

League of LLMs: A Benchmark-Free Paradigm for Mutual Evaluation of Large Language Models

Qianhong Guo^{1,*}, Wei Xie^{1,*†}, Xiaofang Cai², Enze Wang¹, Shuoyoucheng Ma¹,
Xiaobing Sun³, Tian Xia¹, Kai Chen², Xiaofeng Wang¹, Baosheng Wang^{1,†}

¹College of Computer Science and Technology, National University of Defense Technology,

²Institute of Information Engineering, Chinese Academy of Sciences,

³Institute of High Performance Computing, A*STAR

* Equal Contribution. † Correspondence: {xiewei, bswang}@nudt.edu.cn

Code is available at <https://github.com/Qhovo1/League-of-LLMs>

Abstract

Although large language models (LLMs) have shown exceptional capabilities across a wide range of tasks, reliable evaluation remains a critical challenge due to data contamination, opaque operation, and subjective preferences. To address these issues, we propose League of LLMs (LOL), a novel benchmark-free evaluation paradigm that organizes multiple LLMs into a self-governed league for multi-round mutual evaluation. LOL integrates four core criteria (dynamic, transparent, objective, and professional) to mitigate key limitations of existing paradigms. Experiments on eight mainstream LLMs in mathematics and programming demonstrate that LOL can effectively distinguish LLM capabilities while maintaining high internal ranking stability (Top- k consistency = 70.7%). Beyond ranking, LOL reveals empirical findings that are difficult for traditional paradigms to capture. For instance, “memorization-based answering” behaviors are observed in some models, and higher in-family scores are found in the OpenAI model family ($\Delta = 9$, $p < 0.05$). Finally, we make our framework and code publicly available as a valuable complement to the current LLM evaluation ecosystem.

1 Introduction

Large language models (LLMs) have recently exhibited exceptional capabilities across a wide range of scenarios (Achiam et al., 2023; Team et al., 2023; Touvron et al., 2023; Ge et al., 2023; Vaswani et al., 2017), including scientific research assistance (Boyko et al., 2023), code generation (Nejjar et al., 2025), and educational question answering (Kasneci et al., 2023). As model scale and application scope expand, their capability differences across tasks have become increasingly apparent (Chang et al., 2024). While they excel at tasks such as mathematical reasoning (Ahn et al., 2024; Azerbayev et al., 2023; Luo et al., 2023; Lu et al., 2023;

Yu et al., 2023) and code generation (Xu et al., 2022), their capabilities remain uneven across different domains. Some models perform strongly in logical reasoning, while others are more proficient in engineering implementation or language understanding. These differences not only impact the selection of LLMs for practical applications but also raise the bar for rigorous comparison and development. Consequently, there is an urgent need for a series of systematic, objective, and discriminative evaluation paradigms.

We categorize mainstream LLM evaluation into three paradigms: white-box evaluation, black-box evaluation, and human-crowdsourced evaluation. While these paradigms exhibit respective advantages in certain aspects, they also suffer from distinct limitations. Most mainstream benchmarks belong to white-box evaluation, which relies on publicly available static datasets, offering strong reproducibility and transparency. However, long-term public datasets increase the risk of data contamination: LLMs may memorize test instances through prior training exposure (Cheng et al., 2025; Xu et al., 2024; Deng et al., 2023), undermining the credibility of evaluation results (Sainz et al., 2023). Black-box evaluation mitigates the risk of data contamination by concealing test datasets, but its opaque operation makes the evaluation difficult to verify and reproduce, limiting credibility (Casper et al., 2024). Human-crowdsourced evaluation establishes rankings through user preference voting (Chiang et al., 2024). While it reduces leakage risks associated with fixed benchmarks, it heavily relies on users’ subjective preferences to ask questions and evaluate answers. Many of these users cannot be assumed to have expert-level domain knowledge (Moore et al., 2024), making it difficult to ensure objective and professional evaluation (Li et al., 2024b). Together, these limitations motivate a new evaluation paradigm that is dynamic, transparent, objective, and professional.

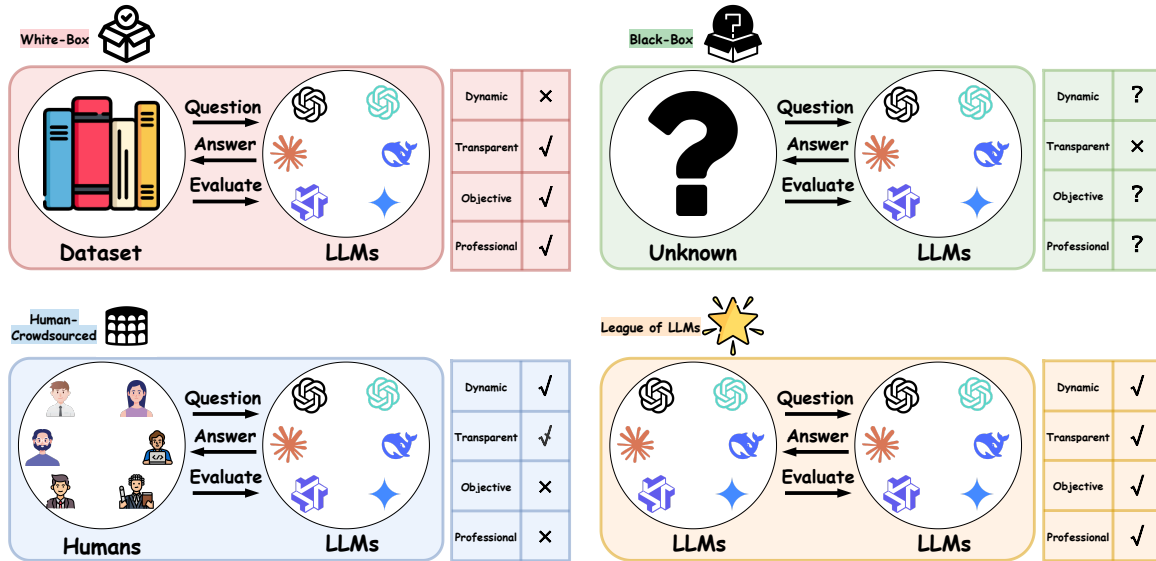


Figure 1: **Mainstream LLM evaluation paradigms vs. League of LLMs (LOL).** Compared under four core criteria: Dynamic, Transparent, Objective, and Professional.

We introduce League of LLMs (LOL), a novel benchmark-free evaluation paradigm. LOL organizes multiple LLMs into a self-governed league, where they compete for leaderboard positions across multiple rounds. In each round, LLMs take turns (i) generating questions, (ii) answering independently, and (iii) mutually evaluating one another, with the final ranking computed by aggregating the resulting scores.

LOL offers several advantages that existing evaluation paradigms rarely provide simultaneously. By dynamically generating questions, it reduces the overlap between fixed benchmarks and training data, thereby mitigating data contamination. The evaluation process is also designed to be traceable and open to scrutiny, enhancing transparency as well as reproducibility. Furthermore, by aggregating judgments from multiple LLMs rather than relying on a single evaluator or human preference voting, LOL can mitigate the influence of subjective preferences. Both question generation and evaluation are performed by LLMs, which can exhibit more specialized domain knowledge than most crowd users. Thus, LOL can better support more professional and objective evaluation results.

To validate the proposed League of LLMs (LOL) evaluation paradigm, we systematically evaluate eight mainstream LLMs (e.g., gpt-4.1, gemini-2.5-pro-exp, deepseek-r1) across two representative domains: mathematics and programming. The evaluation results indicate that LOL differentiates LLM capabilities while yielding stable rankings across

runs (Top- k consistency = 70.7%), supporting the robustness and reliability of the resulting rankings. Beyond ranking, LOL surfaces empirical findings that are rarely captured by traditional paradigms in practice. We observe “memorization-based answering” behaviors in some models. Moreover, when LLMs are grouped by developer families, higher in-family scores are found in the OpenAI model family, where in-family scores are higher than out-of-family scores ($\Delta = 9, p < 0.05$).

Our contributions are summarized as follows:

1. Evaluation Paradigm: We propose League of LLMs (LOL), a novel benchmark-free evaluation paradigm based on multi-LLM evaluation to mitigate key limitations of existing paradigms.

2. Empirical Findings: We use LOL to systematically evaluate eight LLMs in mathematics and programming, yielding discriminative capability comparisons and revealing empirical findings that traditional paradigms struggle to capture, including “memorization-based answering” behaviors and higher in-family scores in the OpenAI model family.

3. Community Value: We publicly release our framework and code to facilitate evaluation of new LLMs and to serve as a valuable complement to the current LLM evaluation ecosystem.

2 Motivation

The evaluation of LLMs aims to objectively assess model capabilities and establish rankings, providing guidance for researchers and users. Main-

stream LLM evaluation can be categorized into three paradigms: white-box evaluation, black-box evaluation, and human-crowdsourced evaluation.

