RANKER-AS-SCORER: Improving Scoring-Based LLM Judges in Zero-Resource Settings

Anonymous ACL submission

Abstract

LLM-as-a-Judge is widely used for generated content evaluation, and scoring-based LLM judges are crucial for content quality qualification. However, existing strategies to improve scoring accuracy are costly, making them impractical in zero-resource settings. To address this, we propose **RANKER-AS-SCORER**, a lightweight strategy that transforms direct scoring into score ranking. It enhances awareness of score rubric and maintains scoring efficiency with requirements of only one LLM run. Experiments on two benchmarks with four judge models show that our proposed method outperforms direct scoring in most cases, approaches optimal aggregation strategy without its overhead, and balances performance and efficiency in zero-resource settings.

1 Introduction

Recently, the LLM-as-a-Judge paradigm has become a critical tool for evaluating AI output, with wide applications in generated content assessment, preference learning, and multi-agent interaction (Li et al., 2024a). Unlike traditional statistical metrics such as BLEU (Papineni et al., 2002) and ROUGE (Lin, 2004), which rely on human-labeled golden answers, LLM-as-a-Judge leverages the understanding and reasoning capabilities of Large Language Models (LLMs) to evaluate content against predefined criteria, outputting judgments as scores, choices, or sentences. Among these, scoring-based LLM judges are valuable for qualifying and comparing generated content quality.

However, deploying scoring-based LLM judges faces a key challenge: existing improvement strategies are costly in resource-constrained scenarios. Prompt-based methods, such as in-context learning (Song et al., 2024), requires manual curation/review of external materials; Tuning-based methods, such as fine-tuning (Saad-Falcon et al., 2024; Kim et al., 2024a,b), demands large annotated datasets and expensive GPU computations; Post-processing methods, such as aggregating multiple LLM judgments (Yang et al., 2024), requires repeated LLM runs, leading to exponential time overhead.

To address this gap, this paper focuses on improving scoring-based LLM judges specifically for zero-resource environments, where we prioritize two core goals: maintaining/enhancing scoring accuracy (aligned with human judgments) and minimizing resource overhead (no manual curation, GPU training, or repeated runs). We propose **RANKER-AS-SCORER**, a lightweight strategy for scoring-based LLM judges in zero-resource settings. Its key insight is transforming the scoring task into a score ranking task: instead of directly outputting a single score, the LLM first generates a ranking of all scores, then selects the top-ranked score as the final judgment. This transformation ensures that the LLM explicitly considers the full range of score rubrics (addressing rubric awareness gap of direct scoring) while matching direct scoring's efficiency (avoiding aggregation's runtime overhead). To validate our method, we conduct experiments on two human-annotated LLM-as-a-Judge benchmarks, using four LLM judges. Our main contributions are threefold:

- We propose RANKER-AS-SCORER, a lightweight strategy that transforms scoring to ranking, enhancing rubric awareness with only one LLM run.
- Extensive experiments show our method outperforms direct scoring in most cases, approaches optimal aggregation accuracy without its overhead, and balances performance and efficiency in zero-resource settings.
- Exploratory experiments clarify the impact of aggregation methods (identifying Mean as optimal) and reveal that LLM judges suffer from self-inconsistency in scoring.

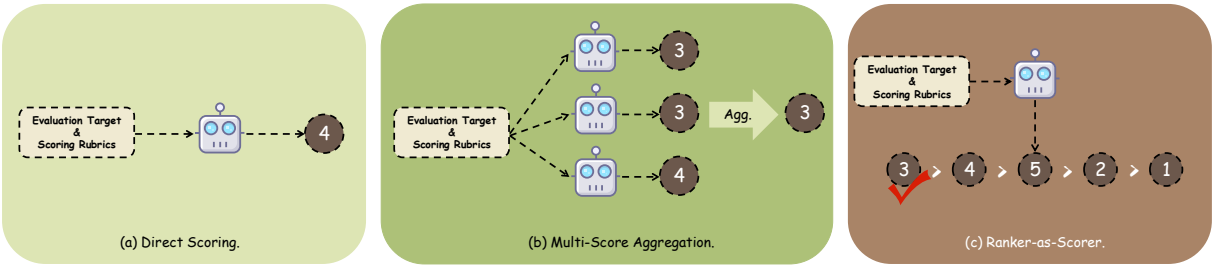

Figure 1: Different scoring-based LLM judgment methods in zero-resource settings. Figure (a) is direct scoring. Figure (b) is the multi-score aggregation method, which runs LLM judge multiple times to generate scores, then aggregate them to obtain a final results. Figure (c) is our proposed method RANKER-AS-SCORER, which outputs a score ranking other than a score.

