Why LLM-Judge Agreement Is Overstated, and How to Fix It

Anonymous authors
Paper under double-blind review

Abstract

To show that automatic grading can be trusted, papers report how often two language-model judges gave the same verdict. Between a judge's reply and that number sits code. A parser that cannot read a reply writes in a default. A tie is broken to a fixed side. An answer that refuses, or is too short, is marked wrong before any judge sees it. The same code runs for every judge, so wherever it fires the judges agree by construction rather than by judgement, and the reported agreement is partly its own work. We read 35 public judging pipelines and find such code in 28 of them (80%), a parser default most often; none reports agreement on the rows it left alone. Applying each kind of rule to 38,601 recorded verdicts, from nine judges on six human-labelled benchmarks, we find that length rules raise Cohen's κ by a median of +0.071 and significantly in 78% of judge pairs, while a parse-failure default moves it either way by up to 0.08 according to which verdict the parser writes. Those same length rules lower agreement with the human labels. Agreement on the untouched rows exposes 87% of the inflated cases and costs nothing to report, so we ask that papers report it, together with the share of rows their code decided.

1 Introduction

A published case report (Anonymous, 2026) gives the cleanest example we know of. Two LLM judges scored the same 600 answers from a memory benchmark, and the reported Cohen's κ between them was 0.714. The pipeline had a rule: any answer beginning with a refusal phrase was marked wrong before either judge ran. The rule wrote both judge columns on 139 of the 600 rows. On the 461 rows it did not touch, κ was 0.064. The judges had not agreed on the rows that produced the number; a filter had written the same value into both columns.

That pipeline is not unusual. Almost every public harness that calls a judge has a parser, and almost every parser has a default. Some drop rows the judge could not be read on; some break ties to a fixed side; some score by rule before the judge runs. Each step is deterministic, applied identically to every judge, and therefore forces agreement wherever it fires. The question this paper asks is how much of the inter-judge agreement that papers report is produced this way, and whether a reader can tell.

We answer it in two parts. The first is a field audit: 42 public repositories that call an LLM judge, cloned at pinned commits and read by hand, with every override recorded by file and line. The second is a pre-registered measurement: six benchmarks with human labels, nine judges, and every category of override from the audit applied analytically to the recorded verdicts, so that each cell is a counterfactual on identical judge output. Cohen's κ before and after, decomposed into the part that comes from forced agreement and the part that comes from shifted class marginals, with bootstrap intervals over items.

Three findings. Overrides are the norm, not the exception: 80% of coded pipelines contain one, and no pipeline reports agreement on the rows it leaves alone. The size and even the sign of the effect depend on the rule: length rules inflate κ substantially; refusal rules do nothing on these benchmarks; a parse-failure default raises or lowers κ according to the constant the parser writes, so two pipelines with the same judge replies can reach opposite

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.

code	category	typical form	applies
R	refusal rule	answer matches "I don't know" → WRONG	row-wise, before judging
L	length / truncation	short or unterminated answer → WRONG; shorter response loses	row-wise
D	unresolved deixis	"yesterday", "last week" with no anchor → WRONG	row-wise
P	parse-failure default	$\perp \to$ a fixed label, tie, or 0	per judge column
T	tie-break	explicit or score tie → fixed side	per judge column
X	exclusion	rows dropped from the denominator	either
F	format gate	substring or last-match parsing that manufactures a label	per judge column

Table 1: Override categories used in the audit and the experiment.

conclusions about agreement. And the inflation is detectable from two numbers that no audited pipeline reports: the share of rows written by rules and κ on the remainder.

Contributions. (i) A definition of deterministic overrides with a row-wise/column-wise distinction that determines whether a rule forces agreement (§2). (ii) A coded audit of 42 public LLM-judge pipelines with file-and-line evidence (§3). (iii) A pre-registered experiment, 944 cells, with five hypotheses stated before any verdict was read, two of which failed as stated and are reported as such (§4–5). (iv) A reporting recommendation that costs two numbers and catches every inflated cell in our data (§6).

2 Deterministic overrides

Let a benchmark have items $i = 1..n$ and a label space $\mathcal{C}$ (for pairwise tasks $\{A, B, \text{TIE}\}$; for correctness and hallucination tasks two classes). A judge J produces a reply r_i^J, and a parser maps it to $v_i^J \in \mathcal{C} \cup \{\perp\}$, where $\perp$ is a parse failure. A deterministic override is a function that writes a value into a judge column without the judge's verdict determining that value. We use seven categories, fixed before the audit (Table 1).

The distinction in the last column matters more than the categories. A rule on the item text (R, L, D) runs before judging and every judge column inherits the written value, so on every row it touches the two columns are equal by construction. A rule on a judge's reply (P, T, F) is applied to each column where that judge's own reply triggers it; it forces agreement only on rows where both judges trigger it, and elsewhere it moves one column's marginal. Our first implementation applied reply rules from judge 1's reply to both columns. That overstated forced agreement, and no number from it is reported; the correction is dated in the pre-registration (Appendix A).

Why κ moves. With observed agreement p_o and chance agreement $p_e = \sum_c p_A(c)\, p_B(c)$, $\kappa = (p_o - p_e)/(1 - p_e)$ (Cohen, 1960). A row-wise override that writes value w on a fraction f of rows changes both terms: p_o rises because the written rows now agree (the forced channel), and the marginals $p_A(w), p_B(w)$ rise, which raises p_e and pulls κ down (the marginal channel). We separate the two by holding one term at its pre-override value: κ_{forced} uses post-override p_o with pre-override p_e, and κ_{marg} the reverse. Which channel dominates is an empirical question, and the marginal channel is the prevalence effect that Feinstein & Cicchetti (1990) and Byrt et al. (1993) describe for a single rater pair.

A closed form. If the written rows were a representative sample of the pre-override table, the post-override κ would follow from the pre-override table and f alone: $p'_o = (1-f)p_o + f$ and $p'_X(c) = (1-f)p_X(c) + f \mathbb{1}[c = w]$. A reader with only the headline table and f could then undo the override. We test whether this holds (H4). It does not, in an informative direction.

code	category	pipelines	share
P	parse-failure default	23	66%
X	silent exclusion	8	23%
T	tie-break	7	20%
F	format gate / lenient parse	7	20%
R	refusal rule	0	0%
L	length rule	1	3%
N	none found	7	20%

Table 2: Override prevalence in the 35 judge pipelines with a decided code (of 42 repositories; four negative controls and three undecided excluded). A pipeline may carry several codes.

Two diagnostics. κ_U is κ on the rows the override does not touch; it is by construction identical before and after, and it is what the judges themselves agree on. The constant-rater control replaces one judge with its majority class and reports raw agreement, a cheap check that high agreement is not a prevalence artefact.

