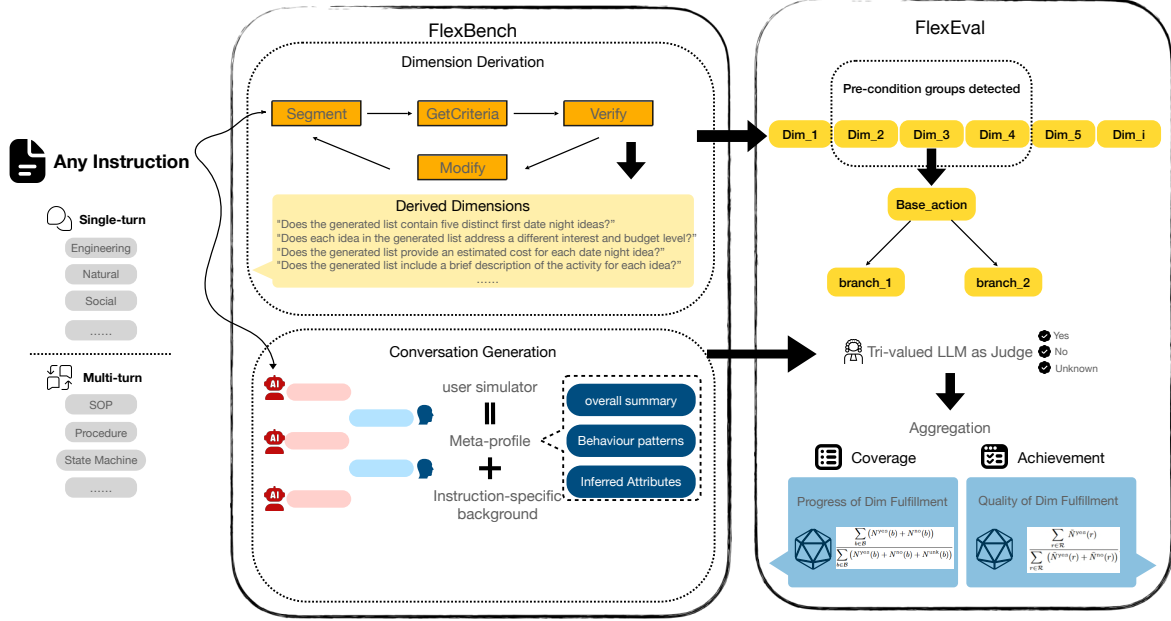


Figure 2: Overview of the FlexBench/FlexEval framework. FlexBench constructs an instruction-specialized benchmark through dimension derivation and conversation generation, while FlexEval performs precondition-aware judging and aggregates dimension-level judgments into Coverage and Achievement.

Work	Multi-turn	Long procedural	Generated corpus	Per-instruction suite	Logic-aware eval.
InFoBench (Qin et al., 2024)	–	–	–	–	–
FollowBench (Jiang et al., 2024)	–	–	–	–	–
ComplexBench (Wen et al., 2024)	–	–	–	–	†
CFBench (Zhang et al., 2025)	–	–	–	–	–
StructFlowBench (Li et al., 2025)	✓	–	✓	–	✓
AgentIF (Qi et al., 2025)	–	✓	–	–	–
FlexBench/FlexEval	✓	✓	✓	✓	†

Table 1: Comparison of representative instruction-following evaluation paradigms. † indicates dependency-based aggregation over decomposed requirements. FlexBench/FlexEval differs from prior fixed evaluation resources by instantiating a specialized evaluation suite from a target instruction, while jointly supporting multi-turn corpus generation, procedural instructions, and logic-aware aggregation.

breadth of instruction-following evaluation. However, they are primarily instantiated as fixed evaluation resources with predefined instructions. FlexBench/FlexEval addresses a complementary setting: rather than providing another fixed evaluation resource, it instantiates an instruction-specialized benchmark from a user-provided target instruction. Table 1 summarizes this positioning relative to representative prior work.

3 FlexBench: Instruction-Specialized Benchmark Construction

As shown in Figure 2, FlexBench constructs an instruction-specialized evaluation suite from a user-provided seed instruction. Given an instruction I ,

FlexBench outputs $\mathcal{B}(I)$, consisting of (i) a set of verifiable evaluation dimensions and (ii) a conversation corpus designed to elicit the behaviors required by I :

$$I \xrightarrow{\text{FLEXBENCH}} \mathcal{B}(I) = (\mathcal{D}, \mathcal{C}), \quad (1)$$

$$\mathcal{D} = \{d_i\}_{i=1}^D, \quad \mathcal{C} = \{c_j\}_{j=1}^M.$$

Unlike a fixed task bank, $\mathcal{B}(I)$ is instantiated on demand from the target instruction itself, so that subsequent evaluation is tailored to the specific requirements, branches, and interaction patterns encoded in I .

Two-stage pipeline. FlexBench is fully automated and contains two modules: (1) **Dimension**

derivation, which converts I into $\mathcal{D}$; and (2) **Conversation generation**, which synthesizes $\mathcal{C}$ to exercise instruction-relevant behaviors in interactive settings. The resulting $\mathcal{B}(I) = (\mathcal{D}, \mathcal{C})$ serves as an instruction-specialized test suite for evaluating whether models faithfully follow I .

3.1 Dimension Derivation

Dimension interface. FlexBench derives dimensions that can be evaluated from a conversation trace τ . Each dimension d_i is an instruction-local compliance signal: it corresponds to one requirement, or a small set of tightly related requirements, rewritten as an entity-explicit and observable criterion. For each d_i , we produce: (i) a verifiable criterion specifying the behavior to be checked, and (ii) when needed, an optional precondition describing when that criterion becomes applicable. Thus, a dimension-level result will not be interpreted as a standalone measure of general model capability, but as evidence of whether the model satisfies a specific requirement of the target instruction.

Operational pipeline. We implement dimension derivation as a structured process with four stages. Prompt templates are provided in Appendix B.

1. **Instruction segmentation:** split I into non-overlapping requirement units that capture behaviorally relevant content.
2. **Criterion generation:** rewrite each requirement unit into an explicit, conversation-observable evaluation criterion, while preserving conditional dependencies.
3. **Coverage verification:** check whether the current dimension set captures the core requirements of I .
4. **Targeted modification:** revise the dimension set when verification identifies missing requirements, unclear wording, overly broad criteria, or incomplete conditional dependencies.

Derivation properties. The pipeline is designed to produce dimensions that are complete with respect to the source instruction, faithful to its stated requirements, non-redundant, and evaluable from conversation traces. We empirically assess these properties in Section 5.3.

3.2 Conversation Generation

Given the target instruction I , FlexBench synthesizes a conversation corpus $\mathcal{C}(I) = \{c_j\}_{j=1}^M$ using a user simulator. The goal is to expose the evaluated model to diverse, instruction-relevant interaction trajectories. The simulator is conditioned on the instruction context and user-profile controls. Conversation-generation details, including user-profile distribution, backend settings, and simulator separation from evaluation dimensions, are provided in Appendix A.

User simulator. We first define a small collection of *meta-profiles* that capture stable interaction styles, such as different levels of user cooperativeness, information disclosure, and problem-solving behavior. Since user-simulation quality affects interactive evaluation (Naous et al., 2025), we combine these profile controls with *instruction-specific background* that adapts the simulator to the corresponding task context.

Rollout. For each conversation c_j , we sample an interaction between an evaluated model f and an instruction-adapted simulator $U_I^{(k)}$, where k indexes the selected meta-profile and z_j is the rollout seed:

$$c_j \sim \text{Rollout}\left(f, U_I^{(k)}; z_j\right). \quad (2)$$

Generation properties. The generated corpus is designed to be instruction-grounded, diverse, and reproducible. Instruction grounding ensures that dialogues surface the behaviors, branches, and decisions implied by the target instruction I . Diversity is introduced through non-deterministic user simulation and rollout seeds, which exercise different paths of the same instruction. Reproducibility is maintained through controlled interaction trajectories, and its effect is evaluated in Section 5.5.

4 FlexEval: Coverage and Achievement

FlexEval aggregates per-dimension judgments into instruction-level scores. It separates two common failure modes in instruction following: (i) **skipping required steps** and (ii) **making mistakes when an applicable step is exercised**. Accordingly, we report two complementary metrics: COVERAGE and ACHIEVEMENT.

4.1 Tri-valued Judging and Logic Groups

Let $\mathcal{D}$ be the derived dimensions (Section 3.1) and $\mathcal{C}(I)$ the conversation corpus for instruction

I (Section 3.2). For each trace $\tau \in \mathcal{C}(I)$ and dimension $d \in \mathcal{D}$, FlexEval assigns $s_d(\tau) \in \{\text{yes}, \text{no}, \text{unknown}\}$. Here, yes means the requirement is satisfied, no means it is applicable but not satisfied, and unknown means the trace lacks sufficient evidence or the relevant precondition is not triggered.

