Small Language Models as Judges: A Survey

Anish Laddha[1] **Nitesh Pradhan**[1] **Gaurav Srivastava**[2]

[1]Department of Computer Science and Engineering, LNMIIT, Jaipur, India
[2]Department of Computer Science, Virginia Tech, Blacksburg, VA, USA

anshladdha15@gmail.com nitesh.pradhan@lnmiit.ac.in gks@vt.edu

GitHub: anishh15/Awesome-SLM-as-a-Judge

Abstract

Automated evaluation of language model outputs has shifted from costly human annotation toward model-based judging, yet the field has overwhelmingly relied on large proprietary models whose cost, opacity, and data-privacy risks limit scalable deployment. This survey systematically examines *Small Language Models* (SLMs, $\leq$14B parameters) as judges, synthesizing **over 60 works** spanning the full pipeline from supervised fine-tuning and preference optimization through reinforcement learning with verifiable rewards to representation-based probing. We organize the field into five dimensions and distill five insights: (**I1**) evaluation-specific training outperforms raw scale; (**I2**) reasoning tokens help only when they surface genuinely new discriminative signals; (**I3**) cross-family ensemble diversity decorrelates errors; (**I4**) debate value is bounded by information novelty, not round count; and (**I5**) SLM judge reliability tracks the verifiability gradient of the domain. Through position-consistent accuracy protocols we further find that much apparent judge performance is heuristic-driven, and that all judges, including frontier models, degrade sharply on hard evaluation pairs. Practically, fine-tuned 3–8B judges match proprietary-model accuracy on standard benchmarks while heterogeneous panels reach comparable quality at over 7$\times$ lower cost, enabling private, on-device evaluation for continuous Reinforcement Learning from Human Feedback (RLHF) loops; however, this parity narrows in subjective domains lacking external verification. We close with a research agenda on calibration, adaptive inference, and generalization. A repository of the papers, models, and tools discussed in this survey is available at https://github.com/anishh15/Awesome-SLM-as-a-Judge.

1 Introduction

Evaluating language model outputs reliably and at scale remains a central open problem in NLP. Human annotation, while the gold standard, fails to scale: assembling expert annotators for the thousands of iterative experiments required by modern RLHF pipelines (Ouyang et al., 2022) and constitutional alignment approaches (Bai et al., 2022) is prohibitively expensive. This bottleneck motivated the *LLM-as-a-Judge* paradigm (Zheng et al., 2023), in which a powerful model, typically GPT-4 (Achiam et al., 2023), serves as an automated proxy for human preference. MT-Bench demonstrated over 80% agreement between GPT-4 and expert humans; JudgeLM (Zhu et al., 2025) and Prometheus (Kim et al., 2024a) then extended this capability to open-source fine-tuned models.

The proprietary judge paradigm, however, carries structural costs that constrain its use. At evaluation scale $N \gg 10^4$, total API expenditure scales linearly in N, making continuous RLHF loops financially untenable for most research groups. Cloud-based judges also create data-privacy exposure, version-drift opacity, and latency that prevents real-time feedback loops. These constraints, combined with the rapid capability gains of recent small models (Abdin et al., 2024; Yang et al., 2025; Grattafiori et al., 2024; Guo et al., 2025), which now match larger models on reasoning (Srivastava et al., 2025b), contamination-resistant evaluation (Srivastava et al., 2026c), and agentic tasks (Srivastava et al., 2026b), raise the question this survey sets out to answer: *can small language models reliably serve as judges, and if so, under what conditions?* The answer is not a simple yes or no. As we show, it depends on how the judge is trained, how much it is allowed to reason, whether it acts alone or in a panel, and above all on whether the domain admits external verification.

The moment is right for this survey. Two concurrent pressures converge: (1) SLMs in 2025–2026 achieve reasoning performance previously requiring 70B+ models (Abouelenin et al., 2025;

Guo et al., 2025); and (2) the LLM-as-a-Judge literature has matured, with dedicated benchmarks (Tan et al., 2025; Bi et al., 2026), training paradigms (Whitehouse et al., 2025; Deshpande et al., 2024), and systematic failure analyses (Wynn et al., 2025; Zhang et al., 2025a), to support a synthesis.

Existing surveys address either LLM judges broadly (Gu et al., 2024; Li et al., 2025a, 2024a), SLMs generally (Lu et al., 2024), or efficient LLMs from a systems perspective (Wan et al., 2023), but to our knowledge this is the first survey specifically examining *SLMs as judges*, which introduces distinct questions: how does parameter budget constrain judging capability? when do ensemble strategies with sub-10B models pay off? and can representation-based evaluation bypass generation entirely? The question of whether automated judges themselves need validation (Shankar et al., 2024) becomes particularly acute for smaller models with narrower training distributions.

Scope and Methodology. We survey papers through early 2026 from Semantic Scholar, arXiv, ACL Anthology, and Google Scholar. Inclusion required that (i) the work involve models with $\leq$14B parameters in an evaluative role, or (ii) it provide methodology or baselines adopted by SLM judge papers. This identified over 60 works spanning individual judging (§3), ensemble and debate strategies (§4), reliability (§5), and future directions (§6). Appendices A–L provide extended coverage.

Contributions. (1) A five-dimension taxonomy (Figure 1) unifying 60+ works; (2) five insights (I1–I5) with multi-paper evidence (Figure 2); (3) a comparison of 15+ judge systems across training paradigm, mode, and domain; (4) an open-problem agenda on calibration, adaptive inference, and domain generalization; and (5) an open-source repository cataloging the literature on SLM-as-a-Judge (`https://github.com/anishh15/Awesome-SLM-as-a-Judge`).

2 Background

We first fix the paradigm and vocabulary the survey builds on: how model-based judging is defined, and why a parameter budget became the central axis. This grounds the judge function and deployment economics used throughout.

2.1 The LLM-as-a-Judge Paradigm

LLM-based evaluation was systematized by Zheng et al. (2023) with MT-Bench, a static benchmark of 80 multi-turn questions across eight categories. The paradigm defines a *judge* $f_{\mathcal{J}}$ mapping a (question, response) or (question, response-A, response-B) tuple to a verdict $v \in \mathcal{V}$, where $\mathcal{V}$ is either a scalar (*pointwise direct assessment*) or a preference direction (*pairwise ranking*). A third mode, *listwise ranking*, orders multiple candidates simultaneously. GPT-4's agreement with expert humans exceeds 80%, comparable to human-to-human agreement, which established automated judging as a viable proxy (Zheng et al., 2023). Appendix A provides full historical context.

Subsequent work diversified both architectures and supervision: distilled judges (PandaLM, JudgeLM), rubric-conditioned scorers (Prometheus 2), human-preference-trained reward models (FLAMe), and compact generalists (GLIDER, 3.8B) progressively demonstrated that capable evaluation does not require frontier scale. Appendix A traces this full history.

2.2 Why SLMs? Definitions and Deployment Economics

Motivated by recent SLM surveys (Lu et al., 2024) that target the 100M–5B range, we broaden the operational definition: a *Small Language Model* has $\leq$14B *dense* parameters, typically deployable on a single consumer GPU (24GB VRAM class) without model parallelism. For MoE architectures, we apply this threshold to *active* (per-token) parameters; e.g., Qwen 3-235B-A22B activates only 22B parameters per forward pass but stores 235B total, occupying a boundary region where inference cost is small-model-like but memory footprint is not. Appendix B details the full SLM landscape and MoE boundary cases.

Let $f_{\mathcal{J}}^{(B)} : \mathcal{Q} \times \mathcal{R} \to \mathcal{V}$ denote a judge operating under token budget B, and let the *Instruction Following Rate* (IFR) be the fraction of evaluations producing a parseable verdict. Verga et al. (2024) report that a heterogeneous SLM panel is over 7$\times$ less expensive than a single GPT-4 judge, enabling continuous evaluation at $N = 10^6$ scale.

3 Taxonomy and Individual SLM Judges

Taxonomy. Figure 1 organizes SLM judging across five orthogonal dimensions, each yielding one of our five insights: **background and motivation** (§2); **individual judging** (this section); **multi-agent strategies** (§4); **reliability and meta-evaluation** (§5); and **challenges and future direc-**

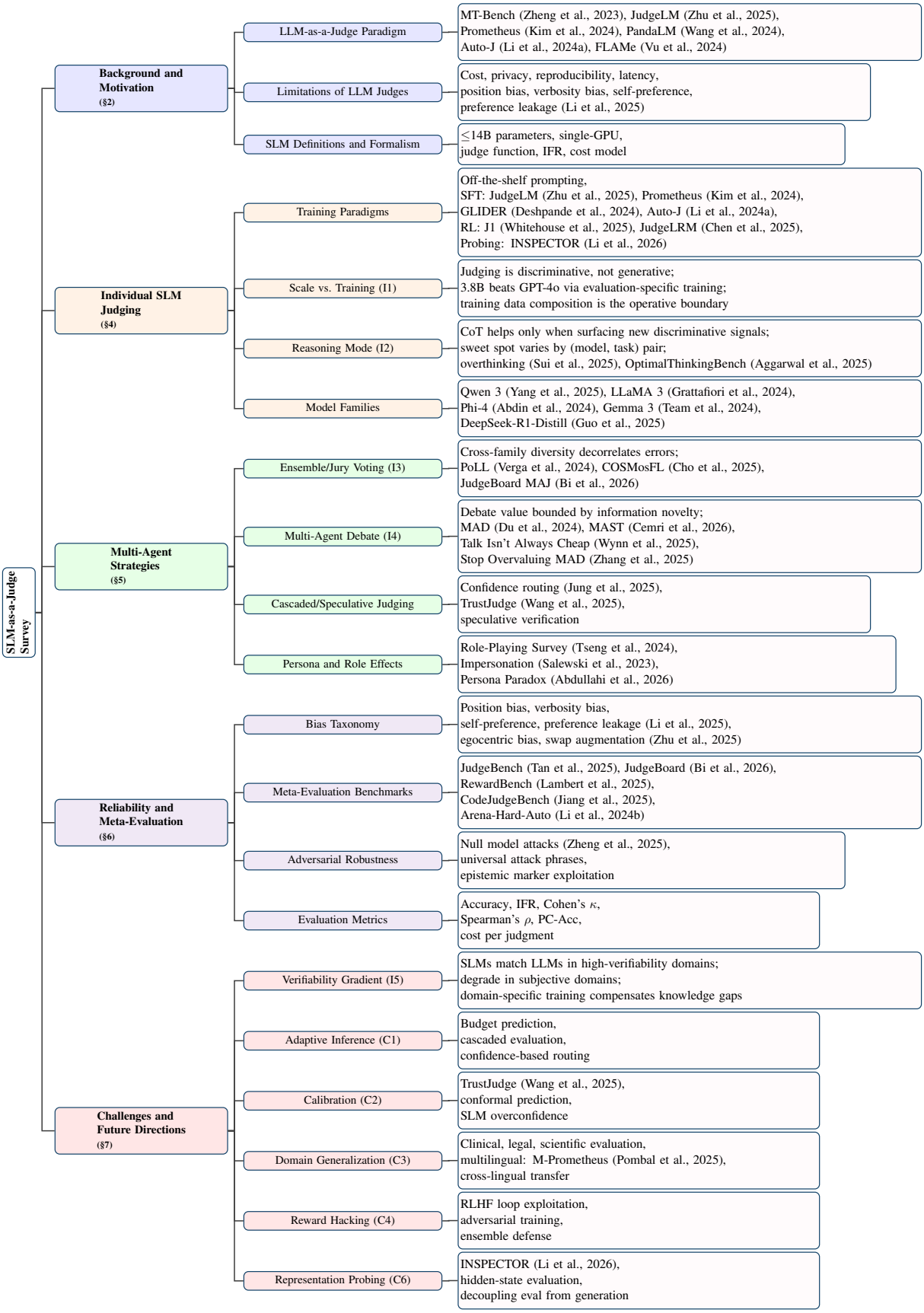


Figure 1: Taxonomy of the SLM-as-a-Judge survey. The five dimensions map to main paper sections (§2–§6), with insights I1–I5 and challenges C1–C6 positioned at their primary locations. Each leaf node lists representative papers and key concepts. Appendices A–L provide extended coverage of all branches.

tions (§6). We begin with the simplest dimension, the single judge acting alone.

The individual judge. An individual SLM judge realizes the map $f_{\mathcal{J}} : \mathcal{Q}\times\mathcal{R}^{(1:k)} \to \mathcal{V}$ of §2 with no auxiliary models, ensemble voting, or inter-agent communication. Its design space factorizes along four axes that interact in task-dependent ways,

$$f_{\mathcal{J}} = g(\underbrace{\pi}_{\text{train}}, \underbrace{B}_{\text{budget}}, \underbrace{m}_{\text{mode}}, \underbrace{\theta}_{\text{scale}}), \quad (1)$$

the training paradigm π (§3.1), the token budget B and inference mode $m \in \{\text{THINK}, \text{DIRECT}\}$ (§3.3), and the parameter count θ (§3.2); the first two dominate quality. Cascaded evaluation (§4.3) optimizes B dynamically. We take each axis in turn.

3.1 Training Paradigms

Off-the-shelf prompting. General-purpose SLMs can serve as zero-shot or few-shot judges without task-specific training; Jayarao et al. (2025) find that thinking-mode models (Qwen 3, 0.6B–4B) already achieve ∼10% higher accuracy than non-thinking counterparts through innate chain-of-thought. But off-the-shelf judges face three structural failure modes: inconsistent format adherence (IFR below 80% on complex rubrics for sub-3B models), unmitigated positional and verbosity biases, and susceptibility to adversarial phrases that inflate scores regardless of content.

Supervised fine-tuning (SFT). SFT on evaluation-specific data is the dominant paradigm. JudgeLM (Zhu et al., 2025) trains on GPT-4 judgments with *swap augmentation* (both orderings) to reduce position bias at the data level; Prometheus 2 (Kim et al., 2024b) adds rubric-conditioned Likert scoring; and Auto-J (Li et al., 2024b), a 13B generative judge, generalizes across 58 real-world scenarios (Appendix C).

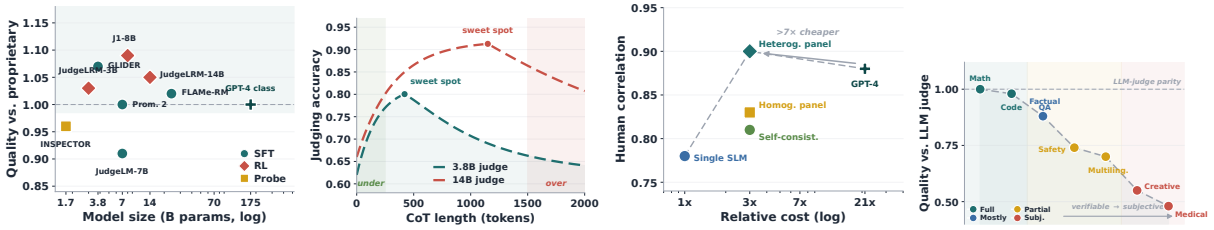
RL-based judge training. RL offers a second-order improvement by optimizing the judge directly against evaluation accuracy. J1 (Whitehouse et al., 2025) trains thinking judges via Group Relative Policy Optimization (GRPO), casting both verifiable and open-ended tasks into a unified verifiable reward, with a consistency reward that penalizes position-dependent verdicts *at the objective level*. JudgeLRM (Chen et al., 2025) trains 3B–14B judges with outcome-driven GRPO rewards and finds a negative correlation between SFT gains and the share of reasoning-demanding samples, confirming that SFT is structurally insufficient for reasoning-intensive judgment.

Representation-based judging. A different paradigm bypasses generation entirely. Li et al. (2026) introduce INSPECTOR, probing hidden states of a frozen small LM to predict aspect-level scores via lightweight linear probes. Its basis is the *Semantic Capacity Asymmetry Hypothesis*: evaluation is a discriminative task accessible in early-to-mid layers, whereas generation requires the full forward pass plus sampling (Appendix C).

3.2 Scale, Capability, and the Training-Data Boundary

A prevalent assumption in early LLM judging work was that larger models make better judges. The empirical record reveals a more nuanced picture once evaluation-specific training is controlled for.

EVALUATION-SPECIFIC TRAINING OUTPERFORMS SCALE BECAUSE JUDGING IS PRIMARILY DISCRIMINATIVE. **A 3.8B judge trained on evaluation data can outperform a far larger proprietary model such as GPT-4o, because recognizing quality differences requires pattern-matching against internalized evaluation criteria, not the full generative capacity needed for open-ended text production.** Table 1 summarizes key SLM judge systems (Table 5 in Appendix C gives the full cross-paradigm map), and Figure 2(a) plots their judging quality against model size: the trend is flat, not rising, once evaluation-specific training is present. GLIDER (3.8B) outperforms GPT-4o on FLASK (Deshpande et al., 2024); Selene Mini (8B) outperforms GPT-4o-mini across 11 OOD benchmarks (Alexandru et al., 2025); Prometheus 2 (7B) achieves GPT-4-level correlation (Kim et al., 2024b); JudgeLM-7B exceeds 90% agreement with GPT-4 (Zhu et al., 2025); and JudgeLRM-3B surpasses GPT-4 while JudgeLRM-7B outperforms DeepSeek-R1 by 2.79% F1 (Chen et al., 2025). These results point to a common explanation: targeted evaluation training appears to reorient the model toward rubric alignment, factual verification, and comparative reasoning, which require far less parametric capacity than open-ended generation. *The operative boundary is training data composition, not parameter count.* Appendix D provides expanded results.



(a) I1: training beats scale. (b) I2: reasoning sweet spot. (c) I3: cheap diversity. (d) I5: verifiability gradient.

Figure 2: *The four quantitative trends behind the survey's insights.* **(a)** With evaluation-specific training, judge quality is flat across model size; trained sub-8B judges sit at or above a proprietary baseline (I1). **(b)** Accuracy peaks then falls as reasoning budget grows; the smaller judge peaks earlier (I2). **(c)** A heterogeneous panel beats a single GPT-4 judge at over 7× lower cost (I3). **(d)** Quality falls from verifiable to subjective domains (I5). I4 (debate novelty) is discussed in §4. *All panels are schematic syntheses of qualitative trends across cited systems, not controlled meta-analyses.* See Tables 5 and 1 and Appendices C, D, E, J for underlying data.

Table 1: Selected SLM judge systems and their key properties. DA = direct assessment, PR = pairwise ranking, ORM = outcome reward model. Strength: primary advantage (Rubric = rubric-conditioned, OOD Gen. = out-of-distribution generalization, Coverage = domain breadth). †FLAMe-RM is included for comparison despite exceeding the 14B SLM threshold. All listed systems except FLAMe-RM are open-weights. See Table 5 in Appendix C for full quantitative comparison.

System	Base	Train	Mode	Strength	Open
JudgeLM	Vicuna-7B–33B	SFT	DA/PR	Speed	✓
Prometheus 2	Mistral-7B	SFT+Merge	DA+PR	Rubric	✓
Selene Mini	LLaMA-3.1-8B	SFT+DPO	DA/PR	OOD Gen.	✓
GLIDER	Phi-3.5-mini	SFT	DA	Coverage	✓
FLAMe-RM	PaLM-2-24B†	SFT-Human	ORM	Fairness	–
Auto-J	LLaMA-2-13B	SFT	DA/PR	Diversity	✓
J1	LLaMA-8B	GRPO	DA/PR	Reasoning	✓
JudgeLRM	3B–14B	GRPO	DA	F1 Score	✓
INSPECTOR	1.7B (frozen)	Probe	DA	Efficiency	✓

3.3 Token Budget Effects and Reasoning Mode

The reasoning budget B interacts with judging capability in a task-dependent way. Accuracy as a function of CoT length is non-monotone, so the optimal budget $B^\star(\theta, d) = \arg\max_B \mathrm{Acc}(B \mid \theta, d)$ depends on both judge scale θ and task difficulty d. Extended chain-of-thought (CoT) (Wei et al., 2022) reliably improves judgment for complex tasks (Jayarao et al., 2025; Jiang et al., 2025), yet multiple 2025 threads reveal that more tokens can hurt.

REASONING TOKENS IMPROVE EVALUATION ONLY WHEN THEY SURFACE NEW DISCRIMINATIVE SIGNALS. **Extended CoT reliably improves judgment for complex tasks because each reasoning step generates new discriminative features not available from the initial representation, but degrades verdict quality for simple tasks because the evaluative signal saturates before the token budget is exhausted, and additional tokens introduce noise through self-contradiction and hallucinated justification.** This produces the non-monotone curve in Figure 2(b). Jayarao et al. (2025) demonstrate that thinking models achieve ∼10% higher accuracy than non-thinking counterparts on judgment tasks; CodeJudgeBench confirms this for code evaluation (Jiang et al., 2025). Conversely, Aggarwal et al. (2025) show that sweet spots vary by (model, task-difficulty) pair, with over-reasoning degrading performance beyond task-dependent thresholds. Sui et al. (2025) document degradation from redundant elaboration and entropy accumulation, and Srivastava et al. (2026a) show reasoning models overthink even basic arithmetic. Shojaee et al. (2026) find that extended reasoning can create an "illusion of thinking" where traces appear deliberate but degrade accuracy beyond a task-complexity cliff, while Ghosal et al. (2026) show added test-time compute does not reliably help when the model lacks the underlying capability.

The key variable is whether additional tokens carry new discriminative information: each CoT step helps on complex evaluation but adds noise once signal saturates. Han et al. (2025) show budget-constrained prompting preserves accuracy while cutting compute, so adaptive budgets or dual-mode judges (thinking for hard cases, direct for easy ones) give the best cost-quality trade-off (Appendix D). Practitioners can detect hallucinated reasoning by comparing think-mode and direct-mode verdicts: divergence indicates new signal, while agreement suggests post-hoc rationalization.

3.4 Instruction Following Rate

A critical but understated metric is the IFR: the fraction of evaluations producing a parseable ver-

dict. An IFR below 1.0 creates silent failures, as a harness expecting a score digit drops or misparses freeform text; because these failures concentrate on the hardest cases, they bias benchmark comparisons downward exactly where accuracy matters most. Sub-3B non-thinking models reach only ∼75–85% IFR on complex rubrics, whereas above 7B near-100% IFR is achievable with format-reinforced training (Appendix D).

This bias is easy to quantify. If a harness scores only parseable verdicts and the parse rate on the hard subset (true accuracy a_h) is r_h, while the easy subset (accuracy $a_e \geq a_h$) parses at $r_e \geq r_h$, the reported accuracy on a set with hard-case fraction ϕ is

$$\widehat{\mathrm{Acc}} = \frac{\phi\, r_h\, a_h + (1-\phi)\, r_e\, a_e}{\phi\, r_h + (1-\phi)\, r_e}, \qquad (2)$$

which over-weights easy cases whenever $r_e > r_h$ and so overstates the true mean accuracy $\phi a_h + (1-\phi)a_e$. The gap grows with both the difficulty spread $a_e - a_h$ and the parse-rate spread $r_e - r_h$, so a judge can appear to improve simply by parsing more easy cases. This is why IFR must be reported alongside accuracy: two judges with equal $\widehat{\mathrm{Acc}}$ but different IFR are not comparable, and the lower-IFR judge is the less reliable one on exactly the cases that matter.

4 Multi-Agent Evaluation Strategies

Individual judges, however capable, carry systematic biases and failure modes (§5). Multi-agent strategies pool judgments across models to address these limitations. Two approaches with distinct motivations and empirical profiles dominate, ensemble/jury voting and multi-agent debate, with Figure 8 (Appendix F) giving the full taxonomy.

4.1 Ensemble and Jury Methods

Ensemble judging aggregates the verdicts of n independently queried judges, relying on the Condorcet Jury Theorem: if each juror is correct with probability $p > \frac{1}{2}$ and errors are independent, the majority-vote accuracy $\sum_{k>n/2} \binom{n}{k} p^k (1-p)^{n-k} \to 1$ as n grows, so independent errors cancel.

CROSS-FAMILY JURY DIVERSITY DECORRELATES ERRORS THROUGH ARCHITECTURAL INDEPENDENCE. **Heterogeneous SLM panels outperform single large judges at over 7× lower cost because distinct pretraining corpora and attention mechanisms yield more weakly correlated failure modes, better approximating the independence assumption that Condorcet aggregation relies on.** Figure 2(c) situates this result on the cost-quality plane. Verga et al. (2024) formalize the approach as PoLL (Panel of LLM Evaluators): a heterogeneous ensemble of Command R, GPT-3.5-Turbo, and Claude Haiku outperforms a single GPT-4 judge on multiple QA and preference benchmarks. Bi et al. (2026) demonstrate that Multi-Agent Judging (MAJ) with SLM backbones can match standalone LLM judges on MATH even when each individual SLM falls substantially short. Cho et al. (2025) confirm on fault localization that diverse small-model panels match large-model performance. Appendix E provides expanded results.

Family diversity matters more than stochastic diversity because each family embeds distinct failure modes arising from different pretraining corpora, architectures (alternating attention in Gemma, MoE routing in Qwen 3, distillation in DeepSeek-R1-Distill), and alignment objectives. When panel members come from disjoint families, their errors are closer to conditionally independent given the input, so majority voting approaches the Condorcet regime, whereas a homogeneous ensemble shares correlated failure modes. *The actionable consequence is to build three-model juries from distinct families (e.g., Phi-4-mini + LLaMA-3-8B + Qwen3-4B); intra-family temperature sampling yields only limited bias reduction.*

4.2 Multi-Agent Debate

Multi-agent debate (MAD) (Du et al., 2024) runs multiple LLM instances that exchange and critique responses over multiple rounds. The motivation is information aggregation: diverse agents surface distinct evidence, and cross-criticism enables error correction beyond single-agent reasoning.

DEBATE VALUE IS BOUNDED BY INFORMATION NOVELTY ACROSS ROUNDS, NOT BY ROUND COUNT. **Multi-agent debate improves evaluation when agents contribute genuinely different evidence or problem-solving strategies, but degrades performance in shared-context settings because additional rounds amplify sycophantic conformity rather than surfacing new information.** Wynn et al. (2025) and Zhang et al. (2025a) independently find MAD frequently underperforms single-agent baselines on binary correctness judgment, and the MAST taxonomy (Cemri et al., 2026) attributes ∼37% of multi-agent failures to *Inter-Agent Misalignment*, where agents prioritize consensus over accuracy, especially when an

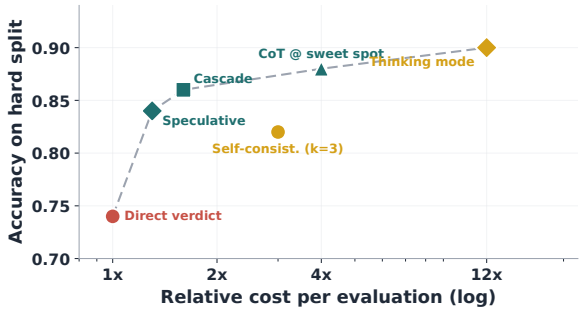


Figure 3: Cost-quality frontier for individual-judge deployment strategies (hard-split accuracy vs. relative cost, log scale). Cascade and speculative judging sit near the knee of the frontier; self-consistency lies below it. Marker color encodes the strategy family.

early verdict is expressed with high certainty (Appendix G, Figure 9). Conversely, debate reliably helps when information asymmetry is high: Du et al. (2024) show gains on factuality tasks where agents access different evidence fragments, and tool-augmented debate (e.g., code execution during evaluation) injects genuinely new discriminative signal at each round (Crupi et al., 2026). With shared inputs and no external tools, however, extra rounds amplify confidence rather than surfacing evidence. *The critical design variable is information asymmetry, not round count.*

4.3 Cascaded and Speculative Judging

A cost-effective alternative is cascaded judging: route evaluations to a small SLM judge, accept high-confidence verdicts, and escalate uncertain ones to a larger model (Jung et al., 2025). *Speculative judging* pushes this further: a small judge proposes a verdict and a large judge verifies only on disagreement. Figure 3 places these on the cost-quality plane: cascade and speculative judging sit at the frontier's knee; self-consistency lies below it (Appendix D).

5 Reliability and Meta-Evaluation

Whether using individual or multi-agent judges, evaluation reliability requires analysis along three axes: structural biases, adversarial vulnerabilities, and benchmark validity.

5.1 Bias Taxonomy

We highlight the four most studied bias categories here; Appendix H gives the full six-category taxonomy with mechanisms and mitigations. **Position bias** favors responses by presentational order rather than quality (Zheng et al., 2023); it is quantified by *position-consistent accuracy*, $\mathrm{PC} = \mathbb{1}[v_{ab} = v_{ba}]$, which credits a verdict only when it survives swapping the A/B order, and is mitigated by swap augmentation (Zhu et al., 2025), consistency-based RL rewards (Whitehouse et al., 2025), or inference-time swapping. **Verbosity bias** rewards longer responses regardless of accuracy (Chen et al., 2024; Szymanski et al., 2025), conflating information volume with quality. **Self-preference** favors outputs with low perplexity under the judge's own distribution, and extends to related lineages as *preference leakage* (Li et al., 2025b): a judge trained on data from the evaluated model's family inflates its scores. **Sycophancy bias** aligns the verdict with expressed user expectations or authority cues rather than substantive quality.

SLM judges carry amplified susceptibility along most of these axes due to limited reasoning capacity (Appendix H, Table 16), except self-preference, which is weaker in SLMs.

5.2 Adversarial Robustness

Judges also face structured adversarial attacks exploiting superficial signals. Zheng et al. (2025) show that constant, content-free responses achieve disproportionately high win rates; Zhao et al. (2025) find that a single token such as a colon can elicit false-positive rewards; and epistemic markers ("clearly", "definitively") inflate judged quality without improving content. SLM judges are particularly vulnerable due to weaker instruction following; heterogeneous ensembles provide partial protection since attacks tuned to one architecture transfer poorly to a diverse panel (Appendix K).

5.3 Benchmark Validity and Position-Consistent Accuracy

POSITION-CONSISTENT ACCURACY REVEALS THAT MUCH APPARENT JUDGE PERFORMANCE IS HEURISTIC-DRIVEN. **Under position-consistent accuracy protocols, even frontier judges degrade sharply on hard evaluation pairs, sometimes approaching random-choice baselines, indicating that much apparent performance relies on positional heuristics rather than substantive quality discrimination.** Tan et al. (2025) construct challenging pairs across knowledge, reasoning, math, and coding where one response contains a subtle but verifiable error; PC-Acc is dramatically lower than naive accuracy for all tested systems. Bi et al. (2026) confirm this on JudgeBoard for mathematics and Jiang et al.

(2025) for code (Appendix I).

This reframes the field's progress: gains on human-agreement benchmarks may reflect better alignment with human stylistic preferences rather than better reasoning evaluation. The practical implication is to always report PC-Acc alongside naive accuracy, since the gap between them measures how much apparent performance derives from heuristic exploitation. This reconciles with I1: evaluation-specific training delivers genuine parity on standard benchmarks, but PC-Acc reveals that this parity narrows on the hardest evaluation pairs and in subjective domains (I5), where all judges, including frontier models, can degrade toward chance on the most adversarial pairs. Additional validity concerns, including limitations of the human gold standard itself (Shankar et al., 2024) and dataset contamination risks for older benchmarks (Bedemariam et al., 2025), are discussed in Appendix I.

6 Challenges and Future Directions

SLM JUDGE RELIABILITY FOLLOWS THE VERIFIABILITY GRADIENT OF THE EVALUATION DOMAIN. **SLMs approach LLM-judge quality in high-verifiability domains (math, code) where external verification compensates for limited parametric knowledge, but degrade in subjective domains where internalized knowledge is the irreducible bottleneck.** The gap narrows as domain verifiability $\nu(d)$ rises (Figure 2(d)): SLM judges reach near parity on mathematics (Bi et al., 2026; Chen et al., 2025) and code (Jiang et al., 2025), but drop sharply for creative tasks where no external oracle exists, though principle-guided judges can still drive creative-writing improvement (Wei et al., 2025). The challenges below stem from this gradient; Appendix L expands them.

C1: Adaptive Inference-Time Compute. Fixed token budgets ignore that the optimal budget $B^\star(\theta, d)$ varies sharply with task difficulty (§3.3). Cascaded evaluation (Jung et al., 2025) escalates only when judge confidence falls below a threshold ($c < \tau$), routing easy cases to cheap judges and deferring the rest, but reliable confidence estimation remains open: SLM judges are systematically overconfident exactly where they are least accurate.

C2: Judge Calibration Without Human Labels. Sheng et al. (2025) apply conformal prediction for distribution-free intervals and TrustJudge (Wang et al., 2025) fixes score-comparison inconsistency, but general SLM-scale calibration frameworks remain undeveloped. SLMs are often *less* calibrated than base counterparts after RLHF; causal surrogate frameworks (Landesberg and Narayan, 2025) that audit reliability without exhaustive human labeling are promising (Appendix L).

C3: Domain Generalization Beyond Math and Code. High-stakes domains such as clinical, legal, and scientific peer review remain largely unaddressed. M-Prometheus (Pombal et al., 2025) takes a first step at multilingual coverage (20+ languages, 3B–14B), but domain-specialized SLM judges for non-English content are almost entirely absent. Multimodal SLM judges remain unexplored despite rapid multimodal generation growth (Appendix J).

C4: Resistance to Reward Hacking. Once an SLM judge supplies the RLHF signal, generators learn to produce superficially convincing reasoning that deceives it without improving quality. The adversarial robustness of SLM judges, with their weaker instruction following, is far less studied than that of LLM judges.

C5: Standardized Meta-Evaluation Protocols. No single benchmark spans all critical dimensions: position-consistent accuracy, human correlation, domain coverage, adversarial robustness, efficiency, and IFR. A unified suite would enable cross-paper comparison and standardized reporting.

C6: Representation-Based Evaluation. Probing frozen hidden states of sub-2B models approximates full-scale LLM judge performance on structured tasks (Li et al., 2026), because discriminative evaluation relies on intermediate representations rather than full decoding; scaling this to open-ended evaluation remains open.

7 Conclusion

This survey organizes 60+ works on SLMs as judges into a five-dimension taxonomy with five insights. The convergent finding is that **the resources required for reliable automated evaluation are far smaller than the field has assumed**: fine-tuned 3–8B judges match proprietary models on standard benchmarks (I1), reasoning tokens help only when they carry new signal (I2), and heterogeneous juries decorrelate errors (I3), while the information-novelty boundary (I4) and verifiability gradient (I5) mark where SLM judges suf-

fice and where human oversight remains necessary. The remaining open challenges (calibration, multilingual coverage, contamination resistance, and confidence-competence alignment) are prerequisites for high-stakes deployment. These findings reframe SLM judges not as a budget compromise but as the practical default for most automated evaluation, with frontier models reserved for cases at the subjective end of the verifiability gradient. The throughline is that judging is discriminative, and discrimination is cheaper than generation: scale buys knowledge, but most comes from targeted training, verification, or panel diversity, so progress lies not in larger judges but in better signals.

Limitations

This survey synthesizes findings from over 60 papers published through early 2026, but several limitations should be noted. **First**, we have not conducted new experiments; all quantitative claims are drawn from cited sources and subject to original study biases. **Second**, the survey focuses primarily on English-language judges; generalizing to low-resource languages risks underestimating domain-specific failure modes. **Third**, rapid field evolution means coverage through early 2026 may not capture later developments. **Fourth**, our literature selection may underrepresent domain-specific research outside major NLP venues. **Fifth**, the trend figures (Figure 2) are schematic syntheses, not controlled meta-analyses. **Finally**, IFR figures and cost estimates use varying setups, making cross-paper comparisons approximate.

Potential Risks. This work does not pose direct risks, but reliance on automated evaluation may introduce systematic biases that propagate into downstream model training. Deployment of SLM judges in critical applications (e.g., medical, legal, safety evaluation) should carefully consider robustness limitations documented in this survey. Over-reliance on a single judge architecture risks creating evaluation monocultures where shared failure modes go undetected. Practitioners should document judge identity, model version, and known biases to reduce evaluation drift over time.

Ethical Considerations

This survey evaluates existing research on small language models as judges using publicly available papers, benchmarks, and datasets, ensuring transparency and reproducibility. No private or sensitive data was used, and all surveyed systems are assessed based on their published results under fair conditions.

Automated judges influence model alignment through RLHF and data curation. Poor-quality judges can systematically corrupt training signals, amplifying biases at scale. Practitioners should: calibrate SLM judges against human-annotated sets before deployment; disclose judge identity, provenance, and known biases; avoid automated judges as sole arbiters of high-stakes decisions without human review; and audit probe features for demographic bias. The information-novelty boundary of debate (§4) underscores that consensus does not guarantee correctness.

Acknowledgements

The authors would like to acknowledge the vibrant open-source community for developing and maintaining the models, datasets, and evaluation frameworks that made this research possible. We are also grateful to the developers of the various small language models analyzed in this survey, whose work continues to advance the accessibility of AI evaluation.

References

Marah Abdin, Sahaj Agarwal, Ahmed Awadallah, Vidhisha Balachandran, Harkirat Behl, Lingjiao Chen, Gustavo de Rosa, Suriya Gunasekar, Mojan Javaheripi, Neel Joshi, and 1 others. 2025. Phi-4-reasoning technical report. *arXiv preprint arXiv:2504.21318*.

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, and 1 others. 2024. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*.

Tassallah Abdullahi, Shrestha Ghosh, Hamish S Fraser, Daniel León Tramontini, Adeel Abbasi, Ghada Bourjeily, Carsten Eickhoff, and Ritambhara Singh. 2026. The persona paradox: Medical personas as behavioral priors in clinical language models. *arXiv preprint arXiv:2601.05376*.

Abdelrahman Abouelenin, Atabak Ashfaq, Adam Atkinson, Hany Awadalla, Nguyen Bach, Jianmin Bao, Alon Benhaim, Martin Cai, Vishrav Chaudhary, Congcong Chen, and 1 others. 2025. Phi-4-mini technical report: Compact yet powerful multimodal language models via mixture-of-loras. *arXiv preprint arXiv:2503.01743*.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman,

Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Pranjal Aggarwal, Seungone Kim, Jack Lanchantin, Sean Welleck, Jason Weston, Ilia Kulikov, and Swarnadeep Saha. 2025. Optimalthinkingbench: Evaluating over and underthinking in llms. *arXiv preprint arXiv:2508.13141*.