3 Field audit of public pipelines

Method. We searched GitHub for LLM-judge harnesses and benchmarks, followed code links from the papers of the benchmarks used in §4, and snowballed to the memory-system evaluations that motivated the study. Inclusion: public code that produces per-item verdicts from an LLM judge, or the official harness of a benchmark that uses one, with a commit between January 2023 and September 2026. Four repositories with no LLM judge (MTEB, vLLM, AgentBench, τ-bench) were coded as negative controls. Each repository was cloned at a pinned commit, grepped with a fixed seed-pattern list, and every judge call site was read by hand; grep located candidates and never decided a code. One row per repository, every row citing file and line at the pinned commit (Appendix C). The full protocol is in the released archive.

Result. Of 35 judge pipelines, 28 (80%) contain at least one override (Table 2). The parse-failure default is in two thirds of them. Representative mechanisms, all at the pinned commits:

- RewardBench (Lambert et al., 2024), scripts/run_generative.py:366-372: a parser "error" scores 0.5, "effectively a tie". A judge that never emits a verdict scores 50%.
- JudgeBench (Tan et al., 2025), utils/judges.py:473,525,564: 'A>B' if score1 > score2 else 'B>A'; an exact score tie is written as B.
- OpenAI simple-evals SimpleQA (Wei et al., 2024), simpleqa_eval.py:125-126: re.search(r"(A|B|C)", text), so the first capital A, B or C anywhere in the grader's text is the grade; a reply that opens "Based on…" grades as B (incorrect). No match defaults to NOT_ATTEMPTED and leaves the accuracy denominator.
- HaluEval (Li et al., 2023), evaluation/evaluate.py:188-206: substring test for Yes/No; a reply containing both or neither counts as incorrect.
- FastChat MT-Bench (Zheng et al., 2023), common.py:263-304, show_result.py:49,64: error rows are excluded from the win rate; swap-inconsistent verdicts are collapsed to a tie.
- PandaLM (Wang et al., 2024) and JudgeLM (Zhu et al., 2025): an unparseable first line is a tie. AlignBench (Liu et al., 2024): parse failure is -1, averaged into the score. LongMemEval (Wu et al., 2025), evaluate_qa.py:113: label = 'yes' in eval_response.lower(). The mem0 and Zep LoCoMo scripts (Chhikara et al., 2025; Rasmussen et al., 2025): anything not exactly CORRECT is wrong, including a malformed JSON reply.
- Counter-examples: Inspect (UK AI Security Institute, 2024), scorer/_model.py:312-335, treats an unparseable reply as "a scoring failure rather than an incorrect answer"; HELM (Liang et al., 2023) and TruLens keep the failure as its own value or raise.

Two observations. The refusal rule that motivated this study appears in none of the public code we read, and a length rule in one: WildBench (Lin et al., 2024) writes the verdict itself when a response is empty (src/eval.py:366-373, the judge is not called) and offers an optional length-margin tie in its Elo script. Both were choices in our own earlier pipeline. What is everywhere is the parser default, usually a single constant in a helper function. We read the paper of each of the 23 benchmark pipelines for any statement of how parse failures, ties or exclusions are handled: none states its parser default; 7 disclose a related rule (MT-Bench, PandaLM and JudgeLM call swap-inconsistent verdicts a tie; JudgeBench reports invalid decisions per judge; WildBench its length margin; AlignBench reports agreement without ties; Auto-J discards unformattable predictions) and 16 say nothing. No repository reports agreement on the untouched rows or the share of rows a rule wrote.

4 Experiment

The design, hypotheses, benchmarks, judges, filters and statistics were frozen in a dated pre-registration on 8 September 2026, before the first full run. Every departure is dated with its reason in Appendix A; the ones that changed a reported number are named in the text.

Benchmarks. Six public benchmarks with human or gold labels, capped at 1,000 items by stratified sampling (seed 42) where larger: MT-Bench human judgments (Zheng et al., 2023) (1,000 pairs), LLMBar (Zeng et al., 2024) (419), RewardBench (Lambert et al., 2024) (1,000), JudgeBench (Tan et al., 2025) (270), HaluEval QA (Li et al., 2023) (1,000: 500 questions, each with its right and its hallucinated answer), and 600 LoCoMo (Maharana et al., 2024) judgments from our earlier memory-system study (150 questions $\times$ four systems, two of which are our own). The first four are pairwise with label space $\{A, B, \text{TIE}\}$; HaluEval asks whether an answer hallucinates; LoCoMo asks whether an answer is correct given a reference.

Judges. Nine, from five model families. Local, via Ollama at temperature 0, seed 42, 160-token output cap: Qwen3 8B, Gemma 4, Llama 3.1 8B, Mistral 7B, Qwen3 32B. API: Gemini 2.5 Flash-Lite and Gemini 2.5 Flash (thinking budget 0), Claude Haiku 4.5 (temperature 0, thinking off) and Claude Sonnet 5 (thinking off; the model rejects sampling parameters). The pre-registration named DeepSeek-R1 8B; under the fixed prompt it emits a reasoning block that the Ollama switch does not disable, so within any bounded budget it returns no verdict, and it was replaced by Mistral 7B before the first full run. The local tier moved mid-run from an Apple M1 Max (Ollama 0.33.2, Metal) to a Google Cloud g2-standard-8 with one NVIDIA L4 (CUDA) because the laptop could not carry the load; same tags, quantisation, prompt, seed and temperature. A same-hardware repeat of one judge on LoCoMo gave 595/600 byte-identical replies; the planned cross-hardware check was not run before the VM was deleted and is reported as not done.

Prompt and parsing. One fixed prompt per task type asks for a single token (Appendix B). Two verdict columns are recorded for every reply and neither coerces: strict requires the entire reply to be one allowed token; lenient accepts the strict token or an explicit verdict statement matched by a regex list frozen before the first full run. Anything else is $\perp$, kept as its own class. The primary analysis uses the lenient column. The API adapters initially capped output at 64 tokens while the local ones allowed 160; on JudgeBench, Claude Haiku 4.5 and Gemini Flash-Lite reason before answering, and 192 and 41 replies respectively were cut at exactly 64 tokens before any verdict. The cap was raised to 160 for parity and only the unparsed rows re-run; with greedy decoding the first 64 tokens do not depend on the cap, so complete verdicts were unchanged. Haiku still emits no verdict on 70% of JudgeBench items at 160 tokens. That is kept as measured behaviour under the fixed prompt, not corrected by re-prompting, and it is the extreme case for the parse-failure override.