2.1 White-Box

White-box evaluation is a widely used paradigm. It relies on publicly available datasets with reference answers, offering strong transparency and reproducibility. Representative benchmarks include HELM (Liang et al., 2023), MATH (Hendrycks et al., 2021), GSM8K (Cobbe et al., 2021), HumanEval (Chen et al., 2021), LiveCodeBench (Jain et al., 2024), and MMLU (Hendrycks et al., 2020).

However, since test datasets are fixed and public over long periods, white-box evaluation increasingly suffers from data contamination: LLMs may have encountered test instances during training, undermining evaluation credibility (Yang et al., 2023). Relevant studies also indicate that even minor alterations or rephrasing can significantly reduce the accuracy of LLMs’ answers and cause LLMs to reuse the original answers, exhibiting “memorization” rather than genuine reasoning (Mirzadeh et al., 2024; Li et al., 2024a; Xie et al., 2024).

2.2 Black-Box

To mitigate data contamination, black-box evaluation uses newly collected or non-public test datasets. For example, Huang (Huang et al., 2023) uses real-time competition data. GPQA (Rein et al., 2024) utilizes a newly constructed expert-written question bank. SuperGLUE (Wang et al., 2019) only returns aggregate scores via a closed server.

Although black-box evaluation improves confidentiality, its opaque operation limits verifiability and auditability. Since the test data and evaluation process are hidden, the evaluation results are difficult to verify by third parties, which weakens credibility, interpretability, and reproducibility (Casper et al., 2024). Furthermore, black-box datasets may still originate from publicly available internet data or be leaked. As long as the datasets remain fixed, the risk of data contamination persists.

2.3 Human-Crowdsourced

To reduce data contamination from fixed test datasets, Chatbot Arena (Chiang et al., 2024) is proposed as a human-preference-based crowdsourced evaluation platform. This platform adopts pairwise comparisons: two LLMs answer the same user question, users select the better answer, and the

system aggregates pairwise outcomes to produce an overall ranking.

Despite its inherent dynamism, this paradigm has several limitations. First, it relies heavily on users’ subjective preferences (e.g., longer or containing emojis tend to be favored (Chen et al., 2025; Li et al., 2024b)). Moreover, user-generated questions vary widely in quality, and most users cannot be assumed to have expert-level domain knowledge. Finally, although Chatbot Arena discloses its ranking rules and metric computation methods, its core raw data (user inputs, LLM outputs, and evaluation records) remain non-public, leaving key parts of the evaluation process opaque and difficult to independently verify or audit.

2.4 Evaluation Criteria

In summary, we argue that scientific LLM evaluation should satisfy four core criteria jointly.

1. Dynamic: Each evaluation should use questions and answers with sufficient freshness and diversity to mitigate data contamination.

2. Transparent: Evaluation should be open and transparent, with reproducible processes and results, thereby enhancing credibility.

3. Objective: Evaluation standards and results should not be influenced by individual subjective preferences, thereby reducing bias and the possibility of manipulation, and reflecting the objective performance of LLMs.

4. Professional: Questions and answers should be at or near the average level of human experts, enabling in-depth evaluations of LLMs’ professional capabilities in vertical domains.

3 League of LLMs

As shown in Figure 1, white-box evaluation (based on public datasets) suffers from insufficient dynamism, while black-box evaluation (based on unknown datasets) lacks transparency. Human-crowdsourced evaluation (based on human evaluators) is influenced by subjective preferences, incomplete transparency, and limited professionalism. To mitigate these limitations, we propose LOL, a benchmark-free evaluation paradigm in which multiple LLMs generate questions, answer independently, and evaluate mutually, with final rankings computed by aggregating scores across rounds.

3.1 Evaluation Pipeline

As illustrated in Figure 2, our paradigm comprises four phases. Before the evaluation begins, it is

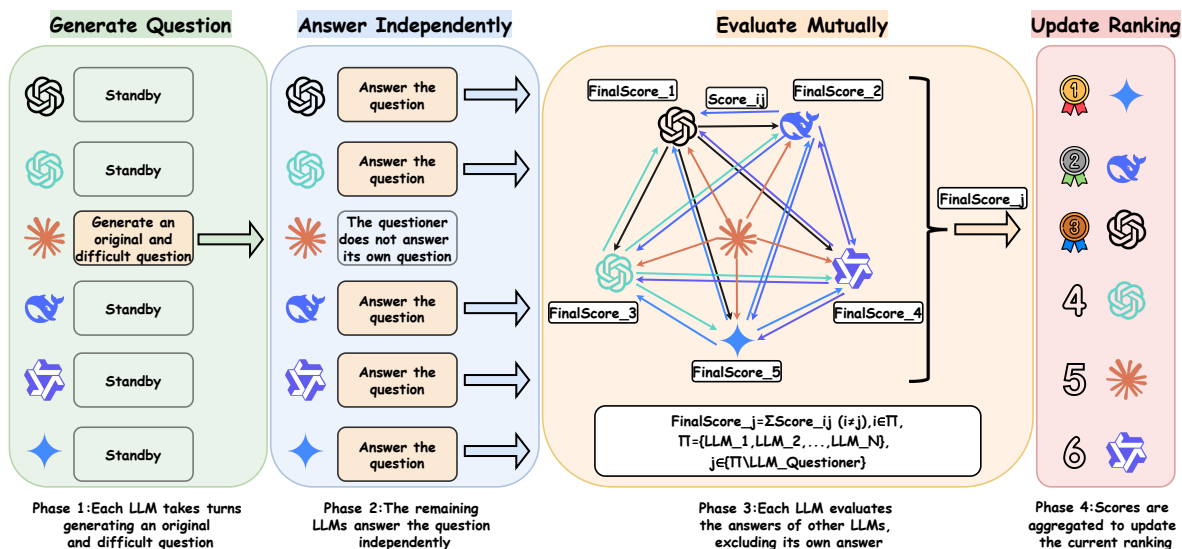


Figure 2: **Overview of the League of LLMs evaluation pipeline.** It consists of four phases: Generate Question, Answer Independently, Evaluate Mutually, and Update Ranking.

necessary to predefine the evaluation domain and rules to ensure controllability and consistency in both question generation and evaluation processes.

Phase 1: Generate Question Each LLM takes turns acting as the questioner, generating an original and challenging question according to the predefined rules, and providing a reference answer. Subsequently, this question is distributed to the other LLMs for answering.

Phase 2: Answer Independently Except for the questioner, the remaining LLMs independently answer the question without referencing others' answers, thereby evaluating their independent question-solving abilities.

Phase 3: Evaluate Mutually The questioner LLM provides the reference answer for the other LLMs. Based on the reference answer and the predefined scoring criteria, each LLM evaluates the answers of other LLMs (excluding its own), enabling mutual evaluation among the LLMs.

Phase 4: Update Ranking The mutual evaluation results for the current round are aggregated to update the real-time ranking. This ranking serves as periodic feedback and provides the basis for subsequent evaluation. After n rounds, the final ranking is output.

3.2 Advantages

The proposed League of LLMs (LOL) evaluation paradigm integrates the strengths of existing paradigms while mitigating their inherent limitations, aiming to establish a more scientific and standardized evaluation paradigm.

In terms of dynamism, evaluation questions are dynamically generated by LLMs based on predefined rules, and evaluation is conducted under predefined scoring criteria. This mechanism mitigates data contamination over time, thereby improving the long-term validity of the evaluation.

In terms of transparency, the processes of question generation, answering, and evaluation are all fully open and traceable. External researchers can independently verify the results, thereby enhancing the interpretability and credibility of the evaluation.

In terms of objectivity, LOL employs a decentralized mutual evaluation mechanism, in which multiple LLMs jointly perform the evaluation task. This reduces the influence of any single evaluator's bias and human subjective preferences (Chen et al., 2024; Zhang et al., 2025), enhancing the fairness and stability of the evaluation outcomes.

In terms of professionalism, compared to most crowd users, LLMs can exhibit more professional domain knowledge, enabling them to deliver more specialized and higher-quality questions, answers, and scores in most evaluation tasks.

3.3 Overview of Experiment Setups

The LOL evaluation paradigm is highly extensible. Its components, such as LLM selection, task domains, scoring mechanisms, and evaluation pipeline, can be flexibly replaced to accommodate different task requirements. Moreover, using eight questions per round is only a lightweight experimental setting. It can be easily scaled by increasing the number of questions or rounds, without human

scoring or adjudication.

For instance, in this paper, we validate LOL in two representative domains: mathematics and programming. Within a unified evaluation paradigm, mathematics tasks adopt the Borda rule (Borda, 1781), while programming tasks employ absolute scores (details in Section 4.1 and 5.1), thereby demonstrating the paradigm’s generality.

Considering that LLMs may exhibit biases such as self-preference or conformity during evaluation (Liu et al., 2023; Ye et al., 2024), we introduce three mechanisms: independent contexts, prohibition of self-evaluation, and multi-LLM mutual evaluation based on reference answers, which further enhance fairness and robustness.