2 Method

2.1 Problem Formulation

Following the definition from (Gu et al., 2024), the scoring-based LLM-as-a-Judge can be formulated as follows:

$$\mathcal{S} \leftarrow \mathcal{P}_{\text{LLM}}(x \oplus \mathcal{C}). \tag{1}$$

Where $\mathcal{S}$ is the final score obtained from the LLM-as-a-Judge; $\mathcal{P}_{\text{LLM}}$ is the probability function of the LLM; x represents the evaluation target; and $\mathcal{C}$ is the context of the input x, such as the scoring rubrics and the reference answers. In this paper, we define **direct scoring** as context $\mathcal{C}$ that does not contain any external knowledge besides task description and scoring rubrics.

2.2 Improvement Strategy

Focusing on cost control, we only compare common-used and training-free improvement methods, including incorporating reference answers into the scoring prompt and ensembling multiple scores, with our proposed method RANKER-AS-SCORER.

Reference Answer Following (Kim et al., 2024a), we introduce a full-marked reference answer into the context $\mathcal{C}$ of scoring prompts. It enables the LLM judge to compare between the reference answer and the evaluation target, which is conducive to making a scoring decision. It should be noted that high quality reference answers often require manual writing or review. So, strictly speaking, **this method is not in zero-resource settings**.

Multi-Score Aggregation Due to the instability of LLMs, running the same task multiple times and aggregating the results has become a common method to improve the model's accuracy (Li et al., 2025). For scoring-based judgment, we run the same evaluation task with the same LLM judge n times to get n scores. The i-th run is as follows:

$$\mathcal{S}_i \leftarrow \mathcal{P}_{\text{LLM}}(x \oplus \mathcal{C})_i. \tag{2}$$

After completing the n runs, we obtain the final score through an aggregation strategy:

$$\mathcal{S}_{\text{final}} = \text{Agg}(\{\mathcal{S}_1, \ldots, \mathcal{S}_i, \ldots, \mathcal{S}_n\}). \tag{3}$$

Where $\text{Agg}(\cdot)$ is the aggregation function. In this paper, we consider 5 aggregation strategies:

- **Max**: the maximum among the scores.
- **Min**: the minimum among the scores.
- **Median**: the median among the scores.
- **Mean**: the average score is computed and rounded to the nearest integer.
- **Majority Voting**: the most frequently scores.

Although this method can improve scoring accuracy, it often requires running the LLM multiple times, resulting in exponential time overhead.

RANKER-AS-SCORER In this paper, we propose an improvement strategy for scoring-based LLM judges called RANKER-AS-SCORER. We transform the output of the scoring-based LLM judge into a score ranking $\mathcal{R}$ based on the score rubric that the evaluation target better meets:

$$\mathcal{R} \leftarrow \mathcal{P}_{\text{LLM}}(x \oplus \mathcal{C}), \tag{4}$$

$$\mathcal{R} = \{\mathcal{S}_i > \cdots > \mathcal{S}_j\}. \tag{5}$$

Afterwards, the top score is taken as the score result. Compared with directly outputting a score,

Model	Scoring Method	BiGGen Bench		FLASK	
		ρ	r	ρ	r
GPT-4.1	*Reference Answer*	0.5989	0.6358	0.6265	0.6630
	Vanilla	0.5977	0.6363	0.5545	0.5984
	Best Aggregation	0.6108	0.6492	0.5679	0.6057
	RANKER-AS-SCORER	0.5962	0.6288	**0.5685**	**0.6115**
Kimi-K2	*Reference Answer*	0.5425	0.5287	0.6161	0.6035
	Vanilla	0.5352	0.5285	0.5665	0.5723
	Best Aggregation	0.5694	0.6066	0.5959	0.6160
	RANKER-AS-SCORER	0.5621	0.5932	0.5663	0.5956
DeepSeek-V3	*Reference Answer*	0.5718	0.6128	0.6071	0.6472
	Vanilla	0.5314	0.5814	0.5105	0.5785
	Best Aggregation	0.5434	0.5923	0.5333	0.5910
	RANKER-AS-SCORER	0.5432	**0.5984**	0.5314	**0.6005**
Qwen3-32B	*Reference Answer*	0.5671	0.6019	0.6003	0.6274
	Vanilla	0.5280	0.5684	0.5110	0.5519
	Best Aggregation	0.5650	0.5952	0.5367	0.5694
	RANKER-AS-SCORER	0.5513	0.5879	0.5174	0.5525