Overrides. Applied analytically to the recorded verdicts, never by re-judging, so every cell is a counterfactual on identical judge output. For the two-class tasks: refusal prefix, no terminal punctuation, length under 30 characters, unresolved relative time expression (all

code	override	cells	$\Delta\kappa > 0$	CI> 0	CI< 0	Holm+	med. $\Delta\kappa$	share written
R	refusal rule	108	51	15	0	1	+0.000	0.001
L	length / truncation	388	346	301	19	248	+0.071	0.275
D	unresolved deixis	72	59	30	3	20	+0.008	0.084
P	parse-failure default	240	133	37	36	11	+0.001	0.029
T	tie-break	136	108	53	12	20	+0.020	0.144
0	control (writes no row)	216	0	0	0	0	+0.000	0.000

Table 3: Change in Cohen's κ per (benchmark, judge pair, override) cell, by override category. CI columns count cells whose 95% bootstrap interval excludes zero; Holm+ counts positive cells surviving Holm–Bonferroni over all 944 cells.

row-wise, writing WRONG or YES); parse failure written each way (column-wise). For pairwise tasks: a refusal in A writes B and vice versa; a response under half the other's length loses; parse failure writes TIE or A; an explicit TIE is broken to A. A control predicate matches no row and must leave every number unchanged.

Statistics. Cohen's κ over the benchmark's label space with $\perp$ as a class unless an override writes it; κ_U; the forced and marginal decompositions; the closed form; the constant-rater control; 2,000-resample bootstrap intervals over items on every difference. The preregistration said 10,000 resamples; 2,000 was used for run time, and the empirical interval half-widths are reported per benchmark (Appendix E) so the resolution is visible. Holm–Bonferroni is applied within H2 over all cells using a normal approximation to each cell's bootstrap distribution, because the corrected threshold is below the resolution of a 2,000-resample p-value. Cells where either judge parses under 80% of items are excluded from the hypothesis checks and reported separately; this threshold was added after seeing Haiku's JudgeBench parse rate and is a deviation.

Hypotheses, as registered. H1: at least a quarter of audited pipelines contain an override. H2: on every benchmark and judge pair, applying an override raises κ, and κ_U is unchanged. H3: the marginal channel is larger whenever a judge's pre-override positive-class rate exceeds 0.85. H4: the closed form is within 0.03 of the observed post-override κ in at least 90% of cells. H5: κ_U, the marginals and the constant-rater control together identify every inflated cell and none alone does.

5 Results

944 cells have a non-empty override and both judges above the parse threshold; 216 control cells all have a gain of exactly 0.0000. Nine judges, six benchmarks, 36 judge pairs per benchmark.

H1, supported. 28 of 35 coded pipelines (80%) against the registered 25%.

H2, not supported as stated. κ rises in 697 of 944 cells (74%); the 95% interval excludes zero upward in 436 (46%) and downward in 70 (7%). After Holm correction over all 944 cells, 300 positive and 12 negative cells survive. Table 3 and Figure 1 split this by rule.

Length and truncation rules are the inflator. They write a median 0.275 of rows and raise κ by a median +0.071, with the interval above zero in 301 of 388 cells (78%). Refusal rules write almost no rows on these benchmarks (median share 0.001) and move nothing: the judges already agree on refusals. Deixis rules sit in between. The tie-break raises κ in most cells because the human labels are almost never ties, so writing A on a judge's ties agrees with the partner more often than TIE did.

The parse-failure default is the instructive case. Applied per column, it raises κ significantly in 37 cells and lowers it in 36: the constant the parser writes sets the sign. Claude Haiku 4.5 on JudgeBench, excluded from the main check, is the extreme. It returns no verdict on 190

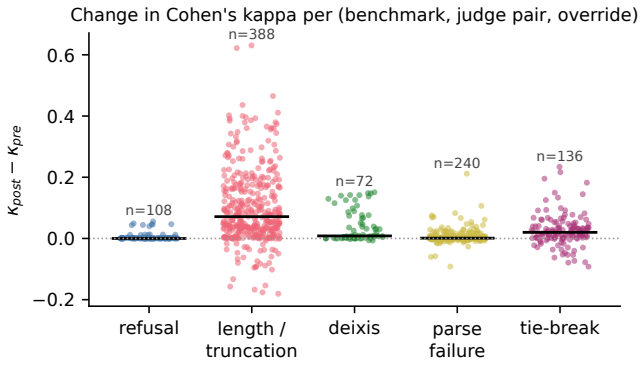

Figure 1: $\kappa_{\text{post}} - \kappa_{\text{pre}}$ for every cell, grouped by override category; bars are medians. Length and truncation rules are the inflator; refusal rules do nothing; parse defaults scatter on both sides of zero.

benchmark	n	pairs	share written	med. $\Delta\kappa$	CI>0 / CI<0
MT-Bench (human)	1000	72	0.167	+0.068	69 / 0
LLMBar	419	72	0.298	+0.168	72 / 0
RewardBench	1000	72	0.220	+0.131	65 / 0
JudgeBench	270	28	0.004	+0.003	0 / 0
HaluEval	1000	72	0.506	+0.008	29 / 19
LoCoMo	600	72	0.251	+0.050	66 / 0

Table 4: Length / truncation rules by benchmark: share of rows written and κ change over the judge pairs above the parse threshold.

of 270 items, so a default writes 192 rows of its column. Writing A raises κ against Gemini 2.5 Flash by +0.080 [+0.015, +0.142]; writing TIE lowers it against Qwen3 8B by -0.080 [-0.133, -0.025]. Same replies, same partner judges, opposite conclusions about agreement, decided by a constant in a helper function.

H3, not supported. The forced channel dominates. Among row-wise cells where a judge's pre-override marginal exceeds 0.85, the marginal channel is larger in 18 of 56; elsewhere in 30 of 512. On these benchmarks the prevalence effect is real but second-order; what moves κ is the rows that were overwritten.

H4, not supported at the 90% bar. The closed form is within 0.03 in 374 of 568 row-wise cells (66%); the median absolute error is 0.015 and the 90th percentile 0.141 (Figure 2). The error has a direction: the closed form under-predicts the observed κ in 211 cells (37%). The representative-rows assumption fails because rules do not land on random rows. A short or unterminated answer is exactly the answer two judges were most likely to disagree about, so writing it removes more disagreement than a random row would. A reader cannot correct for an override from the headline table and f alone; the correction is a lower bound.

H5, partly supported. Of 436 cells with the interval above zero, a κ_U gap over 0.02 flags 378 (87%), the marginal-or-written-share rule flags 394 (90%), and the constant-rater control flags 63 (14%). The first two together flag 406 (93%). κ_U and the written share are sufficient; the constant-rater control adds nothing here, and we no longer recommend it.