To respect conditional structure, FlexEval organizes dimensions into *logic groups*, each consisting of a base action and its dependent branch requirements. Base actions measure workflow progress, while dependent branches measure correctness under activated conditions. Mutually exclusive or inactive branches are marked as unknown and excluded from correctness aggregation, rather than being counted as failures. Logic-group extraction and formal aggregation details are provided in Appendix C.

4.2 Metrics

Coverage (workflow exposure). Coverage measures whether the model attempts the required base actions, regardless of correctness. Let B_{att} be the number of base-action instances judged yes or no, and let B_{all} be the total number of base-action instances:

$$\text{Coverage}(I) = \frac{B_{\text{att}}}{B_{\text{all}}}. \quad (3)$$

Low Coverage indicates that required workflow steps are skipped or not evidenced in the interaction.

Achievement (conditional correctness). Achievement measures correctness once relevant requirements become applicable. Let R_{yes} be the number of applicable branch instances judged yes, and let R_{att} be the number of applicable branch instances judged yes or no:

$$\text{Achievement}(I) = \frac{R_{\text{yes}}}{R_{\text{att}}}. \quad (4)$$

If $R_{\text{att}} = 0$, we report n/a .

Why two metrics? Coverage and Achievement provide complementary views of instruction following. Coverage measures whether the workflow units are exercised, whereas Achievement measures whether applicable requirements are satisfied once they become observable in the dialogue. In relatively linear workflows, low Coverage is consistent with skipped or unevidenced steps. In highly branched workflows, however, a finite corpus of

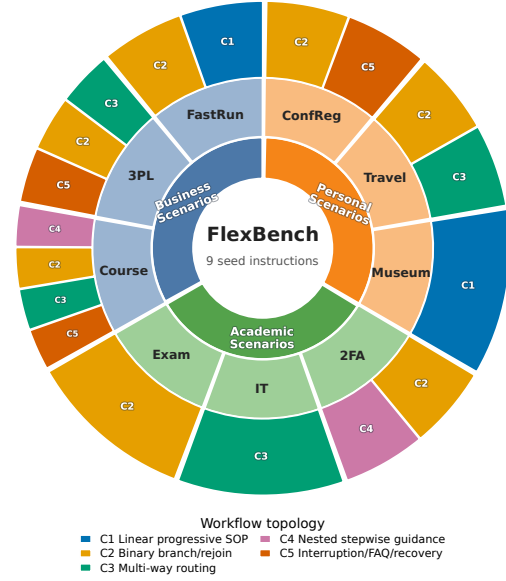


Figure 3: Taxonomy of the nine multi-turn seed instructions used in our experiments. The inner ring groups instructions by scenario category, the middle ring lists individual seed instructions, and the outer ring indicates their workflow-topology patterns.

simulated users may fail to trigger some conditional branches, so low Coverage may also reflect limited branch exposure rather than failure to execute every corresponding requirement.

Reporting both metrics therefore separates workflow-exposure gaps from conditional-correctness failures. Achievement is more interpretable when Coverage is high, while low Coverage indicates that Achievement should not be interpreted as a complete assessment of the entire workflow. A concrete example is provided in Appendix D.

5 Experiments

5.1 Experimental Setup

Seed instruction suite. We instantiate FlexBench/FlexEval on nine multi-turn seed instructions. To ensure diversity beyond a single application setting, we organize the instruction suite along two complementary axes. Figure 3 summarizes this taxonomy.

Models under evaluation. We evaluate 9 LLMs: GPT-4.1, GPT-4.1-mini, and GPT-4.1-nano (OpenAI, 2025); Claude Sonnet 4.5 (Anthropic, 2025); Gemini 2.5 Pro (Comanici et al., 2025); Qwen3-235B-A22B (Yang et al., 2025); GLM-4.5 (Zeng

et al., 2025); DeepSeek-V4-Pro (DeepSeek, 2026); and Doubao-Seed-2.0-Pro (ByteDance, 2026).

Benchmark construction and evaluation backend. LLM-driven components of FlexBench and FlexEval are instantiated with fixed backend models: (1) dimension derivation (SEGMENTATION → CRITERIA → VERIFICATION → MODIFICATION), (2) logic extraction for precondition groups and mutual-exclusion patterns, and (3) per-dimension tri-valued judging used by FlexEval are all performed with Claude Opus 4.6 (Anthropic, 2026), running with temperature 0.

Dialogue rollouts and decoding settings. For each seed instruction and tested model, we generate 48 multi-turn dialogues, yielding 3,888 dialogues in total. The tested models use their provider-specific default decoding settings. Detailed backend settings are provided in Appendix A.

5.2 Main Results

FlexEval reports both COVERAGE and ACHIEVEMENT. Table 3 reports the model-by-seed ACHIEVEMENT scores, while Table 2 provides a compact joint view of mean COVERAGE and ACHIEVEMENT for each seed. Coverage is close to saturation for most seeds, but the branch-heavy 3PL workflow is an important exception. We therefore interpret COVERAGE as a workflow-exposure diagnostic and ACHIEVEMENT as correctness over the requirements that become applicable in the observed dialogues.

Table 3 presents Achievement scores across the nine multi-turn seed instructions, grouped into business, academic, and personal scenarios. Claude Sonnet 4.5 has the highest point-estimate mean Achievement (0.508), but this point estimate does not establish a definitively separated overall winner. The ranking is not uniform across instructions: different models show distinct strengths across scenarios and individual seeds, and models with similar average scores can exhibit noticeably different performance profiles.

To quantify within-seed uncertainty, we resample the 48 dialogue-level outcomes with replacement and bootstrap the best-vs. runner-up Achievement difference. The resulting bootstrap intervals include zero for eight of the nine seeds; Course is the only seed with a positive interval (10.0 percentage points, 95% bootstrap interval [0.3, 19.6]). These intervals quantify dialogue-level sampling

Seed	Achievement (%)	Coverage (%)	Uncovered (%)
FastRun	23.1	100.0	0.0
3PL	41.6	26.3	73.7
Course	34.6	98.2	1.8
Exam	57.8	100.0	0.0
IT	45.4	99.3	0.7
2FA	53.3	99.5	0.5
Museum	63.4	88.1	11.9
Travel	48.9	100.0	0.0
ConfReg	48.2	99.9	0.1

Table 2: Joint seed-level view of mean ACHIEVEMENT and mean COVERAGE, averaged over the nine evaluated models. ACHIEVEMENT measures correctness over applicable requirements, whereas COVERAGE diagnoses workflow exposure.

uncertainty under the realized generation setup, rather than variation from regenerating an entirely new dialogue corpus. The complete per-seed results are reported in Appendix F. We therefore interpret the results as showing that point-estimate leaders and failure profiles vary across target instructions, while most best-vs. runner-up gaps remain statistically close.

The seed-wise results also reveal clear differences in instruction difficulty. Museum Booking and Exam Rescheduling yield relatively high Achievement across models, whereas FastRun and Course Support are more challenging. These differences show why a single aggregate score can obscure instruction-specific difficulty and model-specific variation. The relevant question is therefore not only which model performs better on average, but also which model is more reliable for a particular target instruction and whether its apparent advantage is sufficiently separated under the available evaluation evidence.

5.3 Instruction Decomposition

We next evaluate whether FlexBench recovers externally curated instruction-following constraints. We use two established benchmarks with human-provided decomposition references: the 248 instructions in the InFoBench Hard set and an instruction-following subset of ComplexBench. For ComplexBench, we exclude translation-style prompts and generic quality criteria that do not correspond to explicit instruction-following constraints. This yields 448 external benchmark instructions paired with human-provided reference decompositions. Additional details on matching variants, preference criteria, and mismatch adjudication are provided in Appendix E.

Model	Business			Academic			Personal			Overall	
	FastRun	3PL	Course	Exam	IT	2FA	Museum	Travel	ConfReg	Avg.	Std.
Claude Sonnet 4.5	0.279	<u>0.504</u>	0.468	0.603	0.436	0.542	0.703	0.514	0.525	0.508	0.110
DeepSeek-V4-Pro	0.227	0.474	0.360	0.581	0.460	<u>0.561</u>	0.671	0.494	0.475	<u>0.478</u>	0.121
Doubao-Seed-2.0-Pro	0.253	0.426	0.358	0.623	0.467	0.563	<u>0.674</u>	0.468	0.463	0.477	0.122
GPT-4.1	0.203	0.458	<u>0.368</u>	<u>0.604</u>	0.496	0.553	0.617	0.491	0.496	0.476	0.120
Gemini 2.5 Pro	0.211	0.524	0.336	0.576	<u>0.495</u>	0.540	0.601	0.497	0.500	0.476	0.117
GPT-4.1-mini	0.263	0.392	0.368	0.579	0.470	0.518	0.673	<u>0.518</u>	0.467	0.472	0.114
Qwen3-235B-A22B	0.234	0.348	0.343	0.537	0.459	0.530	0.564	0.529	<u>0.515</u>	0.451	0.108
GLM-4.5	0.135	0.403	0.322	0.528	0.396	0.509	0.638	0.429	0.467	0.425	0.133
GPT-4.1-nano	<u>0.276</u>	0.214	0.186	0.567	0.407	0.478	0.567	0.463	0.434	0.399	0.135

Table 3: Main results across nine instruction-conditioned multi-turn evaluation suites. Scores are ACHIEVEMENT judged by Claude Opus 4.6. Seed instructions are grouped by scenario category. Avg. and Std. are computed across the nine seeds. Best and second-best results in each column are shown in bold and underlined respectively.