Table 1: Correlation between LLM scores and human scores, measured by Spearman's ρ and Person's r correlation coefficients. *Vanilla* is the direct scoring method. *Reference Answer* means that a full-marked answer is introduced as a reference during direct scoring. *Best Aggregation* represents the best score from the 5 aggregation strategies mentioned in this paper under 5 times scoring. RANKER-AS-SCORER is our proposed method. The gray results are not in zero-resource settings. The green results are better than *Vanilla* method, and the red ones means worse. The **boldfaced** results are better than the best aggregation results.

RANKER-AS-SCORER is aware of the rubrics of all scores during decoding. In addition, our method only needs to run the LLM judge once to obtain a scoring result, which is more efficient than the aggregation-based method. Fig. 1 compares the three zero-resource scoring-based LLM judge mentioned in this work for better understanding.

3 Experiments

3.1 Setup

Datasets Following (Kim et al., 2024b; Chiang et al., 2025), we select two LLM-as-a-Judge benchmarks with scores labeled by human annotators and full-marked reference answers for evaluation:

- **BiGGen Bench** (Kim et al., 2025): This is a principled generation benchmark with instance-specific evaluation criteria. We use its non-multilingual test set, which includes 2,780 responses for 695 instances.
- **FLASK** (Ye et al., 2024): A fine-grained evaluation dataset with 200 test prompts, 12 score rubrics, and 2,001 responses.

Metrics We use the correlation between LLM scores $S_{\mathrm{M}} = \{\mathcal{S}_{\mathrm{M}}^1, \ldots, \mathcal{S}_{\mathrm{M}}^n\}$ and human scores $S_{\mathrm{H}} = \{\mathcal{S}_{\mathrm{H}}^1, \ldots, \mathcal{S}_{\mathrm{H}}^n\}$on the same evaluation targets to measure the **scoring accuracy** of LLM judges, reporting Spearman's correlation coefficient ρ (Spearman, 1987) and Pearson's correlation coefficient r (Pearson, 1895).

Since the aggregation-based method requires multiple runs of LLM judges, we would like to explore whether LLM judges can give **self-consistent** scores in multiple runs for the same evaluation target using two metrics. Self-Inconsistency Rate (SIR) is the probability that different scores appear in multiple results for the same evaluation target:

$$\mathrm{SIR} = \frac{1}{M} \sum_{j=1}^{M} \mathbb{I}(|\mathrm{set}(\{\mathcal{S}_{j,1}, \ldots, \mathcal{S}_{j,n}\})| > 1) \quad (6)$$

Where n is the total run time, j is the j-th evaluation target, and M is the total number of evaluation targets. Averaged Normalized Entropy (ANE), inspired by (Li et al., 2023), are used to measure the uncertainty of LLM judges:

$$\mathrm{NE} = \frac{\sum_i p_i \ln p_i}{\ln n} \quad (7)$$

$$\mathrm{ANE} = \frac{1}{M} \sum_{j=1}^{M} \mathrm{NE}_j \quad (8)$$

For an evaluation target with multiple scores, we first calculate the Normalized Entropy (NE) following formula 7, where p_i is the probability of the i-th score, and n is the scoring time. We get the NE of all evaluation targets and then average them to get the ANE.

Judge Models Previous works (Li et al., 2024b; Liu et al., 2024) often use stronger models as judge models to ensure evaluation reliability. Following this, we select recent and advanced LLMs GPT-4.1 (OpenAI, 2025), Kimi-K2 (Bai et al., 2025), DeepSeek-V3 (DeepSeek-AI et al., 2024), and Qwen3-32B (Yang et al., 2025) as judge models.

Judge Prompt The judge prompt we used is shown in Appendix A.1 including both direct scoring prompt (with and without reference answers) and our RANKER-AS-SCORER prompt.

Hyper-Parameters We set the temperature of all LLM judges to 0.7 during experiments. For multiple evaluation runs, we judge the same evaluation target 5 times with the same configuration.