Agreement with the human label moves the other way. This analysis was added after the results were read and is reported as exploratory. Every benchmark carries a human or gold label, so the same overrides can be applied to each judge alone and scored against the label (414 cells; Table 5). Length rules, which raise inter-judge κ most, lower accuracy against the label in 49 of 108 cells and raise it in 12. Refusal and deixis rules change it in none.

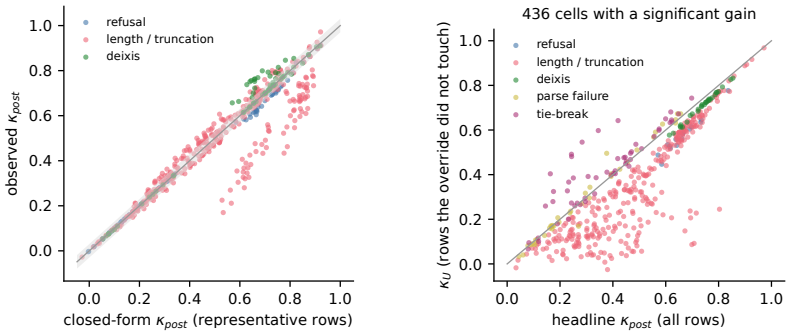

Figure 2: Left: observed post-override κ against the closed-form prediction (row-wise cells); points below the diagonal are under-predictions. Right: reported κ against κ_U, the agreement on rows the override did not touch; the vertical distance is the inflation.

code	override	cells	acc. up (CI> 0)	acc. down (CI< 0)	med. Δacc.
R	refusal rule	90	0	0	+0.0000
L	length / truncation	108	12	49	-0.0049
D	unresolved deixis	18	0	0	+0.0000
P	parse-failure default	108	19	0	+0.0000
T	tie-break	36	23	1	+0.0191

Table 5: Accuracy against the benchmark's human or gold label, per judge, before and after each override (parse rate ≥ 0.8).

Parse defaults raise it in 19 of 108 and lower it in none; the tie-break raises it in 23 of 36, because the labels are almost never ties. Inter-judge agreement and agreement with the truth are different quantities, and one rule can move them in opposite directions.

Parse behaviour is a finding. Under the same one-token prompt, Gemini 2.5 Flash and Qwen3 8B return a bare verdict on 100% of JudgeBench items, Gemma 4 on 96%, Llama 3.1 on 95%, Flash-Lite on 93%, Mistral on 92%, Qwen3 32B on 89%, Sonnet 5 on 87%, and Haiku 4.5 on 30%; on RewardBench Haiku and Llama 3.1 parse at 84% (Appendix D). Parse failure is judge- and benchmark-dependent, so a parser default is not a small uniform correction; it is a judge-specific rewrite of up to two thirds of a column.

6 What to report, and when a rule is the right call

Next to any inter-judge agreement, report three things: the share of rows whose verdict a rule wrote, per rule; κ on the untouched rows; and each judge's marginal before and after. The first two caught every inflated cell in our data and cost nothing to compute. The closed form is a lower bound and should be labelled as one.

None of this says rules are wrong. A deterministic fact about the item (empty output, exceeded token budget, a non-answer by schema) is legitimately scored by rule; the judge is not needed. Report it as a rule, not as agreement. Scoring a refusal as wrong is a modelling choice about the task, not a measurement; state it and report both. A parser default is never a measurement: keep $\perp$ as its own class, as Inspect and HELM do. Ties should be reported as ties; if a binary outcome is needed, break them randomly with a stated seed, or report both sides.

7 Related work

Work on LLM judges has studied the judge: position bias (Wang et al., 2023; Shi et al., 2025), verbosity (Dubois et al., 2024), self-preference (Panickssery et al., 2024), and catalogues of biases (Ye et al., 2024; Chen et al., 2024; Gu et al., 2024). Thakur et al. (2025) measure alignment with human labels across judge sizes and note vulnerability to the output format; they do not vary the pipeline around a fixed judge. Meta-evaluation benchmarks (Zheng et al., 2023; Zeng et al., 2024; Lambert et al., 2024; Tan et al., 2025) supply the human labels we use, and their own harnesses appear in our audit. Verga et al. (2024) propose panels of judges, for which inter-judge agreement is a design quantity; Dorner et al. (2025) bound what a judge can add over ground-truth labels. This paper studies the code between the judge and the number. The largest reliability audit to date, Norman et al. (2026), scores 21 judge models on three benchmarks and reports that raw agreement overstates chance-corrected agreement by 33 to 41 points on MT-Bench; that gap is a property of the coefficient and appears with no code in the loop, whereas the inflation we measure is written into both judge columns by the pipeline and survives chance correction, which is why κ_U rather than κ is the diagnostic. The closest statistical work is on κ's dependence on prevalence and bias for a single rater pair (Feinstein & Cicchetti, 1990; Byrt et al., 1993); our marginal channel is that effect, and our result is that on real pipelines it is dominated by the forced channel. Kapoor et al. (2024) document harness pitfalls for agent benchmarks; the parser default is the judge-pipeline analogue.

8 Limitations and non-claims

The audit was coded by one reader; the second-pass sample in the protocol was not performed. Cells are not independent (36 pairs share nine judges), which the Holm correction does not model; the per-code counts should be read with that in mind. The refusal and deixis rules had few rows to write on these benchmarks, so their null result is a statement about these benchmarks. The length rules are two specific rules; other thresholds would write other rows. Two LoCoMo systems are our own. The cross-hardware replication check for the local judges was not run. We make no claim that any audited pipeline's published conclusions are wrong, and none about which filter policy is correct.

9 Conclusion

An inter-judge κ is a property of the judges only on the rows the judges decided. Public pipelines routinely decide some rows by rule and report the combined number. On six benchmarks, length rules raise κ by a median $+0.071$, parser defaults move it either way by up to ± 0.08, and the gain is not recoverable from the headline table because rules land on the contested rows. Two numbers, the share written and κ_U, make the effect visible. We ask that they be reported.

Ethics statement

The benchmarks are public and carry no personal data beyond what their authors released. Two of the four memory systems whose LoCoMo answers are judged were built by the authors; their rows are labelled and their failures reported in full. The case report of Anonymous (2026), which this paper generalises, describes one of those pipelines; the LoCoMo verdicts released with it are re-used here as one of the six benchmarks, and the nine judges, the audit and the factorial are new. The audit reads public code and cites file and line; it makes no claim about the correctness of any repository's published results.

