RANKJUDGE: A MULTI-TURN LLM-AS-A-JUDGE SYNTHETIC BENCHMARK GENERATOR

Anonymous authors
Paper under double-blind review

ABSTRACT

As interactive LLM-based applications are created and refined, model developers need to evaluate the quality of generated text along many possible axes. LLM judges make this evaluation scalable, but the judges themselves must also be evaluated to ensure that their assessments are reliable. Benchmarks built around isolated responses do not test failures that depend on conversation history, and fixed benchmarks can provide less differentiation among increasingly capable models. Testing judges for such interactions requires multi-turn benchmarks that can be renewed as models and applications evolve. We introduce RankJudge, an end-to-end generator of document-grounded, multi-turn judge benchmarks. Its fully synthetic construction produces labeled conversation pairs through controlled generation and automated verification, without per-item human annotation. A judge-versus-problem formulation then jointly estimates judge ability and pair difficulty using the Bradley–Terry model. These estimates support difficulty-based benchmark curation and judge ranking under incomplete evaluation coverage. We instantiate RankJudge in machine learning, biomedicine, and finance, and evaluate 21 LLM judges, revealing substantial differences in judge ability across models. RankJudge thus provides a reusable construction and calibration pipeline, rather than only a fixed evaluation set.

1 INTRODUCTION

Large language models are increasingly evaluated by other large language models (LLMs). Pairwise judging (Zheng et al., 2023; Chiang et al., 2024) has become the dominant scalable substitute for human preference collection. As judge models are now used to score training data, gate releases, and rank checkpoints, judge *quality* has itself become a central assumption. A leaderboard built on a weak judge may silently reward the wrong behavior.

Several established judge benchmarks, including LLMBar (Zeng et al., 2024) and JudgeBench (Tan et al., 2025), compare responses to standalone prompts. These tests can be demanding, but they do not directly test whether judges detect failures that depend on conversation history. Failure modes that matter in deployment, such as a later turn contradicting an earlier one or a content-level constraint silently dropping after several turns (Cemri et al., 2025; Laban et al., 2026), simply cannot surface in the single-turn setting. We therefore focus on document-grounded, multi-turn conversations, where judges must assess both support from the reference material and consistency across turns.

A fixed collection of such conversations is still not enough. As models improve and applications change, judge evaluation must be able to introduce fresh, relevant challenges (Zhu et al., 2024; Li et al., 2025). Renewing this evaluation requires more than generating additional dialogue: the pipeline must supply labeled comparisons and determine which generated problems are useful for evaluating the current judges. Rebuilding and labeling each benchmark by hand does not scale.

We introduce RANKJUDGE, an end-to-end benchmark generator built around two complementary components (Figure 1). First, fully synthetic construction creates pairs of conversations from reference documents, with controlled assistant failures and varied user behaviors. The generation plan supplies the intended labels, and automated verification can check whether the conversations realize that plan. This makes fresh labeled comparisons available without repeating per-item manual writing or annotation. The corpus, generation model, and conditioning choices can be changed when the evaluation setting changes.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

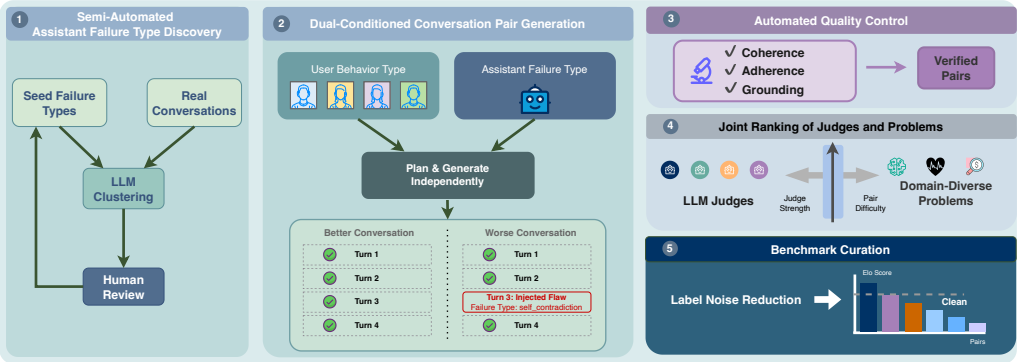


Figure 1: Overview of RANKJUDGE, a benchmark *generator* for multi-turn judge evaluation. Synthetic construction supplies labeled comparisons; joint judge-versus-problem modeling supports ranking and difficulty-based curation.

Second, we treat each judgment as a *judge-versus-problem* outcome and jointly fit judge ability and problem difficulty with the Bradley–Terry model. The same observations therefore inform both the leaderboard and the benchmark itself. Problem ratings support difficulty-based curation, while the joint fit accommodates incomplete judge-item coverage. Together, these components make benchmark renewal practical: generate new comparisons, estimate their difficulty, and curate an evaluation slice for the current judge population.

We instantiate this pipeline in machine learning, biomedicine, and finance and evaluate 21 judges. A human audit of the generation process and a held-out fine-tuning study inform difficulty-based curation. The resulting leaderboard spans nearly 1200 Elo points, showing that current LLMs have greatly different ability to judge multi-turn conversations. Our methodological contributions are:

- An end-to-end synthetic construction pipeline for renewable, document-grounded, multi-turn judge benchmarks, combining controlled generation with automated verification.
- A joint judge-versus-problem formulation that uses evaluation outcomes to rank judges, estimate problem difficulty, and support curation and incomplete evaluation coverage.

2 RELATED WORK

Multi-turn LLM Benchmarks. LLM evaluation has shifted from single-turn benchmarks like MMLU (Hendrycks et al., 2021) and GSM8K (Cobbe et al., 2021), which miss the user-model-environment dynamics that drive real-world utility (Wang et al., 2024; Deshpande et al., 2025), to multi-turn frameworks (Zheng et al., 2023; Kwan et al., 2024; Fan et al., 2026; Eisenstein et al., 2026) that probe correctness, helpfulness, and interactive patterns (Li et al., 2026b). A consistent finding emerges across these works: single-turn ability does not transfer to multi-turn success (Wang et al., 2024), and frontier models degrade sharply across turns due to compounding unreliability (Laban et al., 2026). These dynamics motivate our focus on multi-turn, reference-grounded conversations as the setting in which judge quality must itself be stress-tested.

LLM-as-a-Judge. Reward models are crucial for aligning and improving the capabilities of LLMs (Ouyang et al., 2022; Christiano et al., 2017). The traditional scalar reward model (Stiennon et al., 2020) gives a single "verdict" indicating the response quality. However, scalar models suffer from certain limitations: for example, they are vulnerable to hacking (Xu et al., 2025b) and lack the ability to localize or categorize specific errors. LLMs have demonstrated strong capability to mimic human reasoning and evaluate inputs based on predefined criteria while being scalable and effective. The concept of LLM-as-a-judge (Zheng et al., 2023; Wang et al., 2023; Liu et al., 2023) has become widely used for tasks like providing rich reward signals for LLM alignment (Lee et al., 2024), producing chain-of-thought (CoT) reasoning along with a final judgment as evaluators (Kim et al., 2024; Saha et al., 2025), and data annotation (Luo et al., 2025; Chen et al., 2024). These judges can be implemented either via direct prompting of general-purpose LLMs (Zheng et al., 2023; Wang et al., 2026b) or as specialized fine-tuned evaluators (Whitehouse et al., 2026; Chen et al., 2026a). Existing frameworks typically adopt either pointwise or pairwise evaluation protocols. Pointwise methods (Liu

et al., 2023; Kim et al., 2024) score responses independently, while pairwise methods (Zheng et al., 2023; Whitehouse et al., 2026) compare responses to predict relative preferences.

Benchmarking LLM-as-a-Judge. MT-Bench (Zheng et al., 2023; Bai et al., 2024) helped establish LLM-as-a-judge evaluation for chat assistants by reporting agreement with humans. Initial works on meta-evaluation of judges focused on single-turn settings: RewardBench (Lambert et al., 2025) curates preference pairs for reward models, LLMBar (Zeng et al., 2024) uses natural and adversarial pairwise examples, JudgeBench (Tan et al., 2025) converts factuality and correctness datasets into benchmarks for meta-evaluation, and ContextualJudgeBench (Xu et al., 2025a) grounds evaluation in external documents. DHP (Wang et al., 2025b) measures natural language generation evaluation capabilities using perturbations while ReIFE (Liu et al., 2025) varies LLMs, protocols and datasets. JuStRank (Gera et al., 2025) studies judges through system-level ranking agreement with human rankings, and Judge Arena (Dai et al., 2024) crowdsources evaluations of anonymized judges.

Table 1: Comparison of LLM judge evaluation benchmarks. Turn: S = single, M = multi. Drv. (derived evaluation target): ✓ = no human preference labeling, △ = some manual curation, – = completely dependent on human judgments. User: controlled user behavior. Doc.: document grounding. Diag.: diagnostic labels. Pipe.: automated benchmark pipeline (△ = partial). Eval.: Acc. = item level accuracy or agreement; Disc. = perturbation discernment; Rank = judge ranking.

	Turn	Drv.	User	Doc.	Diag.	Pipe.	Eval.
LLMBar	S	–	–	–	–	–	Acc.
JudgeBench	S	✓	–	–	–	△	Acc.
ContextualJudgeBench	S	△	–	✓	△	△	Acc.
DHP	S	✓	–	–	△	✓	Disc.
ReIFE	S	–	–	–	–	–	Acc.
JuStRank	S	–	–	–	–	–	Rank
Judge Arena	S	–	–	–	–	–	Rank
MEDAL	M	△	△	–	△	✓	Acc.
RankJudge	M	✓	✓	✓	✓	✓	Rank

Other works study LLM-as-a-Judge for code evaluation (Wang et al., 2025a) and evaluator adversarial robustness (Li et al., 2026a), positional bias (Shi et al., 2025), and fairness (Zhang et al., 2023). MEDAL (Mendonça et al., 2026) is closest to our setting since it generates multilingual multi-turn dialogues using a multi-agent pipeline and automates labeling with `gpt-4.1`, followed by filtering with human curation for the final benchmark. MetaJudge (Wang et al., 2026a) and RM-NLHF (Wang et al., 2026c) address rationale-aware evaluation and supervision, complementary to our focus on benchmark construction and calibration. Table 1 compares benchmark settings.

Automated Benchmark Construction. DyVal (Zhu et al., 2024) generates reasoning tasks with controllable complexity, while AutoBencher (Li et al., 2025) constructs benchmarks around criteria such as difficulty and topic salience. BenchMaker (Yuan et al., 2026) combines label refinement, difficulty control, and diversity measures in a general-purpose benchmark generator. Another line of work constructs grounded questions or errors to test models and evaluators (Thorne et al., 2018; Alberti et al., 2019; Kryscinski et al., 2020): HaluEval (Li et al., 2023) uses generated hallucinations alongside human-annotated responses, while SummEdits (Laban et al., 2023) uses controlled summary edits with human consistency labels. Item response theory jointly estimates model ability and item difficulty to weight or subsample benchmark items (Lalor et al., 2016; Rodriguez et al., 2021; Maia Polo et al., 2024). RANKJUDGE brings this joint view to multi-turn judge evaluation, where the fitted difficulties of generated conversation pairs drive curation and support partial evaluation coverage.

3 METHODOLOGY

RANKJUDGE first generates and verifies labeled conversation pairs from a reference corpus. It then evaluates judges on these pairs and jointly estimates judge ability and problem difficulty. The difficulty estimates guide curation of an evaluation slice. Generation can be rerun with new documents, conditioning choices, or models, and the resulting pool can be calibrated against the current judges.

Let $\mathcal{T} = \{t_1, \ldots, t_K\}$ denote a taxonomy of K assistant failure types. A multi-turn conversation $C = (u_1, a_1, \ldots, u_R, a_R)$ consists of R turns, each a (user, assistant) message pair. Each benchmark item is a tuple $(C_A, C_B, y^\star, r^\star, t^\star)$ in which C_A and C_B are two conversations grounded in the same reference documents, $y^\star \in \{A, B\}$ identifies the intended better conversation, $r^\star \in \{1, \ldots, R\}$ is the designated turn of the single injected flaw in the worse conversation, and $t^\star \in \mathcal{T}$ is its predetermined failure category.

Table 2: Multi-turn assistant-failure taxonomy. In each conversation pair, one conversation exhibits one flaw from the available failure types in exactly one turn; the other exhibits no clear flaw throughout. Failure types (left) are the ground-truth labels $t^\star$.

Failure Type	Behavior
`self_contradiction`	**Flaw.** A later turn contradicts an earlier correct claim by shifting a qualifier or reframing. **Benign.** Consistent scope and qualifiers across turns; flags genuine changes in position.
`evasion`	**Flaw.** Hedges or drifts around the central claim, citing peripheral details without the figures or terms needed to verify or act. **Benign.** Answers the question directly with the exact figures and terms required.
`disorganized`	**Flaw.** Buries the key answer in a muddled structure, mixing high- and low-level points so correct content is costly to extract. **Benign.** Leads with the key point and arranges supporting detail in a scannable structure.
`fabricated_answer`	**Flaw.** When a question is unanswerable from the source, invents a confident answer instead of admitting insufficient information. **Benign.** Calibrated to the source: confident when supported, explicit about limits otherwise.
`instruction_forgetting`	**Flaw.** A directive set early (scope, audience, what to skip) is explicitly violated in a later turn while the answer remains plausible. **Benign.** Tracks the early directive across turns, surfacing genuine conflicts rather than deviating.
`no_clarification`	**Flaw.** On an ambiguous question, picks one interpretation and answers confidently rather than asking the user to clarify. **Benign.** Clarifies on truly ambiguous questions; answers unambiguous ones directly.
`unnecessary_refusal`	**Flaw.** Refuses on a question that is answerable from the source, citing insufficient information when the answer is available. **Benign.** Answers confidently from available material; hedges only when genuinely out of scope.

3.1 Benchmark Construction

Failure Taxonomy. The taxonomy defines which failures the generator should target. We seed it from prior work on dialogue evaluation and assistant failures (Cemri et al., 2025; Laban et al., 2026; Kartáč et al., 2026). We then prompt `gemini-3.1-pro` (Google DeepMind, 2026) to examine samples from MT-Bench (Zheng et al., 2023) and MT-Bench 101 (Bai et al., 2024) and propose missing categories. This step grounds category selection in observed dialogue behavior. The authors then reviewed candidate failure categories and merged or dropped overlapping categories.

We focus on failures that depend on conversational context rather than coarse single-turn categories (Leung et al., 2026). For example, contradicting an earlier answer is `self_contradiction`, while dropping a prior user constraint is `instruction_forgetting`. Table 2 summarizes the seven selected types and their benign counterparts. Full definitions and the discovery prompt appear in Appendices B.4 and B.3. Once this specification is fixed, item generation and labeling are automated.

Dual Conditioned Conversation Pair Generation. Each pair is sampled under two independent conditions: an *assistant failure type* and a *user behavior type*. The first specifies the intended failure, while the second varies how users engage with the reference material. User behaviors span seven archetypes: `focused`, `integrative`, `scattered`, `skeptical`, `misinformed`, `exploratory`, and `underspecified` (Appendix B.5). Before generation, we designate the better branch, the flawed turn in the worse branch, and the failure category. These choices supply the intended labels $(y^\star, r^\star, t^\star)$ as a by-product of construction.

Each pair of better and worse conversations is produced by two separate sets of generation calls that share reference documents. Past works have created negative examples by simply injecting errors into existing text (Li et al., 2023; Zeng et al., 2024; Wang et al., 2025a; Kong et al., 2026). However, when comparing two alternate conversations, if the injected error is the only change, the judge can isolate this difference rather than making a complete assessment of quality. Independently sampling two conversations is also insufficient; if only the bad version faced situations where the target failure could surface, judges could again shortcut a holistic comparison by pattern-matching on question types. We therefore generate the branches separately while staging comparable conversational pressures in both. The worse branch is instructed to exhibit the target failure; the better branch is conditioned on its benign counterpart (Table 2). For instance, when the failure type is `fabricated_answer`, the user in the better conversation still asks an out-of-scope question, but the assistant is instructed to explicitly state the limits of its knowledge. This design aims to reduce shortcuts based only on question type or a single edited span.

Both sets of generation calls follow a turn-by-turn *blueprint*. Each turn specifies what the user's question should be about, and the chunk of the reference document the assistant will need to draw on.

Algorithm 1 Data generation and verification

1: **for** each of N_{cand} candidate samples **do**
2: Sample failure type $t^\star \in \mathcal{T}$, user behavior, reference documents, verdict $y^\star \in \{A, B\}$, flawed turn $r^\star \in \{1, \ldots, R\}$
3: Generate paired blueprints conditioned on the sampled inputs
4: If **coherence** check of the blueprints fails, skip
5: Generate paired conversations (C_A, C_B) from the blueprints
6: If **adherence** check of (C_A, C_B) to blueprints fails, skip
7: If **grounding** check of all atomic claims fails, skip
8: Retain $(C_A, C_B, y^\star, r^\star, t^\star)$
9: **end for**

The blueprint for the worse conversation additionally commits to a `bad_round_index` and a sketch of how the selected flaw should manifest. To keep the comparison non-trivial, the blueprint imposes requirements that the flawed turn must maintain the same tone and length as other turns, while lexical announcements of the kind "stepping outside the scope for a moment" are disallowed. The simulated user reacts to earlier answers by following up, pushing back, or redirecting, but must not explicitly name the flaw. We limit conversations to span 3–7 turns, although this parameter can be tuned as desired. In contrast, we fix the flaw count at one to keep the localization target unique. We remove ordering bias by randomizing which conversation in the pair (A/B) is assigned as worse. The full generation prompts are provided in Appendix B.6.

Automated Quality Control. A synthetic pair is only useful to the benchmark if the targeted weakness actually surfaces, appears only in the declared turn, and the rest of the content is free of clear failures. A three-layer verification cascade checks this specification (Cobbe et al., 2021; Saunders et al., 2022; Lightman et al., 2024). Generation and verification use the same model in separate calls. The verifier receives both the reference documents and intended labels, making its task different from the open-ended comparison given to evaluated judges. A pair is retained only if it passes all three checks.

The *coherence* check tests the blueprint against the sampled user behavior, failure specification, and reference material. It flags conflicts in the planned user intent, assistant focus, or failure location before the conversations are written. The *adherence* check tests whether the generated conversations follow that blueprint: the better branch should remain benign, and the worse branch should contain the target failure only at the designated turn. A conversation pair fails the adherence check if the user deviates from the specified behavior, the failure drifts to a different turn, or if multiple flaws are present. The *grounding* check extracts atomic factual claims from each assistant turn and checks them against the source; every turn except the designated flawed turn must pass. The intended flaw at that turn is checked by adherence. These checks reduce construction errors but do not guarantee label validity. Section 3.4 reports consistency checks and validation of the entire process.

Pass rates appear in Table 3 and prompts in Appendix B.7.

3.2 Evaluation Protocol

A judge returns $J(C_A, C_B) = (\hat{y}, \hat{r}, \hat{t})$, its prediction of the better conversation, the flawed turn, and its failure category. We record a judgment as correct only when all three fields match the labels used to construct the pair:

$$c = \mathbf{1}\left[\hat{y} = y^\star \wedge \hat{r} = r^\star \wedge \hat{t} = t^\star\right] \in \{0, 1\}. \tag{1}$$

The turn and type fields provide a limited check that the judge identifies the designated failure. They are inexpensive to score because their targets already exist in the generation plan. Evaluating free-form reasoning would instead require human rationale annotation or another evaluator, introducing an additional layer of judgment into the evaluation of judges. Our pipeline avoids this entirely.

3.3 Joint Ranking of Judges and Conversation Pairs

Let $\mathcal{D} = \{(j_m, q_n, c_{m,n})\}_{m=1,n=1}^{M\ N}$ contain the observed outcomes c for judges $j_m \in \mathcal{J} = \{1, \ldots, M\}$ and conversation pairs $q_n \in \mathcal{Q} = \{1, \ldots, N\}$. We rank judges against each other based on their correctness, and establish difficulty levels for conversation pairs by how often they fool judges. Both ranks and difficulties are produced through an arena-like match scoring system.