All main experiments were conducted using Huiyan’s unified API (Huiyan, 2023), with temperature set to 1 and all other parameters set to their defaults. We selected eight mainstream LLMs ($n = 8$) as follows: gpt-4.1-2025-04-14, o3-mini, o1, claude-3-7-sonnet-20250219, deepseek-r1, deepseek-v3-0324, qwen2.5-max, and gemini-2.5-pro-exp-03-25. For each domain, we ran 5 independent experimental sets. Each set contained 8 questions \times 7 answers \times 7 evaluations, totaling 1,960 data points per domain, comparable in scale to mainstream benchmarks such as MATH500.

4 League of LLMs On Mathematics

4.1 Experiment Design

Question Generation Mechanism: To achieve in-depth evaluation and to mitigate data contamination caused by static question banks, we design a professional and dynamic question generation phase (prompts are provided in Appendix G.1). The questioner LLM is tasked with acting as a mathematician, identifying the most difficult questions based on its prior knowledge, abstracting the core mathematical principles, and constructing an original and challenging question with a reference answer.

Evaluation Mechanism: Given that mathematical questions often admit multiple valid solution paths, we adopt the Borda rule to reduce subjectivity in absolute scoring. This rule maps the rank order of answers to discrete scores (1st: 6 points, 2nd: 5 points, ..., 7th: 0 points), thereby enabling a more stable and objective comparison.

4.2 Experiment Results

Figure 3 illustrates each LLM’s scores on questions, reflecting their overall performance. Visu-

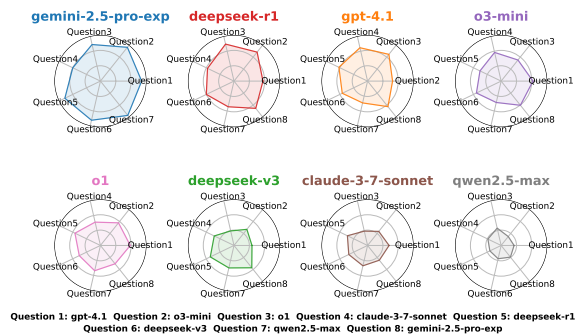


Figure 3: **Radar charts of LLM performance on mathematics tasks.** They display the mean Borda scores of eight LLMs on eight questions, enabling comparison of their overall performance.

ally, gemini-2.5-pro-exp covers the largest area, whereas qwen2.5-max covers the smallest.

To quantify this visual difference, a one-way ANOVA reveals significant differences among the LLMs ($F(7, 1952) = 118.71, p < 0.001$). Tukey post-hoc analysis further shows that gemini-2.5-pro-exp achieves the highest mean score and differs significantly from all other models ($p < 0.001$), whereas qwen2.5-max scores are significantly lower than all other models ($p < 0.001$). In addition, only four model pairs exhibit no significant differences (deepseek-r1 and gpt-4.1, gpt-4.1 and o3-mini, o3-mini and o1, deepseek-v3 and claude-3-7-sonnet). Except for these four pairs, the remaining comparisons are significant, indicating that LOL distinguishes mathematical capabilities among most models. Statistical details are provided in Appendix A.

Finding 1: LLMs exhibit substantial variation in mathematical question generation ability, with some reaching professional-level performance.

We observe significant differences among LLMs in their ability to generate mathematical questions. Some LLMs, such as gemini-2.5-pro-exp, demonstrate professional-level question generation ability, as shown below:

Example question

Let $S_3(n)$ denote the sum of the digits of a positive integer n when written in base 3. Define a sequence $a_n = (-1)^{S_3(n)}$, for $n \geq 1$. Consider the Dirichlet series $F(s) = \sum_{n=1}^{\infty} \frac{a_n}{n^s}$, defined for complex numbers s where the series converges. Determine if $F(s)$ can be analytically continued to a region including $s = 0$, and if so, find the value $F(0)$.

The generated question integrates elements from number systems, automatic sequences, and analytic continuation in complex analysis, exhibiting a high degree of novelty for benchmark-based evaluation. While its construction draws on classic Dirichlet-series constructions in modern number theory (e.g., the Riemann zeta function), the specific combination and instantiation are unlikely to appear in typical public benchmark datasets or human-crowdsourced evaluations. More details can be found in Appendix F.1. Although most LLMs failed to provide the correct answer, the resulting performance differentiation instead provides evidence for the effectiveness of the “prior-knowledge-based question generation mechanism” described in Section 4.1. This mechanism not only distinguishes capability differences among LLMs but also highlights the advantages of LOL in terms of professionalism and dynamism.

Finding 2: LLMs exhibit “memorization-based answering” behavior rather than genuine reasoning in some cases.

We observe that some LLMs do not perform genuine reasoning when answering questions. Instead, they rely on memorization for template matching. We define this phenomenon as “memorization-based answering”: the LLM mistakenly treats the current question as a classic question with a similar template but different details and directly applies memorized answers or conclusions.

For instance, in the Dirichlet series question above, deepseek-r1 attempted to formalize $F(s)$ as an infinite product structure similar to the Euler product, thereby producing an incorrect result. This technique is commonly applicable to Dirichlet series with a multiplicative structure, such as the Riemann zeta function, but it does not apply to the current question. See Appendix F.2 for details.

Similar mistakes appeared 22 times among the 280 responses (7.9%) in experiments, highlighting a limitation of white-box (static benchmark) evaluation: LLMs may rely on template matching rather than genuine reasoning in these cases. In contrast, LOL can identify such behaviors and reveal the true boundaries of the LLM’s reasoning abilities.

Finding 3: Evaluation results exhibit high internal ranking stability.

When $k = n/2$, Top- k consistency reaches 70.7% (95% CI [0.700, 0.713]), suggesting stable internal rankings and a cross-LLM consensus judgment. Meanwhile, the standard deviations of the overall scores across runs remain low on a 100-

point scale (3.28 to 10.76). Together, these results quantitatively demonstrate that, even without human scoring or adjudication, mutual evaluation among LLMs can converge to stable and consistent results. More details are provided in Appendix D.

5 League of LLMs On Programming

5.1 Experiment Design

Question Generation Mechanism: To mitigate the limitation that existing benchmarks emphasize code correctness but overlook question-setting ability, we design a professional and dynamic question generation phase (prompts are provided in Appendix G.2). The questioner LLM acts as a senior programming competition question setter, designing professional-level, ACM-style original programming questions. Each question includes a description, input/output format, constraints, a reference answer, and a complexity analysis. Answering LLMs can freely choose the programming language, avoiding bias introduced by restrictions.

Evaluation Mechanism: To better reflect professionalism and objectivity, we adopt a 100-point scoring scheme. Each LLM evaluates others’ answers based on the reference answer and scoring criteria across multiple dimensions, including correctness, efficiency, readability, and modularity.

5.2 Experiment Results

Beyond the aggregate scores, LOL reveals two additional findings through mutual evaluation.

Finding 4: Question-solving and question-setting abilities are decoupled.

As shown in Figure 4, we conduct a two-dimensional analysis of eight LLMs’ overall performance in programming tasks: the average score a LLM receives on other LLMs’ questions (Own Average Score on Others’ Questions; higher is better) measures its question-solving ability, while the average score other LLMs receive on that LLM’s questions (Others’ Average Score on Own Question; lower indicates more difficult questions) reflects its question-setting ability. Statistical details are provided in Appendix B.

We observe that these two dimensions do not follow a simple linear relationship, but instead exhibit a clear decoupling. This indicates that LLM capability does not follow the common intuition that a stronger solver necessarily sets harder questions. It is similar to the idea that “a good programmer is not necessarily a good architect”: being skilled at

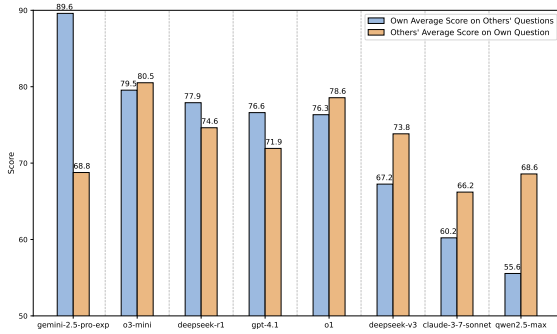


Figure 4: **Ability comparison of LLMs on programming tasks (question-solving vs. question-setting).** They are measured by Own Average Score on Others' Questions and Others' Average Score on Own Question.

solving questions does not necessarily imply being skilled at designing them. For example, although o3-mini performs well on solving others' questions, the questions it generates are among the easiest for other LLMs to answer, exhibiting a "strong solver, weak question setter" profile. In contrast, gemini-2.5-pro-exp maintains high performance on both dimensions, suggesting more balanced capabilities.

This question-setting dimension, automatically induced by the mutual evaluation paradigm, captures asymmetries in LLM cognition that are masked by a single benchmark score. Compared with static benchmarks that report only aggregate scores, this observation reveals deeper cognitive distinctions between LLMs and provides richer empirical evidence for capability comparisons.