Model	Scoring Method	BiGGen Bench							FLASK						
		Correlation with human scores ↑					Self-Consistency ↓		Correlation with human scores ↑					Self-Consistency ↓	
		Max	Min	Med.	Mean	MV	SIR (%)	ANE	Max	Min	Med.	Mean	MV	SIR (%)	ANE
GPT-4.1	*Vanilla*	0.6230	0.6386	0.6486	**0.6492**	0.6484	25.90	0.0987	0.5938	0.6026	0.6017	**0.6057**	0.6034	25.34	0.0946
	RANKER-AS-SCORER	0.6311	0.6286	0.6363	**0.6463**	0.6371	26.58	0.1009	0.6047	0.6184	0.6175	**0.6239**	0.6164	31.18	0.1166
Kimi-K2	*Vanilla*	**0.6066**	0.3852	0.5941	0.5864	0.5897	57.73	0.2465	0.6138	0.4971	0.6131	**0.6160**	0.6063	47.23	0.1999
	RANKER-AS-SCORER	0.6014	0.5798	0.6058	**0.6149**	0.6081	54.10	0.2200	0.6131	0.5971	0.6240	**0.6377**	0.6208	47.58	0.2001
DeepSeek-V3	*Vanilla*	0.5743	**0.5923**	0.5865	0.5909	0.5834	32.01	0.1209	0.5640	0.5848	0.5880	**0.5910**	0.5879	29.74	0.1136
	RANKER-AS-SCORER	0.5813	0.5983	**0.6011**	0.6008	0.5982	31.04	0.1172	0.5875	0.5975	0.5917	**0.5978**	0.5926	30.08	0.1159
Qwen3-32B	*Vanilla*	0.5654	0.5939	0.5926	**0.5952**	0.5912	42.16	0.1616	0.5585	0.5586	0.5597	**0.5694**	0.5582	44.58	0.1745
	RANKER-AS-SCORER	0.5818	**0.6070**	0.5899	0.5981	0.5908	44.50	0.1708	0.5618	0.5577	0.5732	**0.5837**	0.5753	47.33	0.1907

Table 2: Correlation between aggregation LLM scores and human scores measured by Person's r correlation coefficients, and self-consistency metrics of scores obtained from multiple LLM judge runs. *Vanilla* is the direct scoring method. RANKER-AS-SCORER is our proposed method. We run those scoring methods 5 times then get a final score through aggregation strategies shown in line 3, where "Med." is median, and "MV" is majority voting. The best aggregation strategy are **boldfaced**. For self-consistency, "SIR" is self-inconsistency rate, and "ANE" is averaged normalized entropy. The green results are better than *Vanilla* method, and the red ones means worse.

3.2 Main Results

Table 1 shows the correlation between the scores of 4 LLM scoring methods and the human scores. Among all methods, the reference answer enhanced method has the highest scoring accuracy due to its reliance on external knowledge, but it does not meet the zero-resource settings. In zero-resource scenarios, running the scoring model multiple times and adopting the optimal aggregation strategy can obviously improve the scoring accuracy, but the efficiency is low. In most cases, RANKER-AS-SCORER is able to surpass direct scoring and approach the optimal aggregation strategy, even better in some cases. Furthermore, RANKER-AS-SCORER only requires a single run, achieving the same high efficiency as direct scoring. The experimental results verify the practical value of our proposed method in zero-resource settings.

3.3 Exploratory Experiments

The exploratory experiments focus on the aggregation performance and self-consistency of scoring-based LLM judges, and the results are shown in Table 2. The key conclusions are as follows:

The aggregation strategy has a significant impact on scoring accuracy, among which the Mean strategy has the best performance. Different aggregation strategies have different effects on scoring performance. In some cases, the Max strategy is even lower than the result of running only once. It is obvious that the Mean strategy outperforms other strategies in most cases, and this trend exists in both scoring methods.

RANKER-AS-SCORER is improved with the aggregation strategy, and still outperforms direct scoring. Running RANKER-AS-SCORER multiple times and aggregating the scores generally yields higher accuracy than running it once. Furthermore, in most cases, the performance of RANKER-AS-SCORER is higher than direct scoring under the same aggregation strategy. With the optimal strategy, RANKER-AS-SCORER usually performs better. We recommend that aggregation of RANKER-AS-SCORER's results can be considered for extreme performance; running it only once can still achieve good performance while keeping efficiency.