Reproducibility statement

The pre-registration with dated deviations (Appendix A), the audit coding sheet with pinned commits (Appendix C), the prompts and parser (Appendix B), all 38,601 verdicts, the judge runner, and the analysis code that produced every table and figure are in the supplementary

archive; run_all.py regenerates cells.csv and tables.py regenerates every table from it. Judge spend was $10.69 at list price plus about $4 of GPU time.

AI use statement

We used a large language model (Claude) as a coding and writing assistant throughout: to write and refactor the judge runner, the analysis code and the figure and table scripts under our direction; to search for and summarise related work, which we then read; and to draft and edit parts of this text. We did not use generative AI to generate the research idea, to produce any reported number, or to code the audit sheet, whose entries were read from the repositories by a human. All AI-written code was reviewed and tested by us; every number in the paper is generated by the released code from the released verdicts. We take responsibility for the final content of this work, including text, claims and artifacts produced with the aid of generative AI.

References

Anonymous. Agreement by construction: A deterministic filter inflates inter-judge reliability in LLM-judge evaluation. Submission to Transactions on Machine Learning Research, submission 11904, https://openreview.net/forum?id=mbfz5pMxCY. Desk rejected; verdicts and analysis code released with it., 2026.

Ted Byrt, Janet Bishop, and John B. Carlin. Bias, prevalence and kappa. Journal of Clinical Epidemiology, 46(5):423–429, 1993.

Guiming Hardy Chen, Shunian Chen, Ziche Liu, Feng Jiang, and Benyou Wang. Humans or LLMs as the judge? A study on judgement biases. In Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, 2024. arXiv:2402.10669.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory. arXiv preprint arXiv:2504.19413, 2025.

Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Hao Zhang, Banghua Zhu, Michael Jordan, Joseph E. Gonzalez, and Ion Stoica. Chatbot arena: An open platform for evaluating LLMs by human preference. arXiv preprint arXiv:2403.04132, 2024.

Jacob Cohen. A coefficient of agreement for nominal scales. Educational and Psychological Measurement, 20(1):37–46, 1960.

Florian E. Dorner, Vivian Y. Nastl, and Moritz Hardt. Limits to scalable evaluation at the frontier: LLM as judge won't beat twice the data. In International Conference on Learning Representations, 2025. arXiv:2410.13341.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B. Hashimoto. Length-controlled AlpacaEval: A simple way to debias automatic evaluators. In First Conference on Language Modeling, 2024. arXiv:2404.04475.

Alvan R. Feinstein and Domenic V. Cicchetti. High agreement but low kappa: I. The problems of two paradoxes. Journal of Clinical Epidemiology, 43(6):543–549, 1990.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Saizhuo Wang, Kun Zhang, Yuanzhuo Wang, Wen Gao, Lionel Ni, and Jian Guo. A survey on LLM-as-a-judge. arXiv preprint arXiv:2411.15594, 2024.

Sayash Kapoor, Benedikt Stroebl, Zachary S. Siegel, Nitya Nadgir, and Arvind Narayanan. AI agents that matter. arXiv preprint arXiv:2407.01502, 2024.

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. arXiv preprint arXiv:2403.13787, 2024.

Junyi Li, Xiaoxue Cheng, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. HaluEval: A large-scale hallucination evaluation benchmark for large language models. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, 2023. arXiv:2305.11747.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, et al. Holistic evaluation of language models. Transactions on Machine Learning Research, 2023. arXiv:2211.09110.

Bill Yuchen Lin, Yuntian Deng, Khyathi Chandu, Faeze Brahman, Abhilasha Ravichander, Valentina Pyatkin, Nouha Dziri, Ronan Le Bras, and Yejin Choi. WildBench: Benchmarking LLMs with challenging tasks from real users in the wild. arXiv preprint arXiv:2406.04770, 2024.

Xiao Liu, Xuanyu Lei, Shengyuan Wang, Yue Huang, Zhuoer Feng, Bosi Wen, Jiale Cheng, Pei Ke, Yifan Xu, Weng Lam Tam, Xiaohan Zhang, Lichao Sun, Xiaotao Gu, Hongning Wang, Jing Zhang, Minlie Huang, Yuxiao Dong, and Jie Tang. AlignBench: Benchmarking Chinese alignment of large language models. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics, 2024. arXiv:2311.18743.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics, 2024. arXiv:2402.17753.

Justin D. Norman, Michael U. Rivera, and D. Alex Hughes. Reliability without validity: A systematic, large-scale evaluation of LLM-as-a-judge models across agreement, consistency, and bias. arXiv preprint arXiv:2606.19544, 2026.

Arjun Panickssery, Samuel R. Bowman, and Shi Feng. LLM evaluators recognize and favor their own generations. arXiv preprint arXiv:2404.13076, 2024.

Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. Zep: A temporal knowledge graph architecture for agent memory. arXiv preprint arXiv:2501.13956, 2025.

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 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, 2025. arXiv:2406.07791.

Sijun Tan, Siyuan Zhuang, Kyle Montgomery, William Y. Tang, Alejandro Cuadron, Chenguang Wang, Raluca Ada Popa, and Ion Stoica. JudgeBench: A benchmark for evaluating LLM-based judges. In International Conference on Learning Representations, 2025. arXiv:2410.12784.

Aman Singh Thakur, Kartik Choudhary, Venkat Srinik Ramayapally, Sankaran Vaidyanathan, and Dieuwke Hupkes. Judging the judges: Evaluating alignment and vulnerabilities in LLMs-as-judges. In Proceedings of the Fourth Workshop on Generation, Evaluation and Metrics (GEM^2), 2025. arXiv:2406.12624.

UK AI Security Institute. Inspect AI: Framework for large language model evaluations. https://github.com/UKGovernmentBEIS/inspect_ai, 2024.

Pat Verga, Sebastian Hofstatter, Sophia Althammer, Yixuan Su, Aleksandra Piktus, Arkady Arkhangorodsky, Minjie Xu, Naomi White, and Patrick Lewis. Replacing judges with juries: Evaluating LLM generations with a panel of diverse models. arXiv preprint arXiv:2404.18796, 2024.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. Large language models are not fair evaluators. arXiv preprint arXiv:2305.17926, 2023.

Yidong Wang, Zhuohao Yu, Zhengran Zeng, Linyi Yang, Cunxiang Wang, Hao Chen, Chaoya Jiang, Rui Xie, Jindong Wang, Xing Xie, Wei Ye, Shikun Zhang, and Yue Zhang. PandaLM: An automatic evaluation benchmark for LLM instruction tuning optimization. In International Conference on Learning Representations, 2024. arXiv:2306.05087.

Jason Wei, Nguyen Karina, Hyung Won Chung, Yunxin Joy Jiao, Spencer Papay, Amelia Glaese, John Schulman, and William Fedus. Measuring short-form factuality in large language models. arXiv preprint arXiv:2411.04368, 2024.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking chat assistants on long-term interactive memory. In International Conference on Learning Representations, 2025. arXiv:2410.10813.