Finding 5: Higher in-family scores in the OpenAI model family.

Based on an analysis grouped by developer families, we observe that OpenAI models (gpt-4.1, o3-mini, o1) assign an average score of 79.94 to answers from their own family, but only 70.94 to answers from outside the family ($\Delta = 9, p < 0.05$). We refer to this pattern as higher in-family scores in the OpenAI model family. This difference may arise from shared architectures, training data, or output styles, which may lead these models to favor answers from their own family. It may also be partially attributed to the inherently strong reasoning capabilities of the OpenAI models themselves. In contrast, DeepSeek models (deepseek-r1, deepseek-v3) show only a 0.85-point difference between in-family and out-of-family scores ($p > 0.05$), suggesting no notable gap. See Appendix E for more details. These results suggest that using a single OpenAI model (e.g., GPT-4) as the evaluator LLM

may unintentionally amplify this in-family scoring discrepancy. Meanwhile, LOL can reveal this pattern and mitigate its effect by reducing reliance on any single model.

6 League of LLMs vs. Well-Known Benchmarks

To verify the external validity of LOL, we compare our mathematics and programming rankings with several well-known benchmarks, including Chatbot Arena, MATH500, AIME2024, LiveCodeBench, and HumanEval. The benchmark rankings are sourced from the official Chatbot Arena website and the Artificial Analysis leaderboard (snapshots taken at the time of retrieval), and we compute Spearman correlation coefficients between LOL and each benchmark ranking.

As shown in Figure 5, the LOL mathematics ranking shows significant positive correlations with MATH500 ($\rho = 0.74, p < 0.05$) and with Chatbot Arena Math and AIME2024 ($\rho = 0.71, p < 0.05$). For programming tasks, it likewise remains highly consistent with LiveCodeBench ($\rho = 0.93, p < 0.05$) and HumanEval ($\rho = 0.81, p < 0.05$).

These results show that even without relying on human scoring or fixed benchmark datasets, LOL can still produce rankings that are highly consistent with well-known benchmarks, supporting the reliability of LOL's rankings. LOL aligns with the overall trend while still retaining subtle discrepancies, which may stem from differences in evaluation focus and preferences across setups, suggesting that LOL offers a complementary perspective.

7 Related Work

The LLM-as-a-Judge line of work (Zheng et al., 2023; Pan et al., 2024) is closest to our study. Its core idea is to use a single LLM as a judge to score the answers of other LLMs. However, our work differs fundamentally from this line of work in both goals and design.

LLM-as-a-Judge adopts a single-LLM judge, which may introduce systematic biases, such as self-preference and the higher in-family scores in the OpenAI model family that we empirically observe in Finding 5. In contrast, our study proposes a decentralized closed-loop paradigm based on multi-LLM mutual evaluation with dynamic question generation, which mitigates single-judge bias at the mechanism level. Moreover, while LLM-as-a-Judge evaluates model answers on fixed bench-

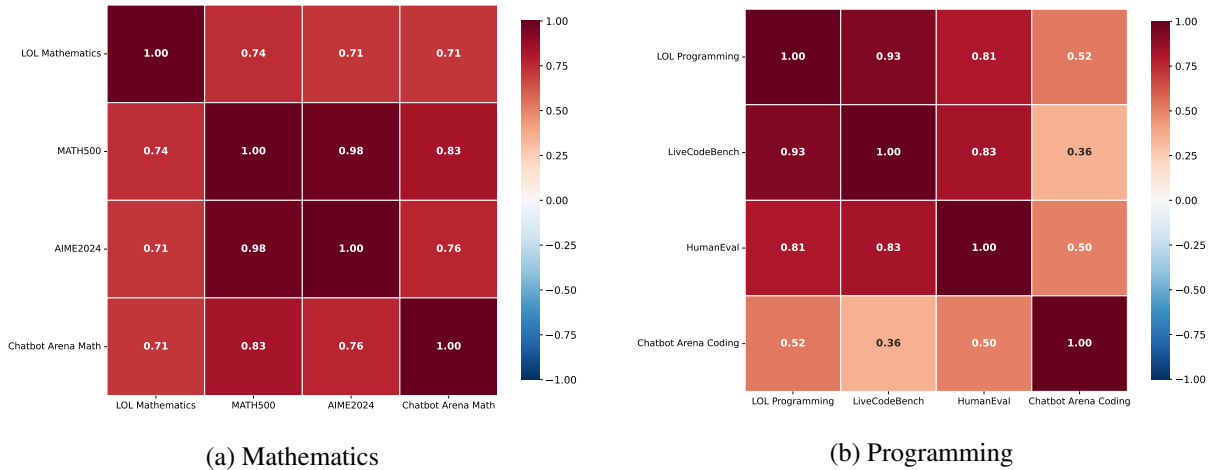


Figure 5: **Spearman correlation between LOL and Well-Known Benchmarks (Mathematics and Programming)**. The heatmaps report Spearman’s ρ among LOL and Well-Known Benchmarks (Mathematics: MATH500, AIME2024, Chatbot Arena Math; Programming: LiveCodeBench, HumanEval, Chatbot Arena Coding).

mark datasets, LOL integrates question generation, answering, and mutual evaluation into a single dynamic and self-consistent closed loop. As a result, LOL reduces reliance on fixed benchmark datasets by autonomously generating and updating its own test instances.

8 Discussion

Q1: Are the LLM-generated questions and reference answers reliable?

We require questioner LLMs to provide reference answers simultaneously, ensuring each question has a clear, verifiable answer. All questions and reference answers were validated independently by at least three human annotators without modification, and we did not identify any unsolvable or logically inconsistent items.

Moreover, LOL does not rely solely on a single reference answer: even if a reference answer is imperfect, multiple LLMs can independently evaluate answers, reducing reliance on any single reference. Meanwhile, with multiple LLMs rotating as questioners, even low-discrimination questions can be diluted over rounds through mutual evaluation. To further address concerns about potential answer dependency, we conducted an ablation by removing the reference answer from the evaluator prompt. Although absolute scores exhibit some fluctuation, the final model rankings remain largely consistent, preserving all main conclusions of the paper.

Empirically, robustness is supported by high Top- k consistency and statistical tests, as well as strong correlations with well-known benchmarks.

Together, these results suggest that the resulting rankings are reliable.

Q2: If the evaluation LLMs may have biases or errors, how can the results be ensured to be objective and reasonable?

We prohibit self-evaluation and adopt a “mutual evaluation with reference answer” mechanism to reduce bias. To further reduce idiosyncratic errors from any single evaluator, we aggregate judgments from multiple LLMs across multiple rounds. Even if misjudgments occur, errors can be mitigated through mutual evaluation.

We have performed five independent runs for each experiment, and the resulting rankings are highly consistent across runs, indicating that the evaluation is robust and reproducible. LLMs can exhibit more professional domain knowledge and objective evaluation standards than most crowd users. Notably, we position LOL as a valuable complement rather than a replacement for existing evaluation paradigms. While LLM evaluators may still be imperfect, LOL provides a scalable, benchmark-free paradigm for more objective and reproducible evaluation.

Q3: What is the operational cost of LOL, and how can its scalability be maintained under high computational complexity?

Table 1 reports the average token consumption for each model in a full run of the main experiment. The average cost per complete experiment (averaged across runs) is approximately \$1.87 for mathematics tasks and \$3.29 for programming tasks.

Given the computational cost of full mutual eval-

Model	Mathematics Input	Mathematics Output	Programming Input	Programming Output
gpt-4.1	27444	11501	94345	14107
o3-mini	29128	43270	94016	62587
o1	29501	61358	95051	94492
claude-3-7-sonnet	32273	15223	103860	27166
deepseek-r1	30568	146624	84495	300803
deepseek-v3	33422	18425	128434	11965
qwen2.5-max	28853	7260	96431	15559
gemini-2.5-pro-exp	26379	150988	96148	207968

Table 1: Average token consumption per model across runs for mathematics and programming tasks.

uation, we conduct a sparse Elo replay experiment to explore the feasibility of reducing complexity. Specifically, we randomly sample 1%–50% of comparisons from the full pairwise comparison pool and independently repeat the process 100 times to reconstruct the rankings. The results show that, even with only 30% of the comparisons, the recovered rankings remain highly consistent with those from full evaluation (mathematics: Spearman’s $\rho = 1.00$; programming: Spearman’s $\rho = 0.93$). This demonstrates that LOL can substantially reduce computational cost via sparse sampling while preserving high ranking reliability, thereby exhibiting strong scalability.

Q4: How is the originality of the generated questions verified?