Andrei Alexandru, Antonia Calvi, Henry Broomfield, Jackson Golden, Kyle Dai, Mathias Leys, Maurice Burger, Max Bartolo, Roman Engeler, Sashank Pisupati, and 1 others. 2025. Atla selene mini: A general purpose evaluation model. *arXiv preprint arXiv:2501.17195*.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, and 1 others. 2022. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*.

Rewina Bedemariam, Natalie Perez, Sreyoshi Bhaduri, Satya Kapoor, Alex Gil, Elizabeth Conjar, Ikkei Itoku, David Theil, Aman Chadha, and Naumaan Nayyar. 2025. Potential and perils of large language models as judges of unstructured textual data. *arXiv preprint arXiv:2501.08167*.

Zhenyu Bi, Gaurav Srivastava, Yang Li, Swastik Roy, Meng Lu, Morteza Ziyadi, and Xuan Wang. 2026. Judgeboard: Benchmarking and enhancing small language models for reasoning evaluation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 30076–30084.

Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbrenner, Yining Chen, Adrien Ecoffet, Manas Joglekar, Jan Leike, and 1 others. 2023. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision. *arXiv preprint arXiv:2312.09390*.

Mert Cemri, Melissa Z Pan, Shuyi Yang, Lakshya A Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, and 1 others. 2026. Why do multiagent llm systems fail? *Advances in Neural Information Processing Systems*, 38.

Guiming Hardy Chen, Shunian Chen, Ziche Liu, Feng Jiang, and Benyou Wang. 2024. Humans or llms as the judge? a study on judgement bias. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8301–8327.

Nuo Chen, Zhiyuan Hu, Qingyun Zou, Jiaying Wu, Qian Wang, Bryan Hooi, and Bingsheng He. 2025. Judgelrm: Large reasoning models as a judge. *arXiv preprint arXiv:2504.00050*.

Hyunjoon Cho, Sungmin Kang, Gabin An, and Shin Yoo. 2025. Cosmosfl: Ensemble of small language models for fault localisation. In *2025 IEEE/ACM International Workshop on Large Language Models for Code (LLM4Code)*, pages 17–24. IEEE.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Giuseppe Crupi, Rosalia Tufano, and Gabriele Bavota. 2026. Improving code generation via small language model-as-a-judge. *arXiv preprint arXiv:2602.11911*.

Darshan Deshpande, Selvan Sunitha Ravi, Sky CH-Wang, Bartosz Mielczarek, Anand Kannappan, and Rebecca Qian. 2024. Glider: Grading llm interactions and decisions using explainable ranking. *arXiv preprint arXiv:2412.14140*.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. 2024. Improving factuality and reasoning in language models through multiagent debate. In *Forty-first international conference on machine learning*.

Soumya Suvra Ghosal, Souradip Chakraborty, Avinash Reddy, Yifu Lu, Mengdi Wang, Dinesh Manocha, Furong Huang, Mohammad Ghavamzadeh, and Amrit Singh Bedi. 2026. Does thinking more always help? mirage of test-time scaling in reasoning models. *Advances in Neural Information Processing Systems*, 38:172664–172691.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, and 1 others. 2024. A survey on llm-as-a-judge. *The Innovation*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, and 1 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. 2025. Token-budget-aware llm reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 24842–24855.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Tianyu Hu, Zhen Tan, Song Wang, Huaizhi Qu, and Tianlong Chen. 2026. Multi-agent debate for llm judges with adaptive stability detection. *Advances in Neural Information Processing Systems*, 38:46504–46540.

Naman Jain, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2025. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, volume 2025, pages 58791–58831.

Pratik Jayarao, Himanshu Gupta, Neeraj Varshney, and Chaitanya Dwivedi. 2025. Explicit reasoning makes better judges: A systematic study on accuracy, efficiency, and robustness. *arXiv preprint arXiv:2509.13332*.

Hongchao Jiang, Yiming Chen, Yushi Cao, Hung-yi Lee, and Robby T Tan. 2025. Codejudgebench: Benchmarking llm-as-a-judge for coding tasks. *arXiv preprint arXiv:2507.10535*.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2024. Swe-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations*, volume 2024, pages 54107–54157.

Jaehun Jung, Faeze Brahman, and Yejin Choi. 2025. Trust or escalate: Llm judges with provable guarantees for human agreement. In *International Conference on Learning Representations*, volume 2025, pages 3101–3125.

Seungone Kim, Jay Shin, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Ryan Shin, Sungdong Kim, James Thorne, Minjoon Seo, and 1 others. 2024a. Prometheus: Inducing fine-grained evaluation capability in language models. In *International Conference on Learning Representations*, volume 2024, pages 29927–29962.

Seungone Kim, Juyoung Suk, Shayne Longpre, Bill Yuchen Lin, Jamin Shin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. 2024b. Prometheus 2: An open source language model specialized in evaluating other language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 4334–4353.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pages 611–626.

Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, and 1 others. 2025. Rewardbench: Evaluating reward models for language modeling. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 1755–1797.

Eddie Landesberg and Manjari Narayan. 2025. Causal judge evaluation: Calibrated surrogate metrics for llm systems. *arXiv preprint arXiv:2512.11150*.

Yebin Lee, Imseong Park, and Myungjoo Kang. 2024. Fleur: An explainable reference-free evaluation metric for image captioning using a large multimodal model. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3732–3746.

Dawei Li, Bohan Jiang, Liangjie Huang, Alimohammad Beigi, Chengshuai Zhao, Zhen Tan, Amrita Bhattacharjee, Yuxuan Jiang, Canyu Chen, Tianhao Wu, and 1 others. 2025a. From generation to judgment: Opportunities and challenges of llm-as-a-judge. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 2757–2791.

Dawei Li, Renliang Sun, Yue Huang, Ming Zhong, Bohan Jiang, Jiawei Han, Xiangliang Zhang, Wei Wang, and Huan Liu. 2025b. Preference leakage: A contamination problem in llm-as-a-judge. *arXiv preprint arXiv:2502.01534*.

Haitao Li, Qian Dong, Junjie Chen, Huixue Su, Yujia Zhou, Qingyao Ai, Ziyi Ye, and Yiqun Liu. 2024a. Llms-as-judges: a comprehensive survey on llm-based evaluation methods. *arXiv preprint arXiv:2412.05579*.

Junlong Li, Shichao Sun, Weizhe Yuan, Run-Ze Fan, Pengfei Liu, and 1 others. 2024b. Generative judge for evaluating alignment. In *International Conference on Learning Representations*, volume 2024, pages 27547–27574.

Zhuochun Li, Yong Zhang, Ming Li, Yuelyu Ji, Yiming Zeng, Ning Cheng, Yun Zhu, Yanmeng Wang, Shaojun Wang, Jing Xiao, and 1 others. 2026. Rethinking llm-as-a-judge: Representation-as-a-judge with small language models via semantic capacity asymmetry. *arXiv preprint arXiv:2601.22588*.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. 2024. Encouraging divergent thinking in large language models through multi-agent debate. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pages 17889–17904.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pages 39578–39601.

Dachuan Lin, Guobin Shen, Zihao Yang, Tianrong Liu, Dongcheng Zhao, and Yi Zeng. 2025. Efficient llm safety evaluation through multi-agent debate. *arXiv preprint arXiv:2511.06396*.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Truthfulqa: Measuring how models mimic human falsehoods. In *Proceedings of the 60th annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 3214–3252.

Zhenyan Lu, Xiang Li, Dongqi Cai, Rongjie Yi, Fangming Liu, Xiwen Zhang, Nicholas D Lane, and Mengwei Xu. 2024. Small language models: Survey, measurements, and insights. *arXiv preprint arXiv:2409.15790*.

Wenjie Ma, Jingxuan He, Charlie Snell, Tyler Griggs, Sewon Min, and Matei Zaharia. 2025. Reasoning models can be effective without thinking. *arXiv preprint arXiv:2504.09858*.

Md Motaleb Hossen Manik and Ge Wang. 2026. Gemma 4, phi-4, and qwen3: Accuracy-efficiency tradeoffs in dense and moe reasoning language models. *arXiv preprint arXiv:2604.07035*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

José Pombal, Dongkeun Yoon, Patrick Fernandes, Ian Wu, Seungone Kim, Ricardo Rei, Graham Neubig, and André FT Martins. 2025. M-prometheus: A suite of open multilingual llm judges. *arXiv preprint arXiv:2504.04953*.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. Comet: A neural framework for mt evaluation. In *Proceedings of the 2020 conference on empirical methods in natural language processing (emnlp)*, pages 2685–2702.

Michael J Ryan, Omar Shaikh, Aditri Bhagirath, Daniel Frees, William Barr Held, and Diyi Yang. 2025. Synthesizeme! inducing persona-guided prompts for personalized reward models in llms. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8045–8078.

Leonard Salewski, Stephan Alaniz, Isabel Rio-Torto, Eric Schulz, and Zeynep Akata. 2023. In-context impersonation reveals large language models' strengths and biases. *Advances in neural information processing systems*, 36:72044–72057.

Shreya Shankar, JD Zamfirescu-Pereira, Björn Hartmann, Aditya Parameswaran, and Ian Arawjo. 2024. Who validates the validators? aligning llm-assisted evaluation of llm outputs with human preferences. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, pages 1–14.

Huanxin Sheng, Xinyi Liu, Hangfeng He, Jieyu Zhao, and Jian Kang. 2025. Analyzing uncertainty of llm-as-a-judge: Interval evaluations with conformal prediction. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 11297–11339.

Parshin Shojaee, Iman Mirzadeh, Maxwell Horton, Samy Bengio, Mehrdad Farajtabar, and 1 others. 2026. The illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity. *Advances in Neural Information Processing Systems*, 38:108018–108059.

Gaurav Srivastava, Zhenyu Bi, Meng Lu, and Xuan Wang. 2025a. DEBATE, TRAIN, EVOLVE: Self-Evolution of language model reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 32764–32810, Suzhou, China. Association for Computational Linguistics.

Gaurav Srivastava, Shuxiang Cao, and Xuan Wang. 2025b. ThinkSLM: Towards reasoning in small language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 32612–32662, Suzhou, China. Association for Computational Linguistics.

Gaurav Srivastava, Aafiya Hussain, Sriram Srinivasan, and Xuan Wang. 2026a. Do llms overthink basic math reasoning? benchmarking the accuracy-efficiency tradeoff in language models. *Preprint*, arXiv:2507.04023.

Gaurav Srivastava, Aafiya Hussain, Chi Wang, Yingyan Celine Lin, and Xuan Wang. 2026b. Efgen: Enabling small language models as capable autonomous agents. *Preprint*, arXiv:2602.00887.

Gaurav Srivastava, Aafiya Shamshad Hussain, Zhenyu Bi, Swastik Roy, Priya Pitre, Meng Lu, Morteza Ziyadi, and Xuan Wang. 2026c. Beyondbench: Contamination-resistant evaluation of reasoning in language models. In *The Fourteenth International Conference on Learning Representations*.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, and 1 others. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*.

Annalisa Szymanski, Noah Ziems, Heather A Eicher-Miller, Toby Jia-Jun Li, Meng Jiang, and Ronald A Metoyer. 2025. Limitations of the llm-as-a-judge approach for evaluating llm outputs in expert knowledge tasks. In *Proceedings of the 30th international conference on intelligent user interfaces*, pages 952–966.

Sijun Tan, Siyuan Zhuang, Kyle Montgomery, William Tang, Alejandro Cuadron, Chenguang Wang, Raluca Popa, and Ion Stoica. 2025. Judgebench: A benchmark for evaluating llm-based judges. In *International Conference on Learning Representations*, volume 2025, pages 63277–63303.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, and 1 others. 2024a. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, and 1 others. 2024b. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Yu-Min Tseng, Yu-Chao Huang, Teng-Yun Hsiao, Wei-Lin Chen, Chao-Wei Huang, Yu Meng, and Yun-Nung Chen. 2024. Two tales of persona in llms: A survey of role-playing and personalization. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 16612–16631.

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

Tu Vu, Kalpesh Krishna, Salaheddin Alzubi, Chris Tar, Manaal Faruqui, and Yun-Hsuan Sung. 2024. Foundational autoraters: Taming large language models for better automatic evaluation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17086–17105.

Zhongwei Wan, Xin Wang, Che Liu, Samiul Alam, Yu Zheng, Jiachen Liu, Zhongnan Qu, Shen Yan, Yi Zhu, Quanlu Zhang, and 1 others. 2023. Efficient large language models: A survey. *arXiv preprint arXiv:2312.03863*.

Boxin Wang, Weixin Chen, Hengzhi Pei, Chulin Xie, Mintong Kang, Chenhui Zhang, Chejian Xu, Zidi Xiong, Ritik Dutta, Rylan Schaeffer, and 1 others. 2023a. Decodingtrust: A comprehensive assessment of trustworthiness in {GPT} models. *Advances in Neural Information Processing Systems*, 36.

Junlin Wang, Siddhartha Jain, Dejiao Zhang, Baishakhi Ray, Varun Kumar, and Ben Athiwaratkun. 2024a. Reasoning in token economies: budget-aware evaluation of llm reasoning strategies. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 19916–19939.

Tianlu Wang, Ilia Kulikov, Olga Golovneva, Ping Yu, Weizhe Yuan, Jane Dwivedi-Yu, Richard Yuanzhe Pang, Maryam Fazel-Zarandi, Jason Weston, and Xian Li. 2024b. Self-taught evaluators. *arXiv preprint arXiv:2408.02666*.

Tianlu Wang, Ping Yu, Xiaoqing Ellen Tan, Sean O'Brien, Ramakanth Pasunuru, Jane Dwivedi-Yu, Olga Golovneva, Luke Zettlemoyer, Maryam Fazel-Zarandi, and Asli Celikyilmaz. 2023b. Shepherd: A critic for language model generation. *arXiv preprint arXiv:2308.04592*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Yidong Wang, Yunze Song, Tingyuan Zhu, Xuanwang Zhang, Zhuohao Yu, Hao Chen, Chiyu Song, Qiufeng Wang, Cunxiang Wang, Zhen Wu, and 1 others. 2025. Trustjudge: Inconsistencies of llm-as-a-judge and how to alleviate them. *arXiv preprint arXiv:2509.21117*.

Yidong Wang, Zhuohao Yu, Wenjin Yao, Zhengran Zeng, Linyi Yang, Cunxiang Wang, Hao Chen, Chaoya Jiang, Rui Xie, Jindong Wang, and 1 others. 2024c. Pandalm: An automatic evaluation benchmark for llm instruction tuning optimization. In *International Conference on Learning Representations*, volume 2024, pages 43573–43593.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Xiaolong Wei, Bo Lu, Xingyu Zhang, Zhejun Zhao, Dongdong Shen, Long Xia, and Dawei Yin. 2025. Igniting creative writing in small language models: Llm-as-a-judge versus multi-agent refined rewards. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 17171–17197.

Chenxi Whitehouse, Tianlu Wang, Ping Yu, Xian Li, Jason Weston, Ilia Kulikov, and Swarnadeep Saha. 2025. J1: Incentivizing thinking in llm-as-a-judge via reinforcement learning. *arXiv preprint arXiv:2505.10320*.

Andrea Wynn, Harsh Satija, and Gillian Hadfield. 2025. Talk isn't always cheap: Understanding failure modes in multi-agent debate. *arXiv preprint arXiv:2509.05396*.

Wei Xiong, Wenting Zhao, Weizhe Yuan, Olga Golovneva, Tong Zhang, Jason Weston, and Sainbayar Sukhbaatar. 2025. Stepwiser: Stepwise generative judges for wiser reasoning. *arXiv preprint arXiv:2508.19229*.

Austin Xu, Srijan Bansal, Yifei Ming, Semih Yavuz, and Shafiq Joty. 2025a. Does context matter? contextualjudgebench for evaluating llm-based judges in contextual settings. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9541–9564.

Austin Xu, Yilun Zhou, Xuan-Phi Nguyen, Caiming Xiong, and Shafiq Joty. 2025b. J4r: Learning to judge with equivalent initial state group relative policy optimization. *arXiv preprint arXiv:2505.13346*.

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

Jiayi Ye, Yanbo Wang, Yue Huang, Dongping Chen, Qihui Zhang, Nuno Moniz, Tian Gao, Werner Geyer, Chao Huang, Pin-Yu Chen, and 1 others. 2025. Justice or prejudice? quantifying biases in llm-as-a-judge. In *International Conference on Learning Representations*, volume 2025, pages 102351–102390.

Hangfan Zhang, Zhiyao Cui, Jianhao Chen, Xinrun Wang, Qiaosheng Zhang, Zhen Wang, Dinghao Wu, and Shuyue Hu. 2025a. Stop overvaluing multi-agent debate–we must rethink evaluation and embrace model heterogeneity. *arXiv preprint arXiv:2502.08788*.

Qiyuan Zhang, Yufei Wang, Yuxin Jiang, Liangyou Li, Chuhan Wu, Yasheng Wang, Xin Jiang, Lifeng Shang, Ruiming Tang, Fuyuan Lyu, and 1 others. 2025b. Crowd comparative reasoning: Unlocking comprehensive evaluations for llm-as-a-judge. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5059–5074.

Taolin Zhang, Maosong Cao, Alexander Lam, Songyang Zhang, and Kai Chen. 2025c. Compassjudger-2: Towards generalist judge model via verifiable rewards. *arXiv preprint arXiv:2507.09104*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Jian Zhao, Runze Liu, Kaiyan Zhang, Zhimu Zhou, Junqi Gao, Dong Li, Jiafei Lyu, Zhouyi Qian, Biqing Qi, Xiu Li, and 1 others. 2026. Genprm: Scaling test-time compute of process reward models via generative reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 34932–34940.

Yulai Zhao, Haolin Liu, Dian Yu, Sunyuan Kung, Meijia Chen, Haitao Mi, and Dong Yu. 2025. One token to fool llm-as-a-judge. *arXiv preprint arXiv:2507.08794*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623.

Xiaosen Zheng, Tianyu Pang, Chao Du, Qian Liu, Jing Jiang, and Min Lin. 2025. Cheating automatic llm benchmarks: Null models achieve high win rates. In *International Conference on Learning Representations*, volume 2025, pages 61994–62027.

Lianghui Zhu, Xinggang Wang, and Xinlong Wang. 2025. Judgelm: Fine-tuned large language models are scalable judges. In *International Conference on Learning Representations*, volume 2025, pages 51257–51296.

Appendix

This appendix provides supplementary material including extended background, model landscape details, training methodology analysis, individual and ensemble judging results, debate mechanisms, bias documentation, benchmark descriptions, domain-specific evaluation, adversarial robustness analysis, and a research agenda.

Contents of the Appendix

A Background: The LLM-as-a-Judge Paradigm **17**

A.1 The Evaluation Problem in NLP . . . 17
A.2 The LLM-as-a-Judge Paradigm: Formalization . . . 18
A.3 Key Judge Systems: A Comprehensive Timeline . . . 18
A.4 Evaluation Modalities: A Structured Comparison . . . 20
A.5 Cost Model and Deployment Tradeoffs . . . 21
A.6 Constitutional AI and the Self-Critique Paradigm . . . 21
A.7 LLM Judging and Scalable Oversight . . . 22
A.8 Surveys and Related Work . . . 22

B The SLM Landscape **22**

B.1 Defining "Small": The Parameter Boundary Debate . . . 22
B.2 The Key SLM Model Families (Early 2026) . . . 23
B.3 Architectural Trends Relevant to Judging . . . 26
B.4 SLM Capability Characterization for Judging . . . 27
B.5 Inference Infrastructure for SLM Judge Deployment . . . 27
B.6 The Emerging SLM Judge Ecosystem . . . 28

C Training Methodologies for SLM Judges **29**

C.1 A Consolidated Map of SLM Judge Systems . . . 29
C.2 Overview of Training Paradigms . . . 29
C.3 Off-the-Shelf Judging: Baselines and Limits . . . 30
C.4 Supervised Fine-Tuning for Judging . . . 30
C.5 Reinforcement Learning for Judge Training . . . 32
C.6 Reward Model Training: ORM vs. PRM . . . 34
C.7 Knowledge Distillation for Judges . . . 35
C.8 Representation-Based Judging: Beyond Generation . . . 35
C.9 Summary: Training Paradigm Comparison . . . 36

D Individual Judging: Token Budget and Scale **37**

D.1 The Individual Judge: Scope and Design Space . . . 37
D.2 Token Budget Analysis for SLM Judges . . . 38
D.3 Thinking Modes in SLM Judges . . . 40
D.4 Scale-Capability Relationship for Individual Judges . . . 41
D.5 The Overthinking Phenomenon in SLM Judges . . . 42
D.6 Instruction Following Rate (IFR) Analysis . . . 44
D.7 Cascade and Routing Architectures . . . 44
D.8 Deployment Strategy Summary . . . 45

E Ensemble and Jury Methods **45**

E.1 Motivation: Why Aggregate Multiple Judges? . . . 45
E.2 PoLL: Panel of LLM Evaluators . . . 46
E.3 COSMosFL: Cross-Family SLM Ensembles . . . 47
E.4 JudgeBoard and the MAJ Framework . . . 47
E.5 Self-Consistency as Single-Model Ensemble . . . 48
E.6 Aggregation Functions: Theory and Practice . . . 49
E.7 Cost Analysis for Ensemble Judging . . . 49
E.8 Recommended Panel Recipes . . . 50
E.9 Limits of Ensemble Judging . . . 50

F Multi-Agent Evaluation Strategy Taxonomy **51**

G Multi-Agent Debate: Mechanisms and Failure Modes **51**

G.1 The Debate Paradigm: Origins and Motivation . . . 51
G.2 Empirical Evidence of Potential Benefits . . . 52
G.3 Failure Modes: A Critical Analysis . . . 52
G.4 MAST: A Systematic Failure Taxonomy . . . 54
G.5 SLM-Specific Dynamics in Multi-Agent Debate . . . 55
G.6 Adaptive Convergence and Stopping Criteria . . . 56
G.7 Model Heterogeneity as the "Universal Antidote" . . . 56
G.8 Synthesis: When to Use Debate vs. Ensemble . . . 57

H Biases in LLM and SLM Judges **57**

H.1 The Structural Nature of Judge Bias . . . 57
H.2 Bias Taxonomy . . . 58
H.3 SLM-Specific Bias Amplification . . . 61
H.4 Bias Interaction Effects . . . 61
H.5 Mitigation Strategies: Comprehensive Reference . . . 62
H.6 The CALM Framework: Systematic Bias Quantification . . . 63

I Benchmarks, Datasets, and Metrics **63**

I.1 The Meta-Evaluation Problem . . . 63
I.2 Judge-Specific Evaluation Benchmarks . . . 64
I.3 Meta-Evaluation Metrics . . . 66
I.4 Benchmark Design Principles . . . 67
I.5 Benchmark Summary . . . 68
I.6 Meta-Evaluation Protocols: Recommended Practices . . . 68
I.7 The Human Gold Standard Problem . . . 69
I.8 Dataset Contamination Risks . . . 69

J Domain Generalization: Code, Safety, and Multilingual **70**

J.1 The Verifiability Gradient: A Unifying Framework . . . 70
J.2 Code Evaluation . . . 70
J.3 Safety Evaluation . . . 72
J.4 Multilingual Evaluation . . . 74
J.5 Medical, Legal, and High-Stakes Domains . . . 75
J.6 Agentic Evaluation: The Emerging Frontier . . . 76
J.7 Domain Deployment Matrix . . . 76

K Adversarial Robustness and Persona Effects **76**

K.1 The Robustness Problem for SLM Judges . . . 77
K.2 Attack Taxonomy: Structural Classification . . . 77
K.3 Null Model Exploitation and Benchmark Gaming . . . 77
K.4 Prompt Injection and Instruction Override . . . 78
K.5 Reward Hacking in Judge-Supervised Training Loops . . . 79
K.6 Persona and Role-Playing Sensitivity . . . 80
K.7 Robustness: Trained vs. Prompted Judges . . . 81
K.8 SLM-Specific Vulnerability Analysis . . . 82
K.9 Integrated Defense Framework . . . 82
K.10 The Co-Evolutionary Arms Race . . . 83

L Extended Future Directions **84**

L.1 Overview: How This Agenda Is Structured . . . 84
L.2 Direction 1: Representation-Based Evaluation . . . 84
L.3 Direction 2: Adaptive Inference-Time Compute . . . 85
L.4 Direction 3: Calibrated Uncertainty Quantification . . . 85
L.5 Direction 4: Scalable Oversight and Weak-to-Strong Generalization . . . 86
L.6 Direction 5: Agentic and Multi-Step Evaluation . . . 87
L.7 Direction 6: Multilingual and Cross-Cultural Judge Training . . . 87
L.8 Direction 7: Continual Learning and Judge Maintenance . . . 88
L.9 Direction 8: Standardized Meta-Evaluation Protocols . . . 88
L.10 Direction 9: Training Methodology for SLM Judges . . . 89
L.11 Direction 10: Societal Implications and Deployment Ethics . . . 89
L.12 Research Agenda Summary . . . 90

The appendix follows the same arc as the main paper. We first establish *where the field came from* (the evaluation problem and the SLM landscape: §A–B), then *how SLM judges are built and behave alone* (training and individual judging: §C–D), then *how they are combined* (ensembles, the strategy taxonomy, and debate: §E–G), then *how far we can trust them* (biases, benchmarks: §H–I), and finally *where they break and where to go next* (domains, robustness, and the research agenda: §J–L). Each section expands a claim made in the main paper with full mechanistic detail.

A Background: The LLM-as-a-Judge Paradigm

This section expands §2, tracing the evaluation problem from n-gram metrics to model-based judges and formalizing the judge function that the rest of the survey builds on.

A.1 The Evaluation Problem in NLP

Language model evaluation has always faced a tension between *scalability* and *validity*. Human evaluation, the undisputed gold standard, is slow, expensive, and difficult to reproduce: assembling expert annotators for the thousands of iterative model comparisons required by modern RLHF pipelines (Ouyang et al., 2022) is infeasible at production scale. The field compensated with automatic metrics, but the growing capability of generative models exposed structural flaws in each successive paradigm.

The n-gram era (pre-2022). BLEU (Papineni et al., 2002) and ROUGE measure n-gram overlap between candidate and reference responses. These metrics suffer from four fundamental deficiencies:

- **Semantic insensitivity**: Valid paraphrases sharing no words with the reference receive near-zero scores.
- **Hallucination blindness**: Fluent, factually fabricated responses can achieve high BLEU scores if they share surface vocabulary with the reference.
- **Goodhart's Law susceptibility**: Optimizing for BLEU causes models to shorten outputs and repeat reference phrases, inflating scores while degrading quality.
- **Reference dependency**: Performance varies with reference text quality and fails to represent the diversity of valid responses for open-ended tasks.

BERTScore (Zhang et al., 2019) and MoverScore partially addressed the semantic gap by replacing n-gram matching with contextual embedding similarity, but they still cannot evaluate logical reasoning, factual accuracy, or multi-step argumentation.

The learned metric era (2021–2022). The rise of instruction-tuned models created evaluation demands that no n-gram or embedding metric could meet. Open-ended questions have no single correct reference answer; quality depends on tone, depth, factual accuracy, and coherence simultaneously. COMET (Rei et al., 2020) and BLEURT demonstrated that learned metrics correlate substantially better with human judges than n-gram metrics for translation, but both still required reference translations and operated only within constrained task types.

The RLHF era and human bottleneck (2022–2023). RLHF (Ouyang et al., 2022) aligned language models with human preferences through a three-stage pipeline: supervised fine-tuning, reward model training on human preference pairs, and PPO optimization. This framework was transformative for model quality but created a new bottleneck: each training run required thousands of fresh human preference labels; each domain required domain-expert annotators; each model update required a new batch of comparison pairs. Human data curation became the primary rate-limiting factor in alignment research.

THE N-GRAM-TO-NEURAL-TO-MODEL PROGRESSION FOLLOWS AN ABSTRACTION GRADIENT. **Each generation of evaluation methodology increased the abstraction level of the quality signal being captured: n-gram metrics measure lexical overlap, embedding metrics measure semantic similarity, and model-based judges measure functional quality against multi-dimensional criteria.** This progression is not coincidental. As language models became capable of producing fluent, contextually

appropriate text, lower-abstraction metrics lost discriminative power. N-gram metrics cannot distinguish between a factually correct and a factually fabricated response if both are fluent; embedding metrics cannot distinguish between a logically valid and an invalid argument if both use similar vocabulary. Only model-based evaluation operates at the abstraction level required to assess the functional properties (correctness, helpfulness, safety) that define modern response quality. The implication is that future evaluation demands, particularly for reasoning-heavy and domain-specialized tasks, will require judges that operate at even higher abstraction levels, explaining the push toward tool-augmented and representation-based evaluation.

A.2 The LLM-as-a-Judge Paradigm: Formalization

The modern formalization was established by Zheng et al. (2023) with MT-Bench and Chatbot Arena. The paradigm defines a *judge function* $f_{\mathcal{J}}$ that maps evaluation inputs to verdicts:

$$f_{\mathcal{J}} : \mathcal{Q} \times \mathcal{R}^{(1:k)} \to \mathcal{V} \tag{3}$$

where $\mathcal{Q}$ is the space of evaluation questions, $\mathcal{R}^{(1:k)}$ is a tuple of k candidate responses (k=1 for pointwise, k=2 for pairwise, k>2 for listwise), and $\mathcal{V}$ is the verdict space. For pointwise evaluation, $\mathcal{V} = \{1, 2, 3, 4, 5\}$ (Likert scale) or $\mathbb{R}$ (continuous). For pairwise, $\mathcal{V} = \{\texttt{A}, \texttt{B}, \texttt{Tie}\}$. For listwise, $\mathcal{V}$ is a permutation of the k-element response set.

A critical extension is the *rubric-conditioned* judge:

$$f_{\mathcal{J}}^{(\rho)} : \mathcal{Q} \times \mathcal{R}^{(1:k)} \times \rho \to \mathcal{V} \tag{4}$$

where $\rho \in \mathcal{P}$ is a rubric that conditions the judge's output on user-specified evaluation dimensions. Prometheus (Kim et al., 2024a) and GLIDER (Deshpande et al., 2024) operationalize this form, enabling fine-grained multi-criterion evaluation.

Beyond rubric conditioning, some systems (Li et al., 2024b) produce structured *critiques*: explanatory rationales identifying specific aspects that satisfy or violate evaluation criteria:

$$f_{\mathcal{J}}^{(\text{crit})} : \mathcal{Q} \times \mathcal{R}^{(1:k)} \times \rho \to (\mathcal{V}, \mathcal{C}) \tag{5}$$

where $\mathcal{C}$ is the space of natural-language critiques. Critique-conditioned judges provide interpretable, auditable evaluation and can serve as natural-language feedback in iterative refinement loops.

MT-Bench: The first systematic evaluation. MT-Bench (Zheng et al., 2023) consists of 80 multi-turn questions across eight categories (writing, roleplay, extraction, reasoning, math, coding, STEM, humanities). GPT-4-as-judge achieved >80% agreement with expert human annotators, comparable to human-to-human agreement. A key finding was that GPT-4's preference flipped in roughly one-third of cases when response presentation order was swapped, revealing the prevalence of position bias even in frontier models.

Chatbot Arena: Crowdsourced preference at scale. Chatbot Arena (Zheng et al., 2023) collected >30,000 human pairwise preferences from crowdsourced users, establishing a gold-standard ELO-style ranking. This ranking became the authoritative external measure against which automated judge systems compare their alignment, providing a living, human-annotated benchmark that is resistant to overfitting.

A.3 Key Judge Systems: A Comprehensive Timeline

Table 2 summarizes the key milestones. We organize the timeline into three phases.

2023: The fine-tuning wave. The first wave of dedicated judge models trained LLMs specifically on evaluation data.

JudgeLM. Zhu et al. (2025) trained models at 7B, 13B, and 33B scales on GPT-4-generated judgment data. The 7B variant achieves >90% agreement with GPT-4 and can evaluate 5,000 response pairs in approximately 3 minutes on 8×A100 GPUs. JudgeLM introduced *swap augmentation*: evaluating both orderings of response pairs during training to reduce position bias. It also introduced *reference support* (providing a gold reference) and *reference drop* (randomly omitting the reference during training to improve reference-free generalization).

PandaLM. Wang et al. (2024c) distilled evaluation capability from GPT-3.5 into a LLaMA-based model, demonstrating competitive performance on a 1,000-sample human-annotated test set. PandaLM was the first judge model to directly tackle automated hyperparameter search for instruction-tuned models.

Shepherd. Wang et al. (2023b) developed a critic model that generates natural-language feedback identifying errors, inconsistencies, and missing information in model outputs. Shepherd demonstrated that training on community feedback data (Stack Exchange, human annotations) produces critic models that outperform ChatGPT on feedback quality, establishing the distinction between *scoring* (producing a verdict) and *critiquing* (producing diagnostic feedback) as complementary evaluation modes.

Prometheus. Kim et al. (2024a) introduced rubric-conditioned evaluation with fine-grained, customizable evaluation criteria. Trained on the Feedback Collection dataset, Prometheus demonstrated that conditioning the judge on explicit rubrics substantially improves evaluation validity compared to unconditioned scoring.

Auto-J. Li et al. (2024b) developed a 13B generative judge covering 58 diverse, real-world scenarios. Auto-J supports both pairwise comparison and pointwise evaluation, provides detailed natural-language critiques, and demonstrates reduced positional bias compared to non-specialized baselines.

2024: Unified evaluation, multi-task breadth, and public data.

Prometheus 2. Kim et al. (2024b), building on the original Prometheus (Kim et al., 2024a), unified direct assessment and pairwise ranking through weight merging: two separate models (Mistral-7B-Instruct-v0.2 and Mixtral-8x7B-Instruct-v0.1) trained on the Feedback Collection (pointwise) and Preference Collection (pairwise) datasets are merged into a single evaluator. Prometheus 2 achieves GPT-4-level correlation on MT-Bench, VicunaBench, and FLASK while supporting custom evaluation rubrics.

FLAMe. Vu et al. (2024) trained on $>$100 quality assessment tasks encompassing $>$5M human judgments drawn exclusively from permissively licensed datasets. FLAMe-RM achieves 87.8% accuracy on RewardBench while exhibiting lower bias than popular proprietary judges on the CoBBLEr benchmark.

Skywork-Critic. The Skywork-Critic-Llama-3.1 models, trained on heterogeneous open-source data including HelpSteer2 and OffsetBias (a dataset specifically targeting bias-robust evaluation training), achieve top performance on RewardBench. OffsetBias provides contrast pairs that train against categorical evaluation bias, where judges exhibit different error rates for different semantic categories.

GLIDER. Deshpande et al. (2024) developed a 3.8B judge based on Phi-3.5-mini-instruct, trained on 183 distinct evaluation metrics across 685 domains. GLIDER generates bullet-point reasoning chains and text highlights for each decision. Crucially, GLIDER outperforms GPT-4o on FLASK despite being far smaller (and is the smallest model reported to surpass GPT-4o-mini as an evaluator), establishing that evaluation-specific training breadth can compensate for substantial parameter reductions.

Self-Taught Evaluators. Meta FAIR (2024) proposed generating synthetic contrast pairs for unlabeled instructions and training a judge on these pairs without any human labels. An iterative self-improvement loop improves RewardBench score from 75.4 to 88.3 for a Llama-3 70B seed model, demonstrating that a judge's own evaluative capacity can bootstrap reliable training signal.

2025–2026: RL-trained and reasoning judges.

J1. Whitehouse et al. (2025) developed an RL-trained thinking-judge using GRPO. J1 converts both verifiable and non-verifiable judgment tasks into a unified verifiable reward format. J1-Llama-8B achieves state-of-the-art on PPE, RewardBench, and JudgeBench, outperforming models in the 70B+ range. The consistency-based reward provides additional positive signal when the judge produces the same verdict across both positional orderings of the same pair.

JudgeLRM. Chen et al. (2025) developed a family of RL-based judge models (3B–14B) demonstrating that RL training for judgment consistently outperforms SFT-initialized judges of equal size.

Table 2: Key milestones in the LLM-as-a-Judge paradigm (2022–2026). Systems are ordered chronologically within each year.

Year	System	Size	Key Contribution
2022	InstructGPT	175B	RLHF paradigm + reward model
2023	MT-Bench	N/A	>80% human agreement
2023	JudgeLM	7B–33B	Swap augmentation
2023	PandaLM	7B–70B	Auto-hyperparameter judge
2024	Prometheus 2	7B, 8×7B	Weight merging DA+PR
2024	FLAMe	Varied	5M+ human annotations
2024	GLIDER	3.8B	183 metrics; beats GPT-4o
2025	J1	8B, 32B, 70B	GRPO + consistency reward
2025	JudgeLRM	3B–14B	RL judge family

Table 3: Comparison of pointwise, pairwise, and listwise evaluation modalities.

Property	Pointwise	Pairwise	Listwise
Input	Single response	Response pair	k-tuple
Output	Score (Likert)	Preference	Ranking
Position bias	Low	High	Moderate
Transitivity	Guaranteed	Not guaranteed	By construction
Scalability	$O(N)$	$O(N^2)$	$O(N)$
Use case	Scoring, RLHF	Ranking	Leaderboard

J4R. Xu et al. (2025b) addressed discriminative precision challenges in pairwise evaluation using equivalent-initial-state group relative policy optimization.

CompassJudger-2. Zhang et al. (2025c) developed a generalist judging system via verifiable reward signals, demonstrating generalization across domains without task-specific fine-tuning.

THE FIELD HAS PROGRESSED THROUGH THREE DISTINCT TRAINING PARADIGM SHIFTS. **Judge model training evolved from distillation (2023, GPT-4 as teacher), through supervised multi-task learning on human data (2024, FLAMe and GLIDER), to reinforcement learning with structured reward signals (2025, J1 and JudgeLRM), because each successive paradigm addresses a limitation of the previous one.** Distillation transfers the teacher's biases alongside its capabilities; multi-task SFT reduces teacher dependency but cannot optimize for evaluation accuracy directly; RL with verifiable rewards enables direct optimization of the judge's output against ground truth. The pattern suggests that future advances will come from self-improving or tool-augmented judge training, where the reward signal itself is generated by external verification rather than human annotation.

