SELECTIVE SELF-REFERENCE FOR LLM-AS-A-JUDGE: USING SELF-CONSISTENCY TO REDUCE ERROR PROPAGATION

FARS
Analemma
fars@analemma.ai

ABSTRACT

Self-reference, where an LLM judge first solves a task before evaluating candidate responses, has been shown to improve judgment accuracy. However, when the judge's self-generated answer is incorrect, using it as reference can propagate errors into the final judgment. We identify this error propagation problem and observe that self-consistency among multiple sampled solutions serves as a reliable proxy for answer correctness: unanimous agreement across five samples yields 78.66% correctness compared to only 35.51% for three-of-five agreement. Building on this insight, we propose Selective Self-Reference (SSR-Judge), which uses an agreement gate to conditionally enable self-reference only when the model exhibits high confidence. On MMLU-Pro pairwise preference judgment, SSR-Judge achieves 58.93% accuracy, outperforming both no-reference (52.07%) and always self-reference (58.21%) baselines by avoiding error propagation on uncertain items while preserving self-reference benefits when confidence is high.

WARNING: This paper was generated by an automated research system. The code is publicly available.[1]

1 INTRODUCTION

Large language models (LLMs) have emerged as powerful evaluators for assessing the quality of text generation, a paradigm known as LLM-as-a-Judge (Zheng et al., 2023). This approach has been widely adopted in evaluation benchmarks such as MT-Bench and Chatbot Arena (Chiang et al., 2024), enabling scalable assessment of model outputs without expensive human annotation. Recent work has demonstrated that providing the judge model with its own solution to the task before evaluation, termed self-reference, can substantially improve judgment accuracy (Lin et al., 2025). By first solving the problem independently, the judge gains deeper understanding of the task requirements and can make more informed comparisons between candidate responses.

However, self-reference introduces a critical vulnerability: when the judge's self-generated answer is incorrect, using it as a reference can propagate errors into the final judgment. Our analysis reveals that on items where the model's self-answers exhibit low agreement across multiple samples, indicating uncertainty, the always-reference approach achieves only 24.50% accuracy compared to 37.75% for no-reference judging. This error propagation effect undermines the benefits of self-reference precisely when the judge is least confident in its own solution.

We observe that self-consistency among multiple sampled solutions serves as a reliable proxy for answer correctness. When all five sampled answers agree unanimously, the correctness rate reaches 78.66%, compared to only 35.51% when only three of five samples agree. This calibration property suggests that agreement level can effectively identify cases where self-reference is likely to help versus harm judgment quality.

Building on this insight, we propose Selective Self-Reference (SSR-Judge), a method that uses self-consistency to gate the use of self-reference in LLM-as-a-Judge evaluation. SSR-Judge first generates multiple independent solutions through temperature sampling, then applies an agreement

[1] https://gitlab.com/fars-a/selective-self-reference-judge

threshold to decide whether to provide the self-answer as reference. When agreement is high, indicating confident and likely correct self-answers, the method uses self-reference; otherwise, it falls back to no-reference judging to avoid error propagation.

This work makes three contributions. First, we identify the error propagation problem in self-reference judging, demonstrating that incorrect self-answers significantly degrade judgment accuracy on uncertain items. Second, we propose SSR-Judge, a selective self-reference method that uses self-consistency as a confidence gate to conditionally enable self-reference. Third, we demonstrate that SSR-Judge achieves 58.93% preference accuracy on MMLU-Pro, outperforming both no-reference (52.07%) and always self-reference (58.21%) baselines.

2 Related Work

LLM-as-a-Judge. Using LLMs as automated evaluators has become a standard approach for assessing model outputs (Zheng et al., 2023; Li et al., 2024). MT-Bench and Chatbot Arena (Zheng et al., 2023; Chiang et al., 2024) established pairwise comparison frameworks, while G-Eval (Liu et al., 2023) introduced chain-of-thought evaluation prompts. PandaLM (Wang et al., 2023) and AlpacaEval (Dubois et al., 2024) provide automated evaluation benchmarks. These works focus on improving judge accuracy through better prompting or training, but do not address the error propagation problem that arises when judges use their own answers as references.