Semantic recovery against human decompositions. We assess the semantic alignment between FlexBench decompositions and human references using SBERT cosine matching. Specifically, each FlexBench dimension is matched to its most similar human reference item, and vice versa. As shown in Table 4, FlexBench achieves an overall semantic recovery F1 of 80.0 across the 448 external instructions, with 79.5 precision and 80.6 recall. At the same time, the overall granularity ratio of 1.160 indicates that FlexBench produces moderately finer-grained decompositions than the human references, rather than simply reproducing them verbatim.

Preference-based validation. Semantic recovery alone does not capture whether a decomposition is useful for evaluation. We therefore conduct a randomized A/B comparison between FlexBench and human decompositions using GPT-5.4 (OpenAI, 2026) as an independent judge. Overall, FlexBench is preferred in 75.0% of cases. The advantage is strongest on evaluation suitability and atomicity/granularity, showing that FlexBench tends to produce dimensions that are more directly usable as evaluation criteria and more fine-grained for diagnosing model behavior.

Interpreting low-similarity and extra dimensions. We further analyze low-similarity matches and additional FlexBench dimensions that are not matched to any human reference item. Among low-similarity cases, only 3.1% are true FlexBench misses. Moreover, 21.6% of the additional FlexBench dimensions are judged to be valid constraints omitted by the human references. These results indicate that FlexBench does more than

Benchmark	N	P	R	F1	Gran.
InFoBench	248	83.9	85.2	84.5	1.139
ComplexBench	200	70.6	71.7	71.1	1.186
All	448	79.5	80.6	80.0	1.160

Table 4: External semantic recovery of FlexBench decompositions against human reference decompositions from InFoBench and ComplexBench. P, R, and F1 are computed using SBERT cosine best-match alignment; Gran. denotes $|D_{\text{flex}}|/|D_{\text{human}}|$.

Benchmark	Flex Win	Human Win	Tie
InFoBench	64.9% (161)	33.1% (82)	2.0% (5)
ComplexBench	87.5% (175)	11.5% (23)	1.0% (2)
All	75.0% (336)	23.4% (105)	1.6% (7)

Table 5: GPT-5.4 pairwise preference between FlexBench and human instruction decompositions.

recover existing human decompositions: it can produce more atomic and evaluation-oriented dimensions that expose instruction-following requirements left implicit or under-specified in human references.

5.4 Reliability of FlexEval

To assess the reliability of the main FlexEval judge, we construct a human meta-evaluation set over the three business scenarios, three tested models, and 40 dialogues for each scenario-model pair, yielding 360 dialogues and 8,880 dimension-level judgment instances. We employ three trained annotators and use a binary checklist interface: annotators judge whether each applicable requirement is satisfied, while conditional dependencies are deterministically mapped back to the tri-valued

Scenario	Human–Judge Agreement
FastRun	88.02%
3PL Caller	86.25%
Course Support	87.63%
Overall Cohen’s κ	0.772

Table 6: Human validation of the Claude Opus 4.6 FlexEval judge. We report dimension-level agreement with human annotations and overall Cohen’s κ .

FlexEval label space of *yes/no/unknown*. Before full annotation, the annotators complete a pilot round on 10 dialogues covering roughly 100 dimensions, followed by scenario-specific calibration until their agreement with reference annotations exceeds 95%. As shown in Table 6, agreement between calibrated human labels and the Claude Opus 4.6 judge ranges from 86.25% to 88.02% across scenarios, with an overall three-label Cohen’s κ of 0.772. These results support both the reliability of the main automatic judge and the evaluability of the dimensions produced by FlexBench.

5.5 Robustness Analysis

Robustness is especially important in multi-turn settings, where recent work shows that LLMs can degrade substantially relative to single-turn settings and may fail to recover after early conversational mistakes (Laban et al., 2025). We further examine the robustness of FlexBench/FlexEval from three perspectives: stability under stochastic dialogue rollouts, sensitivity to reformulations of the seed instruction, and robustness to changing the LLM backend used by the construction/evaluation pipeline. Detailed robustness tables and backend-correlation plots are provided in Appendix F.

5.5.1 Self and Cross-model Consistency

We first assess whether dialogue simulation randomness materially affects the evaluation outcome. Using *Travel Change* as the seed instruction, we independently generate five dialogue corpora for each model, with 48 dialogues per corpus. We evaluate five representative models: Claude Sonnet 4.5, GPT-4.1-mini, GPT-4.1, GPT-4.1-nano, and GLM-4.5. Figure 4(a)–(d) summarizes the results.

Coverage is nearly saturated across corpora and models. Achievement shows larger but still modest variation: the mean absolute within-model deviation is 0.88 percentage points, and the 95% limits

of agreement are $[-2.15, 2.15]$ percentage points. The broad cross-model performance pattern is also preserved across corpora, with stronger and weaker model groups remaining largely stable. These results indicate that FlexEval provides a stable aggregate evaluation signal under dialogue rollout randomness.

5.5.2 Sensitivity to Seed Reformulation

We next examine whether FlexBench/FlexEval remains stable when the same seed instruction is expressed in different forms. Using *Travel Change*, we keep the dialogue corpus fixed for each model and vary only the evaluation seed and the resulting decomposed dimensions. In the main analysis, we consider two reformulations: a format-only variant and a lexical-rewrite variant. Both variants preserve the overall dimension count and remain semantically close to the original decomposition.

5.5.3 Pipeline-Backend Robustness

Finally, we examine whether the aggregate conclusions depend on using a single LLM backend for the FlexBench/FlexEval pipeline. In addition to the main Claude Opus 4.6 pipeline, we rerun the LLM-driven construction and evaluation pipeline with GPT-5.4 and Gemini 3.1 Pro Preview (Google, 2026). Across the nine tested models, model-level average Achievement remains highly correlated across backends: Claude vs. GPT-5.4 obtains a Spearman rank correlation of 0.917 and Pearson score correlation of 0.984, while Claude vs. Gemini 3.1 Pro obtains 0.883 and 0.976, respectively. These results suggest that the main model-level conclusions are not an artifact of a single construction/evaluation backend.

Computational cost and scalability. For the main 9-seed, 9-model, and 48-dialogue-per-cell experiment, token usage averages 10.2k input and 4.2k output tokens for construction per seed, 13.1k input and 1.5k output tokens for dialogue generation per dialogue, and 1,112 input and 159 output tokens for each dimension–dialogue judgment. The experiment contains $9 \times 48 \times 140 = 60,480$ dimension–dialogue judgments. The judging stage scales linearly with the number of models, dialogues, and derived dimensions, and individual judgments can be parallelized. Detailed token usage is reported in Appendix F, Table 19.