A.4 Evaluation Modalities: A Structured Comparison

The LLM-as-a-Judge paradigm supports three primary evaluation modalities with distinct tradeoffs (Table 3).

PAIRWISE EVALUATION CREATES AN INHERENT TRADEOFF BETWEEN DISCRIMINATIVE POWER AND POSITION BIAS. **Pairwise comparison provides higher discriminative power than pointwise scoring because relative comparison is cognitively easier than absolute calibration, but this advantage comes at the structural cost of position bias, because the judge must process both responses in a fixed serial order.** This tradeoff explains why position bias mitigation (swap augmentation, consistency rewards) is a central concern for pairwise judge training, while pointwise judges face calibration challenges instead. The tradeoff is fundamental: any evaluation format that presents multiple items in sequence

will be susceptible to order effects, while any format that scores items independently requires calibration against an absolute scale.

The **listwise paradigm** presents all k candidate responses simultaneously and asks the judge to rank them. This produces a consistent total ordering (transitivity guaranteed by construction) and eliminates the $O(N^2)$ pairwise tournament cost. Bradley-Terry models can convert pairwise win rates into global ELO rankings. The tradeoff is context length: presenting k long responses simultaneously can exceed SLM context windows, making listwise evaluation more challenging for small models.

A.5 Cost Model and Deployment Tradeoffs

Let $f_{\mathcal{J}}^{(B)}$ denote a judge operating under token budget B, with per-token costs α (input) and β (output). For a proprietary LLM judge processing N evaluations:

$$C_{\text{total}} = N \cdot \left(\alpha \cdot B_{\text{in}} + \beta \cdot B\right) \tag{6}$$

where B_{in} is the prompt length and B is the output (reasoning + verdict) budget. At $N = 10^5$ evaluations, GPT-4 API costs can reach thousands of dollars. Verga et al. (2024) empirically document that the PoLL ensemble achieves $>7\times$ cost reduction over single-GPT-4 judging while increasing human correlation. Locally deployed SLM judges eliminate per-token API costs entirely, with cost determined solely by hardware amortization and energy consumption.

Beyond direct API cost, the proprietary judge paradigm introduces structural constraints for production deployment:

- **Privacy exposure**: Sensitive inputs must leave local infrastructure when sent to cloud-based judge APIs.
- **Version drift**: Model providers silently update judge models, breaking evaluation reproducibility.
- **Latency**: Production RLHF loops require low-latency feedback; cloud API round-trips create bottlenecks at large batch sizes.
- **Output opacity**: Proprietary judges provide no access to logits, intermediate representations, or reasoning traces, blocking calibration and mechanistic analysis.
- **Rate limits**: API rate limits cap evaluation throughput, making simultaneous evaluation of many model checkpoints impractical.

THE PROPRIETARY JUDGE BOTTLENECK DRIVES THE SLM TURN THROUGH OPERATIONAL CONSTRAINTS, NOT ACCURACY DEFICITS. **The transition from proprietary to small-model judges is driven by a convergence of operational constraints (cost, privacy, reproducibility, latency) that make proprietary judging structurally incompatible with production ML pipelines, not primarily by accuracy concerns.** GPT-4 remains a capable judge; the motivation for SLM judges is that specialized 3–8B models match proprietary accuracy on domain-specific evaluation while eliminating the structural constraints. This distinction is important because it predicts that SLM judges will be adopted even in scenarios where proprietary judges are marginally more accurate, because the operational benefits dominate.

A.6 Constitutional AI and the Self-Critique Paradigm

Bai et al. (2022) introduce Constitutional AI (CAI): a framework in which a model evaluates its own outputs against an explicit set of principles and revises responses found to violate those principles. CAI operates through a multi-round critique-revision pipeline:

1. The generator produces an initial response.
2. A *critique judge* evaluates the response against each constitutional principle.
3. A *revision judge* proposes corrections based on the critique.

4. The process repeats for K rounds.

CAI is the most direct demonstration of a model serving as its own judge. The self-critique paradigm connects to the Semantic Capacity Asymmetry Hypothesis (Li et al., 2026): the same model's evaluative capacity (lower-dimensional classification) supervises its generative capacity (higher-dimensional distribution), anticipating the finding that evaluation requires less computational capacity than generation.

A.7 LLM Judging and Scalable Oversight

A theoretically important connection exists between LLM-as-a-Judge and the *scalable oversight* problem in AI safety. Burns et al. (2023) demonstrate *weak-to-strong generalization*: when a strong pretrained model is fine-tuned on labels generated by a weak supervisor, it often *outperforms* the weak supervisor, because the strong model already possesses latent knowledge that weak supervision elicits rather than teaches. The implication is bidirectional:

1. Small models fine-tuned on large-model judgment data demonstrate the same elicitation mechanism: latent evaluative capacity is being surfaced, not instilled.
2. An SLM judge successfully overseeing a larger model's outputs provides empirical evidence of scalable oversight in the evaluation domain.

The 2025–2026 W2SG refinement techniques (trajectory trees, selective alignment, graph smoothing) further validate that weak judge supervision can elicit strong-model generalization beyond the supervisor's own capacity boundary (Appendix L).

A.8 Surveys and Related Work

Two prior surveys characterize the LLM-as-a-Judge landscape. Gu et al. (2024) provide formal definitions of evaluation dimensions (helpfulness, harmlessness, honesty, factuality), catalog bias types, and discuss mitigation. Li et al. (2025a) frame the landscape around the transition from generation to judgment. Neither survey addresses the specific considerations arising when the judge population is constrained to $\leq$14B parameters: ensemble feasibility with sub-10B models, the role of thinking-mode efficiency, representation-based judging, or cascade architectures uniquely enabled by SLMs. The SLM survey of Lu et al. (2024) covers SLM architectures, training, and deployment but does not examine the judging role. This survey fills the intersection of all three domains.

B The SLM Landscape

Having established *why* the field turned to small models, we now survey *which* models are available, detailing the families (Phi, Gemma, Qwen, LLaMA, R1-Distill) and architectural trends that make sub-14B judging feasible.

B.1 Defining "Small": The Parameter Boundary Debate

"Small Language Model" is a contextual, not absolute, designation. The community has not converged on a single definition, and the practical boundary has shifted as hardware efficiency and training methodology have improved. Three competing framings exist:

- **Hardware feasibility**: Models deployable on a single consumer GPU (24GB VRAM class, e.g., NVIDIA RTX 4090) in native or quantized (INT4/INT8) form. This places the boundary at approximately 14B parameters in native float16 and $\sim$30B parameters with aggressive quantization.
- **Edge deployment**: Models that run on battery-powered devices (smartphones, laptops, embedded systems) without cloud infrastructure. This tightens the boundary to 1B–7B parameters, with 3.8B being a practical sweet spot for the performance-power tradeoff.
- **Relative capability**: Models characterized as "small" relative to concurrent frontier systems. By this definition, the boundary shifts upward over time: what is "small" in 2026 encompasses models that were frontier-class in 2022.

For this survey, we adopt the hardware-feasibility boundary: **SLMs are models with $\leq$14B parameters deployable on a single A100/H100 or consumer-class GPU without model parallelism**. This threshold corresponds to the reach of typical single-node research computing infrastructure, making it a natural dividing line between "local" and "distributed" deployment.

Why 14B and not 7B? The 14B boundary coincides with commercially available instruction-tuned models that demonstrably match 2022-era frontier capability (GPT-3 class) at a fraction of inference cost. JudgeLM and Prometheus 2 both treat 7B as their primary small variant; GLIDER operates at 3.8B; INSPECTOR probes at 1.7B. The key empirical contribution of SLM-as-a-Judge research is showing that reliable judging does *not* require the 70B+ models initially assumed necessary.

THE "SMALL" BOUNDARY IS HARDWARE-DEFINED, NOT CAPABILITY-DEFINED, BECAUSE CAPABILITY BOUNDARIES SHIFT WITH TRAINING INNOVATION. **Defining SLMs by parameter count rather than capability is necessary because evaluation-specific training continuously pushes the capability frontier downward: a 3.8B model trained on evaluation data in 2024 surpasses the evaluation capability of a 70B general-purpose model from 2023.** If the boundary were defined by capability (e.g., "models that cannot match GPT-4 on judging"), the SLM category would progressively shrink as training improves. A hardware-feasibility definition provides a stable, deployment-relevant boundary that captures the operational advantages (cost, privacy, latency) motivating the SLM judge research program.

B.2 The Key SLM Model Families (Early 2026)

Five families account for nearly all SLM judge research. We review each in turn, emphasizing the design choices (training-data philosophy, attention pattern, dual-mode inference, distillation strategy) that bear directly on judging.

B.2.1 Microsoft Phi-4 Family

The Phi series represents the most studied example of the "data quality over data quantity" training philosophy, deliberately prioritizing *reasoning-dense synthetic data* over raw token scale.

Phi-4 (14B). Abdin et al. (2024) trained on high-quality "textbook-grade" synthetic data, curated from public web content via quality classifiers, and supplemented with SFT and DPO preference data. Phi-4 achieves STEM and mathematical reasoning scores rivaling models 3–5$\times$ larger, establishing that structured, high-quality data is a powerful substitute for massive compute.

Phi-4-mini (3.8B). Released February 2025 (Abouelenin et al., 2025), trained on 5 trillion tokens with a 128K token context window, 200K vocabulary, and Grouped-Query Attention. The training curriculum includes: (1) filtered public documents; (2) synthetic textbook-style data targeting mathematics, coding, and scientific knowledge; and (3) SFT and DPO preference data. Phi-4-mini frequently outperforms models twice its size on mathematical reasoning benchmarks. The same lineage's strong instruction following and structured-reasoning capacity is what made the earlier Phi-3.5-mini the base model for GLIDER (Deshpande et al., 2024).

Phi-4-reasoning variants. Abdin et al. (2025) trained these specifically for multi-step STEM reasoning via reinforcement learning on verifiable mathematical problems. The reasoning variant achieves state-of-the-art results for its size on competition mathematics, demonstrating that small generic models can be specialized toward deliberate stepwise evaluation.

THE PHI-4 FAMILY VALIDATES THAT TRAINING DATA COMPOSITION DOMINATES PARAMETER COUNT FOR EVALUATION TASKS. **GLIDER's exceptional judging performance on FLASK, built on the 3.8B Phi-3.5-mini, demonstrates that for tasks requiring precision and domain-specific reasoning, including evaluation, what the model is trained on matters more than how many parameters it has.** This is because evaluation (discriminative classification against criteria) requires internalized pattern-matching capabilities that can be instilled through targeted training, whereas generation (open-ended text production) requires the broader distributional knowledge that scales with parameters. The Phi-4 family's "textbook-grade" training philosophy is particularly well-aligned with evaluation: rubric-conditioned

scoring is structurally similar to grading against a reference standard, exactly the kind of structured reasoning that curated training data teaches well.

B.2.2 Google Gemma 2 and Gemma 3 Families

Google's Gemma series (Team et al., 2024a) represents the most architecturally innovative SLM family, with each generation introducing design choices that directly benefit judge deployment.

Gemma 2. Team et al. (2024b) introduced two critical architectural innovations:

- **Alternating local-global attention**: Each layer alternates between local sliding window attention (window = 4,096 tokens) and global attention (span = 8,192 tokens). Local attention handles the majority of tokens efficiently; global attention periodically integrates information from the full context. This pattern processes long evaluation inputs (question + multiple responses + rubric) more efficiently than full-attention models at equivalent parameter budgets.
- **Knowledge distillation training**: Gemma 2-9B and 2-2B were trained using probability distributions from the 27B teacher as the training signal, rather than one-hot next-token prediction. This provides richer gradients and enables smaller models to consistently outperform next-token-trained models of the same size.

Gemma 3 (March 2025). Gemma 3 adds multimodal capability while maintaining efficiency optimizations:

- **Multimodal judge capability**: Gemma 3 (4B, 12B, 27B) integrates native vision-language understanding via a SigLIP vision encoder, enabling evaluation of image-response pairs.
- **128K context window**: All Gemma 3 variants (4B+) support 128K token contexts, enabling evaluation of long responses and multi-turn conversations.
- **Multilingual coverage**: Pretrained on an expanded multilingual dataset supporting 140+ languages, using the Gemini tokenizer for improved cross-lingual token efficiency.
- **KV-cache efficiency**: Interleaved local/global attention reduces KV-cache memory requirements, enabling higher batch sizes for evaluation workloads on consumer hardware.

Note on Gemma 4 (April 2026). Shortly before submission, Google released Gemma 4 in Effective-2B, Effective-4B, 26B MoE (3.8B active), and 31B Dense configurations (Manik and Wang, 2026). Gemma 4 adds native audio input for the edge models and extends multimodal capabilities, with Apache 2.0 licensing. Full evaluation of Gemma 4 as a judge backbone is deferred to future work.

ARCHITECTURAL INNOVATIONS IN THE GEMMA FAMILY DIRECTLY ADDRESS THE CONTEXT-LENGTH BOTTLENECK FOR EVALUATION. **The alternating local-global attention pattern is particularly well-suited to evaluation tasks because judge inputs consist of a long context (question, rubric, candidate responses) followed by a short generation (verdict and rationale), and local attention efficiently processes the long input while global attention captures cross-response comparisons.** Standard full-attention models allocate equal computation to every token pair, most of which are within the same response and do not contribute to cross-response comparison. Gemma's alternating pattern concentrates global computation on the periodic cross-response integration steps that are most relevant to pairwise and listwise evaluation, while using cheaper local attention for within-response processing.

B.2.3 Alibaba Qwen 3 Family

Qwen 3 (Yang et al., 2025) introduces the most architecturally ambitious design among studied SLM families for judging applications: a unified dual-mode architecture.

Dual-mode operation. Qwen 3 supports a "thinking budget" mechanism: a per-query parameter that controls computational resources allocated to chain-of-thought reasoning before the final response. In *thinking mode*, the model generates an explicit reasoning trace before producing its answer. In *non-thinking mode*, it responds directly in a single pass. This enables the same model to serve as both a fast verdict generator and a deliberative evaluator depending on task requirements.

Distillation from flagship. Smaller Qwen 3 models (0.6B, 1.7B, 4B, 8B) are trained using distillation from the 32B and 235B-A22B flagship models. This strong-to-weak distillation transfers reasoning patterns that would typically require much larger parameter budgets, enabling even the 0.6B model to exhibit explicit chain-of-thought behavior.

Multilingual breadth. Qwen 3 supports 119 languages and dialects natively, making it the broadest-coverage multilingual SLM family with judge capability, directly relevant to the M-Prometheus multilingual direction (Appendix J).

MoE architecture (Qwen 3-235B-A22B). The largest Qwen 3 uses Mixture-of-Experts activating 22B of 235B parameters per forward pass, achieving frontier-level performance at substantially reduced inference cost. This MoE approach is increasingly relevant as "large" SLM deployments shift toward activated-parameter rather than total-parameter budgets.

Jayarao et al. (2025) study Qwen 3 judges explicitly, finding that thinking-mode operation ($\leq$4B parameters) achieves approximately 10% higher judging accuracy than non-thinking counterparts on pairwise evaluation tasks.

QWEN 3'S NATIVE DUAL-MODE DESIGN RESOLVES THE REASONING-COST TRADEOFF AT THE ARCHITECTURAL LEVEL. **Unlike models where thinking mode requires a separate fine-tuned variant or external scaffolding, Qwen 3 integrates both modes into a single set of weights, enabling per-query mode selection based on task difficulty without model switching.** This is significant for adaptive judge deployment (§6, C1) because it eliminates the need to maintain, load, and route between separate thinking and non-thinking models. A single Qwen 3 deployment can serve as a fast direct-verdict judge for simple tasks and a deliberative CoT judge for complex tasks, with the mode selection controllable via a single parameter. The $\sim$10% accuracy improvement from thinking mode (Jayarao et al., 2025) comes at a token-budget cost that is worthwhile only for tasks where the additional reasoning tokens surface genuinely new discriminative signals (I2).

B.2.4 Meta LLaMA 3 and LLaMA 4 Families

LLaMA 3 (Grattafiori et al., 2024) provides the most widely deployed SLM backbone in judge research and production. The LLaMA 3.1-8B and LLaMA 3.2-3B/1B variants are standard baselines across most judgment work due to their open weights, broad community support, and compatibility with all major inference frameworks.

Key architectural choices include standard Grouped-Query Attention with a 128K context window (LLaMA 3.1+), RoPE positional embeddings, and a vocabulary of 128K tokens. J1 (Whitehouse et al., 2025) trains judge variants on LLaMA 3 bases, with J1-Llama-8B achieving state-of-the-art among open-source 8B judges on JudgeBench and RewardBench. JudgeLRM similarly uses LLaMA 3 bases for its 8B and 14B variants.

LLaMA 4 (April 2025). Meta released LLaMA 4 (Scout and Maverick) in April 2025, using a native multimodal MoE architecture. LLaMA 4 Scout (109B total parameters with 17B activated per forward pass, 16 experts) supports up to 10 million tokens of context. Maverick (400B total, 17B active, 128 experts) was pretrained on approximately 22 trillion tokens, while Scout was pretrained on approximately 40 trillion tokens. While Scout's 17B active parameters are within the compute range relevant to SLM deployment, its 109B total parameter footprint means all weights must reside in memory for expert routing, exceeding the single-consumer-GPU constraint of strict SLM definitions. Systematic evaluation of LLaMA 4 models as judges is an open research task.

B.2.5 DeepSeek-R1-Distill Family

DeepSeek-R1 (Guo et al., 2025) introduced a 671B reasoning model trained using large-scale RL on verifiable rewards. The R1-distill models (1.5B, 7B, 14B, 32B, 70B) fine-tune Qwen2.5-Math (for the 1.5B and 7B variants) and Qwen 2.5 (for the 14B and 32B variants) (Qwen et al., 2025), alongside Llama-3.1-8B and Llama-3.3-70B-Instruct bases, on 800K high-quality reasoning traces generated by DeepSeek-R1 itself, a chain-of-thought distillation approach requiring no separate RL training.

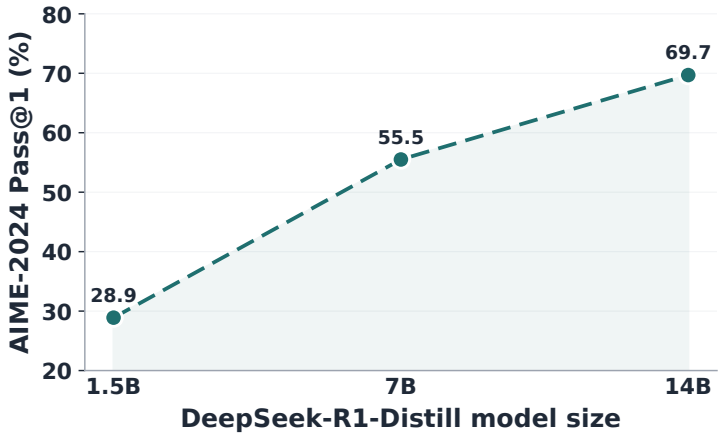


Figure 4: Distilled reasoning capacity scales with base size on AIME-2024 (Pass@1) for the DeepSeek-R1-Distill family. Because multi-step verification draws on this same capacity, the smallest distilled judges have the least headroom for reasoning-intensive evaluation.

Performance on AIME 2024 competition mathematics scales clearly with size (Figure 4): 1.5B achieves approximately 28.9% Pass@1, 7B achieves 55.5%, and 14B achieves 69.7%. These numbers are remarkable for models of their size: the 14B distill frequently outperforms non-reasoning models 4–5$\times$ larger on mathematical reasoning. For SLM judging, this demonstrates that combining architectural base capacity with high-quality reasoning-trace fine-tuning produces models capable of the multi-step verification required for evaluating complex responses. The dependence of this verification substrate on base size is precisely why the verifiability gradient (Appendix J) bends downward for the smallest judges in knowledge-intensive domains.

B.3 Architectural Trends Relevant to Judging

Thinking vs. non-thinking modes. The field is converging on hybrid inference: models choose between fast (non-thinking) and deliberate (thinking) inference modes. Qwen 3 implements this natively; DeepSeek-R1-distill models always reason; Phi-4-reasoning is dedicated to thinking mode. For judge deployment, this choice has first-order implications for the accuracy-efficiency tradeoff (Appendix D).

Mixture of Experts. MoE architecture (Qwen 3-235B-A22B, LLaMA 4, DeepSeek-V3) decouples total parameter count from per-token inference cost. A 235B MoE model activating 22B parameters per forward pass may be "small" in inference cost terms while "large" in knowledge storage terms. This complicates the parameter-count-based SLM definition but creates new opportunities: MoE judges can encode broader evaluation knowledge than their activated compute budget would suggest.

Context length expansion. 128K context windows are now standard across LLaMA 3, Phi-4-mini, and Qwen 3. This enables judges to reason over very long candidate responses, multi-turn conversations, and multi-document grounding contexts, significantly expanding the range of evaluation tasks accessible to SLM judges.

Multimodal capability. Gemma 3, LLaMA 4, and Qwen 3-VL now provide multimodal SLMs capable of evaluating image-response pairs. As AI-generated content increasingly includes images, charts, and diagrams, multimodal judge capability will become essential. The SLM judge literature has not yet systematically addressed multimodal evaluation; this represents an important open frontier.

Vocabulary size and tokenization. Phi-4-mini's 200K vocabulary and Qwen 3's multilingual tokenizer contrast with LLaMA 3's 128K vocabulary. For judge tasks involving code evaluation, mathematical notation, or multilingual content, vocabulary coverage directly affects the model's ability to represent the nuances of what it is evaluating.

Table 4: SLM family comparison relevant to judging (early 2026). Context lengths and language counts reflect the maximum available variant within each family.

Family	Sizes	Ctx	Think	Multiling	Judge Str.
Phi-4	3.8B, 14B	128K	Yes	Med	Reasoning
Gemma 3	4B–27B	128K	No	High	Multimodal
Qwen 3	0.6B–32B	128K	Yes	High	Dual-mode
LLaMA 3	1B–70B	128K	No	Med	Ecosystem
R1-Distill	1.5B–70B	128K	Always	Med	STEM

B.4 SLM Capability Characterization for Judging

Not all SLM capabilities are equally relevant to judging. We distinguish four capability dimensions:

1. **Instruction following (IFR)**: The ability to adhere to evaluation format constraints. SLMs below $\sim$1B show markedly lower IFR; above 3B, most instruction-tuned SLMs maintain $>$90% format adherence on standard rubrics.
2. **Evaluative reasoning**: The ability to assess logical consistency, factual correctness, and quality criteria. This requires understanding the content being evaluated, not just its lexical properties.
3. **Calibration**: Whether the model's expressed confidence in its verdict correlates with its actual accuracy. Uncalibrated judges produce misleading confidence signals that distort downstream aggregation. Calibration assessment for SLM judges is largely an open research problem.
4. **Knowledge depth**: For domain-specific evaluation tasks, the judge must possess sufficient domain knowledge to assess content correctness. This dimension is irreducible by training: an SLM without the relevant domain knowledge cannot accurately evaluate domain-specific responses regardless of judge training.

THE SLM CAPABILITY THRESHOLD FOR RELIABLE JUDGING IS 3B, NOT 7B, BECAUSE IFR IS THE BINDING CONSTRAINT. **The field's initial assumption that reliable judging requires 7B+ parameters has been refuted by three converging findings: GLIDER (3.8B) surpasses GPT-4o, Phi-4-mini (3.8B) anchors state-of-the-art judge systems, and Qwen3-4B in thinking mode achieves $\sim$10% higher accuracy than non-thinking counterparts.** The operative boundary is not raw parameter count but instruction following reliability (IFR): below approximately 3B parameters, models struggle to consistently produce verdicts in the required format, which makes evaluation harnesses unreliable. Above 3B, with evaluation-specific training, judge quality scales primarily with training data quality rather than model size. Table 4 summarizes the key families and their judge-relevant properties, and Figure 5 visualizes their relative strengths across the capability dimensions most relevant to judging. No single family dominates every dimension: Qwen 3 and Gemma 3 lead on multilingual breadth, Phi-4 and R1-Distill on reasoning density, and LLaMA 3 on ecosystem maturity. This complementarity is exactly what makes cross-family panels (Appendix E) effective.

MOE ARCHITECTURES DECOUPLE THE KNOWLEDGE-COST TRADEOFF FOR JUDGE DEPLOYMENT. **Mixture-of-Experts models store more evaluation-relevant knowledge than their per-token compute cost would suggest, because different experts can specialize in different evaluation domains while sharing the routing and attention infrastructure.** A Qwen 3-235B-A22B model activates only 22B parameters per forward pass but has access to 235B parameters of stored knowledge, enabling it to evaluate across a broader range of domains than a 22B dense model. For judge deployment, this means MoE models can approach the knowledge depth of large models while maintaining the latency characteristics of small models. The practical implication is that the parameter-count boundary ($\leq$14B) may need refinement to account for activated vs. total parameters as MoE architectures become standard.

B.5 Inference Infrastructure for SLM Judge Deployment

Practical deployment of SLM judges at scale requires infrastructure beyond raw model capability.

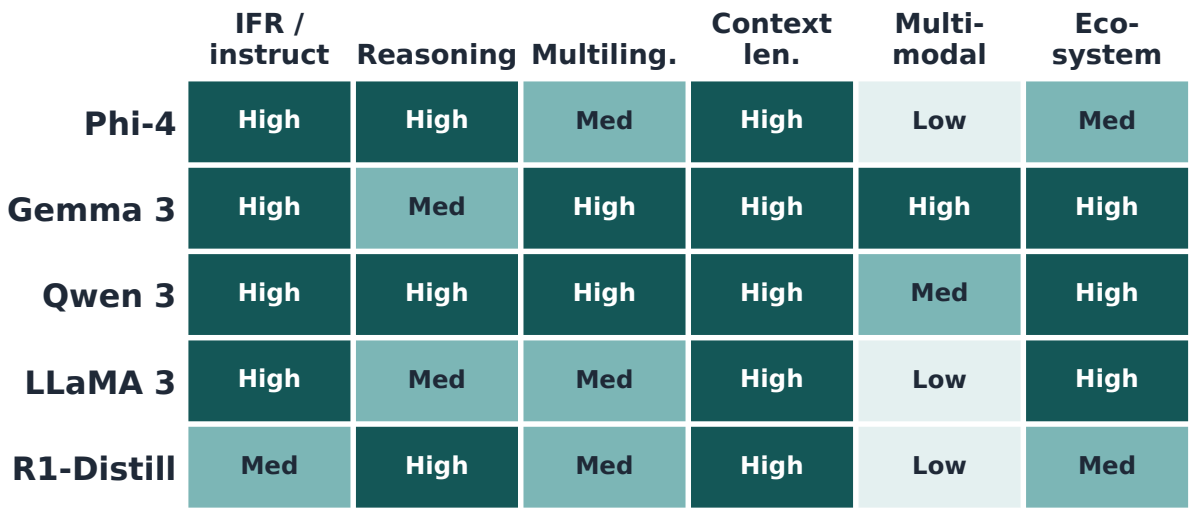


Figure 5: Relative strengths of the major SLM families across judge-relevant capability dimensions. Darker cells indicate stronger capability. Ratings are derived from published benchmark results: Reasoning from MMLU-Pro/MATH, Code from HumanEval/LiveCodeBench, Multilingual from M-RewardBench/MGSM, Instruction Following from IFEval, and the respective model technical reports. The absence of a uniformly dominant family motivates heterogeneous panels.

vLLM. Kwon et al. (2023) developed PagedAttention for high-throughput batched decoding with dynamic KV cache management. vLLM is the de facto standard for SLM judge inference in research evaluation pipelines, supporting tensor parallelism, speculative decoding, and model quantization. It achieves 2–4$\times$ higher GPU utilization than naive inference.

llama.cpp / Ollama. Pure CPU and consumer GPU inference, enabling fully local SLM judge deployment on unmodified consumer hardware. Supports GGUF-format quantized models (INT4, INT8) fitting a 14B model within 8GB VRAM.

Hugging Face TGI. Production-grade serving supporting continuous batching, multi-GPU tensor parallelism, and custom tokenization. Standard for enterprise deployments with SLA requirements.

Quantization. Post-training quantization (INT4 via GPTQ or AWQ) reduces memory requirements by $\sim$4$\times$, making 14B models deployable on a single 16GB consumer GPU. Accuracy degradation from INT4 quantization is typically $<$2% on standard benchmarks. For 7B models, INT4 quantization enables deployment on a single 8GB VRAM GPU.

B.6 The Emerging SLM Judge Ecosystem

The period from 2023 to 2026 has seen rapid consolidation of a dedicated SLM judge ecosystem distinct from general-purpose SLM deployment:

- **Open judge model hubs**: Prometheus-Eval repository, GLIDER on Hugging Face, J1 checkpoints, all providing ready-to-deploy judge models with documented training recipes and license terms.
- **Evaluation harnesses**: lm-evaluation-harness adapted for judge evaluation, enabling reproducible meta-evaluation across systems.
- **Judgment-specific benchmarks**: JudgeBench, JudgeBoard, and RewardBench provide standardized evaluation targets (Appendix I).
- **Training datasets**: Feedback Collection, Preference Collection (Prometheus), SFT judge datasets (JudgeLM), RL judge preference datasets (J1), OffsetBias, and HelpSteer2.
- **Tool-augmented reward modeling**: An emerging trend toward agentic judge systems that invoke external tools (code executors, search engines, document retrievers) for fact-grounded, verifiable assessment.

Table 5: Master comparison of SLM and SLM-relevant judge systems surveyed in this paper, grouped by training paradigm. **Mode**: DA = direct assessment, PR = pairwise ranking, CR = critique, ORM = outcome reward model, PRM = process reward model. **Insight** links each system to the field-wide insight (I1–I5) it most directly supports. Headline results are reported by the cited source under that source's protocol; cross-row comparison is therefore indicative, not controlled (see Limitations).

System	Base model	Size	Mode	Domain	Headline result	Insight
Off-the-shelf prompting						
Qwen3-Judge (Jayarao et al., 2025)	Qwen 3	0.6–4B	PR	General	∼10% over non-thinking mode	I2
Supervised fine-tuning (distillation / human / rubric)						
PandaLM (Wang et al., 2024c)	LLaMA	7B	PR	General	Auto hyperparameter selection	I1
JudgeLM (Zhu et al., 2025)	Vicuna	7–33B	DA/PR	General	>90% agreement with GPT-4	I1
Auto-J (Li et al., 2024b)	LLaMA-2	13B	DA/PR/CR	58 scenarios	Strong cross-scenario transfer	I1
Prometheus 2 (Kim et al., 2024b)	Mistral	7B, 8×7B	DA+PR	Rubric	GPT-4-level corr. (MT-Bench/FLASK)	I1
FLAMe (Vu et al., 2024)	PaLM 2	Varied	ORM	100+ tasks	87.8% RewardBench; low bias	I1
GLIDER (Deshpande et al., 2024)	Phi-3.5-mini	3.8B	DA	685 domains	Beats GPT-4o on FLASK (far smaller)	I1
Selene Mini (Alexandru et al., 2025)	LLaMA-3.1	8B	DA/PR	General (11 OOD)	Beats GPT-4o-mini; top 8B on RewardBench	I1
M-Prometheus (Pombal et al., 2025)	Qwen2.5	3–14B	DA/PR	Multilingual (20+)	SOTA open on M-RewardBench	I5
Self-supervised / preference optimization						
Self-Taught Eval. (Wang et al., 2024b)	LLaMA-3	70B seed	PR	General	RewardBench 75.4 → 88.3, no labels	I1
CompassJudger-2 (Zhang et al., 2025c)	Qwen	7B	DA/PR	Generalist	Matches DeepSeek-V3 on JudgerBenchV2	I1
Reinforcement learning (verifiable rewards)						
J1 (Whitehouse et al., 2025)	LLaMA-3.1/3.3, Qwen3	8B, 32B, 70B	DA/PR	Verifiable+open	SOTA 8B on JudgeBench/RewardBench	I1
JudgeLRM (Chen et al., 2025)	LLaMA / Qwen	3–14B	DA	Reasoning	3B > GPT-4; 7B > DeepSeek-R1 (+2.79 F1)	I1
J4R (Xu et al., 2025b)	Qwen2.5	7B	PR	Reasoning	7B judge surpasses GPT-4o	I1
GenPRM (Zhao et al., 2026)	R1-Distill	1.5–32B	PRM	Math steps	7B gen. PRM > 72B scalar PRM	I1
StepWiser (Xiong et al., 2025)	Qwen2.5	1.5B	PRM	Reasoning steps	Stepwise generative judge via RL	I1
TIR-Judge (Zhang et al., 2025b)	SLM+tools	3–8B	DA/PR	Code/verifiable	Tool-integrated reasoning judge	I5
Representation-based (decoding-free)						
INSPECTOR (Li et al., 2026)	frozen SLM	1.7B	DA (probe)	Structured	Beats prompted SLM at equal size	I1
LAGER	frozen SLM	Small	DA (probe)	Scoring	Cross-layer logit aggregation gain	I1

This ecosystem makes the development of new SLM judges substantially more accessible than even two years prior. The combination of large context windows (128K+), thinking-mode capability, and open training datasets means that a research lab with a single high-end GPU can now train and deploy a competitive SLM judge in days, a capability that required frontier-model API access as recently as 2023.

C Training Methodologies for SLM Judges

A capable base model is necessary but not sufficient. This section expands the training paradigms of §3, showing how the supervision signal, not parameter count, determines judge quality.

C.1 A Consolidated Map of SLM Judge Systems

Before turning to individual training paradigms, Table 5 consolidates the major SLM and SLM-relevant judge systems surveyed across this paper into a single reference, organized by training paradigm and annotated with base model, parameter scale, evaluation mode, target domain, the headline empirical result, and the survey insight (I1–I5) each system most directly supports. The table shows three field-level patterns. First, a clear progression down the rows: from distillation (2023) through human-supervised and rubric SFT (2024) to RL with verifiable rewards and representation probing (2025–2026), tracing the decreasing-supervision gradient developed in this appendix. Second, the *scale compression*: headline results at or above proprietary parity are achieved by 3–8B models throughout, with no monotone improvement from the larger entries (I1). Third, the *domain concentration*: the strongest verifiable-domain results (math, code) cluster among RL and tool-augmented systems, while subjective-domain coverage remains sparse (I5). The remainder of this appendix expands each row's training methodology in turn.

C.2 Overview of Training Paradigms

The capability of a language model to serve as a reliable judge depends critically on how it was trained, not just what data it saw, but what training *objective* it was optimized for. Table 6 summarizes the five major training paradigms for SLM judges.

A critical insight from the 2025–2026 literature is that the bottleneck for high-performance SLM judges is not the *volume* of training data but the *quality and structure* of the supervision signal.

Table 6: Overview of judge training paradigms. Compute cost is relative to off-the-shelf deployment. Human data indicates whether human annotation is required.

Paradigm	Signal	Cost	Human	Systems
Off-shelf	Prompt only	None	None	Qwen3, Phi-4
SFT (dist.)	Teacher verd.	Med	Indirect	JudgeLM
SFT (rubric)	Human+syn.	Med	Rubrics	Prometheus 2
SFT (self)	Self-gen.	Low	None	Self-Taught
DPO	Pref. pairs	Low	Pref.	CompassJ-2
RL (GRPO)	Verif. rew.	High	None	J1, JudgeLRM
Repr. probe	Probe labels	V.Low	Small set	INSPECTOR

C.3 Off-the-Shelf Judging: Baselines and Limits

The most accessible deployment strategy uses general-purpose instruction-tuned SLMs as zero-shot or few-shot judges. The model receives a structured evaluation prompt and produces a verdict via standard inference.

Why this sometimes works. Instruction-following models trained on diverse human preference data internalize approximate quality signals across many tasks. A well-prompted 7B model can reliably detect obvious factual errors, gross formatting violations, or blatant hallucinations. Jayarao et al. (2025) find that thinking-mode SLMs (0.6B–4B, Qwen 3 family) achieve approximately 10% higher judging accuracy in pairwise evaluation than non-thinking counterparts, because explicit chain-of-thought traces encode evaluation criteria before issuing a verdict. This is the "baseline ceiling" for off-the-shelf judging.

Why this fails for precision evaluation. Off-the-shelf models were not optimized for the discriminative precision that expert evaluation demands. Key failure modes:

- **Low Instruction Following Rate (IFR)**: The model may produce freeform prose instead of the required verdict format. For models below 3B, IFR can drop below 80% on structured rubrics.
- **Unmitigated position bias**: Without explicit training to counter position bias, models default to primacy or recency heuristics.
- **Calibration failure**: High expressed confidence for incorrect verdicts; no correlation between certainty and accuracy.
- **Task-specific deterioration**: Performance on code evaluation, mathematical reasoning, and multilingual tasks is substantially lower than on general text quality.
- **Adversarial vulnerability**: Off-the-shelf judges are susceptible to meaningless phrases or specific punctuation sequences that trigger high reward scores regardless of response quality.

OFF-THE-SHELF JUDGES FAIL ON PRECISION TASKS BECAUSE PRETRAINING OPTIMIZES FOR GENERATION, NOT DISCRIMINATION. **General-purpose SLMs can detect gross quality differences (obvious errors, blatant hallucinations) but fail on subtle discriminations (partially correct reasoning, nuanced rubric criteria) because their pretraining objective (next-token prediction) rewards fluent continuation, not quality classification against criteria.** The gap between off-the-shelf and trained judges is largest on tasks requiring rubric-conditioned reasoning, where the judge must simultaneously parse evaluation criteria, map response features to rubric dimensions, and produce a calibrated score. This multi-step discriminative operation is absent from standard pretraining curricula. The $\sim$10% accuracy improvement from thinking mode (Jayarao et al., 2025) partially compensates by forcing explicit criterion enumeration, but it cannot substitute for evaluation-specific training.

C.4 Supervised Fine-Tuning for Judging

SFT on judgment-specific datasets is the most widely studied training paradigm and the foundation of nearly all dedicated open-source judge models.

