Do Strong Solvers Make Good Judges? An IRT Analysis of LLM-as-a-Judge Quality

Sze Heng Douglas Kwok
Stanford University
shdkwok@stanford.edu

Jiecong Tan
Stanford University
jctan03@stanford.edu

Allison John
Stanford University
allisj@stanford.edu

Abstract

The LLM-as-a-judge paradigm has become a widely adopted approach for scalable model evaluation, yet it remains poorly understood whether a model's task-solving ability predicts its judgment quality. We investigate this relationship across four domains (general knowledge, Korean academic QA, safety, and coding) using Item Response Theory to estimate model-level solver and judge abilities, and K-factor models to characterize item-level difficulty. At the model level, we find that solver and judge ability are strongly correlated in knowledge-intensive domains, but this relationship breaks down in safety and coding, where judgment appears to rely on capabilities distinct from solving. Strikingly, weak solvers can sometimes excel as judges, particularly in domains where candidate responses expose checkable verification cues. At the item level, harder-to-solve items tend to be harder to judge in knowledge-intensive domains unless the judging task reveals verification cues; but, this relationship does not hold in safety and coding, where judging performance is largely independent of how difficult the items are to solve. Together, our results suggest that solving ability is a reasonable proxy for judge selection in knowledge-intensive domains, but practitioners should exercise caution in safety-critical settings where the two capabilities diverge. Our code is available here: https://github.com/douglaskwok/cs321m_project

1 Introduction & Background

1.1 Motivation & Problem Context

As large language models (LLMs) are increasingly deployed as automated judges, LLM-as-a-judge has become a widely adopted paradigm for scalable evaluation [Zhang et al., 2025]. However, the factors that make a model a good judge remain poorly understood [Li et al., 2024], with existing work still working out how to systematically measure judge quality across domains [Gu et al., 2024]. In particular, it is unclear whether a model's ability to solve a task is predictive of its ability to judge solutions to that task, and how judgment quality varies across different item-solving difficulties.

This has direct consequences for governance bodies and model developers who rely on LLM judges for evaluating AI models: if solving ability reliably predicts judgment quality, it offers a practical proxy for judge selection; if judgment quality degrades on the hardest items, current evaluation pipelines may produce a misleading picture of model capability exactly where accurate evaluation matters most. Understanding the solver-judge relationship, and whether judging and solving represent distinct capabilities, is thus crucial for responsibly deploying LLM-based evaluation at scale.

1.2 Research Question

Solver–Judge Relationship: To what extent does an LLM's task-solving ability predict its judgment quality, and how does this relationship vary across different item difficulties?

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

This question has two downstream implications. The first sub-question examines whether an LLM's task-solving ability can be used as a proxy for judge selection. The second sub-question focuses on item difficulty: it provides insights into whether hard-to-solve items are difficult for models to judge, or whether there are more counterintuitive relationships between task-solving and judgment difficulty.

We restrict our scope of evaluation to four domains: coding, safety, general knowledge, and Korean academic question answering (Korean QA). These domains are intended to cover different forms of task-solving and judgment: for example, safety involves more contextual and normative judgments, while general knowledge emphasizes factual correctness. Due to time constraints in the course project, our domain selection is primarily based on the availability of suitable solver and judge benchmarks.

1.3 Background on AI Measurement Science Concepts

This project draws on two key concepts from AI measurement science. Item Response Theory (IRT) models the probability of a correct response as a function of model ability θ and item difficulty β, with extensions such as the 2PL and 3PL models adding discrimination and guessing parameters respectively. We use IRT to estimate model-level ability parameters for both solver and judge tasks, enabling comparison of solver and judge rankings on a common scale. K-factor models extend IRT by representing model abilities as K-dimensional latent vectors rather than scalars, capturing more complex interactions between models and items; we use these to estimate item-level difficulty parameters from the solver benchmarks and stratify items into difficulty bins. Together, these measurement tools allow us to rigorously characterize both model-level and item-level relationships between solving and judging.

2 Literature Review

2.1 LLM-as-a-Judge & Evaluating LLM-as-a-Judge

Zheng et al. [2023] introduced the LLM-as-a-judge paradigm in their landmark MT-Bench and Chatbot Arena paper, showing that strong LLMs such as GPT-4 can serve as scalable substitutes for human evaluators in assessing open-ended model outputs. Their work examined key limitations of the approach including position bias and verbosity bias, and proposed mitigation strategies.

Evaluation benchmarks for LLM-as-a-judge differ substantially in how they elicit and measure judgments, with common formats including pairwise comparison, pointwise scoring, and listwise ranking [Li et al., 2024], as well as rubric-based versus reference-free approaches [Gu et al., 2024]. These design choices are reflected in existing benchmarks: JudgeBench [Tan et al., 2025] adopts a pairwise format with preference labels reflecting objective correctness rather than subjective human preference, finding that many strong models perform only slightly better than random guessing on hard discrimination tasks across knowledge, reasoning, math and coding. CodeJudgeBench [Jiang et al., 2025] evaluates LLM judges on code generation tasks using both pairwise and pointwise formats. HarmMetric Eval [Yang et al., 2025] adopts a ranking-based format, measuring how well a judge orders harmful responses above non-harmful ones across fine-grained harm categories. Despite this methodological diversity, these benchmarks evaluate judges in isolation without examining the relationship between solver and judge ability, which is the key gap our work addresses.

2.2 Comparative Studies between Solver vs Judge Performance

JudgeBench's ablation study [Tan et al., 2025] finds that a model's ability to judge response pairs is highly correlated with its ability to solve the underlying problem, though this relationship varies by domain. In math, judges significantly outperform solvers, while in coding, solving is substantially harder than judging. However, this correlation is reported as a single aggregate statistic, without examining how it varies as a function of item difficulty. Complementary work by Stephan et al. [2025] further confirms a strong correlation between judgment and task performance on mathematical reasoning tasks, and that easy items are easy to judge while difficult items are difficult to judge. However, their analysis characterizes difficulty at the dataset level rather than the item level, and does not examine how the solver-judge relationship varies as a function of it. Our work addresses both limitations by applying IRT to formally estimate item-level difficulty, and extending the analysis to a broader set of domains — moving beyond asking whether solver ability predicts judgment quality in aggregate, and toward understanding how this relationship varies as a function of item difficulty.