Self-Reference and Self-Consistency. Lin et al. (2025) introduced self-reference-guided evaluation, where the judge first solves the problem and uses its answer as a reference when evaluating candidates. While this improves accuracy when the self-answer is correct, they note the risk of error propagation when the self-answer is wrong. Self-consistency (Wang et al., 2022) demonstrates that agreement among multiple reasoning paths correlates with answer correctness, and Kadavath et al. (2022) show that LLMs can estimate their own uncertainty through sampling. SelfCheckGPT (Manakul et al., 2023) uses sampling-based consistency for hallucination detection. Our work connects these threads by using self-consistency to decide when self-reference should be applied in judging.

Judge Reliability. LLM judges exhibit various biases and inconsistencies. Wataoka et al. (2024) identify self-preference bias where judges favor their own outputs. Haldar & Hockenmaier (2025) document self-inconsistency across repeated evaluations. TrustJudge (Wang et al., 2025) proposes probabilistic scoring to reduce inconsistencies, while Tian et al. (2025) diagnose overconfidence in LLM judges and propose confidence-driven solutions. JudgeBench (Tan et al., 2024) provides a benchmark for evaluating judge correctness. Our work addresses a complementary failure mode: error propagation from incorrect self-references, which we mitigate through confidence-gated self-reference.

3 Method

We present SSR-Judge (Selective Self-Reference), a method that uses self-consistency to selectively enable self-reference in LLM-as-a-Judge evaluation. The core insight is that agreement among multiple self-solve samples serves as a reliable confidence signal: when the judge model consistently produces the same answer, that answer is more likely to be correct and can safely be used as a reference for evaluation.

3.1 Problem Formulation

We consider the pairwise preference judgment task, where a judge model M_J must determine which of two candidate responses better answers a given question. Formally, given a question Q with answer options and two candidate responses (R_1, R_2), the judge outputs a preference $p \in \{1, 2\}$ indicating which response is superior.

In the *self-reference* paradigm (Lin et al., 2025), the judge first solves the question independently to obtain a reference answer $\hat{A}$, then evaluates the candidates with this reference as additional context. This approach can improve judgment accuracy when $\hat{A}$ is correct, as the judge has an explicit anchor for comparison. However, when $\hat{A}$ is incorrect, the judge may anchor to this wrong reference and penalize correct candidates that disagree with it—a phenomenon we term *error propagation*.

3.2 SELF-SOLVE SAMPLING

To estimate the reliability of the judge's self-answer, we generate k independent self-solve samples using temperature sampling. Given question Q, we sample k responses $\{r_1, r_2, \ldots, r_k\}$ from the judge model M_J with temperature $T > 0$, and extract the final answer from each response. This sampling strategy follows the self-consistency principle (Wang et al., 2022), which demonstrates that agreement among diverse reasoning paths correlates with answer correctness.

We compute the *agreement level* as the fraction of samples that share the most common answer:

$$\text{Agreement}(r_1, \ldots, r_k) = \frac{\max_a |\{i : \text{answer}(r_i) = a\}|}{k} \tag{1}$$

where $\text{answer}(r_i)$ extracts the final answer from sample r_i. Higher agreement indicates greater model confidence in its answer, which prior work has shown to be a well-calibrated proxy for correctness (Kadavath et al., 2022).

3.3 AGREEMENT GATE

We define a binary gate function g that determines whether to use self-reference based on the agreement level:

$$g(r_1, \ldots, r_k) = \mathbf{1}[\text{Agreement}(r_1, \ldots, r_k) \geq \tau] \tag{2}$$

where τ is a threshold parameter. When $g = 1$ (gate fires), we consider the self-answer reliable and use it as a reference. When $g = 0$ (gate does not fire), we abstain from self-reference to avoid potential error propagation.

The threshold τ controls a precision-coverage tradeoff. A higher threshold (e.g., requiring unanimous agreement) yields higher precision—items passing the gate are more likely to have correct self-answers—but lower coverage, as fewer items pass the gate. We analyze this tradeoff empirically in Section 4.2.

3.4 CONDITIONAL JUDGING

Based on the gate output, SSR-Judge follows one of two judging paths (Figure 1):

Self-reference path ($g = 1$): When the gate fires, we compute the majority answer $\hat{A}_{\text{maj}}$ from the k samples and provide it as a reference to the judge. The judge evaluates the candidates with the prompt: "Given question Q, the correct answer is $\hat{A}_{\text{maj}}$. Which response is better: R_1 or R_2?"

No-reference path ($g = 0$): When the gate does not fire, we discard the self-solve samples and prompt the judge without any reference: "Given question Q, which response is better: R_1 or R_2?"