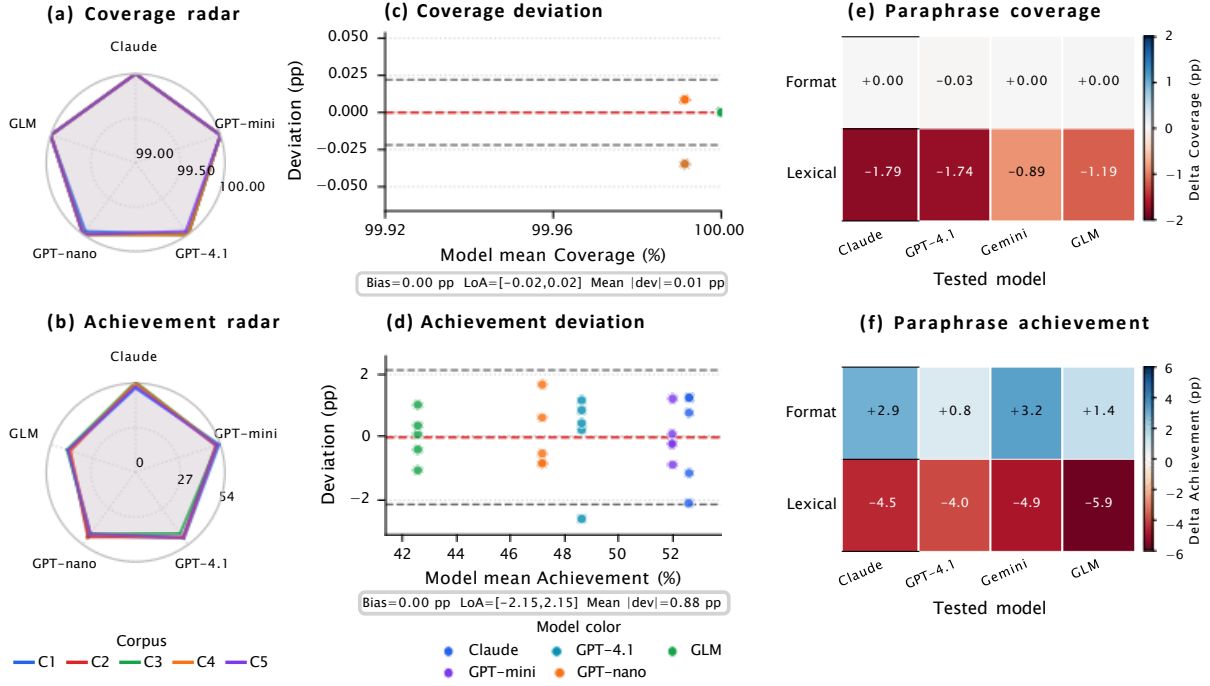


Figure 4: Robustness analyses of FlexBench/FlexEval. Panels (a)–(d) examine self and cross-model consistency across five independently generated dialogue corpora for each model. Panels (e)–(f) report changes in Coverage and Achievement under instruction reformulations.

6 Conclusion

We introduced **FlexBench** and **FlexEval**, an instruction-conditioned evaluation framework that instantiates a specialized test suite from a target instruction. FlexBench derives verifiable evaluation dimensions and generates instruction-conditioned dialogue corpora, while FlexEval aggregates dimension-level judgments into COVERAGE and ACHIEVEMENT to separate workflow progress from conditional correctness. Across diverse multi-turn seed instructions and contemporary LLMs, our results show that model strengths vary substantially across target instructions, supporting the need for instruction-conditioned evaluation rather than relying only on static aggregate rankings.

Limitations

FlexBench/FlexEval is designed for instruction-conditioned evaluation, so its scores should be interpreted as compliance with a specific target instruction rather than as a universal measure of general model capability. The quality of the instantiated test suite therefore depends on the clarity and specificity of the seed instruction; highly ambiguous, underspecified, or strongly interdependent requirements may be difficult to decompose

into verifiable dimensions.

In addition, although FlexBench/FlexEval mitigates backend dependence through external decomposition validation, calibrated human judgments, and pipeline-backend robustness experiments, the framework still relies on LLM-driven construction and judging. Prior work shows that LLM judges can be sensitive to perturbations and systematic biases, including position bias (Chen et al., 2024; Shi et al., 2025). Future work will further explore stronger judge calibration, multi-backend consensus, real-user validation, multimodal traces, and more complex dependency structures in long-horizon workflows.

References

- Vaibhav Adlakha, Parishad BehnamGhader, Xing Han Lu, Nicholas Meade, and Siva Reddy. 2024. Evaluating correctness and faithfulness of instruction-following models for question answering. *Transactions of the Association for Computational Linguistics*, 12:681–699.
- Anthropic. 2025. [Introducing Claude Sonnet 4.5](#).
- Anthropic. 2026. [Introducing Claude Opus 4.6](#).
- Ge Bai, Jie Liu, Xingyuan Bu, Yancheng He, Jiaheng Liu, Zhanhui Zhou, Zhuoran Lin, Wenbo Su, Tiezheng Ge, Bo Zheng, and Wanli Ouyang. 2024.

- Mt-bench-101: A fine-grained benchmark for evaluating large language models in multi-turn dialogues. *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7421–7454.
- ByteDance. 2026. [Doubao-Seed-2.0 system card](#).
- Guiming Hardy Chen, Shunian Chen, Ziche Liu, Feng Jiang, and Benyou Wang. 2024. Humans or llms as the judge? a study on judgement bias. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8301–8327.
- Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.
- DeepSeek. 2026. [DeepSeek V4 preview release](#).
- Tingchen Fu, Jiawei Gu, Yafu Li, Xiaoye Qu, and Yu Cheng. 2025. Scaling reasoning, losing control: Evaluating instruction following in large reasoning models. *arXiv preprint arXiv:2505.14810*.
- Google. 2026. [Gemini 3.1 Pro](#).
- Zhouhong Gu, Haoning Ye, Xingzhou Chen, Zeyang Zhou, Hongwei Feng, and Yanghua Xiao. 2025. Structtext-eval: Evaluating large language model’s reasoning ability in structure-rich text. *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 223–244.
- Yuxin Jiang, Yufei Wang, Xingshan Zeng, Wanjun Zhong, Liangyou Li, Fei Mi, Lifeng Shang, Xin Jiang, Qun Liu, and Wei Wang. 2024. Follow-bench: A multi-level fine-grained constraints following benchmark for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4667–4688.
- Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. 2025. Llms get lost in multi-turn conversation. *arXiv preprint arXiv:2505.06120*.
- Jinnan Li, Jinzhe Li, Yue Wang, Yi Chang, and Yuan Wu. 2025. Structflowbench: A structured flow benchmark for multi-turn instruction following. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 9322–9341.
- Subhrangshu Nandi, Arghya Datta, Rohith Nama, Udit Patel, Nikhil Vichare, Indranil Bhattacharya, Prince Grover, Shivam Asija, Giuseppe Carenini, Wei Zhang, Arushi Gupta, Sreyoshi Bhaduri, Jing Xu, Huzefa Raja, Shayan Ray, Aaron Chan, Esther Xu Fei, Gaoyuan Du, Zuhair Akhtar, and 5 others. 2025. Sop-bench: Complex industrial sops for evaluating llm agents. *arXiv preprint arXiv:2506.08119*.
- Tarek Naous, Philippe Laban, Wei Xu, and Jennifer Neville. 2025. Flipping the dialogue: Training and evaluating user language models. *arXiv preprint arXiv:2510.06552*.
- OpenAI. 2025. [Introducing GPT-4.1 in the API](#).
- OpenAI. 2026. [GPT-5.4 Thinking System Card](#).
- Preethi Prem, Kie Shidara, Vikasini Kuppa, Esmé Wheeler, Feng Liu, Ahmed Alaa, and Danilo Bernardo. 2026. Medevalarena: A self-generated, peer-judged benchmark for medical reasoning. *medRxiv*, pages 2026–01.
- Yunjia Qi, Hao Peng, Xiaozhi Wang, Amy Xin, Youfeng Liu, Bin Xu, Lei Hou, and Juanzi Li. 2025. Agentif: Benchmarking instruction following of large language models in agentic scenarios. *arXiv preprint arXiv:2505.16944*.
- Yiwei Qin, Kaiqiang Song, Yebowen Hu, Wenlin Yao, Sangwoo Cho, Xiaoyang Wang, Xuansheng Wu, Fei Liu, Pengfei Liu, and Dong Yu. 2024. Infobench: Evaluating instruction following ability in large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 13025–13048.
- Sumuk Shashidhar, Clémentine Fourrier, Alina Lozovskia, Thomas Wolf, Gokhan Tur, and Dilek Hakkani-Tür. 2025. Yourbench: Easy custom evaluation sets for everyone. *arXiv preprint arXiv:2504.01833*.
- Lin Shi, Chiyu Ma, Wenhua Liang, Xingjian Diao, Weicheng Ma, and Soroush Vosoughi. 2025. Judging the judges: A systematic study of position bias in llms-as-a-judge. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pages 292–314.
- Tingyu Song, Guo Gan, Mingsheng Shang, and Yilun Zhao. 2025. Ifir: A comprehensive benchmark for evaluating instruction-following in expert-domain information retrieval. *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 10186–10204.
- Peiding Wang, Li Zhang, Fang Liu, Lin Shi, Minxiao Li, Bo Shen, and An Fu. 2025. Codeif-bench: Evaluating instruction-following capabilities of large language models in interactive code generation. *arXiv preprint arXiv:2503.22688*.
- Bosi Wen, Pei Ke, Xiaotao Gu, Lindong Wu, Hao Huang, Jinfeng Zhou, Wenchuan Li, Binxin Hu, Wendy Gao, Jiaying Xu, and 1 others. 2024. Benchmarking complex instruction-following with multiple constraints composition. *Advances in Neural Information Processing Systems*, 37:137610–137645.

Xiaodong Wu, Minhao Wang, Yichen Liu, Xiaoming Shi, He Yan, Xiangju Lu, Junmin Zhu, and Wei Zhang. 2025. Lifbench: Evaluating the instruction following performance and stability of large language models in long-context scenarios. *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16445–16468.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou, Bin Chen, Chengxing Xie, Cunxiang Wang, Da Yin, Hao Zeng, Jiajie Zhang, Kedong Wang, Lucen Zhong, Mingdao Liu, Rui Lu, Shulin Cao, Xiaohan Zhang, Xuancheng Huang, Yao Wei, Yean Cheng, and 151 others. 2025. Glm-4.5: Agentic, reasoning, and coding (arc) foundation models. *arXiv preprint arXiv:2508.06471*.