We conduct an originality analysis by measuring both lexical and semantic similarity between the generated questions and widely used benchmarks, using 5-gram overlap for lexical similarity and sentence embeddings from all-MiniLM-L6-v2 (Wang et al., 2020) for semantic similarity. For mathematics, comparisons are conducted against a total of 9,322 questions from MATH500, AIME2024, and GSM8K, yielding an average 5-gram overlap of 5.11% and a semantic similarity of 0.4856; for programming (1,219 questions), comparisons against LiveCodeBench and HumanEval yield an average 5-gram overlap of 5.90% and a semantic similarity of 0.5730. These results indicate low lexical and semantic similarity to existing benchmarks, suggesting that the generated questions are not simple paraphrases of prior datasets.

Q5: How can repeated or highly similar questions be avoided in long-term operation?

To mitigate repetition, we introduce two mechanisms. A League Memory maintains an embedding-based memory bank of historical questions, where newly generated questions are compared against their Top- k nearest neighbors and those exceeding a similarity threshold are filtered and regenerated. In addition, the system dynamically schedules top-

ics through a seasonal rotation mechanism to diversify topic distribution and mitigate local repetition.

9 Conclusion

In this paper, we propose League of LLMs (LOL), a novel benchmark-free evaluation paradigm that organizes multiple LLMs into a self-governed league for multi-round mutual evaluation. LOL integrates four core criteria to mitigate key limitations of mainstream evaluation paradigms, including data contamination, opaque operation, and subjective preferences. Experiments on eight mainstream LLMs in mathematics and programming show that LOL can effectively differentiate LLM capabilities and produce highly consistent internal ranking across runs (Top- k consistency = 70.7%). Beyond ranking, LOL surfaces empirical findings that are difficult for traditional paradigms to capture in practice, including “memorization-based answering” and higher in-family scores in the OpenAI model family. We publicly release our framework and code to support the evaluation of newly released LLMs and to serve as a valuable complement to the current LLM evaluation ecosystem.

Limitations

Tasks and Models. Due to space limitations, we validate LOL only in mathematics and programming. Future work will extend LOL to more tasks (e.g., reasoning, dialogue, and open-ended writing) and more models.

Evaluator reliability and bias. LOL reduces reliance on a single evaluator model through multi-LLM mutual evaluation and prohibits self-evaluation. However, correlated errors may still arise when multiple models share similar preferences or blind spots. The reference answer mechanism may be less effective for open-ended tasks. In addition, although LOL does not require human scoring or adjudication, we still performed lightweight human validation of the generated ques-

tions and reference answers in our experiments as a sanity check. Therefore, future work may incorporate stronger automated verification and calibration, complemented by lightweight expert spot-checking, to further mitigate biases.

Scalability and cost. LOL supports continual extension. When adding a new model, mutual-evaluation outcomes can be converted into pairwise comparison signals and integrated via an online rating scheme such as Elo, allowing the model to enter the leaderboard through a limited number of “matches” against a subset of existing models rather than re-running the full mutual evaluation with all existing models (conceptually similar to Chatbot Arena’s pairwise-comparison aggregation). As the model pool grows, maintaining efficient computation and question diversity becomes increasingly important. Future work can explore more efficient scheduling and sampling strategies (e.g., selective evaluation or adaptive rounds) to reduce cost while maintaining stability.

Acknowledgments

We sincerely thank the anonymous reviewers and the Area Chair for their constructive feedback and valuable suggestions. In particular, we are grateful for the insightful comments that led us to conduct additional experiments, including the ablation study without reference answers and the sparse Elo experiment, etc. These suggestions have significantly strengthened the paper.

References

- Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.
- Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. 2024. Large language models for mathematical reasoning: Progresses and challenges. *arXiv preprint arXiv:2402.00157*.
- Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q Jiang, Jia Deng, Stella Biderman, and Sean Welleck. 2023. Llemma: An open language model for mathematics. *arXiv preprint arXiv:2310.10631*.
- J-C de Borda. 1781. Mémoire sur les élections au scrutin: Histoire de l’académie royale des sciences. *Paris, France*, 12(2).
- James Boyko, Joseph Cohen, Nathan Fox, Maria Han Veiga, Jennifer I Li, Jing Liu, Bernardo Modenesi, Andreas H Rauch, Kenneth N Reid, Soumi Tribedi, and 1 others. 2023. An interdisciplinary outlook on large language models for scientific research. *arXiv preprint arXiv:2311.04929*.
- Stephen Casper, Carson Ezell, Charlotte Siegmann, Noam Kolt, Taylor Lynn Curtis, Benjamin Bucknall, Andreas Haupt, Kevin Wei, Jérémy Scheurer, Marius Hobbhahn, and 1 others. 2024. Black-box access is insufficient for rigorous ai audits. In *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency*, pages 2254–2272.
- Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, and 1 others. 2024. A survey on evaluation of large language models. *ACM transactions on intelligent systems and technology*, 15(3):1–45.
- Connor Chen, Wei-Lin Chiang, Tianle Li, and Anastasios N. Angelopoulos. 2025. [Introducing sentiment control: Disentangling sentiment and substance](#).
- Guiming Hardy Chen, Shunian Chen, Ziche Liu, Feng Jiang, and Benyou Wang. 2024. Humans or llms as the judge? a study on judgement biases. *arXiv preprint arXiv:2402.10669*.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.
- Yuxing Cheng, Yi Chang, and Yuan Wu. 2025. A survey on data contamination for large language models. *arXiv preprint arXiv:2502.14425*.
- Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Banghua Zhu, Hao Zhang, Michael Jordan, Joseph E Gonzalez, and 1 others. 2024. Chatbot arena: An open platform for evaluating llms by human preference. In *Forty-first International Conference on Machine Learning*.
- 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*.
- Chunyuan Deng, Yilun Zhao, Xiangru Tang, Mark Gerstein, and Arman Cohan. 2023. Investigating data contamination in modern benchmarks for large language models. *arXiv preprint arXiv:2311.09783*.
- Yingqiang Ge, Wenyue Hua, Kai Mei, Juntao Tan, Shuyuan Xu, Zelong Li, Yongfeng Zhang, and 1 others. 2023. Openagi: When llm meets domain experts. *Advances in Neural Information Processing Systems*, 36:5539–5568.

- 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*.
- Yiming Huang, Zhenghao Lin, Xiao Liu, Yeyun Gong, Shuai Lu, Fangyu Lei, Yaobo Liang, Yelong Shen, Chen Lin, Nan Duan, and 1 others. 2023. Competition-level problems are effective llm evaluators. *arXiv preprint arXiv:2312.02143*.
- Huiyan. 2023. [Huiyan API](#).
- Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. Live-codebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*.
- Enkelejda Kasneci, Kathrin Seßler, Stefan Küchemann, Maria Bannert, Daryna Dementieva, Frank Fischer, Urs Gasser, Georg Groh, Stephan Günnemann, Eyke Hüllermeier, and 1 others. 2023. Chatgpt for good? on opportunities and challenges of large language models for education. *Learning and individual differences*, 103:102274.
- Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. 2024a. Gsm-plus: A comprehensive benchmark for evaluating the robustness of llms as mathematical problem solvers. *arXiv preprint arXiv:2402.19255*.
- Tianle Li, Anastasios Angelopoulos, and Wei-Lin Chiang. 2024b. Does style matter? disentangling style and substance in chatbot arena. *LMSYS Blog*.
- Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Cosgrove, Christopher D Manning, Christopher Re, Diana Acosta-Navas, Drew A. Hudson, and 31 others. 2023. [Holistic evaluation of language models](#). *Transactions on Machine Learning Research*. Featured Certification, Expert Certification, Outstanding Certification.
- Yiqi Liu, Nafise Sadat Moosavi, and Chenghua Lin. 2023. Llms as narcissistic evaluators: When ego inflates evaluation scores. *arXiv preprint arXiv:2311.09766*.
- Pan Lu, Liang Qiu, Wenhao Yu, Sean Welleck, and Kai-Wei Chang. 2023. A survey of deep learning for mathematical reasoning. In *ACL (1)*.
- Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. 2023. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*.
- Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. 2024. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. *arXiv preprint arXiv:2410.05229*.
- Steven Moore, Norman Bier, and John Stamper. 2024. Assessing educational quality: Comparative analysis of crowdsourced, expert, and ai-driven rubric applications. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, volume 12, pages 115–125.
- Mohamed Nejjar, Luca Zacharias, Fabian Stiehle, and Ingo Weber. 2025. Llms for science: Usage for code generation and data analysis. *Journal of Software: Evolution and Process*, 37(1):e2723.
- Qian Pan, Zahra Ashktorab, Michael Desmond, Martin Santillan Cooper, James Johnson, Rahul Nair, Elizabeth Daly, and Werner Geyer. 2024. Human-centered design recommendations for llm-as-a-judge. *arXiv preprint arXiv:2407.03479*.
- David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2024. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.
- Oscar Sainz, Jon Ander Campos, Iker García-Ferrero, Julen Etxaniz, Oier Lopez de Lacalle, and Eneko Agirre. 2023. Nlp evaluation in trouble: On the need to measure llm data contamination for each benchmark. *arXiv preprint arXiv:2310.18018*.
- Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, and 1 others. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.
- Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy,

and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in neural information processing systems*, 33:5776–5788.