Jiayi Ye, Yanbo Wang, Yue Huang, Dongping Chen, Qihui Zhang, Nuno Moniz, Tian Gao, Werner Geyer, Chao Huang, Pin-Yu Chen, Nitesh V. Chawla, and Xiangliang Zhang. Justice or prejudice? Quantifying biases in LLM-as-a-judge. arXiv preprint arXiv:2410.02736, 2024.

Zhiyuan Zeng, Jiatong Yu, Tianyu Gao, Yu Meng, Tanya Goyal, and Danqi Chen. Evaluating large language models at evaluating instruction following. In International Conference on Learning Representations, 2024. arXiv:2310.07641.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In Advances in Neural Information Processing Systems 36, Datasets and Benchmarks Track, 2023. arXiv:2306.05685.

Lianghui Zhu, Xinggang Wang, and Xinlong Wang. JudgeLM: Fine-tuned large language models are scalable judges. In International Conference on Learning Representations, 2025. arXiv:2310.17631.

A Pre-registration and deviations

The pre-registration was frozen on 8 September 2026 before the first full run; the file with every dated deviation is in the archive. The deviations that touched a reported number:

1. DeepSeek-R1 8B replaced by Mistral 7B before the first full run (reasoning block cannot be disabled; no verdict within any bounded budget).
2. The parser's allowed-token table had no entry for the correctness task, so a reply of WRONG was a parse failure; found in a three-item smoke test before any LoCoMo verdict existed and fixed.
3. The API tier was Gemini 2.5 Flash-Lite, Gemini 2.5 Flash, Claude Haiku 4.5 and Claude Sonnet 5 rather than the registered Gemini and Llama 3.3 70B (not enabled in the project).
4. The local tier moved from an M1 Max laptop to an L4 VM mid-run; items already judged were kept.
5. The API output cap was raised from 64 to 160 tokens for parity and only unparsed rows re-run.
6. Reply-based overrides (P, T) are applied per column, item-text overrides (R, L, D) row-wise; the first implementation applied reply predicates from judge 1 to both columns and its numbers are discarded.
7. Haiku 4.5's JudgeBench parse rate (30%) is kept as measured; an 80% parse threshold was added for the hypothesis checks.

8. RewardBench's id field is not unique (id 3692 occurs twice); ids were changed to the row index and existing rows re-keyed by prompt hash.
9. The cross-hardware check was not run; the same-hardware repeat (595/600 identical) is the stated floor.
10. The human-label analysis (Table 5) was added after results were read and is labelled exploratory.
11. 2,000 bootstrap resamples instead of 10,000; Holm–Bonferroni via a normal approximation.

B Prompts and parser

Pairwise (MT-Bench human, LLMBar, RewardBench, JudgeBench).

```
You are an impartial judge. Compare the two responses to the user prompt
below and decide which is better overall, considering correctness,
helpfulness, and adherence to the instruction. Do not favor length or
position.

[User prompt]
{prompt}

[Response A]
{a}

[Response B]
{b}

Answer with exactly one of: A, B, or TIE. Output only that token.
```

Hallucination (HaluEval).

```
You are a strict fact checker. Given the reference knowledge and the
question, decide whether the candidate answer contains hallucinated content
that is not supported by the knowledge.

[Knowledge]
{knowledge}

[Question]
{question}

[Candidate answer]
{answer}

Answer with exactly one of: YES (hallucinated) or NO (faithful). Output
only that token.
```

Correctness (LoCoMo).

```
You are grading an answer to a question about a long conversation. Decide
whether the predicted answer is correct, judging against the reference
answer. Minor wording differences do not matter; missing or wrong facts do.

[Question]
{question}

[Reference answer]
{gold}

[Predicted answer]
{prediction}
```

Answer with exactly one of: CORRECT or WRONG. Output only that token.

Lenient parser. A reply is first stripped of any <think> block and whitespace-normalised and upper-cased. Strict: the whole reply, stripped of quotes, asterisks and a final period, is one allowed token. Lenient: the strict token, or else the first match of the following patterns, in order; INCORRECT maps to WRONG, HALLUCINAT*/NOT SUPPORTED to YES, FAITHFUL/SUPPORTED to NO, and BOTH ARE EQUAL to TIE. No match is a parse failure.

```
pats = [r"\b(?:RESPONSE|ANSWER|OPTION)\s*\[?([AB])\]?\s+(?:IS|WAS)\s+(?:THE
    \s+)?(?:BETTER|PREFERRED|
STRONGER|SUPERIOR|MORE)",
r"\b(?:VERDICT|WINNER|FINAL
    ANSWER|ANSWER|CHOICE|DECISION)\s*[:=]?\s*\[?\[?([AB]|TIE)\]?\]?\b",
r"\[\[([AB]|TIE)\]\]", r"\b(TIE)\b|\bBOTH (?:RESPONSES )?ARE
    (?:EQUALLY|EQUAL|SIMILAR)",
r"(?:^|[^A-Z])([AB])[.!]?\s*$"]
elif task == "qa_correctness":
pats = [r"\b(?:VERDICT|ANSWER|DECISION|GRADE|RESULT)\s*[:=]?\s*(CORRECT|INC
    ORRECT|WRONG)\b",
r"\b(?:IS|WAS)\s+(CORRECT|INCORRECT|WRONG)\b",
    r"(?:^|[^A-Z])(CORRECT|INCORRECT|WRONG)[.!]?\s*$"]
else:
pats = [r"\b(?:VERDICT|ANSWER|DECISION)\s*[:=]?\s*(YES|NO)\b",
    r"\b(HALLUCINAT\w*|NOT
SUPPORTED|UNSUPPORTED)\b",
r"\b(FAITHFUL|SUPPORTED|NO HALLUCINATION)\b",
    r"(?:^|[^A-Z])(YES|NO)[.!]?\s*$"]
```

C Audit coding sheet

One row per repository at its pinned commit; codes as in Table 1 (N: none found; na/tbd: not decided). "paper" is what the pipeline's paper says about parse failures, ties or exclusions: partial (a related rule is stated), n (nothing), na (no paper or not a judge pipeline). Full snippets and notes are in the released sheet.