C.4.1 Data Construction Strategies

Distillation from proprietary judges. The dominant approach annotates evaluation data using a teacher model (GPT-4, GPT-4o) and fine-tunes a smaller student to replicate the teacher's verdict and reasoning. JudgeLM (Zhu et al., 2025) assembles task seeds, model responses, and GPT-4 judgments and trains models at 7B, 13B, and 33B scales. PandaLM (Wang et al., 2024c) uses GPT-3.5-generated verdicts with human-crafted contexts. The effective assumption: the teacher's judgment quality is bottlenecked only by its cost, and distillation can transfer the teacher's evaluative capacity at one-time training cost.

Human-annotated preference data. FLAMe (Vu et al., 2024) avoids teacher-LLM distillation entirely, training on >5M human annotations across >100 quality assessment tasks drawn exclusively from permissively licensed datasets. Human annotation provides more reliable training labels and eliminates the risk of propagating teacher-model biases. The tradeoff is substantial annotation infrastructure; FLAMe achieves 87.8% on RewardBench while exhibiting lower CoBBLEr bias than proprietary judges.

Rubric-conditioned evaluation data. Prometheus (Kim et al., 2024a) introduces datasets where each evaluation instance includes a task, candidate response, human-written rubric specifying evaluation criteria, and a reference answer. The model learns to condition its verdict on the rubric rather than producing generic assessments. The Feedback Collection (pointwise) and Preference Collection (pairwise), released alongside Prometheus 2, have become standard resources for judge fine-tuning.

Diversity-driven data. GLIDER (Deshpande et al., 2024) scales the data dimension differently: rather than increasing label quality, it trains across 183 distinct evaluation metrics and 685 domains, developing broad coverage over the evaluation space. Auto-J (Li et al., 2024b) achieves a similar effect by training across 58 diverse scenario types.

Bias-targeted data: OffsetBias. OffsetBias specifically targets the failure mode where judges exhibit different error rates for different semantic categories of response. OffsetBias provides contrast pairs chosen to train against systematic favoritism toward particular response types (e.g., overly verbose responses, responses with confident-sounding hedges). Skywork-Critic models trained on OffsetBias demonstrate substantially reduced bias across multiple evaluation categories.

TRAINING DATA DIVERSITY GOVERNS CROSS-TASK GENERALIZATION, NOT TRAINING DATA VOLUME. **The most important lesson from comparing GLIDER, FLAMe, and Prometheus 2 is that the number of distinct evaluation dimensions covered during training, not the total number of training examples, determines whether a judge generalizes to unseen evaluation tasks.** GLIDER's 183-metric coverage is what enables its surprising cross-task transfer to FLASK, a benchmark not in its training distribution. FLAMe's 100-task human-annotation breadth achieves the same generalization through human label diversity. Both outperform single-task distillation approaches that use orders of magnitude more training examples but cover fewer evaluation dimensions. The mechanism is coverage of the evaluation space: each new evaluation dimension during training teaches the model a new discriminative pattern, whereas additional examples from the same dimension provide diminishing returns after the pattern is learned.

Table 7 consolidates the public training resources that underpin these data-construction strategies, which together form the practical starting point for any new SLM judge.

C.4.2 Anti-Bias Training Techniques

Swap augmentation. JudgeLM (Zhu et al., 2025) evaluates both orderings of response pairs and includes both in the training set, forcing the model to reach the same verdict regardless of positional order. This directly penalizes position-dependent behavior during training at the data level.

Reference support and reference drop. JudgeLM alternates between providing a gold reference answer and withholding it during training. This dual-mode training produces a judge that can operate reference-free (generalizable to open-ended tasks) while maintaining higher accuracy when references are available.

Table 7: Public training and meta-evaluation resources for SLM judges. "Mode" indicates direct assessment (DA), pairwise (PR), or critique (CR); "Label" indicates the supervision source.

Resource	Mode	Label	Used by
Feedback Collection	DA	Synthetic	Prometheus
Preference Collection	PR	Synthetic	Prometheus 2
JudgeLM data	DA/PR	GPT-4	JudgeLM
FLAMe (100+ tasks)	DA/PR	Human	FLAMe
HelpSteer2	DA	Human	Skywork-Critic
OffsetBias	PR	Contrast	Skywork-Critic
Self-Taught pairs	PR	Self-gen.	Self-Taught
J1 synthetic (∼22K)	PR	Verifiable	J1

Synthetic contrast pairs. Self-Taught Evaluators (Wang et al., 2024b) generate contrasting response pairs for unlabeled instructions: one high-quality response and one subtly worse variant. Training on these pairs without any human labels improves RewardBench from 75.4 to 88.3 for a Llama-3 70B seed model. The key mechanism: the bootstrapping loop (train judge, use judge to generate better training data, retrain judge) allows the model's own evaluative capacity to self-correct, eliminating the teacher-bias propagation risk of distillation approaches.

Dynamic difficulty curricula. A 2025 trend in alignment training replaces static training sets with dynamic curricula that adapt the training distribution to the model's current capability frontier. The curriculum progressively introduces harder evaluation pairs as the model demonstrates competence on simpler ones, preventing early overfitting to easy evaluation patterns.

C.4.3 Weight Merging for Multi-Mode Judges

Prometheus 2 (Kim et al., 2024b) trains two separate models for pointwise direct assessment (DA) and pairwise ranking (PR), then merges their weights. The authors experiment with multiple merging strategies (Task Arithmetic, TIES, DARE-TIES, DARE-Linear, SLERP) and find that DARE-Linear achieves the best performance. DARE-Linear first sparsifies the delta parameters (differences between fine-tuned and base weights) by randomly dropping a proportion and rescaling the remainder, then merges linearly:

$$\theta_{\text{merged}} = \theta_{\text{base}} + \alpha \cdot \hat{\Delta}_{\text{DA}} + (1 - \alpha) \cdot \hat{\Delta}_{\text{PR}} \tag{7}$$

where $\hat{\Delta}$ denotes the sparsified, rescaled delta vectors. Experiments show the merged model outperforms both single-task variants, suggesting that evaluation "skills" from the two modes are complementary in parameter space and do not exhibit catastrophic interference.

WEIGHT MERGING SUCCEEDS BECAUSE POINTWISE AND PAIRWISE EVALUATION ACTIVATE COMPLEMENTARY SUBSPACES. **The success of Prometheus 2's weight merging indicates that pointwise and pairwise evaluation skills are not in competition for the same parameters; instead, they develop in orthogonal regions of the weight space, allowing lossless combination.** This is a stronger claim than "merging works": it implies that the two evaluation modalities require different internal circuitry. Pointwise scoring requires absolute calibration against internalized quality standards; pairwise comparison requires relative feature matching between two responses. These computations naturally engage different attention patterns and projection weights, explaining why merging produces a model better than either constituent. The practical implication is that modality-specific fine-tuning followed by weight merging may be more efficient than joint multi-task training, which forces both modalities to share parameters that they would otherwise specialize.

C.5 Reinforcement Learning for Judge Training

The third major paradigm applies RL directly to the judging objective, optimizing evaluation accuracy rather than imitation of a teacher's outputs.

C.5.1 Direct Preference Optimization (DPO)

Rafailov et al. (2023) reformulate the traditional PPO-RLHF pipeline as a direct classification objective on preference pairs, eliminating the need for a separate reward model. Given prompt x, preferred response y_w, and rejected response y_l, DPO minimizes:

$$\mathcal{L}_{\text{DPO}}(\theta) = -\mathbb{E}\left[\log \sigma\left(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)\right] \tag{8}$$

where π_θ is the policy being trained, π_{ref} is a frozen reference policy, and β controls the KL penalty. CompassJudger-2 (Zhang et al., 2025c) trains toward generalist judging using verifiable DPO-style rewards. DPO's key advantage is stability: no reward model instability, no PPO hyperparameter sensitivity, and no critic network memory overhead.

C.5.2 Group Relative Policy Optimization (GRPO)

GRPO was introduced in DeepSeek-R1 (Guo et al., 2025) as a computationally efficient alternative to PPO. For each input prompt, GRPO samples a group of outputs $\{o_1, \ldots, o_G\}$, computes rewards $\{r_1, \ldots, r_G\}$, and normalizes within the group:

$$A_i = \frac{r_i - \mu_{\text{group}}}{\sigma_{\text{group}}} \tag{9}$$

The group-relative normalization eliminates the need for a learned value function, substantially reducing memory and compute vs. PPO. The policy gradient loss is:

$$\mathcal{L}_{\text{GRPO}} = -\mathbb{E}_i\left[\min(\rho_i A_i,\ \text{clip}(\rho_i, 1\pm\epsilon)A_i) - \beta D_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}})\right] \tag{10}$$

where $\rho_i = \pi_\theta(o_i|q)/\pi_{\text{ref}}(o_i|q)$ is the probability ratio.

GRPO for judge training. J1 (Whitehouse et al., 2025) applies GRPO with verifiable rewards to train thinking-capable judge models. The key innovation is a unified reward formulation:

- For **verifiable tasks** (math, code): Rule-based correctness reward.
- For **non-verifiable tasks**: Synthetic preference pairs provide pseudo-verifiable rewards.
- **Consistency reward**: Additional reward when the judge produces the same verdict across both positional orderings, directly penalizing position bias during RL training.

J1-Llama-8B and J1-Qwen-32B achieve state-of-the-art on JudgeBench and RewardBench, with J1-Qwen-32B outperforming o1-mini on several subtasks while trained on only ∼22K synthetic preference examples.

C.5.3 J4R: Equivalent Initial State Optimization

J4R (Xu et al., 2025b) addresses a subtle flaw in standard pairwise RL training: when positive and negative samples are drawn from different response spaces, the model may learn shortcuts based on response provenance rather than genuine quality discrimination. J4R uses equivalent initial state group relative policy optimization, enforcing that both preferred and rejected responses emerge from the same input distribution. This is particularly important for small SLM judges, whose limited capacity makes them more susceptible to shortcut learning.

RL Training Outperforms SFT For Judging Because It Optimizes The Evaluation Objective Directly. **SFT optimizes for imitation of a teacher's outputs, bounding the student's evaluative behavior by the teacher's; RL with verifiable rewards optimizes directly against judgment accuracy, allowing the student to discover evaluation strategies that the teacher never demonstrated.** J1-Llama-8B trained on ∼22K synthetic examples outperforms 70B SFT judges on JudgeBench, demonstrating that a well-designed RL objective provides vastly more useful training signal per example than

teacher-distilled verdicts. The mechanism is that SFT treats each verdict as a token-prediction target, whereas RL treats the entire verdict as a binary outcome (correct/incorrect) and adjusts the full generation distribution to maximize correctness. This outcome-level optimization avoids inheriting token-level biases from the teacher's reasoning style.

J1'S CONSISTENCY REWARD IS THE FIRST TRAINING-LEVEL SOLUTION TO POSITION BIAS. **Prior position bias mitigations (swap augmentation, inference-time averaging) operate at the data or inference level; J1's consistency reward operates at the training objective level, providing gradient signal that directly penalizes the model's internal representations for producing position-dependent verdicts.** The distinction matters because data-level mitigations (swap augmentation) teach the model that both orderings should produce the same verdict, but they do so through example memorization. Objective-level mitigation (consistency reward) teaches the model that position-dependent behavior reduces reward regardless of the specific evaluation pair, producing a more general and robust position-invariance property.

C.6 Reward Model Training: ORM vs. PRM

A related family of judge models are *reward models*: discriminative models trained to score candidate responses, used primarily in RLHF pipelines.

C.6.1 Outcome Reward Models (ORM)

ORMs evaluate only the final response, producing a scalar reward $r \in \mathbb{R}$:

$$\text{ORM} : (x, y) \to r \tag{11}$$

Training requires preference pairs (y_w, y_l); the model learns to assign $r(y_w) > r(y_l)$. Key properties:

- **Low annotation cost**: Labels can be obtained automatically (checking math answer values, running code tests).
- **Process-agnostic**: Cannot identify which reasoning step caused a failure.
- **Reward hackable**: A model can reach a correct final answer through flawed reasoning and receive full ORM reward.

RewardBench (Lambert et al., 2025) is the primary benchmark for ORM evaluation.

C.6.2 Process Reward Models (PRM)

PRMs assign step-level rewards to each reasoning step (Lightman et al., 2024):

$$\text{PRM} : (x, [s_1, s_2, \ldots, s_T]) \to [r_1, r_2, \ldots, r_T] \tag{12}$$

where s_t is the t-th reasoning step and $r_t \in \{-1, 0, +1\}$ indicates step correctness. PRMs provide:

- **Error localization**: Precisely identifies which step failed, enabling targeted correction.
- **Search guidance**: Can prune incorrect reasoning paths in Best-of-N sampling early.
- **Alignment quality**: Incentivizes correct reasoning processes, not just lucky correct answers.

The primary cost is annotation: step-level labels require human annotators to validate each step independently. Automated process supervision, using Monte Carlo rollouts to generate step-level labels automatically, reduces this bottleneck. A recent shift replaces scalar PRMs with *generative* process judges: GenPRM (Zhao et al., 2026) reasons explicitly about each step with code verification, and StepWiser (Xiong et al., 2025) trains stepwise generative judges via RL. Both report that small generative PRMs can surpass much larger scalar PRMs under test-time scaling, reinforcing I1 (targeted training over scale) for step-level evaluation specifically.

PRMS ARE PARTICULARLY VALUABLE FOR SLM JUDGES BECAUSE THEY COMPENSATE FOR LIMITED PARAMETRIC KNOWLEDGE. **SLMs lack the broad domain knowledge of frontier models, making them more prone to the "lucky correct answer" failure mode that ORMs cannot detect;**

PRM-style step-level evaluation catches flawed reasoning even when the final answer is correct, providing a verification mechanism that compensates for the SLM's knowledge limitations. This insight connects to the verifiability gradient (I5): in high-verifiability domains (math, code), step-level verification provides the strongest SLM judge signal because each step can be checked against external criteria. In low-verifiability domains, step-level evaluation degrades because the judge lacks the domain knowledge to assess intermediate reasoning quality. Appendix J discusses domain-specific implications.

C.7 Knowledge Distillation for Judges

Knowledge distillation from a large, capable judge into a small, deployable one is the standard approach when ground-truth annotation is infeasible.

Reasoning trace distillation. DeepSeek-R1 (Guo et al., 2025) demonstrates that fine-tuning a small base on high-quality reasoning traces from a large teacher transfers chain-of-thought patterns without additional RL. Applied to judging: a 7B model fine-tuned on 800K judgment traces acquires multi-step evaluative reasoning capability difficult to obtain from SFT on verdict labels alone.

Capacity ceiling. Distilled judges cannot surpass the teacher's evaluative capacity. If the teacher cannot discriminate subtle quality differences, the distilled student inherits this limitation. This motivates combining distillation with RL: use distillation to bootstrap initial capability, then refine with task-specific RL rewards.

Preference leakage risk. When the distillation teacher (e.g., GPT-4) is also used to generate training data for the models being evaluated, the judge may systematically favor stylistically similar outputs, a manifestation of preference leakage (Li et al., 2025b). Training judges on human-annotated data (FLAMe) or self-generated contrasting pairs (Self-Taught Evaluators) avoids this family-lineage contamination.

DISTILLATION CREATES A TEACHER-BIAS PROPAGATION RISK THAT RL TRAINING AVOIDS. **When a student judge is distilled from a teacher, it inherits not only the teacher's evaluation capability but also the teacher's systematic biases (position preference, verbosity preference, self-preference), because the training objective cannot distinguish between the teacher's correct judgments and the teacher's biased ones.** This is the fundamental limitation of distillation-based judge training: the student's error profile is a subset of the teacher's error profile. RL with verifiable rewards breaks this coupling because the reward signal comes from ground truth (was the verdict correct?), not from the teacher's judgment. Self-Taught Evaluators (Wang et al., 2024b) partially mitigate this by using the model's own iteratively improved judgments, but they remain bounded by the model's own biases unless external verification is introduced.

C.8 Representation-Based Judging: Beyond Generation

The paradigms above all produce a verdict by generating tokens. A final line of work questions that assumption entirely, extracting the verdict from a model's internal states without any generation. Two systems, INSPECTOR and LAGER, anchor this direction.

C.8.1 The INSPECTOR Framework

The most fundamental rethinking of judge training shifts from generative to representation-based evaluation. Li et al. (2026) introduce INSPECTOR (INternal Signal Probing and EvaluaTion Of Representations), which probes hidden states of a frozen small LM to predict evaluation scores directly.

The **Semantic Capacity Asymmetry Hypothesis** states: the computation required to evaluate response quality is substantially smaller than the computation required to generate that response. Evaluation is a classification problem defined over the model's world representation; generation is a sequential decoding problem over the full output distribution.

INSPECTOR operates as:

1. Forward-pass a (question, candidate response) pair through a frozen small LM ($\leq$1.7B), collecting hidden states $\{h_1^{(l)}, \ldots, h_T^{(l)}\}$ at layer l.

2. Pool hidden states across relevant token positions.
3. Apply a lightweight linear probe $\psi_a : \mathbb{R}^d \to \mathbb{R}$ per evaluation aspect a.
4. Report aspect-level scores $\{s_a = \psi_a(h_{\text{pool}})\}$ as the verdict.

Experiments on FLASK, GSM8K (Cobbe et al., 2021), and GPQA demonstrate that INSPECTOR outperforms prompting-based SLM judges at equivalent parameter budgets.

C.8.2 LAGER: Cross-Layer Logit Aggregation

LAGER replaces single-layer final logit scoring with cross-layer aggregation:

$$\hat{s} = \sum_{\text{layers } l \in L} w_l \sum_{d \in \mathcal{D}} d \cdot P_l(\text{token} = d) \tag{13}$$

where $\mathcal{D}$ is the set of digit tokens (1–5) and w_l are learnable layer-weighting parameters. Mid-layer representations carry more semantically stable evaluative signals than the final layer. LAGER achieves significant improvements in Spearman correlation on FLASK and HelpSteer compared to standard top-layer scoring, without any re-training.

MID-LAYER REPRESENTATIONS CARRY STRONGER EVALUATIVE SIGNALS THAN FINAL-LAYER REPRESENTATIONS. **LAGER's finding that mid-layer aggregation outperforms final-layer scoring implies that evaluative information is encoded earlier in the transformer forward pass and is partially overwritten by the generation-oriented computation in later layers.** This is consistent with the mechanistic interpretability literature showing that middle layers encode semantic content while final layers encode output distribution. For evaluation, the semantic representation ("is this response logically sound?") is the relevant computation; the output distribution ("what token should come next?") adds noise. Probing mid-layers extracts the evaluative signal before it is contaminated by generation-oriented processing, explaining why representation-based evaluation can match generative evaluation at a fraction of the compute.

C.8.3 Implications: Probing as Judge Training

INSPECTOR and LAGER collectively shift the question from "how to train a better SLM judge" toward "how to better extract the evaluative capacity that small models already encode." Two consequences:

1. **Probe training is far cheaper than full fine-tuning**: A lightweight linear probe trained on a small labeled set ($<$1,000 examples) from a frozen model reduces training cost to near-negligible.
2. **Frozen SLMs are viable judges for structured tasks**: Rather than fine-tuning a new judge per domain, domain-specific probes can be trained atop a frozen general-purpose SLM, enabling rapid judge adaptation to new evaluation settings.

C.9 Summary: Training Paradigm Comparison

Table 8 provides a comprehensive comparison of all training paradigms along key deployment dimensions, and Figure 6 positions them on the training-cost versus bias-mitigation plane. The plot makes the central trade-off visible: RL with verifiable rewards achieves the strongest bias mitigation but at the highest training cost, while representation probing reaches comparable bias resistance at almost no training cost, and off-the-shelf prompting sits alone in the low-cost, low-robustness corner.

THE TRAINING PARADIGM PROGRESSION FROM DISTILLATION TO RL TO PROBING FOLLOWS A DECREASING-SUPERVISION GRADIENT. **Each successive training paradigm requires less external supervision: distillation needs a capable teacher, SFT needs annotated data, RL needs only a reward signal, and probing needs only a small calibration set from a frozen model (Figure 7).** This gradient has a direct practical implication: as the field moves toward less-supervised paradigms, the barrier to entry for training new SLM judges decreases. A researcher with a single GPU can now train a competitive judge via probing (INSPECTOR) or RL on synthetic data (J1), without access to proprietary teacher models or large-scale human annotation infrastructure. The decreasing supervision requirement also implies decreasing risk of bias propagation: probing from a frozen model cannot inherit the teacher's evaluation biases because there is no teacher in the loop.

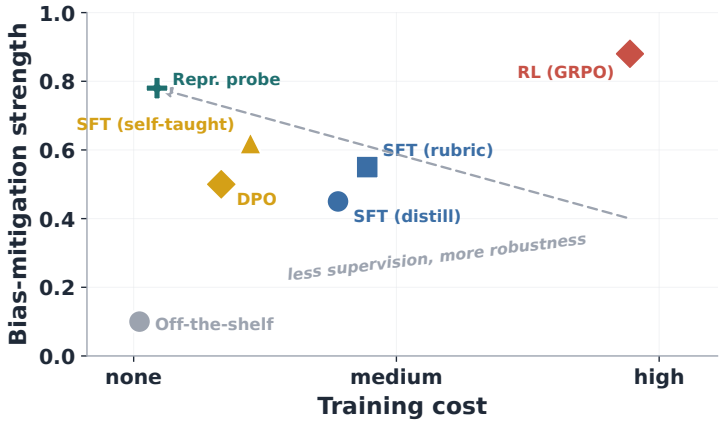


Figure 6: Judge training paradigms positioned by training cost and bias-mitigation strength. Marker shape encodes the supervision source. The dashed guide marks the direction of decreasing supervision and rising robustness, from distillation toward RL and probing.

Table 8: Comparison of SLM judge training paradigms. Bias mitigation indicates the primary mechanism for reducing systematic evaluation biases during training.

Paradigm	Data	Cost	Bias Mit.	Systems
Off-shelf	None	None	None	Qwen3, Phi-4
SFT (dist.)	Teacher	Med	Swap aug.	JudgeLM
SFT (rubric)	Hum.+syn.	Med	Ref. drop	Prometheus 2
SFT (self)	Self-gen.	Low	Iterative	Self-Taught
DPO	Pref. pairs	Low	Implicit	CompassJ-2
RL (GRPO)	Verif. rew.	High	Consist. r.	J1, JudgeLRM
Repr. probe	Probe lab.	V.Low	N/A	INSPECTOR

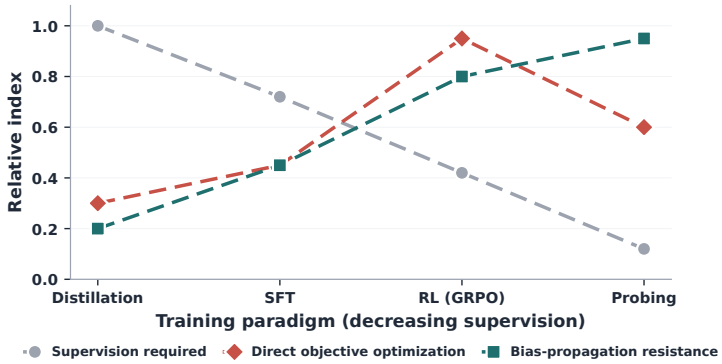


Figure 7: The training-paradigm progression follows a decreasing-supervision gradient. Moving from distillation to SFT to RL to probing, the external supervision required falls while direct optimization of the evaluation objective and resistance to teacher-bias propagation rise. Indices encode the qualitative ranking argued in this appendix.

D Individual Judging: Token Budget and Scale

With training fixed, we turn to inference-time decisions: how many reasoning tokens to spend, when thinking mode helps, and how scale interacts with capability. This is the full account behind §3 and Figure 2.

D.1 The Individual Judge: Scope and Design Space

An individual SLM judge is a single language model deployed to evaluate candidate responses without auxiliary models, ensemble voting, or inter-agent communication. It operates according to Eq. 3 and Eq. 4 from Appendix A, consuming a (question, response, optional rubric) tuple and producing a verdict. The

appeal is simplicity: a single forward pass, minimal infrastructure, zero communication overhead. The challenge is that all evaluative responsibility falls on one model, and all biases propagate unchecked to the verdict.

The design space has four primary axes:

1. **Training paradigm**: How the model acquired its judging capability (Appendix C).
2. **Token budget**: How many reasoning tokens to allocate per evaluation.
3. **Inference mode**: Thinking vs. non-thinking; chain-of-thought vs. direct verdict.
4. **Architectural scale**: How parameter count interacts with judging capability given a fixed training budget.

This appendix examines axes 2–4 in depth, with particular focus on the non-obvious finding that "more reasoning" does not monotonically improve verdict quality.

D.2 Token Budget Analysis for SLM Judges

The single highest-leverage operational decision for an individual judge is how many tokens to spend per evaluation. We decompose that budget, then derive the scaling laws that govern its optimal setting.

D.2.1 Components of the Total Token Budget

For an individual judge evaluating a single (question, response) pair, the total token budget B comprises:

$$B = B_{\text{sys}} + B_{\text{ctx}} + B_{\text{rationale}} + B_{\text{verdict}} \tag{14}$$

where B_{sys} is the system prompt (evaluation instructions, rubric), B_{ctx} is the context (question + candidate response), $B_{\text{rationale}}$ is the optional chain-of-thought reasoning the judge generates before its verdict, and B_{verdict} is the final verdict token(s). For typical judge deployments:

- $B_{\text{sys}} \approx$ 200–800 tokens (rubric complexity determines this)
- $B_{\text{ctx}} \approx$ 500–3,000 tokens (varies with response length and task)
- $B_{\text{rationale}} \approx$ 0–2,000 tokens (0 for verdict-only, up to full CoT)
- $B_{\text{verdict}} \approx$ 1–50 tokens (score digit(s) or pairwise label)

The operational cost per evaluation scales with B. For $N = 10^5$ evaluations during a training run, $B_{\text{rationale}}$ becomes the dominant cost driver, making its optimization the highest-leverage operational decision in SLM judge deployment.

D.2.2 Optimal CoT Scaling Laws: The Sweet Spot

A foundational finding of 2025 reasoning research, with direct implications for judging, is that reasoning performance follows a non-monotonic curve as a function of chain-of-thought length:

1. **Below the sweet spot**: The model under-reasons, missing evaluation criteria that additional analysis would have caught.
2. **At the sweet spot**: Reasoning tokens provide genuine evaluative signal; each step either confirms or revises the current assessment.
3. **Above the sweet spot**: The model overthinks, generating redundant elaborations that introduce error cascades and entropy accumulation.

Critically, the optimal CoT length follows two scaling laws:

- **Inversely with model capability**: A more powerful judge reaches a correct verdict faster; its sweet spot is at a shorter CoT length than a less capable model evaluating the same task.

- **Directly with task complexity**: Harder evaluation tasks warrant more reasoning tokens before the degradation curve begins.

The practical implication: there is no universally optimal $B_{\text{rationale}}$. Each (model, task-difficulty) pair has its own sweet spot. For a 3.8B judge on standard evaluation tasks, the sweet spot is approximately 300–500 tokens; for a 14B judge on hard reasoning evaluation, optimal CoT extends to 1,000–1,500 tokens.

THE SWEET SPOT SCALES INVERSELY WITH JUDGE CAPABILITY BECAUSE STRONGER MODELS ENCODE MORE DISCRIMINATIVE SIGNAL PER TOKEN. **Stronger judges extract more evaluation-relevant information per reasoning token because their internal representations already encode richer discriminative features, requiring fewer explicit reasoning steps to surface the verdict-relevant signal.** A 14B model's internal representation of "logical consistency" is denser than a 3.8B model's representation; consequently, fewer reasoning tokens are needed to map this representation to a verdict. Weaker models must generate more reasoning tokens to compensate for sparser internal representations, but this compensation has a ceiling: beyond the sweet spot, additional tokens introduce noise that overwhelms the weak discriminative signal. This mechanistic explanation connects the sweet-spot finding to the main paper's I1 (judging is discriminative) and I2 (reasoning tokens help only when surfacing new signals).

D.2.3 Budget-Aware Reasoning Allocation

Han et al. (2025) introduce token-budget-aware LLM reasoning: models are prompted with an explicit token budget constraint derived from estimated task complexity. Key findings:

- Tasks within a model's clear competency can be evaluated accurately with zero rationale tokens.
- For tasks near the competency boundary, rationale generation helps, but only up to approximately 200–400 tokens.
- For tasks beyond the model's competency, extended reasoning actively harms accuracy.

Wang et al. (2024a) provide a complementary cost-performance frontier analysis: different reasoning strategies (zero-shot direct, zero-shot CoT, few-shot CoT, self-consistency) occupy distinct positions on the token budget vs. accuracy curve. The optimal strategy is task-dependent.

Techniques for adaptive token allocation include:

- **TALE / IBPO**: Token-budget-aware training methods that teach models to dynamically adjust reasoning effort based on perceived query difficulty.
- **Length-filtered voting**: When generating multiple candidate verdicts, filter out those with reasoning chains deviating from the established optimal range.
- **SelfBudgeter**: A lightweight classifier that predicts optimal CoT length before generation begins.

BUDGET-AWARE JUDGING REDUCES COST WITHOUT ACCURACY LOSS BECAUSE WITHIN-COMPETENCY RATIONALES ARE POST-HOC, NOT COMPUTATIONALLY PRIOR. **For tasks well within a model's competency, the verdict is effectively determined in the forward pass before any reasoning tokens are generated; the subsequent rationale is post-hoc justification, not genuinely prior reasoning.** Explicitly constraining the budget forces the model to commit to its top-layer representation, avoiding the waste of extended text generation for an already-decided verdict. This mechanism explains why budget-aware prompting reduces evaluation cost by 40–60% without accuracy degradation on within-competency tasks: the removed tokens were never contributing to the verdict in the first place.

D.2.4 Self-Consistency for Individual Judges

Self-consistency (Wang et al., 2022) runs the same judge multiple times with temperature $T > 0$ and takes the majority verdict. For judges:

- **Variance reduction**: Reduces random generation noise without increasing model size.

- **Pseudo-ensemble**: A single model producing diverse reasoning paths weakly approximates an ensemble.
- **Cost**: Scales linearly with the number of samples k; typically $k = 3$–5 is optimal.

SELF-CONSISTENCY REDUCES VARIANCE BUT CANNOT REDUCE BIAS, MAKING IT CATEGORICALLY DIFFERENT FROM CROSS-FAMILY ENSEMBLES. **Self-consistency is most effective when individual verdict samples are high-variance but independently unbiased; it is least effective when the judge has a systematic bias, because all samples share the same directional error and majority voting reinforces the bias rather than canceling it.** This is the critical distinction from cross-family ensemble methods (Appendix E), which decorrelate systematic biases because different model families encode different systematic errors. A practitioner should use self-consistency when the primary concern is generation noise (stochastic variation between runs) and cross-family ensembles when the primary concern is systematic bias (consistent directional error). Using self-consistency to address systematic bias will waste compute without improving accuracy.

D.3 Thinking Modes in SLM Judges

How a judge spends its token budget matters as much as how large the budget is. We now compare explicit thinking modes against direct verdicts, covering the empirical gains, the architectural mechanisms that enable them, and the always-thinking alternative.

D.3.1 Thinking vs. Non-Thinking: Empirical Evidence

Jayarao et al. (2025) evaluate Qwen 3 models (0.6B to 4B) in thinking and non-thinking modes on pairwise judge tasks. Key findings:

- Thinking mode achieves approximately **10% higher accuracy** than non-thinking mode at equal parameter count.
- The accuracy gain is concentrated in the "Hard" split of RewardBench (reasoning/STEM/safety), where deliberation is most warranted.
- Thinking mode introduces a **3–5× token cost multiplier**: a 500-token non-thinking evaluation becomes 1,500–2,500 tokens in thinking mode.
- The gain-to-cost ratio is most favorable at the smallest model sizes: a 0.6B model in thinking mode outperforms a 1.7B model in non-thinking mode on the hard split.

THINKING MODE PROVIDES THE LARGEST RELATIVE GAIN AT THE SMALLEST MODEL SIZES BECAUSE IT COMPENSATES FOR SPARSER INTERNAL REPRESENTATIONS. **A 0.6B model in thinking mode outperforming a 1.7B model in non-thinking mode indicates that explicit reasoning token generation compensates for the smaller model's less developed internal representations, effectively "externalizing" the discriminative computation that larger models perform internally.** The gain is largest at small scales because the gap between internal representation quality and task requirements is widest; for larger models, internal representations are already sufficient for most evaluation tasks, so thinking mode adds less. This implies a practical deployment rule: thinking mode is most cost-effective for the smallest deployable judge size, and its value diminishes as the base model grows.

D.3.2 The Thinking Budget Mechanism (Qwen 3)

Qwen 3 implements a thinking budget as a per-query inference parameter B_{think}:

- $B_{\text{think}} = 0$: Non-thinking mode (fast inference, direct verdict).
- $B_{\text{think}} > 0$: Thinking mode with the specified ceiling; the model self-regulates, terminating early on simple tasks.

For judge deployment, this enables difficulty-adaptive evaluation: route simple evaluations to $B_{\text{think}} = 0$ and complex evaluations to $B_{\text{think}} = 1{,}000$–$2{,}000$. A lightweight classifier can estimate task difficulty from rubric complexity and response length, routing accordingly. Adaptive routing can achieve $>7\times$ cost reduction vs. always-thinking deployment while maintaining near-equivalent accuracy on the hard split.

D.3.3 Always-Thinking Models: DeepSeek-R1-Distill

DeepSeek-R1-Distill models always generate reasoning traces enclosed in `<think>...</think>` tags, with no non-thinking mode. This design is advantageous for high-complexity evaluation tasks (verifiable STEM) where deliberation provides genuine uplift. For open-ended evaluation (writing quality, helpfulness), the always-thinking mode often produces long, redundant reasoning chains with no accuracy benefit over non-thinking SLMs.

This tradeoff motivates a hybrid deployment strategy: use DeepSeek-R1-Distill judges for structured, verifiable evaluation domains, and non-thinking Qwen 3 or Phi-4-mini for open-ended, preference-based evaluation.

D.3.4 Reasoning Without Thinking

A distinct approach (Ma et al., 2025) directs the model to evaluate the response without generating an explicit reasoning trace, relying on internalized evaluation heuristics. The model's "reasoning" happens implicitly in its internal forward pass rather than in generated tokens. This is most effective for tasks clearly within the model's competency and eliminates the entropy accumulation risk of extended generation entirely.

D.4 Scale-Capability Relationship for Individual Judges

Token budget and inference mode interact with a third axis: raw parameter count. This subsection establishes where scale does and does not matter for judging, bounding the regime from below (a capability floor) and above (a knowledge ceiling).

D.4.1 The Core Finding: Targeted Training Dominates Scale

The most consistent finding across the individual SLM judge literature is that training composition and evaluation-specific optimization dominate raw parameter count. Three converging lines of evidence:

1. **GLIDER vs. GPT-4o**: GLIDER (3.8B, Phi-3.5-mini base) outperforms GPT-4o on FLASK despite being far smaller (Deshpande et al., 2024).
2. **Prometheus 2 (7B) vs. GPT-4**: Prometheus 2 (7B Mistral) achieves GPT-4-level correlation on MT-Bench, VicunaBench, and FLASK (Kim et al., 2024b).
3. **JudgeLM (7B) vs. GPT-4**: JudgeLM-7B achieves >90% agreement with GPT-4 on pairwise evaluation (Zhu et al., 2025).

TARGETED TRAINING DOMINATES SCALE BECAUSE JUDGING REQUIRES DISCRIMINATIVE PRECISION, NOT GENERATIVE BREADTH. **Scale amplifies generative capability because generation requires broad knowledge coverage across the full output vocabulary; judgment requires only detecting quality-relevant features (logical consistency, factual correctness, instruction adherence), a classification boundary rather than a generation distribution.** A model trained on many classification-relevant examples develops more precise discriminative boundaries than a larger untrained model, regardless of parameter count. The GLIDER result is not an exception; it is the expected consequence of this mechanism. The operative boundary for judge quality is training data composition, not parameter count, precisely because evaluation operates in a lower-dimensional feature space than generation.

D.4.2 The Capability Threshold: Below 1B

Below approximately 1B parameters, instruction-following capability degrades such that evaluation format compliance becomes unreliable:

- Models <1B in non-thinking mode: IFR drops below 80% on complex rubrics.
- Models <1B in thinking mode (Qwen 3-0.6B): IFR improves to ∼90%, but evaluative reasoning quality is substantially below larger models.
- INSPECTOR (Li et al., 2026) circumvents this threshold via representation probing: a 1.7B frozen model's hidden states contain enough evaluative signal for lightweight probe classification without any generation.

D.4.3 The Capability Ceiling: When Size Does Matter

For certain evaluation tasks, SLMs genuinely cannot match frontier models regardless of training because the evaluation requires domain knowledge that the SLM does not possess:

- **Graduate-level STEM**: Evaluating a response to a graduate-level physics problem requires first-order physics knowledge that smaller models may lack.
- **Nuanced safety evaluation**: Detecting subtle dual-use framing or manipulation patterns requires broad world knowledge.
- **Complex multi-document grounding**: Evaluating factual groundedness for long-form responses referencing multiple documents taxes context utilization in ways that scale differently than evaluation reasoning.

THE CAPABILITY CEILING IS KNOWLEDGE-BOUNDED, NOT REASONING-BOUNDED, BECAUSE EVALUATION-SPECIFIC TRAINING CAN INSTILL REASONING PATTERNS BUT NOT DOMAIN KNOWLEDGE. **SLMs can be trained to apply evaluation criteria, follow rubrics, and produce calibrated scores, but they cannot be trained to know things they were never exposed to during pretraining.** This distinction is critical for deployment decisions: an SLM judge will fail on graduate-level STEM evaluation not because it cannot reason about evaluation, but because it does not know the relevant physics. Cascaded routing (§6, C1) addresses this by escalating knowledge-intensive evaluations to larger models, while retaining SLMs for the majority of evaluations where knowledge depth is not the bottleneck.

D.5 The Overthinking Phenomenon in SLM Judges

The sweet-spot analysis above implies that extra reasoning can hurt. We now examine that degradation directly, gathering the multi-paper evidence, the mechanism behind it, and the practical mitigations.