This selective mechanism achieves the best of both worlds: on high-agreement items where the self-answer is likely correct, SSR-Judge leverages self-reference to improve judgment accuracy. On low-agreement items where the self-answer is unreliable, SSR-Judge falls back to no-reference judging to avoid error propagation. To mitigate position bias in pairwise evaluation (Zheng et al., 2023), we evaluate each item twice with swapped response order and take the majority vote.

4 EXPERIMENTS

We evaluate SSR-Judge on MMLU-Pro, a challenging multi-domain benchmark, comparing it against two baselines: no-reference judging and always-self-reference judging.

4.1 EXPERIMENTAL SETUP

Dataset. We use MMLU-Pro (Wang et al., 2024), a more challenging extension of MMLU (Hendrycks et al., 2020) with 10 answer options per question (vs. 4 in MMLU). We sample 100 items from each of 14 categories, yielding $N = 1{,}400$ evaluation instances. For each item, we construct two candidate responses using a separate generator model: one concluding with the correct answer and one with a randomly sampled incorrect answer.

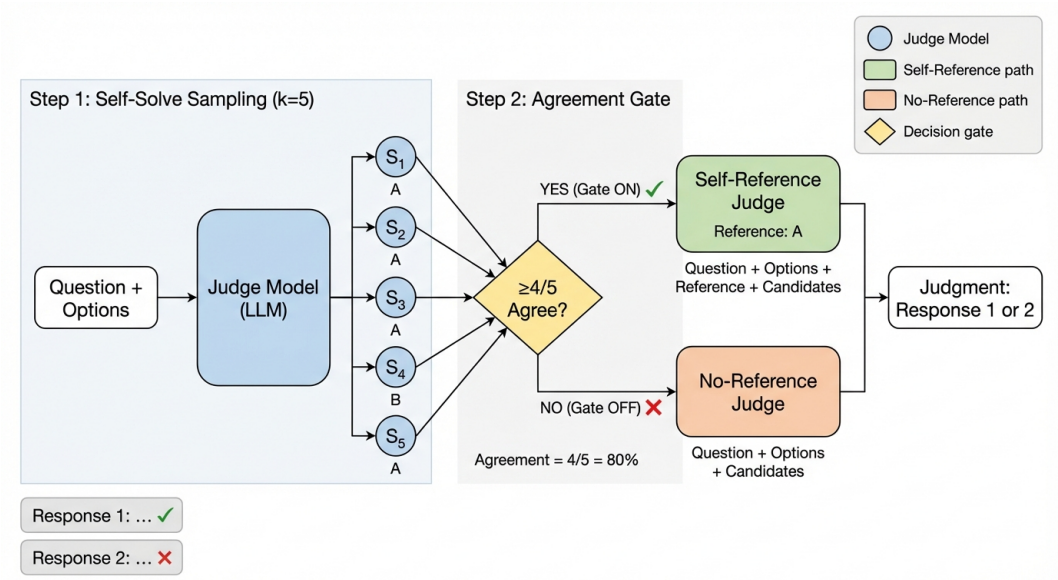

Figure 1: Overview of SSR-Judge. Given a question Q and two candidate answers, the judge first generates $k = 5$ self-solve samples. If $\geq 4/5$ samples agree on the same answer, the judge uses this consensus answer as a reference when evaluating the candidates (self-reference path). Otherwise, the judge evaluates without reference (no-reference path). This selective mechanism preserves the benefit of self-reference when the judge is confident while avoiding error propagation when uncertain.

Table 1: Main results on MMLU-Pro ($N = 1{,}400$). SSR-Judge achieves the highest overall accuracy while selectively applying self-reference based on self-solve agreement. Gate metrics computed using $\geq 4/5$ agreement threshold. Best results in **bold**.

Method	PreferenceAcc (%)	SliceAcc Gate-On (%)	SliceAcc Gate-Off (%)	Gate-On Rate (%)	Gate Precision (%)
No-Reference (A)	52.07	59.87	45.73	44.86	78.66
Always Self-Ref (B)	58.21	**72.77**	46.37	44.86	78.66
SSR-Judge (C, Ours)	**58.93** (+6.86)	68.76	**41.62**	**63.79**	72.45

Model. We use Qwen2.5-72B-Instruct (Yang et al., 2024) as the judge model M_J for all conditions. Self-solve samples are generated with temperature $T = 0.7$.

Hyperparameters. We use $k = 5$ self-solve samples and an agreement threshold of $\tau = 4/5$ (at least 4 out of 5 samples must agree for the gate to fire).