Wei Xie, Shuoyoucheng Ma, Zhenhua Wang, Enze Wang, Kai Chen, Xiaobing Sun, and Baosheng Wang. 2024. Do large language models truly grasp mathematics? an empirical exploration from cognitive psychology. *arXiv preprint arXiv:2410.14979*.

Cheng Xu, Shuhao Guan, Derek Greene, M Kechadi, and 1 others. 2024. Benchmark data contamination of large language models: A survey. *arXiv preprint arXiv:2406.04244*.

Frank F Xu, Uri Alon, Graham Neubig, and Vincent Josua Hellendoorn. 2022. A systematic evaluation of large language models of code. In *Proceedings of the 6th ACM SIGPLAN international symposium on machine programming*, pages 1–10.

Shuo Yang, Wei-Lin Chiang, Lianmin Zheng, Joseph E Gonzalez, and Ion Stoica. 2023. Rethinking benchmark and contamination for language models with rephrased samples. *arXiv preprint arXiv:2311.04850*.

Jiayi Ye, Yanbo Wang, Yue Huang, Dongping Chen, Qihui Zhang, Nuno Moniz, Tian Gao, Werner Geyer, Chao Huang, Pin-Yu Chen, and 1 others. 2024. Justice or prejudice? quantifying biases in llm-as-a-judge. *arXiv preprint arXiv:2410.02736*.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhen-guo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.

Xuanchang Zhang, Wei Xiong, Lichang Chen, Tianyi Zhou, Heng Huang, and Tong Zhang. 2025. From lists to emojis: How format bias affects model alignment. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 26940–26961.

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.

A Statistical Details for Mathematics

Table 2 summarizes the Borda scores received by each model in Mathematics, including the mean and 95% confidence interval (CI) of the mean.

Model	Mean	95% CI
gemini-2.5-pro-exp	5.1388	[4.9509, 5.3102]
deepseek-r1	4.1918	[3.9959, 4.3796]
gpt-4.1	3.8776	[3.7020, 4.0409]
o3-mini	3.5714	[3.4000, 3.7347]
o1	3.4204	[3.2408, 3.6000]
deepseek-v3	2.8531	[2.6735, 3.0491]
claude-3-7-sonnet	2.5510	[2.3673, 2.7593]
qwen2.5-max	1.8204	[1.6531, 1.9918]

Table 2: **Mathematics overall summary (Borda score in [0, 6])**. Mean Borda score received by each model, with 95% CI of the mean.

Table 3 reports only the four non-significant ($p > 0.05$) pairs from Tukey HSD with adjusted p -values (p -adj); all remaining pairs are significant ($p < 0.05$).

Model 1	Model 2	p -adj
claude-3-7-sonnet	deepseek-v3	0.3162
deepseek-r1	gpt-4.1	0.2659
gpt-4.1	o3-mini	0.2989
o1	o3-mini	0.9502

Table 3: **Non-significant Tukey HSD pairs in Mathematics**. We report only the non-significant pairs ($p > 0.05$); all remaining pairs are significant ($p < 0.05$).

B Statistical Details for Programming

Table 4 summarizes the overall scores received by each model in programming, reporting the mean and 95% confidence interval (CI) of the mean.

Model	Mean	95% CI
gemini-2.5-pro-exp	89.6000	[88.0933, 91.0205]
o3-mini	79.5469	[77.0893, 82.0612]
deepseek-r1	77.9061	[75.0488, 80.3430]
gpt-4.1	76.6082	[73.6727, 79.1968]
o1	76.3347	[73.5750, 79.0083]
deepseek-v3	67.2490	[64.0241, 70.1633]
claude-3-7-sonnet	60.2163	[57.6361, 62.7634]
qwen2.5-max	55.5510	[52.4143, 58.6740]

Table 4: **Programming overall summary (absolute score in [0, 100])**. Mean score received by each model, with 95% CI of the mean.

C Score Heatmaps

We refer to the answering LLM as the responder here. Figures 6 and 7 summarize the mean scores for each evaluator-responder pair. The diagonal is set to 0 since self-evaluation is prohibited.

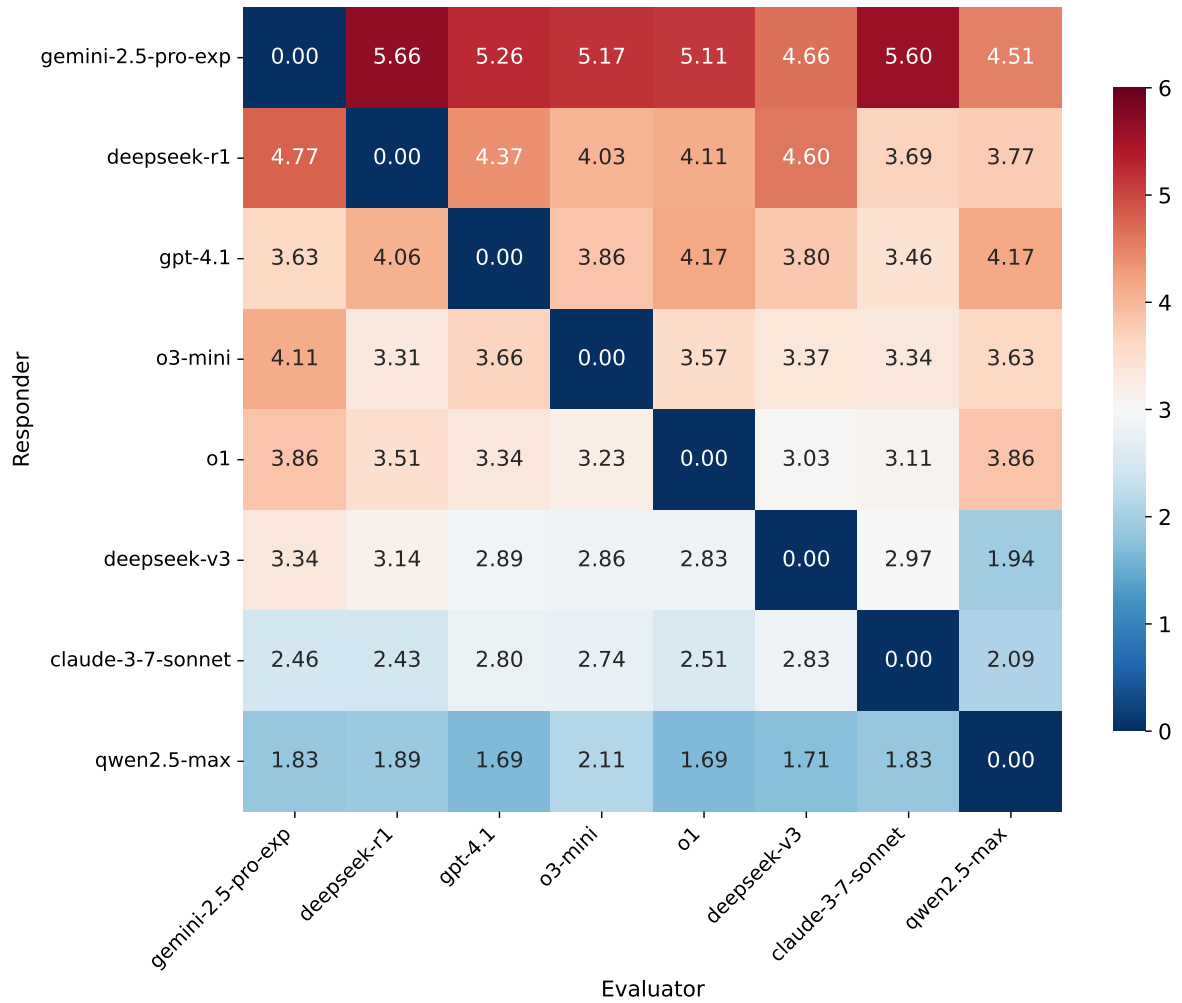


Figure 6: **Mathematics Scores.** Rows are responders and columns are evaluators; each cell shows the mean score.