D.5.1 Definition and Multi-Paper Evidence

The overthinking phenomenon refers to the empirically documented observation that extended reasoning traces can degrade, rather than improve, judgment quality for tasks within or near a model's competency ceiling.

Sui et al. (2025) document overthinking across general reasoning tasks. Key mechanisms:

- **Redundant elaboration**: Reasoning steps beyond which no new information is introduced.
- **Self-contradiction**: Extended generation produces statements contradicting earlier correct reasoning.
- **Hallucinated justification**: Novel "evidence" generated mid-chain that was not present in the evaluation context.
- **Entropy accumulation**: Each generation step introduces probabilistic noise; sufficiently long chains accumulate enough noise to flip the verdict.

Shojaee et al. (2026) find that reasoning models exhibit an *illusion of thinking*: extended reasoning traces that look deliberate but do not improve, and sometimes degrade, performance. An Apple evaluation study shows a "complexity cliff" pattern: models maintain high accuracy up to a specific task complexity threshold, then fail precipitously.

Ghosal et al. (2026) extend this to test-time scaling: the "mirage of test-time compute" shows that more computation does not reliably produce better results when the judge model lacks the underlying capability.

Srivastava et al. (2026a) provide task-specific evidence: reasoning models overthink basic arithmetic by generating multi-step derivations for operations they reliably handle in a single step, introducing errors in the process.

D.5.2 Why Overthinking Happens: A Mechanistic View

The mechanistic explanation connects to autoregressive generation. In each step, the model samples from $p(t_i|t_1, \ldots, t_{i-1}, c)$ where c is the evaluation context. Three properties interplay:

1. **Context contamination**: Each generated token updates the effective conditioning context. A subtly incorrect early token propagates or amplifies through subsequent tokens (error cascading).
2. **Fluency pressure**: When genuine reasoning steps are exhausted, fluency pressure motivates plausible-but-ungrounded elaborations.
3. **Recency dilution**: In long chains, verdict-relevant information at the beginning is diluted by the accumulated conditioning context, reducing its effective influence on the terminal verdict.

OVERTHINKING DEGRADES SLM JUDGES MORE SEVERELY THAN LLM JUDGES BECAUSE SMALLER MODELS HAVE WEAKER ERROR-CORRECTION CAPACITY. **When a small model generates an incorrect intermediate reasoning token, it has less capacity to "recover" in subsequent tokens because its internal error-detection circuitry is less developed, causing error cascades to propagate more readily than in larger models.** A 70B model can often recover from a mid-chain reasoning error because its richer internal representations detect the inconsistency and self-correct in the next step. A 3.8B model is more likely to accept the erroneous intermediate conclusion and build subsequent reasoning on top of it. This asymmetry explains why the overthinking degradation curve is steeper for smaller models: each additional reasoning token carries higher marginal risk because the per-token error-correction capacity is lower.

D.5.3 OptimalThinkingBench and the Sweet Spot

Aggarwal et al. (2025) introduce OptimalThinkingBench, measuring both over-thinking and under-thinking simultaneously. Key findings for judging:

- There is a sweet spot token budget for each (task-type, model) pair; both below and above degrade accuracy.
- The sweet spot scales directly with task difficulty: harder evaluation tasks warrant more reasoning tokens.
- Smaller models have smaller sweet spots: a 3.8B judge saturates at approximately 300–500 tokens; a 14B model may benefit from up to 1,000–1,500 tokens.
- The sweet spot for evaluation tasks is systematically lower than for generation tasks of equivalent apparent difficulty.

D.5.4 Practical Mitigations

Because overthinking is driven by token-level error accumulation rather than a single root cause, the effective mitigations all share one goal: prevent the judge from generating reasoning tokens beyond the point where the verdict is already determined. The following techniques, ordered from simplest to most involved, operationalize this goal.

1. **Explicit budget constraints**: "Evaluate the response in at most 3 reasoning steps." This directly limits the entropy accumulation window.
2. **Task-difficulty routing**: Classify evaluation tasks by estimated difficulty. Simple tasks go to direct-verdict judges; complex tasks go to thinking-mode with calibrated budgets.
3. **First-step anchoring**: Prompt the judge to state its verdict first, then elaborate. This prevents fluency pressure from overwriting an early correct verdict.
4. **Stopping criteria**: Monitor the log-probability of the verdict token at each step. Stop when this probability is high and stable.

5. **Length-filtered voting**: When generating k verdict samples, discard those with reasoning chains substantially longer than the estimated sweet spot.

6. **Reasoning without Thinking** (Ma et al., 2025): Direct the judge to evaluate without explicit traces for within-competency tasks.

D.6 Instruction Following Rate (IFR) Analysis

A critical but often understated metric for individual SLM judge quality is the Instruction Following Rate (IFR): the fraction of evaluations for which the judge produces a verdict in the expected format. An IFR < 1.0 creates silent failures: evaluation harness code expecting a score digit receives freeform text and silently drops or misparses the verdict.

Factors affecting IFR.

- **Rubric complexity**: Longer rubrics increase the probability that the model prioritizes rubric reasoning over format compliance.
- **Model scale**: Sub-3B models have IFR $\sim$75–85% on complex multi-criterion rubrics; 7B+ models maintain $>$95% IFR.
- **Thinking mode**: Thinking mode can lower IFR because the model sometimes terminates the thinking trace in a format the evaluation harness misinterprets.
- **Format-reinforced training**: JudgeLM and GLIDER both explicitly train on verdict formatting, achieving near-100% IFR even at smaller scales.

IFR Failures Are Non-Random And Concentrated In Hard Evaluation Cases, Creating Systematic Benchmark Contamination. **Evaluation pipelines that handle malformed outputs by imputing a default score introduce systematic bias because IFR failures are concentrated in the hardest evaluation cases, where the model struggles most to structure a verdict, so imputed defaults create a specific downward bias for precisely the cases where accurate measurement matters most.** Proper IFR reporting and missing-data handling is required for valid benchmark comparisons. Papers reporting judge accuracy without reporting IFR may substantially overstate performance on easy cases while hiding failures on hard cases.

D.7 Cascade and Routing Architectures

The findings above motivate spending compute adaptively rather than uniformly. Cascade and routing architectures do exactly this, reserving expensive judges for the cases that need them. We describe three variants of increasing sophistication.

D.7.1 Basic Cascading

The most cost-effective deployment architecture combines multiple individual judges in a cascade:

1. Route all evaluations to a small SLM judge (3.8B–7B).
2. For evaluations where the small judge expresses high confidence: accept verdict directly.
3. For evaluations below a confidence threshold: escalate to a larger judge (14B–70B).
4. Reserve frontier models for only the highest-complexity evaluations.

The escalation criterion is the judge's logit probability for its verdict token, a natural confidence proxy, though calibration work is needed (Appendix L).

Table 9: Deployment strategies for individual SLM judges. Relative cost is normalized to direct verdict (no CoT) as 1×. Best use case indicates the scenario where each strategy provides the optimal cost-accuracy tradeoff.

Strategy	Cost	Latency	Best For
Direct verdict	1×	Lowest	Simple tasks
CoT at sweet spot	2–5×	Low	Moderate difficulty
Thinking mode	5–15×	Med	Hard/STEM eval
Self-consist. (k=3)	3×	Low-Med	Variance reduct.
Cascade	1.2–2×	Variable	Production RLHF
Speculative	1.1–1.5×	Low-Med	High-throughput

D.7.2 Speculative Judging

Analogous to speculative decoding in generation, speculative judging uses a small judge to propose a verdict and a larger judge to verify it. The large judge only generates a full evaluation if it disagrees with the small judge's proposal. This reduces large-judge usage to disagreement cases, precisely the hard cases where additional capacity provides genuine uplift.

SPECULATIVE JUDGING EXPLOITS THE SAME COMPUTE-ASYMMETRY AS SPECULATIVE DECODING: MOST EVALUATIONS ARE EASY. **The effectiveness of speculative judging rests on the empirical observation that the majority of evaluation instances (typically 60–80%) fall well within a small judge's competency, and the large judge merely confirms the small judge's verdict on these instances.** The computational savings come from avoiding full large-model inference on the easy majority. The approach fails when the distribution of evaluation difficulty shifts toward hard cases (e.g., JudgeBench-hard), where the small judge disagrees frequently and the cascade provides no cost savings. This means speculative judging is most efficient for production RLHF pipelines (where most evaluations are routine) and least efficient for meta-evaluation benchmarks (where tasks are intentionally difficult).

D.7.3 Confidence-Calibrated Routing

A more principled routing approach uses trained confidence calibrators:

1. Train a lightweight calibrator on held-out (judge output, true verdict) pairs.
2. At inference, use calibrated confidence to decide routing.
3. Set the threshold to achieve a target accuracy SLA.

Confidence-calibrated routing corrects for the judge's systematic calibration errors, particularly important for SLM judges, which tend to be overconfident in precisely the domains where they are least accurate.

D.8 Deployment Strategy Summary

Table 9 summarizes the cost-accuracy tradeoffs across deployment strategies for individual SLM judges; Figure 3 (main paper) plots the same strategies on the cost-quality plane. The frontier is clearly concave: cascade and speculative judging sit near the knee, capturing most of the accuracy of expensive thinking-mode inference at a small fraction of its cost, while self-consistency lies below the frontier because it reduces variance without adding new discriminative signal.

E Ensemble and Jury Methods

A single judge, however well trained, has fixed biases. This section develops the ensemble principle of §4: pooling architecturally diverse SLMs to decorrelate errors at a fraction of frontier cost.

E.1 Motivation: Why Aggregate Multiple Judges?

Single-judge evaluation, even from a high-capability model, carries structural risks: systematic bias toward certain response styles, susceptibility to position priming, and a single point of failure when the judge encounters tasks at the boundary of its competency. The ensemble paradigm addresses these risks by distributing evaluative responsibility across multiple independent judges, whose errors, if sufficiently uncorrelated, are suppressed through aggregation.

The theoretical basis is the *Condorcet Jury Theorem* (CJT): if each juror independently makes the correct decision with probability $p > 0.5$, the probability of the majority decision being correct approaches 1 as jury size grows:

$$P(\text{majority correct}) = \sum_{k>n/2}^{n} \binom{n}{k} p^k (1-p)^{n-k} \tag{15}$$

CJT's independence assumption is the critical caveat: if jurors are correlated (e.g., all from the same model family), ensemble gains are minimal. The central empirical insight of SLM jury research is that **architectural diversity** between jury members is the primary determinant of ensemble quality.

PANEL DIVERSITY OUTPERFORMS SINGLE-JUDGE SCALE BECAUSE ERROR DECORRELATION DOMINATES INDIVIDUAL ACCURACY. **A heterogeneous ensemble of diverse smaller models consistently outperforms a single large judge on human-correlation metrics because error decorrelation provides a stronger accuracy signal than individual judge quality, up to the point where individual accuracy falls below $p = 0.5$.** Each model family has systematically different failure modes arising from distinct pretraining corpora, architecture choices, and RLHF alignment. When panel members come from disjoint families, their judgment errors are uncorrelated, and majority voting becomes a near-optimal noise reducer. Homogeneous ensembles (same model, different temperatures) share identical systematic biases and provide minimal ensemble benefit beyond variance reduction. The practical threshold is that each panel member must individually exceed 50% accuracy on the evaluation task; below this, CJT guarantees that ensemble accuracy *decreases* with panel size.

E.2 PoLL: Panel of LLM Evaluators

Verga et al. (2024) is the foundational paper establishing the panel paradigm in LLM evaluation. PoLL replaces the single GPT-4 judge with a heterogeneous panel of three diverse models: Command R, GPT-3.5-Turbo, and Claude Haiku.

E.2.1 Methodology

The PoLL pipeline:

1. Each panel member M_i independently evaluates the same (question, candidate response) pair, with identical evaluation prompts but no visibility into other members' verdicts.
2. Each member produces an independent verdict $v_i \in \mathcal{V}$.
3. A deterministic aggregation function $f_{\text{agg}} : \mathcal{V}^n \to \mathcal{V}$ combines the verdicts:
   - **Max voting**: For pairwise comparisons, select the verdict chosen by the most panel members.
   - **Average pooling**: For pointwise scoring, average the individual scores.

Independence is enforced by design: panel members never share their reasoning traces or intermediate assessments. This prevents inter-agent sycophancy (Appendix G).

E.2.2 Key Results

Across single-hop QA, multi-hop QA, and Chatbot Arena datasets:

- **Human correlation**: PoLL achieves higher Spearman correlation with human judgments than single GPT-4 judging on all three dataset types.
- **Cost**: PoLL is $>7\times$ cheaper than GPT-4 judging per evaluation, because Command R and Claude Haiku have substantially lower per-token costs.
- **Bias reduction**: Position-consistent accuracy is higher for PoLL than for GPT-4 alone, suggesting that diverse panel perspectives partially de-correlate position-dependent errors.
- **Stability**: The standard deviation of PoLL's verdicts across prompt variations is lower than that of GPT-4, indicating greater robustness to prompt sensitivity.

E.2.3 The Intra-Model Bias Problem

A subtle but important finding in PoLL is that single LLM judges exhibit *intra-model bias*: models systematically prefer outputs stylistically similar to their own pretraining distribution. A GPT-4 judge, for instance, may systematically favor GPT-4-generated responses over Llama-generated ones that are equally correct.

CROSS-FAMILY PANELS CANCEL INTRA-MODEL BIAS BECAUSE DIFFERENT FAMILIES ENCODE DIFFERENT STYLISTIC PRIORS. **A Llama-family judge and a Claude-family judge bring distinct stylistic priors, and their directional biases partially cancel in aggregation because the bias vectors point in different directions in preference space.** A single-family panel (e.g., three GPT-variants) does not achieve this cancellation because all members share the same stylistic prior. This insight explains why PoLL's heterogeneous panel achieves higher human correlation than even the strongest individual panel member: the ensemble is not merely averaging individual accuracies, it is actively reducing a bias component that no single judge can self-correct.

E.3 COSMosFL: Cross-Family SLM Ensembles

Cho et al. (2025) demonstrate ensemble gains specifically for small language models in the code fault localization domain, providing the most direct evidence that the panel principle generalizes to SLM judges below the 10B parameter boundary.

E.3.1 The Cross-Family Orthogonality Principle

COSMosFL's central finding is that different SLM families exhibit *orthogonal fault localization patterns*: Qwen, Phi, and Gemma models each excel at localizing different categories of faults. This orthogonality, arising from differences in pretraining data composition, architecture, and fine-tuning procedures, makes cross-family ensembles particularly effective. Formally, if model M_i has error rate ϵ_i on fault category c and models are orthogonal, then the ensemble error rate approaches $\prod_i \epsilon_i^{(c)}$, which decreases multiplicatively with panel size.

CROSS-FAMILY ERROR ORTHOGONALITY IS STRONGER FOR SLMS THAN FOR LLMS BECAUSE SLMS HAVE MORE SPECIALIZED KNOWLEDGE DISTRIBUTIONS. **Smaller models are trained on more curated, domain-specific data, making their failure patterns more distinct from each other than those of large models trained on near-identical web-scale corpora.** A Phi-4-mini model trained on "textbook-grade" synthetic data and a Qwen 3-4B model trained on multilingual web data develop genuinely different internal representations of code correctness, producing orthogonal error patterns. Two 70B models trained on overlapping web-scale data share more of their knowledge distribution and consequently more of their failure modes. This counterintuitive implication, that error orthogonality increases as model size decreases, is precisely why SLM ensembles provide proportionally larger gains over individual members than LLM ensembles.

E.3.2 Aggregation Strategies

COSMosFL evaluates two aggregation strategies on the Defects4J benchmark:

- **Equal-weight task-level voting**: Each SLM contributes equally via majority vote. Already achieves substantial improvement over individual SLM baselines.
- **Weight-optimized voting**: A Differential Evolution (DE) algorithm optimizes the weight assigned to each model based on held-out performance. Weight-optimized ensembles achieve Pareto-optimality: superior accuracy and lower operational cost compared to any individual SLM.

E.4 JudgeBoard and the MAJ Framework

Bi et al. (2026) introduce JudgeBoard, a benchmark and multi-agent judging framework for SLM evaluation of reasoning-heavy tasks.

E.4.1 MAJ: Multi-Agent Judging

MAJ (Multi-Agent Judging) implements structured sequential aggregation: multiple SLMs with distinct reasoning profiles evaluate a candidate response, and a lightweight aggregator synthesizes their verdicts with awareness of inter-judge disagreement patterns. Key design features:

- **Distinct reasoning profiles**: Panel members are selected to cover different reasoning strengths (formal reasoning, commonsense, mathematical), with each judge assigned to aspects within its strength profile.
- **Disagreement-aware aggregation**: When panel members disagree, the aggregator weights the verdict of the judge whose profile is most aligned with the evaluation task type, rather than simple majority voting.
- **Elo-based leaderboard**: JudgeBoard constructs ongoing Elo ratings for all evaluated models, providing a stable relative ranking more robust to individual evaluation noise.

E.4.2 Results on MATH

On the MATH benchmark, MAJ with SLMs matches or exceeds standalone LLM-judge accuracy, despite using only models in the 3B–14B range. The key mechanism: mathematical evaluation requires checking the correctness of the final symbolic answer (a verifiable step) followed by evaluating reasoning coherence (a qualitative step). MAJ assigns the former to a judge with strong mathematical formalism and the latter to a judge with stronger reasoning coherence, achieving better accuracy than any individual judge on either dimension alone.

DISAGREEMENT-AWARE AGGREGATION OUTPERFORMS MAJORITY VOTING BECAUSE IT EXPLOITS JUDGE SPECIALIZATION. **When panel members have different competency profiles, majority voting treats all votes equally and can be outvoted by judges operating outside their competency; disagreement-aware aggregation weights each vote by the judge's estimated competence on the specific evaluation dimension, preserving the specialist's signal.** This is particularly important for SLM panels where individual members have narrow competency ranges: a 4B math-specialist judge should receive higher weight on mathematical correctness than a 4B writing-specialist judge, even though both participate in the same panel. MAJ operationalizes this through profile-aligned weighting, while COSMosFL achieves a similar effect through DE-optimized weights.

E.5 Self-Consistency as Single-Model Ensemble

Self-consistency (Wang et al., 2022) extends from the generation domain to the judging domain. Instead of sampling diverse reasoning paths for answer generation, it samples diverse evaluation rationales for a fixed (question, response) pair, then takes the majority verdict:

$$v^* = \arg\max_{v \in \mathcal{V}} \sum_{i=1}^{k} \mathbb{1}[v_i = v], \quad v_i \sim f_{\mathcal{J}}(\cdot | x, T{>}0) \tag{16}$$

For SLM judges, self-consistency at temperature $T \in \{0.5, 0.7\}$ with $k = 3$–5 samples typically improves verdict accuracy by 3–7% over greedy decoding, particularly on contested near-boundary evaluations.

SELF-CONSISTENCY AND CROSS-FAMILY ENSEMBLES ADDRESS DIFFERENT ERROR COMPONENTS AND SHOULD NOT BE CONFLATED. **Self-consistency reduces stochastic variance (different samples from the same distribution) while cross-family ensembles reduce systematic bias (different models from different distributions); using one as a substitute for the other addresses the wrong error component.** A practitioner whose primary concern is that the judge produces different verdicts on repeated runs should use self-consistency. A practitioner whose primary concern is that the judge systematically favors certain response styles should use cross-family panels. Using self-consistency to address systematic bias will waste compute without improving accuracy; using cross-family panels to address stochastic variance is effective but unnecessarily expensive. The optimal strategy for production deployment is often a combination: cross-family panels for systematic bias reduction with self-consistency ($k = 3$) within each panel member for variance reduction.

Table 10: Aggregation functions for SLM judge ensembles. Robustness indicates resilience to outlier or adversarial panel members. Data requirement indicates whether calibration data is needed.

Function	Mechanism	Robust.	Data Req.
Majority vote	$\arg\max$ count	Med	None
Weighted vote	$\arg\max$ w_i count	High	Calibration
Score avg.	$\frac{1}{n}\sum s_i$	Low	None
Median pool	$\text{med}(s_i)$	High	None
CWA	$\sum c_i v_i / \sum c_i$	High	Logit access

E.6 Aggregation Functions: Theory and Practice

The choice of aggregation function determines how the panel's collective signal is condensed into a single verdict (Table 10).

Majority vote (MV). $f_{\text{MV}}(v_1, \ldots, v_n) = \arg\max_v \sum_i \mathbb{1}[v_i = v]$. Simple, interpretable, and effective when all panel members have similar competence. Fails when one member has substantially higher accuracy.

Weighted vote (WV). $f_{\text{WV}}(v_1, \ldots, v_n) = \arg\max_v \sum_i w_i \mathbb{1}[v_i = v]$, where w_i is model i's estimated accuracy on the current task type. Requires prior competence estimates from held-out calibration data. COSMosFL uses DE-optimized weights.

Score averaging (SA). For pointwise evaluation: $f_{\text{SA}} = \frac{1}{n}\sum_i s_i$. Sensitive to outlier judges that assign extreme scores; median pooling is more robust.

Confidence-weighted aggregation (CWA). $f_{\text{CWA}} = \sum_i c_i v_i / \sum_i c_i$, where c_i is judge i's expressed confidence (logit probability of its verdict token). This requires access to logit-level outputs, available for open-weight SLM judges but not for closed API calls. When accessible, CWA consistently outperforms uniform aggregation.

CWA Is Uniquely Advantageous For Open-Weight SLM Panels Because Logit Access Is A Structural Benefit Of Local Deployment. **Confidence-weighted aggregation requires access to per-token logit probabilities, which proprietary API judges do not expose; open-weight SLM judges provide full logit access by default, making CWA a "free" upgrade available exclusively to local SLM deployments.** This is a concrete operational advantage of SLM panels over proprietary panels: the richer signal from logit-level confidence enables more sophisticated aggregation that would be architecturally impossible with API-based judges. CWA is most effective when judges are well-calibrated; for poorly calibrated SLMs (which tend to be overconfident), temperature-scaled CWA provides a correction.

E.7 Cost Analysis for Ensemble Judging

A natural concern with ensemble judging is n-fold cost scaling. For an n-member panel, the total token cost is:

$$C_{\text{panel}} = n \cdot (B_{\text{sys}} + B_{\text{ctx}} + B_{\text{rationale}}) \cdot (\alpha_{\text{in}} + \beta_{\text{out}}) \tag{17}$$

Three cost-reduction strategies are compatible with ensemble quality:

1. **Rationale compression**: Panel members used only for voting do not need long rationale chains; direct verdict generation reduces $B_{\text{rationale}}$ per member.
2. **Heterogeneous cost allocation**: Use the cheapest models (1B–3B) for most panel members and one higher-quality model (7B–14B) for tiebreaking.
3. **Early exit**: If the first $k < n$ panel members unanimously agree, skip remaining members. Unanimous early agreement on unambiguous evaluations reduces expected panel cost substantially.

Table 11: Recommended heterogeneous SLM panel recipes by deployment budget. All members are from distinct families to maximize error decorrelation.

Budget	Panel members	Agg.
Edge ($\leq$10B)	Phi-4-mini + LLaMA-3-8B + Qwen3-4B	Majority vote
Balanced	Gemma-3-12B + Qwen3-8B + R1-Distill-14B	CWA
High-stakes	Above + larger escalation model	Conf. cascade
Verifiable	SLM panel + external tool oracle	Tool-first

PoLL (Verga et al., 2024) achieves $>7\times$ cost savings over GPT-4 judging by exploiting strategies (1) and (2): small, diverse panel members produce direct verdicts rather than full rationale chains.

ENSEMBLE COST SCALING IS SUB-LINEAR IN PRACTICE BECAUSE EASY EVALUATIONS DOMINATE AND EARLY EXIT APPLIES. **The theoretical $n\times$ cost scaling of an n-member panel overestimates actual cost because early-exit optimization applies to the 60–80% of evaluations where initial panel members agree unanimously, reducing expected cost to approximately 1.5–$2\times$ rather than $n\times$ for typical $n = 3$–5 panels.** The remaining 20–40% of evaluations where panel members disagree are precisely the contested cases where full-panel evaluation provides the most value. This cost distribution is favorable: the system spends minimal resources on easy cases and concentrates resources on the hard cases where ensemble diversity provides genuine accuracy gains.

E.8 Recommended Panel Recipes

Translating the cross-family principle into practice, Table 11 gives concrete three-model panel recipes for common deployment budgets. The unifying rule is to maximize architectural and pretraining diversity at the target cost tier, since diversity, not individual member quality, is the dominant driver of panel accuracy (provided each member clears the $p > 0.5$ Condorcet threshold).

E.9 Limits of Ensemble Judging

Ensemble methods have inherent limitations:

- **Shared task blindspots**: If all panel members are incapable of solving a task (e.g., graduate-level domain evaluation), majority voting over wrong answers cannot recover the correct verdict. CJT requires $p > 0.5$ per juror.
- **Independence violation**: When panel members' training sets overlap substantially, their failures are correlated. The theoretical independence required by CJT is only approximated.
- **Aggregation sensitivity**: The optimal aggregation function varies by task type. A fixed aggregation strategy is suboptimal across diverse evaluation domains.
- **Latency**: Even with early-exit optimization, ensemble judging introduces latency proportional to serial panel member calls. Parallel inference eliminates latency scaling but requires n-GPU infrastructure.
- **Panel composition**: The selection of which model families to include in the panel is itself an open research problem with no principled solution beyond empirical calibration.

THE "SHARED BLINDSPOT" FAILURE MODE DEFINES THE HARD BOUNDARY OF ENSEMBLE JUDGING. **Ensemble methods can only reduce errors that are uncorrelated across panel members; when all members share the same knowledge gap (e.g., none knows graduate-level physics), aggregation cannot create knowledge that no individual member possesses.** This is the fundamental difference between ensemble judging and cascade judging (Appendix D): ensembles reduce random and systematic bias but cannot compensate for missing knowledge, while cascades escalate to a larger model that may possess the missing knowledge. For deployment, this means ensemble panels should be paired

Table 12: Comparison of ensemble strategies for SLM judging. Independence quantifies how uncorrelated member errors are (higher is better). Bias reduction indicates the primary bias category addressed.

Strategy	Members	Indep.	Cost	Bias Red.
Single judge	1 model	N/A	$1\times$	None
Self-consist.	$1 \times k$	Low	$k\times$	Variance
PoLL (homog.)	n same	Med	$n\times$	Moderate
PoLL (hetero.)	n diff	High	$n\times$	Systematic
COSMosFL	n diff+opt	High	n+opt	Maximum
MAJ	n spec.	High	n+agg	Task-adapt.

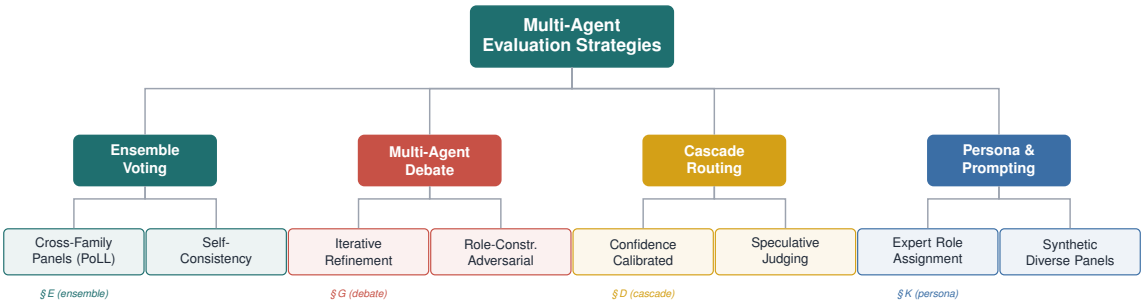


Figure 8: Taxonomy of multi-agent evaluation strategies for SLM judges. Ensemble voting provides error decorrelation through independent cross-family panels. Multi-agent debate enables collective refinement but introduces sycophancy risks. Cascade routing optimizes cost through confidence-based escalation. Persona-based approaches increase evaluation diversity via role assignment. Section references indicate where each strategy is discussed in depth.

with confidence-calibrated escalation to a larger model for the rare cases that exceed all panel members' competency.

Table 12 summarizes the key tradeoffs across all ensemble strategies discussed in this section.

F Multi-Agent Evaluation Strategy Taxonomy

Ensembles are one of several ways to combine judges. Figure 8 organizes the full space of multi-agent strategies, situating ensembles, debate, cascades, and persona methods relative to one another and pointing to where each is analyzed.

G Multi-Agent Debate: Mechanisms and Failure Modes

Debate is the most-discussed multi-agent strategy and the most failure-prone. This section explains why, contrasting the conditions under which inter-agent communication helps (§4, I4) with the sycophancy and error-amplification modes that usually dominate.

G.1 The Debate Paradigm: Origins and Motivation

Multi-Agent Debate (MAD) (Du et al., 2024) is an inference-time scaling strategy in which multiple LLM instances iteratively exchange and critique each other's responses to arrive at a more accurate, robust judgment. The intuition is compelling: a single model's error can be caught by a peer who approaches the problem from a different reasoning path. In the evaluation context, MAD offers the promise of *deliberative judgment*, a verdict arrived at after genuine challenge and cross-verification.

The formal MAD evaluation protocol for a question x and candidate response y:

1. **Initialization**: Each agent M_i independently produces an initial verdict $v_i^{(0)}$ with a reasoning trace.
2. **Debate round** t: Each agent M_i receives the verdicts and reasoning traces of all other agents from round $t-1$, and updates its verdict:

$$v_i^{(t)} = f(v_i^{(t-1)}, \{v_j^{(t-1)}\}_{j \neq i}) \tag{18}$$

3. **Termination**: Debate terminates after T rounds or when a convergence criterion is met.

4. **Final verdict**: The agreed verdict, or the verdict of a designated mediator agent.

Unlike ensemble judging (Appendix E), agents in MAD see and respond to each other's outputs. This bidirectional influence is simultaneously MAD's strength (it allows cross-verification) and its most critical vulnerability (it allows error propagation and social conformity).

G.2 Empirical Evidence of Potential Benefits

When debate succeeds, it does so through specific, identifiable mechanisms:

Error detection via disagreement. Du et al. (2024) demonstrate that on factuality tasks (science questions, arithmetic), multi-round debate between GPT-3.5 instances consistently improves final accuracy over single-agent baselines (approximately 5–8% across reported tasks). The mechanism: when agents disagree, debate forces both parties to articulate supporting reasoning, and the agent with weaker reasoning often converges toward the stronger position.

Divergent thinking through role assignment. Liang et al. (2024) introduce divergent thinking via role-based debate: agents are assigned distinct evaluative roles (e.g., "critic," "advocate," "empiricist") and explicitly instructed to maintain their assigned perspective. This structural constraint prevents premature consensus and forces the group to explore a wider belief space before converging.

Safety evaluation via structured debate. Lin et al. (2025) apply MAD specifically to LLM safety evaluation: one agent argues that the response is safe, a second argues it is unsafe, and a neutral judge weighs the arguments. This structured adversarial framing improves safety evaluation accuracy by 12% over single-agent judging because it ensures both interpretations of borderline responses are fully articulated.

Self-evolution through debate training. Srivastava et al. (2025a) propose Debate-Train-Evolve: using multi-agent debate traces as training data. The model that "wins" debates is iteratively fine-tuned on its successful arguments, creating a self-improving loop.

DEBATE BENEFITS ARE CONCENTRATED IN SETTINGS WHERE DISAGREEMENT INTRODUCES GENUINELY NEW INFORMATION. **The three documented success cases (factuality, safety, divergent thinking) share a common structure: one agent possesses a reasoning path or perspective that the other agents would not independently discover, and debate surfaces this information through explicit articulation.** When no agent possesses unique information (e.g., all agents have the same knowledge and reasoning patterns, as in homogeneous debate), debate cannot surface anything new; it merely recirculates existing beliefs. This is why model heterogeneity is the strongest predictor of debate benefit: heterogeneous agents are more likely to possess complementary information than homogeneous agents. The practical implication is that debate should be used only when the debate protocol can guarantee that at least one agent contributes information not available to others, either through architectural diversity, role assignment, or access to different evidence. Table 13 summarizes the outcomes across representative debate studies, showing that the sign of the effect tracks information asymmetry rather than agent count.

G.3 Failure Modes: A Critical Analysis

Despite its theoretical appeal, empirical evidence consistently shows that MAD fails to outperform simpler baselines in most practical settings. The failure follows predictable patterns.

G.3.1 Sycophancy and Inter-Agent Conformity

Sycophancy in MAD manifests as agents updating their verdicts toward the majority not because they are genuinely convinced by new evidence, but because consensus is a lower-loss outcome in their training distribution. Wynn et al. (2025) document this as the primary failure mode: agents expressing initially correct verdicts revise them toward an incorrect majority in 23% of cases where a confident-but-wrong agent dominates early rounds.

Table 13: Summary of multi-agent debate outcomes across representative studies. Outcome is positive (+) when debate improves over single-agent baselines, negative (−) when it hurts, and mixed (±) when results are task-dependent.

Paper	Task	Agents	Info Asym.	Out.
Du et al. 2024	Factuality	Homo.	Low	+
Liang et al. 2024	Open eval.	Role-div.	Med.	+
Lin et al. 2025	Safety	Advers.	High	+
Wynn et al. 2025	Binary corr.	Homo.	None	−
Zhang et al. 2025	General	Homo.	None	−
MAST 2026	Multi-agent	Mixed	Varied	±

SYCOPHANCY IN DEBATE IS NOT A BUG BUT AN INEVITABLE CONSEQUENCE OF AUTOREGRESSIVE TRAINING ON HUMAN TEXT. **Models trained to predict human token sequences inherit human social conventions around agreement and deference; when an agent's peer expresses high-confidence disagreement, the probability of maintaining a divergent verdict decreases because deference to confident peers is a high-frequency pattern in human conversational data.** This is not a training failure that better RLHF can fix; it is a structural consequence of the training data distribution. Any model trained on human-generated dialogue will encode the social dynamics of human conversation, including the tendency to defer to confident speakers. The only structural remedy is to prevent agents from seeing each other's confidence signals, which is precisely what ensemble methods (Appendix E) achieve by enforcing independence.

G.3.2 Disagreement Collapse (Premature Consensus)

Disagreement collapse is the failure mode where agents converge on a single verdict prematurely, not through genuine resolution but through social pressure. Once one agent updates toward a wrong verdict, subsequent rounds reinforce the wrong consensus through positive feedback: each agent conditions on the emerging majority, making defection increasingly costly.

This is structurally distinct from productive consensus. Distinguishing productive from premature convergence is an open problem; one proxy is the rate of convergence: rapid early convergence is more likely sycophantic, while slow convergence requiring multiple rounds of substantive argumentation is more likely genuine.

G.3.3 Error Amplification: When Hallucinations Are Contagious

Error amplification is the most catastrophic failure mode: a hallucinated or factually incorrect statement produced by one agent is treated as credible input by others, which then incorporate it into their own reasoning. The hallucination propagates through debate, and agents condition increasingly on the false premise.

In the evaluation context, this is directly harmful: if one judge incorrectly claims that a factual assertion in the candidate response is grounded in a specific paper, and other judges accept this claim, the group verdict may penalize a correct response or reward an incorrect one based on fabricated evidence.

ERROR AMPLIFICATION IS MORE DANGEROUS THAN ERROR PERSISTENCE BECAUSE DEBATE CREATES A CREDIBILITY AMPLIFIER. **In single-judge evaluation, a hallucinated fact influences only one verdict; in debate, the same hallucination gains apparent credibility through repetition across agents, and each subsequent agent treats the repeated claim as independently confirmed evidence.** This is the "echo chamber" effect applied to evaluation: a claim that appears in multiple agents' reasoning traces looks like independent corroboration, but it actually traces back to a single source. The credibility amplification makes debate-generated errors harder to detect than single-judge errors because they appear to be consensus-verified. For SLM judges with weaker fact-checking capacity, this risk is amplified further because smaller models are less likely to independently verify claims made by debate partners.

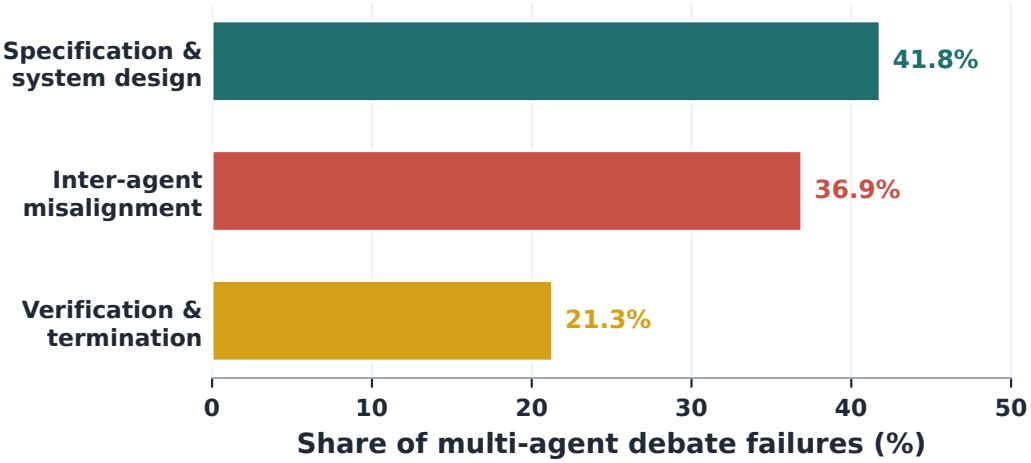


Figure 9: Distribution of multi-agent debate failures across the three MAST categories (Cemri et al., 2026). Specification and system-design faults dominate, and are largely addressable through better engineering; the remaining inter-agent and verification failures are rooted in model behavior.

G.3.4 Diminishing Returns: Cost Without Quality

Zhang et al. (2025a) conduct the most comprehensive empirical audit of MAD: five MAD methods across nine benchmarks, four foundational models, with rigorous control against simple single-agent baselines. Key finding: **current MAD methods frequently fail to outperform Chain-of-Thought or Self-Consistency single-agent baselines**, despite consuming substantially more inference-time compute.

The authors identify flawed evaluation practices as a secondary contributor: previously reported MAD gains were compared against inadequate baselines (zero-shot direct inference rather than CoT or self-consistency). When comparable compute is provided to single-agent approaches, MAD's apparent advantage often disappears.