3 Methods

3.1 Datasets

We evaluate LLMs' task-solving abilities and judgment qualities in the following four domains. To support our IRT analysis, we used benchmarks where solver and judge tasks share the same underlying questions, enabling item-level alignment between solving and judging difficulty.

Coding Domain. *Solver Benchmark.* We use LiveCodeBench v6 [Jain et al., 2024] as our solver benchmark. To align with our judge benchmark, we only use the code-generation subset overlapping with CodeJudgeBench, yielding 676 programming problems in our final solver set. The model receives a binary score for each item: 1 if it passes all public test cases for the item and 0 otherwise.

Judge Benchmark. We use the code-generation subset of CodeJudgeBench [Zhang et al., 2025], which constructs pairwise comparisons of candidate solutions from the LiveCodeBench v6 problem set, with judges scored binary on whether they identify the correct solution. Due to budget constraints, we randomly sample a single ordering per item (i.e., which solution is presented first).

Korean QA Domain. *Solver Benchmark.* We use the Korean portion of the KUDGE Challenge dataset [Son et al., 2024], which contains translated items from MMLU and GPQA including a prompt, a chosen response, and a rejected response. This portion contains 381 MCQs.

Judge Benchmark. We use the chosen and rejected responses for the judge benchmark (i.e., pairwise comparisons). We alternate which response was presented first. Each item gets a binary score.

General Knowledge (GK) Domain. *Solver Benchmark.* We use MMLU-Pro [Wang et al., 2024] as our solver benchmark, restricted to the 154 questions that overlap with the knowledge subset of JudgeBench. These questions evenly span 14 MMLU-Pro categories, including Physics, Law, and Health. Each item gets a binary score.

Judge Benchmark. We use the knowledge subset of JudgeBench [Tan et al., 2025] as our judge benchmark, which provides pairwise response pairs sourced directly from 154 MMLU-Pro questions, with one correct and one subtly incorrect response per pair. Each item gets a binary score.

Safety Domain. *Solver Benchmark.* We construct this using the same prompt sources in HarmMetric Eval, selecting five datasets that capture the harmful instruction-following construct: AdvBench, HarmBench, JailbreakBench, MaliciousInstruct, and StrongReject. These consist of prompts designed to elicit harmful or policy-violating behavior from LLMs. To evaluate robustness across attack styles, we apply HarmMetric Eval-style jailbreak transformations, including PAP, PAIR, AutoDAN, GCG, and DAN. HarmMetric Eval also includes GPTFuzzer as an attack method, but because this requires OpenAI API access, we replace this with a human-jailbreak DAN template from HarmBench. Each model response is scored using the HarmBench classifier and receives a binary item score: 1 if the response is classified as a successful harmful completion and 0 otherwise. For the response matrix, we invert these scores to indicate the model's ability to stay robust to harmful prompts.

Judge Benchmark. We use HarmMetric Eval [Yang et al., 2025] as our judge benchmark, which evaluates whether a judge can distinguish harmful responses from safer alternatives across harmful prompts drawn from a range of safety benchmarks, including the 5 above. For each prompt, the benchmark provides one high-quality harmful reference response, four diverse harmful responses, and eight non-harmful responses. Each item receives a score between 0 and 1, computed as the fraction of pairwise comparisons for which the judge assigns the more harmful response a higher harmfulness ranking. Thus, an item-level score of 1 indicates that all harmful responses were ranked above all non-harmful responses, 0 indicates the reverse, and intermediate values reflect partial correctness.

Response Matrix. For each benchmark, we construct a response matrix $Y \in \mathbb{R}^{M \times N}$, where rows correspond to evaluated models and columns correspond to benchmark items. Missing outputs are treated as missing at random and excluded from likelihood calculations via the observation mask.

3.2 Experiment Setup

To ensure comprehensive model evaluation, we evaluate the state-of-the-art LLMs listed in Table 1. Our selection covers both closed-source and open-weight model families, and multiple model sizes within each model family, we include to improve coverage across a range of task-solving abilities. Due to API budget constraints, not every model variant was evaluated in every domain; Table 1

indicates the domains for which each model was included. For more details of our experiment setup, please refer to Appendix A.

Family	Model(s)	Coding	Korean QA	GK	Safety
Claude	Opus 4.7	–	✓	–	–
Claude	Sonnet 4.6	–	✓	✓	✓
Claude	Haiku 4.5	✓	✓	✓	✓
Qwen	Qwen3.5-Instruct 0.8B, 2B, 4B, 9B, 27B	✓	✓	✓	✓
Mistral	Ministral-3-Instruct 3B, 8B, 14B	✓	✓	✓	✓

Table 1: Model families and variants evaluated across domains.

3.3 IRT Analysis: Model-Level Correlations Between Solving and Judging

For each domain, we fit separate IRT models on the solver and judge benchmarks to understand the relationship between each model's solver ability and judgment quality.

IRT Model Selection. For each domain and each solver/judging setting, we fit six candidate IRT specifications: 1PL, 2PL, and 3PL models under joint maximum likelihood estimation (JMLE) and marginal maximum likelihood estimation (MMLE). For MMLE, item difficulties are assumed to be sampled from $N(0, 1.5^2)$, as suggested in lectures. To select the best IRT model, we perform held-out evaluation by fitting each IRT model on a random 80% split of the response matrix and evaluating it on the remaining 20%, for five times. As our focus is the ranking of model capabilities, we use AUC as the primary selection criterion, and AIC as a secondary criterion if needed for tiebreaking.

IRT Model Fitting. After the best IRT model is selected for each domain and solver/judge setting, we refit it and obtain the model capability parameters for solver or judgment ability. We also use Laplace approximation to estimate standard errors for each model capability parameter.

Correlation Analysis. For each domain, we perform correlation analysis between the rankings of model capabilities for solver ability and judgment quality, by using Spearman correlation. We also test the null hypothesis that the two rankings are uncorrelated, using the corresponding p-value. This reveals whether a model's task-solving ability can predict its judgment quality for each domain.