The scoring-based LLM-judge suffers from self-inconsistency, and RANKER-AS-SCORER does not effectively alleviate this. Many evaluation targets will get different scores after multiple judgments. GPT-4.1 has inconsistent scores in a quarter of cases, while Kimi-K2 has a self-inconsistency rate of more than 50% on BiGGen Bench. Fortunately, the low averaged normalized entropy suggests that self-inconsistent scores are not chaotic, but rather likely concentrated in 2-3 types of scores. Experimental results show that RANKER-AS-SCORER does not alleviate self-inconsistency, which may due to the complexity of the ranking task and the incorrect output format parsing.

4 Conclusion

We propose **RANKER-AS-SCORER**, a lightweight strategy that transforms direct scoring into scoring ranking, enabling LLMs to explicitly consider full score rubrics while requiring only a single run. Experimental results show that RANKER-AS-SCORER strikes an optimal balance between performance and efficiency in zero-resource settings. Exploratory experiments further confirm the impact of aggregation strategies, and reveal self-inconsistency in LLM judges.

Limitations

First, our method lacks generalization. It is only applicable to scoring-based LLM judges and cannot be used for other methods such as comparative LLM judges and critic-based LLM judges. Second, the ranking task is more complex than direct scoring. We observe that LLM judges' failure to follow instructions is more pronounced during ranking than during scoring, but we do not conduct a complete experimental investigation. We also do not propose a strategy to restrict the output format. Third, experimental results show that our method fails to alleviate the self-inconsistency of LLM judges. Fourth, our method often generates more tokens than direct scoring because of the complexity of the ranking task, which may diminish the efficiency advantage in token-cost-sensitive scenarios. Finally, the marginal effect of the number of runs on the magnitude of the performance improvement is not quantified. For example, when the number of runs increases from 5 to 10, the performance still improves linearly, or the performance no longer improves after 8 runs, which requires further experimental research.

References

Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, Zhuofu Chen, Jialei Cui, Hao Ding, Mengnan Dong, Angang Du, Chenzhuang Du, Dikang Du, Yulun Du, Yu Fan, Yichen Feng, and 80 others. 2025. Kimi K2: open agentic intelligence. *CoRR*, arXiv:2507.20534.

Cheng-Han Chiang, Hung-yi Lee, and Michal Lukasik. 2025. TRACT: regression-aware fine-tuning meets chain-of-thought reasoning for llm-as-a-judge. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2934–2952.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, and 80 others. 2024. Deepseek-v3 technical report. *CoRR*, arXiv:2412.19437.

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. 2024. A survey on llm-as-a-judge. *CoRR*, arXiv:2411.15594.

Seungone Kim, Jamin Shin, Yejin Choi, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, and Minjoon Seo. 2024a. Prometheus: Inducing fine-grained evaluation capability in language models. In *The Twelfth International Conference on Learning Representations*.

Seungone Kim, Juyoung Suk, Ji Yong Cho, Shayne Longpre, Chaeeun Kim, Dongkeun Yoon, Guijin Son, Yejin Choi, Sheikh Shafayat, Jinheon Baek, Sue Hyun Park, Hyeonbin Hwang, Jinkyung Jo, Hyowon Cho, Haebin Shin, Seongyun Lee, Hanseok Oh, Noah Lee, Namgyu Ho, and 13 others. 2025. The biggen bench: A principled benchmark for fine-grained evaluation of language models with language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5877–5919.

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

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

Minzhi Li, Taiwei Shi, Caleb Ziems, Min-Yen Kan, Nancy F. Chen, Zhengyuan Liu, and Diyi Yang. 2023. Coannotating: Uncertainty-guided work allocation between human and large language models for data annotation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1487–1505.

Yuan Li, Yue Huang, Hongyi Wang, Xiangliang Zhang, James Zou, and Lichao Sun. 2024b. Quantifying AI psychology: A psychometrics benchmark for large language models. *CoRR*, arXiv:2406.17675.

Zichong Li, Xinyu Feng, Yuheng Cai, Zixuan Zhang, Tianyi Liu, Chen Liang, Weizhu Chen, Haoyu Wang, and Tuo Zhao. 2025. Llms can generate a better answer by aggregating their own responses. *CoRR*, arXiv:2503.04104.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

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

Linguistics (Volume 1: Long Papers), pages 11621–11640.

OpenAI. 2025. Gpt-4.1. https://platform.openai.com/docs/models/gpt-4.1. Accessed: Sept. 10, 2025.

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

Karl Pearson. 1895. Vii. note on regression and inheritance in the case of two parents. *proceedings of the royal society of London*, 58(347-352):240–242.