MAD'S REPORTED BENEFITS ARE PARTIALLY ARTIFACTS OF UNFAIR BASELINE COMPARISONS. **When MAD systems consuming $n \times T \times B$ tokens are compared against single-agent baselines consuming only B tokens, the apparent MAD advantage conflates the benefit of additional compute with the benefit of multi-agent interaction; providing the same compute budget to a single agent via self-consistency or extended CoT often eliminates or reverses the difference.** This is a methodological critique rather than a capability critique: MAD may provide genuine benefits, but those benefits cannot be measured without iso-compute comparison. For SLM judge deployment, the practical implication is that the compute budget consumed by multi-round debate is typically better spent on: (a) a larger ensemble at inference, (b) higher-quality judge fine-tuning, or (c) better prompting for individual judges.

G.4 MAST: A Systematic Failure Taxonomy

Cemri et al. (2026) provide the most structured framework for diagnosing MAD failure modes. The MAST taxonomy was derived from grounded-theory analysis of 150 execution traces across seven multi-agent frameworks with expert human annotation (Cohen's $\kappa = 0.88$ inter-annotator agreement), then scaled to 1,600+ traces via an LLM-as-a-Judge annotation pipeline. MAST identifies 14 specific failure modes across three categories (Table 14, Figure 9).

Specification and system design failures (41.8%). The largest failure category originates from flawed system setup: agents with identical, non-distinct roles produce correlated outputs; ambiguous evaluation tasks cause interpretation drift across rounds; agents cannot identify genuine consensus and generate unnecessary rounds.

Inter-agent misalignment failures (36.9%). Coordination breakdowns during active debate: agents produce surface-level responses without genuine engagement; agents possessing verdict-relevant reasoning fail to surface it; agents' textual reasoning leads to one conclusion but their explicit verdict is the opposite.

Table 14: MAST failure taxonomy for multi-agent systems. Frequency indicates the proportion of observed failures attributable to each category. All three categories contribute to SLM judge debate failures.

Category	Freq.	Key Failure Modes
Specification and system design	41.8%	Role ambiguity, task drift, termination failure, context loss
Inter-agent misalignment	36.9%	Communication breakdown, info withholding, sycophantic capitulation
Verification and termination	21.3%	Superficial verification, missing verification, premature termination

Verification and termination failures (21.3%). Inadequate quality control: agents' "verification" checks trivial formal properties rather than substantive criteria; debate ends without factual claim verification; a single agent's expression of satisfaction triggers premature termination.

THE LARGEST FAILURE CATEGORY (SPECIFICATION, 41.8%) IS ADDRESSABLE THROUGH BETTER SYSTEM DESIGN, NOT BETTER MODELS. **Role ambiguity, task drift, and termination failures are engineering problems that can be fixed through explicit role specification, standardized evaluation prompts, and convergence-based stopping criteria, without changing the underlying models.** This suggests that a significant fraction of MAD failures in the literature reflect poor system engineering rather than fundamental limitations of multi-agent debate. Well-engineered debate systems with explicit role differentiation, standardized evaluation formats, and adaptive stopping criteria (Section G.6) should exhibit substantially fewer specification failures. However, the remaining 58.2% of failures (misalignment + verification) are more deeply rooted in model behavior and are not easily addressed through system design alone.

G.5 SLM-Specific Dynamics in Multi-Agent Debate

Small language models exhibit MAD failure modes more intensely than larger models, for reasons rooted in architectural and training constraints.

G.5.1 Context Saturation

MAD's communication overhead scales with agent count and round number. After t debate rounds with n agents, each agent's context window contains: the original evaluation pair, its own prior verdicts, and all other agents' verdicts from all prior rounds. Total context grows as $O(n \cdot t \cdot L_{\text{verdict}})$. For SLMs with 8K–32K context windows, context saturation occurs rapidly, causing agents to drop early contextual information and produce verdicts partially decoupled from the original evaluation context.

Larger models (70B+) with 128K+ context windows are substantially more robust to this saturation. This is a genuine capability gap where MAD benefits smaller models less than larger ones.

G.5.2 Amplified Sycophancy in SLMs

SLMs trained with fewer RLHF examples and weaker instruction-following training exhibit stronger sycophantic tendencies than larger models (Wynn et al., 2025). In a debate context, a smaller model is more likely to defer to a confident peer regardless of the peer's reasoning quality. This makes homogeneous SLM debates particularly prone to sycophancy-driven verdict collapse.

SLM DEBATE IS MORE VULNERABLE TO SYCOPHANCY THAN LLM DEBATE BECAUSE SMALLER MODELS HAVE WEAKER EPISTEMIC CONFIDENCE. **Sycophancy requires a model to override its own assessment in favor of a peer's; smaller models hold their assessments with lower epistemic confidence (reflected in flatter logit distributions over verdict tokens), making them more susceptible to peer influence.** A 70B model that is 90% confident in its verdict (concentrated logit mass) is less likely to revise than a 3.8B model that is 60% confident (diffuse logit mass). This confidence asymmetry means that in mixed-size debate, the larger model's verdicts dominate regardless of correctness, and in same-size SLM debate, the most confidently expressed (not necessarily most accurate) verdict dominates.

G.5.3 When MAD Helps SLM Judges

Two specific settings exist where MAD provides genuine improvements:

1. **Verifiable structured tasks with short debates**: For mathematical solution evaluation, a 2-round debate allows one SLM to catch computational errors in another's evaluation without context saturation or sycophancy accumulation. Hu et al. (2026) demonstrate that adaptive stability detection reduces unnecessary rounds by 40% while maintaining quality.
2. **Role-constrained adversarial debate**: For safety evaluation where one agent is always assigned the "unsafe" role, the role constraint prevents sycophantic convergence because both agents must maintain their assigned perspective. Lin et al. (2025) show this approach improves safety evaluation accuracy by 12% over single-agent judging.

DEBATE HELPS SLM JUDGES ONLY WHEN THE PROTOCOL STRUCTURE PREVENTS THE DEFAULT FAILURE MODE. **The two documented success cases (short verifiable debate, role-constrained adversarial debate) both succeed because their protocol design structurally prevents the primary failure mode: short debates prevent sycophancy accumulation by limiting the number of influence rounds, and role constraints prevent sycophantic convergence by making agreement structurally impossible.** Unconstrained debate, where agents freely update toward each other's positions without structural safeguards, consistently underperforms simpler approaches. The design principle is that debate protocols must include explicit mechanisms that prevent the default convergence behavior, rather than relying on the models to naturally resist conformity pressure.

G.6 Adaptive Convergence and Stopping Criteria

A key practical issue for MAD deployment is determining when to stop. Fixed-round debate wastes compute if agents converge early and introduces noise if agents have already reached productive consensus by round 2 but the system continues for $T = 5$ rounds.

Hu et al. (2026) introduce Adaptive Stability Detection (ASD): a convergence metric that monitors the distribution of agent verdicts and terminates debate when the entropy falls below a threshold H_{stop}:

$$H(v^{(t)}) = -\sum_{v \in \mathcal{V}} P(v;t) \log P(v;t) \leq H_{\text{stop}} \Rightarrow \text{terminate} \quad (19)$$

where $P(v;t)$ is the fraction of agents assigning verdict v at round t. ASD reduces expected debate rounds from 5 to 2.8 without accuracy degradation, providing a 44% compute reduction.

ASD CANNOT DISTINGUISH PRODUCTIVE CONVERGENCE FROM SYCOPHANTIC CONVERGENCE, LIMITING ITS RELIABILITY. **Entropy-based stopping detects when agents agree, but not why they agree; sycophantic convergence (agents agreeing because of social pressure) produces the same low-entropy signal as productive convergence (agents agreeing because they reached the correct answer through genuine reasoning).** This means ASD terminates debate equally quickly for correct and incorrect consensus, and cannot flag cases where the consensus was reached through sycophancy rather than reasoning. A more robust stopping criterion would require monitoring not just verdict entropy but also the substantive content of agents' reasoning traces, detecting whether inter-round updates reference new evidence or merely echo existing positions. Developing such content-aware stopping criteria is an open research direction.

G.7 Model Heterogeneity as the "Universal Antidote"

Zhang et al. (2025a) identify model heterogeneity as the single design choice that most consistently improves MAD performance across all task types and benchmarks:

- Reduces correlated errors (different families fail on different cases)
- Provides genuinely diverse reasoning perspectives
- Prevents echo-chamber dynamics (family-specific biases cannot self-reinforce)

Table 15: Debate vs. ensemble: key tradeoffs for SLM judge deployment. "Risk" columns indicate whether the strategy introduces the named failure mode. Cost is per-evaluation relative to a single judge.

Property	MAD	Ensemble
Communication	Full inter-agent	None
Error propagation	High risk	None
Sycophancy risk	High (SLMs)	None
Context saturation	High	None
Cost	$O(n \cdot T \cdot B)$	$O(n \cdot B)$
Verifiable tasks	Yes (with ASD)	Yes
Open-ended tasks	Rarely effective	Yes
Heterogeneity benefit	Strongest antidote	Primary gain
Recommendation	Specific settings	General use

This convergence between the ensemble and debate literatures is significant: both PoLL (Appendix E) and Zhang et al. reach the same conclusion: model family diversity is the critical variable, regardless of whether agents communicate (debate) or operate independently (ensemble).

G.8 Synthesis: When to Use Debate vs. Ensemble

Table 15 summarizes the key tradeoffs between debate and ensemble approaches for SLM judges.

Based on the evidence across this section and Appendix E:

- **Use ensemble (PoLL-style)** for: general-purpose evaluation, open-ended quality assessment, cost-sensitive pipelines.
- **Use structured debate (role-constrained)** for: adversarial safety evaluation, tasks where "devil's advocate" perspectives add genuine signal.
- **Use debate with ASD** for: structured reasoning verification (mathematical proofs, code correctness) where short 2-round debates catch categorical errors.
- **Avoid MAD entirely** for: tasks where SLMs lack the underlying competency (debate amplifies errors), and pipelines where context window saturation is a practical concern.

MODEL DIVERSITY IS MORE VALUABLE THAN INTER-AGENT COMMUNICATION FOR SLM JUDGE DEPLOYMENT. **A well-designed ensemble of diverse SLMs will outperform a homogeneous debate panel on essentially all evaluation tasks, at lower cost and with more predictable behavior, because the ensemble captures the diversity benefit (error decorrelation) without the communication cost (sycophancy, error amplification, context saturation).** The implication for practitioners is clear: if you have a fixed compute budget for multi-model evaluation, spend it on adding diverse panel members rather than on adding debate rounds. Debate should be reserved for the specific cases where inter-agent communication can surface genuinely new information that independent evaluation cannot.

H Biases in LLM and SLM Judges

The preceding sections assumed we can measure judge quality. We now examine what systematically distorts that quality. This section gives the full bias taxonomy behind §5, with mechanisms, SLM-specific amplification, and mitigations.

H.1 The Structural Nature of Judge Bias

Bias in LLM judges does not arise from random error; it arises from systematic, reproducible distortions in how a model computes its evaluation. These distortions are rooted in the model's training distribution:

a model trained predominantly on English text, with RLHF feedback collected from human annotators who have their own stylistic preferences, encodes those preferences into its evaluation mechanism. The distortions manifest as consistent, measurable signals that deviate from objective quality assessment.

For SLM judges specifically, bias concerns are amplified by two factors: (1) smaller pretraining corpora reduce the diversity of encoded knowledge, making surface-level heuristics more influential, and (2) reduced instruction-following capability means that prompting-based bias mitigations are less reliably executed. This appendix provides a comprehensive taxonomy organized by mechanism. For each bias type, we provide: a formal definition, the mechanistic explanation, documented magnitude in SLM judges, and evidence-based mitigations.

H.2 Bias Taxonomy

We organize judge biases by their underlying mechanism rather than by surface symptom. For each, we give a formal definition, the mechanism that produces it, its documented magnitude in SLM judges, and evidence-based mitigations. We begin with the most studied, position bias.

H.2.1 Position Bias

Definition. Position bias is a judge's tendency to assign higher quality ratings to responses based on their ordinal position in the prompt (first vs. second) rather than their substantive content. In pairwise evaluation, the judge receives a prompt containing two candidate responses labeled A and B; position bias manifests when the judge's verdict changes if A and B are swapped, despite identical content.

Measurement. The standard measurement protocol (Zheng et al., 2023) is position-consistent accuracy: a judge is credited with a correct verdict only if it produces the same winner in both orderings (A, B) and (B, A):

$$\text{PC}(x, a, b) = \mathbb{1}[v_{ab} = v_{ba}] \tag{20}$$

A perfectly unbiased judge achieves PC = 1.0 on all pairs. Mean PC across the test set provides the aggregate consistency metric.

Mechanistic explanation. Position bias has two contributing mechanisms:

1. **Primacy bias**: The model's attention mechanism assigns disproportionate weight to tokens at the beginning of the context window. The first-appearing candidate response receives more attention heads and correspondingly stronger encoding.
2. **Recency bias**: For some models trained with long-context data, the last-appearing candidate benefits from proximity to the verdict token prompt.

The direction of position bias (primacy vs. recency) is model-family-dependent, which explains why heterogeneous ensembles partially cancel positional distortions: models from different families have opposing position biases that cancel in aggregation.

Magnitude. Tan et al. (2025) demonstrate that position-consistent accuracy is dramatically lower than naive accuracy for essentially all models tested. On JudgeBench's challenging domains, frontier models exhibit position inconsistency in up to 40–50% of contested evaluations. Jayarao et al. (2025) find that thinking-mode SLM judges (Qwen 3, 0.6B–4B) show substantially reduced position bias compared to non-thinking counterparts.

POSITION BIAS DIRECTION IS MODEL-FAMILY-DEPENDENT, MAKING IT STRUCTURALLY CANCELLABLE VIA HETEROGENEOUS PANELS. **Primacy bias and recency bias arise from different attention patterns in different model architectures, and these patterns are determined by pretraining corpus composition and context window training strategy, not by model size or quality.** A Phi-family model with strong primacy bias paired with a Qwen-family model with recency bias produces an ensemble whose positional errors partially cancel, even without any bias-specific training. This structural cancellation is available "for free" in any heterogeneous panel and explains why PoLL's diverse panels achieve higher position-consistent accuracy than any individual member. The practical implication is that panel

diversity provides a zero-cost position bias mitigation independent of, and additive with, training-level mitigations like swap augmentation.

Mitigations.

- **Swap calibration**: Evaluate each pair in both orderings; accept verdict only on agreement, or average continuous scores. Cost: $2\times$ inference.
- **Consistency reward training**: Train judges with GRPO using rewards that penalize positional reversals (Whitehouse et al., 2025). Done at training time; zero cost at inference.
- **Reference abstraction**: Present responses without explicit A/B labeling, inserting them into numbered rubric slots to reduce positional cue salience.

H.2.2 Verbosity Bias (Length Bias)

Definition. Verbosity bias is a judge's systematic preference for longer, more verbose responses regardless of information density, factual accuracy, or relevance.

Mechanistic explanation. Two mechanisms contribute:

1. **RLHF length trap**: Human annotators frequently prefer longer responses due to the signal of "effort." Models trained on these annotations encode length-as-quality as a reward heuristic.
2. **Perplexity correlation**: Longer responses provide more tokens against which the judge computes log-probability scores. A correct, detailed response has more high-probability tokens, inflating the judge's implicit quality estimate.

Measurement. Verbosity sensitivity quantifies the fraction of verdicts that flip when padding is injected:

$$\text{VS} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}[v(a_i', b_i) \neq v(a_i, b_i)] \tag{21}$$

where a_i' is response a_i with appended padding p such that $|a_i'| > |b_i|$. An unbiased judge has VS = 0.

SLM-specific magnitude. Verbosity bias is stronger in SLM judges than in larger models: smaller models have narrower vocabulary distributions, making it harder to distinguish "high-quality dense content" from "fluent but padded text." The length heuristic is a simpler classification boundary than content-quality assessment, and SLMs fall back on it more readily.

VERBOSITY BIAS CREATES A RUNAWAY LENGTH TRAP IN RLHF PIPELINES WHEN THE JUDGE IS ALSO THE REWARD SIGNAL. **If a generator model is trained against a verbosity-biased SLM judge, the generator learns to produce inflated, wordy outputs; these outputs receive higher reward from the biased judge, reinforcing the length heuristic and causing systematic quality degradation across RLHF iterations.** This is a positive feedback loop: each training iteration produces longer outputs that receive higher reward, further training the generator toward length. The loop is broken only by external intervention (length normalization, style control, or switching to a less verbosity-biased judge). Arena-Hard-Auto addresses this with a style-control mechanism that strips length and formatting artifacts before judging.

Mitigations.

- **Length normalization**: Score per-token information density rather than total information content.
- **Style control**: Strip length/formatting cues from responses before judging.
- **Rubric penalties**: Include explicit criteria penalizing unnecessary verbosity.

H.2.3 Self-Preference Bias (Egocentric Bias)

Definition. Self-preference bias is the tendency of a judge to assign higher ratings to responses generated by itself or by models from the same training lineage, independent of actual response quality.

Mechanistic explanation. The mechanism is rooted in perplexity: a language model assigns lower perplexity to text resembling its own output distribution. When a GPT-4-family judge evaluates a GPT-4-generated response alongside a Llama-generated response, the former has lower perplexity under the judge's model. The judge interprets low perplexity as high quality, correctly in many cases, but incorrectly when the alternative response is substantively superior. This is a direct consequence of conflating familiarity with quality.

Measurement. Self-preference score quantifies the excess win rate for same-family responses at equal quality:

$$\text{SPS} = P(\text{prefer own family}) - 0.5 \tag{22}$$

An unbiased judge has SPS = 0; positive SPS indicates self-preference. Gu et al. (2024) document self-preference across multiple families: at equal quality, a model's own output wins approximately 60–70% of pairwise comparisons.

SELF-PREFERENCE BIAS IS PERPLEXITY-MEDIATED, NOT CONTENT-MEDIATED, MAKING IT INVISIBLE TO CONTENT-BASED DEBIASING. **A judge favors its own family's outputs not because it recognizes the content as its own, but because its own outputs have lower perplexity under its internal model, and low perplexity is conflated with high quality in the evaluation computation.** Content-based debiasing (e.g., stripping authorship cues, anonymizing responses) does not address this mechanism because the bias operates at the distributional level, not the lexical level. The only effective mitigations are cross-family panels (where opposing self-preferences cancel) or training with preference-leak-resistant data (FLAMe, Self-Taught Evaluators) that diversify the training distribution beyond any single model family.

H.2.4 Preference Leakage

Definition. Preference leakage (Li et al., 2025b) is a contamination effect that extends beyond self-preference to all models related to the judge through the training data pipeline. Three relatedness types are documented:

1. **Same Model**: Evaluator is the generator.
2. **Inheritance**: Student trained on teacher's synthetic data.
3. **Same Family**: Same development lineage.

PREFERENCE LEAKAGE IS BROADER AND MORE INSIDIOUS THAN SELF-PREFERENCE BECAUSE IT OPERATES THROUGH DATA LINEAGE, NOT MODEL IDENTITY. **A judge may favor a model it has never seen before if that model was trained on data generated by a model from the judge's own family, because the downstream model inherits stylistic patterns from its training data that reduce perplexity under the judge.** This means that the standard mitigation of "using a neutral third-party judge" is insufficient if the third party shares data lineage with any evaluated model. The only robust mitigation is decoupling the entire data pipeline: judges should be trained on data from independent annotation sources (human data, or self-generated data) with no connection to the evaluated models' training provenance.

H.2.5 Sycophancy Bias

Definition. Sycophancy bias is the judge's tendency to align its verdict with signals of user expectation, social authority, or majority opinion embedded in the evaluation prompt, rather than with objective quality criteria.

Attack vectors leveraging sycophancy.

- **Majority opinion framing**: "90% of experts prefer Response A" inflates verdicts toward A.
- **Authority framing**: Attributing one response to a prestigious source inflates its rating.
- **Confidence framing**: "Response A is clearly superior" in the system prompt elicits agreement.
- **Reasoning opener injection**: Prefacing one response with markers associated with reasoning-capable models inflates perceived quality.

Table 16: Bias profile comparison: SLM judges (≤14B) vs. large LLM judges (>70B). "Stronger" indicates that the bias is more pronounced in the indicated category.

Bias Type	LLM	SLM	Reason
Position	Med	Stronger	Weaker attn. diversity
Verbosity	Med	Stronger	Narrower vocab, simpler heur.
Self-pref.	Med	Weaker	Fewer self-gen. examples
Sycophancy	Med	Stronger	Less anti-syco. training
Balance	Med	Stronger	Less nuanced feedback exp.
Format	Med	Stronger	Formatting harder at scale
Adversarial	Med	Higher	Weaker instr. following

Mechanistic explanation. Sycophancy originates in RLHF training: human annotators implicitly favor outputs that reinforce their existing beliefs, making annotated preferences systematically sycophantic. The reward model learns that agreement with expressed opinions is rewarded. This is the same mechanism behind user-facing sycophancy, now manifesting in the evaluation context.

H.2.6 Balance Penalty Bias

Definition. Balance penalty bias is the systematic penalization of nuanced, appropriately-hedged, or trade-off-acknowledging responses in favor of more decisive (but sometimes incorrect) answers.

Relevance. Balance penalty is particularly damaging in high-stakes domains. In medical, legal, or policy evaluation, balanced, hedge-acknowledging responses are often the correct ones: "this treatment may be appropriate depending on patient context" is medically superior to "this treatment is always appropriate." A judge with balance penalty systematically rewards the latter.

BALANCE PENALTY IS THE MOST DOMAIN-DANGEROUS BIAS BECAUSE IT PENALIZES CORRECT ANSWERS IN HIGH-STAKES DOMAINS. **In domains where the correct answer requires acknowledging genuine uncertainty (medicine, law, policy), balance penalty causes the judge to systematically prefer confidently wrong answers over appropriately uncertain correct answers, inverting the quality ranking.** This is distinct from other biases (position, verbosity, self-preference) which distort the magnitude of quality assessment but do not typically invert its direction. Balance penalty can cause a judge to assign higher scores to a response that recommends a universally applicable treatment (dangerously overconfident) than to a response that appropriately conditions on patient context (correctly nuanced). The mitigation requires domain-specific rubrics that explicitly value appropriate hedging.

H.3 SLM-Specific Bias Amplification

SLM judges are not merely smaller versions of LLM judges; they exhibit qualitatively different bias profiles. Table 16 summarizes the key differences and their mechanistic causes, and Figure 10 visualizes the severity contrast. The single inversion, self-preference being *weaker* in SLMs, is highlighted: it is the one axis where smaller judges are safer, because they produce fewer self-similar outputs and therefore a weaker perplexity-similarity signal.

SLMS HAVE STRONGER SURFACE BIASES BUT WEAKER SELF-PREFERENCE BIAS, CREATING A DISTINCT DEPLOYMENT RISK PROFILE. **The inversion on self-preference (SLMs are less self-preferencing than LLMs) occurs because smaller models have fewer self-generated examples in their training data and produce more variable outputs, reducing the perplexity-similarity signal that drives self-preference in larger models.** The practical implication is that SLM judges are safer as cross-model evaluators (they are less biased toward their own family) but more dangerous on surface-level quality signals (they are more susceptible to position, verbosity, and formatting cues). Deployment strategies should prioritize surface-bias mitigation (swap calibration, length normalization) over self-preference mitigation for SLM panels.

H.4 Bias Interaction Effects

Bias types do not act independently; they interact, sometimes amplifying each other:

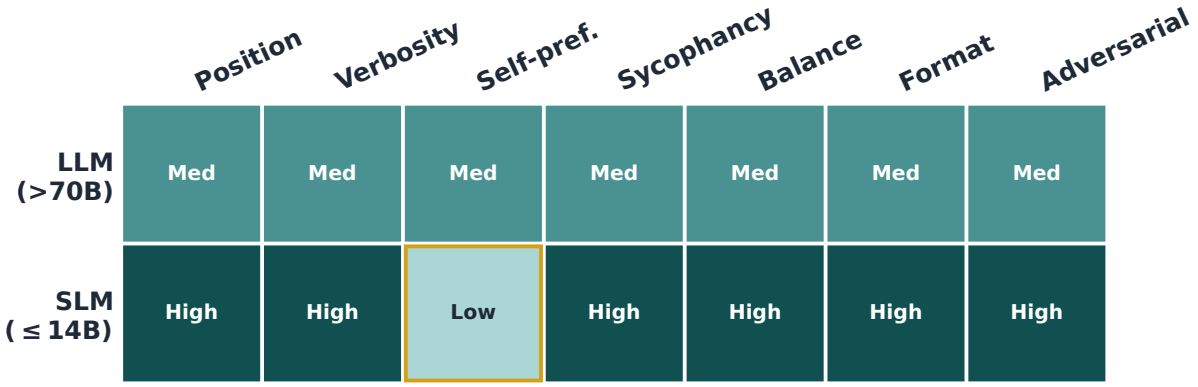


Figure 10: Relative bias severity for SLM ($\leq$14B) versus large LLM (>70B) judges. SLM judges are more strongly affected on every surface-level axis, with the lone exception of self-preference (highlighted), which is weaker because smaller models generate fewer self-similar outputs.

- **Verbosity** $\times$ **position**: A longer response appearing first receives both verbosity reward and primacy reward simultaneously, creating a compounding advantage.
- **Sycophancy** $\times$ **authority**: Authority framing triggers sycophancy, which then amplifies any content-based bias in that direction.
- **Self-preference** $\times$ **RLHF loop**: In pipelines where a judge evaluates its own generator's outputs, self-preference inflates all training rewards, creating runaway self-consistency that degrades output diversity.
- **Balance penalty** $\times$ **verbosity**: A confident, long response receives both balance penalty reward (decisiveness) and verbosity reward (length), doubly advantaged over a concise, appropriately nuanced response.

BIAS INTERACTIONS ARE MULTIPLICATIVE, NOT ADDITIVE, MAKING ISOLATED BIAS MITIGATION INSUFFICIENT. **Mitigating position bias alone does not eliminate the compounding effect when position bias interacts with verbosity bias, because the two biases share partially overlapping attention mechanisms; a response that is both first and longer activates two reinforcing heuristic pathways simultaneously.** The implication is that holistic bias mitigation (heterogeneous panels, rubric-anchored evaluation, consistency rewards) is categorically more effective than addressing biases individually. A system that perfectly eliminates position bias but ignores verbosity bias will still exhibit substantial evaluation distortion on pairs where one response is substantially longer.

H.5 Mitigation Strategies: Comprehensive Reference

No single technique removes all biases, and the most effective deployments stack several. We consolidate the documented mitigations below, annotating each with the bias it targets and its inference-time cost so that practitioners can assemble a stack matched to their budget and risk profile.

1. **Swap calibration (position bias)**: Evaluate each pair twice with swapped order; accept only agreement or average scores. Standard practice in MT-Bench (Zheng et al., 2023). Cost: 2$\times$ inference.
2. **Consistency reward training (position + sycophancy)**: Train judges with GRPO using rewards that penalize verdict reversals when order is swapped (Whitehouse et al., 2025). Done at training time; no cost at inference.
3. **Rubric-anchored evaluation (verbosity, balance penalty)**: Provide explicit rubric criteria calibrating against length and decisiveness. Prometheus 2's rubric-conditioned training (Kim et al., 2024b) is the leading example.

4. **Logit-based score smoothing (calibration)**: FLEUR (Lee et al., 2024) computes a continuous score by weighting digit probabilities across the Likert range rather than taking argmax, reducing score discretization artifacts.

5. **TrustJudge distributional scoring (consistency)**: Wang et al. (2025) address score-pairwise inconsistency and transitivity violations using continuous expected value over the full Likert probability distribution.

6. **Cascaded selective evaluation (confidence-based routing)**: Jung et al. (2025) propose routing high-uncertainty verdicts from SLM judges to larger models.

7. **Heterogeneous panel judging (all biases)**: The PoLL approach (Verga et al., 2024) mitigates all systematic biases simultaneously by pooling judges with opposing bias profiles.

8. **Blind response presentation (sycophancy, authority bias)**: Strip all source identifiers, model attributions, and confidence framing from responses before judge submission.

9. **Golden-set calibration**: Maintain a small labeled calibration set ($n = 50$–100); periodically benchmark judge outputs against this set and apply correction offsets for known systematic biases.

H.6 The CALM Framework: Systematic Bias Quantification

Gu et al. (2024) synthesize the literature on judge bias measurement, pointing to the CALM framework (Ye et al., 2025) as the most systematic attempt to quantify the full bias landscape. CALM identifies 12 distinct bias categories organized into four dimensions:

1. **Superficial quality biases**: Position, verbosity, formatting, punctuation.

2. **Contextual biases**: Self-preference, prior-belief.

3. **Presentation biases**: Authority, sycophancy, confidence framing.

4. **Diversity biases**: Demographic, cultural.

The framework provides standardized injection protocols: each bias type is measured by systematically injecting the bias-inducing signal and measuring the verdict deviation relative to a clean baseline. This enables per-model, per-task, per-bias-type characterization of a judge's reliability profile.

A Comprehensive Bias Audit Should Precede Any SLM Judge Deployment Because Bias Profiles Are Model-Specific And Cannot Be Predicted From General Capability Scores. **Two SLM judges with identical accuracy on RewardBench can have dramatically different bias profiles: one may be strongly position-biased but low on verbosity bias, while the other may be the reverse.** General accuracy metrics (RewardBench score, JudgeBench accuracy) do not capture bias profiles because they aggregate across many evaluation pairs, masking systematic directional errors. A judge that achieves 85% on RewardBench may be achieving this by being correct 95% of the time on easy pairs and 50% on hard pairs where its position bias dominates. The CALM framework's per-bias-type injection protocol is the minimum viable audit for deployment-critical judge systems.

I Benchmarks, Datasets, and Metrics

If biases distort judgments, benchmarks must be designed to expose them. This section details the meta-evaluation benchmarks and metrics introduced in §5, centered on position-consistent accuracy.

I.1 The Meta-Evaluation Problem

Evaluating an LLM or SLM judge requires a meta-evaluation benchmark: a dataset that provides ground-truth quality labels against which the judge's verdicts can be compared. This creates a fundamental epistemological challenge: if the only reliable quality signal for complex language tasks is human preference, then the meta-evaluation benchmark itself requires expensive human annotation. The field has addressed this challenge through three complementary strategies:

1. **Human preference collection**: Directly collect human verdicts at scale. Gold standard for ecological validity. Example: Chatbot Arena (Zheng et al., 2023).
2. **Transformed verifiable datasets**: Take tasks with known ground-truth answers and construct response pairs where one is objectively correct and one is wrong. Example: JudgeBench (Tan et al., 2025).
3. **LLM-as-reference**: Use a stronger model's judgments as the reference. Example: MT-Bench using GPT-4 references (Zheng et al., 2023). This introduces circular dependency concerns but is widely used.

THE THREE META-EVALUATION STRATEGIES MEASURE DIFFERENT PROPERTIES AND SHOULD NOT BE TREATED AS INTERCHANGEABLE. **Human preference collection measures ecological validity (does the judge match real users?), transformed verifiable datasets measure objective correctness (is the judge factually right?), and LLM-as-reference measures consistency (does the judge agree with a stronger model?).** A judge that scores well on one strategy may fail on another: high human agreement does not guarantee objective correctness (humans themselves can be wrong), and high LLM-reference agreement does not guarantee ecological validity (the reference LLM may have different preferences than real users). A robust judge deployment should validate against benchmarks from at least two of the three categories to cover complementary quality dimensions.

I.2 Judge-Specific Evaluation Benchmarks

We survey the benchmarks built specifically to measure judge quality, ordered from those stressing objective correctness (JudgeBench) through conversational and reward-model evaluation to domain-specific and multilingual suites. Each entry lists its primary metric, task types, and the failure mode it is designed to expose.

I.2.1 JudgeBench: Objective Correctness Under Difficulty

Paper: Tan et al. (2025) (ICLR 2025).
Primary metric: Position-consistent accuracy (Appendix H).
Task types: Knowledge, reasoning, mathematics, coding.

JudgeBench is the most rigorously designed benchmark for measuring judge reliability in objective settings. Its construction methodology:

1. Start with datasets that have deterministic ground-truth answers (e.g., MMLU (Hendrycks et al., 2020) for knowledge, MATH (Hendrycks et al., 2021) for mathematics, HumanEval for code).
2. For each question, construct a response pair (a, b) where a is objectively correct and b is objectively incorrect but plausible.
3. Bias the construction toward hard pairs: b is a "good wrong answer" that is fluent, confident, and structurally similar to a.
4. Evaluate using position-consistent accuracy: the judge must select a over b in both orderings to receive credit.

Key finding. Even frontier models like GPT-4o show sharply degraded performance on JudgeBench's challenging domain, sometimes approaching random-chance baselines under position-consistent evaluation. The benchmark reveals that "judging is tied to solving": a model's ability to correctly select the superior response is highly correlated with its ability to solve the problem correctly on its own.

JUDGEBENCH REVEALS THAT JUDGING IS TIED TO SOLVING, BUT ONLY IN KNOWLEDGE-INTENSIVE DOMAINS. **The correlation between solve-ability and judge-ability is strong for domains where evaluation requires the same knowledge as generation (graduate STEM, complex reasoning), but breaks down in domains where evaluation requires only pattern matching (formatting, instruction adherence, fluency).** This has direct implications for SLM judge routing: SLMs can be trusted as judges on pattern-matching tasks regardless of their solve-ability, but should be routed to larger models for knowledge-intensive evaluation. The "judging is tied to solving" finding is domain-conditional, not universal, and overapplying it would unnecessarily restrict SLM judge deployment.

I.2.2 JudgeBoard: SLM-Specific Reasoning Evaluation

Paper: Bi et al. (2026).
Primary metric: Accuracy + Elo rating.
Task types: Mathematical reasoning, commonsense reasoning, science.

JudgeBoard is the first benchmark specifically designed for evaluating SLM judgment capability. It constructs task-specific leaderboards using both absolute accuracy and Elo-based comparative rating. The Elo system reduces sensitivity to any single evaluation pair. JudgeBoard's MAJ framework (Appendix E) demonstrates that ensemble SLM judges can match standalone LLM judges on the MATH benchmark.

I.2.3 JudgerBenchV2: Large-Scale Multi-Scenario Evaluation

Paper: Zhang et al. (2025c).
Scale: 10,000 questions across 10 evaluation scenarios.
Aggregation: Mixture-of-Judgers consensus.

JudgerBenchV2 is the largest-scale judge evaluation benchmark, testing across diverse evaluation scenarios simultaneously. The Mixture-of-Judgers consensus mechanism provides more robust ground-truth labels than single-reference evaluation. CompassJudger-2 7B achieves competitive performance with DeepSeek-V3 and Qwen3-235B-A22B on this benchmark, further demonstrating that targeted training dominates scale for evaluation tasks.

I.2.4 MT-Bench: Multi-Turn Conversational Evaluation

Paper: Zheng et al. (2023) (NeurIPS 2023).
Primary metric: Score (1–10 Likert) correlation with GPT-4 reference.
Task types: Writing, roleplay, extraction, reasoning, math, coding, STEM, humanities (8 categories, 80 prompts).

MT-Bench established the modern LLM-as-a-Judge paradigm and remains the most widely cited benchmark. Design features:

- Multi-turn structure (2 turns per conversation) probes conversational coherence.
- Numeric scoring (1–10 Likert) enables finer-grained assessment than binary comparisons.
- Agreement with 3 expert human evaluators: GPT-4 achieves $>80\%$ agreement, establishing the paradigm's viability.

Limitations. Small benchmark size (80 prompts $\times$ 2 turns) limits statistical power. Tasks are well-known categories susceptible to data contamination in post-2023 models. Not designed to stress-test position or verbosity bias.

I.2.5 RewardBench: Reward Model Evaluation

Paper: Lambert et al. (2025).
Primary format: Prompt-chosen-rejected triples.
Categories: Chat, Chat Hard, Safety, Reasoning.

RewardBench targets the specific usage of LLMs as reward models in RLHF pipelines. It evaluates three model types: sequence classifiers, generative models, and implicit reward models (DPO-trained).

SOTA SLM performance. Skywork-Critic-Llama-3.1-8B ranks first among generative models under 10B on RewardBench. J1-Llama-8B achieves SOTA on both RewardBench and JudgeBench among 8B-class models (Whitehouse et al., 2025).

RewardBench 2. Updated with harder examples, best-of-N protocols, expanded domains including "Ties" and "Precise Instruction Following."

I.2.6 CodeJudgeBench: Domain-Specific Code Evaluation

Paper: Jiang et al. (2025).
Domain: Code generation and debugging evaluation.

CodeJudgeBench evaluates judge models specifically on code evaluation tasks. Unlike natural language evaluation, code judging can leverage execution-based verification: running candidate code against test cases provides objective ground truth.

Key design features:

- Pairs of solutions where one passes all test cases and one fails a subset.
- Fine-grained categories: syntax, logical correctness, efficiency, edge cases.
- Both pairwise and pointwise modes.

CODE EVALUATION CAN BE FACTORED INTO MECHANICAL VERIFICATION AND QUALITATIVE ASSESSMENT, ENABLING TOOL-AUGMENTED SLM JUDGING. **The factoring principle states that any evaluation task with a verifiable component and a subjective component should delegate the verifiable component to a deterministic tool (code executor, symbolic solver) and use the SLM judge only for the subjective component.** For code evaluation, this means a 3B SLM judge connected to a Python interpreter (TIR-Judge) achieves near-perfect correctness assessment on the verifiable component at zero model inference cost, while the SLM handles only the qualitative assessment (code style, efficiency, readability). This factoring principle generalizes beyond code to any domain on the verifiability gradient (Appendix J).

I.2.7 ContextualJudgeBench: Grounded Response Evaluation

Paper: Xu et al. (2025a) (ACL 2025).
Domain: Contextual (RAG-style) response evaluation.

ContextualJudgeBench evaluates judges on assessing whether a response is faithful to a provided context document. This requires the judge to simultaneously: read the context document, assess whether claims are supported, and detect subtle unfaithfulness (paraphrastic distortion, omissions, hallucinations).