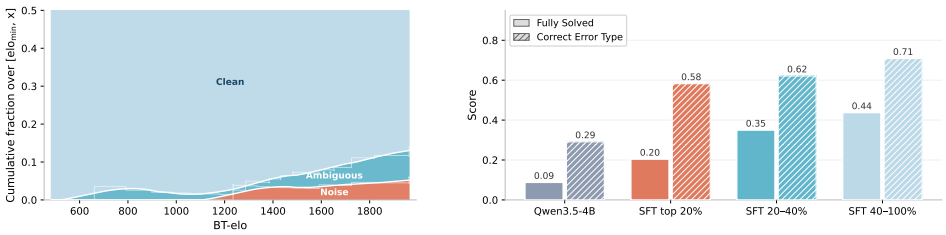


Figure 2: Evidence informing difficulty-based curation. Left: cumulative fractions of audited pairs labeled ambiguous or noisy among those with Elo at most x. Right: fine-tuning `qwen3.5-4b` on the top-20% difficulty band underperforms training on lower bands. These analyses assess the process; they are not required steps for generating new items.

Unlike an LLM-versus-LLM arena (Chiang et al., 2024), the two sides of each match are a judge and a benchmark item $(C_A, C_B, y^\star, r^\star, t^\star)$. Joint fitting assigns both judge ability and item difficulty from the same outcomes. This supports two uses beyond a simple accuracy report: curating the generated pool by difficulty, and fitting a leaderboard when judge-item coverage is incomplete.

Under the Bradley–Terry scoring model (BT), which in this judge-versus-item form is the Rasch model of item response theory (Rodriguez et al., 2021), each judge and pair has a positive strength $\theta_i > 0, i \in \mathcal{J} \cup \mathcal{Q}$. The probability of a correct outcome is $\Pr(c = 1 \mid j, q) = \theta_j/(\theta_j + \theta_q)$. We fit by maximum likelihood using minorization–maximization with mean-normalization (Hunter & Li, 2005), then map strengths to Elo via $400 \log_{10}$. With full coverage, the BT judge ordering agrees with raw correct counts. Its role here is to estimate item difficulty alongside judge ability and to fit incomplete observations on a connected judge-item graph. Ratings are relative to the fitted population. Full equations and confidence intervals appear in Appendix B.1. Empirical Interaction Propagation (EIP) (Hu et al., 2026), a random-walk alternative, yields a closely matching ordering (Spearman $\rho \geq 0.94$), so the joint ranking does not hinge on the choice of estimator (Appendix B.2).

Unanimous failure of judges on an item is not perfect evidence that the item is mislabeled; however, it does mean the item is useless for ranking judges. Hence, before fitting we remove pairs on which all judges are correct or all are incorrect.

3.4 Benchmark Curation

Fully synthetic construction of benchmark items can supply new comparisons without manual annotation, but imperfect automated verification may result in ambiguous or mislabeled items. Joint modeling makes the benchmark's quality inspectable through difficulty estimates. Across the benchmarks we generate for machine learning, biomedicine, and finance (Section 4), our automated checks retain 821 of 1200 items. Removing unanimous outcomes leaves 703 informative items.

Validation of the Generation Process. We manually audited the generated benchmarks as an experiment to validate the generation process. It is not strictly required as a stage of benchmark generation or a requirement for labeling new items. We inspected 120 of the 703 informative items, assigning `clean`, `ambiguous`, or `noise` labels for the quality of items. An item was `ambiguous` if the failure could fit more than one taxonomy category, and `noise` if it did not match the intended labels, for example because another flaw was present. The audit used the construction metadata and judge outputs, as documented in Appendix C. Figure 2 (left) shows problematic audit labels concentrated in the high-difficulty tail. Within the audited sample, 12 of the 30 highest-Elo pairs were marked ambiguous or noisy, compared with 5 of the remaining 90 (Fisher's exact test, $p < 10^{-4}$). This concentration of lower-quality items motivates filtering out the tail.

Fine-tuning Across Difficulty Bands. We split the informative items not used in the human audit into three Elo bands (top 20%, 20–40%, and the remaining 60%), sample an equal number of pairs from each, and fine-tune `qwen3.5-4b` (Qwen Team, 2026) with LoRA (Hu et al., 2022) under an identical recipe per band (full hyperparameters in Appendix B.9). The `clean` pairs from the audit are held out for evaluation. Joint correctness of the fine-tuned judge improves monotonically as the training band moves down the ranking, and the bottom band achieves more than twice the score of the top band (Figure 2, right). This result further supports avoiding the highest-difficulty band.

Table 3: Pair counts after verification and curation, by domain. Coherence, adherence, and grounding are automated generation checks. Uninformative removes pairs with unanimous correctness outcomes. Human records 17 known issues excluded after our manual validation, but not a required generation step. Top 5% trims the highest-difficulty pairs by Elo.

Domain	Generated	Coherence	Adherence	Grounding	Uninformative	Human	Top 5%	Survival
ML	400	397 (−3)	342 (−55)	304 (−38)	263 (−41)	255 (−8)	240 (−15)	60.0%
MED	400	400 (−0)	344 (−56)	243 (−101)	208 (−35)	202 (−6)	194 (−8)	48.5%
FIN	400	397 (−3)	345 (−52)	274 (−71)	232 (−42)	229 (−3)	218 (−11)	54.5%
All	1200	1194 (−6)	1031 (−163)	821 (−210)	703 (−118)	686 (−17)	652 (−34)	54.3%

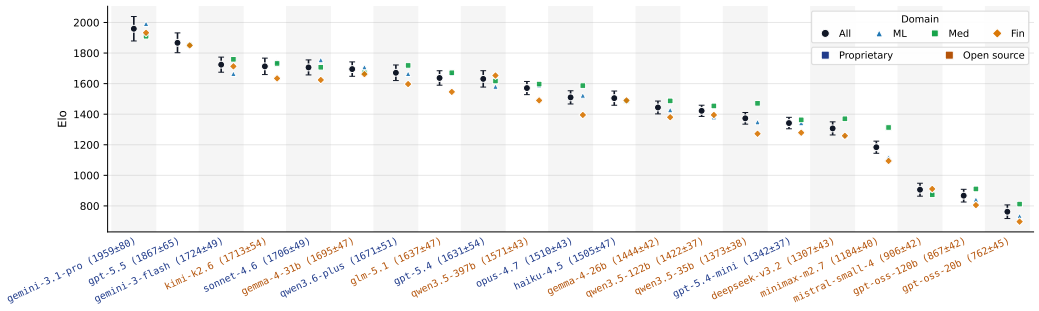


Figure 3: Elo scores of 21 judges on the combined dataset. Black circles give the combined Elo with 95% CI; colored markers show per-domain Elo scores. Tick-label color denotes proprietary (blue) vs. open-source (orange) judges. Table 4 in Appendix A.1 gives detailed per-domain results.

For the published evaluation slice, we trim the top 5% of pairs by Elo as an empirical curation choice. Since the audit had already identified 17 problematic pairs, we also excluded these known issues from the published slice. Table 3 tracks the full construction and curation process, which leaves 652 pairs. Appendix A.4 breaks down retention by failure type and user behavior. Pair difficulty also varies across failure types (Appendix A.2), providing a more detailed view of the generated items beyond aggregate judge accuracy.

Because item difficulty is relative to the judge population, curation can be recomputed when that population changes. The same construction and fitting steps can then be applied to new candidate pools automatically.

4 EXPERIMENTS

Experimental Settings. Our pipeline is domain-agnostic, generating a benchmark of multi-turn conversation pairs through the same generation, verification, and ranking components from any corpus of knowledge-intensive context. We instantiate it on three domains, each viewable as a standalone benchmark: machine learning (CS papers from RPC-Bench, Chen et al., 2026b; custom license), biomedicine (PubMedQA, Jin et al., 2019; MIT license), and finance (S&P 500 10-K filings, Jerry Loh, 2026; MIT license). Per pair, the user behavior and assistant failure types are sampled independently and uniformly to form a balanced stress test. All generator, verifier, and judge calls are routed through OpenRouter (OpenRouter, 2026) with `gpt-5.5` used for generation and verification. Every judge in our registry supports thinking, which, like sampling parameters, we leave at the provider's default. We exclude judge-specialized fine-tuned evaluators to keep the focus on general-purpose LLMs, though judge-specific fine-tuning can improve performance on static problems (Whitehouse et al., 2026; Huang et al., 2026).

RankJudge Leaderboard. Figure 3 reports Elo ratings for 21 judges. The scale spans nearly 1200 Elo points from `gemini-3.1-pro` (1959) to `gpt-oss-20b` (762), implying that the benchmark separates judges across a wide range of capability. `gemini-3.1-pro` leads with a 92-point margin over the next judge `gpt-5.5` (also the generator), while the per-domain rankings broadly preserve the order, indicating that judge skill is largely consistent across domains. The leaderboard shows a trend that

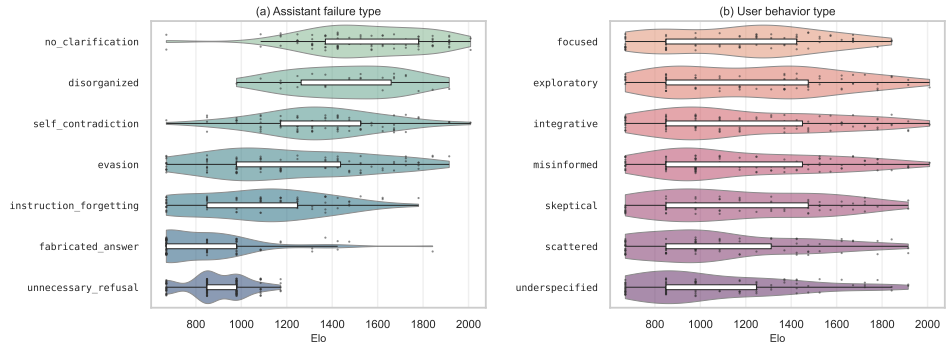


Figure 4: Per-pair Elo grouped by (a) assistant failure type and (b) user behavior. Categories are sorted by median Elo. Violins show the distribution, boxes the interquartile range, red lines the median, and points individual pairs.

proprietary LLMs are still stronger than open-source models, although `kimi-k2.6` and `gemma-4-31b` are competitive with several proprietary frontier models, while `opus-4.7` underperforms, indicating that strong generation capability does not directly translate to strong judging.

Since `gpt-5.5` was also used for generation and verification, Appendix A.3 examines generator dependence by regenerating the conversation pairs with `opus-4.7` as the generator. Each generator model rises in rank on its own pairs, but the rankings of the remaining 19 judges are preserved almost identically (Spearman $\rho = 0.979$, Kendall $\tau = 0.906$). `gemini-3.1-pro` in particular ranks first under either generator, so the top of the leaderboard does not depend on generator choice.

Problem Difficulty. The same fit also characterizes the evaluation problems (Figure 4). Assistant failure type separates pair difficulty: `no_clarification` (median Elo 1525) and `disorganized` (1475) are harder, while `unnecessary_refusal` and `fabricated_answer` (both 849) are easier. User behavior shows less separation, with medians from 979 to 1247 and heavily overlapping interquartile ranges. This axis diversifies conversational contexts rather than directly specifying difficulty. Joint ranking thus identifies which generated categories challenge the current judges, information that an aggregate judge leaderboard alone does not provide.

Class Bias is a Capability Gap. Figure 8 in Appendix A.6 reports the per-judge gap between predicted and ground-truth failure type share. As judge strength declines, predictions collapse onto `evasion`, with over-prediction rising from near zero for the strongest judges to $+41.6$pp at the bottom of the leaderboard, leading to under-prediction of `no_clarification` and `unnecessary_refusal`, which fall to roughly -15pp and -18pp for the weakest judges. No such bias is observed at the top, where the biases of `gemini-3.1-pro` remain within 2pp of zero on every class, so the seven failure types remain separable for a strong judge. The class bias is therefore a property of the weaker judges rather than of the data. Weaker judges default to `evasion` when uncertain. The same collapse is reproduced in each domain (Figure 9), and the per-judge confusion matrix (Figure 10 in Appendix A.7) corroborates this at the prediction level.

Effect of Failure-type Prediction. We drop the failure-type component from the correctness criterion, leaving $(y^\star, r^\star)$, and refit Elo. Panel (a) of Figure 5 compares judge Elo with and without the type label $t^\star$, showing high correlation at Spearman $\rho = 0.968$ and Kendall $\tau = 0.895$. This shows that including failure type prediction as a correctness criterion widens the Elo gap between judges while largely preserving the leaderboard order. Hence, the failure taxonomy further distinguishes judges by testing whether they identify the type of failure, not only its presence. The same pattern holds within each domain (Figure 11 in Appendix A.8).

Error Analysis and Mitigation. The leaderboard exposes a surprising anomaly: `opus-4.7`, the headline model of the Claude family at the time of writing, sits below `sonnet-4.6`. This motivates a deeper investigation, including whether prompt-level intervention can recover the gap. We repurpose the human audit of Section 3.4 for error analysis of `opus-4.7`'s incorrect verdicts on the `clean` pairs, summarize its recurring failure modes, and design four prompt revisions targeting them: category hints (v2), explicit candidate comparison (v3), decision rules (v4), and chain-of-thought (v5), detailed

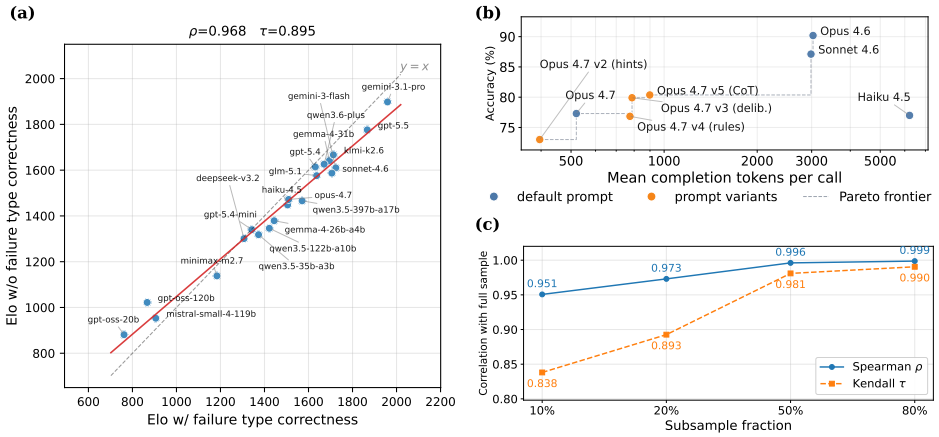


Figure 5: (a) Judge Elo with and without failure-type correctness as a criterion. (b) Accuracy vs. mean completion tokens for four default-prompt baselines and four `opus-4.7` prompt revisions; the dashed step is the empirical Pareto frontier. (c) Rank correlation between the full-sample judge ranking and rankings recomputed on uniform subsamples.

in Appendix B.8. To bracket the magnitude of the gap we additionally ran `opus-4.6` under the default prompt. Figure 5(b) plots accuracy against mean completion tokens per call for the four revisions and the four default-prompt baselines. `opus-4.6` substantially outperforms every variant of `opus-4.7` we tried, and none of the prompt revisions improves `opus-4.7` to the accuracy level of `opus-4.6` or `sonnet-4.6`, indicating a gap that prompting does not recover. The shift aligns with documented changes in the `opus-4.7` release (Anthropic, 2026): thinking is now an adaptive effort that the model can downscale at will, in place of `opus-4.6`'s fixed budget. Consistent with this, `opus-4.7` spends the fewest completion tokens per match of all 21 judges (Appendix A.5). Our four revisions cover the usual toolkit (more guidance, more deliberation, more structure, more thinking), and their joint failure suggests that, for this judge, switching models is more effective than tuning prompts.

Robustness and Benchmark Cost. To verify the incomplete-coverage argument of Section 3.3, we refit Elo on uniform subsamples of the judgment results at fractions $\{10\%, 20\%, 50\%, 80\%\}$ and report the rank correlation with the full data in Figure 5(c). Spearman ρ exceeds 0.95 from 10% onward and Kendall τ crosses the same threshold at 50%, so the leaderboard can be reproduced at a fraction of the full cost with negligible distortion to rankings. The same trend holds within each domain (Figure 12 in Appendix A.9). The leaderboard is also stable to curation: refitting on the pool before curation, with the top-5% Elo pairs and audit exclusions restored, matches the published ordering at Spearman $\rho \geq 0.997$, with the top-10 judge ranking preserved (Figure 13 in Appendix A.10), confirming that the Elo ranking is robust under residual noise. Constructing the full benchmark and evaluating all 21 judges cost $645 in API calls, and the released judge outputs allow the leaderboard to be recomputed without new calls (Appendix A.5).

Pairwise versus Pointwise. RANKJUDGE can also be run with the judges scoring conversations pointwise on a 1–10 Likert scale rather than choosing between candidates. The two protocols agree at a coarse level (Spearman $\rho = 0.809$, Appendix A.12) but the judge rankings diverge, consistent with the two protocols measuring different capabilities. Pairwise tests discrimination between two candidates, while pointwise tests absolute calibration of conversation quality without a reference.

5 CONCLUSION

We presented RANKJUDGE, an end-to-end generator for renewable evaluation of LLM judges on document-grounded, multi-turn conversations. Fully synthetic construction supplies fresh labeled comparisons, while joint judge-versus-problem modeling supports difficulty-based curation and ranking with incomplete evaluation coverage. Across three domains and 21 judges, the current instantiation yields differentiated rankings that remain stable under uniform subsampling. RANKJUDGE provides a reusable pipeline for constructing and calibrating judge benchmarks. Natural extensions include additional domains, longer interactions, and agentic or tool-use settings.

AI USE STATEMENT

In this work, we used generative AI tools for coding assistance, implementation of methods, feedback on experimental methods, and initial drafting of the paper. We note however that the authors substantially rewrote the initial draft by hand. Generative AI was also crucial to our proposed method for curating and cleaning synthetic datasets, as explained throughout the paper. Specifically, we used LLMs to propose assistant failure categories, generate conversation blueprints and dialogue pairs, verify candidate data, and produce the judge outputs evaluated in this study. The model roles and procedures are described in Sections 3.1 and 4, with prompts in Appendices B.3, B.6, B.7, and B.8. Generated items underwent automated coherence, adherence, and grounding checks; the authors also audited a sample as described in Appendix C. We have not used generative AI tools for independent data analysis, interpreting results, or proposing or refining hypotheses. Tasks related to theory, mathematical claims, proofs, or translation are not applicable to this work. Additionally, we used generative AI tools for producing code to plot experimental data and conducting literature searches. We have reviewed all AI-assisted work, for example by substantially rewriting the paper, and manually curating papers from the literature search into our bibliography and discussions of related work. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

Our work involved some manual annotation of data, which was entirely carried out by the authors. We do not foresee potentially harmful insights or applications of our methods.

REPRODUCIBILITY STATEMENT

We have made significant efforts to ensure that this work is reproducible. Our methodology is thoroughly explained in Sections 3 and 4. Additional experimental details have been included in Appendix B. Finally, we have included source code for our implementation at `anonymous.4open.science/r/RankJudge-Submission-11D1/`, which includes scripts for reproducing important parts of our work. We also release the curated evaluation slice as a fixed, versioned dataset at `huggingface.co/datasets/anonymous-rankjudge-neurips/RankJudge`. The codebase and data have been released publicly in a non-anonymized version.

REFERENCES

Chris Alberti, Daniel Andor, Emily Pitler, Jacob Devlin, and Michael Collins. Synthetic QA Corpora Generation with Roundtrip Consistency. In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 6168–6173, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1620. URL https://aclanthology.org/P19-1620/.

Anthropic. Claude Opus 4.7 System Card. https://www.anthropic.com/claude-opus-4-7-system-card, April 2026.

Ge Bai, Jie Liu, Xingyuan Bu, Yancheng He, Jiaheng Liu, Zhanhui Zhou, Zhuoran Lin, Wenbo Su, Tiezheng Ge, Bo Zheng, and Wanli Ouyang. MT-bench-101: A fine-grained benchmark for evaluating large language models in multi-turn dialogues. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7421–7454, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.401. URL https://aclanthology.org/2024.acl-long.401/.

Mert Cemri, Melissa Z Pan, Shuyi Yang, Lakshya A Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, Matei A Zaharia, Joseph Gonzalez, and Ion Stoica. Why Do Multi-Agent LLM Systems Fail? In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference. Curran Associates, Inc., 2025. doi: 10.52202/085713-4082. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/b1041e52d3be19f0a9bc491657488e4a-Paper-Datasets_and_Benchmarks_Track.pdf.

Dongping Chen, Ruoxi Chen, Shilin Zhang, Yaochen Wang, Yinuo Liu, Huichi Zhou, Qihui Zhang, Yao Wan, Pan Zhou, and Lichao Sun. MLLM-as-a-Judge: Assessing Multimodal LLM-as-a-Judge with Vision-Language Benchmark. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 6562–6595. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/chen24h.html.