repository (commit)	benchmark	codes	paper	where
lm-sys/FastChat (587d5cf)	MT-Bench	P;X;T	partial	fastchat/llm_judge/common.py: 263,290,304; :158,183; show_result.py:49,64
allenai/reward-bench (05a9005)	RewardBench	P;T	n	rewardbench/generative.py: 462-498,506; scripts/run_generative.py: 252-257,366-372
RUCAIBox/ HaluEval (b7253db)	HaluEval	F;P	n	evaluation/evaluate.py:188-206
ScalerLab/ JudgeBench (e2c52c2)	JudgeBench	T;P;X	partial	utils/judges.py: 80-86,305-318,473,525,564
xiaowu0162/ LongMemEval (9e0b455)	LongMemEval	F;P	n	src/evaluation/evaluate_qa.py: 113
getzep/zep-papers (4b7f26c)	LoCoMo, LongMemEval	F;P	n	kg_architecture_agent_memory/ locomo_eval/zep_locomo_eval. py:51-60
openai/simple-evals (652c89d)	SimpleQA	F;X	n	simpleqa_eval.py:125-126
UKGovernmentBEIS/ inspect_ai (58f2e9e)	Inspect model_graded_qa	N	na	src/inspect_ai/scorer/_model.py: 312-335,433
lmarena/arena-hard-auto (196f6b8)	Arena-Hard	F;X	n	gen_judgment.py:21-31,67-68; show_result.py:31-32

repository (commit)	benchmark	codes	paper	where
tatsu-lab/alpaca_eval (cd543a1)	AlpacaEval 2.0	P;X	n	src/alpaca_eval/completion_parsers.py:127,157,284; annotators/base.py:88,357,413; metrics/helpers.py:137-141,164
prometheus-eval/prometheus-eval (dcfb442)	Prometheus 2	P	n	libs/prometheus-eval/prometheus_eval/parser.py:2-71
nlpyang/geval (6f84404)	SummEval (G-Eval)	P	n	meta_eval_summeval.py:30-37
EleutherAI/lm-evaluation-harness (b954108)	lm-eval regex filters	P	na	lm_eval/filters/extraction.py:20-40
open-compass/opencompass (8522274)	AlignBench and subjective evals	P	na	opencompass/datasets/subjective/alignbench.py:151-220
stanford-crfm/helm (63754d0)	HELM model-as-judge annotators	N	na	src/helm/benchmark/annotation/model_as_judge.py:54-57,92-100
mem0ai/mem0 (4b61c5d)	LoCoMo (mem0 evaluation)	F;P	n	evaluation/benchmarks/locomo/run.py:481-487,552
allenai/WildBench (d6b8dca)	WildBench	P;L	partial	src/eval.py:101-116; src/eval.py:366-373,389; leaderboard/wb_elo.py:153-156
WeOpenML/PandaLM (3871806)	PandaLM test set	P;T	partial	pandalm/utils/pandalm_inference.py:85-92
baaivision/JudgeLM (00c4516)	JudgeLM-val (PandaLM test)	P;T	partial	judgelm/llm_judge/common.py:10-25
GAIR-NLP/auto-j (2ae17a3)	Auto-J pairwise test	P	partial	codes/usage/bilingual_example.py:8-34
chanchimin/ChatEval (56b320c)	FairEval / Vicuna-80	P;T	n	calc_adversarial_results_calc_score_calibration.py:40-82
openai/evals (8eac7a7)	OpenAI Evals modelgraded classify	P	na	evals/elsuite/modelgraded/classify_utils.py:10,86,99-128
kaistAI/FLASK (3b4e22b)	FLASK	X	n	gpt_review/aggregate_skill.py:30-52
snap-research/locomo (3eb6f2c)	LoCoMo (original)	N	n	task_eval/evaluation.py:95-135
promptfoo/promptfoo (e5767f2)	promptfoo llm-rubric / model-graded assertions	P	na	src/matchers/shared.ts:53-60; src/matchers/rubric.ts:781-800
confident-ai/deepeval (f94d940)	DeepEval G-Eval and metrics	X	na	deepeval/metrics/utils.py:464-480; g_eval/g_eval.py:325-330
truera/trulens (531a8bb)	TruLens feedback functions	N	na	src/feedback/trulens/feedback/generated.py:104-160; llm_provider.py:53-57
THUDM/AlignBench (6505913)	AlignBench	P	partial	judge.py:95-128; show_result.py:52-54
huggingface/lighteval (932e1f2)	LightEval llm-as-judge metrics	P	na	src/lighteval/metrics/utils/llm_as_judge.py:438-460
prometheus-eval/BiGGen-Bench (dcfb442)	BiGGen Bench	P	n	(uses prometheus_eval/parser.py:2-58)
embeddings-benchmark/mteb (692bc29)	MTEB (control)	N	na	(no judge code; 'LLM-as-a-judge' appears only in dataset descriptions)
vllm-project/vllm (c6ba5e8)	vLLM (control)	N	na	(no judge code)

repository (commit)	benchmark	codes	paper	where
THUDM/ AgentBench (d1e4a10)	AgentBench (control)	N	na	(no judge code)
sierra-research/tau-bench (59a200c)	tau-bench (control)	N	na	(no judge code)
LiveBench/ LiveBench (bc5164c)	LiveBench	N	na	livebench/process_results/*
letta-ai/letta (4511fa0)	Letta (memory system)	N	na	(no judge code in main repo)
langchain-ai/ langmem (f8c7ebd)	LangMem	N	na	(no judge code)
explodinggradients/ ragas (298b682)	RAGAS faithfulness and judge metrics	tbd	na	src/ragas/metrics/_faithfulness. py:60-125; _nv_metrics.py
open-compass/ CompassJudger (9cdf659)	CompassJudger (judge model)	tbd	na	(model release; scoring via OpenCompass)
princeton-nlp/ LLMBar (900616b)	LLMBar	T;P	n	LLMEvaluator/evaluate.py: 337-342,389,447-457
lmarena/PPE (173b588)	PPE (Preference Proxy Evaluations)	F;X	n	llm_judge/judges/arena_hard.py: 56-66
mtbench101/mt-bench-101 (bc18b3e)	MT-Bench-101	na	na	(data only; evaluation code lives in OpenCompass)

D Parse rates

judge	MT-Bench (human)	LLMBar	RewardBench	JudgeBench	HaluEval	LoCoMo
Qwen3 8B	100 / 100	100 / 100	99 / 98	89 / 79	100 / 100	100 / 100
Gemma 4	100 / 98	100 / 97	100 / 99	96 / 37	100 / 97	100 / 100
Llama 3.1 8B	99 / 90	89 / 64	84 / 47	95 / 6	100 / 100	100 / 100
Mistral 7B	99 / 87	99 / 93	100 / 93	92 / 62	100 / 99	100 / 100
Qwen3 32B	100 / 100	100 / 100	100 / 100	100 / 100	100 / 100	100 / 100
Gemini 2.5 Flash-Lite	100 / 99	100 / 98	100 / 97	93 / 55	100 / 77	100 / 100
Gemini 2.5 Flash	100 / 100	100 / 100	100 / 100	100 / 100	100 / 100	100 / 100
Claude Haiku 4.5	98 / 97	99 / 99	84 / 84	30 / 27	100 / 100	100 / 100
Claude Sonnet 5	96 / 96	100 / 100	99 / 99	87 / 87	100 / 100	100 / 100

Lenient / strict parse rate in percent for each judge and benchmark. Strict requires the whole reply to be one token.