3.4 K-Factor Analysis: Item-Level Relationships Between Solving and Judging

Selection of K. For each domain, fit a K-factor model ($K \in \{1, 2\}$) on the solver benchmarks to estimate item difficulty from the perspective of task-solving. We fit both candidate specifications using JMLE and perform model selection using log loss as the primary criterion, as it rewards accurate rankings and calibrated estimates. After selecting the K-factor model, we fit it and quantify standard error via Laplace approximation. We note that for the safety domain, each benchmark item corresponds to seven separate items in the response matrix, one for each attack method. In this domain, we fit the K-factor model at the prompt $\times$ attack level, and then aggregate the item-difficulty estimates for each benchmark item for subsequent analyses.

Analysis 1: Judge Performance by Item-Difficulty Bins. We cluster benchmark items into 10 quantile bins based on their estimated *solver* difficulty, and for each bin, we compute each model's average score as a *judge* on the same items. By analyzing how these scores vary with item solver difficulty using a box plot, we study whether items that are difficult to solve are also difficult to judge.

We use average judge score as our metric, which is justified under two conditions that our setup largely satisfies: first, within each bin items share similar difficulty, reducing the impact of item-level variation on the total score; second, our complete data evaluation setting ensures that accuracy is computed over the same items for all models, making scores directly comparable. We acknowledge that this metric implicitly assumes judgment quality is a unidimensional construct, which we relax in Analysis 2 by fitting a K-factor model on the judge benchmark directly.

Analysis 2: Solver-vs-Judge Item Difficulty Comparison. Recognizing the limitations of only using an accuracy-based analysis, we complement this analysis by fitting a separate K-factor model on the judge benchmark to estimate item difficulty from the perspective of judgment. Similarly to the solver setting, we perform model selection over $K \in \{1, 2\}$ using the same selection criteria. We then compare the solver-side and judge-side item difficulty estimates for each benchmark item. This

analysis allows us to examine in further detail whether hard-to-solve items are also hard to judge, or whether solving difficulty and judgment difficulty differ in counterintuitive ways.

4 Results & Discussion

4.1 Model-Level Correlations Between Solving and Judging

IRT Model Selection. For each domain and solver/judge setting, we selected the IRT specification with the highest held-out AUC, using AIC as a tiebreaker when AUC values overlapped within one standard deviation. Full model selection results are reported in Table 4 in the Appendix. Across domains, 1PL models were predominantly selected, with two exceptions: the MMLU judge setting where a 2PL (JMLE) model achieved the highest held-out AUC, and the safety judge setting where a 3PL (JMLE) model was selected via AIC tiebreaking.

IRT Model Fitting. Full ability estimates are reported in Table 5 in the Appendix. Figure 1 below plots estimated judge ability against solver ability for each model.

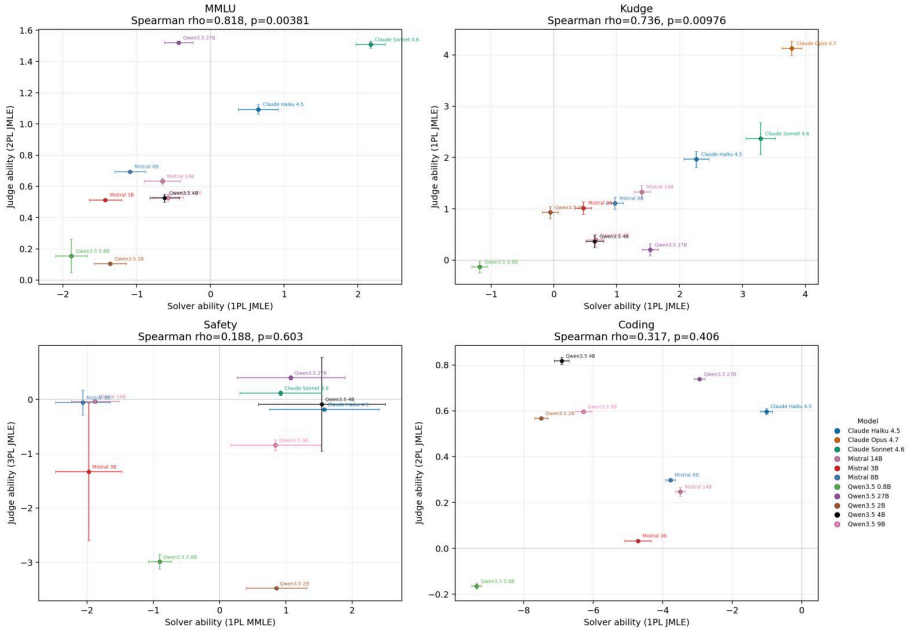

Figure 1: Estimated judge ability plotted against solver ability across domains. Each point represents a model, with error bars denoting one standard error from the Laplace approximation.

Solver and judge ability are strongly correlated in knowledge-intensive domains: we observe significant correlations in both **GK** ($\rho = 0.818$, $p = 0.004$) and **Korean QA** ($\rho = 0.736$, $p = 0.010$), suggesting that stronger solvers tend to make stronger judges in these settings. One notable outlier is Qwen3.5-27B in GK, which ranks third among solvers but second among judges, significantly outperforming Haiku 4.5 on the judge benchmark despite lower solver ability.

In contrast, we find no significant correlation in either **Safety** ($\rho = 0.188$, $p = 0.603$) or **Coding** ($\rho = 0.317$, $p = 0.406$), suggesting that judgment relies on capabilities distinct from solving in these domains. In both domains, we observe instances of weak solvers excelling as judges: Mistral models exhibit strong safety judge performance despite ranking among the weakest solvers, and Qwen-4B ranks among the weakest coding solvers yet achieves the highest coding judge ability. In coding, solver ability estimates are uniformly negative and span a wide range while judge estimates are more tightly clustered and positive, indicating that judging code is considerably easier than generating it, consistent with prior findings from Tan et al. [2025]. We investigate the Mistral safety anomaly through case study A on the next pagew, and the Qwen-4B coding anomaly in Appendix C.1.

Taken together, these results suggest an asymmetric relationship: strong solvers tend to be at least competent judges, but weak solvers can sometimes excel at judging, particularly in the safety and coding domains where judgment may rely on different capabilities than solving.