Tao Zhang, Chenglin Zhu, Yanjun Shen, Wenjing Luo, Yan Zhang, Hao Liang, Fan Yang, Mingan Lin, Yujing Qiao, Weipeng Chen, and 1 others. 2025. Cfbench: A comprehensive constraints-following benchmark for llms. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 32926–32944.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*.

Jianqun Zhou, Yuanlei Zheng, Wei Chen, Qianqian Zheng, Hui Su, Wei Zhang, Rui Meng, and Xiaoyu Shen. 2025. Beyond content relevance: Evaluating instruction following in retrieval models. *International Conference on Learning Representations*, 2025:84965–84996.

Yajie Zhou, Jiajun Ruan, Eric S Wang, Sadjad Fouladi, Francis Y Yan, Kevin Hsieh, and Zaoxing Liu. 2026. Netarena: Dynamic benchmarks for ai agents in network automation. In *The Fourteenth International Conference on Learning Representations*.

A Experimental Assets and Setup

This appendix summarizes the seed instructions, backend settings, and conversation-generation setup used in the main experiments.

A.1 Seed Instruction Suite

Table 7 summarizes the nine multi-turn seed instructions used in the main experiments. The

suite covers three scenario categories and multiple workflow-topology patterns, matching the taxonomy in Figure 3. Topology labels are defined as follows: C1 = linear progressive SOP, C2 = binary branch/rejoin, C3 = multi-way routing, C4 = nested stepwise guidance, and C5 = interruption/FAQ/recovery. These labels are used only to characterize the seed suite; FlexBench does not take them as input.

A.2 Backend Models and Decoding Settings

Table 8 lists the backend models used by the LLM-driven components of FlexBench/FlexEval. The main experiments use Claude Opus 4.6 for dimension derivation, logic extraction, and FlexEval judging. For pipeline-backend robustness, we additionally rerun these three construction/evaluation components with GPT-5.4 and Gemini 3.1 Pro Preview. User simulation and completion detection are kept fixed. The tested models use their provider-specific default decoding settings; we do not override their temperatures.

All LLM-driven construction and evaluation stages are executed automatically with fixed prompts and programmatic wrappers. For dimension derivation, the verification–modification loop is repeated until the verifier returns complete or the retry budget is exhausted. The completion detector is used only to decide whether a dialogue should stop; it is not used to score model behavior. Full pipeline-backend robustness results are reported in Appendix F.

A.3 Conversation Generation Setup

For each seed instruction and tested model, FlexBench generates 48 multi-turn dialogues using a fixed user-profile distribution over cooperation scores: 7 dialogues with score 1, 12 with score 2, 12 with score 3, 12 with score 4, and 5 with score 5. With nine seed instructions and nine tested models, the main experiment contains $9 \times 9 \times 48 = 3,888$ dialogues.

The simulator receives the seed instruction, dialogue history, and a selected user profile. It does not receive the derived evaluation dimensions, logic groups, or model scores. This separation helps ensure that generated user turns elicit instruction-relevant behavior without directly revealing the evaluation criteria. Rollouts terminate when the task is completed, when the simulator ends the interaction, or when the maximum turn limit is reached.

Seed	Scenario	Topology	Words	Sent.	#Dims
FastRun	Business	C1/C2	351	26	8
3PL	Business	C3/C2/C5	848	130	42
Course	Business	C4/C2/C3/C5	353	37	24
Exam	Academic	C2	207	20	9
IT	Academic	C3	222	23	12
2FA	Academic	C4/C2	250	33	10
Museum	Personal	C1	245	26	12
Travel	Personal	C3/C2	310	26	13
ConfReg	Personal	C5/C2	256	22	10
Total	—	—	3,042	343	140

Table 7: Statistics of the nine seed instructions used in the main multi-turn experiments. Sent. denotes sentence-like instruction units, and #Dims denotes the number of final FlexBench evaluation dimensions.

B Prompt Templates

This section summarizes the prompt templates used in the dimension derivation pipeline. The pipeline uses four prompts: instruction segmentation, criteria generation, dimension-set verification, and targeted modification. Bracketed fields such as `[[INSTRUCTION]]` and `[[DIMENSIONS]]` are filled programmatically.

Prompt A: instruction segmentation. This prompt converts an assistant instruction into behaviorally relevant requirement units.

You are given a fully specified instruction for a conversational AI assistant. Identify requirement units that describe observable assistant behavior or concrete task constraints. Each unit should capture one coherent requirement that can later be converted into an evaluation dimension.

Return valid JSON only, using this format: `{ "segment": "...", ... }`

Rules: (1) Requirement units should be non-overlapping. (2) Preserve the core behavioral requirements of the instruction. (3) Do not split a single coherent action into unnecessary fragments. (4) Focus on observable conversation behavior, communication patterns, branching conditions, and concrete requirements that can be checked from generated dialogues. (5) Do not add requirements that are not supported by the instruction.

Instruction: `[[INSTRUCTION]]`

Prompt B: criteria generation. This prompt rewrites requirement units into conversation-observable evaluation criteria.

You are given an assistant instruction and its segmented requirement units. Convert each segment into an evaluation criterion for judging whether a generated conversation follows the instruction.

Return valid JSON only.

Rules: (1) Every criterion must be grounded in the instruction and the corresponding segment.

(2) Do not introduce new entities, numbers, formats, policies, or constraints. (3) If a requirement is conditional, make the condition explicit using “if” or “when”. (4) Replace vague step references with concrete observable actions. (5) Preserve recommended scripts or required phrasings when they are explicitly specified. (6) Each criterion should be answerable from the conversation transcript.

Instruction: `[[INSTRUCTION]]`

Segments: `[[SEGMENTS]]`

Prompt C: dimension-set verification. This prompt checks whether the derived dimension set covers the instruction’s core observable requirements.

You are given an assistant instruction and a list of derived evaluation dimensions. Check whether the dimensions cover the instruction’s core observable requirements.

If the dimensions are sufficient, return: `{ "coverage": "complete" }`

If important requirements are missing or under-specified, return: `{ "coverage": "incomplete", "missing_segment": "...", ... }`

Rules: (1) Consider only requirements grounded in the instruction. (2) Focus on requirements that would affect evaluation of generated conversations. (3) Do not require dimensions for irrelevant formatting or structural text unless it changes assistant behavior. (4) Return valid JSON only.

Instruction: `[[INSTRUCTION]]`

Dimensions: `[[DIMENSIONS]]`

Prompt D: targeted modification. This prompt revises the dimension set using verifier feedback while preserving the output schema.

You are given an assistant instruction, the current evaluation dimensions, and verification feedback. Revise the dimension set only as needed to address the feedback.

Return the complete revised JSON only.

Rules: (1) Add missing dimensions when core observable requirements are uncovered. (2) Clarify dimensions that are vague, overly broad, or missing necessary conditions. (3) Preserve grounding to the original instruction. (4) Complete explicit conditional dependencies when needed. (5) Avoid adding unsupported requirements or expanding the evaluation scope. (6) Preserve the existing JSON schema.

Instruction: `[[INSTRUCTION]]`

Current dimensions: `[[CURRENT_DIMENSIONS]]`

Verification feedback: `[[VERIFICATION_FEEDBACK]]`

The verification–modification loop is repeated until the verifier returns complete or the retry budget is exhausted. If the retry budget is reached, the pipeline returns the latest revised dimension set.

Component	Main backend	Alternative pipeline backends	Temp.
Dimension derivation	Claude Opus 4.6	GPT-5.4; Gemini 3.1 Pro Preview	0; retry +0.05
Logic extraction	Claude Opus 4.6	GPT-5.4; Gemini 3.1 Pro Preview	0
FlexEval judge	Claude Opus 4.6	GPT-5.4; Gemini 3.1 Pro Preview	0
User simulator	Gemini 3.1 Pro Preview	–	0.7
Completion detector	Qwen-Max	–	0.1
Tested models	Provider APIs	–	Default

Table 8: Backend models and decoding settings. The main experiments use Claude Opus 4.6 for dimension derivation, logic extraction, and FlexEval judging. For pipeline-backend robustness, the construction/evaluation backend is switched as a whole to GPT-5.4 or Gemini 3.1 Pro Preview. User simulation and completion detection are kept fixed.

C Logic Group Extraction and Formal Aggregation

This appendix describes how FlexEval extracts logic groups from derived dimensions and how the formal aggregation of COVERAGE and ACHIEVEMENT is computed.

C.1 Logic Group Extraction

Given a set of derived dimensions $\mathcal{D}$, FlexEval identifies precondition-based logic groups. Each group contains a base action and the branch requirements that become applicable only after the base action is attempted. This structure is used to distinguish workflow progress from conditional correctness.