E Minimum detectable difference

benchmark	n	cells	median half-width	90th percentile
MT-Bench (human)	1000	160	0.014	0.027
LLMBar	419	160	0.035	0.054
RewardBench	1000	246	0.010	0.039
JudgeBench	270	110	0.023	0.050
HaluEval	1000	124	0.040	0.054
LoCoMo	600	144	0.023	0.038

Half-width of the 95% bootstrap interval on the κ gain, by benchmark; a difference smaller than this is not distinguishable from zero in one cell.

F Extensions run after the pre-registered study

Three additions were run after the pre-registered results were read, funded by an expiring cloud credit; none changes a number in §5, and they are reported here apart from H1–H5.

F.1 Stronger judges

Three further judges from a newer model generation (Gemini 3.8 Flash, Gemini 3.5 Flash-Lite, Gemini 3.1 Pro, thinking off, temperature 0, same prompt and parser) were run over all six benchmarks and paired with every other judge: 810 cells. Table 7 repeats Table 3 for cells involving at least one of them.

code	override	cells	CI>0	CI<0	med. $\Delta\kappa$	share written
R	refusal rule	90	6	0	+0.000	0.001
L	length / truncation	317	215	33	+0.046	0.275
D	unresolved deixis	60	21	7	+0.002	0.084
P	parse-failure default	236	48	31	+0.003	0.021
T	tie-break	107	46	5	+0.017	0.104

Table 7: Cells involving a Gemini 3.x judge, by override category.

F.2 Two further human-labelled pairwise benchmarks

A stratified 1,000-item subset of the Chatbot Arena human-preference set (Chiang et al., 2024) (400 A, 400 B, 200 tie; seed 42) and the PandaLM test set (Wang et al., 2024) (999 items; label = majority of three annotators, 105 ties) were judged by 9 judges: 440 cells. The length rule raises κ with the interval above zero in 56 of 56 Arena cells (median +0.086) and 72 of 72 PandaLM cells (median +0.104).

benchmark	code	override	cells	CI>0	CI<0	med. $\Delta\kappa$	share written
Arena-HP	R	refusal rule	56	35	0	+0.005	0.011
Arena-HP	L	length / truncation	56	56	0	+0.086	0.178
Arena-HP	P	parse-failure default	54	12	9	+0.001	0.012
Arena-HP	T	tie-break	28	16	1	+0.027	0.289
PandaLM	R	refusal rule	72	0	0	+0.002	0.003
PandaLM	L	length / truncation	72	72	0	+0.104	0.202
PandaLM	P	parse-failure default	66	27	3	+0.002	0.026
PandaLM	T	tie-break	36	27	0	+0.038	0.318

F.3 Position swaps

Each pairwise item was re-judged with A and B exchanged. A judge is swap-consistent on an item when the second verdict mirrors the first (A↔B, TIE↔TIE). Over 30 (benchmark, judge) columns the consistency rate has median 0.784 (range 0.276–0.972). The inconsistencies are not noise. Of the 5,756 of them, 3,581 (62%) are cases where the judge named the same letter both times, which after the exchange is a vote for the position rather than for the response. The three Gemini 3.x judges sit at 0.95 and above; the four small local judges sit between 0.28 and 0.57, and for two of them more than nine in ten of their inconsistencies are position votes. The tie-break override (code T) was then re-analysed on rows where both judges are swap-consistent: of 74 judge pairs, the interval is above zero in 47 on all rows and 55 on consistent rows (median gain +0.034 and +0.061). Restricting to rows the judge decides the same way in both orders does not remove the effect of writing its ties to one side.

benchmark	judge	n	both parsed	consistent	rate	position votes
Arena-HP	Llama 3.1 8B	1000	333	92	0.276	0.92
Arena-HP	Mistral 7B	1000	982	474	0.483	0.51
Arena-HP	Qwen3 8B	1000	998	496	0.497	0.99
PandaLM	Llama 3.1 8B	999	674	374	0.555	0.57
PandaLM	ollama_gemma4	999	999	568	0.569	0.64
PandaLM	Qwen3 8B	999	999	574	0.575	0.97
Arena-HP	ollama_gemma4	1000	999	675	0.676	0.85
Arena-HP	Gemini 3.5 Flash-Lite	1000	888	626	0.705	0.66
JudgeBench	Gemini 3.5 Flash-Lite	270	78	55	0.705	0.78
PandaLM	Mistral 7B	999	989	706	0.714	0.23
Arena-HP	Claude Sonnet 5	1000	985	715	0.726	0.26
PandaLM	Gemini 3.5 Flash-Lite	999	859	645	0.751	0.54
JudgeBench	Gemini 3.1 Pro	270	269	203	0.755	0.45
Arena-HP	Claude Haiku 4.5	1000	990	767	0.775	0.94
LLMBar	Gemini 3.5 Flash-Lite	419	372	291	0.782	0.73
PandaLM	Claude Sonnet 5	999	995	781	0.785	0.36
MT-Bench (human)	Gemini 3.5 Flash-Lite	1000	820	664	0.810	0.77
Arena-HP	Gemini 3.1 Pro	1000	999	814	0.815	0.18
PandaLM	Claude Haiku 4.5	999	959	787	0.821	0.89
MT-Bench (human)	Gemini 3.1 Pro	1000	1000	850	0.850	0.19
PandaLM	Gemini 3.1 Pro	999	998	849	0.851	0.13
Arena-HP	Gemini 3.8 Flash	1000	982	853	0.869	0.63
RewardBench	Gemini 3.5 Flash-Lite	1000	723	633	0.875	0.68
PandaLM	Gemini 3.8 Flash	999	998	896	0.898	0.27
LLMBar	Gemini 3.1 Pro	419	418	376	0.899	0.26
MT-Bench (human)	Gemini 3.8 Flash	1000	971	878	0.904	0.53
JudgeBench	Gemini 3.8 Flash	270	258	234	0.907	0.67
LLMBar	Gemini 3.8 Flash	419	418	397	0.950	0.86
RewardBench	Gemini 3.1 Pro	1000	1000	952	0.952	0.27
RewardBench	Gemini 3.8 Flash	1000	988	960	0.972	0.79