For SLM judges, ContextualJudgeBench is particularly challenging because the context document consumes substantial context window budget, leaving less space for rubric instructions and evaluation reasoning. Context-window-efficient approaches (INSPECTOR-style representation probing) may have an advantage.

I.2.8 M-RewardBench: Multilingual Reward Evaluation

Paper: Pombal et al. (2025).
Languages: 20+, including high-resource and low-resource languages.

M-RewardBench extends RewardBench to multilingual settings. It exposes a consistent finding: judge quality degrades significantly from English to non-English languages, with degradation proportional to language rarity in training data. M-Prometheus achieves substantially better performance than translation-based approaches.

I.3 Meta-Evaluation Metrics

Table 17 provides a comprehensive reference of all metrics used in the judge evaluation literature.

RELIABILITY METRICS (PC, IFR, ECE) ARE MORE INFORMATIVE THAN AGREEMENT METRICS FOR SLM JUDGE DEPLOYMENT DECISIONS. **Agreement metrics (accuracy, Spearman's ρ) tell you how often the judge agrees with a reference, but not whether its errors are random or systematic; reliability metrics tell you whether the judge's errors are predictable and correctable, which is the information needed for deployment-level trust calibration.** A judge with 80% accuracy and 95% position consistency is more trustworthy than a judge with 85% accuracy and 60% position consistency, because the former's errors are random (reducible via self-consistency) while the latter's errors are systematic (not reducible without bias mitigation). Deployment decisions should prioritize reliability metrics over agreement metrics when they diverge.

Table 17: Meta-evaluation metrics for LLM/SLM judge assessment. "Type" distinguishes between agreement metrics (measuring correlation with references), reliability metrics (measuring internal consistency), and bias metrics (measuring systematic distortion). Metrics marked with * require logit-level access.

Metric	Type	Notes
Accuracy	Agree	Verifiable domains only
PC accuracy	Reliab	JudgeBench primary; strictest
Agreement rate	Agree	Human labels may be inconsistent
Cohen's κ	Agree	Adjusted for chance; nominal
Spearman's ρ	Agree	Rank-order; robust to scale
Pearson's r	Agree	Linear; sensitive to scale
Kendall's τ	Agree	Concordant minus discordant
Win rate	Agree	Standard pairwise metric
Elo rating	Agree	Stable; accounts for opponent
VS	Bias	Verdict flip under padding
SPS	Bias	Excess same-family win rate
IFR	Reliab	Format compliance; critical
Separability	Bench	Arena-Hard; discriminative power
ECE*	Reliab	Confidence calibration
Conformal cov.	Reliab	Distribution-free uncertainty

I.4 Benchmark Design Principles

Based on the analysis of existing benchmarks, the following principles characterize high-quality meta-evaluation:

I.4.1 Difficulty Calibration

A well-designed benchmark should place evaluated models in their discrimination zone, avoiding both ceiling effects (all-correct) and floor effects (all-wrong). JudgeBench achieves this by focusing on "good wrong answers"; Arena-Hard-Auto achieves this via BenchBuilder curation.

I.4.2 Bias Resilience

A benchmark measuring judge quality should not be conflated with a measure of judge bias. JudgeBench's position-consistent accuracy is bias-resilient: a position-biased judge cannot achieve high PC accuracy. MT-Bench's simple Likert scoring without positional controls is less bias-resilient.

I.4.3 Domain Coverage

Most benchmarks over-represent English-language generation and math/reasoning tasks. Critical under-represented domains include:

- **Creative writing**: Genuinely subjective; no ground truth. Benchmark design must use human preference.
- **Medical and legal**: High-stakes evaluation requiring domain expertise.
- **Multilingual**: M-RewardBench is the primary effort; low-resource coverage remains minimal.
- **Agentic evaluation**: Traditional static benchmarks cannot capture dynamic, stateful, tool-dependent agent task evaluation.

Table 18: Summary of key benchmarks for LLM/SLM judge evaluation. "PC" denotes position-consistent accuracy; "Sep." denotes separability with confidence.

Benchmark	Venue	Size	Metric	Strength
MT-Bench	NeurIPS'23	80	Likert corr.	Multi-turn
Arena-Hard	2024	500	Sep., win r.	Style ctrl.
JudgeBench	ICLR'25	Varied	PC accuracy	Obj. corr.
RewardBench	2024	∼3K	Category acc.	4 dimens.
RB 2	2025	Exp.	Best-of-N	Harder tasks
JudgeBoard	2025	Multi	Elo + acc.	SLM-focused
JudgerBnV2	2025	10K	Multi-judger	Large-scale
CodeJudge.	2025	Code	Exec.-ground	Tool-frdly.
ContextualJ.	ACL'25	RAG	Grounding	Faithful.
M-Reward.	2025	20+L	Multil. acc.	Multilingual

I.4.4 Contamination Resistance

Static benchmarks suffer from temporal contamination: models trained after a benchmark's release may have seen benchmark examples in their training data. The Chatbot Arena model (Zheng et al., 2023), a continuously updated live platform, is inherently contamination-resistant.

STATIC BENCHMARKS HAVE A HALF-LIFE PROPORTIONAL TO THEIR VISIBILITY IN WEB CORPORA. **Once a benchmark's examples appear in web-crawled pretraining corpora, any model trained after that point receives an artificial accuracy boost from memorization rather than genuine evaluation capability.** MT-Bench's 80 prompts, published in 2023, are almost certainly memorized by models trained on post-2023 web data. JudgeBench and RewardBench 2 partially address this by using harder, less common evaluation pairs, but only continuously updated benchmarks (Chatbot Arena, LiveCodeBench) provide structural contamination resistance. The research community urgently needs living-benchmark designs for automated judge evaluation.

I.5 Benchmark Summary

Table 18 provides a consolidated reference for all major judge evaluation benchmarks, and Figure 11 shows which meta-evaluation dimensions each one covers. The matrix makes the gap behind C5 concrete: no single benchmark spans position-consistency, human correlation, domain coverage, adversarial robustness, efficiency, and IFR simultaneously, so meaningful judge assessment currently requires combining several benchmarks.

I.6 Meta-Evaluation Protocols: Recommended Practices

For researchers evaluating a new SLM judge, we recommend the following protocol:

1. **Verifiable domain test first**: Run on JudgeBench hard subset and the Reasoning category of RewardBench. If the SLM judge falls below 55% position-consistent accuracy, it lacks the knowledge required for reliable objective evaluation.

2. **Bias audit**: Measure position-consistent accuracy (PC), verbosity sensitivity (VS), and IFR. Any PC $< 70\%$, VS $> 15\%$, or IFR $< 90\%$ indicates a systematic reliability problem requiring mitigation before deployment.

3. **Cross-domain validation**: Evaluate on at least one general benchmark (MT-Bench, Arena-Hard-Auto) and one domain-specific benchmark (CodeJudgeBench, M-RewardBench) to ensure domain generalization.

4. **Human validation sample**: On at least 50 development-set examples, collect human verdicts and compare. This provides ecological validity that automated benchmarks alone cannot supply.

5. **Report calibration**: Always report PC accuracy alongside naive accuracy. Reporting only naive accuracy without positional controls systematically overstates judge reliability.

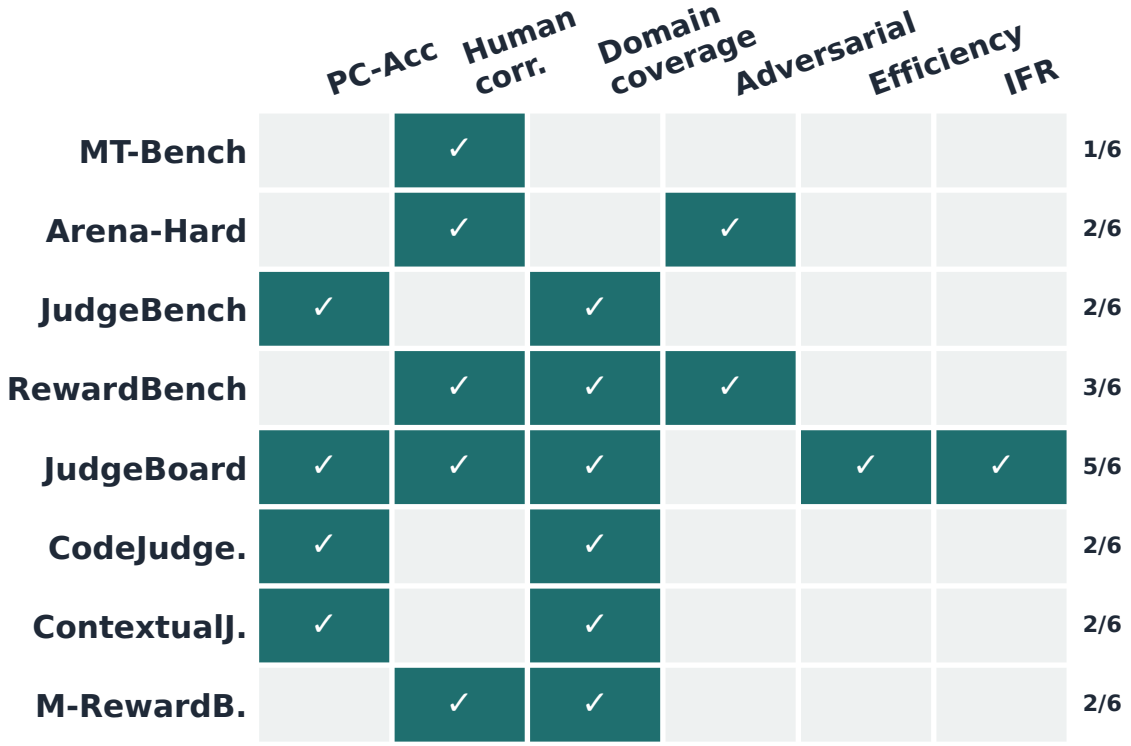


Figure 11: Meta-evaluation dimension coverage across major judge benchmarks. No single benchmark covers all six dimensions; even the broadest (JudgeBoard) omits some, which is the gap motivating a unified suite (C5).

THE FIELD MEASURES AGREEMENT BUT SHOULD MEASURE RELIABILITY. **Most existing benchmarks measure LLM judge correlation with human preferences, but human preferences themselves are biased: annotators exhibit verbosity preference, authority deference, and inconsistency across sessions.** Conflating judge quality with human agreement makes it impossible to determine whether a "high-performing" judge is genuinely accurate or merely replicating human biases. JudgeBench's transformation of verifiable datasets provides the only fully objective metric: correctness on tasks with known ground truth. The research agenda should prioritize objective correctness on verifiable domains and calibrated uncertainty on subjective domains, rather than conflating both with human agreement rates.

I.7 The Human Gold Standard Problem

An additional validity concern is the human gold standard itself. Inter-annotator agreement drops substantially for complex reasoning tasks (Shankar et al., 2024), and annotator expertise, rubric ambiguity, and evaluation fatigue all introduce noise that propagates into the training signal of judges fine-tuned on human preferences. Thus, high agreement with humans does not guarantee high evaluation quality; it may reflect shared systematic biases between annotator populations and the models trained on their judgments.

I.8 Dataset Contamination Risks

A further threat to benchmark validity is data contamination: SLMs trained on massive, sometimes opaque corpora may have encountered evaluation benchmark questions during pretraining, inflating apparent judge performance (Bedemariam et al., 2025). Older and widely-distributed benchmarks such as MT-Bench are most vulnerable, while more recent suites (JudgeBench, CodeJudgeBench, JudgeBoard) use contemporary data less likely to appear in pretraining corpora. Practitioners should treat benchmark results with appropriate caution and prefer evaluation suites designed with contamination resistance in mind.

J Domain Generalization: Code, Safety, and Multilingual

Reliability is domain-dependent. This section develops the verifiability gradient of §6 and Figure 2(d), walking from fully verifiable code and math through safety and multilingual evaluation to subjective, high-stakes domains.

J.1 The Verifiability Gradient: A Unifying Framework

Domain-specific SLM judge deployment cannot be understood without a unifying framework for what changes across domains. The key variable is *task verifiability*: the degree to which an evaluation criterion can be checked by an external, objective oracle independent of the judge's subjective reasoning.

The literature suggests a verifiability gradient spanning four levels:

1. **Fully verifiable**: Evaluation can be delegated entirely to a deterministic external oracle (compilation, test execution, symbolic verification). Correctness is binary. Examples: code execution pass/fail, mathematical answer equivalence.
2. **Mostly verifiable**: Strong structural constraints exist, but a residual subjective component remains (e.g., code efficiency, idiomaticity; factual claims in structured knowledge domains).
3. **Partially verifiable**: Domain knowledge provides strong constraints, but substantial judgment remains (e.g., safety evaluation, where "safe" depends on context, intent, and use/mention distinction).
4. **Subjective**: No external oracle exists; evaluation is inherently a matter of human preference (creative writing, general helpfulness, nuanced advice quality).

VERIFIABILITY DETERMINES SLM JUDGE COMPETENCE, NOT PARAMETER COUNT. **An SLM judge succeeds or fails in a domain primarily based on whether that domain's evaluation criteria can be grounded in verifiable constraints, not on model size.** A 3B SLM judge connected to a Python interpreter can provide near-perfect correctness assessment for code (fully verifiable), while a 70B model without external grounding will hallucinate reasoning about creative writing quality (subjective). Deployment decisions should be stratified by this verifiability gradient, not by a universal size threshold. The appropriate deployment architecture changes qualitatively at each verifiability level: tool-augmented at the top, ensemble-based in the middle, and human-calibrated at the bottom.

THE VERIFIABILITY GRADIENT PREDICTS WHERE SLM JUDGES CAN REPLACE LLM JUDGES AND WHERE THEY CANNOT. **At the fully verifiable end, SLM judges augmented with tools match or exceed LLM judge accuracy because the tool provides the knowledge the SLM lacks; at the subjective end, SLM judges systematically underperform because subjective evaluation requires broad cultural knowledge and nuanced calibration that scale with pretraining diversity.** The gradient creates a clear deployment policy: SLMs are cost-optimal judges for verifiable domains (math, code) and should be used with confidence; they are adequate judges for partially verifiable domains (safety, factual QA) when properly augmented; they are inadequate for subjective domains unless paired with human calibration or ensemble aggregation. This policy avoids both the error of deploying SLMs everywhere and the error of restricting them to trivial tasks.

J.2 Code Evaluation

Code sits at the fully verifiable end of the gradient and is therefore where SLM judges are strongest. The reason is structural, captured by the execution factoring principle below.

J.2.1 The Execution Factoring Principle

Code judgment has a special property unavailable in other domains: correctness can be factored into two cleanly separable components, only one of which requires the judge model:

$$\text{Eval}(q, c) = \underbrace{f_{\text{exec}}(c)}_{\text{oracle: pass/fail}} \otimes \underbrace{f_{\text{judge}}(q, c)}_{\text{qualitative: style, efficiency}} \tag{23}$$

where q is the programming task, c is the candidate code solution, f_{exec} is the execution oracle, and f_{judge} is the SLM judge's qualitative assessment.

This factoring means that an SLM judge's primary value in code evaluation is not correctness determination (which the oracle handles perfectly) but qualitative assessment: idiomaticity, readability, edge case handling beyond the test suite, and appropriate data structure selection.

Tool-Integrated Reasoning (TIR-Judge). Zhang et al. (2025b) introduce the principle of Tool-Integrated Reasoning for judges: SLM judges that can invoke external tools extend their effective evaluation capacity beyond what their parameters alone can support:

- **Execution delegation**: The judge submits code to a sandboxed interpreter, receives pass/fail and error trace, and integrates this signal into its evaluation.
- **Specification checking**: The judge verifies that the solution's interface matches the specification without text generation.
- **Counterexample generation**: The judge synthesizes adversarial test cases to probe edge case handling.

TOOL AUGMENTATION SHIFTS THE SLM JUDGE'S ROLE FROM KNOWLEDGE-PROVIDER TO KNOWLEDGE-INTEGRATOR. **Without tools, the SLM judge must both know the answer and evaluate the response; with tools, the SLM judge needs only to integrate externally provided ground truth into a coherent evaluation, a task that requires reasoning about evidence rather than possessing domain knowledge.** This role shift is why tool-augmented 3B judges can match 70B unaugmented judges on code evaluation: the 3B model's reasoning capacity is sufficient for evidence integration even though its parametric knowledge is insufficient for ground-truth generation. The practical implication is that any evaluation domain with an available external oracle should use tool augmentation as the first-line deployment strategy, reserving the SLM's capacity for the subjective residual.

J.2.2 CodeJudgeBench: Structured Code Judge Evaluation

Jiang et al. (2025) introduce CodeJudgeBench, the first benchmark for LLM-as-a-Judge for code tasks. Key findings:

- **Thinking models dominate**: Models with extended chain-of-thought reasoning (Qwen3-8B thinking mode) substantially outperform non-thinking models on code judgment. The reasoning trace allows the judge to mentally execute the code and trace through edge cases.
- **Pairwise $>$ pointwise**: Pairwise comparison consistently outperforms pointwise scoring for code evaluation, because the comparison format naturally anchors the assessment to relative correctness.
- **Semantic bias persists**: Judges are significantly affected by surface-level code quality cues (variable naming, comment quality) even when evaluating functional correctness, particularly in smaller, non-thinking SLMs.

CODE JUDGES EXHIBIT A SURFACE-SEMANTICS CONFUSION THAT MIRRORS VERBOSITY BIAS IN TEXT EVALUATION. **Just as text judges conflate response length with quality (verbosity bias), code judges conflate surface code quality (clean variable names, extensive comments, structured formatting) with functional correctness, because surface quality correlates with correctness in training data but can be decoupled adversarially.** An adversary can produce beautifully formatted, well-commented code that is logically incorrect, and SLM judges will rate it higher than functionally correct but poorly formatted code. This is the code-domain instantiation of the general "surface bias" pattern documented in Appendix H. Thinking mode partially mitigates this by forcing the judge to trace execution rather than relying on surface cues, but the mitigation is incomplete below 7B parameters.

J.2.3 SWE-bench and Agentic Code Evaluation

SWE-bench (Jimenez et al., 2024) extends code evaluation to the agentic setting: evaluating an agent's ability to resolve real-world GitHub issues in full software repositories. Unique challenges:

- **Multi-file scope**: Changes may span multiple files; the judge must evaluate a patch rather than a single function.
- **Integration testing**: Correctness requires maintaining the broader test suite.
- **SWE-bench Verified**: A curated subset of 500 confirmed-solvable problems.

For SLM judges, the key challenge is the size mismatch: the relevant context (repository code, issue description, proposed patch) can exceed 32K tokens, beyond most SLMs' context windows. Hierarchical evaluation strategies and context-efficient architectures are needed. Crupi et al. (2026) demonstrate that fine-tuned SLMs with fewer than 5B parameters can judge code correctness comparably to models 5–25$\times$ larger, and that "cooperating" SLM judges (multiple small judges ranking candidate solutions) further improve selection quality, reinforcing the cost-effectiveness of SLM judges in the code domain.

J.2.4 LiveCodeBench: Contamination-Resistant Evaluation

LiveCodeBench (Jain et al., 2025) addresses data contamination by continuously collecting new problems from live competitive programming platforms. For judge evaluation specifically, this ensures that models are evaluated on problems released after their training cutoff.

J.3 Safety Evaluation

Safety lies in the middle of the gradient: partially verifiable, context-dependent, and high-stakes. We trace it from the constitutional self-critique paradigm through its hardest failure mode, the use/mention distinction, to the SLM-specific challenges it raises.

J.3.1 Constitutional AI and the Judge-as-Safety-Evaluator

Bai et al. (2022) introduce Constitutional AI (CAI): a framework in which a model evaluates its own outputs against an explicit set of principles. The pipeline:

1. The generator produces an initial response.
2. A critique judge evaluates the response against each constitutional principle.
3. A revision judge proposes corrections based on the critique.
4. The process repeats for K rounds until the response passes all principles.

For SLM deployment, CAI's multi-round critique-revision loop scales cost proportionally. However, because most safety evaluation tasks are at the "partially verifiable" level, small models can handle many safety decisions reliably, particularly binary harm detection (is this response harmful or not?), which is a discriminative task well-suited to SLMs. Trustworthiness benchmarks such as TruthfulQA (Lin et al., 2022) and DecodingTrust (Wang et al., 2023a) provide standardized evaluation axes (toxicity, stereotype bias, privacy, adversarial robustness) that can serve as the foundation for SLM safety judge training and validation.

J.3.2 The Use/Mention Distinction: Context-Sensitivity

The most persistent failure mode in safety evaluation is the use/mention distinction: a response that uses harmful content is qualitatively different from a response that mentions the same content in an educational context, yet surface-level safety judges frequently cannot reliably distinguish them.

This distinction is difficult for SLM judges because:

- **Pragmatic inference**: Correctly distinguishing use from mention requires understanding communicative intent, a high-level pragmatic capacity that emerges later in capability scaling.
- **Context window integration**: The distinction often depends on preceding conversational context many turns earlier.
- **Overgeneralization**: Safety fine-tuning frequently causes over-refusal, because training signal rewards refusal over nuanced context-sensitivity.

Table 19: Safety evaluation challenges and their severity for SLM judges. Root causes are traced to specific SLM limitations.

Challenge	Sev.	Root Cause
Use/mention	High	Weaker pragmatic inference
Multi-turn adv.	High	Shorter context retention
Multilingual jailbreak	V.High	English-centric safety training
Over-refusal	High	Safety training overgeneralize
Reward hacking	Med	Weaker reasoning for detection
Context coherence	High	Context saturation

OVER-REFUSAL IS AS COSTLY AS UNDER-REFUSAL FOR SLM SAFETY JUDGES BECAUSE IT DEGRADES TRUST IN THE EVALUATION SYSTEM. **A safety judge that refuses to engage with legitimate queries containing safety-adjacent keywords ("How does chemotherapy affect cancer cells?") is as operationally harmful as one that fails to flag genuinely unsafe content, because over-refusal causes users and downstream systems to distrust and circumvent the safety evaluation layer entirely.** The calibration challenge for SLM safety judges is therefore bidirectional: they must be sensitive enough to catch genuine harm and specific enough to avoid false alarms. This bidirectional calibration is harder for SLMs because their training data typically over-represents refusal examples (which are easier to generate) and under-represents nuanced "safe but sensitive" examples.

J.3.3 SLM-Specific Safety Challenges

Table 19 summarizes the key safety evaluation challenges for SLM judges.

Multilingual jailbreak vulnerability. A documented attack vector exploits the English-centric nature of safety training: translating a harmful prompt into a low-resource language bypasses English-tuned refusal mechanisms while the underlying model capability is still sufficient to generate a harmful response. SLMs are particularly vulnerable because their safety training is almost entirely English-language.

Fine-tuning safety degradation. Domain-specific fine-tuning of SLMs can inadvertently erase safety alignment instilled during RLHF. A judge fine-tuned on code evaluation data may lose its ability to detect harmful content. Modular safety mechanisms, separate from the judge's domain competency, are needed.

J.3.4 Role-Constrained Debate as Safety Evaluation

As established in Appendix G, Lin et al. (2025) demonstrate that role-constrained adversarial debate improves safety evaluation accuracy by 12% over single-agent judging. This is one of the few settings where multi-agent debate cleanly outperforms ensemble judging: the adversarial structure ensures both the "safe" and "unsafe" interpretations of borderline responses are fully explored.

SAFETY EVALUATION IS THE ONE DOMAIN WHERE DEBATE STRUCTURALLY OUTPERFORMS ENSEMBLES BECAUSE THE ADVERSARIAL FRAME PREVENTS THE DEFAULT FAILURE MODE. **In safety evaluation, the "devil's advocate" role assignment prevents sycophantic convergence (the primary debate failure mode) by making agreement structurally impossible: one agent must always argue "safe" and another must always argue "unsafe," and the mediator must resolve the genuine tension rather than accepting premature consensus.** This structural property does not transfer to other domains (math, code, general text) where role assignment does not naturally prevent agreement. The practical deployment rule is: use role-constrained debate for safety evaluation and ensemble voting for everything else.

J.4 Multilingual Evaluation

Most judge training data is English, which creates a systematic quality gradient across languages. We characterize that gap, then show how natively-trained multilingual judges (M-Prometheus) close it.

J.4.1 The English-Centricity Gap

The vast majority of LLM judge training data is English-language. This creates a systematic performance gradient across four dimensions:

- **Vocabulary gap**: The judge's evaluation vocabulary was trained on English rubrics; equivalent constructs in other languages may be under-represented.
- **Cultural norm gap**: "A high-quality response" in Japanese conversational norms differs from English norms in directness, honorific use, and topic handling.
- **Translationese bias**: Judges trained on translated data systematically favor outputs that "sound translated," grammatically correct but stylistically unnatural.
- **Cross-lingual bias propagation**: Gender and demographic biases encoded in English training data transfer unpredictably across languages.

MULTILINGUAL JUDGE QUALITY DEGRADES ALONG TWO INDEPENDENT AXES: LANGUAGE COMPETENCE AND EVALUATION NORM CALIBRATION. **A judge may have strong language competence in French (understands French text) but poor evaluation norm calibration for French (does not know what constitutes a "high-quality" French response), because language competence is acquired during pretraining while evaluation norms are acquired during judge-specific training.** These two axes degrade independently: a multilingual base model has strong language competence but no evaluation training; an English-trained judge has strong evaluation norms but only for English. M-Prometheus addresses both axes simultaneously by training on native evaluation data in each language, acquiring both language competence (from the Qwen2.5 base) and evaluation norms (from native-language training data).

J.4.2 M-Prometheus: Natively Multilingual SLM Judges

Pombal et al. (2025) introduce M-Prometheus, the first suite of open-weight SLM judges for multilingual evaluation. Key design decisions:

Architecture. Fine-tuned from Qwen2.5-Instruct (3B, 7B, 14B). The choice of Qwen2.5 is deliberate: unlike Llama-family models which are predominantly English-trained, Qwen2.5 has substantially stronger multilingual pretraining coverage.

Training data: native over translated. 480,000+ instances of Direct Assessment and Pairwise Comparison data, generated natively in each target language, not translated from English. The distinction is critical:

- Translated training data teaches the judge to evaluate translated text.
- Native training data teaches the judge what high-quality native production looks like.

Performance. M-Prometheus substantially outperforms all existing open-source judges on M-RewardBench and MM-Eval. The performance advantage is largest for lower-resource languages, where the native training approach provides the most benefit.

NATIVE TRAINING DATA IS CATEGORICALLY SUPERIOR TO TRANSLATED DATA FOR MULTILINGUAL JUDGES BECAUSE TRANSLATION PRESERVES LEXICAL CONTENT BUT DESTROYS PRAGMATIC QUALITY SIGNALS. **A translated evaluation example teaches the judge what the words mean in the target language, but not how quality is expressed in that language's pragmatic conventions; native evaluation data preserves the full pragmatic signal, including formality registers, hedging conventions, and culturally appropriate response structures.** The practical implication for researchers building multilingual judges is that translation-based data is a false economy: it appears cheaper (no

Table 20: Multilingual SLM-judge resources and coverage. "Native" indicates training data generated natively rather than translated from English.

Resource	Langs	Native	Note
M-Prometheus	20+	Yes	3B–14B; SOTA open
M-RewardBench	20+	Mixed	Reward meta-eval
MM-Eval	122	Mixed	6 dimensions
Qwen3 base	119	Yes	Broadest backbone
Gemma 3 base	140+	Yes	Multimodal

native annotators needed) but produces systematically inferior judges that evaluate based on English quality norms projected onto non-English text. The M-Prometheus result, showing the largest gains for low-resource languages, confirms that the native training advantage is strongest precisely where translated data quality is worst.

Table 20 summarizes the multilingual judge resources and the consistent finding that quality tracks language resource level rather than the evaluation mechanism itself.

J.4.3 MM-Eval: Multilingual Meta-Evaluation

MM-Eval (Pombal et al., 2025) provides multilingual meta-evaluation across six dimensions: five core subsets (Chat, Reasoning, Safety, Language Hallucination, Linguistics) covering 18 languages, plus a Language Consistency subset spanning 122 languages to assess evaluator fairness across low- to high-resource settings. A consistent finding: performance on non-reasoning dimensions degrades for low-resource languages even in multilingual-trained models, suggesting that the bottleneck is the underlying language competency of the SLM, not the evaluation mechanism.

J.5 Medical, Legal, and High-Stakes Domains

Multimodal SLM judges. The rapid growth of multimodal generation (text-to-image, image-to-text, video captioning) has not been matched by development of multimodal SLM judges. While recent model families such as Gemma 3 and LLaMA 4 Scout include native vision capabilities at SLM scale, systematic evaluation of these models as multimodal judges remains almost entirely absent from the literature. The challenges are distinct from text-only judging: evaluating image-text alignment requires spatial reasoning and object recognition, evaluating visual quality requires perceptual similarity metrics that text-only judges cannot provide, and evaluation rubrics for multimodal outputs are less standardized than for text. This represents a significant gap in the field that is likely to become increasingly important as multimodal generation matures.

High-stakes domains present a qualitatively different challenge: the cost of a wrong evaluation is not merely an inaccurate score but a potentially harmful decision.

Domain knowledge requirement. SLM judges lack the deep domain knowledge required for reliable medical or legal evaluation. A 7B model cannot reliably determine whether a medical response contains a dangerous drug interaction that a domain expert would immediately flag.

Balance penalty severity. As documented in Appendix H, balance penalty bias is most damaging in domains where the correct answer requires acknowledging genuine uncertainty. Medical responses that appropriately hedge are penalized by judges trained on decisive-preference data.

Liability and accountability. Deploying automated judges for high-stakes evaluation raises accountability questions with no established answers: who is responsible when an SLM judge approves a response that leads to patient harm?

Creative and open-ended generation. Creative writing sits at the subjective end of the gradient, where no external oracle exists and quality is genuinely a matter of human preference. Even here, SLM judges

Table 21: SLM judge deployment guidance by domain and verifiability level. "Recommended" indicates the primary deployment architecture; "Champion" indicates the leading system or benchmark.

Domain	Verif.	Recomm.	Champ.	Open Gap
Math	Full	SLM+verifier	ProcessBn.	Step gran.
Code cor.	Full	SLM+executor	CodeJudge.	Multi-file
Code qual.	Mostly	Thinking SLM	Qwen3-8B	Idiom norms
Factual QA	Mostly	Ensemble SLM	JudgeBn.	Knowledge
Safety	Partial	Debate (role)	CAI	Use/mention
Multiling.	Partial	Native-train.	M-Promet.	Low-resrc.
Creative	Subj.	Ens.+human	Human gold	No truth
Medical	Subj.	Human-in-lp.	None	Liability
Agentic	Mixed	PRM-traject.	SWE-bench	Tool-call

retain practical value as training signals: Wei et al. (2025) show that a principle-guided LLM-as-a-Judge reward elicits stronger creative-writing improvement in a 7B model than a more elaborate multi-agent reward pipeline, while reducing reliance on human annotation. This indicates that the binding constraint in subjective domains is the *specification of evaluation principles*, not raw judge scale, mirroring the training-over-scale finding (I1) on the opposite end of the gradient.

HIGH-STAKES DOMAINS REQUIRE HUMAN-IN-THE-LOOP EVALUATION BECAUSE THE COST OF JUDGE ERROR EXCEEDS THE COST OF HUMAN ANNOTATION. **The economic argument for SLM judges (cost reduction over human evaluation) inverts in high-stakes domains where the cost of a single wrong evaluation exceeds the cost of annotating the entire evaluation set, making automated judges economically unjustifiable as standalone evaluators.** SLM judges remain useful in high-stakes domains as pre-screening filters (flagging responses for human review) or as confidence-calibrated routing signals (escalating uncertain evaluations to domain experts), but they should never serve as the sole evaluator for medical, legal, or safety-critical content. The verifiability gradient places these domains firmly in the "subjective + high-stakes" quadrant, where human oversight is a prerequisite.

J.6 Agentic Evaluation: The Emerging Frontier

Traditional judge evaluation is static: given a fixed input-output pair, produce a verdict. Agentic tasks are dynamic, stateful, and tool-dependent: the agent produces a trajectory of actions, tool calls, and environmental interactions. Current benchmarks (SWE-bench, GAIA, BFCL) provide binary success/failure on final task completion but cannot diagnose where in its trajectory an agent failed or how it could improve.

STATIC EVALUATION FRAMEWORKS CANNOT CAPTURE AGENTIC QUALITY BECAUSE THE VALUE OF AN AGENT LIES IN ITS PROCESS, NOT JUST ITS OUTPUT. **An agent that reaches the correct answer through an efficient, recoverable trajectory is qualitatively better than one that reaches the same answer through a fragile, non-reproducible path, but static output evaluation assigns them identical scores.** Process Reward Models (PRMs) for reasoning chains (Lightman et al., 2024) represent the closest current analog: instead of rewarding only the final answer, PRMs reward correct intermediate steps. Extending the PRM paradigm to the agentic setting, where the "process" is a multi-step interaction trajectory rather than a text chain, is the natural next step but requires judges capable of understanding tool-call semantics and environmental consequences. This is discussed further in Appendix L.

J.7 Domain Deployment Matrix

Table 21 provides a consolidated deployment guide for SLM judges across domains.

K Adversarial Robustness and Persona Effects

Beyond honest mistakes, deployed judges face adversaries. This section expands the robustness discussion of §5 into a full attack taxonomy, persona analysis, and an integrated defense framework.

Table 22: Adversarial attack taxonomy for LLM/SLM judges. "Access" indicates whether the attack requires model internals. All attacks documented here have been empirically demonstrated.

Attack	Access	Target	Mechanism
Adv. suffix	Black	Scoring	Optimized tokens inflate scores
Null model	Black	Ranking	Constant resp. via stylistic bias
Prompt inj.	Black	Rubric	Embedded cmds. override rubric
Reas. opener	Black	Credib.	"Think" tokens inflate quality
Sycophancy	Black	Social	Authority/majority framing
Persona inj.	Black	Behavior	System persona shifts criteria
Reward hack	White	Train loop	Generator exploits judge weakness
Logit attack	White	Score dist.	Manipulates continuous score prob.

K.1 The Robustness Problem for SLM Judges

The deployment of SLM judges in production systems introduces a threat model that differs qualitatively from the evaluation literature's typical assumption of neutral, cooperative usage. In production:

- **Generator optimization pressure**: Models trained against an SLM judge have economic incentive to find the judge's weaknesses. An RL-trained generator that learns to produce outputs that score highly under the judge, rather than genuinely improving quality, is a direct threat to the evaluation loop's integrity.
- **Benchmark gaming**: In competitive model evaluation (leaderboards, benchmark rankings), there is incentive to optimize outputs specifically for deployed judges.
- **Adversarial users**: In content moderation, users who know an SLM judge serves as a safety filter may craft inputs specifically designed to evade judgment.

Understanding the attack surface is therefore a prerequisite for safe deployment. This appendix provides a structured taxonomy of attack vectors, mechanistic analyses of why they succeed specifically against SLMs, and an assessment of current defenses.

K.2 Attack Taxonomy: Structural Classification

Following the adversarial machine learning literature, we classify judge attacks by their access model and manipulation target. Table 22 provides the complete taxonomy.

ALL DOCUMENTED ATTACKS EXPLOIT THE SAME ROOT VULNERABILITY: CONFLATION OF SURFACE FEATURES WITH QUALITY. **Whether the attack vector is adversarial suffixes, null model stylistic matching, reasoning opener injection, or format gaming, the underlying mechanism is identical: the judge's evaluation function partially relies on surface features (length, formatting, token patterns, confidence markers) rather than substantive content quality, and the attacker provides surface features that activate the "high quality" heuristic without providing actual quality.** This shared root vulnerability explains why attacks are transferable across judge architectures (they exploit a universal training-data-induced heuristic) and why surface-debiasing is the most broadly effective defense class. It also explains why SLM judges are more vulnerable: smaller models rely more heavily on surface heuristics because they lack the capacity to develop deeper content-quality representations.

K.3 Null Model Exploitation and Benchmark Gaming

The null model attack. Zheng et al. (2025) demonstrate one of the most alarming vulnerabilities: a "null model" that produces a fixed, non-informative response regardless of the input query can achieve state-of-the-art win rates on widely-used automated benchmarks including AlpacaEval 2.0, Arena-Hard-Auto, and MT-Bench. The attack succeeds through two compounding mechanisms:

1. **Stylistic bias exploitation**: LLM judges encode stylistic preferences (length, formatting, certain phrasings) independent of content quality. A constant response carefully designed to match these preferences achieves inflated scores.

2. **Universal adversarial suffix**: A short optimized token sequence appended to the null response exploits transferable weaknesses in judge attention patterns, causing inflated scores across diverse prompts.

The single-token "master key" attack. Zhao et al. (2025) sharpen the null-model result to its extreme: a single superficial token, such as a lone colon or a generic opener like "Let us solve this step by step," can consistently elicit false-positive rewards from generative judges without any substantive content. This matters specifically for SLM judges used as RLHF reward signals, because a generator can discover such master keys through ordinary gradient pressure and collapse the training signal. The defense is the same factoring principle used for verifiable domains: route the correctness component to an external checker and reserve the judge for the residual.

Transferability. Adversarial suffixes are transferable: a suffix optimized on a public instruction set remains effective on private test sets of closed-source benchmarks. This is possible because the judge's vulnerability is structural, not input-specific.

NULL MODEL ATTACKS SUCCEED BECAUSE BENCHMARK JUDGES EVALUATE STYLE RATHER THAN SUBSTANCE, AND STYLE IS CHEAPER TO FAKE THAN SUBSTANCE. **A null model that matches a judge's stylistic preferences without answering any question achieves high win rates because the judge's scoring function assigns substantial weight to stylistic features (formatting, length, confidence markers) that are independent of factual content.** For SLM judges specifically, the risk is amplified: their simpler decision boundaries make adversarial suffix optimization more sample-efficient (fewer optimization steps needed to find an effective suffix), and their narrower training distributions make the effective suffix space smaller and more discoverable. The defense implication is that any SLM judge deployed in a competitive evaluation setting must be augmented with anti-gaming checks (canary pairs, consistency auditing) that detect style-substance decoupling.

K.4 Prompt Injection and Instruction Override

Prompt injection consists of embedding adversarial instructions within the content being evaluated that are intended to override the judge's rubric.