Output schema. The logic extractor returns a list of groups in the following format:

```
{
  "logic_groups": [
    {
      "base_action": "...",
      "branches": [
        {
          "dimension": "...",
          "condition": "...",
          "condition": "..."
        }, ...
      ]
    }
  ]
}
```

Logic extraction prompt. We use the following prompt template to identify base actions, conditional branches, and mutually exclusive alternatives.

You are given a list of evaluation dimensions used to assess whether an assistant follows an instruction.

Identify groups of dimensions that form conditional logic structures. Each group should contain: (1) a base action that triggers a branch or downstream requirement, and (2) the branch dimensions that become applicable only under specific outcomes or conditions of that base action.

Focus on: - if/else or when/not-when conditions; - success/failure, yes/no, visible/not-visible, available/unavailable branches; - multi-way routing among alternative user states or tools; - mutually exclusive alternatives under the same scenario.

Rules: 1. Use only dimensions from the provided list. 2. Do not invent new dimensions or conditions. 3. Do not mark sequential actions as mutually exclusive. 4. Do not group dimensions unless they share the same underlying scenario or base action. 5. If no clear base action exists, return an empty group.

Return valid JSON only.

Dimensions: [[DIMENSIONS]]

C.2 Formal Aggregation

Let $\mathcal{C}(I)$ be the conversation corpus for instruction I . For compactness, we write the three verdict labels as $y = \text{yes}$, $n = \text{no}$, and $u = \text{unknown}$. For each dimension d , let $s_d(\tau) \in \{y, n, u\}$ denote its verdict on trace τ .

For a verdict $v \in \{y, n, u\}$, define the corpus-level count:

$$N_v(d) = \sum_{\tau \in \mathcal{C}(I)} \mathbb{I}[s_d(\tau) = v]. \quad (5)$$

A dimension is considered attempted when it is judged y or n :

$$N_{\text{att}}(d) = N_y(d) + N_n(d). \quad (6)$$

Let $\mathcal{G}$ be the set of extracted logic groups. Each group $g \in \mathcal{G}$ contains a base action b_g and a set of dependent branch dimensions $\mathcal{R}_g$. We define the set of base actions and branch dimensions as:

$$\mathcal{B} = \{b_g : g \in \mathcal{G}\}. \quad (7)$$

$$\mathcal{R} = \bigcup_{g \in \mathcal{G}} \mathcal{R}_g. \quad (8)$$

For each branch dimension $r \in \mathcal{R}$, let $b(r)$ denote its associated base action. We use $a_r(\tau)$ to indicate whether branch r is active on trace τ :

$$a_r(\tau) = \mathbb{I}[s_{b(r)}(\tau) \in \{y, n\}]. \quad (9)$$

The gated count for branch verdict $v \in \{y, n\}$ is:

$$\tilde{N}_v(r) = \sum_{\tau \in \mathcal{C}(I)} a_r(\tau) \mathbb{I}[s_r(\tau) = v]. \quad (10)$$

Coverage. Coverage measures whether required base actions are exercised. We define:

$$B_{\text{att}} = \sum_{b \in \mathcal{B}} N_{\text{att}}(b). \quad (11)$$

$$B_{\text{all}} = \sum_{b \in \mathcal{B}} (N_y(b) + N_n(b) + N_u(b)). \quad (12)$$

Then:

$$\text{Coverage}(I) = \frac{B_{\text{att}}}{B_{\text{all}}}. \quad (13)$$

Achievement. Achievement measures correctness over applicable branch requirements. We define:

$$R_y = \sum_{r \in \mathcal{R}} \tilde{N}_y(r). \quad (14)$$

$$R_{\text{att}} = \sum_{r \in \mathcal{R}} (\tilde{N}_y(r) + \tilde{N}_n(r)). \quad (15)$$

Then:

$$\text{Achievement}(I) = \frac{R_y}{R_{\text{att}}}. \quad (16)$$

If $R_{\text{att}} = 0$, we report n/a .

Inactive and mutually exclusive branches. If a branch precondition is not triggered, its verdict is treated as u and excluded from the Achievement denominator. For mutually exclusive alternatives, only the branch activated by the observed dialogue contributes to correctness aggregation. This prevents models from being penalized for requirements that are logically inapplicable to the current interaction.

D Case Study

We provide an end-to-end case study using the *Museum* seed instruction. The example illustrates how FlexBench derives evaluation dimensions from a target instruction, how FlexEval respects conditional branches, and how COVERAGE and ACHIEVEMENT are computed from tri-valued judgments.

End-to-end pipeline. Table 9 summarizes the full pipeline for one dialogue. The seed instruction asks the assistant to help a user prepare a guided group-visit request by confirming booking intent, collecting visit and contact information, explaining the five-day advance-request rule, summarizing the request, and closing politely.

Dialogue excerpt. The following excerpt shows the relevant interaction:

Assistant: May I first confirm whether you are planning a guided visit for a group?

User: Yes, that's right.

Assistant: What is your preferred visit date?

User: We're planning for the 15th of next month.

Assistant: How many people are you expecting in your group?

User: About 30 people.

Assistant: Is your group from a school, company, or another type of organization?

User: We're a company.

Assistant: Guided group visits should be requested at least five days in advance. Since the 15th of next month is more than five days away, your date works well. May I have a contact name?

User: My name is Alex.

Assistant: I just need your email address for the museum staff to send you the booking confirmation.

User: My email is alex@company.com.

Assistant: You're requesting a group visit for about 30 people from a company on the 15th of next month, and we'll contact you at alex@company.com. Museum staff will review your request and reply by email with next steps. Thank you, and have a great day!

Verdict summary. Table 10 summarizes the dimension-level verdicts. Most applicable requirements are satisfied, but the final summary omits the contact name, yielding one no. Two conditional dimensions are marked unknown because their preconditions are not triggered.

Resulting scores. In this dialogue, 10 of the 12 dimensions are applicable, and 9 of those applicable dimensions are satisfied:

$$\text{Coverage} = \frac{10}{12} = 83.3\%. \quad (17)$$

$$\text{Achievement} = \frac{9}{10} = 90.0\%. \quad (18)$$

This example illustrates the role of tri-valued judgments: inactive branches are not penalized, while genuine omissions in applicable requirements are still counted as failures.

E External Decomposition Validation

This appendix provides additional details for the external decomposition validation in Section 5.3. We compare FlexBench dimensions against human-curated reference decompositions

Stage	Case-specific artifact	Interpretation
Seed instruction (excerpt)	Museum guided group-visit booking assistant. The assistant should confirm booking intent, collect the visit date, group size, organization type, contact name, and email, explain the five-day rule, summarize the request, and close politely.	The target instruction defines the workflow to be evaluated.
Segmentation (examples)	Prompt A segments the instruction into requirement units, such as: (i) ask for a contact name; (ii) ask for an email address for follow-up confirmation; (iii) if the preferred date is less than five days away, warn that confirmation may not be guaranteed and suggest an alternative date.	The instruction is decomposed into behaviorally relevant clauses before being converted into evaluation dimensions.
Dimension derivation (examples)	The selected segments become checkable dimensions: D8 asks whether the conversation requests a contact name; D9 asks whether it requests a follow-up email address; D7 checks the short-notice branch. FlexBench derives 12 dimensions in total.	The long instruction is transformed into conversation-observable criteria. The verification/modification step also refines D10 so that the final summary must include the contact name, which was collected earlier in the flow.
Logic grouping	D2 is gated by booking intent and applies only if the user is not planning a guided visit. D7 is gated by the visit date and applies only if the date is less than five days away. The remaining dimensions form the main collection and closure flow.	Conditional branches are represented explicitly instead of being treated as ordinary failures when inactive.
Dialogue rollout	The user confirms a guided visit, provides the 15th of next month, about 30 people, a company organization type, the contact name Alex, and the email alex@company.com.	The dialogue exercises the main booking flow but does not trigger the general-question branch or the short-notice branch.
Tri-valued judging	D1, D3–D9, and D11–D12 are judged yes; D10 is judged no; D2 and D7 are judged unknown.	Most applicable requirements are satisfied. The final summary omits the contact name, while inactive branches are not penalized.
Aggregation	10 of 12 dimensions are applicable, and 9 of the 10 applicable dimensions are satisfied.	COVERAGE measures exercised workflow requirements, while ACHIEVEMENT measures correctness over applicable requirements.

Table 9: End-to-end Museum case study. The table shows representative excerpts from the seed instruction, segmentation output, derived dimensions, logic grouping, dialogue rollout, tri-valued judgments, and aggregation. It is intended to illustrate the pipeline rather than exhaustively list all intermediate artifacts.

from the InFoBench Hard set and an instruction-following subset of ComplexBench. For ComplexBench, we remove translation-style prompts, long background-only items, and generic quality criteria that do not correspond to explicit instruction-following constraints.