Xiusi Chen, Gaotang Li, Ziqi Wang, Bowen Jin, Cheng Qian, Yu Wang, Hongru WANG, Yu Zhang, Denghui Zhang, Tong Zhang, Hanghang Tong, and Heng Ji. RM-R1: Reward Modeling as Reasoning. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=1ZqJ6jj75q.

Yelin Chen, Fanjin Zhang, Suping Sun, Yunhe Pang, Yuanchun Wang, Jian Song, XiaoYan Li, Lei Hou, Shu Zhao, Jie Tang, and Juanzi Li. RPC-bench: A fine-grained benchmark for research paper comprehension. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27683–27717, San Diego, California, United States, July 2026b. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1277. URL https://aclanthology.org/2026.acl-long.1277/.

Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Banghua Zhu, Hao Zhang, Michael Jordan, Joseph E. Gonzalez, and Ion Stoica. Chatbot Arena: An Open Platform for Evaluating LLMs by Human Preference. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 8359–8388. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/chiang24b.html.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep Reinforcement Learning from Human Preferences. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/d5e2c0adad503c91f91df240d0cd4e49-Paper.pdf.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training Verifiers to Solve Math Word Problems, 2021. URL https://arxiv.org/abs/2110.14168.

Kyle Dai, Maurice Burger, Roman Engeler, Max Bartolo, Clémentine Fourrier, Toby Drane, Mathias Leys, and Jake Golden. Judge Arena: Benchmarking LLMs as Evaluators, 2024. URL https://huggingface.co/blog/arena-atla. Accessed: 2026-05-01.

Kaustubh Deshpande, Ved Sirdeshmukh, Johannes Baptist Mols, Lifeng Jin, Ed-Yeremai Hernandez-Cardona, Dean Lee, Jeremy Kritz, Willow E. Primack, Summer Yue, and Chen Xing. MultiChallenge: A realistic multi-turn conversation evaluation benchmark challenging to frontier LLMs. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 18632–18702, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.958. URL https://aclanthology.org/2025.findings-acl.958/.

Jacob Eisenstein, Fantine Huot, Adam Fisch, Jonathan Berant, and Mirella Lapata. MT-PingEval: Evaluating Multi-Turn Collaboration with Private Information Games, 2026. URL https://arxiv.org/abs/2602.24188.

Dongyang Fan, Sebastien Delsad, Nicolas Flammarion, and Maksym Andriushchenko. HalluHard: A Hard Multi-Turn Hallucination Benchmark, 2026. URL https://arxiv.org/abs/2602.01031.

Ariel Gera, Odellia Boni, Yotam Perlitz, Roy Bar-Haim, Lilach Eden, and Asaf Yehudai. JuStRank: Benchmarking LLM judges for system ranking. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of*

the Association for Computational Linguistics (Volume 1: Long Papers), pp. 682–712, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.34. URL https://aclanthology.org/2025.acl-long.34/.

Google DeepMind. Gemini 3.1 Pro Model Card. https://deepmind.google/models/model-cards/gemini-3-1-pro/, February 2026.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring Massive Multitask Language Understanding. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Xingjian Hu, Ziqian Zhang, Yue Huang, Kai Zhang, Ruoxi Chen, Yixin Liu, Qingsong Wen, Kaidi Xu, Xiangliang Zhang, Neil Zhenqiang Gong, and Lichao Sun. EIP: Weighted Ranking of LLMs by Quantifying Question Difficulty. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=jnX5GJIoYt.

Hui Huang, Yancheng He, Hongli Zhou, Rui Zhang, Wei Liu, Weixun Wang, Jiaheng Liu, and Wenbo Su. Think-J: Learning to Think for Generative LLM-as-a-Judge. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(37):31158–31166, March 2026. ISSN 2159-5399. doi: 10.1609/aaai.v40i37.40377. URL http://dx.doi.org/10.1609/aaai.v40i37.40377.

David R. Hunter and Runze Li. Variable selection using MM algorithms. *The Annals of Statistics*, 33 (4), August 2005. ISSN 0090-5364. doi: 10.1214/009053605000000200. URL http://dx.doi.org/10.1214/009053605000000200.

Jerry Loh. SP500-EDGAR-10K, 2026. URL https://huggingface.co/datasets/jlohding/sp500-edgar-10k. Accessed: 2026-05-06.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. PubMedQA: A dataset for biomedical research question answering. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 2567–2577, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1259. URL https://aclanthology.org/D19-1259/.

Ivan Kartáč, Mateusz Lango, and Ondrej Dusek. Reasoning gets harder for LLMs inside a dialogue. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12263–12303, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.560. URL https://aclanthology.org/2026.acl-long.560/.

Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, and Minjoon Seo. Prometheus: Inducing Fine-Grained Evaluation Capability in Language Models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=8euJaTveKw.

Fanqi Kong, Ruijie Zhang, Huaxiao Yin, Guibin Zhang, Xiaofei Zhang, Ziang Chen, Zhaowei Zhang, Xiaoyuan Zhang, Song-Chun Zhu, and Xue Feng. Aegis: Automated Error Generation and Attribution for Multi-Agent Systems. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=zqcYoxXiN3.

Wojciech Kryscinski, Bryan McCann, Caiming Xiong, and Richard Socher. Evaluating the Factual Consistency of Abstractive Text Summarization. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 9332–9346, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.750. URL https://aclanthology.org/2020.emnlp-main.750/.

Bruce Kuwahara, Chen-Yuan Lin, Xiao Shi Huang, Kin Kwan Leung, Jullian Yapeter, Ilya Stanevich, Felipe Perez, and Jesse Cresswell. Document Summarization with Conformal Importance Guarantees. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 67107–67152. Curran Associates, Inc., 2025. doi: 10.52202/085713-2249. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/60cefc59610f8f59ea10099f99a36726-Paper-Conference.pdf.

Wai-Chung Kwan, Xingshan Zeng, Yuxin Jiang, Yufei Wang, Liangyou Li, Lifeng Shang, Xin Jiang, Qun Liu, and Kam-Fai Wong. MT-eval: A multi-turn capabilities evaluation benchmark for large language models. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 20153–20177, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.1124. URL https://aclanthology.org/2024.emnlp-main.1124/.

Philippe Laban, Wojciech Kryscinski, Divyansh Agarwal, Alexander Fabbri, Caiming Xiong, Shafiq Joty, and Chien-Sheng Wu. SummEdits: Measuring LLM ability at factual reasoning through the lens of summarization. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 9662–9676, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.600. URL https://aclanthology.org/2023.emnlp-main.600/.

Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. LLMs Get Lost In Multi-Turn Conversation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=VKGTGGcwl6.

John P. Lalor, Hao Wu, and Hong Yu. Building an Evaluation Scale using Item Response Theory. In Jian Su, Kevin Duh, and Xavier Carreras (eds.), *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 648–657, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1062. URL https://aclanthology.org/D16-1062/.

Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, Noah A. Smith, and Hannaneh Hajishirzi. RewardBench: Evaluating Reward Models for Language Modeling. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1755–1797, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-195-7. doi: 10.18653/v1/2025.findings-naacl.96. URL https://aclanthology.org/2025.findings-naacl.96/.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Ren Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, and Sushant Prakash. RLAIF vs. RLHF: Scaling Reinforcement Learning from Human Feedback with AI Feedback. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 26874–26901. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/lee24t.html.

Kin Kwan Leung, Mouloud Belbahri, Yi Sui, Alex Labach, Xueying Zhang, Stephen Anthony Rose, and Jesse C. Cresswell. Classifying and addressing the diversity of errors in retrieval-augmented generation systems. In Vera Demberg, Kentaro Inui, and Lluís Marquez (eds.), *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3185–3207, Rabat, Morocco, March 2026. Association for Computational Linguistics. ISBN 979-8-89176-380-7. doi: 10.18653/v1/2026.eacl-long.147. URL https://aclanthology.org/2026.eacl-long.147/.

Junyi Li, Xiaoxue Cheng, Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. HaluEval: A large-scale hallucination evaluation benchmark for large language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 6449–6464, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.397. URL https://aclanthology.org/2023.emnlp-main.397/.

Songze Li, Chuokun Xu, Jiaying Wang, Xueluan Gong, Chen Chen, Jirui Zhang, Jun Wang, Kwok-Yan Lam, and Shouling Ji. LLMs Cannot Reliably Judge (Yet?): A Comprehensive Assessment on the Robustness of LLM-as-a-Judge. *IEEE Transactions on Information Forensics and Security*, pp. 1–1, 2026a. ISSN 1556-6021. doi: 10.1109/tifs.2026.3737714. URL http://dx.doi.org/10.1109/tifs.2026.3737714.

Xiang Lisa Li, Farzaan Kaiyom, Evan Zheran Liu, Yifan Mai, Percy Liang, and Tatsunori Hashimoto. AutoBencher: Towards Declarative Benchmark Construction. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=ymt4crbbXh.

Yubo Li, Xiaobin Shen, Yidi Miao, Xinyu Yao, Xueying Ding, Ramayya Krishnan, and Rema Padman. Beyond Single-Turn: A Survey on Multi-Turn Interactions with Large Language Models, 2026b. URL https://arxiv.org/abs/2504.04717.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's Verify Step by Step. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-eval: NLG evaluation using gpt-4 with better human alignment. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 2511–2522, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.153. URL https://aclanthology.org/2023.emnlp-main.153/.

Yixin Liu, Kejian Shi, Alexander Fabbri, Yilun Zhao, PeiFeng Wang, Chien-Sheng Wu, Shafiq Joty, and Arman Cohan. ReIFE: Re-evaluating instruction-following evaluation. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *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)*, pp. 12247–12287, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.610. URL https://aclanthology.org/2025.naacl-long.610/.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jian-Guang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, Yansong Tang, and Dongmei Zhang. WizardMath: Empowering Mathematical Reasoning for Large Language Models via Reinforced Evol-Instruct. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=mMPMHWOdOy.

Felipe Maia Polo, Lucas Weber, Leshem Choshen, Yuekai Sun, Gongjun Xu, and Mikhail Yurochkin. tinyBenchmarks: evaluating LLMs with fewer examples. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 34303–34326. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/maia-polo24a.html.

John Mendonça, Alon Lavie, and Isabel Trancoso. MEDAL: A framework for benchmarking LLMs as multilingual open-domain dialogue evaluators. In Vera Demberg, Kentaro Inui, and Lluís Marquez (eds.), *Findings of the Association for Computational Linguistics: EACL 2026*, pp. 2069–2097, Rabat, Morocco, March 2026. Association for Computational Linguistics. ISBN 979-8-89176-386-9. doi: 10.18653/v1/2026.findings-eacl.109. URL https://aclanthology.org/2026.findings-eacl.109/.

Christopher Mohri and Tatsunori Hashimoto. Language Models with Conformal Factuality Guarantees. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 36029–36047. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/mohri24a.html.

OpenRouter. OpenRouter, 2026. URL https://openrouter.ai. Accessed: 2026-05-06.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 27730–27744. Curran Associates, Inc., 2022. doi: 10.52202/068431-2011. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/b1efde53be364a73914f58805a001731-Paper-Conference.pdf.

Qwen Team. Qwen3.5: Towards Native Multimodal Agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Pedro Rodriguez, Joe Barrow, Alexander Hoyle, John P. Lalor, Robin Jia, and Jordan Boyd-Graber. Evaluation Examples are not Equally Informative: How should that change NLP Leaderboards? In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4486–4503, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.346. URL https://aclanthology.org/2021.acl-long.346/.

Brendan Leigh Ross, Noël Vouitsis, Atiyeh Ashari Ghomi, Rasa Hosseinzadeh, Ji Xin, Zhaoyan Liu, Yi Sui, Shiyi Hou, Kin Kwan Leung, Gabriel Loaiza-Ganem, and Jesse C. Cresswell. Textual Bayes: Quantifying Prompt Uncertainty in LLM-Based Systems. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=VPmsAr1OTl.

Swarnadeep Saha, Xian Li, Marjan Ghazvininejad, Jason E Weston, and Tianlu Wang. Learning to Plan & Reason for Evaluation with Thinking-LLM-as-a-Judge. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 52565–52583. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/saha25b.html.

William Saunders, Catherine Yeh, Jeff Wu, Steven Bills, Long Ouyang, Jonathan Ward, and Jan Leike. Self-critiquing models for assisting human evaluators, 2022. URL https://arxiv.org/abs/2206.05802.

Lin Shi, Chiyu Ma, Wenhua Liang, Xingjian Diao, Weicheng Ma, and Soroush Vosoughi. Judging the judges: A systematic study of position bias in LLM-as-a-judge. In Kentaro Inui, Sakriani Sakti, Haofen Wang, Derek F. Wong, Pushpak Bhattacharyya, Biplab Banerjee, Asif Ekbal, Tanmoy Chakraborty, and Dhirendra Pratap Singh (eds.), *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*, pp. 292–314, Mumbai, India, December 2025. The Asian Federation of Natural Language Processing and The Association for Computational Linguistics. ISBN 979-8-89176-298-5. doi: 10.18653/v1/2025.ijcnlp-long.18. URL https://aclanthology.org/2025.ijcnlp-long.18/.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 3008–3021. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1f89885d556929e98d3ef9b86448f951-Paper.pdf.

Sijun Tan, Siyuan Zhuang, Kyle Montgomery, William Yuan Tang, Alejandro Cuadron, Chenguang Wang, Raluca Popa, and Ion Stoica. JudgeBench: A Benchmark for Evaluating LLM-Based Judges. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=G0dksFayVq.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. FEVER: a Large-scale Dataset for Fact Extraction and VERification. In Marilyn Walker, Heng Ji, and Amanda Stent (eds.), *Proceedings of the 2018 Conference of the North American Chapter of the Association*

for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pp. 809–819, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1074. URL https://aclanthology.org/N18-1074/.

Binghai Wang, Yantao Liu, Yuxuan Liu, Tianyi Tang, Shenzhi Wang, Chang Gao, Chujie Zheng, Yichang Zhang, Le Yu, Shixuan Liu, Tao Gui, Qi Zhang, Xuanjing Huang, Bowen Yu, Fei Huang, and Junyang Lin. Outcome accuracy is not enough: Aligning the reasoning process of reward models. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 41482–41508, San Diego, California, United States, July 2026a. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1924. URL https://aclanthology.org/2026.acl-long.1924/.

Jiaan Wang, Yunlong Liang, Fandong Meng, Zengkui Sun, Haoxiang Shi, Zhixu Li, Jinan Xu, Jianfeng Qu, and Jie Zhou. Is ChatGPT a good NLG evaluator? a preliminary study. In Yue Dong, Wen Xiao, Lu Wang, Fei Liu, and Giuseppe Carenini (eds.), *Proceedings of the 4th New Frontiers in Summarization Workshop*, pp. 1–11, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.newsum-1.1. URL https://aclanthology.org/2023.newsum-1.1/.

Ruiqi Wang, Xinchen Wang, Cuiyun Gao, Chun Yong Chong, Xin Xia, and Qing Liao. AXIOM: Benchmarking LLM-as-a-Judge for Code via Rule-Based Perturbation and Multisource Quality Calibration, 2025a. URL https://arxiv.org/abs/2512.20159.

Xingyao Wang, Zihan Wang, Jiateng Liu, Yangyi Chen, Lifan Yuan, Hao Peng, and Heng Ji. MINT: Evaluating LLMs in Multi-turn Interaction with Tools and Language Feedback. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=jp3gWrMuIZ.

Yicheng Wang, Jiayi Yuan, Yu-Neng Chuang, Zhuoer Wang, Yingchi Liu, Mark Cusick, Param Kulkarni, Zhengping Ji, Yasser Ibrahim, and Xia Hu. DHP benchmark: Are LLMs good NLG evaluators? In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 8094–8109, Albuquerque, New Mexico, April 2025b. Association for Computational Linguistics. ISBN 979-8-89176-195-7. doi: 10.18653/v1/2025.findings-naacl.451. URL https://aclanthology.org/2025.findings-naacl.451/.

Yutong Wang, Pengliang Ji, Chaoqun Yang, Kaixin Li, Ming Hu, Jiaoyang Li, and Guillaume Sartoretti. MCTS-Judge: Test-Time Scaling in LLM-as-a-Judge for Code Correctness Evaluation, 2026b. URL https://arxiv.org/abs/2502.12468.

Zongqi Wang, Rui Wang, Yuchuan Wu, Yiyao Yu, Pinyi Zhang, Shaoning Sun, Yujiu Yang, and Yongbin Li. Reward Modeling from Natural Language Human Feedback. In *Forty-third International Conference on Machine Learning*, 2026c. URL https://openreview.net/forum?id=nd0hT1eyEo.

Chenxi Whitehouse, Tianlu Wang, Ping Yu, Xian Li, Jason E Weston, Ilia Kulikov, and Swarnadeep Saha. J1: Incentivizing Thinking in LLM-as-a-Judge via Reinforcement Learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=dnJEHl6DI1.

Austin Xu, Srijan Bansal, Yifei Ming, Semih Yavuz, and Shafiq Joty. Does context matter? ContextualJudgeBench for evaluating LLM-based judges in contextual settings. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9541–9564, Vienna, Austria, July 2025a. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.470. URL https://aclanthology.org/2025.acl-long.470/.

Zhenghao Xu, Qin Lu, Qingru Zhang, Liang Qiu, Ilgee Hong, Changlong Yu, Wenlin Yao, Yao Liu, Haoming Jiang, Lihong Li, Hyokun Yun, and Tuo Zhao. Ask a Strong LLM Judge when Your Reward Model is Uncertain. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 74639–74664. Curran Associates, Inc., 2025b. doi:

10.52202/085713-2506. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/6c312cf14321a99ae3788a41593cf2bd-Paper-Conference.pdf.

Peiwen Yuan, Shaoxiong Feng, Yiwei Li, Xinglin Wang, Yueqi Zhang, Jiayi Shi, Chuyi Tan, Boyuan Pan, Yao Hu, and Kan Li. LLM-powered benchmark factory: Reliable, generic, and efficient. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 35882–35903, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1661. URL https://aclanthology.org/2026.acl-long.1661/.

Zhiyuan Zeng, Jiatong Yu, Tianyu Gao, Yu Meng, Tanya Goyal, and Danqi Chen. Evaluating Large Language Models at Evaluating Instruction Following. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=tr0KidwPLc.

Xinghua Zhang, Bowen Yu, Haiyang Yu, Yangyu Lv, Tingwen Liu, Fei Huang, Hongbo Xu, and Yongbin Li. Wider and Deeper LLM Networks are Fairer LLM Evaluators, 2023. URL https://arxiv.org/abs/2308.01862.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph Gonzalez, and Ion Stoica. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 46595–46623. Curran Associates, Inc., 2023. doi: 10.52202/075280-2020. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/91f18a1287b398d378ef22505bf41832-Paper-Datasets_and_Benchmarks.pdf.

Kaijie Zhu, Jiaao Chen, Jindong Wang, Neil Zhenqiang Gong, Diyi Yang, and Xing Xie. DyVal: Dynamic Evaluation of Large Language Models for Reasoning Tasks. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=gjfOL9z5Xr.

A Additional Results

A.1 Detailed Judge Elo Ratings

Table 4 gives each judge's combined Elo together with the per-domain breakdown, where confidence intervals are obtained from the cluster-robust sandwich estimator of Appendix B.1. The intervals tighten toward the middle of the leaderboard, where contested matches concentrate, and widen at the extremes, where the strongest and weakest judges win or lose decisively against most opponents and so accumulate little discriminative information. Finance is the lowest-Elo domain for most mid-leaderboard judges, consistent with the heavier numerical reasoning load imposed by 10-K filings; `gpt-5.4`, the only mid-leaderboard judge whose Finance Elo exceeds its ML and Biomedical scores, is a notable exception.

Table 4: Elo of LLM judges (95% CI). Best per column in bold. Sorted by Elo measured by the combined dataset (All).