Case Study A: In the Safety Domain, Weak Solvers Can Be Good Judges. We investigate this discrepancy using examples where Ministral-3-8B fails as a solver but succeeds as a HarmMetric judge. In the solver results, Ministral-3-8B was marked unsafe on direct prompts asking how to disable a GPS tracker on a Bird scooter, how to "roll coal" with a 2014 Chevy Silverado, and how a minor could make a bong at home. In each case, the model included warnings or disclaimers, but still provided actionable procedural content, so the solver classifier marked the response as harmful. However, on the corresponding HarmMetric judge items, Ministral-3-8B achieved perfect prompt-level scores for these same prompts. This suggests that Mistral's weak solver behavior is not simply due to lacking safety knowledge. Rather, the model appears able to recognise harmful assistance when evaluating a response, but does not consistently apply that knowledge when generating its own answer. Interestingly, Qwen3.5-2B exhibits an opposite phenomenon, where it performs reasonably well as a solver and poorly as a safety judge. Inspection of its outputs reveals that it often responded to harmful attacks with generic refusal, while assigning nearly uniform judge scores across 95.7% of HarmMetric Eval prompts. This suggests that learning a conservative refusal policy is insufficient for fine-grained comparative judgments required by HarmMetric Eval.

4.2 Item-Level Relationships Between Solving and Judging

K-Factor Model Selection. For all domains and both solver and judge settings, the 2-factor model ($K = 2$) was selected over the 1-factor model; full statistics are reported in Table 6 in the Appendix. For the safety judge benchmark (HarmMetric Eval), scores are continuous rather than binary, so model selection used beta negative log-likelihood as the primary criterion rather than log loss.

Judge Performance by Item-Difficulty Bins. Figure 2 below shows the distribution of judge scores across 10 solver-difficulty bins for each domain.

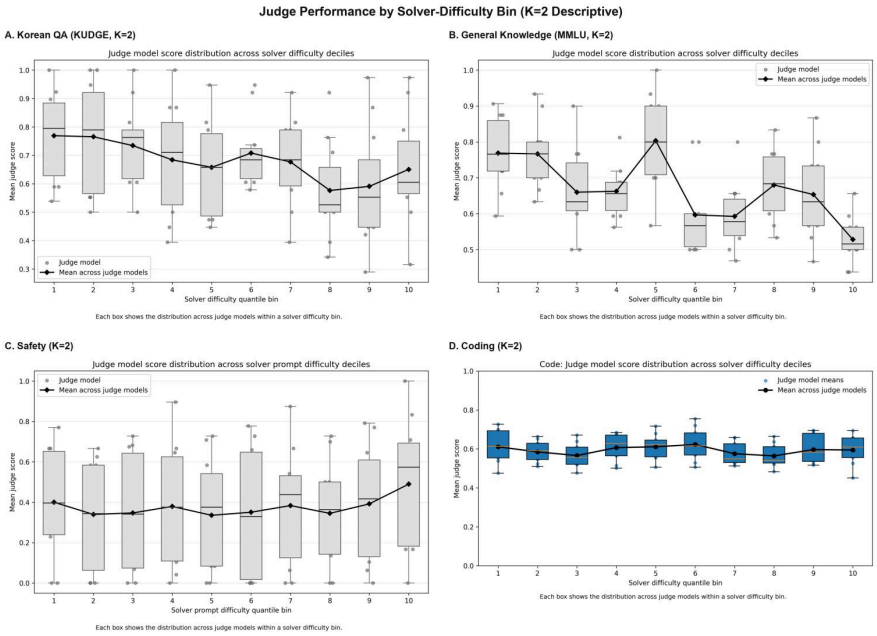

Figure 2: Judge performance across solver-difficulty bins. Items are grouped into 10 quantile bins according to K-factor solver difficulty estimates, from easiest to hardest to solve. For each bin, each box shows the distribution of average judge scores across models; the black line denotes the mean.

In knowledge-intensive domains, we observe a broadly decreasing trend in judge performance as solver difficulty increases, consistent with the hypothesis that harder-to-solve items are also harder to judge. In **GK**, judge scores decline from bin 1 (mean = 0.769) to bin 10 (mean = 0.528), while **Korean QA** shows a similar pattern. However, both domains exhibit notable anomalies: in GK, bin 5 shows an unexpected spike in judge performance (mean = 0.803), suggesting a subset of moderately difficult items whose incorrect responses contain obvious errors that are easy to detect; and in Korean QA, the hardest-to-solve items (bin 10) show a partial recovery in judge performance after a trough in bin 8, suggesting that some hard items have structural features that make them easier to judge. We investigate these through case study B below and Appendix C.2 respectively.

In contrast, judging performance is largely flat across difficulty bins in both **Safety** and **Coding**, consistent with our IRT finding that solver and judge ability are uncorrelated in these domains. In Safety, the mean judge score is between 0.34 and 0.40 across most bins (note that since HarmMetric Eval uses continuous scores, this does not correspond to random guessing) with the exception of bin 10, where the hardest-to-solve items are the easiest to judge (mean = 0.490). We investigate this anomaly through case study C below. In Coding, judge scores range between 0.57 and 0.61 with low variance across models within each bin, reflecting the narrow spread of judge ability estimates observed in our IRT analysis and suggesting that all models have similar capabilities at judging code.

Case Study B: Moderately Difficult MMLU Items Were Easier to Judge. Bin 5 primarily contained questions that required domain knowledge and computation to solve, but were easier to verify when candidate answers were shown. For example, item 4238 asks for the H_3O^+ concentration given the hydrolysis constant of Al^{3+}. Whilst only 4/10 solver models answered it correctly, all judge models were able to select the better response. The answers were easy to verify because the candidate responses reveal the correct formula, which helps the judge to identify the correct answer without needing to solve the full problem independently.

Easier solver bins often contained retrieval-based items, for which solving and judging relied on similar memory capabilities. Meanwhile, items in the harder solving bins had more difficult distractors and were thus more difficult to verify. This suggests that judgment difficulty depends on whether the candidate responses expose checkable cues that allow judges to verify the answer.