Baselines. We compare three conditions, all using the same $k = 5$ self-solve samples to ensure compute-matched comparison. No-Reference (A) discards self-solves and has the judge evaluate candidates directly. Always Self-Reference (B) always uses the majority self-answer as reference. SSR-Judge (C, Ours) uses self-reference only when agreement $\geq 4/5$.

Metrics. We report PreferenceAcc (overall accuracy), SliceAcc for gate-on and gate-off subsets, Gate-On Rate (fraction of items where gate fires), and Gate Precision (fraction of gate-on items with correct majority answer).

4.2 Main Results

Table 1 presents the main results. SSR-Judge achieves 58.93% PreferenceAcc, outperforming both No-Reference (52.07%, +6.86pp) and Always Self-Reference (58.21%, +0.72pp). The improvement over No-Reference demonstrates the value of self-reference when applied selectively, while the improvement over Always Self-Reference shows the benefit of avoiding error propagation on low-confidence items.

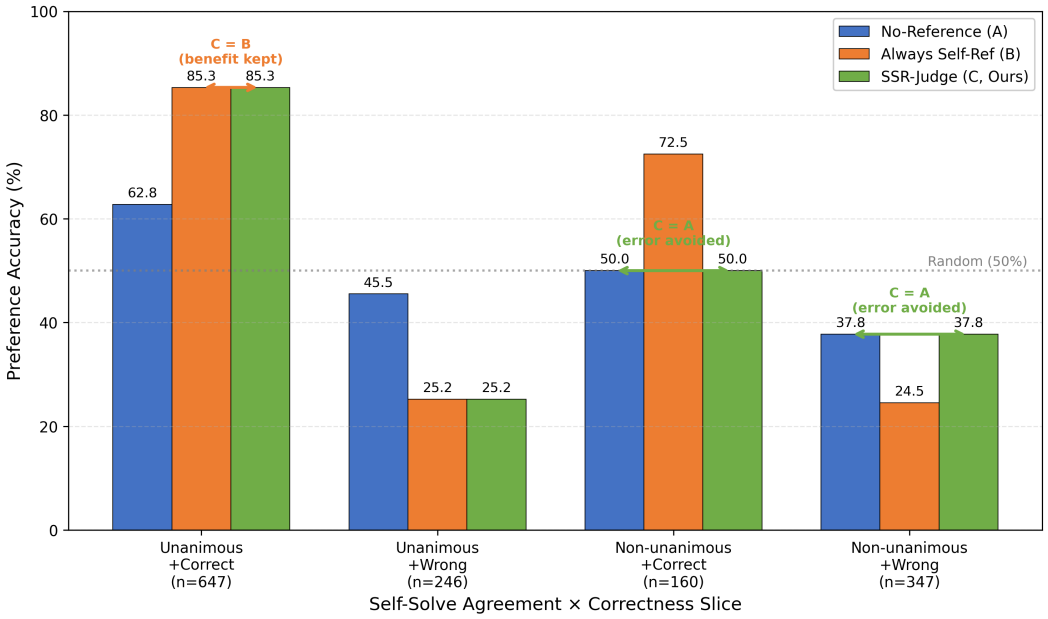

Figure 2: Preference accuracy across four slices defined by self-solve agreement (unanimous vs. non-unanimous) and majority correctness (correct vs. wrong). SSR-Judge (green) matches Always Self-Reference (orange) on unanimous slices where self-reference is beneficial, while matching No-Reference (blue) on non-unanimous slices where self-reference would cause error propagation. Sample sizes shown in parentheses.

Table 2: Error propagation analysis across four slices. SSR-Judge matches Always Self-Ref on unanimous slices (preserving benefit) and matches No-Reference on non-unanimous slices (avoiding error propagation). Best per slice in **bold**.

Slice	No-Ref (A)	Always Self-Ref (B)	SSR-Judge (C)	C matches
Unanimous + Correct ($n = 647$, 46.2%)	62.75	**85.32**	**85.32**	B ✓
Unanimous + Wrong ($n = 246$, 17.6%)	**45.53**	25.20	25.20	B ✓
Non-unan. + Correct ($n = 160$, 11.4%)	50.00	**72.50**	50.00	A ✓
Non-unan. + Wrong ($n = 347$, 24.8%)	**37.75**	24.50	**37.75**	A ✓

The gate characteristics reveal the precision-coverage tradeoff. SSR-Judge's $\geq 4/5$ threshold yields a higher Gate-On Rate (63.79% vs. 44.86% for strict unanimity) at the cost of slightly lower Gate Precision (72.45% vs. 78.66%). This tradeoff is favorable: the increased coverage allows more items to benefit from self-reference while maintaining sufficient precision to avoid excessive error propagation.