Judge	All	ML	Med	Fin	Type
`gemini-3.1-pro`	**1959 ± 80**	**1992 ± 156**	**1910 ± 131**	**1933 ± 129**	Prop.
`gpt-5.5`	1867 ± 65	1854 ± 109	1852 ± 119	1850 ± 107	Prop.
`gemini-3-flash`	1724 ± 49	1665 ± 81	1759 ± 93	1713 ± 83	Prop.
`kimi-k2.6`	1713 ± 54	1743 ± 99	1732 ± 93	1634 ± 89	Open
`sonnet-4.6`	1706 ± 49	1755 ± 88	1707 ± 90	1624 ± 80	Prop.
`gemma-4-31b`	1695 ± 47	1708 ± 81	1671 ± 87	1662 ± 76	Open
`qwen3.6-plus`	1671 ± 51	1665 ± 88	1719 ± 103	1597 ± 79	Prop.
`glm-5.1`	1637 ± 47	1665 ± 87	1671 ± 89	1546 ± 73	Open
`gpt-5.4`	1631 ± 54	1581 ± 90	1617 ± 83	1653 ± 99	Prop.
`qwen3.5-397b-a17b`	1571 ± 43	1590 ± 74	1597 ± 80	1490 ± 72	Open
`opus-4.7`	1510 ± 43	1523 ± 75	1587 ± 82	1395 ± 71	Prop.
`haiku-4.5`	1505 ± 47	1492 ± 80	1487 ± 78	1490 ± 82	Prop.
`gemma-4-26b-a4b`	1444 ± 42	1428 ± 70	1487 ± 75	1380 ± 74	Open
`qwen3.5-122b-a10b`	1422 ± 37	1381 ± 60	1454 ± 68	1395 ± 62	Open
`qwen3.5-35b-a3b`	1373 ± 38	1349 ± 65	1471 ± 73	1272 ± 63	Open
`gpt-5.4-mini`	1342 ± 37	1343 ± 62	1363 ± 73	1279 ± 61	Prop.
`deepseek-v3.2`	1307 ± 43	1260 ± 69	1370 ± 85	1259 ± 72	Open
`minimax-m2.7`	1184 ± 40	1119 ± 68	1313 ± 70	1094 ± 70	Open
`mistral-small-4-119b`	906 ± 42	887 ± 68	873 ± 80	911 ± 76	Open
`gpt-oss-120b`	867 ± 42	842 ± 67	911 ± 72	805 ± 80	Open
`gpt-oss-20b`	762 ± 45	734 ± 73	812 ± 81	697 ± 81	Open

A.2 Problem Rankings

Figure 4 in the main text reports the pair-difficulty distributions for both conditioning axes.

Assistant failure type separates pair difficulty cleanly. `no_clarification` (median Elo 1525) and `disorganized` (1475) consistently beat judges, while `unnecessary_refusal` and `fabricated_answer` (both 849) are the easiest to catch: refusals stand out in context and fabrications are directly checkable against the grounding document. User behavior, by contrast, barely shifts pair difficulty: the seven categories span only 979 to 1247 in median Elo with heavily overlapping IQRs. This is by design. The user behavior axis is included to diversify the conversational contexts in the benchmark rather than to differentiate judges, so the lack of separation here is the expected outcome.

A.3 Model Bias and Self-Preference

Setup. We reran generation with `opus-4.7` in place of `gpt-5.5`, using the same source pool, conditioning prompts, verification gates, and 21-judge panel. To bound API cost, the parallel run used 50 candidates per domain, yielding 150 pairs of which 131 pass verification. BT is fit independently

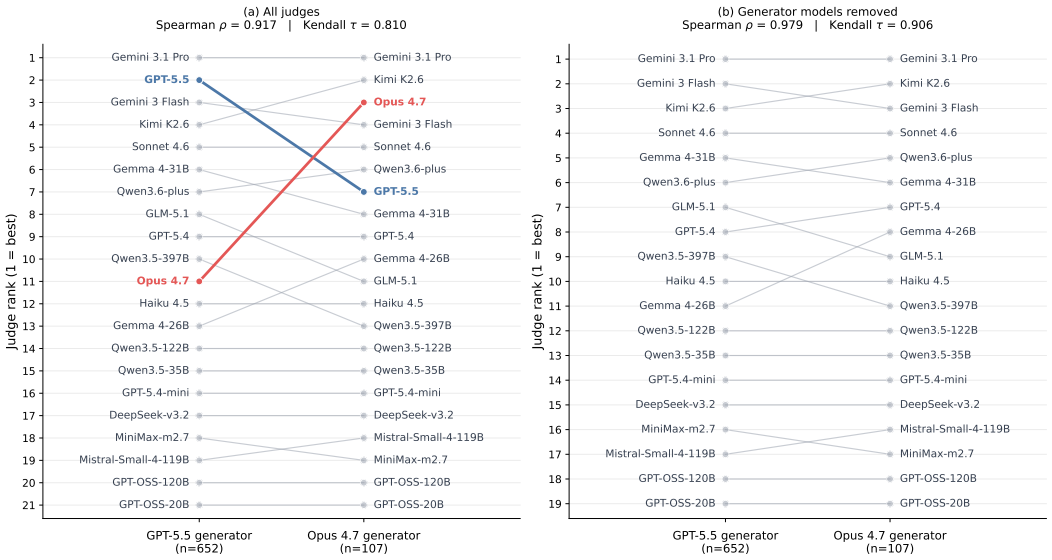


Figure 6: Generator sensitivity. Judge ranks under the published `gpt-5.5` pair set (left axis) and the parallel `opus-4.7` set (right axis). Panel (b) displays the relative order of the 19 non-generator judges.

to the published `gpt-5.5` slice (652 pairs) and the parallel filtered slice (107 informative pairs). Panel (b) restricts the displayed rankings to the 19 non-generator judges after the full-panel fits.

Findings. In Figure 6(a), `gpt-5.5` moves from rank 2 to rank 7, while `opus-4.7` moves from rank 11 to rank 3. `gemini-3.1-pro` ranks first in both conditions. Across all judges, Spearman $\rho = 0.917$ and Kendall $\tau = 0.810$; restricting the comparison to the other 19 judges gives $\rho = 0.979$ and $\tau = 0.906$. The shifts concentrate on the two generator models, while the order of the remaining judges is stable.

A.4 Filtering Cascade

Tables 3, 5, and 6 report the filtering cascade sliced three ways: by domain, by assistant failure type, and by user behavior type. Of 1200 generated pairs, 652 survive every layer (overall survival 54.3%). The dominant signal sits on the assistant-failure axis. `unnecessary_refusal` and `fabricated_answer` clear the adherence verifier with zero loss, while `disorganized` loses 130 of its 174 generated pairs at that single layer and ends up the rarest type in the published evaluation set (30 pairs, 17.2% survival). The per-domain and per-user-behavior survival rates are far tighter and have no comparable bottleneck (48.5–60.0% across domains, 46.4–61.8% across user behaviors), so the shape of the cascade is set almost entirely by which flaw is being injected. The next paragraph and worked example explain why `disorganized` is the hard case: it is the only assistant flaw whose detection condition is structural rather than categorical, so a turn must lack logical flow strongly enough for the verifier to confirm the flaw, but not so strongly that every judge picks the better conversation on sight.

Why `disorganized` struggles at adherence. Refusals, fabrications, contradictions, and instruction violations are binary properties of a turn's content: a turn either declines an answerable question or it does not. `disorganized` is different. The flaw lives in the ordering and foregrounding of *correct* content, not in the content itself, so the bad turn must reshuffle the same facts that the good turn presents cleanly. Make the reshuffle too mild and the adherence verifier rejects the pair, with rejection reasons like "*Round 2 is mostly clear and foregrounds the main correction, so it does not actually*

Table 5: Verification and filtering cascade, split by assistant failure type.

Assistant Failure Type	Generated	Coherence	Adherence	Grounding	Uninformative	Human	Top 5%	Survival
`self_contradiction`	132	131 (−1)	117 (−14)	96 (−21)	94 (−2)	92 (−2)	88 (−4)	66.7%
`evasion`	165	165 (−0)	152 (−13)	126 (−26)	113 (−13)	111 (−2)	108 (−3)	65.5%
`disorganized`	174	174 (−0)	44 (−130)	34 (−10)	34 (−0)	33 (−1)	30 (−3)	17.2%
`fabricated_answer`	180	180 (−0)	180 (−0)	156 (−24)	104 (−52)	104 (−0)	104 (−0)	57.8%
`instruction_forgetting`	147	144 (−3)	143 (−1)	86 (−57)	80 (−6)	80 (−0)	79 (−1)	53.7%
`no_clarification`	231	229 (−2)	224 (−5)	178 (−46)	137 (−41)	126 (−11)	103 (−23)	44.6%
`unnecessary_refusal`	171	171 (−0)	171 (−0)	145 (−26)	141 (−4)	140 (−1)	140 (−0)	81.9%
All	1200	1194 (−6)	1031 (−163)	821 (−210)	703 (−118)	686 (−17)	652 (−34)	54.3%

Table 6: Verification and filtering cascade, split by user behavior type.

User Behavior Type	Generated	Coherence	Adherence	Grounding	Uninformative	Human	Top 5%	Survival
`focused`	150	148 (−2)	140 (−8)	115 (−25)	93 (−22)	90 (−3)	87 (−3)	58.0%
`integrative`	222	222 (−0)	192 (−30)	139 (−53)	117 (−22)	110 (−7)	103 (−7)	46.4%
`scattered`	144	142 (−2)	126 (−16)	103 (−23)	92 (−11)	90 (−2)	89 (−1)	61.8%
`skeptical`	138	138 (−0)	123 (−15)	94 (−29)	80 (−14)	79 (−1)	77 (−2)	55.8%
`misinformed`	174	173 (−1)	141 (−32)	132 (−9)	122 (−10)	119 (−3)	107 (−12)	61.5%
`exploratory`	174	173 (−1)	151 (−22)	111 (−40)	89 (−22)	88 (−1)	83 (−5)	47.7%
`underspecified`	198	198 (−0)	158 (−40)	127 (−31)	110 (−17)	110 (−0)	106 (−4)	53.5%
All	1200	1194 (−6)	1031 (−163)	821 (−210)	703 (−118)	686 (−17)	652 (−34)	54.3%

exhibit the required disorganized flaw".[1] Make the reshuffle too aggressive and the bad turn either drops or distorts a fact, which a strong judge then catches as a fabrication or evasion rather than as disorganization, and the pair is removed at the informativeness, human, or top-Elo layer.

The example below is from pair `id=40589ed7f7f0` (ML, `user_behavior_type=misinformed`, `assistant_behavior_type=disorganized`, `bad_round_index=2`). The user's second turn pushes a misinformed claim that the paper's main novelty was DP-Adam beating DP-SGD, that no DP-SGD baseline was reported, and that prior work had privately fine-tuned GPT-2 with DP-SGD. Both assistant responses contain the same three corrections: the E2E DP-SGD BLEU number (63.17), that DP-Adam was not claimed as the contribution, and that the cited GPT-2 prior work fine-tuned GPT-2 non-privately. The only difference is structural: `convo_a` leads with the answer to the user's question ("DP-Adam was not claimed as a new algorithm"), while `convo_b` opens with a peripheral hyperparameter detail and threads the lead correction through a privacy-by-post-processing aside, leaving it half a paragraph in.

[1] Verbatim from the `adherence.bad_issue` field of a rejected `disorganized` pair in `verification.json`; analogous strings ("clear and logically organized", "gives a clear lead answer") account for the bulk of the 130-pair attrition in row `disorganized` of Table 5.

Clean

DP-Adam was not claimed as a new algorithm; it was already common in DP libraries, and it uses the same gradient privatization mechanism as DP-SGD before the Adam update. The paper's contribution was showing that full fine-tuning of large pretrained language models can work well under DP when done with suitable hyperparameters, aligned objectives, and the memory trick. Also, the referenced prior work did not privately fine-tune GPT-2: it fine-tuned GPT-2 non-privately and used a privately trained feedforward network for private experiments.

Disorganized

On E2E at epsilon $= 8$, their extra DP-SGD experiment used learning rate 3 and clipping norm 0.1 and reached BLEU 63.17 over 3 seeds, which was on par with DP-Adam. DP-Adam itself was not claimed as the contribution, even though after the same subsampled Gaussian gradient privatization, Adam-style updates preserve privacy by post-processing. The contribution was applying known DP optimization effectively to full language-model fine-tuning, with the memory trick making it computationally practical. Also, the cited GPT-2 work did not privately fine-tune GPT-2 with DP-SGD; it fine-tuned GPT-2 non-privately…

The two responses are factually equivalent against the source; only the ordering differs. This pair sits in the narrow band where the verifier accepts (the lead answer is genuinely buried, mid-paragraph, after a hyperparameter aside and a post-processing detour) and judges still split (the correct facts are all present, so a judge has to actually read for structure rather than skim for missing content). Most generation attempts miss this band in one direction or the other, which is why `disorganized` loses three quarters of its candidate pairs at the adherence layer alone.

A.5 Elo vs. Compute

For the instantiation in Section 4, generating 1200 candidate pairs cost \$121, verification cost \$155, and evaluation by all 21 judges cost \$369, for \$645 in total. Released judge outputs allow the leaderboard to be recomputed without new API calls.

Compute cost is the other consideration for determining the best judge. Figure 7 relates each judge's Elo to its mean per-match compute. The token-axis frontier collapses to three judges, `opus-4.7` (549 tokens, 1510 Elo), `gpt-5.5` (934 tokens, 1867), and `gemini-3.1-pro` (1723 tokens, 1959); a long tail of mid-Elo judges spends 7–10k tokens per match while sitting well off the frontier, so verbosity alone does not buy ranking accuracy. The cost-axis frontier tells a different story, with open-source judges holding most of the curve: `gemma-4-31b` reaches 1695 Elo at \$0.0023 per match, less than a tenth the price of `gemini-3.1-pro` for 264 fewer Elo points, and `gpt-oss-120b` extends the frontier into the sub-cent regime at the bottom of the leaderboard. Only `gemini-3.1-pro` lies on both frontiers; `gpt-5.5` is dominated on cost, so budget-bound users can reach competitive Elo at a fraction of the spend via `gemini-3-flash` or an open-source judge such as `gemma-4-31b`.

A.6 Class Bias by Domain

Figure 8 reports the gap between each judge's predicted and labeled failure-type shares. Lower-ranked judges increasingly over-predict `evasion`, reaching $+41.6$ percentage points at the bottom of the leaderboard. For the weakest judges, `no_clarification` and `unnecessary_refusal` are under-predicted by roughly 15 and 18 points, respectively. The strongest judge, `gemini-3.1-pro`, stays within 2 points on every class.

Figure 9 shows that the prediction-bias signature is essentially domain-invariant: across Machine Learning, Biomedical, and Finance, the same column pattern repeats, with `evasion` the dominant red column and `no_clarification` the dominant blue one, and the magnitude of the bias growing as Elo decreases from left to right. The other five categories carry only small, noisy biases. This mirrors the confusion structure of Appendix A.7: weak judges fold ambiguous failures into `evasion` and rarely commit to `no_clarification`, regardless of the underlying domain.

A.7 Assistant Failure Type Confusion

Figure 10 contrasts how the four highest- and lowest-Elo judges classify the assistant failure type. The top row concentrates nearly all of its mass on the diagonal across most categories; the only

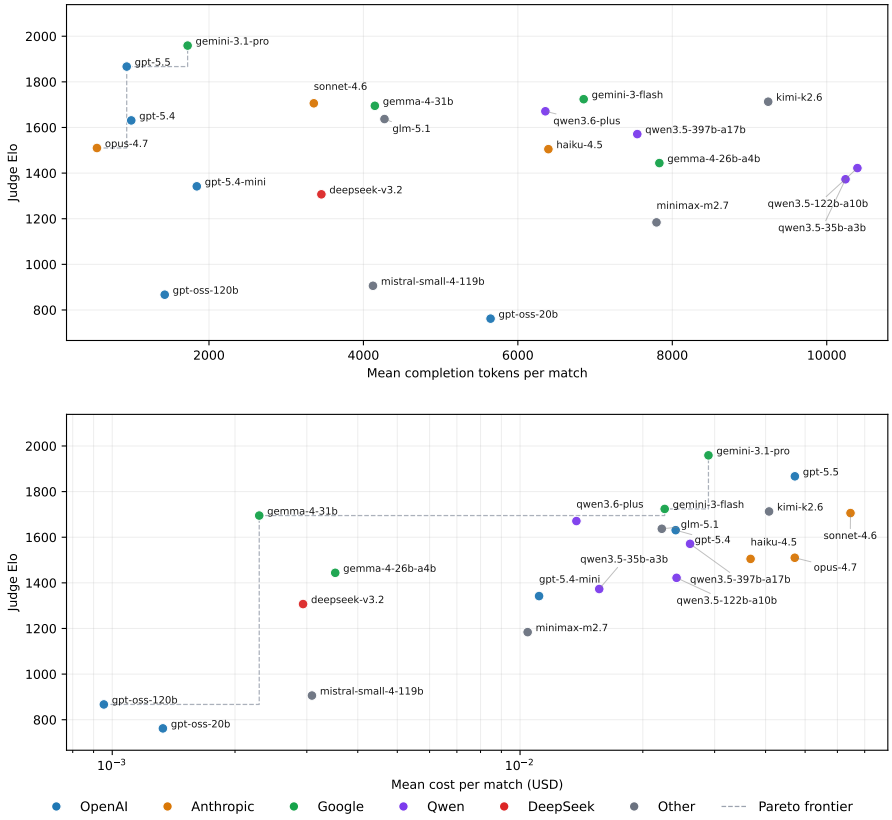


Figure 7: Judge Elo against per-match compute. Top: mean completion tokens per match (linear). Bottom: mean cost per match in USD (log). Markers are colored by provider and the dashed grey step line traces the empirical Pareto frontier.

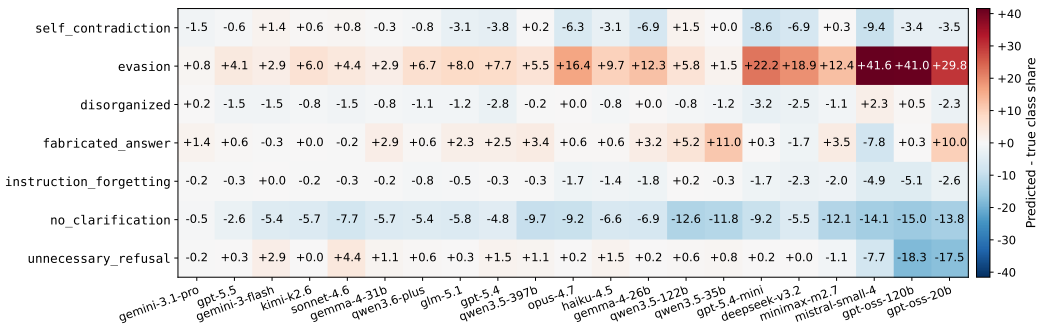
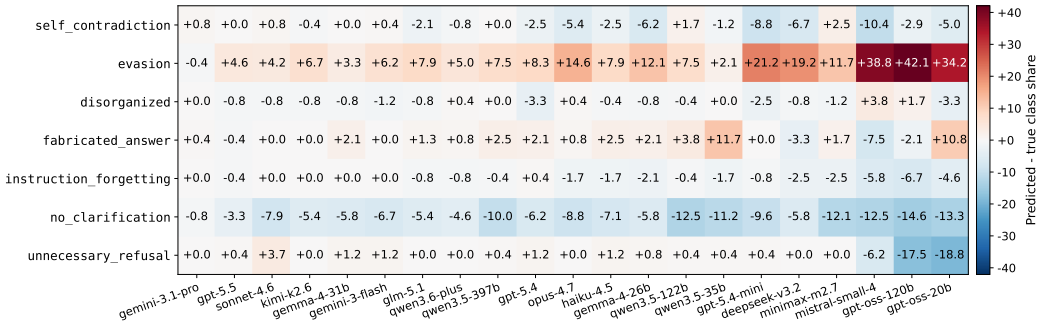
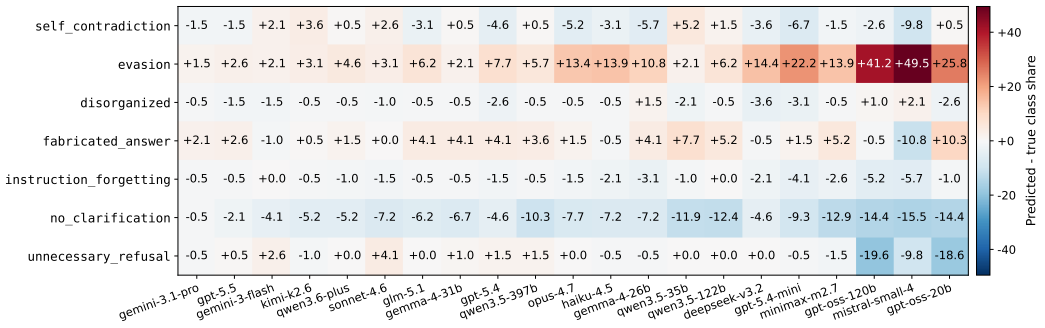


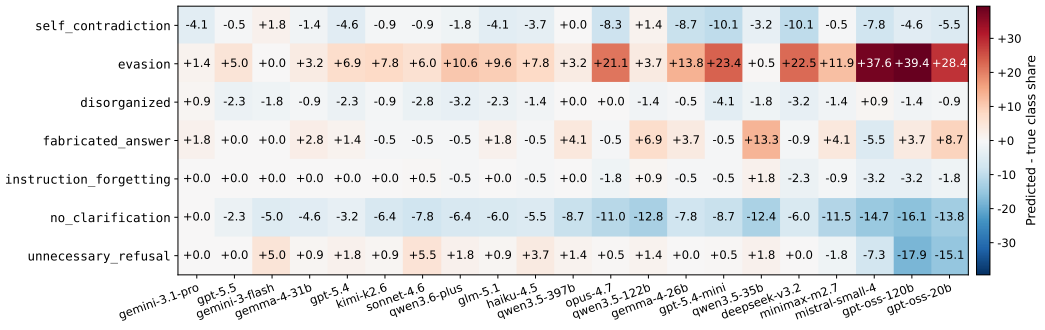
Figure 8: Per-class prediction bias for each judge. Each cell gives the difference in percentage points between the judge's predicted share of the assistant failure type and the ground-truth share. Red indicates over-prediction, while blue shows under-prediction. Columns are sorted by Elo.