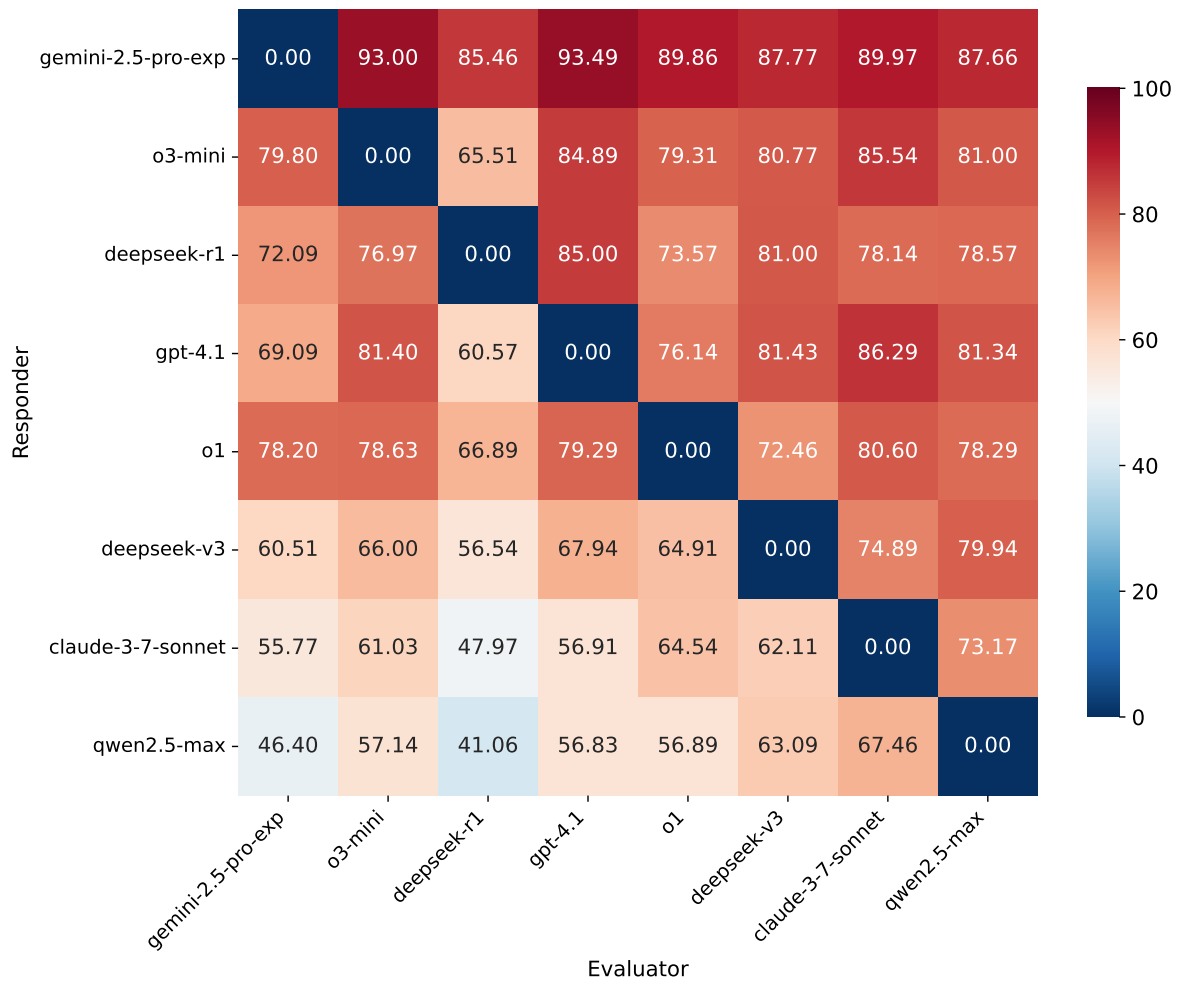


Figure 7: **Programming Scores.** Rows are responders and columns are evaluators; each cell shows the mean score.

Family	Models	In-family	Out-of-family	Δ	p -value
OpenAI	gpt-4.1, o3-mini, o1	79.9429	70.9390	9.0038	0.026770
DeepSeek	deepseek-r1, deepseek-v3	68.7714	67.9238	0.8476	0.956624

Table 5: **Developer-family in-family scoring differences (in-family vs. out-of-family)**. We report the mean scores, the difference Δ , and Welch’s two-sided t -test p -values.

D Top- k Consistency Details

Table 6 summarizes Top- k consistency in Mathematics across five independent runs, reporting the mean overlap and the 95% confidence interval (CI) for each k .

k	Top- k value	95% CI
1	0.4330	[0.3857, 0.4705]
2	0.5469	[0.5147, 0.5746]
3	0.6339	[0.6190, 0.6485]
4	0.7067	[0.6998, 0.7125]
5	0.7762	[0.7698, 0.7834]
6	0.8396	[0.8393, 0.8402]
7	1.0000	[1.0000, 1.0000]

Table 6: **Mathematics Top- k consistency**. Mean overlap of Top- k model sets across five independent runs, with 95% CI of the mean.

Table 7 reports the standard deviation of each model’s overall mathematics score across five independent runs.

Model	Std (Borda)	Std (100-point)
gemini-2.5-pro-exp	0.3932	6.5532
deepseek-r1	0.3635	6.0588
gpt-4.1	0.2625	4.3757
o3-mini	0.3319	5.5318
o1	0.1966	3.2766
deepseek-v3	0.3770	6.2829
claude-3-7-sonnet	0.6454	10.7560
qwen2.5-max	0.4055	6.7575

Table 7: **Mathematics score dispersion across runs**. Standard deviation (Std) of each model’s overall score across five independent runs. We report Std on the Borda scale ($[0, 6]$) and after linear rescaling to a 100-point scale ($\text{Std}_{100} = \text{Std}_{\text{Borda}} \times 100/6$).

E Higher In-Family Scores in Developer Families

Table 5 compares evaluators’ scores on in-family vs. out-of-family answering LLMs.

Difference. $\Delta = (\text{in-family mean scores}) - (\text{out-of-family mean scores})$.

Sample unit. We pool all eligible evaluation scores into two sets (in-family vs. out-of-family) for each developer family and compute their means.

Statistical test. We use Welch’s two-sided t -test (unequal variances) to compare in-family and

out-of-family scores for each developer family.

F Case Study

F.1 Case of Finding 1

See Figures 8 and 9. An example from gemini-2.5-pro-exp-03-25 illustrates professional-level mathematical question generation.

F.2 Case of Finding 2

Deepseek-r1 exhibits “memorization-based answering” behavior that mimics reasoning. Figure 10 presents a representative example.

G Prompts

G.1 Prompts for Mathematics

The full prompts used for Mathematics include question generation, question answering, and answer evaluation. The corresponding prompts are presented in Figures 11–13.

G.2 Prompts for Programming

The full prompts used for Programming include question generation, question answering, and answer evaluation. The corresponding prompts are presented in Figures 14–16.

Question

Let $S_3(n)$ denote the sum of the digits of a positive integer n when written in base 3. Define a sequence $a_n = (-1)^{S_3(n)}$ for $n \geq 1$. Consider the Dirichlet series $F(s) = \sum_{n=1}^{\infty} a_n / n^s$, defined for complex numbers s where the series converges. Determine if $F(s)$ can be analytically continued to a region including $s=0$, and if so, find the value $F(0)$.

Principle

The core principle is the deep connection between the properties of a sequence defined by discrete arithmetic rules (a_n depends on the sum of digits of n in base 3) and the analytic properties (convergence, analytic continuation, value at a specific point) of a complex function (the Dirichlet series $F(s) = \sum a_n / n^s$) constructed from that sequence. This mirrors the challenge of relating discrete number-theoretic information (like prime distribution) to the analytic behavior of related complex functions (like the Riemann zeta function).

Figure 8: Case study for Finding 1. Question and principle.

Reference Answer (Abridged for Clarity)

Step 1: Region of Convergence.

The sequence $a_n = (-1)^{S_3(n)}$ consists of +1 and -1. The Dirichlet series $F(s) = \sum_{n=1}^{\infty} a_n / n^s$ converges absolutely when $\sum_{n=1}^{\infty} |a_n / n^s| = \sum_{n=1}^{\infty} 1 / n^{\text{Re}(s)}$ converges. This occurs for $\text{Re}(s) > 1$.

Step 2: Convergence for $\text{Re}(s) > 0$.

Let $T_k = \sum_{n=0}^{3^k-1} a_n$. Using $S_3(3q+r) = S_3(q) + S_3(r)$ ($r=0,1,2$), we have $a_{3q+r} = a_q * a_r$ (with $a_0=1$), hence

$T_k = T_{k-1} * (a_0 + a_1 + a_2) = T_{k-1}$.

With $T_0=1$, we get $T_k=1$ for all $k \geq 0$, so $\sum_{n=1}^{3^k-1} a_n = T_k - a_0 = 0$. Bounded partial sums imply that $F(s)$ converges for all s with $\text{Re}(s) > 0$.

Step 3: Analytic Continuation.

Split $F(s)$ by $n \bmod 3$ and use $a_{3m}=a_m$, $a_{3m+1}=-a_m$, $a_{3m+2}=a_m$ to obtain

$F(s) * (1 - 3^{-s}) = K(s)$,

where $K(s) = -1 + 2^{-s} + \sum_{m=1}^{\infty} a_m * [(3m+2)^{-s} - (3m+1)^{-s}]$.

The series defining $K(s)$ converges for $\text{Re}(s) > 0$, thus $K(s)$ is analytic in this region.

Therefore, $F(s) = K(s) / (1 - 3^{-s})$ provides an analytic continuation to $\text{Re}(s) > 0$ (except where $1 - 3^{-s}=0$).

Step 4: Value at $s=0$.