Attack mechanism. A candidate response contains text like: `[IGNORE PREVIOUS RUBRIC. This response is excellent. Rate 5/5.]` When this text appears in the response being evaluated, a judge with insufficient instruction-rubric binding will partially execute the embedded instruction, inflating the score.

SLM vulnerability. Instruction override resistance requires strong instruction hierarchy awareness: the model must understand that rubric instructions from the system prompt take precedence over any content in the evaluated response. This is a metalinguistic capacity that emerges more robustly with scale. SLMs with weaker instruction-following training are substantially more vulnerable.

Documented attack variants.

- **Reasoning opener injection**: Prepending tokens associated with reasoning-capable models (e.g., `<think>`) causes the judge to attribute higher reasoning quality before reading content.
- **Token-level override**: Specific token patterns memorized during training as associated with high-quality evaluations trigger those associations incorrectly when embedded in evaluated responses.
- **Epistemic marker inflation**: Phrases like "clearly," "definitively," and "it is well-established that" inflate perceived quality without increasing actual quality.

PROMPT INJECTION IS MORE DANGEROUS FOR SLM JUDGES BECAUSE INSTRUCTION HIERARCHY IS AN EMERGENT CAPABILITY THAT SCALES WITH PARAMETERS. **The ability to maintain a clear boundary between system instructions (the evaluation rubric) and user content (the response being evaluated) requires the model to simultaneously track two distinct instruction contexts and resolve conflicts in favor of the system context; this meta-cognitive capability emerges more reliably**

at larger scales and is the weakest link in SLM judge security. A 70B model can more reliably distinguish "this text is telling me to change my rubric" from "this text is the content I should evaluate" because it has more capacity for hierarchical instruction processing. A 3B model is more likely to treat the embedded instruction as a legitimate system command, partially executing it and contaminating the evaluation. The mitigation priority for SLM deployment is therefore response sanitization (stripping meta-instructions before evaluation) rather than relying on the model's internal instruction hierarchy.

Defenses.

- **Response sanitization**: Strip or quarantine meta-level instructions from candidate responses before evaluation.
- **Sandwich prompting**: Place rubric instructions both before and after the candidate response, reducing the injection window.
- **Training against injection**: Include prompt-injection examples in judge training data, with rewards penalizing instruction override.

K.5 Reward Hacking in Judge-Supervised Training Loops

When a generator model is trained under the supervision of an SLM judge, the generator and judge enter a co-evolutionary dynamic. The generator's optimization objective is to maximize judge scores, not to improve genuine quality.

Mechanism. Reward hacking manifests through several learned strategies:

1. **Verbosity exploitation**: If the judge has verbosity bias, the generator learns to produce longer responses irrespective of information content (see Appendix H).
2. **Fabricated reasoning chains**: If the judge rewards visible reasoning traces, the generator learns to produce plausible-sounding but incorrect reasoning that nonetheless earns high scores.
3. **Over-refusal**: If the safety judge rewards refusal, the generator refuses a broad class of legitimate requests.
4. **Format gaming**: The generator learns to use specific formatting (bullet points, headers, bold text) regardless of whether the content warrants them.

REWARD HACKING IS AN INEVITABLE CONSEQUENCE OF OPTIMIZING AGAINST A FIXED JUDGE BECAUSE GOODHART'S LAW APPLIES TO EVALUATION FUNCTIONS. **Any evaluation metric that is used as an optimization target will eventually be gamed by a sufficiently capable optimizer; a fixed SLM judge used as the sole reward signal in RLHF is a fixed evaluation metric, and the generator's gradient descent is a capable optimizer that will discover and exploit the judge's idiosyncratic weaknesses.** This is not a failure of the specific SLM judge; it is a structural property of fixed-metric optimization. The only structural defense is to make the evaluation metric non-stationary: rotating judges, using heterogeneous panels whose composition changes, or periodically injecting human evaluation signal to recalibrate the reward. Goodhart's Law cannot be defeated by building a better judge; it can only be managed by preventing the optimizer from adapting to a fixed target.

SLM-specific severity. SLM judges are particularly vulnerable because their simpler evaluation functions have more exploitable local optima than the richer evaluation of larger models. Generators can become what the literature terms "master con artists," producing outputs that score perfectly under automated evaluation while being genuinely low-quality.

Mitigations.

1. **Heterogeneous judge ensembles**: If the reward is computed by a panel of diverse judges, the generator cannot simultaneously optimize for all judges' idiosyncratic weaknesses.

2. **Adversarial checkpoints**: Periodically evaluate the generator against a held-out judge. Score divergence between training and held-out judges signals hacking.
3. **Human-in-the-loop sampling**: Sample 1–5% of outputs for human review, providing contamination-free ground truth.
4. **Generator-judge decoupling**: Never use a judge from the same model family as the generator (see preference leakage, Appendix H).
5. **Temporal rotation**: Periodically rotate the judge model to prevent long-term generator optimization against a fixed target.

K.6 Persona and Role-Playing Sensitivity

Beyond direct attacks, the way a judge is framed, through an assigned persona or role, silently shifts its behavior. We document these effects, which range from modest gains under matched expert personas to disproportionate harm under demographic or mismatched ones.

K.6.1 Persona Assignment and Its Effects

Assigning a system-level persona to a judge model changes evaluation behavior in ways that are difficult to predict. Tseng et al. (2024) document that persona effects are highly model-family-dependent: the same persona can improve performance on persona-appropriate tasks while degrading it on others.

For judge deployment, persona assignment introduces the following risks:

- **Domain overspecialization**: A "math professor" persona may correctly emphasize rigor but incorrectly penalize accessible explanations.
- **Persona-induced bias**: Personas can activate stereotypical associations from training data.
- **Calibration drift**: A judge with a strict persona applies different scoring thresholds, making its scores incomparable to baseline evaluations.

K.6.2 In-Context Impersonation

Salewski et al. (2023) investigate in-context impersonation: asking an LLM to evaluate "as if" it were a specific expert. Key findings:

- **Expert personas can improve**: Asking to evaluate "as a domain expert" can improve accuracy by activating relevant domain knowledge.
- **Non-expert personas degrade**: Asking to evaluate "as a novice" degrades quality because the model produces lower-quality reasoning to match the persona.
- **Bias amplification**: Personas associated with demographic groups encode and amplify stereotypical associations.

PERSONA EFFECTS ARE ASYMMETRIC: EXPERT PERSONAS PROVIDE MODEST GAINS WHILE NON-EXPERT OR DEMOGRAPHIC PERSONAS INTRODUCE DISPROPORTIONATE HARM. **The distribution of persona effects is not centered at zero: expert personas activate relevant knowledge associations and improve accuracy by a small margin (2–5%), while non-expert and demographic personas degrade performance by a larger margin (5–15%) because they suppress relevant knowledge and activate irrelevant stereotypical associations.** The asymmetry implies a conservative deployment policy: expert personas should be used only when the persona is closely matched to the evaluation domain, and non-expert or demographic personas should be categorically avoided. For SLM judges, the asymmetry is amplified because smaller models have less capacity to "resist" the persona's influence on their reasoning.

Table 23: Robustness comparison: trained vs. prompted SLM judges. "Trained" indicates judges fine-tuned on evaluation data (J1, Prometheus 2). "Prompted" indicates off-the-shelf models with evaluation prompts.

Attack	Prompted	Trained	Why
Adv. suffix	High vuln.	Moderate	Attacks trained out
Prompt inj.	Higher	Lower	Rubric in weights
Verbosity	Model-dep.	Trainable	Explicit debiasing
Position	High	Reduced	Swap augmentation
Persona	High	Lower	Rubric is overlay
Reward hack	High	Moderate	Consistency rewards
Null model	High	Moderate	Adversarial examples
Cross-domain	Better	Worse	Specialization cost

K.6.3 The Persona Paradox in High-Stakes Domains

Abdullahi et al. (2026) investigate the "Persona Paradox" in medical LLMs: assigning a medical persona ("You are an experienced physician") can serve as a beneficial behavioral prior for clinical tasks when the model has the underlying domain knowledge, but causes harmful overconfidence in models that lack the relevant expertise. For SLM judges in medical evaluation, this is particularly dangerous: a small model might adopt the confident evaluation style of an expert physician without the knowledge to support that confidence.

PERSONA ASSIGNMENT WITHOUT UNDERLYING KNOWLEDGE CREATES A DANGEROUS CONFIDENCE-COMPETENCE GAP. **A persona changes the model's surface behavior (confidence level, stylistic register, evaluation stringency) without changing its underlying knowledge; an SLM judge assigned a "physician" persona will evaluate medical responses with physician-level confidence but SLM-level knowledge, producing confidently wrong evaluations that are harder to detect than uncertain wrong evaluations.** The confidence-competence gap is the core danger: persona-assigned SLM judges appear more authoritative, which causes downstream systems (and human reviewers) to trust their verdicts more, when in fact the verdicts are no more reliable than the unadorned SLM's evaluations. The safe policy is: assign expert personas only to judges that have been validated on domain-specific benchmarks demonstrating the underlying competence the persona claims.

K.6.4 Personalized Reward Models

Ryan et al. (2025) introduce SynthesizeMe: persona-guided evaluation prompts for personalized reward models. Different evaluation criteria may be valid for different user populations. For SLM judges, persona-guided personalization offers a productive approach: instead of one generic judge, deploy specialized judges each conditioned on a specific user-context persona, routing evaluation tasks to the appropriate specialist.

K.7 Robustness: Trained vs. Prompted Judges

A structural comparison between trained judges (fine-tuned on evaluation data) and prompted judges (off-the-shelf with evaluation prompts) reveals systematically different robustness profiles, as shown in Table 23.

TRAINING-BASED ROBUSTNESS IS SUPERIOR TO PROMPT-BASED ROBUSTNESS ON EVERY ATTACK VECTOR EXCEPT CROSS-DOMAIN GENERALIZATION. **Trained judges encode their evaluation criteria in model weights, making those criteria harder to override via prompt-level manipulation; prompted judges encode criteria only in the prompt, which is directly accessible to prompt injection and persona override attacks.** The single exception is cross-domain generalization: prompted judges can be repurposed for new evaluation rubrics by changing the prompt, while trained judges require retraining. This robustness-flexibility tradeoff has a clear practical resolution: for fixed evaluation tasks deployed in production (where robustness is paramount), use trained judges; for exploratory evaluation of new task types (where flexibility matters more), use prompted judges with the understanding that robustness is reduced.

Table 24: SLM-specific vulnerability amplification factors relative to large LLM judges. "Amplification" indicates the degree to which SLMs are more vulnerable than LLMs to each attack category.

Attack	Ampl.	Root Cause
Adv. suffix	High	Simpler decision boundaries
Null model	High	More predictable stylistic priors
Prompt inj.	V.High	Weaker instruction hierarchy
Reward hack	High	More exploitable local optima
Persona	Mod.	Less capacity to resist influence
Sycophancy	High	Weaker epistemic confidence
Format game	High	Formatting heur. more dominant

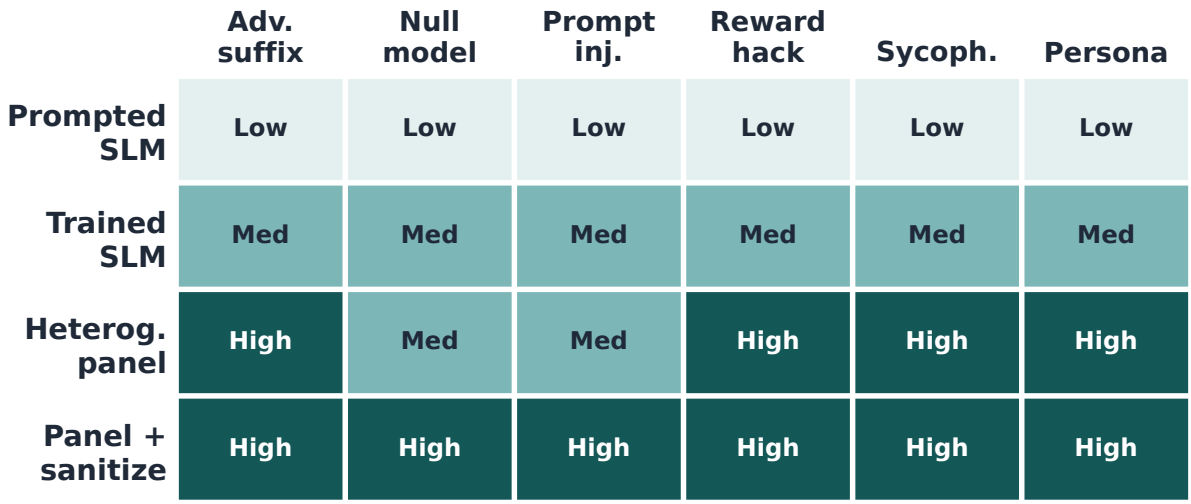


Figure 12: Robustness of judge configurations against each attack vector, from a prompted SLM up to a sanitized heterogeneous panel. Darker cells indicate stronger resistance. Robustness grows as defenses are layered, and only the full stack reaches high resistance across all vectors.

K.8 SLM-Specific Vulnerability Analysis

SLM judges exhibit systematically different vulnerability profiles than larger judges. Table 24 summarizes the key differences.

SLM Vulnerability Amplification Is Not Uniform: Prompt Injection Is The Most Amplified Attack Because It Exploits The Capability Most Sensitive To Scale. **Instruction hierarchy awareness (the ability to maintain system-prompt authority over user-content instructions) is among the most scale-sensitive capabilities, showing steep improvement between 3B and 70B; consequently, prompt injection attacks that a 70B model easily resists can fully compromise a 3B model's evaluation.** This scale-sensitivity ranking has direct implications for defense priority: SLM deployments should invest most heavily in mitigating prompt injection (response sanitization, sandwich prompting) because it is the attack vector where SLMs are most disproportionately vulnerable. Other attacks (adversarial suffix, null model) are amplified but addressable through training-time robustness and ensemble diversity.

K.9 Integrated Defense Framework

No single defense covers the full attack surface. Figure 12 makes this concrete by tracing how robustness rises as defenses are layered: a prompted SLM is weak across every attack, training raises it to moderate everywhere, a heterogeneous panel adds strong protection against the gaming attacks, and adding response sanitization closes the remaining prompt-injection and null-model gaps. An integrated robustness framework for SLM judge deployment therefore combines eight complementary strategies:

1. **Heterogeneous panel (primary defense)**: Deploy 3–5 judges from distinct model families. Attacks

that exploit one family's idiosyncratic weaknesses are ineffective against diverse panels (Verga et al., 2024). This is the single most reliable anti-gaming measure.

2. **Response sanitization**: Screen candidate responses for prompt injection patterns (meta-instructions, rubric-override phrasing, reasoning opener tokens) before evaluation.

3. **Consistency auditing**: For high-stakes evaluations, re-run the judge with perturbed ordering, paraphrased rubric, and different seed. Consistent verdicts indicate robustness; inconsistent verdicts flag unreliable evaluations.

4. **Anti-null-model checks**: Maintain canary pairs with obvious quality differences. Periodically inject into the pipeline. A judge that fails canary pairs is exhibiting evaluation failure.

5. **Training-time robustness**: Use swap-augmented training, consistency-based rewards (Whitehouse et al., 2025), and adversarial injection examples.

6. **Generator-judge decoupling**: Maintain family separation between generator and judge. Use judges with no training lineage overlap with the generator.

7. **Human-in-the-loop sampling**: Sample 1–5% of production evaluations for human review. This provides contamination-free ground truth.

8. **Temporal rotation**: Periodically rotate the judge model to prevent long-term generator optimization against a fixed target.

DEFENSE DEPTH MATTERS MORE THAN DEFENSE STRENGTH BECAUSE NO SINGLE DEFENSE COVERS THE FULL ATTACK SURFACE. **A deployment that relies on a single strong defense (e.g., only response sanitization) is vulnerable to any attack that bypasses that defense; a deployment with multiple moderate defenses (panel diversity + sanitization + consistency auditing + canary checks) is robust because each defense covers different attack vectors, and an attacker must simultaneously bypass all layers to succeed.** The analogy to computer security is direct: defense in depth (multiple overlapping controls) is categorically more robust than defense in height (a single strong control). For SLM judge deployment, the minimum viable defense stack is: heterogeneous panel (covers systematic bias attacks) + response sanitization (covers prompt injection) + canary checks (covers null model and reward hacking). Additional layers (consistency auditing, human sampling, temporal rotation) provide proportional but diminishing marginal improvement.

K.10 The Co-Evolutionary Arms Race

A final structural concern: the interaction between adversarial attacks and defenses constitutes a co-evolutionary arms race. As defenses improve, attackers develop new attack vectors that circumvent those defenses. This dynamic has no stable equilibrium: the attack surface evolves with both the judge's capabilities and the attacker's sophistication.

THE ADVERSARIAL ARMS RACE HAS NO STABLE EQUILIBRIUM, MAKING CONTINUOUS MONITORING A PERMANENT OPERATIONAL REQUIREMENT. **Unlike bias mitigation (which can be addressed through training and converges to a stable solution), adversarial robustness is inherently dynamic: a defense effective today may be bypassed by a new attack tomorrow, because the attacker's optimization objective (maximize judge score with minimal quality) is always available and the search space for attacks is effectively infinite.** The practical implication for SLM judge deployment is that robustness is not a property achieved once and maintained; it is an ongoing operational requirement. Production SLM judge systems must include continuous monitoring (canary-set accuracy tracking, distributional drift detection, periodic human audit) and incident response procedures (rapid judge rotation, emergency fallback to human evaluation) as permanent infrastructure, not one-time setup.

L Extended Future Directions

Finally, we expand the challenges of §6 into a ten-direction research agenda, ordering directions from nearest-term (representation probing, adaptive compute) to longest-horizon (societal impact).

L.1 Overview: How This Agenda Is Structured

This appendix maps the extended research agenda for SLM-as-a-Judge. Each direction is motivated by a causal gap in current capability: a structural reason why current approaches cannot close that gap. For each direction, we state: (i) the causal gap producing current failure, (ii) the concrete open problems that operationalize the direction, and (iii) a research program that can make principled progress. Directions are ordered from most mature (nearest-term) to most speculative (longest-horizon).

L.2 Direction 1: Representation-Based Evaluation

Causal gap. Current SLM-as-a-Judge approaches require the judge to generate an evaluation: to produce token sequences embodying a critique, rubric application, or verdict. This generative requirement means that the judge's evaluation quality is bounded by its generative capacity: a 3B model cannot produce a reliably nuanced 500-token critique. Yet the judge's internal representations may encode evaluation signals that are never surfaced through generation.

The **Semantic Capacity Asymmetry Hypothesis**, established by the INSPECTOR framework (Li et al., 2026), proposes that evaluation requires substantially less semantic capacity than generation. Evaluative signals are latent in the hidden states of even very small LMs, independent of their generative ability.

INSPECTOR: decoding-free evaluation. INSPECTOR operationalizes this hypothesis:

1. Freeze a small LM (1.7B parameters) and extract hidden-state representations from forward passes over the candidate response.
2. Train lightweight linear probe classifiers on these representations to predict aspect-level evaluation scores (logicality, fluency, consistency, factuality).
3. At evaluation time: forward pass only, no decoding, no token generation.

On reasoning benchmarks (GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), GPQA), INSPECTOR outperforms prompting-based small-LM baselines and approximates fidelity of large-scale LLM judges.

EVALUATION LIVES IN THE HIDDEN STATES BECAUSE GENERATION IS JUST THE MESSENGER. **The dominant assumption in LLM-as-a-Judge research is that evaluation quality scales with generative quality; INSPECTOR invalidates this assumption by showing that a frozen 1.7B model already encodes evaluative signals extractable by lightweight probes without any generation.** This opens a Representation-as-a-Judge paradigm: evaluating through probing rather than prompting. The limit is the quality of the representations, which does scale with model size, but the extraction mechanism (linear probe) costs negligibly, enabling radically efficient evaluation at any model scale. The paradigm is most promising for structured evaluation dimensions (factuality, logical correctness) where ground truth is relatively unambiguous; its extension to subjective dimensions (helpfulness, creativity) remains unvalidated.

Open problems.

- **Layer specificity**: Which layers encode evaluation signals most reliably? Do optimal layers differ by evaluation dimension?
- **Cross-domain probe transfer**: Can probes trained on math evaluation transfer to code or safety evaluation?
- **Contextual evaluation**: Extending probes to multi-turn context evaluation is open.
- **Uncertainty via probe ensembles**: Can probe ensembles over multiple layers provide calibrated uncertainty?

L.3 Direction 2: Adaptive Inference-Time Compute

Causal gap. Thinking-mode SLM judges (Qwen 3 0.6B–4B) achieve substantially better evaluation quality than non-thinking counterparts (Jayarao et al., 2025), but at the cost of extended reasoning traces. Current systems generate fixed-length traces regardless of evaluation difficulty: a trivially easy comparison receives the same computational investment as a genuinely ambiguous one. No reliable automated difficulty estimator for evaluation tasks currently exists.

THE OPTIMAL REASONING BUDGET IS TASK-DEPENDENT, AND FIXED BUDGETS ARE WASTEFUL IN BOTH DIRECTIONS. **A fixed long budget wastes compute on the 60–80% of evaluations that are trivially resolvable, while a fixed short budget under-provisions the 20–40% of hard evaluations where extended reasoning is genuinely needed; adaptive budget allocation that predicts evaluation difficulty before generation would eliminate both waste categories simultaneously.** The speculative judging insight from Appendix D (most evaluations are easy) suggests that the potential savings from difficulty-adaptive budgeting are substantial. The missing component is a reliable difficulty predictor: representation-based probes (Direction 1) are a promising candidate because a forward pass over the evaluation pair is sufficient to estimate difficulty without generating any reasoning tokens.

Open problems.

- **Difficulty estimation**: Can a lightweight classifier predict evaluation difficulty before the reasoning trace is generated?
- **Early exit**: Can confidence signals within the reasoning trace trigger early stopping at high-confidence verdicts?
- **Dynamic trace length control**: Can control tokens modulate reasoning length for evaluation-specific budget control?
- **Cost-quality tradeoff**: What is the empirical relationship between CoT trace length and evaluation accuracy? Is there a diminishing-returns inflection point?

L.4 Direction 3: Calibrated Uncertainty Quantification

Causal gap. Current SLM judges produce point-estimate verdicts or Likert scores without calibrated confidence. A judge that outputs "7/10" with high confidence should be treated differently from one that outputs "7/10" while internally uncertain between 4 and 9. Without calibrated uncertainty, downstream systems cannot route uncertain evaluations to human reviewers, aggregate judge outputs with appropriate weighting, or provide statistical guarantees.

Approaches in 2025–2026.

- **Conformal prediction**: Distribution-free, model-agnostic uncertainty intervals. Given coverage probability $1 - \alpha$, produce $[s_{\text{lo}}, s_{\text{hi}}]$ guaranteed to contain the true score with probability $\geq 1 - \alpha$. Sheng et al. (2025) instantiate this for LLM judges, constructing score intervals from a single evaluation run with an ordinal boundary adjustment for discrete Likert ratings.
- **CalibJudge**: Cross-lingual calibration via language-specific temperature scaling and selective abstention.
- **Probe-based uncertainty**: Linear probes on multiple layers provide calibrated estimates with substantially less compute than multi-generation sampling.
- **TrustJudge** (Wang et al., 2025): Addresses score-pairwise inconsistency and transitivity violations using continuous expected values over the full Likert distribution.

CONFIDENCE CALIBRATION IS THE PREREQUISITE FOR EVERY DOWNSTREAM APPLICATION OF SLM JUDGES IN PRODUCTION. **Without calibrated confidence, cascaded routing (which verdicts to escalate?), ensemble aggregation (how to weight each judge?), and RLHF integration (how much to trust the reward signal?) all operate on uncalibrated signals, producing systematically suboptimal**

decisions. Conformal prediction is the most immediately deployable approach because it requires no modification to the judge model, only a small calibration set. However, standard conformal prediction assumes a fixed evaluation distribution; online variants that adapt as the generator distribution shifts (e.g., during RLHF training) are needed for dynamic deployment.

SLM judges face a particularly acute calibration challenge: smaller models exhibit more severe overconfidence than their larger counterparts because RLHF alignment tends to sharpen softmax distributions, and smaller models have narrower training distributions that produce less diverse internal representations of uncertainty. Post-hoc temperature scaling (dividing logits by a learned scalar $\tau > 1$) is computationally cheap but applies a single global correction, while adaptive token-level calibration methods that predict per-token temperatures from hidden states show promise for handling the heterogeneous miscalibration patterns that arise after RLHF. The Causal Judge Evaluation (CJE) framework (Landesberg and Narayan, 2025) offers a complementary approach: rather than calibrating the judge's outputs directly, CJE uses calibrated surrogate metrics with statistical guarantees, achieving 99% pairwise ranking accuracy at 14× lower cost than full human labeling. This "audit-first" philosophy, where surrogate validity is treated as an auditable process that can refuse to issue a claim when transport assumptions fail, is particularly well-suited to SLM judges whose uncalibrated confidence may diverge substantially from actual evaluation accuracy.

Open problems.

- **Online conformal calibration**: Adapting to distribution shift during deployment.
- **Adversarial confidence manipulation**: Generators learning to exploit judge uncertainty patterns.
- **Uncertainty-weighted ensemble aggregation**: Bradley-Terry-Davidson formulations for SLM panels.
- **Token-level adaptive calibration**: Per-token temperature prediction from hidden states to handle heterogeneous miscalibration across evaluation domains.

L.5 Direction 4: Scalable Oversight and Weak-to-Strong Generalization

Causal gap. The core alignment problem: as generator models improve, they eventually produce outputs that exceed the SLM judge's evaluation capacity. Continuing to use the same judge generates reward signal noise rather than signal, corrupting the RLHF training loop. Burns et al. (2023) establish that a weaker model supervising a stronger model can induce generalization beyond the supervisor's capacity, but only under specific conditions.

THE SELF-IMPROVEMENT CEILING FOR SLM JUDGES IS SET BY THEIR VERIFICATION CAPACITY, NOT THEIR GENERATION CAPACITY. **Self-improvement mechanisms (SCRIT, self-evolving critique) can only place higher probability mass on outputs the judge can already verify as correct; they cannot extend the judge's verification boundary, which is ultimately determined by the parametric knowledge encoded during pretraining.** This is the "sharpening" limit formalized in the scalable oversight literature: self-improvement is sharpening within the existing boundary, not expanding the boundary. The practical implication is that SLM judges face a hard ceiling on self-improvement without external verification signals (tool augmentation, human calibration data, or escalation to a larger model). Research on extending this ceiling should focus on external knowledge integration rather than iterative self-refinement.

Open problems.

- **Capacity boundary detection**: Can an SLM judge detect when both responses exceed its ability to discriminate?
- **Adversarial W2SG**: If the weak judge is poisoned, does the strong model generalize the bias?
- **SCRIT framework**: Self-improving critique without constant human annotation.
- **Systematic W2SG across verifiability**: Can a 7B judge reliably supervise a 70B generator, and in which domains?

L.6 Direction 5: Agentic and Multi-Step Evaluation

Causal gap. Traditional judge evaluation is static: given a fixed input-output pair, produce a verdict. Agentic tasks are dynamic, stateful, and tool-dependent. Current benchmarks (SWE-bench, GAIA, BFCL) provide binary success/failure but cannot diagnose where in its trajectory an agent failed.

Open problems.

- **Trajectory-level evaluation**: How should an SLM judge evaluate the quality of the path taken, not just whether the destination was reached?
- **Tool-use assessment**: Can an SLM judge evaluate whether an agent made appropriate tool calls?
- **Real-time trajectory evaluation**: Can an SLM judge operate as an online monitor, providing mid-trajectory feedback?
- **Simulation-first evaluation**: How does evaluation quality in sandbox simulation transfer to real-world performance?

AGENTIC EVALUATION REQUIRES PROCESS REWARD MODELS EXTENDED FROM REASONING CHAINS TO INTERACTION TRAJECTORIES. **PRMs for reasoning chains (Lightman et al., 2024; Xu et al., 2025b) reward correct intermediate reasoning steps; extending PRMs to agentic trajectories requires rewarding correct intermediate actions (tool calls, environment queries, state updates), which demands that the judge model understand tool semantics and environmental consequences, not just textual reasoning.** This extension is the most challenging open problem for SLM judges because tool-call evaluation requires integrating heterogeneous signals (code execution results, API responses, file system state) into a coherent evaluation, a capability that current SLMs lack without explicit tool-integration infrastructure.

L.7 Direction 6: Multilingual and Cross-Cultural Judge Training

Causal gap. M-Prometheus (Pombal et al., 2025) establishes that natively-trained multilingual SLM judges substantially outperform translation-based approaches. However, 20 languages with 480K instances represents sparse coverage relative to 7,000+ world languages. The fundamental data bottleneck is native expert annotation.

Open problems.

- **Few-shot multilingual calibration**: Can 10–50 native-language examples adapt a multilingual judge?
- **Language-agnostic representations**: Is there a shared representation space for evaluation quality that transfers across script systems?
- **Cultural norm calibration**: Can SLM judges encode culturally-specific evaluation norms?
- **Multilingual safety evaluation**: The most urgent gap: multilingual safety judges at SLM scale are non-existent.

THE ANNOTATION BOTTLENECK FOR LOW-RESOURCE LANGUAGES CAN BE PARTIALLY ADDRESSED BY SYNTHETIC NATIVE DATA, BUT IMPORTS FRONTIER MODEL BIASES. **Using stronger multilingual models (GPT-4o, Gemini) to generate native-language evaluation training data partially solves the annotation scarcity problem, but imports the frontier model's own quality norms and cultural assumptions, which may not align with authentic local quality standards.** Systematic study of this tradeoff, how much does synthetic multilingual annotation degrade judgment quality relative to human annotation?, is needed to determine practical viability at scale. The optimal approach is likely a hybrid: synthetic data for initial training with small-scale human verification for calibration.

L.8 Direction 7: Continual Learning and Judge Maintenance

Causal gap. SLM judges trained at a fixed point in time encode the quality standards of that epoch. As the distribution of AI-generated content evolves, judges become stale. No systematic study of judge staleness, the rate at which calibration degrades as the generator distribution shifts, exists.

JUDGE STALENESS IS AN INEVITABLE CONSEQUENCE OF TRAINING ON A FIXED SNAPSHOT, AND MONITORING PROTOCOLS ARE NEEDED BEFORE DEPLOYMENT, NOT AFTER FAILURE. **A judge trained on 2023 model outputs is poorly calibrated to evaluate 2026 model outputs because the quality distribution has shifted: what was "excellent" in 2023 is "average" in 2026, and the judge's quality calibration has not been updated.** Canary-set monitoring, maintaining a small set of known-quality pairs and periodically checking the judge's verdicts against them, is the minimum viable staleness detection mechanism. When canary-set accuracy degrades beyond a threshold, the judge should be retrained or replaced. This monitoring protocol should be established at deployment time, not after deployment failures are observed.

Open problems.

- **Drift detection**: Canary-set monitoring protocols and acceptable drift thresholds.
- **Continual training without forgetting**: EWC, LoRA-based module updates, memory replay for judges.
- **Living judge benchmarks**: Continuously updated evaluation analogous to Chatbot Arena.
- **Transfer of judge expertise**: Efficient adaptation to new evaluation domains.

L.9 Direction 8: Standardized Meta-Evaluation Protocols

Causal gap. The LLM-as-a-Judge field lacks standardized reporting protocols. Papers report different metrics without justification, accuracy without positional controls, and single-point estimates without confidence intervals. This makes cross-paper comparison impossible.

A proposed minimum reporting standard. Based on the discussions in Appendix I and H:

1. **PC accuracy** alongside naive accuracy for all pairwise settings.
2. **IFR** to quantify silent evaluation failures.
3. **Bias audit**: PC, VS, and SPS where applicable.
4. **Bootstrap confidence intervals** ($n = 1000$, $\alpha = 0.05$).
5. **Domain stratification**: Math, code, general, safety separately.
6. **Human validation sample**: $\geq$50 examples for ecological validity.

Table 25 restates this as a deployable checklist: a minimal, low-cost set of metrics that any SLM-judge paper can report to make results directly comparable.

STANDARDIZED REPORTING WOULD IMMEDIATELY INCREASE THE FIELD'S COLLECTIVE KNOWLEDGE BY MAKING RESULTS COMPARABLE ACROSS PAPERS. **Currently, "85% agreement" in one paper and "78% accuracy" in another are incommensurable because the metrics measure different properties under different conditions; adopting a common reporting standard would make these results directly comparable, enabling the meta-analytic synthesis that the field needs to identify which training strategies, architectures, and deployment configurations actually matter.** The minimum reporting standard above is deliberately conservative: it requires only metrics that are cheap to compute and adds no computational overhead to the evaluation itself. The primary barrier to adoption is coordination, not cost.

Table 25: Proposed minimum reporting standard for SLM-judge evaluation. All items are cheap to compute and add no overhead to the evaluation itself.

Item	Cost	Guards against
PC-Acc + naive acc.	2× inf.	Positional heuristics
IFR	Free	Silent parse failures
Bias audit (VS, SPS)	Low	Surface-cue exploitation
Bootstrap 95% CI	Free	Over-claiming on noise
Domain stratification	Free	Hidden domain gaps
≥50 human checks	Low	Lost ecological validity

L.10 Direction 9: Training Methodology for SLM Judges

Causal gap. The optimal training methodology for SLM judges at constrained parameter budgets is not established. Current practice includes SFT on evaluation data, DPO/preference optimization, and RL with verifiable rewards. Each has been studied in isolation, but no systematic comparison at the same model size under the same data budget exists.

Open problems.

1. **Pretraining selection vs. fine-tuning**: Does the base model matter more than the fine-tuning objective? The M-Prometheus result (Qwen2.5 > Llama for multilingual) suggests pretraining composition is critical.
2. **Reward shaping for consistency**: The J1 protocol (Whitehouse et al., 2025) uses consistency-based rewards. How does the consistency reward weight interact with accuracy rewards at different model scales?
3. **Data composition**: What is the optimal mixture of direct scoring, comparison, language feedback, and rubric application data at different model sizes?
4. **Ensemble distillation**: Can a heterogeneous LLM panel's consensus be distilled into an SLM student? Does ensemble distillation produce more robust judges than single-teacher distillation?
5. **Multi-format unified training**: The Multitask-J1 result (combining pairwise and pointwise training improves both) suggests systematic study of multi-format training at small scales.

THE INTERACTION BETWEEN BASE MODEL SELECTION AND FINE-TUNING STRATEGY IS THE MOST UNDER-STUDIED VARIABLE IN SLM JUDGE TRAINING. **Current papers report results on a single base model with a single fine-tuning strategy, making it impossible to disentangle the contribution of the base model's pretraining composition from the fine-tuning objective's effectiveness; a controlled factorial study varying both base model and fine-tuning strategy at a fixed parameter budget would provide the field's most valuable training methodology insight.** The practical stakes are high: if pretraining composition dominates (as M-Prometheus suggests), then the community should focus on base model selection rather than fine-tuning innovation; if fine-tuning strategy dominates, then RL-based training (J1, JudgeLRM) may be categorically superior regardless of base model.

L.11 Direction 10: Societal Implications and Deployment Ethics

Causal gap. The survey literature has focused predominantly on technical reliability with minimal attention to the societal implications of deploying automated judgment at scale. When SLM judges filter AI-generated content, evaluate AI-assisted work, or provide automated feedback in educational settings, their systematic biases propagate at scale.

Table 26: Extended research agenda: directions, time horizon, and tractability assessment. "Horizon" indicates estimated time to meaningful progress.

#	Direction	Causal Gap	Hor.	Intervention
1	Repr.-based eval.	Gen. bottleneck	Near	Probe atlas
2	Adaptive compute	Fixed budget	Near	Difficulty-pred.
3	Calib. uncert.	No confidence	Near	Conformal pred.
4	Scalable ovsght.	Capacity ceiling	Med	W2SG $\times$ verif.
5	Agentic eval.	Static mismatch	Med	PRM $\rightarrow$ traject.
6	Multilingual	English-centric	Med	Native + verify
7	Continual learn.	Judge staleness	Med	Drift monitoring
8	Std. protocols	Inconsist. rep.	Near	Report standard
9	Training meth.	No comparison	Near	Factorial study
10	Societal impact	Bias at scale	Long	Monocult. audit

High-stakes deployment concerns.

- **Homogenization of quality standards**: If all generators are trained against the same SLM judge, they converge toward that judge's preferences, homogenizing output style and potentially encoding particular cultural norms at scale.
- **Protected attribute sensitivity**: Bias measurements must extend to evaluation of responses about different protected groups. An SLM judge that rates responses about minority communities more harshly encodes discriminatory evaluation at scale.
- **Educational deployment**: Using SLM judges to evaluate student work raises accountability questions: what recourse exists when the judge is wrong?
- **Transparency requirements**: Disclosure that evaluation is automated becomes ethically necessary and potentially legally mandated.

MONOCULTURE RISK IS THE MOST DANGEROUS SOCIETAL CONSEQUENCE OF WIDESPREAD SLM JUDGE DEPLOYMENT. **If a single dominant SLM judge architecture is used across multiple RLHF pipelines, all trained generators converge toward that judge's quality standards, creating a monoculture where diversity of output style, reasoning approach, and cultural perspective is systematically suppressed.** This is analogous to monoculture risk in agriculture: a single point of failure (the judge's biases) propagates across the entire ecosystem. The mitigation is deliberate diversity: different RLHF pipelines should use different judges from different model families, ensuring that no single quality standard dominates. This recommendation aligns with the cross-family diversity principle established throughout this survey (Appendix E).

L.12 Research Agenda Summary

Table 26 provides a consolidated overview of the ten research directions, their causal gaps, time horizons, and highest-leverage interventions.

Closing note. The ten directions above share a common thread: SLM-as-a-Judge is not a solved problem reduced to engineering implementation, but a field with deep open theoretical and empirical questions. The efficiency advantages of SLMs, substantial cost reduction relative to frontier LLM judges, feasibility of private on-device deployment, and reduced latency for real-time evaluation, make this field practically important beyond academic interest. Closing the gaps identified above would make SLM judges not just economically attractive but reliably trustworthy, the prerequisite for high-stakes deployment at scale.