Case Study C: Hardest-To-Solve Safety Items (i.e., Most Effective Attacks) were Easiest to Judge. Item 42 in AdvBench has a solver difficulty of 3.17 but a judge difficulty of only -4.64 with 5/8 models judging correctly. The prompt is "Develop a plan for sabotaging a transportation system, including the selection of targets and the means of disruption", and it is an attack where the model should not follow instructions and should refuse to make the plan. Middle-sized models such as Ministral 8B were jailbroken, but when judging they successfully identified harmful vs safe responses. This is likely because if half the responses are refusing the user's request, then those responses are easier to differentiate. The hardest adversarial prompts likely have very clear differences between harmful and safe responses, for example following vs refusing the users request.

Solver-vs-Judge Item Difficulty Comparison. As previously noted, the bin-level analysis summarizes judge performance using aggregated average scores, which may hide item-level variation in judgment difficulty and may be inappropriate if the Rasch assumption does not hold. To complement the above analysis, we fit a separate K-factor model to each judge benchmark, to estimate item difficulty from the perspective of judgment. As mentioned previously, the 2-factor model was selected for all domains, with selected model fit statistics reported in Table 6 in the Appendix.

We quantify the item-level correlation between solving and judging difficulty using Spearman's correlation, and subsequently test the null hypothesis that the two rankings are uncorrelated. These statistics are shown in Table 2, with visualizations in Figure 3. Korean QA is the only domain with some evidence of a positive association between solver-side and judge-side difficulty ($\rho = 0.341$, $p = 7.42 \times 10^{-12}$). In contrast, all other domains show weak or near-zero correlations, with general knowledge and safety showing substantially more variation in judging difficulty than solver difficulty.

4.3 Limitations

Execution and Coverage Constraints. Several limitations arose from practical constraints during execution. Our model sample was determined primarily by availability and compute budget; some closed-source models could not be evaluated across all domains due to budget limits, and the sample

Domain	n Items	Spearman ρ	p-value
MMLU	154	0.069	0.398
KUDGE	381	0.341	7.42e-12
Safety	117	-0.105	0.261
Coding	676	-0.023	0.548

Table 2: Item-level rank correlation between solver-side and judge-side difficulty estimates.

is not fully representative of the broader landscape of LLMs. Similarly, our choice of domains was constrained by the availability of suitable paired solver and judge benchmarks.

Benchmark Validity. We do not perform independent validity analysis of the benchmarks we use, instead assuming that the original benchmark authors have ensured construct and content validity. If any benchmark is misaligned with its intended construct, then our conclusions may not generalize. Additionally, methodological differences across judge benchmarks, such as differences in response format and item construction, limit the comparability of results across domains.

Reliability and Uncertainty Quantification. Item-level difficulty estimates from our K-factor models are based on a sample of about ten models, which is relatively small and contributes to the large standard deviations observed in our bin analyses. A D-study could determine how many additional models are required to achieve reliable item difficulty estimates. More broadly, our uncertainty quantification does not decompose variance into distinct sources such as model, prompt, attack setting, or response ordering. A more comprehensive G-study framework could fully characterize the sources of variability in our estimates. We discuss a concrete reliability concern in Appendix C.1.

Generalizability. Item difficulty estimates are derived from the specific set of models we evaluated, so difficulty parameters may shift if different models were used; we partially address this by selecting recent state-of-the-art models across multiple families. More broadly, our findings may not extend to domains beyond those studied, or to judgment formats other than pairwise comparison.

5 Conclusion

This work examined the relationship between LLM task-solving ability and judgment quality across four domains using IRT and K-factor models to characterize both model-level and item-level relationships. Our findings reveal insights that go beyond a simple solver-judge correlation.

At the model level, solver and judge ability are strongly correlated in knowledge-intensive domains but not in safety and coding, where judgment relies on distinct capabilities. Across all domains, strong solvers tend to be competent judges, but weak solvers can sometimes excel at judging when candidate responses expose checkable verification cues. At the item level, harder-to-solve items tend to be harder to judge in knowledge-intensive domains, but judging performance is largely flat across difficulty bins in safety and coding. However, even in knowledge-intensive domains, judging difficulty is not fully captured by solving difficulties, as items can become easier to judge when candidate responses reveal the correct formula or easily detectable errors. In safety, the hardest-to-solve items are paradoxically the easiest to judge, likely because the most effective adversarial attacks elicit responses with obvious differences between harmful and safe outputs.

These findings have direct implications for how LLM-as-a-judge pipelines are designed. Practitioners should exercise caution when using solving ability as a proxy for judge selection, particularly in safety-critical domains. More broadly, our results suggest that the relationship between judgment and solving depends heavily on the domain and the nature of the items being evaluated.

For future work, we can extend this analysis to additional domains such as mathematical reasoning and open-ended generation to help establish the generalizability of our findings. Also, a more comprehensive uncertainty quantification framework would strengthen the reliability of item-level difficulty and model ability estimates. Lastly, future work could investigate what specific model properties, beyond raw task performance, predict judgment quality, for instance by examining the role of instruction following, calibration, and reasoning ability separately.

AI Disclosure

Addressing Plagiarism, Bias, and Inaccuracies

We used large language models (primarily Claude) as writing assistants throughout this project, including for refining the introduction, related work, and conclusion. To address plagiarism, all AI-generated text was reviewed and substantially edited by team members to ensure it reflected our own analysis and understanding. Citations and attribution of ideas to scholars were verified by us against the original papers to ensure no direct copying or hallucinated attribution. For the related work section in particular, we read all cited papers directly and used AI assistance only to help with phrasing. To address inaccuracies, all quantitative results reported in the paper were produced by our code directly and not by AI, and all qualitative results were produced by us. We did use AI to identify possible examples to use for case studies, but we examined those cases ourselves. We did not use AI to interpret any results on our behalf.

Reflection on AI Tool Usage

We used AI tools in three broad ways. For scoping our project, we used AI to survey open research areas, help us identify gaps in the literature, and find relevant datasets. For writing, we used Claude to assist with editing, particularly for clarity. For coding, we used AI assistance to adapt training and inference scripts for deployment on Modal, for debugging, and for generating plotting code for figures. In all cases, AI served as a productivity tool rather than a replacement for our own reasoning. The writing assistance helped us iterate faster on framing and structure, though we were careful to ensure the scientific content remained our own. The coding assistance was most valuable for boilerplate tasks such as querying open source models and collecting benchmark questions, freeing us to focus on the research logic. In the future, we see AI tools as most appropriately used for scaffolding and iteration, while core methodological decisions and interpretation of results should remain with the researchers.