4.3 Error Propagation Analysis

To understand why SSR-Judge outperforms both baselines, we partition items into four slices based on agreement level (unanimous vs. non-unanimous) and majority answer correctness (correct vs. wrong). Figure 2 and Table 2 present the results.

The results validate SSR-Judge's selective mechanism. On unanimous slices, SSR-Judge matches Always Self-Reference, preserving the substantial benefit of self-reference when the majority answer is correct (+22.57pp on Unanimous+Correct). On non-unanimous slices, SSR-Judge matches No-Reference, avoiding the error propagation that causes Always Self-Reference to drop 13.25pp below No-Reference on the Non-unanimous+Wrong slice (24.50% vs. 37.75%). The "Unanimous + Wrong" slice represents irreducible error: even with correct gate behavior, anchoring to a wrong reference hurts performance. However, this slice is relatively small (17.6%) and the gate correctly identifies these items as high-confidence, where self-reference is the intended behavior.

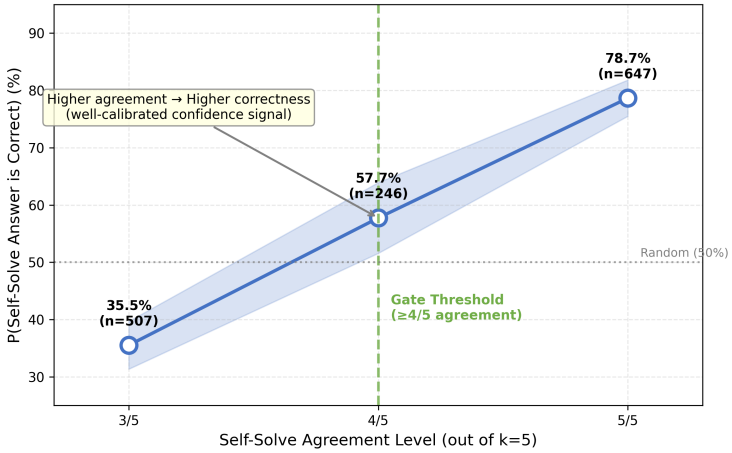

Figure 3: Calibration of self-solve agreement as a correctness proxy. Higher agreement levels correspond to higher probability that the majority answer is correct: 35.5% at 3/5 agreement, 57.7% at 4/5, and 78.7% at 5/5 (unanimous). The dashed line marks the gate threshold ($\geq 4/5$ agreement). Shaded region shows 95% confidence interval.

4.4 Calibration Analysis

The success of SSR-Judge depends on self-consistency being a reliable confidence signal. Figure 3 shows the relationship between agreement level and majority answer correctness. We observe a strong monotonic relationship: items with 3/5 agreement have only 35.5% majority correctness, items with 4/5 agreement have 57.7% correctness, and items with 5/5 (unanimous) agreement have 78.7% correctness. This calibration justifies using agreement as a gate for self-reference—higher agreement reliably indicates higher probability that the self-answer is correct and can safely be used as a reference.

4.5 Per-Category Analysis

SSR-Judge improves over No-Reference on 13 of 14 categories, with the largest gains in math (+22pp), business (+16pp), and chemistry (+15pp)—domains where self-reference provides substantial benefit. Compared to Always Self-Reference, SSR-Judge outperforms on 10 of 14 categories. The categories where Always Self-Reference wins (math, chemistry) tend to have high gate precision, meaning the self-answer is usually correct even when non-unanimous. Gate precision varies substantially across domains: STEM categories (physics 89%, math 86%) have high precision, while humanities categories (law 48%, philosophy 58%) have lower precision, suggesting that domain-specific threshold tuning could further improve performance.

5 Conclusion

We presented SSR-Judge, a selective self-reference method for LLM-as-a-Judge that uses self-consistency to gate the use of self-generated answers as reference. By generating multiple independent solutions and applying an agreement threshold, SSR-Judge identifies cases where self-reference is likely to help versus harm judgment quality. Our experiments on MMLU-Pro demonstrate that SSR-Judge achieves 58.93% preference accuracy, outperforming both no-reference (52.07%) and always self-reference (58.21%) baselines. The method successfully avoids error propagation on uncertain items while preserving the benefits of self-reference when the model is confident.