Jon Saad-Falcon, Omar Khattab, Christopher Potts, and Matei Zaharia. 2024. ARES: an automated evaluation framework for retrieval-augmented generation systems. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 338–354.

Mingyang Song, Mao Zheng, and Xuan Luo. 2024. Can many-shot in-context learning help long-context LLM judges? see more, judge better! *CoRR*, arXiv:2406.11629.

Carl Spearman. 1987. The proof and measurement of association between two things. *The American journal of psychology*, 100(3/4):441–471.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 40 others. 2025. Qwen3 technical report. *CoRR*, arXiv:2505.09388.

Joshua C. Yang, Damian Dailisan, Marcin Korecki, Carina I. Hausladen, and Dirk Helbing. 2024. LLM voting: Human choices and AI collective decision-making. In *Proceedings of the Seventh AAAI/ACM Conference on AI, Ethics, and Society*, pages 1696–1708.

Seonghyeon Ye, Doyoung Kim, Sungdong Kim, Hyeonbin Hwang, Seungone Kim, Yongrae Jo, James Thorne, Juho Kim, and Minjoon Seo. 2024. FLASK: fine-grained language model evaluation based on alignment skill sets. In *The Twelfth International Conference on Learning Representations*.

A Appendix

A.1 Judge Prompts

Fig. 2 is the direct scoring prompt, and Fig. 3 is the version with reference answer. The above two prompts are from (Kim et al., 2024b). Fig. 4 is our proposed RANKER-AS-SCORER prompt.

Direct Scoring Prompt

###Task Description:
An instruction (might include an Input inside it), a response to evaluate, and a score rubric representing a evaluation criteria are given.
1. Write a detailed feedback that assess the quality of the response strictly based on the given score rubric, not evaluating in general.
2. After writing a feedback, write a score that is an integer between 1 and 5. You should refer to the score rubric.
3. The output format should look as follows: "Feedback: (write a feedback for criteria) [RESULT] (an integer number between 1 and 5)
4. Please do not generate any other opening, closing, and explanations.
###The instruction to evaluate:
{orig_instruction}
###Response to evaluate:
{orig_response}
###Score Rubrics:
[{score_criteria}]
Score 1: {score1_description}
Score 2: {score2_description}
Score 3: {score3_description}
Score 4: {score4_description}
Score 5: {score5_description}
###Feedback:

Figure 2: The direct scoring prompt used in our work following (Kim et al., 2024b).

Direct Scoring Prompt with Reference Answer

###Task Description:
An instruction (might include an Input inside it), a response to evaluate, a reference answer that gets a score of 5, and a score rubric representing a evaluation criteria are given.
1. Write a detailed feedback that assess the quality of the response strictly based on the given score rubric, not evaluating in general.
2. After writing a feedback, write a score that is an integer between 1 and 5. You should refer to the score rubric.
3. The output format should look as follows: "Feedback: (write a feedback for criteria) [RESULT] (an integer number between 1 and 5)
4. Please do not generate any other opening, closing, and explanations.
###The instruction to evaluate:
{orig_instruction}
###Response to evaluate:
{orig_response}
###Reference Answer (Score 5):
{orig_reference_answer}
###Score Rubrics:
[{score_criteria}]
Score 1: {score1_description}
Score 2: {score2_description}
Score 3: {score3_description}
Score 4: {score4_description}
Score 5: {score5_description}
###Feedback:

Figure 3: The direct scoring prompt with reference answer used in our work following (Kim et al., 2024b).

RANKER-AS-SCORER Prompt

###Task Description:
An instruction (might include an Input inside it), a response to evaluate, and a score rubric representing a evaluation criteria are given.
1. Write a detailed feedback that assess the quality of the response strictly based on the given score rubric, not evaluating in general.
2. After writing the feedback, rank all possible scores (1 to 5) from most to least likely for this response, according to the score rubric. For each score, briefly explain why the response could or could not fit that score.
3. The output format should look as follows: "Feedback: (write the reason for ranking for criteria) [RESULT] (most likely score) > (second most likely score) > ... > (least likely score)"
4. Please do not generate any other opening, closing, and explanations.
###The instruction to evaluate:
{orig_instruction}
###Response to evaluate:
{orig_response}
###Score Rubrics:
[{score_criteria}]
Score 1: {score1_description}
Score 2: {score2_description}
Score 3: {score3_description}
Score 4: {score4_description}
Score 5: {score5_description}
###Feedback:

Figure 4: The RANKER-AS-SCORER prompt.