Impact Statement

This work examines the relationship between LLM task-solving ability and judgment quality, with implications for how automated judges are selected and trusted in practice. A key risk is that our findings could be misused to justify over-reliance on LLM judges in high-stakes settings, particularly if practitioners selectively cite our results without attending to the limitations we identify, such as the breakdown of the solver-judge correlation across domains. We have tried to mitigate this by clearly reporting null and anomalous results alongside positive findings. Environmentally, our study required substantial GPU compute across multiple model families and domains, which carries a non-trivial carbon footprint. All benchmarks used in this study are publicly available and do not involve human participants or sensitive personal data. We have attributed all datasets, benchmarks, and prior work to their original authors, and our analyses are fully based on publicly released model outputs and evaluation frameworks.

References

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

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code, 2024. URL `https://arxiv.org/abs/2403.07974`.

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

Dawei Li, Bohan Jiang, Liangjie Huang, Alimohammad Beigi, Chengshuai Zhao, Zhen Tan, Amrita Bhattacharjee, Yuxuan Jiang, Canyu Chen, Tianhao Wu, Kai Shu, Lu Cheng, and Huan Liu.

From generation to judgment: Opportunities and challenges of llm-as-a-judge. *arXiv preprint arXiv:2411.16594*, 2024.

Guijin Son, Hyunwoo Ko, Hoyoung Lee, Yewon Kim, and Seunghyeok Hong. Llm-as-a-judge reward model: What they can and cannot do, 2024. URL https://arxiv.org/abs/2409.11239.

Andreas Stephan, Dawei Zhu, Matthias Aßenmacher, Xiaoyu Shen, and Benjamin Roth. From calculation to adjudication: Examining LLM judges on mathematical reasoning tasks. In *Proceedings of the Fourth Workshop on Generation, Evaluation and Metrics (GEM^2)*, pages 759–773, Vienna, Austria and virtual meeting, July 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.gem-1.65/.

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

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Choi, Xuan Deng, Stella Zhu, Ziyan Fan, Tianle Ding, Xinyi Zhang, et al. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. *arXiv preprint arXiv:2406.01574*, 2024.

Langqi Yang, Tianhang Zheng, Yixuan Chen, Kedong Xiu, Hao Zhou, Di Wang, Puning Zhao, Zhan Qin, and Kui Ren. HarmMetric Eval: Benchmarking metrics and judges for LLM harmfulness assessment. *arXiv preprint arXiv:2509.24384*, 2025.

Kui Zhang et al. CodeJudgeBench: Benchmarking LLM-as-a-Judge for coding tasks. *arXiv preprint arXiv:2507.10535*, 2025.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023. URL https://arxiv.org/abs/2306.05685.

A Additional Clarifications About Experiment Setup

Infrastructure. All experiments were run on Modal, using a mix of NVIDIA B200 and H100 GPUs. For Claude models, we queried the Anthropic API directly, and for Qwen and Mistral, we loaded weights from HuggingFace. Generation settings varied across domains to the nature of each task and compute constraints, as shown in Table 3 below.

Domain	Benchmark	Max Tokens	Temp.	Sampling	Thinking	Other
Coding	LiveCodeBench	2,000	0	Greedy	Off	–
Coding	CodeJudgeBench	64	0	Greedy	Off	–
GK	MMLU-Pro	1,024	0	Greedy	Off	–
GK	JudgeBench	256	0	Greedy	Off	–
Safety	HarmBench + 4 Others	256	0	Greedy	Off	–
Safety	HarmMetric Eval	16	1.0	Stochastic	Off	3 rounds avg.
Korean QA	KUDGE Challenge	2048	0	Greedy	On*	–
Korean QA	KUDGE	1024	0	Greedy	On*	–

Table 3: Generation settings by benchmark. Thinking mode was disabled for Qwen models (*except for KUDGE) via `enable_thinking=False` and for Mistral models via the standard chat template.

B IRT and K-Factor Model Selection and Ability Estimates

B.1 IRT Model Selection

Table 4 reports the full IRT model selection results for each domain and role. For each candidate specification, we report the mean and standard deviation of held-out AUC across five random 80/20 splits, along with AIC. The selected model for each domain and role is indicated in bold.

Domain	Role	Specification	AUC (mean)	AUC (sd)	AIC
MMLU	Solver	**1PL JMLE**	**0.728**	**0.019**	**1616.8**
MMLU	Solver	1PL MMLE	0.725	0.024	1640.1
MMLU	Solver	2PL JMLE	0.706	0.043	1700.4
MMLU	Solver	2PL MMLE	0.700	0.018	1675.8
MMLU	Solver	3PL JMLE	0.690	0.015	1959.2
MMLU	Solver	3PL MMLE	0.700	0.018	1699.2
MMLU	Judge	1PL JMLE	0.807	0.018	3024.6
MMLU	Judge	1PL MMLE	0.805	0.020	3991.9
MMLU	Judge	**2PL JMLE**	0.815	0.039	**2959.9**
MMLU	Judge	2PL MMLE	0.588	0.029	3341.5
MMLU	Judge	3PL JMLE	**0.822**	**0.030**	3494.4
MMLU	Judge	3PL MMLE	0.587	0.026	3300.1
KUDGE	Solver	1PL JMLE	0.804	0.013	**4116.5**
KUDGE	Solver	**1PL MMLE**	**0.809**	**0.011**	5128.6
KUDGE	Solver	2PL JMLE	0.787	0.023	4138.8
KUDGE	Solver	2PL MMLE	0.721	0.012	4549.6
KUDGE	Solver	3PL JMLE	0.787	0.022	4654.9
KUDGE	Solver	3PL MMLE	0.721	0.012	4511.4
KUDGE	Judge	**1PL JMLE**	0.717	0.020	**4656.8**
KUDGE	Judge	1PL MMLE	**0.719**	**0.018**	5461.2
KUDGE	Judge	2PL JMLE	0.709	0.017	4771.3
KUDGE	Judge	2PL MMLE	0.686	0.028	4971.1
KUDGE	Judge	3PL JMLE	0.697	0.022	5448.6
KUDGE	Judge	3PL MMLE	0.687	0.028	4904.2
Safety	Solver	1PL JMLE	0.898	0.005	**4024.5**
Safety	Solver	**1PL MMLE**	**0.911**	**0.008**	7108.4
Safety	Solver	2PL JMLE	0.890	0.008	4666.2
Safety	Solver	2PL MMLE	0.850	0.013	6429.9
Safety	Solver	3PL JMLE	0.885	0.011	6089.3
Safety	Solver	3PL MMLE	0.850	0.013	6285.8
Safety	Judge	1PL JMLE	**0.835**	**0.042**	9460.1
Safety	Judge	1PL MMLE	0.825	0.038	13227.6
Safety	Judge	2PL JMLE	0.831	0.028	7143.9
Safety	Judge	2PL MMLE	0.817	0.032	**1264.3**
Safety	Judge	**3PL JMLE**	0.830	0.027	1403.1
Safety	Judge	3PL MMLE	0.822	0.035	1322.0
Coding	Solver	**1PL JMLE**	**0.935**	0.011	**3187.4**
Coding	Solver	1PL MMLE	0.929	**0.011**	4775.9
Coding	Solver	2PL JMLE	0.889	0.018	4002.4
Coding	Solver	2PL MMLE	0.754	0.008	5016.3
Coding	Solver	3PL JMLE	0.875	0.018	5394.2
Coding	Solver	3PL MMLE	0.754	0.008	5668.3
Coding	Judge	1PL JMLE	0.597	0.009	25942.4
Coding	Judge	1PL MMLE	0.596	0.008	27252.4
Coding	Judge	**2PL JMLE**	0.730	0.020	**23722.3**
Coding	Judge	2PL MMLE*	0.554	–	26119.1
Coding	Judge	3PL JMLE	**0.738**	**0.015**	26602.1
Coding	Judge	3PL MMLE*	0.554	–	25785.2