E.1 Semantic Matching Metrics

Let D_F be the FlexBench dimension set and D_H be the human reference decomposition. For the main semantic recovery metric, we embed each item using sentence-transformers/all-MiniLM-L6-v2 with mean pooling, and compute cosine similarity $S(f, h)$ between FlexBench dimension f and human item h . Precision averages the best human match for each FlexBench dimension, while recall averages the best FlexBench

match for each human item:

$$P_{\cos} = |D_F|^{-1} \sum_{f \in D_F} \max_{h \in D_H} S(f, h), \quad (19)$$

$$R_{\cos} = |D_H|^{-1} \sum_{h \in D_H} \max_{f \in D_F} S(f, h). \quad (20)$$

We compute $F1_{\cos}$ as the harmonic mean of $P_{\cos}$ and $R_{\cos}$, and report granularity as $|D_F|/|D_H|$.

E.2 Preference-based Validation Details

We conduct a randomized A/B comparison between FlexBench decompositions and human reference decompositions. The judge compares the two decompositions with respect to evaluation suitability, atomicity/granularity, completeness, faithfulness, and non-redundancy. The A/B order is randomized to reduce position bias. Benchmark-level win rates are reported in the main paper; Table 12 shows criterion-level results.

Verdict	Dimensions	Reason
yes	D1, D3–D9, D11–D12	The assistant confirms the booking purpose, collects the required visit and contact information, explains the five-day rule, states that staff will review and reply by email, and closes politely.
no	D10	The final summary includes the preferred date, group size, organization type, and email address, but omits the contact name.
unknown	D2, D7	D2 is inactive because the user is planning a visit; D7 is inactive because the requested date is more than five days away.

Table 10: Verdict summary for the *Museum* case study.

Metric	Info	Complex	All
SBERT best-match	84.5	71.1	80.0
Sent-cos 1:1	81.6	69.0	77.4
BERTScore 1:1	77.1	52.4	68.8
BERT/CLS	93.8	90.6	92.7

Table 11: Additional F1 metrics for external decomposition validation. SBERT best-match is the main metric used in the paper. The 1:1 variants use strict one-to-one matching. BERT/CLS is included only for legacy compatibility, since raw BERT embedding spaces may suffer from anisotropy.

Criterion	Flex	Human	Tie
Evaluation suitability	74.8%	24.1%	1.1%
Atomicity / granularity	66.7%	24.3%	8.9%
Completeness	51.6%	16.7%	31.7%
Faithfulness	36.8%	9.4%	53.8%
Non-redundancy	26.8%	24.6%	48.7%

Table 12: Criterion-level preference results. FlexBench shows the largest advantage on evaluation suitability and atomicity/granularity, while faithfulness and non-redundancy are dominated by ties.

E.3 Low-similarity and Additional-dimension Adjudication

We further adjudicate low-similarity cases to determine whether apparent mismatches reflect true FlexBench failures, embedding artifacts, or granularity differences in the human references. Table 13 summarizes the adjudication categories.

We also inspect additional FlexBench dimensions that are not matched to any human reference item under the low-similarity criterion. As shown in Table 14, 21.6% of these additional dimensions are judged to be valid constraints omitted by the human references.

Category	Count	Rate
Embedding false negative	35	54.7%
Human generic quality	17	26.6%
Flex covers by group	7	10.9%
Flex valid, more atomic	3	4.7%
True FlexBench miss	2	3.1%

Table 13: Adjudication of 64 low-similarity cases. Only 3.1% are true FlexBench misses; most apparent mismatches are caused by embedding false negatives, generic human criteria, or granularity differences.

Benchmark	Cand.	Valid	Rate
InFoBench	16	3	18.8%
ComplexBench	81	18	22.2%
All	97	21	21.6%

Table 14: Validity of additional FlexBench dimensions not matched to human reference items. A non-trivial fraction captures valid instruction-following constraints omitted by the human decompositions.

F Additional Results and Robustness

This appendix provides additional results omitted from the main text, including Coverage diagnostics, rollout stability details, seed-reformulation results, and pipeline-backend robustness.

F.1 Coverage Diagnostics

FlexEval reports both COVERAGE and ACHIEVEMENT. Since Coverage is often close to saturation and less discriminative at the model-ranking level, we summarize it through the uncovered rate, $1 - \text{Coverage}$, which more directly highlights which workflows or dimensions are not exercised. Table 15 summarizes representative uncovered-rate patterns rather than reproducing the full 9×9 Coverage matrix. Most seeds are nearly saturated, while the largest uncovered rate occurs in the highly branched 3PL scenario, where many

Seed group	Mean 1 – Cov.	Range
3PL	73.7	72.5–74.8
Museum	11.9	10.9–12.4
Course	1.8	1.2–2.4
Other six seeds	0.2	0.0–1.8

Table 15: Uncovered-rate summary across tested models. Values are percentages. We report 1 – Coverage to emphasize workflow or branch exposure gaps.

Model	Mean	Std.	Min	Max
Claude Sonnet 4.5	52.60	1.54	50.48	53.85
GPT-4.1-mini	51.99	0.77	51.12	53.21
GPT-4.1	48.67	1.54	45.99	49.84
GPT-4.1-nano	47.19	1.13	46.31	48.88
GLM-4.5	42.56	0.79	41.51	43.59

Table 16: Achievement stability across five independently generated dialogue corpora for the *Travel Change* seed instruction. All values are percentages.

conditional paths are not expected to be triggered in every dialogue. This illustrates the intended role of Coverage: it diagnoses workflow exercise and branch exposure, whereas Achievement measures correctness once requirements become applicable.

Figure 5 provides a dimension-level diagnostic view for the representative *Course* seed. The main heatmap shows per-dimension Achievement across models, while the top row reports the mean uncovered rate 1 – Coverage for each dimension. Markers indicate dimensions whose Coverage falls below 95%. The figure shows that most dimensions are exercised, but a few localized coverage gaps co-exist with substantial differences in Achievement. Thus, FlexEval provides two complementary diagnostic signals: Coverage reveals which workflow requirements are reached, and Achievement reveals how well models satisfy the applicable requirements.

F.2 Rollout Stability Details

For the rollout-stability analysis, we use the *Travel Change* seed instruction and independently generate five dialogue corpora for each model, with 48 dialogues per corpus. Table 16 reports model-level Achievement statistics across the five corpora.

Table 17 summarizes deviation statistics. Coverage is nearly saturated, while Achievement varies modestly across independent rollouts.

Metric	Bias	95% LoA	Mean $ \Delta $
Coverage	0.00	[−0.02, 0.02]	0.01
Achievement	0.00	[−2.15, 2.15]	0.88

Table 17: Deviation statistics for rollout stability. Values are reported in percentage points. LoA denotes limits of agreement.

F.3 Ranking Uncertainty

For each seed, we identify the point-estimate leader and runner-up in Table 3. We resample the 48 dialogue-level outcomes with replacement and recompute the best-vs. runner-up Achievement difference for 10,000 bootstrap replicates. The interval measures dialogue-level sampling uncertainty under the realized generation setup; it does not measure variation from regenerating an entirely new dialogue corpus.

The bootstrap intervals include zero for eight of the nine seeds. Course is the only seed whose best-vs. runner-up interval is strictly positive. These results support interpreting the main table in terms of point-estimate leaders and statistically close top groups rather than definitive winners for every seed.

F.4 Computational Cost and Scaling

Table 19 summarizes token usage for the main 9-seed, 9-model, and 48-dialogue-per-cell experiment.

The main experiment contains $9 \times 48 \times 140 = 60,480$ dimension–dialogue judgments. The judging stage scales linearly with the number of models, dialogues, and derived dimensions. Since individual judgments are independent, the judging stage can be parallelized.

F.5 Seed Reformulation Details

We evaluate sensitivity to reformulations of the *Travel Change* seed instruction while keeping the dialogue corpus fixed. Table 20 reports the similarity between the reformulated dimensions and the original dimensions.

Table 21 reports the absolute Achievement scores under each reformulation. Format-only changes have limited impact, lexical rewriting lowers scores consistently, and compression raises scores because it weakens or omits some fine-grained conditions, making the evaluation target looser.