(a) Machine Learning



(b) Biomedical



(c) Finance

Figure 9: Per-class prediction bias for each judge across the three domains. Each cell gives the difference between the judge's predicted share of the assistant failure type and its ground-truth share, in percentage points; red indicates over-prediction and blue under-prediction. Columns are sorted by Elo.

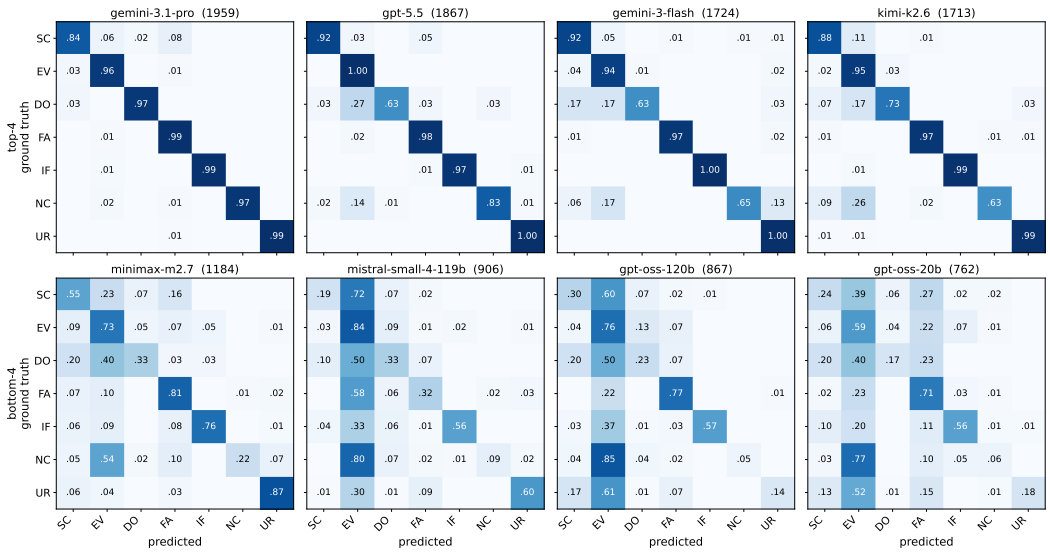


Figure 10: Per-judge confusion of assistant failure type predictions, row-normalized so each row sums to one. Top row: the four highest-Elo judges; bottom row: the four lowest. Class labels abbreviate the seven assistant failure types (SC `self_contradiction`, EV `evasion`, DO `disorganized`, FA `fabricated_answer`, IF `instruction_forgetting`, NC `no_clarification`, UR `unnecessary_refusal`).

residual confusion is on `disorganized` and `no_clarification`, both of which leak modestly into `evasion`. The bottom row collapses toward a single dump bin: weak judges relabel a large share of `no_clarification`, `self_contradiction`, `disorganized`, and `unnecessary_refusal` pairs as `evasion`, suggesting that taxonomy resolution, not failure detection, is what separates the leaderboard tail from its head.

A.8 Failure-type Correctness by Domain

We drop the failure-type component from the correctness criterion, leaving $(y^\star, r^\star)$, and refit BT. Figure 5(a) compares this fit with the full criterion. The rankings agree strongly (Spearman $\rho = 0.968$, Kendall $\tau = 0.895$). We also rescore the same judge outputs under two further criteria, applying the informative-item filter separately under each. Against the full criterion, verdict-only correctness gives $\rho = 0.927$ ($\tau = 0.790$), and verdict plus type gives $\rho = 0.997$ ($\tau = 0.981$). `gemini-3.1-pro` and `gpt-5.5` are the top two judges under every criterion, so the diagnostic fields widen the gaps between adjacent judges without changing which judges lead.

Figure 11 repeats the with-vs.-without-type analysis inside each domain. Spearman ρ ranges from 0.969 (ML) to 0.993 (Finance) and Kendall τ from 0.885 to 0.959, with the red fits sitting below the diagonal at the top of every panel. The two criteria therefore produce consistent rankings within each domain.

A.9 Subsample Stability by Domain

Figure 12 reproduces the subsample-stability check within each domain. Spearman ρ exceeds 0.98 from the 50% subsample onward in all three domains and Kendall τ crosses 0.9 for Machine Learning and Finance at the same fraction. Biomedical is the slowest to stabilize ($\rho = 0.786$ at 10%), consistent with the tighter mid-leaderboard band for that domain reported in Appendix A.1, but joins the other two by the 50% fraction. The match-cost reduction therefore transfers to per-domain leaderboards as well, with a marginally larger minimum sample required for Biomedical.

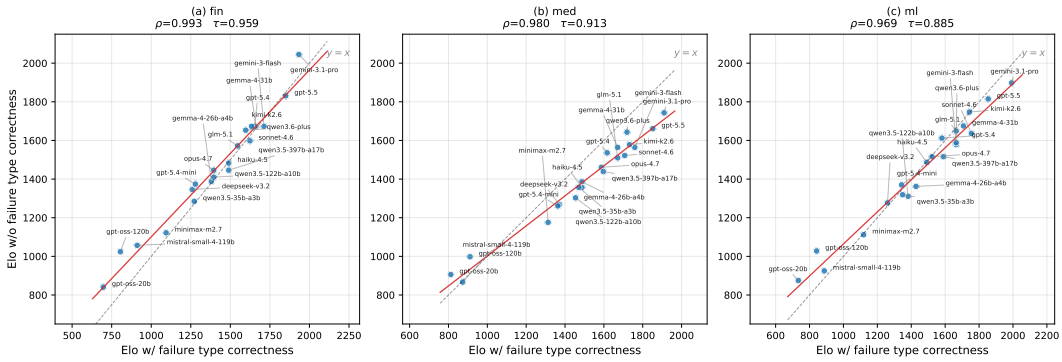


Figure 11: Per-domain version of Figure 5(a). Judge Elo with the full correctness criterion (x-axis) vs. Elo with the failure-type label dropped from the correctness criterion (y-axis), refit independently within each domain. Each marker is a judge; the dashed line marks $y = x$ and the red line is the least-squares fit. Per-panel Spearman ρ and Kendall τ are computed on the underlying Elo scores.

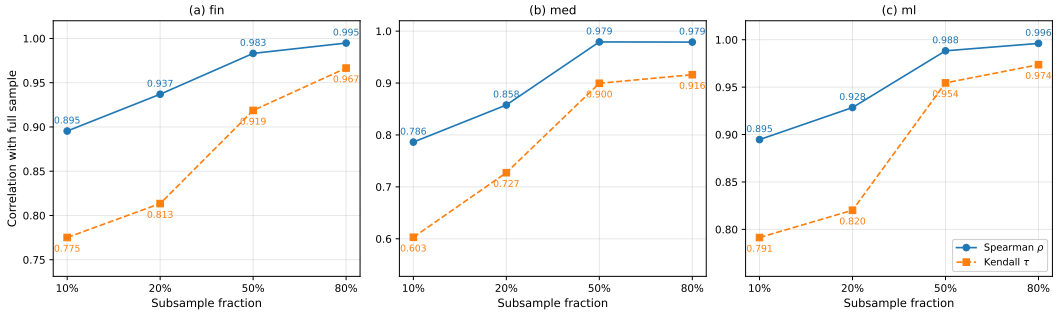


Figure 12: Per-domain version of Figure 5(c). Each panel reports Spearman ρ and Kendall τ between the full-sample BT ranking and the BT ranking refit on uniform subsamples at fractions $\{10\%, 20\%, 50\%, 80\%\}$, computed independently within each domain.

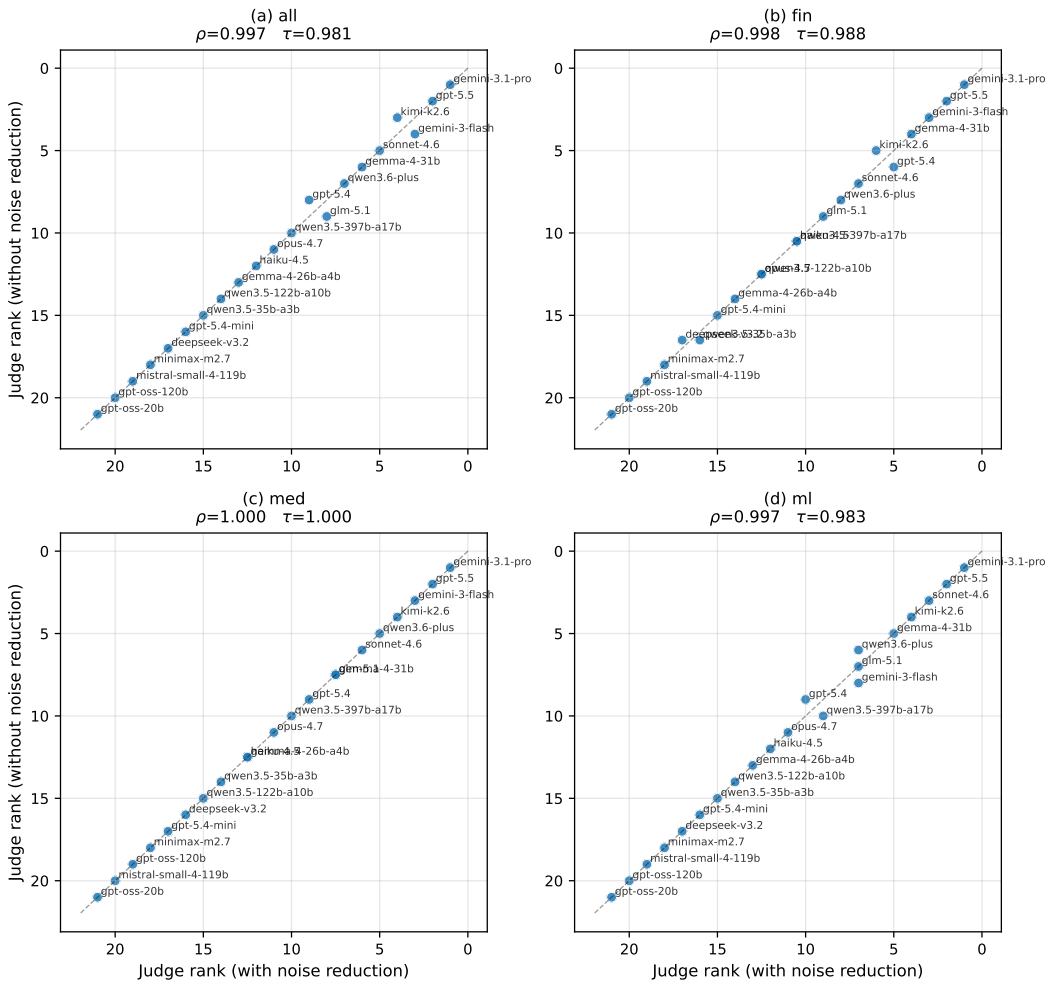


Figure 13: Judge ranks on the published slice (x-axis) and the filtered pool before release-specific curation (y-axis), one panel per domain and the combined set. The published slice excludes known audit issues and applies the top-Elo trim. Each marker is a judge; the dashed line is $y = x$. Both axes are inverted so that rank 1 sits at the top-right. Per-panel Spearman ρ and Kendall τ are computed on the underlying Elo scores.

A.10 Top-Ranked Pair Removal

The process audit and fine-tuning study motivate the top-Elo curation rule, but are not required pipeline components. For this release, we also removed the 17 known issues found during the audit. Figure 13 compares the resulting published slice with the filtered pool before these release-specific exclusions. Rankings agree closely, with Spearman $\rho \geq 0.997$, Kendall $\tau \geq 0.981$, and the top-10 set preserved on all four slices. The reported ranking is therefore stable across curation variants.

A.11 Ranking Methods: BT vs. EIP

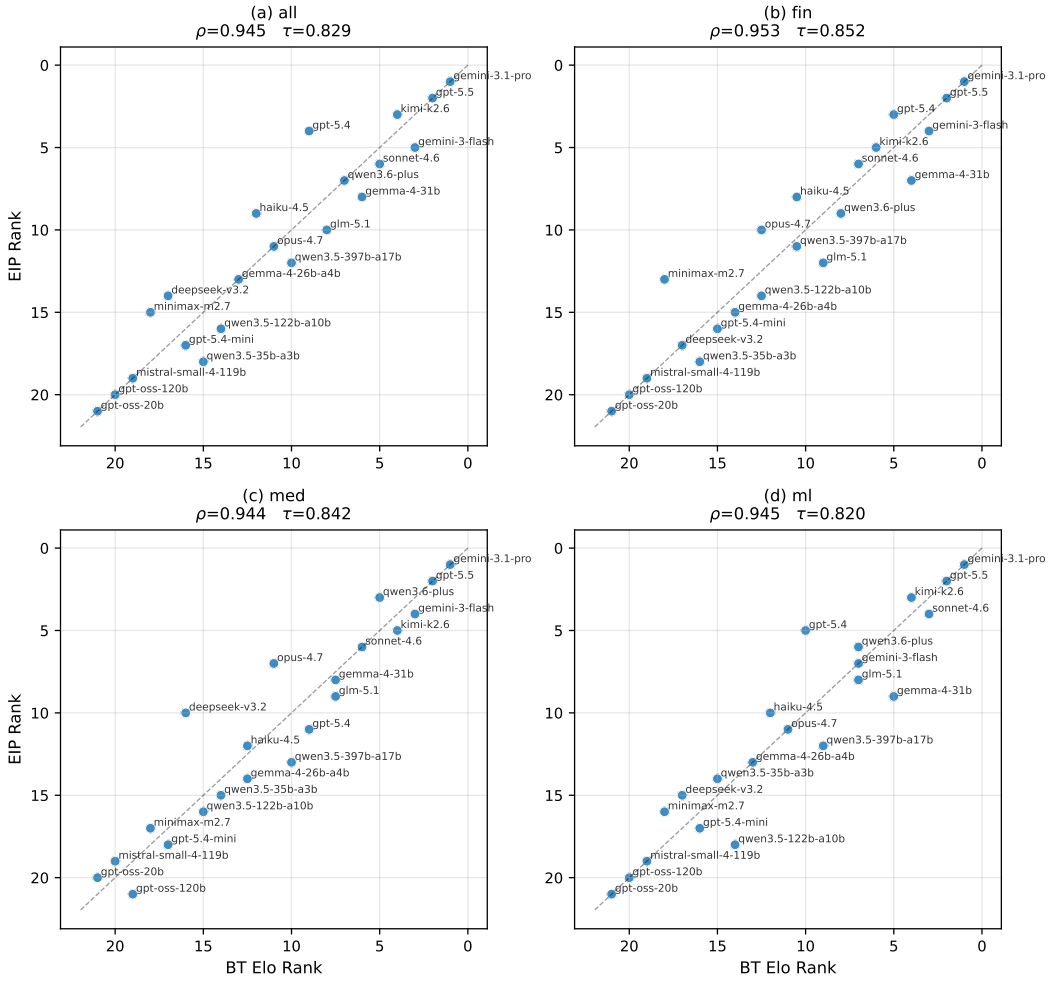


Figure 14: Judge ranks under Bradley–Terry Elo (x-axis) and Empirical Interaction Propagation (y-axis), one panel per slice. Each marker is a judge; the dashed line is $y = x$. Both axes are inverted so that rank 1 sits at the top-right. Per-panel Spearman ρ and Kendall τ are computed on the underlying scores.

Figure 14 compares the Bradley–Terry Elo leaderboard with the difficulty-aware EIP ranking of Appendix B.2, on the combined dataset and each domain. The two methods agree strongly across all four slices: Spearman ρ ranges from 0.944 to 0.953 and Kendall τ from 0.820 to 0.852, with the top-3 set always sharing two of three judges and the top-10 sets overlapping by 80% to 90%. EIP is designed to up-weight judges that solve pairs which fool strong judges, so its close agreement with Elo indicates that the benchmark does not contain a pocket of hard pairs whose signal Elo discards. We adopt Bradley–Terry Elo as the primary score throughout the paper for its closed-form cluster-robust confidence intervals (Appendix B.1) and its stability under incomplete coverage (Figure 12).

A.12 POINTWISE JUDGES

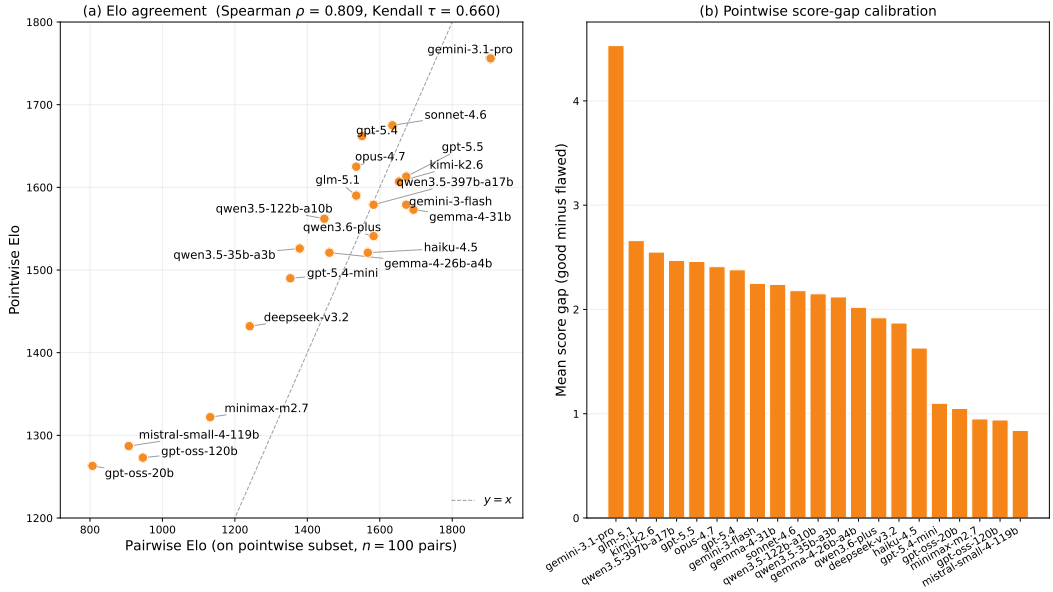


Figure 15: Pointwise vs. pairwise judging on a 100-pair stratified sample. (a) Each judge's pointwise Elo plotted against its pairwise Elo recomputed on the same pair set; the dashed line marks $y = x$. (b) Per-judge pointwise score-gap: the mean Likert score on the good conversation in each pair minus the mean score on the flawed one, sorted descending.

Setup. The pointwise prompt of Appendix B.8 differs from the pairwise default in that each judge sees a single conversation in isolation and returns a 1–10 Likert score together with an `is_flawed` flag, the worst-turn index, and the A–G flaw category. Pairwise verdicts are derived post hoc by scoring `convo_a` and `convo_b` independently and taking the higher-scoring side as the winner; equal scores are recorded as ties and given 0.5/0.5 credit so the same Bradley–Terry machinery applies. The experiment is run on a 100-pair sample stratified by domain and assistant failure type, judged by the same 21-model panel; the pairwise axis in Figure 15(a) is recomputed on the identical sample, so the only difference between the two axes is the prompt mode.

Findings. Pointwise and pairwise rankings agree at the coarse level (Spearman $\rho = 0.809$, Kendall $\tau = 0.660$) but the judge rankings differ. This divergence is expected, because the two protocols measure different judge capabilities: pairwise tests discrimination between two candidate conversations under direct comparison, while pointwise tests absolute calibration of conversation quality without a reference. Panel (b) makes the calibration axis explicit: the mean score gap between the good and flawed conversations exceeds 4 Likert points for `gemini-3.1-pro` but falls below 1 for the weakest three judges, a per-judge resolution that pairwise discards once scores collapse to a binary winner. Pairwise remains the more discriminative protocol for ranking judges; the pointwise score-gap serves as a complementary diagnostic for absolute calibration.