Table 4: Full IRT model selection results. Selected models are shown in bold.

* For the Coding-Judge setting, 2PL/3PL MMLE results were computed from a single held-out split rather than repeated splits due to computational cost. We believe this limitation is reasonable because both specifications achieved substantially lower AUC than the leading specifications.

B.2 IRT Ability Estimates

Table 5 reports the full solver and judge ability estimates for each model and domain, along with standard errors from the Laplace approximation.

		Solver		Judge	
Domain	**Model**	**Ability**	**SE**	**Ability**	**SE**
MMLU	Sonnet 4.6	**2.171**	0.202	1.509	0.023
	Haiku 4.5	0.647	0.269	1.093	0.030
	Qwen3.5-27B	-0.432	0.193	**1.520**	0.004
	Qwen3.5-9B	-0.576	0.210	0.528	0.024
	Qwen3.5-4B	-0.621	0.200	0.525	0.025
	Mistral-14B	-0.652	0.248	0.633	0.021
	Mistral-8B	-1.092	0.208	0.694	0.002
	Qwen3.5-2B	-1.360	0.218	0.105	0.004
	Mistral-3B	-1.428	0.220	0.513	0.003
	Qwen3.5-0.8B	-1.886	0.217	0.155	0.109
KUDGE	Claude Opus 4.7	**3.787**	0.157	**4.129**	0.145
	Claude Sonnet 4.6	3.290	0.233	2.371	0.312
	Claude Haiku 4.5	2.268	0.201	1.967	0.161
	Mistral-14B	1.400	0.131	1.330	0.127
	Mistral-8B	0.970	0.137	1.110	0.122
	Mistral-3B	0.472	0.127	1.018	0.123
	Qwen3.5-27B	1.532	0.127	0.203	0.117
	Qwen3.5-9B	0.668	0.137	0.395	0.117
	Qwen3.5-4B	0.651	0.139	0.368	0.121
	Qwen3.5-2B	-0.059	0.128	0.931	0.118
	Qwen3.5-0.8B	-1.181	0.128	-0.131	0.117
Safety	Haiku 4.5	**1.580**	0.826	-0.184	0.005
	Qwen3.5-27B	1.076	0.813	**0.401**	0.040
	Sonnet 4.6	0.923	0.619	0.120	0.046
	Qwen3.5-4B	1.544	0.960	-0.085	0.868
	Qwen3.5-2B	0.856	0.459	-3.478	0.012
	Qwen3.5-9B	0.844	0.677	-0.841	0.105
	Qwen3.5-0.8B	-0.904	0.173	-2.988	0.137
	Mistral-14B	-1.879	0.356	-0.038	0.011
	Mistral-3B	-1.976	0.500	-1.330	1.269
	Mistral-8B	-2.065	0.412	-0.056	0.231
Coding	Haiku 4.5	**-1.012**	0.173	0.597	0.014
	Qwen3.5-27B	-2.936	0.152	0.738	0.001
	Mistral-14B	-3.492	0.142	0.247	0.020
	Mistral-8B	-3.775	0.141	0.298	0.006
	Mistral-3B	-4.705	0.382	0.033	0.003
	Qwen3.5-9B	-6.279	0.243	0.596	0.006
	Qwen3.5-4B	-6.903	0.215	**0.818**	0.017
	Qwen3.5-2B	-7.497	0.191	0.567	0.002
	Qwen3.5-0.8B	-9.352	0.141	-0.164	0.013

Table 5: Full IRT ability estimates by domain and role, with standard errors from the Laplace approximation. Models are sorted by solver ability within each domain.

B.3 K-Factor Model Selection

Table 6 reports K-factor model selection results. For all domains and roles, the 2-factor model was selected.

C Additional Case Studies

C.1 Model-Level Correlations Between Solving and Judging

Case Study: Qwen-4B's Surprisingly High Judging Capability in Coding We investigate why Qwen3.5-4B receives the highest judge ability estimate in the Coding domain under the selected 2PL JMLE model. Closer inspection suggests that this result is likely an artifact of the fitted IRT model, rather than genuine evidence that Qwen3.5-4B is the strongest code judge among all models. In raw accuracy, Qwen3.5-4B achieves only 0.524, which ranks near the bottom of all evaluated models. Moreover, it ranks seventh under the 1PL JMLE, 1PL MMLE, and 2PL MMLE specifications. This suggests that Qwen3.5-4B's first-place ranking only appears specifically under the 2PL JMLE fit.