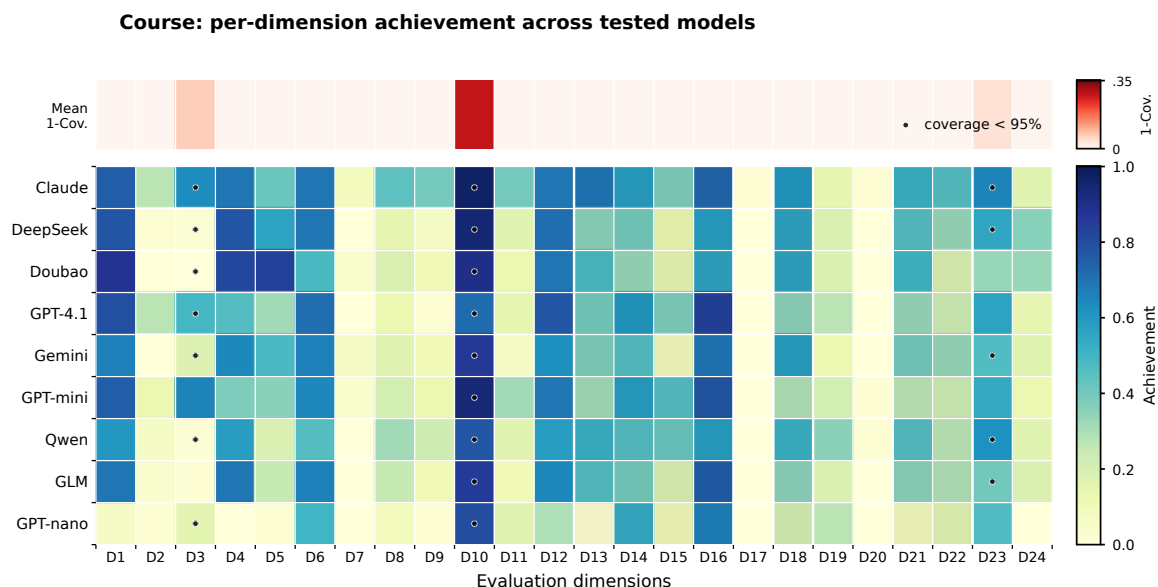


Figure 5: Dimension-level diagnostics for the *Course* seed. The heatmap reports per-dimension Achievement across tested models. The top row shows mean uncovered rate $1 - \text{Coverage}$, and markers indicate dimensions with Coverage below 95%. This visualization illustrates how Coverage and Achievement jointly localize workflow-exposure gaps and conditional-correctness failures.

Seed	Best	Runner-up	Gap (pp)	95% CI (pp)	$P(\text{best} > \text{runner-up})$
FastRun	Claude	GPT-4.1-nano	0.3	$[-4.9, 5.7]$	0.52
3PL	Gemini	Claude	2.0	$[-8.9, 13.1]$	0.64
Course	Claude	GPT-4.1	10.0	$[0.3, 19.6]$	0.98
Exam	Doubao	GPT-4.1	1.9	$[-6.7, 10.2]$	0.65
IT	GPT-4.1	Gemini	0.1	$[-5.6, 5.7]$	0.51
2FA	Doubao	DeepSeek	0.1	$[-5.7, 5.9]$	0.51
Museum	Claude	Doubao	2.9	$[-9.3, 14.7]$	0.67
Travel	Qwen	GPT-4.1-mini	1.1	$[-3.2, 5.6]$	0.68
ConfReg	Claude	Qwen	1.0	$[-3.8, 5.8]$	0.65

Table 18: Dialogue-level bootstrap uncertainty for the point-estimate best-vs. runner-up Achievement difference in each seed. Gaps and confidence intervals are reported in percentage points. Model names are shortened only in this table.

Table 22 summarizes rank stability. The format-only variant preserves the original ranking exactly, the lexical rewrite changes only one pairwise ordering, and the compressed variant causes larger but still limited rank changes among closely performing models.

F.6 Pipeline-Backend Robustness

To test whether FlexBench/FlexEval depends on a single LLM backend, we rerun the construction/evaluation pipeline with GPT-5.4 and Gemini 3.1 Pro Preview, in addition to the main Claude Opus 4.6 pipeline. The correlations in Table 23 are computed over model-level average Achievement across the nine tested models.

Figure 6 provides a more detailed view. Panel (a) reports model-level average Achievement and ranks under each pipeline backend. Panel (b) com-

pares alternative pipelines against the main Claude Opus 4.6 pipeline. Panel (c) shows the backend-induced score range for each tested model. Although absolute scores vary across backends, the relative model conclusions are largely preserved.

Stage	Unit	Avg. input	Avg. output	Total tokens
Dimension derivation	seed	8,459.6	3,110.2	104,128
Logic extraction	seed	1,757.1	1,090.0	25,624
Dialogue generation: assistant	seed-model	214,889.9	36,370.5	20,352,089
Dialogue generation: simulator	seed-model	306,729.7	25,306.0	26,894,892
Completion detector	seed-model	105,520.8	10,757.6	9,418,547
FlexEval judging	dimension-dialogue	1,112.2	158.9	76,878,314
Total	—	—	—	133,673,594

Table 19: Token usage and computational scaling for the main experiment. Average input and output values are reported per unit.

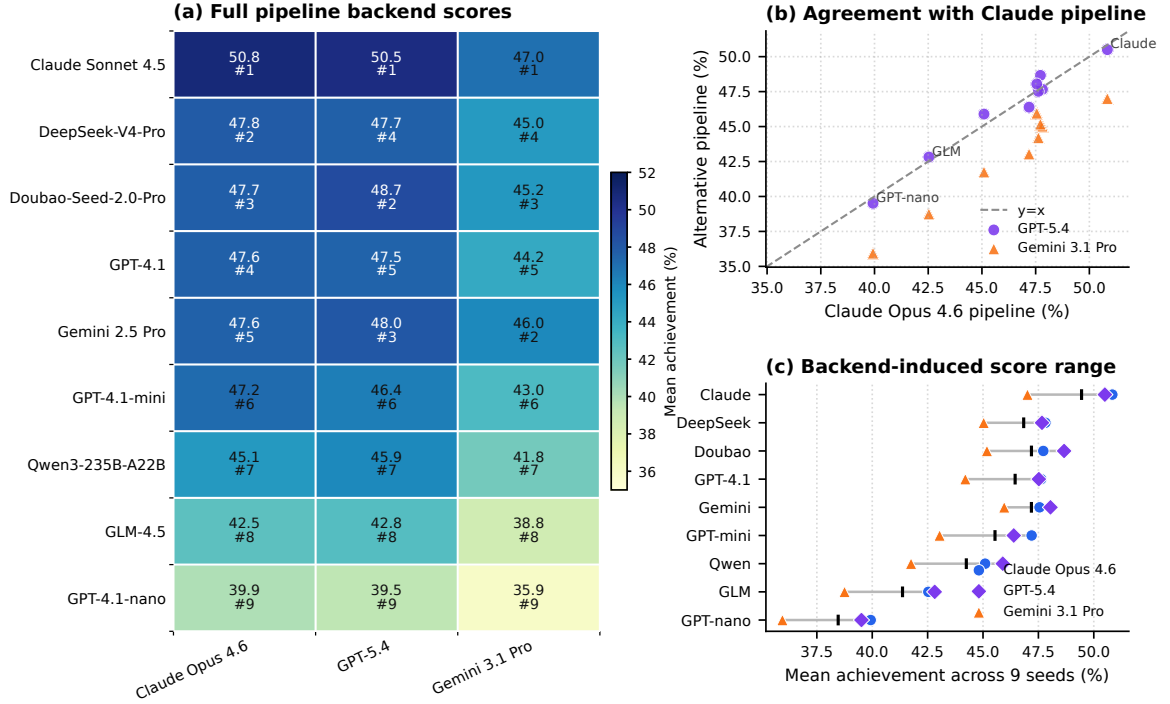


Figure 6: Pipeline-backend robustness across the nine tested models. Panel (a) shows model-level average Achievement and ranks under the Claude Opus 4.6, GPT-5.4, and Gemini 3.1 Pro Preview pipelines. Panel (b) shows agreement between the alternative pipelines and the main Claude Opus 4.6 pipeline. Panel (c) shows the backend-induced score range for each model.

Variant	#Dims	F1 vs. orig.	Gran.
Format-only	13	98.0	1.00
Lexical rewrite	13	89.4	1.00
Compressed flow	13	92.8	1.00

Table 20: Dimension similarity between seed reformulations and the original *Travel Change* seed. F1 is computed using SBERT best-match alignment, and Gran. denotes the granularity ratio relative to the original dimension set.

Variant	Claude	GPT-4.1	Gemini	GLM
Original	51.44	49.12	49.68	42.95
Format-only	54.33	49.92	52.88	44.39
Lexical rewrite	46.93	45.15	44.77	37.10
Compressed flow	61.06	61.06	60.83	51.60

Table 21: Achievement scores under seed reformulations for the *Travel Change* seed. All values are percentages.

Variant	Rank change	Pairwise agree.
Format-only	Same ranking	6/6
Lexical rewrite	One pair swapped	5/6
Compressed flow	Top models closer	4/6

Table 22: Rank stability under seed reformulation. Pairwise agreement is computed over the four models included in the reformulation analysis.

Pipeline pair	Spearman	Pearson
Claude vs. GPT-5.4	0.917	0.984
Claude vs. Gemini	0.883	0.976
GPT-5.4 vs. Gemini	0.983	0.986

Table 23: Correlation of model-level average Achievement across pipeline backends. Spearman measures rank consistency, and Pearson measures score-level correlation.