Co-Eval: Augmenting LLM-based Evaluation with Machine Metrics

Anonymous ACL submission

Abstract

Large language models (LLMs) are increasingly used as evaluators in natural language generation tasks, offering advantages in scalability and interpretability over traditional evaluation methods. However, existing LLM-based evaluations often suffer from biases and misalignment, particularly in domain-specific tasks, due to limited functional understanding and knowledge gaps. To address these challenges, we first investigate the relationship between an LLM-based evaluator's familiarity with the target task and its evaluation performance. We then introduce the Co-Eval framework, which leverages a criteria planner model and optimized machine metrics to enhance the scalability and fairness of LLM-based evaluation. Experimental results on both general and domain-specific tasks demonstrate that Co-Eval reduces biases, achieving up to a 0.4903 reduction in self-preference bias, and improves alignment with human preferences, with gains of up to 0.324 in Spearman correlation.

1 Introduction

Evaluating the quality of natural language generation (NLG) is inherently challenging due to the subjective nature of such tasks, where the criteria for high-quality output can vary based on context and audience. While human evaluation remains a common method for assessing generated text, it is also time-consuming. Recently, researchers (Liu et al., 2023; Chan et al., 2023; Zheng et al., 2023a) have turned to large language models (LLMs) as evaluators, noting their impressive alignment with human preferences in text assessment.

However, studies (Koo et al., 2023; Panickssery et al., 2024) have revealed that LLMs exhibit biases, such as a preference for text generated by the models themselves. Additionally, factors like presentation order (Wang et al., 2023) and text length (Hu et al., 2024) can affect the fairness of LLM evaluations. General-purpose LLMs also tend to under-

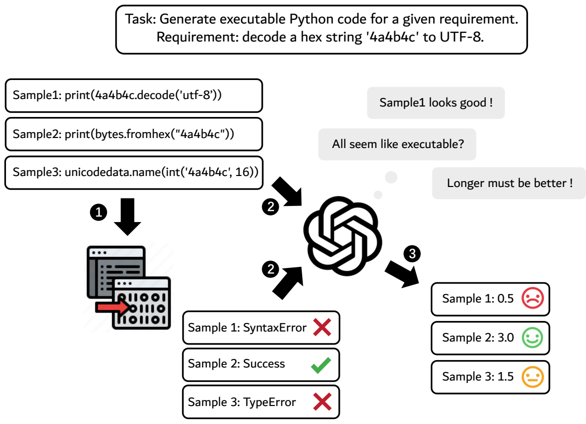


Figure 1: Machine metrics augment scalability and fairness of LLM-based evaluation.

perform when evaluating NLG tasks in specialized domains (Dorner et al., 2025), which can either exacerbate or mitigate biases depending on the task. These limitations raise concerns about the objectivity of LLMs, especially when evaluating domain-specific tasks, where their performance may still be prone to significant "hallucinations" (Tong and Zhang, 2024; Davoodi et al., 2025).

To mitigate these issues, prior research has attempted to reduce biases by incorporating reference points (Jiao et al., 2024; Lu et al., 2023) and enhancing human evaluation (Xu et al., 2024b) or using multi-agent systems (Chan et al., 2023). However, these methods are often too resource-intensive for real-time applications in online systems.

In this paper, we first investigate how self-preference, position, verbosity, and format biases in LLM-based evaluations vary with the model's familiarity with the target task. Using translation as a case study, we reflect an LLM's familiarity through its performance in the target language. We then introduce Co-Eval, a zero-shot reference-free LLM-based evaluation framework that enhances LLM-based evaluation through domain-specific machine metrics. Recognizing that individual metrics often assess only specific aspects of a task, we fine-tuned a LLaMA-3.1-8B-Instruct model to serve as a criteria planner. This planner interprets diverse task descriptions to establish evaluation criteria, assign weights, and generate score-level descriptions. Next, we developed a comprehensive machine metrics library to link relevant metrics to the generated criteria based on similarity of their description. The criteria planner is then utilized to refine the machine metric descriptions, ensuring they align closely with the specified criteria. Finally, the prompt-based LLM evaluator is used to generate the final evaluation of each sample, with the overall score calculated as a weighted sum across criteria.