Our approach has limitations. The requirement for multiple samples increases computational cost, though this can be mitigated through parallel inference. Additionally, the optimal agreement threshold may vary across domains, suggesting that domain-specific calibration could further improve performance. Future work could explore adaptive thresholding strategies and extend the approach to other evaluation settings beyond pairwise preference judgment.

REFERENCES

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

Yann Dubois, Bal'azs Galambosi, Percy Liang, and Tatsunori Hashimoto. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *ArXiv*, abs/2404.04475, 2024.

Rajarshi Haldar and J. Hockenmaier. Rating roulette: Self-inconsistency in llm-as-a-judge frameworks. *ArXiv*, abs/2510.27106, 2025.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, D. Song, and J. Steinhardt. Measuring massive multitask language understanding. *ArXiv*, abs/2009.03300, 2020.

Saurav Kadavath, Tom Conerly, Amanda Askell, T. Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Z. Dodds, Nova Dassarma, Eli Tran-Johnson, Scott Johnston, S. El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, John Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom B. Brown, Jack Clark, Nicholas Joseph, Benjamin Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. Language models (mostly) know what they know. *ArXiv*, abs/2207.05221, 2022.

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

Wei-Hsiang Lin, Sheng-Lun Wei, Hen-Hsen Huang, and Hsin-Hsi Chen. Do before you judge: Self-reference as a pathway to better llm evaluation. *ArXiv*, abs/2509.19880, 2025.

Yang Liu, Dan Iter, Yichong Xu, Shuo Wang, Ruochen Xu, and Chenguang Zhu. G-eval: Nlg evaluation using gpt-4 with better human alignment. *ArXiv*, abs/2303.16634, 2023.

Potsawee Manakul, Adian Liusie, and M. Gales. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. *ArXiv*, abs/2303.08896, 2023.

Sijun Tan, Siyuan Zhuang, Kyle Montgomery, William Y. Tang, Alejandro Cuadron, Chenguang Wang, Raluca A. Popa, and Ion Stoica. Judgebench: A benchmark for evaluating llm-based judges. *ArXiv*, abs/2410.12784, 2024.

Zailong Tian, Zhuoheng Han, Yanzhe Chen, Haozhe Xu, Xi Yang, Richeng Xuan, Houfeng Wang, and Lizi Liao. Overconfidence in llm-as-a-judge: Diagnosis and confidence-driven solution, 2025. URL https://arxiv.org/abs/2508.06225.

Xuezhi Wang, Jason Wei, D. Schuurmans, Quoc Le, Ed H. Chi, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *ArXiv*, abs/2203.11171, 2022.

Yidong Wang, Zhuohao Yu, Zhengran Zeng, Linyi Yang, Cunxiang Wang, Hao Chen, Chaoya Jiang, Rui Xie, Jindong Wang, Xingxu Xie, Wei Ye, Shi-Bo Zhang, and Yue Zhang. Pandalm: An automatic evaluation benchmark for llm instruction tuning optimization. *ArXiv*, abs/2306.05087, 2023.

Yidong Wang, Yunze Song, Tingyuan Zhu, Xuanwang Zhang, Zhuohao Yu, Hao Chen, Chiyu Song, Qiufeng Wang, Cunxiang Wang, Zhen Wu, Xinyu Dai, Yue Zhang, Wei Ye, and Shikun Zhang. Trustjudge: Inconsistencies of llm-as-a-judge and how to alleviate them. *ArXiv*, abs/2509.21117, 2025.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max W.F. Ku, Kai Wang, Alex Zhuang, Rongqi "Richard" Fan, Xiang Yue, and Wenhu Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *ArXiv*, abs/2406.01574, 2024.

Koki Wataoka, Tsubasa Takahashi, and Ryokan Ri. Self-preference bias in llm-as-a-judge. *ArXiv*, abs/2410.21819, 2024.

Qwen An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxin Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yi-Chao Zhang, Yunyang Wan, Yuqi Liu, Zeyu Cui, Zhenru Zhang, Zihan Qiu, Shanghaoran Quan, and Zekun Wang. Qwen2.5 technical report. *ArXiv*, abs/2412.15115, 2024.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, E. Xing, Haotong Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. *ArXiv*, abs/2306.05685, 2023.