B Reproducibility

B.1 Bradley–Terry Elo Details

Setup, log-likelihood, and MM iteration. With notation as in Section 3.3, the BT win probability $\Pr(c = 1 \mid j, q) = \theta_j/(\theta_j + \theta_q)$ induces the log-likelihood

$$\ell(\boldsymbol{\theta}) \;=\; \sum_{(j,q,c)\in\mathcal{D}} \Big[c \log \tfrac{\theta_j}{\theta_j+\theta_q} + (1-c) \log \tfrac{\theta_q}{\theta_j+\theta_q} \Big]. \tag{2}$$

Let $w_i = \sum_{(j,q,c)\in\mathcal{D}} \big[c\,\mathbf{1}[i = j] + (1-c)\,\mathbf{1}[i = q] \big]$ denote the total wins of player i, and let $n_{ik} = \big|\{t : \{j_t, q_t\} = \{i, k\}\}\big|$ denote the total number of matches between players i and k. The standard MM update (Hunter & Li, 2005) maximizes (2) via

$$\theta_i^{(t+1)} \;=\; \frac{w_i}{\displaystyle\sum_{k\,:\,n_{ik}>0} \frac{n_{ik}}{\theta_i^{(t)} + \theta_k^{(t)}}}. \tag{3}$$

After each iteration we mean-normalize $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}/\bar{\theta}$ to fix the otherwise unidentifiable global scale, and we declare convergence when $\|\boldsymbol{\theta}^{(t+1)} - \boldsymbol{\theta}^{(t)}\|_\infty < 10^{-6}$ (or after 1000 iterations). To avoid numerical issues we floor the strengths at 10^{-10} before taking logarithms, and finally we map the strengths to the Elo scale,

$$R_i \;=\; 400 \log_{10}\big(\theta_i\big) + R_0, \tag{4}$$

with $R_0 = 1500$.

Cluster-robust sandwich standard errors. A single conversation pair contributes one match per judge, so the residuals within a pair are not independent: a pair that happens to be ambiguous will fool many judges simultaneously. We therefore report a sandwich variance estimator with clustering at the pair level. Working in the log-strength parameterization $\beta_i = \log \theta_i$, define for each match (j, q, c)

$$\eta_{j,q} \;=\; \beta_j - \beta_q, \qquad \sigma_{j,q} \;=\; \frac{1}{1 + e^{-\eta_{j,q}}}, \qquad w_{j,q} \;=\; \sigma_{j,q}\big(1 - \sigma_{j,q}\big). \tag{5}$$

The observed information matrix $I \in \mathbb{R}^{(M+N)\times(M+N)}$ accumulates the symmetric contribution

$$I_{jj} \mathrel{+}= w_{j,q}, \quad I_{qq} \mathrel{+}= w_{j,q}, \quad I_{jq} \mathrel{-}= w_{j,q}, \quad I_{qj} \mathrel{-}= w_{j,q}, \tag{6}$$

for every $(j, q, c) \in \mathcal{D}$. For each pair-cluster $p \in \mathcal{Q}$ we accumulate a score vector $\boldsymbol{s}_p \in \mathbb{R}^{M+N}$ from its residuals $r_{j,q} = c - \sigma_{j,q}$:

$$\boldsymbol{s}_p[j] \mathrel{+}= r_{j,q}, \qquad \boldsymbol{s}_p[q] \mathrel{-}= r_{j,q}, \qquad \forall (j, q, c) \in \mathcal{D} \text{ with } q = p. \tag{7}$$

The cluster-robust "meat" matrix is then

$$B \;=\; \sum_{p\in\mathcal{Q}} \boldsymbol{s}_p \boldsymbol{s}_p^\top, \tag{8}$$

and the sandwich variance on the log-strength scale is

$$V \;=\; I^+ \, B \, I^+, \tag{9}$$

where I^+ denotes the Moore–Penrose pseudoinverse, used because I has an unavoidable rank-1 deficiency from the global-shift invariance of $\boldsymbol{\beta}$ (and may have additional deficiencies if the comparison graph splits into disconnected components, which we detect and warn about). Mapping back to the Elo scale via (4) gives

$$\mathrm{SE}_{\mathrm{Elo}}(i) \;=\; \frac{400}{\ln 10} \sqrt{\big[V\big]_{ii}}, \qquad \mathrm{CI}_{95}(i) \;=\; 1.96 \cdot \mathrm{SE}_{\mathrm{Elo}}(i). \tag{10}$$

We report $R_i \pm \mathrm{CI}_{95}(i)$ for every judge and pair.

B.2 Empirical Interaction Propagation (EIP) Details

Let $\mathcal{J} = \{1, \ldots, M\}$ be the set of LLM judges and $\mathcal{Q} = \{1, \ldots, N\}$ be the set of conversation pairs. We assume full observability: every judge has been evaluated on every pair, with outcome $c_{j,q} \in \{0, 1\}$ equal to 1 iff judge j produces the correct verdict on pair q under the joint criterion of Eq. (1) (matching the ground-truth verdict $y^\star$, flawed turn $r^\star$, and failure category $t^\star$). EIP casts judge-quality scoring as a PageRank-style random walk on the bipartite graph of judges and pairs. We compute it on the *filtered* match set, where any pair on which a judge is missing or on which all judges agree (unanimously correct or unanimously incorrect) is removed; such pairs carry no discriminative signal and break the identifiability of the random walk.

Let $A \in \{0, 1\}^{N \times M}$ be the binary competency matrix, with $A_{q,j} = c_{j,q}$. Let $\mathbf{1}$ denote the $N \times M$ matrix of ones, and define the failure matrix

$$\widehat{A} \;=\; \mathbf{1} - A. \tag{11}$$

The per-pair correctness count and per-judge failure count are floored at 1 to avoid division by zero on (rare) all-correct rows or all-failed columns,

$$S_q \;=\; \max\Big(\sum_j A_{q,j},\; 1\Big), \qquad F_j \;=\; \max\Big(\sum_q \widehat{A}_{q,j},\; 1\Big). \tag{12}$$

We form two row-stochastic transition matrices on the bipartite graph,

$$P_{Q\to M}[q, j] \;=\; \frac{A_{q,j}}{S_q}, \qquad P_{M\to Q}[j, q] \;=\; \frac{\widehat{A}_{q,j}}{F_j}. \tag{13}$$

Walking from a pair, the chain hops to a judge that solved it (weighted by that judge's competence on this pair); walking from a judge, the chain hops to a pair that judge *failed* (weighted by that judge's failure mass). Iterating this walk concentrates probability mass on judges that solve hard pairs and on pairs that fool strong judges, yielding a difficulty-aware ranking.

With damping factor $\alpha = 0.85$ and uniform teleport vectors $\boldsymbol{u}_M = \mathbf{1}_M/M$ and $\boldsymbol{u}_Q = \mathbf{1}_N/N$, we run the coupled iteration

$$\boldsymbol{\pi}_Q^{(t+1)} \;=\; \alpha\, P_{M\to Q}^\top\, \boldsymbol{\pi}_M^{(t)} \;+\; (1-\alpha)\, \boldsymbol{u}_Q, \tag{14}$$

$$\boldsymbol{\pi}_M^{(t+1)} \;=\; \alpha\, P_{Q\to M}^\top\, \boldsymbol{\pi}_Q^{(t+1)} \;+\; (1-\alpha)\, \boldsymbol{u}_M, \tag{15}$$

initialized at $\boldsymbol{\pi}_M^{(0)} = \boldsymbol{u}_M$ and $\boldsymbol{\pi}_Q^{(0)} = \boldsymbol{u}_Q$. Both vectors are renormalized to sum to 1 after each step to prevent numerical drift, and we declare convergence when

$$\big\|\boldsymbol{\pi}_M^{(t+1)} - \boldsymbol{\pi}_M^{(t)}\big\|_1 + \big\|\boldsymbol{\pi}_Q^{(t+1)} - \boldsymbol{\pi}_Q^{(t)}\big\|_1 \;<\; 10^{-6}, \tag{16}$$

or after 100 iterations.

Finally, both stationary distributions are min–max rescaled to the interpretable range $[0, 100]$. For $\boldsymbol{v} \in \{\boldsymbol{\pi}_M, \boldsymbol{\pi}_Q\}$,

$$\mathrm{score}_{\mathrm{EIP}}(i) \;=\; 100 \cdot \frac{v_i - \min_k v_k}{\max_k v_k - \min_k v_k}, \tag{17}$$

with the degenerate convention $\mathrm{score}_{\mathrm{EIP}}(i) = 50$ for all i whenever $\max_k v_k - \min_k v_k < 10^{-12}$. The reported judge ranking is the descending sort of $\{\mathrm{score}_{\mathrm{EIP}}(j)\}_{j\in\mathcal{J}}$, and the corresponding pair difficulty is $\{\mathrm{score}_{\mathrm{EIP}}(q)\}_{q\in\mathcal{Q}}$.

B.3 Prompt for Assistant Failure Type Discovery

The following box gives the prompt used for automated discovery and refinement of the assistant failure type taxonomy used in Section 3.1.

Assistant Failure Type Discovery Prompt

You are an expert in multi-turn conversation evaluation, specializing in identifying the challenging scenarios, conducting failure mode and taxonomy analysis.

Your task is fourfold:
(1) Task 1: per-sample annotation. Apply an existing taxonomy to label assistant and user behaviors in multi-turn conversations
(2) Task 2: identify taxonomy gaps. Identify cases that do NOT fit the taxonomy
(3) Task 3: taxonomy refinement. Categorize the missed cases and propose refined or new categories grounded in evidence
(4) Task 4: insights generation. Produce outputs for downstream analysis, explanations for the proposal.
TAXONOMY DEFINITIONS

```
ASSISTANT_BEHAVIORS = {
  "disorganized": {...},
  "fabricated_answer": {...},
  "unnecessary_refusal": {...},
  ...
}
USER_BEHAVIORS = {
  "focused": "...",
  "skeptical": "...",
  "misinformed": "...",
  "exploratory": "...",
  ...
}
```

GENERAL GUIDELINES

```
- Be precise, not verbose; avoid generic explanations.
- Ground reasoning in observed data; do NOT hallucinate missing context.
- Prefer concrete behavioral descriptions over abstract wording. Identify
ambiguities in definitions.
- Be critical of the taxonomy when necessary, but only propose changes
supported by repeated evidence.
```

DATA EXAMPLES

B.4 Assistant Failure Types

The following boxes define the expected behaviors for assistant failure types, and the corresponding benign behaviors when no failure is present. These are the full versions of the behaviour specifications that assistants are conditioned on, corresponding to Table 2.

`self_contradiction` — Flaw

The assistant subtly contradicts a claim from an earlier correct turn — not by lexically flipping it, but by shifting a qualifier ('primary factor' becomes 'one of several factors'), reframing emphasis, or stating something whose implications conflict with what the assistant said before. Use different wording so the inconsistency isn't surface-detectable; catching it requires mentally reconciling the flawed later turn against an earlier one. The flawed round is the later turn whose claim no longer matches what was said before, while the earlier reference turn remains accurate and grounded in the source. Avoid lexically obvious flips, and do not restate the earlier claim verbatim immediately before contradicting it. Non-flawed turns stay consistent with the assistant's own prior statements.

`self_contradiction` — Benign

Stay consistent with your own earlier turns — match the scope, strength, and qualifiers of prior claims across the conversation. If your understanding genuinely evolves, explicitly acknowledge what changed and why rather than silently shifting.

`evasion` — Flaw

The chatbot produces a response that looks thorough and knowledgeable but avoids giving the user what they actually asked for. This can take two forms, or blend them: (a) surface specificity that hedges on the central claim — citing a number for a minor result while glossing the key finding, or naming a method without explaining what makes it distinct; (b) tangential drift — emphasizing adjacent or related material so the answer reads plausibly while missing the core point. Either way, the response should look reasonable to a skimming reader but leave the user unable to verify or act on the central claim. Non-flawed turns answer the user's actual question directly with the key details, exact numbers, and specific terms the question demands.

`evasion` — Benign

For each question, provide a complete, specific, on-topic answer — every sentence directly addresses what the user asked, with the key details, exact numbers, and specific terms that are relevant to the actual question.

`disorganized` — Flaw

The chatbot buries key information in text that looks similar in length and formatting to clean responses but lacks logical flow — jumping between points, mixing high-level claims with low-level details mid-paragraph, and failing to foreground the answer the user actually asked for. The flaw is structural, not visual: the response should not look like a wall of text. The correct content is present but requires significant effort to extract. Non-flawed turns should present information clearly and directly.

`disorganized` — Benign

Present information with clear structure — lead with the key point, organize logically, make the answer easy to scan and extract insights from.

`fabricated_answer` — Flaw

The chatbot fabricates a confident-sounding answer when the question is not answerable from the reference material, instead of honestly saying it doesn't have enough information. Non-flawed turns should acknowledge limitations when the question goes beyond what the reference material supports.

`fabricated_answer` — Benign

Some of the user's questions go beyond what the reference material covers. The assistant recognizes these and honestly says it doesn't have enough information rather than guessing, while answering confidently when the material does cover the question. Demonstrate clear calibration between what the source supports and what it doesn't — at least one turn should acknowledge a limit.

`instruction_forgetting` — Flaw

Early in the conversation the user establishes an explicit content-level directive — a scope restriction ('focus only on the ablation results, skip the architecture discussion'; 'only pull from the abstract, not the full methods'), an audience/framing ('I'm a clinician making treatment decisions, not a student'; 'graduate-level CS, assume transformer basics'), self-context ('I'm researching Alzheimer's specifically, not dementia broadly'; 'I've already read section 3, don't re-explain it'), a prioritization directive ('I care more about limitations than strengths'; 'methodology over results'), or a negative content instruction ('don't walk me through the reasoning, just give the conclusion'; 'don't re-cover material we already discussed'). The assistant honors the directive through the intermediate rounds. In the flawed round it silently violates the directive while otherwise answering competently — pulling in out-of-scope material 'to provide context', drifting to a different audience level, re-explaining what the user said to skip, or slipping into the deprioritized angle. The violation must be partial or plausibly-justified (not a wholesale abandonment) and must not be lexically announced — no 'let me expand beyond…', 'stepping outside the scope for a moment…'. Do not use format/stylistic directives (bullets vs prose, word counts, citation style) since those are surface-detectable and don't test reasoning. Non-flawed turns apply the directive consistently.

`instruction_forgetting` — Benign

Early in the conversation the user establishes an explicit content-level directive — a scope restriction, audience/framing, self-context, prioritization, or negative-content instruction (not surface formatting). The assistant tracks the directive and applies it consistently across every subsequent turn, even when not restated. Responses should demonstrate that the directive is actively shaping what gets included, emphasized, or omitted — not by naming the directive, but through the content choices themselves. If a later question would genuinely conflict with the directive, flag the conflict and ask rather than silently deviating.

`no_clarification` — Flaw

When the user's question is ambiguous or underspecified — e.g., it could refer to multiple parts of the material, or lacks enough context to give a precise answer — the chatbot picks one interpretation and answers confidently instead of asking the user to clarify. Non-flawed turns should ask for clarification when the question is genuinely ambiguous.

`no_clarification` — Benign

At least one of the user's questions is genuinely ambiguous or underspecified (could refer to multiple parts of the material, or lacks enough context for a precise answer). On those, the assistant asks for clarification before answering rather than silently assuming one interpretation. On unambiguous questions, the assistant answers directly without performative clarification requests.

`unnecessary_refusal` — Flaw

The chatbot unnecessarily refuses or hedges on questions that ARE answerable from the reference material, saying things like 'I don't have enough information' or 'I can't determine that' when the answer is clearly available. Non-flawed turns should answer confidently using the reference material.

`unnecessary_refusal` — Benign

The user asks questions that sit near the edge of what the source covers but are answerable if the assistant reads the material carefully. The assistant answers confidently using the available

information without unnecessary hedging or refusal. Only hedge when a question is genuinely outside the source's scope.

B.5 User Behavior Types

The following boxes define the user-behavior taxonomy used in Section 3.1. Each generated pair samples one user behavior, which conditions how the simulated user interacts with the assistant across all turns. The seven behaviors split into navigation patterns (how the user moves through the material) and epistemic stances (how the user relates to the material's content). Keys match the labels used by the pipeline.

focused

The user picks one or two specific aspects of the material and explores them in depth across multiple turns, asking follow-up questions that build on previous answers.

integrative

The user asks broad questions connecting different parts of the material - linking details across sections, comparing information, wanting synthesis rather than isolated facts.

scattered

The user jumps abruptly between unrelated aspects of the material without transitions, forcing the assistant to handle rapid context switches.

skeptical

The user plays devil's advocate - questioning claims, asking for evidence, pointing out apparent tensions between answers, and requiring justification before accepting conclusions.

misinformed

The user confidently states incorrect things about the material (wrong numbers, misattributed claims, flipped conclusions) and builds questions on these false premises. The assistant should correct these errors.

exploratory

The user treats the material as a starting point and asks about implications, broader connections, or extensions that may go beyond what the reference material covers.

underspecified

The user sometimes asks vague or ambiguous questions - using unclear references, omitting key qualifiers, or phrasing questions that could apply to multiple parts of the material - forcing the assistant to either guess the intent or ask for clarification.

B.6 Prompts for Dual Conditioned Conversation Pair Generation

The following boxes display the prompts used for generating conversations based on blueprints, as described in Section 3.1. The two prompts give instructions for generating a good conversation, and a worse conversation which intentionally has weaknesses.

good_convo

```
This conversation will be used in a pairwise comparison against a counterpart
conversation that contains a subtle "{flaw_name}" flaw in exactly one round. The
goal of the pair is to test whether an LLM judge can pick the better conversation
 --- which only works if your conversation is absolutely correct on every
dimension and naturally embodies the paired virtue, without performing it loudly.
 If your conversation is obviously stronger than a clean conversation with one
subtle flaw, the test becomes trivial. Aim to read as ordinary high quality, not
as a demonstration of the virtue.

Given this reference document, generate a natural multi-turn conversation ({n}
rounds, each round = 1 user + 1 assistant message).

The context is reference material - it defines what the assistant knows about the
 topic. The conversation does NOT need to cover all of it. A focused discussion
exploring a few topics in depth is better than superficially touching everything.

First, write a turn-by-turn plan with one labeled entry per round in the form `
Round 1: ...`, `Round 2: ...`, ..., `Round {n}: ...`. Every round MUST have its
own entry --- do not collapse rounds into a single paragraph or skip ahead. For
each round, capture (a) what the user asks and what motivates it (building on the
 prior round where natural), and (b) what the assistant's response will focus on
from the reference material. The plan should form a coherent arc, not isolated Q&
A. Then write the conversation faithfully executing the plan.

User behavior: "{user_behavior_name}" - {user_behavior_desc}

Assistant virtue to embody naturally (the quality every round must quietly live
up to --- do NOT overact it): {assistant_virtue}

Paired flaw the counterpart conversation will contain (for your awareness only
--- do NOT mention, contrast, or over-compensate against it; just never exhibit
it): {assistant_flaw}

User rules:
- Has read the reference material. Asks questions driven by genuine curiosity,
not to quiz the assistant.
- Apply the above behavior consistently across ALL rounds - it should shape what
the user asks, how they react, and how the conversation progresses.
- NOT a passive question-asker - reacts to answers, builds on them, pushes back,
or redirects.
- Keep messages SHORT and natural (1-3 sentences). Real users don't write essays.

Assistant rules:
- Every round must be clean on every dimension: accurate, grounded, specific, on-
topic, well-structured, responsive, and appropriately calibrated. No flaw
category should surface in any round.
- Answers must be grounded in the provided context. Do NOT fabricate information
beyond what the context provides.
- If a user question goes beyond what the context covers, honestly acknowledge
the limitation rather than guessing or fabricating.
- Be specific - use the actual numbers, details, and conclusions from the
reference material.
- Be concise and direct (2-5 sentences per response). Avoid filler, repetition,
and preamble. Match the tone, length, and confidence of a typical clean response
--- do not inflate quality signals to appear more virtuous.

=== Reference Context ===
{context}
```

worse_convo

```
Given this reference document, generate a natural multi-turn conversation ({n}
rounds, each round = 1 user + 1 assistant message) where the chatbot has a
specific weakness.

Chatbot weakness: **{behavior_type}**
Description: {behavior_desc}

The context is reference material - it defines what the assistant knows about the
 topic. The conversation does NOT need to cover all of it. A focused discussion
is fine.

First, write a turn-by-turn plan in the "reasoning" field with one labeled entry
per round in the form `Round 1: ...`, `Round 2: ...`, ..., `Round {n}: ...`.
Every round MUST have its own entry --- do not collapse the plan into a single
justification paragraph or skip ahead. For each round, capture (a) what the user
asks and what motivates it (reacting to the prior round where natural), and (b)
what the assistant's response will focus on from the reference material. The plan
 should form a coherent arc, not isolated Q&A. Pick one round to be the flawed
one and, inside that round's entry, describe concretely how the "{behavior_type}"
 weakness will surface there --- the clean rounds before and after should carry
the conversation as if no flaw exists. Set "bad_round_index" to that round (1-
indexed). Then write the conversation faithfully executing the plan.

User behavior: "{user_behavior_name}" - {user_behavior_desc}

User rules:
- Has read the reference material. Asks questions driven by genuine curiosity.
- Apply the above behavior consistently across ALL rounds.
- NOT a passive question-asker - reacts to answers naturally. If the chatbot
gives an evasive answer, the user may follow up, push back, or move on - whatever
 fits. Do NOT have the user explicitly diagnose or name the flaw.
- Keep messages SHORT and natural (1-3 sentences).

Chatbot (assistant) rules:
- Answers must be grounded in the provided context. Do NOT fabricate information
beyond what the context provides.
- Be specific - use the actual numbers, details, and conclusions from the
reference material.
- Be concise and direct (2-5 sentences per response). Avoid filler, repetition,
and preamble.
- In exactly 1 round, the assistant response should exhibit the "{behavior_type}"
 weakness. The flaw should require careful reading to detect --- a reader
skimming should NOT notice anything wrong. Keep the same confident tone and
similar response length as other rounds.
- The flawed round must exhibit ONLY the "{behavior_type}" flaw --- no other
weakness category should appear in it. Every dimension of the response other than
 the named flaw must remain clean: accurate, grounded, specific, on-topic, well-
structured, and responsive. All non-flawed rounds must be clean on every
dimension.
- Do NOT mention or acknowledge that responses are flawed.

=== Reference Context ===
{context}
```