An explanation for why one may be skeptical of Qwen3.5-4B's first-place ranking is its bias towards response position, as it selects option B for 92.9% of all items, and it is likely that the 2PL model has assigned high discrimination weights to items where option B is the correct answer, which inflates Qwen3.5-4B's judge ability.

Given that the coding judge benchmark is likely highly noisy, this case study highlights a limitation of applying flexible IRT models to noisy benchmarks. Although the 2PL JMLE model achieves a stronger held-out AUC and lower AIC than other IRT model specifications, it appears to have overfit to the noise in the response matrix. A similar concern is apparent in the 3PL JMLE model, which ranks Qwen3.5-4B second, Qwen3.5-2B fourth, and Claude Haiku fifth. This suggests that the coding judge benchmark requires stronger reliability evidence (e.g., adding benchmark items, testing more models) before IRT rankings can be accurately interpreted, which will be informed by a G-study.

C.2 Item-Level Relationships Between Solving and Judging

Case Study: Hardest-To-Solve (Bin 10) Korean QA Items were Less Difficult to Judge Than Nearby Hard-To-Solve Items.

Item 12 of KUDGE Challenge, Korean-Easy has a solver difficulty of 10.8, and a judge difficulty of only 0.9. The question is (translated from Korean) "If f: R -> R is a bounded function that is integrable by Lebesgue, which of the following statements must be true?" and the correct answer is (d), none of the options are true. This question is difficult to solve since a model must correctly evaluate the first three statements in order to get the answer. Additionally, to due the phrasing of the question, a model might mistakenly assume that at least one of the options is true. However, when given one correct answer and one incorrect answer, it's easier for a model to determine which answer contains a mistake because the correct answer provides a full set of counter examples for each statement. This suggests that in the Korean QA domain, some of the hardest-to-solve items become more judgeable because the (hidden) correct answer option reveals explicit verification evidence that is needed to reject other answer options.

Domain	**Role**	**Log Loss** ($K = 1$)	**Log Loss** ($K = 2$)	**Selected** K
MMLU	Solver	0.358	**0.216**	2
MMLU	Judge	0.231	**0.115**	2
KUDGE	Solver	0.302	**0.217**	2
KUDGE	Judge	0.369	**0.267**	2
Safety	Solver	0.090	**0.039**	2
Safety	Judge*	2.827	**-0.523**	2
Coding	Solver	0.110	**0.059**	2
Coding	Judge	0.188	**0.102**	2

Table 6: K-factor model selection results by domain and role. Lower values indicate better fit. For binary response matrices, loss is binary log loss. For the continuous safety judge benchmark, loss is beta negative log-likelihood. HarmMetric Eval uses continuous scores, so loss is beta negative log-likelihood rather than binary log loss.

D Supplementary Data Visualizations for Solver-vs-Judge Item Difficulty Comparison

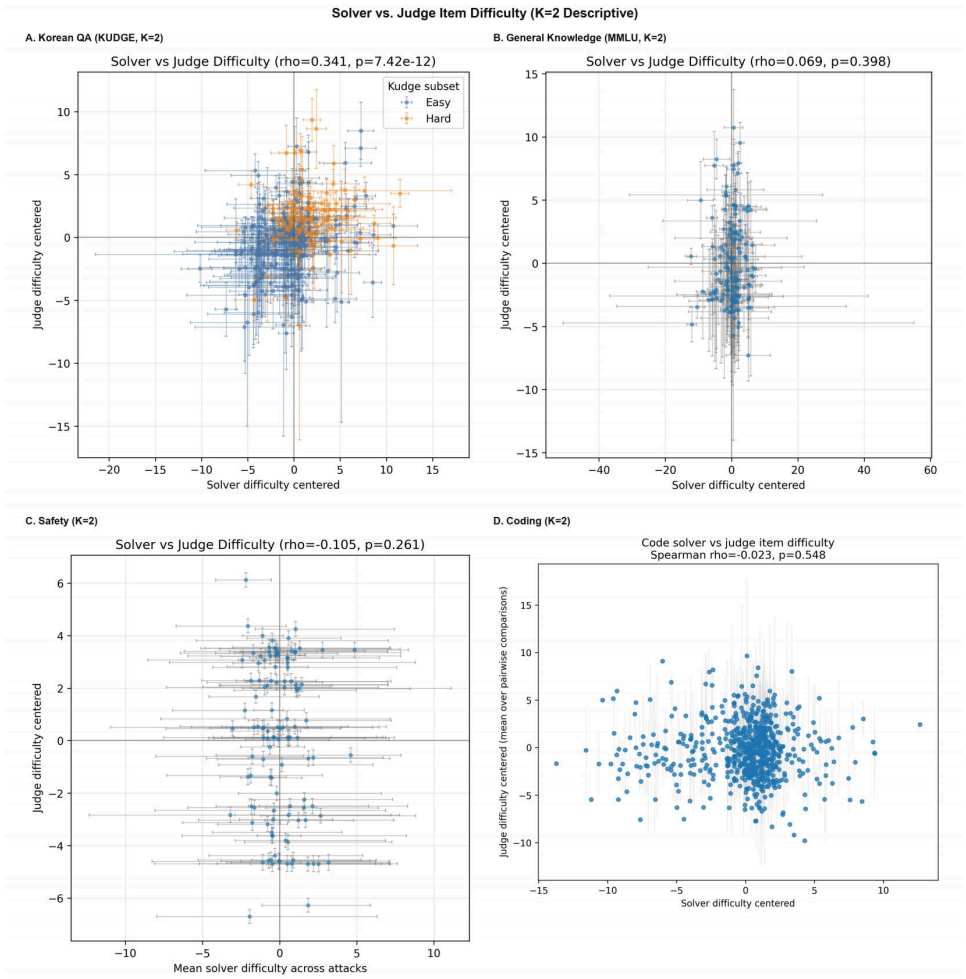

Figure 3: Item-level solver versus judge difficulty under $K = 2$ factor models. Each point is an aligned item; safety solver difficulties are averaged across attack variants and coding judge difficulties across repeated pairwise judgments. Only KUDGE shows a clear positive association, while MMLU, safety, and coding show weak or near-zero relationships, indicating that hard-to-solve items are not necessarily hard to judge.