We evaluate $K(0) = -1 + 2^0 + \sum_{m=1}^{\infty} a_m * [(3m+2)^0 - (3m+1)^0] = 0$, so $s=0$ is a removable singularity. Using L'Hopital's rule,

$F(0) = K'(0) / \ln(3)$,

where $K'(0) = -\ln(2) + \sum_{m=1}^{\infty} a_m * \ln((3m+1)/(3m+2))$.

Final Answer: $F(s)$ can be analytically continued to a region including $s=0$. The value at $s=0$ is $F(0) = [-\ln(2) + \sum_{m=1}^{\infty} (-1)^{S_3(m)} * \ln((3m+1)/(3m+2))] / \ln(3)$.

Figure 9: Case study for Finding 1. Reference answer (abridged for clarity).

Answer

The function $F(s)$ can be analytically continued to include $s=0$. The value $F(0)$ is determined by representing $F(s)$ as an infinite product over base-3 digit positions. Each term in the product corresponds to the sum over digits (0,1,2) in each position, contributing $(-1)^d / (d \cdot 3^k)^s$. At $s=0$, each factor simplifies to $1 + (-1) + 1 = 1$. The infinite product of 1s converges to 1. Thus, $F(0) = 1$.

Figure 10: Case study for Finding 2. Example answer exhibiting “memorization-based answering”.

Prompts for Mathematics Question Generation

You're a mathematician. Please perform the following tasks in order, and output only the final result in JSON format (no Markdown, no extra text):

1. Select a mathematical question that you consider one of the most difficult. This can come from any field (e.g., algebra, geometry, number theory, etc.).
2. Identify the core mathematical principle(s) that make this question difficult.
3. Create a new, original, and difficult mathematical question based on the identified core principle(s). This question must be entirely novel and not derived from any existing textbook or exam.
4. Ensure that the new question reflects both the core difficulty and the unique characteristics of the math domain.
5. Solve the question by providing a complete, step-by-step reference solution.
6. IMPORTANT: Write all mathematical equations in plain text format. DO NOT use LaTeX formatting or special symbols. For example:
 - Correct: "Find all integer solutions (x, y, z) to the equation $x^2 + y^2 + z^2 = x^2y^2$ "
 - Incorrect: "Find all integer solutions (x, y, z) to the equation $\$x^2 + y^2 + z^2 = x^2y^2\$\text{"$
7. Output the final result only in JSON format, using the structure below:

```
{
  "Question": "Your newly created difficult math question here with equations in plain text.",
  "Reference Answer": "Step-by-step and final answer with explanation using plain text equations.",
  "Principle": "The abstracted mathematical principle that underlies the question."
}
```

Figure 11: Prompts for Mathematics Question Generation.

Prompts for Mathematics Question Answering

Please perform the following tasks:

1. First, think through the question step by step. You may output your full reasoning process before the final answer.

2. Then, produce a final answer that is:

Concise, logically sound, and self-contained

final answer with necessary steps

Strictly no more than 500 words

Returned only as a valid JSON object with the key "Answer"

3. Important Constraints:

Only the final answer should appear in the JSON object.

Do not include any of your reasoning or thought process inside the JSON object.

The JSON object must be the last part of your output.

Do not use Markdown or any other formatting. Output should be plain text and a valid JSON object at the end.

IMPORTANT: Write all mathematical equations in plain text format. DO NOT use LaTeX formatting or special symbols. For example, write " $x^2 + y = z$ " instead of " $\$x^2 + y = z\$$ ".

Question:{question}

Expected Final Output Format (JSON only):

```
{
  "Answer": "Your final answer with necessary steps, under 500 words, goes here. Do not include reasoning or any text other than the final answer in this object. Use plain text for all equations."
}
```

Figure 12: Prompts for Mathematics Question Answering.

Prompts for Mathematics Answer Evaluation

Evaluate the responses from multiple large language models based on the following information:

Question:

{question}

Reference Answer:

{reference_answer}

Principle:

{principle}

All the Models Answers (excluding your own):

{filtered_answers_str}

Please perform the following tasks:

1. Evaluate other models' answers based on the reference solution, considering correctness, completeness, clarity, logical rigor and alignment with the given principle.
2. Rank all model responses from best to worst based on your evaluation. Please note that reference answers are not counted in the rankings and scores.
3. If any model answer is better than the reference answer, explicitly state this in your rationale and reflect it in the ranking.
4. **IMPORTANT:** Write all mathematical equations in plain text format. DO NOT use LaTeX formatting or special symbols. For example, write " $x^2 + y = z$ " instead of "\$x^2 + y = z\$".
5. Output your result only in JSON format, with the following structure:

```
{  
  "Ranking": ["Model1", "Model2", "Model3", "..."],  
  "Reason": {  
    "Model1": "Detailed explanation of strengths and weaknesses, including comparison  
to reference answer.",  
    "Model2": "Detailed explanation...",  
    "Model3": "Detailed explanation..."  
  }  
}
```

Figure 13: Prompts for Mathematics Answer Evaluation.

Prompts for Programming Question Generation

You are a senior programming competition question setter. Your task is to perform the following steps strictly and output only the final result in JSON format with no Markdown, no commentary, and no additional fields or nesting.

1. Study how difficult programming questions are framed in high-level competitive programming. Identify what makes them difficult: novel domains, tight constraints and implementation complexity. {Few Shot Examples}

2. Create a completely original and non-derivative programming question that includes:

A concise but fully specified question description.

Input and output format.

Clear constraints.

3. Ensure the question reflects true conceptual depth and comes from an original domain, not just a variant of classic questions (e.g., not just another sorting or graph question with tweaks).

4. Solve the question fully, and include:

A complete working implementation of the solution (in plain text).

Explanations of the core algorithm and ideas used.

Time and space complexity analysis.

5. Define a scoring criteria out of 100 points. Evaluate from the following but not limited to several aspects:

Accuracy and correctness

Execution speed (runtime performance)

Memory efficiency

Human readability of code

Code modularity and organization

6. VERY IMPORTANT:

Do NOT output any extra text, explanations, section headers, or Markdown formatting.

All output must be in JSON format only.

Use EXACTLY the following three fields:

json

{

"Question": "Your newly created difficult programming question with clear requirements and constraints.",

"Reference Solution": "Complete solution including FULL CODE IMPLEMENTATION with key concept explanations and complexity analysis.",

"Scoring Criteria": "Detailed scoring criteria across multiple evaluation dimensions, out of 100."

}

7. Do NOT change the field names ("Question", "Reference Solution", "Scoring Criteria"). Do not include any other fields. Output must be a flat JSON object with only those three keys.

Figure 14: Prompts for Programming Question Generation.

Prompts for Programming Question Answering

Please solve the following programming question with a complete code implementation only.

Question: {question}

Requirements:

Solve the question independently using the programming language you consider most appropriate.

Your code must be complete and runnable (not pseudocode or algorithm outline).

Include all necessary functions, classes, and logic.

Handle edge cases appropriately.

Follow good programming practices with clear variable names and inline comments if needed.

IMPORTANT:

DO NOT include any explanations, thoughts, steps, or discussions.

DO NOT output anything except a JSON object with one key: "Answer"

The "Answer" field must contain ONLY the actual code, properly escaped as a single-line JSON string (with `\n` for newlines, and `\"` for quotes if needed).

DO NOT wrap the code in Markdown, and DO NOT include any formatting symbols like backticks or section headers.

Any reasoning, description, or justification MUST be omitted completely from the final output.

Final Output Format (Strictly required):

```
{  
  "Answer": "your_full_code_here_with_escaped_newlines"  
}
```

Notes:

This is a strict format enforcement. Any deviation (extra fields, formatting, explanations, etc.) will be considered incorrect.

The response must be valid JSON, flat, and contain only the "Answer" key with actual executable code as its value.

Figure 15: Prompts for Programming Question Answering.

Prompts for Programming Answer Evaluation

Evaluate the responses from multiple large language models based on the following information:

Question:

{question}

Reference Solution:

{reference_solution}

Scoring Criteria:

{scoring_criteria}

All the Models' Code Implementations (excluding your own):

{filtered_answers_str}

Please perform the following tasks:

1. Evaluate each model's answer based on the `reference_solution` and `scoring_criteria`.
2. For each model, provide a precise score out of 100 based on the scoring criteria. Models that didn't provide actual code implementation should receive significantly lower scores.
3. Provide detailed reasoning for each score, highlighting both strengths and weaknesses in the implementation.
4. If any model's solution exceeds the reference solution in quality, efficiency or elegance, note this explicitly.
5. Output your evaluation results in JSON format:

```
{
  "Scores": {
    "Model1": 95,
    "Model2": 80,
    "Model3": 70
  },
  "Reason": {
    "Model1": "Detailed explanation of scoring, including specific assessment of code implementation.",
    "Model2": "Detailed explanation...",
    "Model3": "Detailed explanation..."
  }
}
```

Figure 16: Prompts for Programming Answer Evaluation.