B.7 Prompts for Automated Quality Control

The following boxes display the prompts used for automated quality control of generated conversations, namely for checking coherence, adherence, and grounding as described in Section 3.1. Breaking generated text into atomic claims is a common practice when verifying factuality of statements (Mohri & Hashimoto, 2024). Rather than demand exact factuality of all claims, it is also possible to relax to a specified precision level and calibrate removal of conversations using statistical techniques (Kuwahara et al., 2025).

coherence

```
You are auditing a synthetic-data recipe before any conversation has been
rendered. Given a reference document and a sampled plan, decide whether the plan
can plausibly produce a natural multi-turn conversation.

A plan is INCOHERENT only when executing the stated virtue/flaw is semantically
at odds with the user style for THIS specific reference material - not merely
unusual. Surface-level mismatches (e.g. a skeptical user asking precise questions
) are fine. Flag only genuine conflicts, e.g.:
- the flaw requires conditions the user style will never create (a "focused" user
 asking tightly scoped questions cannot surface a "no_clarification" flaw)
- the reference material lacks the content needed for the flaw to appear
naturally (e.g. "factual_error" flaw but the material is too thin to have subtle
facts to misstate)
- the bad_round_index sits outside [1, n_rounds] or collides with where the plan
claims the conversation is just getting started / wrapping up

User behavior: "{user_behavior_name}" - {user_behavior_desc}
Assistant virtue (good convo): {virtue}
Assistant flaw (bad convo, "{assistant_behavior_name}"): {flaw}
n_rounds: {n_rounds}
bad_round_index (1-indexed): {bad_round_index}

=== Good plan (from generator) ===
{good_plan}

=== Bad plan (from generator) ===
{bad_plan}

=== Reference Context ===
{context}

Respond in JSON: {"good_ok", "good_issue", "bad_ok", "bad_issue"}. Leave *_issue
as "" when ok. Keep issues to one sentence.
```

adherence

```
You are auditing whether two rendered conversations actually followed their
generation plans. Read the plans and then the conversations and answer yes/no
globally for each side.

For the GOOD conversation:
- Did the assistant consistently display the stated virtue across ALL rounds?
- Did the user maintain the stated user behavior across ALL rounds?

For the BAD conversation:
- Did the user maintain the stated user behavior?
- Does round {bad_round_index} (1-indexed) actually exhibit the "{
assistant_behavior_name}" flaw described below?
- Is that the ONLY flawed assistant round? (If another round is clearly flawed,
bad_flaw_round_correct is false.)

User behavior: "{user_behavior_name}" - {user_behavior_desc}
Assistant virtue: {virtue}
Assistant flaw "{assistant_behavior_name}": {flaw}
n_rounds: {n_rounds}

=== Good plan ===
{good_plan}

=== Good conversation ===
{good_convo}

=== Bad plan ===
{bad_plan}

=== Bad conversation (declared bad_round_index = {bad_round_index}) ===
{bad_convo}

Respond in JSON: {"good_followed", "good_issue", "bad_followed", "
bad_flaw_round_correct", "bad_issue"}. Leave *_issue as "" when ok. One sentence
per issue.
```

grounding

```
You are fact-checking an assistant's responses against a reference document. For
each listed assistant turn, extract every factual atomic claim it makes (specific
 numbers, named entities, mechanisms, cause/effect statements, attributions) and
mark each claim as grounded or not.

Rules:
- A claim is `grounded: true` only if it is directly supported by the reference
context below.
- Paraphrases and reasonable summaries of supported content count as grounded.
- Reasonable inferences or synthesis across multiple context statements also
count as grounded, even if no single sentence states them verbatim.
- Unsupported extrapolation, invented details, or claims beyond the context are `
grounded: false`.
- Ignore conversational filler, hedges, and meta-statements ("good question", "
let me explain").
- Do NOT extract negative/absence claims ("X is not Y", "X doesn't mention Z")
unless the context directly contradicts them - absence is not verifiable at claim
 granularity.
- Do NOT extract meta-statements where the assistant is evaluating another
statement rather than asserting a fact ("it is inaccurate to claim...", "that
earlier answer was wrong").
- Aim for 1-5 atomic claims per turn. Skip turns whose round_index is in
skip_rounds: {skip_rounds}.

=== Assistant turns ({label}) ===
{turns}

=== Reference Context ===
{context}

Respond in JSON: {"rounds": [{"round_index", "claims": [{"claim", "grounded"}]}]}.
 Only include rounds you actually analyzed (exclude skip_rounds).
```

B.8 Prompt for Judges and its Variants

The following boxes display the prompts used for the main judge results across all 21 models, and variants used as ablations throughout Section 4. LLM-as-a-judge systems can be sensitive to prompt phrasing. Our prompt revisions were refined manually, but future iterations of the benchmark could evaluate judge robustness using prompt-level uncertainty quantification (Ross et al., 2026).

judge

```
You are evaluating two multi-turn conversations about the same reference material
 between a user and an assistant. The two conversations may have different user
questions and different assistant responses - your job is to judge which
conversation's assistant provides better overall quality.
{context_section}
=== Conversation A ===
{text_a}

=== Conversation B ===
{text_b}

The worse conversation has exactly one problematic assistant response. Classify
the flaw using exactly one of these categories:
[The seven assistant-weakness definitions A-G from the taxonomy are inserted
verbatim here, one bullet per category in the form "- **<letter> (<name>)**: <flaw
description>".]

Quality flaws (evasion, disorganization, missed clarification, unnecessary
refusal) count as much as factual mistakes when deciding which conversation is
worse - a response can be problematic with no factual error at all. When
classifying, pick the category whose definition the flawed round most closely
matches; do not default to whichever category feels most common or most obvious.
Each of A-G is a distinct failure mode and exactly one should fit.

Respond in JSON with these fields in order:
- "analysis": your reasoning about which conversation is better and why
- "worst_round": the round number (starting from 1) containing the worst
assistant response in the conversation you did NOT pick, where each round is one
user message followed by one assistant response
- "problem_type": the letter (A-G) for the flaw category from the list above
- "verdict": "A" or "B" for the better conversation
```

judge_v2: confusion-pair hints

```
You are evaluating two multi-turn conversations about the same reference material
 between a user and an assistant. The two conversations may have different user
questions and different assistant responses - your job is to judge which
conversation's assistant provides better overall quality.
{context_section}
=== Conversation A ===
{text_a}

=== Conversation B ===
{text_b}

The worse conversation has exactly one problematic assistant response. Classify
the flaw using exactly one of these categories:
[The seven assistant-weakness definitions A-G from the taxonomy (Sec. 3.1) are
inserted verbatim here, one bullet per category in the form "- **<letter>
(<name>)**: <flaw description>".]

Quality flaws (evasion, disorganization, missed clarification, unnecessary
refusal) count as much as factual mistakes when deciding which conversation is
worse - a response can be problematic with no factual error at all. When
classifying, pick the category whose definition the flawed round most closely
matches; do not default to whichever category feels most common or most obvious.
Each of A-G is a distinct failure mode and exactly one should fit.

Additional guidance for two categories that are easy to confuse with B (evasion):
- **F (no_clarification)** requires that (i) the user's question is genuinely
ambiguous AND (ii) the substantive answer would change depending on how the
ambiguity is resolved. If an assistant asks a clarifying question and then gives a
reasonable answer in the next round (or the user does not clarify), that is
appropriate clarification behavior — it is NOT B (evasion). Only treat a missed
clarification as F when the disambiguation would meaningfully change the answer.
- **A (self_contradiction)** does not require the contradicted claim to be on the
round's main question. A claim that contradicts an earlier turn — even when the
contradiction is a tangential or seemingly random fact rather than the central
point — is still A, not B (evasion). Do not default to B just because the
inconsistency is off-topic relative to the user's current question.

Respond in JSON with these fields in order:
- "analysis": your reasoning about which conversation is better and why
- "worst_round": the round number (starting from 1) containing the worst
assistant response in the conversation you did NOT pick, where each round is one
user message followed by one assistant response
- "problem_type": the letter (A-G) for the flaw category from the list above
- "verdict": "A" or "B" for the better conversation
```

judge_v3: forced candidate comparison

```
You are evaluating two multi-turn conversations about the same reference material
 between a user and an assistant. The two conversations may have different user
questions and different assistant responses - your job is to judge which
conversation's assistant provides better overall quality.
{context_section}
=== Conversation A ===
{text_a}

=== Conversation B ===
{text_b}

The worse conversation has exactly one problematic assistant response. Classify
the flaw using exactly one of these categories:
[The seven assistant-weakness definitions A-G from the taxonomy (Sec. 3.1) are
inserted verbatim here, one bullet per category in the form "- **<letter>
(<name>)**: <flaw description>".]

Quality flaws (evasion, disorganization, missed clarification, unnecessary
refusal) count as much as factual mistakes when deciding which conversation is
worse - a response can be problematic with no factual error at all. When
classifying, pick the category whose definition the flawed round most closely
matches; do not default to whichever category feels most common or most obvious.
Each of A-G is a distinct failure mode and exactly one should fit.

Before committing to a category, you MUST explicitly compare candidates. Identify
the two or three categories from A-G that could plausibly fit the flawed round,
then for each candidate state (i) the specific evidence in the round that fits its
definition and (ii) the strongest reason it might not fit. Only after that
comparison should you commit to the single best fit, briefly explaining why it
beats the runners-up. Do not pick the first category that comes to mind without
weighing at least one alternative against it.

Respond in JSON with these fields in order:
- "analysis": your reasoning. This must include the candidate comparison
described above before you commit to a problem_type.
- "worst_round": the round number (starting from 1) containing the worst
assistant response in the conversation you did NOT pick, where each round is one
user message followed by one assistant response
- "problem_type": the letter (A-G) for the flaw category from the list above
- "verdict": "A" or "B" for the better conversation
```

judge_v4: decision rules + comparison

```
You are evaluating two multi-turn conversations about the same reference material between a user and an
assistant. The two conversations may have different user questions and different assistant responses -
your job is to judge which conversation's assistant provides better overall quality.
{context_section}
=== Conversation A ===
{text_a}

=== Conversation B ===
{text_b}

The worse conversation has exactly one problematic assistant response. Classify the flaw using exactly
one of these categories:
[The seven assistant-weakness definitions A–G from the taxonomy (Sec. 3.1) are inserted verbatim here,
one bullet per category in the form "- **<letter> (<name>)**: <flaw description>".]

Quality flaws (evasion, disorganization, missed clarification, unnecessary refusal) count as much as
factual mistakes when deciding which conversation is worse - a response can be problematic with no
factual error at all. When classifying, pick the category whose definition the flawed round most closely
 matches; do not default to whichever category feels most common or most obvious. Each of A-G is a
distinct failure mode and exactly one should fit.

Discriminating tests for the categories most often confused with B (evasion):

- **F (no_clarification) decision rule.** F applies only when BOTH conditions hold: (i) the user's
question is genuinely ambiguous or underspecified, AND (ii) the substantive answer would meaningfully
change depending on the disambiguation. If either condition fails, the round is not F. Crucially, if the
assistant *did* ask a clarifying question, the round is not F and is also not B (evasion) — asking for
clarification is appropriate behavior even when the question turns out not to need it. F means *failing
to ask when asking was needed*; it does not mean *asking unhelpfully*. Do not label a round F or B simply
because the assistant's clarifying question feels unnecessary in hindsight.

- **A (self_contradiction) decision rule.** A applies whenever the flawed round contains an assertion
that is inconsistent with something the assistant said in an earlier turn — even if the contradicted
claim is tangential, a side-remark, or a random fact unrelated to the round's main question. The
contradiction does not have to be on the central topic. If you can identify any earlier assistant
statement that the flawed round's claims are at odds with, prefer A over B (evasion). Do not default to
evasion just because the inconsistency feels off-topic or minor.

- **B (evasion) is reserved for** rounds where the assistant engages with the central question but
produces a response that hedges, drifts to adjacent material, or substitutes surface specificity for the
actual answer. If the flawed behavior is "asked an unhelpful clarifying question" → not B (consider F's
rule above; usually it is none of A-G). If the flawed behavior is "made an inconsistent claim" → prefer
A.

Before committing to a category, you MUST explicitly compare candidates. Identify the two or three
categories from A-G that could plausibly fit the flawed round, then for each candidate state (i) the
specific evidence in the round that fits its definition and (ii) the strongest reason it might not fit.
When F or B is among your candidates, you MUST apply the F decision rule by name. When A or B is among
your candidates, you MUST apply the A decision rule by name. Only after that comparison should you commit
to the single best fit, briefly explaining why it beats the runners-up. Do not pick the first category
that comes to mind without weighing at least one alternative against it.

Respond in JSON with these fields in order:
- "analysis": your reasoning. This must include the candidate comparison described above before you
commit to a problem_type.
- "worst_round": the round number (starting from 1) containing the worst assistant response in the
conversation you did NOT pick, where each round is one user message followed by one assistant response
- "problem_type": the letter (A-G) for the flaw category from the list above
- "verdict": "A" or "B" for the better conversation
```

judge_v5: structured two-stage CoT

```
You are evaluating two multi-turn conversations about the same reference material between a user and an
assistant. The two conversations may have different user questions and different assistant responses -
your job is to judge which conversation's assistant provides better overall quality.
{context_section}
=== Conversation A ===
{text_a}

=== Conversation B ===
{text_b}

The worse conversation has exactly one problematic assistant response. Classify the flaw using exactly
one of these categories:
[The seven assistant-weakness definitions A-G from the taxonomy (Sec. 3.1) are inserted verbatim here,
one bullet per category in the form "- **<letter> (<name>)**: <flaw description>".]

Quality flaws (evasion, disorganization, missed clarification, unnecessary refusal) count as much as
factual mistakes when deciding which conversation is worse - a response can be problematic with no
factual error at all. When classifying, pick the category whose definition the flawed round most closely
 matches; do not default to whichever category feels most common or most obvious. Each of A-G is a
distinct failure mode and exactly one should fit.

Before committing to a category, your "analysis" field MUST proceed in two stages.

**Stage 1 — Findings.** Once you have located the flawed round, answer the following five diagnostic
questions in order. Each answer is "Yes" or "No" followed by one short sentence of evidence drawn from
the conversation. Do not skip any question; do not collapse them into prose. These findings constrain
which categories are even eligible — you will use them in Stage 2.

  Q1. Is the user's question in the flawed round genuinely ambiguous or underspecified (could plausibly
refer to multiple distinct things in the source)?
  Q2. If Q1 is Yes: would the substantive answer meaningfully change depending on the disambiguation? (If
Q1 is No, answer "N/A".)
  Q3. Did the assistant in the flawed round ask a clarifying question (rather than committing to one
interpretation)?
  Q4. Does the flawed round contain any assertion — central or tangential — that is inconsistent with
something the assistant said in an earlier turn?
  Q5. Does the flawed round decline, hedge, or claim insufficient information for a question that the
reference material actually answers?

**Stage 2 — Comparison and commit.** Using your findings, identify the two or three categories from A-G
that could plausibly fit, and for each candidate state (i) the specific evidence in the round that fits
its definition and (ii) the strongest reason it might not fit, citing the relevant findings (Q1-Q5) by
number. The findings are operational tests:

  - F (no_clarification) is eligible only if Q1=Yes AND Q2=Yes AND Q3=No. If Q3=Yes, F is ruled out
(asking for clarification, even unhelpfully, is not the F flaw).
  - A (self_contradiction) is eligible whenever Q4=Yes, regardless of whether the contradiction is on the
round's main question or a tangential claim.
  - B (evasion) is the right call when the assistant engages with the central question but hedges, drifts,
or substitutes surface specificity for the actual answer. If Q3=Yes (assistant asked a clarifier), B is
usually ruled out for that reason. If Q4=Yes, prefer A over B.
  - D (fabricated_answer) requires the assistant to confidently produce content the source does not
support. Do not promote D to B just because the answer feels evasive.
  - G (unnecessary_refusal) requires Q5=Yes AND the question being answerable from the source.

Only after this comparison should you commit to the single best fit, briefly explaining why it beats the
runners-up. Do not pick the first category that comes to mind without going through Stages 1 and 2.

Respond in JSON with these fields in order:
- "analysis": your reasoning. This must include the candidate comparison described above before you
commit to a problem_type.
- "worst_round": the round number (starting from 1) containing the worst assistant response in the
conversation you did NOT pick, where each round is one user message followed by one assistant response
- "problem_type": the letter (A-G) for the flaw category from the list above
- "verdict": "A" or "B" for the better conversation
```

judge_pointwise

```
You are evaluating a single multi-turn conversation about reference material
between a user and an assistant. Your job is to score the overall quality of the
assistant's responses across the conversation.

The conversation may be entirely clean, or it may contain exactly one problematic
 assistant response that exhibits one of the categorized weaknesses below. You
see this conversation in isolation - there is no counterpart to compare against.
Score on its own merits.
{context_section}
=== Conversation ===
{text}

Weakness categories:
[The seven assistant-weakness definitions A–G from the taxonomy (Sec. 3.1) are
inserted verbatim here, one bullet per category in the form "- **<letter>
(<name>)**: <flaw description>".]

Quality flaws (evasion, disorganization, missed clarification, unnecessary
refusal) count as much as factual mistakes when judging quality - a response can
be problematic with no factual error at all. When classifying, pick the category
whose definition the worst round most closely matches; do not default to
whichever category feels most common or most obvious.

Scoring guide (1-10 Likert, integer):
- 10: every round is clean - accurate, grounded, specific, on-topic, well-
structured, responsive, and appropriately calibrated.
- 7-9: minor stylistic weaknesses but no real flaw in any round.
- 4-6: one round contains a clear but moderate flaw from the categories above.
- 1-3: one round contains a severe flaw, or multiple rounds are problematic.

Respond in JSON with these fields in order:
- "analysis": your reasoning about the conversation's quality, including which
round (if any) is the weakest and why
- "score": integer 1-10 reflecting overall assistant quality
- "is_flawed": true if you believe the conversation contains a problematic round
from the categories above, false otherwise
- "worst_round": the 1-indexed round number containing the worst assistant
response (always required, even if you believe no flaw is present - name the
round whose response is comparatively weakest)
- "problem_type": the letter (A-G) for the flaw category that best describes the
worst round (always required - if you do not believe there is a real flaw, name
the category that comes closest to the weakest round)
```

Table 7: Supervised fine-tuning hyperparameters. We train one LoRA adapter per Elo difficulty band (q_1: top 20%, q_2: 20–40%, q_3: 40–100%), each with 100 randomly sampled pairs, on top of a frozen `qwen3.5-4b` base. Held-out evaluation uses vLLM with guided JSON decoding against the schema in Appendix B.8.

Hyperparameter	Value
Base model and adapter	
Base model	`qwen3.5-4b`
Chat template	`qwen3_5_nothink`
Finetuning method	LoRA
LoRA rank	16
LoRA α	32
LoRA target modules	all linear layers
Precision	bf16
Data	
Examples per band	100 (ALPACA format)
Bands (q_1, q_2, q_3)	top 20% / 20–40% / 40–100% by Elo
Input	full judge prompt with reference context
Output	no-CoT JSON (`verdict`, `worst_round`, `problem_type`)
Cutoff length	8192 tokens
Optimization	
Optimizer	AdamW
Learning rate	1×10^{-4}
LR schedule	cosine
Warmup ratio	0.1
Epochs	5
Per-device batch size	2
Gradient accumulation steps	2
Effective batch size	4
Checkpointing	every epoch
Held-out evaluation	
Decoding	greedy (`temperature` $= 0$, `top_p` $= 1$)
Max generated tokens	32,768
Max model length	49,152
Constrained decoding	guided JSON against `JUDGE_SCHEMA`

B.9 SFT Details

Table 7 displays the model architecture and all the hyperparameters used for supervised fine-tuning with LoRA.

C Human Audit

C.1 Participants

The authors performed the audit themselves. The audit was carried out as part of regular research duties; no additional compensation was paid and no external annotators were recruited, so no IRB review was required. Each annotator was assigned a disjoint bundle of 20 pairs.

C.2 Annotation Tool

The pair-audit interface is a Streamlit application that surfaces every artifact relevant to a single pair on one screen. The sidebar exposes filters for domain, declared assistant behavior, and turn count, together with sort modes including ascending joint correctness (suspicious first) and per-component accuracy. The main view splits into five tabs: **Overview** (ground-truth verdict,

bad_round_index, assistant_behavior_type, user_behavior_type, and a per-judge accuracy summary), **Plan** (plan.good and plan.bad side-by-side), **Conversations** (convo_a and convo_b rendered as collapsible per-turn cards, with the injected-flaw turn opened by default and tagged with the declared assistant_behavior_type), **Verification** (per-claim grounding decisions plus the coherence and adherence gates from Section 3.1), and **Judges** (each of the 21 judges' raw verdict, picked turn, picked behavior type, and reasoning). The annotator commits one of three labels (clean, ambiguous, or noise) together with a free-text note; labels persist to a per-annotator JSON file.

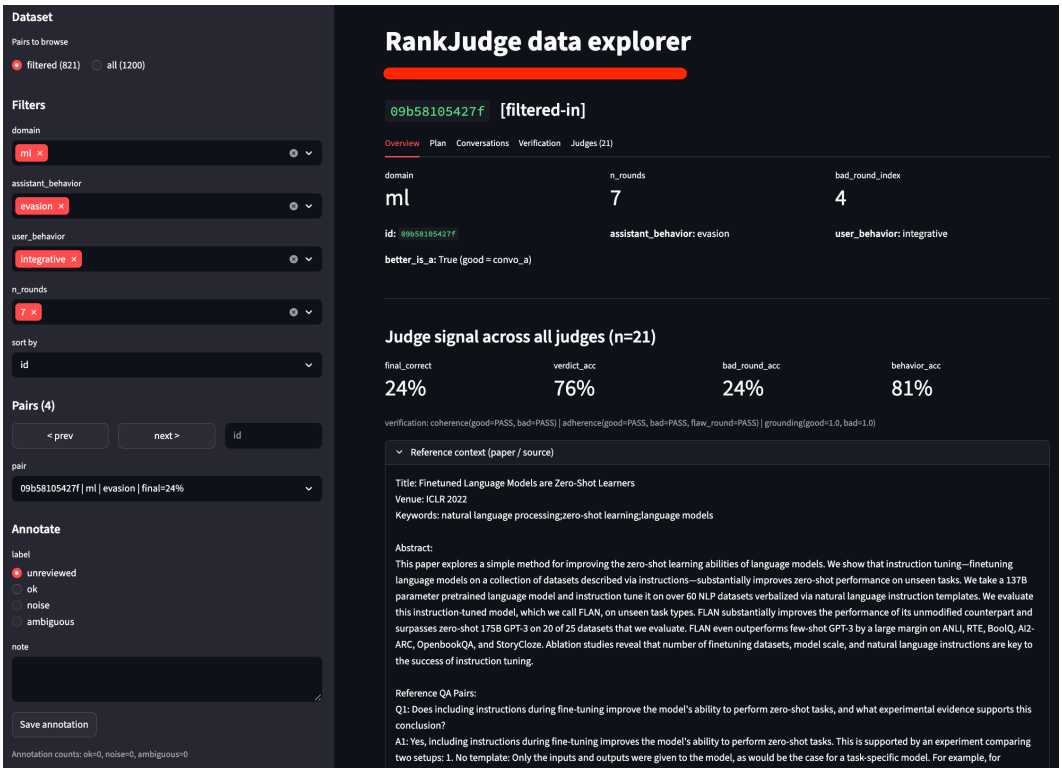


Figure 16: The pair-audit interface used for the human label-noise audit. The annotator inspects the ground-truth metadata, the good and bad plans, the two conversation transcripts (with the injected-flaw turn opened and tagged), the three verifier outputs, and each of the 21 judges' verdicts and reasoning, then commits one of three labels (clean, ambiguous, noise) along with a free-text note. The screenshot shows the Conversations tab on a representative ML pair.

C.3 Annotation Procedure

Auditing every pair end-to-end is impractical, so each annotator works through an assigned bundle of 20 pairs (two parts of 10), drawn uniformly at random from the informative items as described in Section 3.4. Each pair is processed using the following five-step procedure, and the disposition rubric at the end of the subsection maps the resulting observations to one of clean, ambiguous, or noise.

1. **Triage by judge-disagreement signal.** Open the Overview tab and scan the per-judge accuracy summary. Pairs that nearly all judges miss, or that split close to $50/50$ between the two candidate conversations, are the most likely to be mislabeled and are inspected first; pairs on which the ensemble agrees with the declared verdict are quickly skimmed and moved through.
2. **Confirm the injected flaw is on-target.** On the Plan tab, read plan.bad and verify that the planned weakness is consistent with the declared assistant_behavior_type. The plan is a single narrative paragraph rather than a turn-by-turn script, but it must be specific enough to determine which turn realizes the flaw. Hold the planned weakness in mind for

the remaining steps; if the plan describes a different category of flaw than the one declared, label `noise` immediately.

3. **Confirm the bad conversation executes the plan.** On the Conversations tab, locate the turn at `bad_round_index` in the worse conversation and verify that the assistant response realizes the planned weakness, neither a different category nor a milder version. If the worse conversation exhibits a different `assistant_behavior_type` than the one declared, or if no turn realizes the planned flaw, label `noise`.

4. **Cross-check using the judge ensemble.** Open the Judges tab. If the strongest judges in the registry (`gemini-3.1-pro`, `gpt-5.5`, `gemini-3-flash`) consistently miss the verdict while several weaker judges (`gpt-oss-20b`, `gpt-oss-120b`) get it right, treat the pair as suspicious. The strong judges' reasoning text usually points at a competing flaw in the better conversation, which is the next thing to verify.

5. **Verify the good conversation has no competing flaw.** For each strong judge that picked the wrong side: (i) locate the disputed turn in the better conversation on the Conversations tab, (ii) retrieve the relevant span from `metadata.context` (the source document), and (iii) consult an external LLM with a focused fact-check prompt that contains only the disputed turn, the matching source span, and the user's preceding question. Verify the LLM's answer by hand against the source. This step is the most expensive and is the rate-limiting one on MED and FIN pairs, where domain expertise is needed to resolve subtle factual disputes.

The disposition rubric maps observations to labels: `clean` when the injected weakness is the only material flaw and both conversations are otherwise faithful to their plans and to the source; `ambiguous` when failure could fit more than one type defined in the taxonomy; and `noise` when there was any deviation from the intended label $(y^\star, r^\star, t^\star)$. Examples of `noise` include when the better conversation contains any clear flaw, the worse conversation has no clear flaw or a flaw in the wrong turn, or when the failure type is different than $t^\star$. The label-noise rates surfaced by this procedure are reported in Section 3.4 and motivate the top-Elo trim of the published evaluation slice.

D EXAMPLES

We illustrate the joint correctness criterion on two pairs labeled `clean` by the audit of Section C. The first is presented in full (source context, both plans, both transcripts, and a curated subset of judge responses) to make every component of the criterion visible. The second uses a single 21-row table to show how the criterion partitions the full judge ensemble on a representative pair.

D.1 FULL EXAMPLE: UNNECESSARY REFUSAL IN ML

The pair (`id = 2d63594b210e`) is drawn from the ML domain, with `user_behavior_type = underspecified` and `assistant_behavior_type = unnecessary_refusal`. The injected flaw lives in the second assistant turn (`bad_round_index = 2`); `convo_b` is the better conversation (`better_is_a` = FALSE).

Source. The reference document is the ICLR 2022 submission *MCL-GAN: Generative Adversarial Networks with Multiple Specialized Discriminators*. The relevant content for the second turn is the paper's description of the flag variables $v_{i,m}$ and $u_{j,m}$, which are set by the k-largest discriminator scores per sample and are explicitly *not* included in backpropagation; a balance loss prevents collapse onto a single discriminator.

Plans. The full text of both plans is shown in the two boxes below. The behavioral divergence between them is highlighted in red: turn 2 is where the injected flaw lives, and turn 3 picks a different follow-up topic in each plan (GMAN comparison vs. ADA trick) while keeping both planned responses clean.

plan.good

Turn 1. The user asks whether MCL-GAN was only validated in small or toy settings, motivated by concern about mode-collapse methods not scaling. The assistant will point to the paper's three-task experimental coverage: unconditional image generation, image-to-image translation, and text-to-image synthesis, including high-resolution conditional synthesis and comparisons with strong baselines.

Turn 2. Building on that, the user asks vaguely how the discriminators actually specialize and whether the flag variables are learned. The assistant will explain the MCL k-max selection rule for v and u flags, that they are not backpropagated through, and how the balance loss prevents collapse to one discriminator.

Turn 3. The user pushes on comparison fairness, asking why GMAN was emphasized and other multi-discriminator methods were not central. The assistant will explain that GMAN is the closest baseline for testing specialization, describe the matched DCGAN setup and GMAN variants, and mention the broader comparisons including Inclusive GAN and Albuquerque et al. with the 42.6% memory result.

plan.bad

Turn 1. The user vaguely asks whether the method was only validated in small/simple settings or in broader ones. The assistant will answer cleanly that the paper evaluated MCL-GAN across unconditional image generation, image-to-image translation, and text-to-image synthesis, including high-resolution conditional synthesis and comparisons to multiple baselines including SOTA models.

Turn 2. Reacting to the training explanation, the user asks about "those little indicator things," ambiguously referring to the flag variables. The assistant will correctly identify them as $v_{i,m}$ and $u_{j,m}$, explain that they are set by the k-largest discriminator scores and used to update only a subset of discriminators, but will unnecessarily refuse on the clearly available point of whether they are included in backpropagation by saying it cannot determine that from the material.

Turn 3. The user moves on with another underspecified question about "the ADA trick." The assistant will answer cleanly that StyleGAN2-ADA discriminator augmentation was not used because the authors followed the StyleGAN2/Inclusive GAN setup for fair comparison and to isolate the effect of augmentation, keeping the setting equivalent.

Conversation A (worse)

User. Was this only shown on small-ish setups, or did they test it on broader stuff too?

Assistant. They tested it beyond small settings. The experiments covered three tasks: unconditional image generation, image-to-image translation, and text-to-image synthesis. For unconditional generation they compared against multiple baselines, including a SOTA model, and the conditional synthesis tasks were used to check applicability to high-resolution image generation.

User. Got it. And those little indicator things in the training bit, are they actually learned or just picked somehow?

Assistant. **injected:** `unnecessary_refusal`

Those are the flag variables $v_{i,m}$ and $u_{j,m}$. They are set to 1 when discriminator D_m has one of the k-largest scores, so they select which discriminators handle each sample under the MCL-style k-max rule. I can't determine from the provided material whether those flags are included in backpropagation, but their role is to update only a subset of discriminators and keep training efficient.

User. Okay. Also, why did they leave out that ADA trick then?

Assistant. They did not use the discriminator augmentation from StyleGAN2-ADA to keep the comparison fair. The authors followed the StyleGAN2 setup used by Inclusive GAN, which compared mode-collapse methods with the same backbone. Leaving out ADA helped isolate the effect of augmentation and maintain an equivalent experimental setting.

Conversation B (better)

User. I'm trying to tell if this was just a small-setting fix. Did they actually test it beyond plain toy image generation?

Assistant. Yes. The paper evaluates MCL-GAN across three tasks: unconditional image generation, image-to-image translation, and text-to-image synthesis. For unconditional generation, it compares against multiple baselines including a state-of-the-art model, and the conditional synthesis experiments are used to show applicability to high-resolution image generation rather than only small settings.

User. Okay, but mechanically, how do the discriminators avoid all doing the same thing? Those v and u flags seemed important; are they learned?

Assistant. They are not learned by backpropagation. The flags $v_{i,m}$ and $u_{j,m}$ are set to 1 when discriminator D_m's score is among the k largest values for that sample, so only a selected subset of discriminators is updated. This k-max MCL rule encourages different discriminators to specialize on different subsets of the real data. The balance loss also helps by preventing the selection frequency vector $\hat{q}$ from becoming too sparse, which would effectively collapse the setup back to a single discriminator.

User. Then why lean so much on GMAN instead of all the other multi-discriminator GANs? Was that comparison actually apples-to-apples?

Assistant. GMAN was chosen because its configuration is closest to MCL-GAN for testing whether explicit discriminator specialization helps. The comparison used the same DCGAN architecture and the official GMAN code, with the best GMAN score reported across GMAN-1, GMAN-0, and GMAN*, using dropout rates in the 0.4 to 0.6 range and split mini-batches to differentiate discriminators. The paper also compares against mode-collapse methods in Tables 2 and 3, including Inclusive GAN results that outperform Dist-GAN, PacGAN, VAEGAN, and VEEGAN. For Albuquerque et al.'s multi-discriminator method, they reimplemented it on CIFAR-10 and report better performance while using only 42.6% of its memory.

Judge responses. We show six of the 21 judges, chosen for spread across the leaderboard. Each block reports the judge's verdict, picked turn, picked behavior type, and a trimmed excerpt of its raw analysis. A check mark indicates joint correctness on this pair.

`gemini-3.1-pro` (Elo 1959). ✓
Verdict: **B** *Picked turn:* 2 *Picked type:* `unnecessary_refusal`.
Analysis (excerpt). Conversation B is better because Conversation A contains an unnecessary refusal. In turn 2 of Conversation A, the user asks if the indicator variables are learned, and the assistant claims it cannot determine from the provided material whether the flags are included in backpropagation. However, the reference explicitly states that they are not included in backpropagation. Conversation B correctly identifies this information and provides a comprehensive and accurate answer.

`gpt-5.5` (Elo 1867). ✓
Verdict: **B** *Picked turn:* 2 *Picked type:* `unnecessary_refusal`.
Analysis (excerpt). Conversation B is better overall. It answers each user question directly and accurately using the reference material. In particular, it correctly states that the flag variables are not learned/backpropagated, explains the k-max selection role, and adds the balance-loss connection. Conversation A is mostly good, but its second assistant response unnecessarily says it cannot determine whether the flags are included in backpropagation, even though the reference explicitly says they are not.

`kimi-k2.6` (Elo 1713). ✓
Verdict: **B** *Picked turn:* 2 *Picked type:* `unnecessary_refusal`.
Analysis (excerpt). Conversation B is the better conversation. Its assistant gives precise, well-supported answers that directly address the user's questions without hedging on answerable points. Conversation A is mostly correct, but in turn 2 the assistant correctly explains that the flags $v_{i,m}$ and $u_{j,m}$ are set by the k-max rule yet incorrectly states it cannot determine from the provided material whether those flags are included in backpropagation. The reference material explicitly states they are not, so this is an unnecessary refusal on a clearly answerable question.

`opus-4.7` (Elo 1510). ✓
Verdict: **B** *Picked turn:* 2 *Picked type:* `unnecessary_refusal`.
Analysis (excerpt). Conversation B answers all three questions confidently and accurately based on the reference material. Conversation A's second turn unnecessarily refuses to answer whether the flag variables are included in backpropagation, when the reference material clearly states they are not included in backpropagation. This is an unnecessary refusal.

`gpt-oss-120b` (Elo 867). ✓
Verdict: **B** *Picked turn:* 2 *Picked type:* `unnecessary_refusal`.
Analysis (excerpt). Conversation B provides accurate and complete answers to all user questions, directly citing information from the reference material. Conversation A has a flawed response in the second turn where the assistant claims it cannot determine whether the flag variables are included in backpropagation, despite the reference explicitly stating they are not. This amounts to an unnecessary refusal to answer a question that is answerable from the given material.

`gpt-oss-20b` (Elo 762). ×
Verdict: **B** *Picked turn:* 2 *Picked type:* `evasion`.
Analysis (excerpt). Conversation B provides accurate, comprehensive answers that directly address each user question and align precisely with the reference material. Conversation A, while generally correct, includes a problematic response in turn 2 where the assistant states uncertainty about whether the flag variables are included in backpropagation, despite the reference clearly stating they are not back-propagated. This omission fails to answer the user's question directly and introduces an inaccuracy.

The five other judges shown identify the same flaw and label it `unnecessary_refusal`; the weakest judge in the registry (`gpt-oss-20b`) still picks the correct conversation and the correct turn but

mislabels the flaw type as evasion, so it scores zero on the joint criterion despite a defensible reading. This pattern, weak judges getting the verdict but missing the type, is the dominant failure mode the joint criterion is designed to surface.

D.2 Compact Example: Evasion in Fin

The pair (id = 4b6d7345f058) is drawn from the Fin domain. The reference document is Monsanto Company's 2010-10-27 10-K filing (SIC 2870). The user takes an integrative stance, asking the assistant to connect Monsanto's two segments (Seeds and Genomics; Agricultural Productivity) through their risk factors. convo_a is the better conversation (better_is_a = TRUE); the injected evasion flaw is in the second assistant turn of convo_b, which sounds substantive (naming competition, IP, Roundup, and the 37%/57% foreign-origin sales) but evades the user's central request to spell out the concrete mechanisms (seed-saving in non-hybrid crops, inconsistent jurisdictional IP enforcement, generic glyphosate supply, distributor channel inventory, and margin erosion). Full transcripts are available in the released pairs_filtered.json.

Table 8 reports each of the 21 judges' joint correctness on this pair, partitioned into the verdict, the picked bad turn, and the picked behavior type. Every judge picks evasion as the flaw type, so the only disagreement is on the verdict and on which turn carries the flaw; six judges localize the flaw to the third turn and one to the first, and four miss the verdict outright. The pair scores $13/21$ joint-correct, close to the median $(15/21)$ of the audited sample and a representative target for the audit procedure.

Table 8: Per-judge joint correctness on the compact Fin example (id = 4b6d7345f058, assistant_behavior_type = evasion, bad_round_index = 2). Judges are listed in descending Elo order on the published evaluation slice. Verdict = picked convo_a; Turn = picked turn 2; Type = picked evasion; Joint = all three correct simultaneously.

Judge	Elo	Verdict	Turn	Type	Joint
gemini-3.1-pro	1959	✓	✓	✓	✓
gpt-5.5	1867	×	✓	✓	×
gemini-3-flash	1724	✓	✓	✓	✓
kimi-k2.6	1713	✓	✓	✓	✓
sonnet-4.6	1706	✓	✓	✓	✓
gemma-4-31b	1695	✓	✓	✓	✓
qwen3.6-plus	1671	✓	✓	✓	✓
glm-5.1	1637	✓	×	✓	×
gpt-5.4	1631	✓	✓	✓	✓
qwen3.5-397b-a17b	1571	✓	×	✓	×
opus-4.7	1510	✓	✓	✓	✓
haiku-4.5	1505	✓	✓	✓	✓
gemma-4-26b-a4b	1444	✓	×	✓	×
qwen3.5-122b-a10b	1422	×	×	✓	×
qwen3.5-35b-a3b	1373	×	×	✓	×
gpt-5.4-mini	1342	✓	✓	✓	✓
deepseek-v3.2	1307	✓	✓	✓	✓
minimax-m2.7	1184	×	×	✓	×
mistral-small-4-119b	906	✓	✓	✓	✓
gpt-oss-120b	867	✓	×	✓	×
gpt-oss-20b	762	✓	✓	✓	✓