Extensive experiments conducted across multiple tasks, including four domain-specific ones, demonstrate that the Co-Eval framework enhances LLM-based evaluators. It reduces self-preference bias by up to 0.4903 and improves agreement with human preferences by up to 0.324 on domain-specific tasks.

To summarize, the main contributions of this paper are as follows:

- We conduct an in-depth study of the biases in LLM-based evaluators, focusing on how an LLM's familiarity with the task being evaluated can influence its judgments, and uncover several meaningful insights.
- We introduce Co-Eval, a novel LLM-based evaluation framework that enhances scalability and fairness in evaluation by incorporating machine metrics. We also provide a theoretical proof and extensive experiments to demonstrate that our framework reduces bias in LLM-based evaluations and improves alignment with human preferences.
- We present a multi-task supervised fine-tuning dataset for the criteria planner, along with a comprehensive machine metric library that includes approximately 50 machine metrics with their implementations.

2 Related Work

2.1 Metric-based Evaluation

Formula-based metrics rely on predefined rules to evaluate the quality of generated responses. Examples include BLEU (Papineni et al., 2002) and METEOR (Banerjee and Lavie, 2005) for machine translation tasks, ROUGE (Lin, 2004) for text summarization, and Flesch-Kincaid score (Flesch, 1943) for readability in educational content.

Model-based metrics leverage pre-trained neural networks to assess the quality of generated responses. For example, BERTScore (Zhang et al., 2019) computes cosine similarity between BERT embeddings (Devlin, 2018), while GPTScore (Fu et al., 2023) utilizes embeddings from GPT (Radford, 2018). More recently, like UNIEVAL (Zhong et al., 2022), improve embedding-based evaluation by incorporating multiple evaluation dimensions.

Both kinds of machine metrics offer reliable and consistent evaluations but are constrained by their applicability. When used for inappropriate tasks, they can introduce significant biases, leading to misalignment with human preferences.

2.2 LLM-based Evaluation

LLM-based evaluation methods utilize LLMs as sophisticated judges of text quality, often referred to as LLMs-as-judges (Ashktorab et al., 2024; Bavaresco et al., 2024; Tseng et al., 2024).

Prompt-based methods aim to teach LLMs how to evaluate complex tasks through in-context learning. This includes providing fine-grained task criteria (Liu et al., 2023; Zhuo, 2024; Yi et al., 2024; Song et al., 2024a), learning from examples (shot learning) (Fu et al., 2024; Lin and Chen, 2023; Zhang et al., 2024; Jain et al., 2023; Song et al., 2024b), or breaking into multiple iterations (Hasanbeig et al., 2023; Chiang and Lee, 2023; Liu et al., 2024b; Xu et al., 2024a; Saha et al., 2024).

Tuning-based methods (Deshwal and Chawla, 2024; Yue et al., 2023; Ye et al., 2024b; Wang et al., 2024; He et al., 2024; Kim et al., 2024; Liu et al., 2024a; Ke et al., 2024), on the other hand, involve training a pre-existing LLM on a specialized dataset to adapt it to specific judgment tasks.

Unlike single-LLM systems, Multi-LLM evaluation (Liang et al., 2024; Zhao et al., 2024a; Moniri et al., 2025; Chan et al., 2023) leverages the collective intelligence of multiple LLMs to enhance evaluation performance.

Despite extensive research, issues such as hallucinations and domain-specific knowledge gaps undermine the robustness of LLM-based evaluation, manifesting as biases, including self-preference bias (Li et al., 2024; Panickssery et al., 2024), position bias (Shi et al., 2024; Zhao et al., 2024b), and verbosity bias (Chen et al., 2024; Zheng et al., 2023b). Avoiding self-evaluation (Ye et al., 2024a) and reference-based approaches (Badshah and Saj-

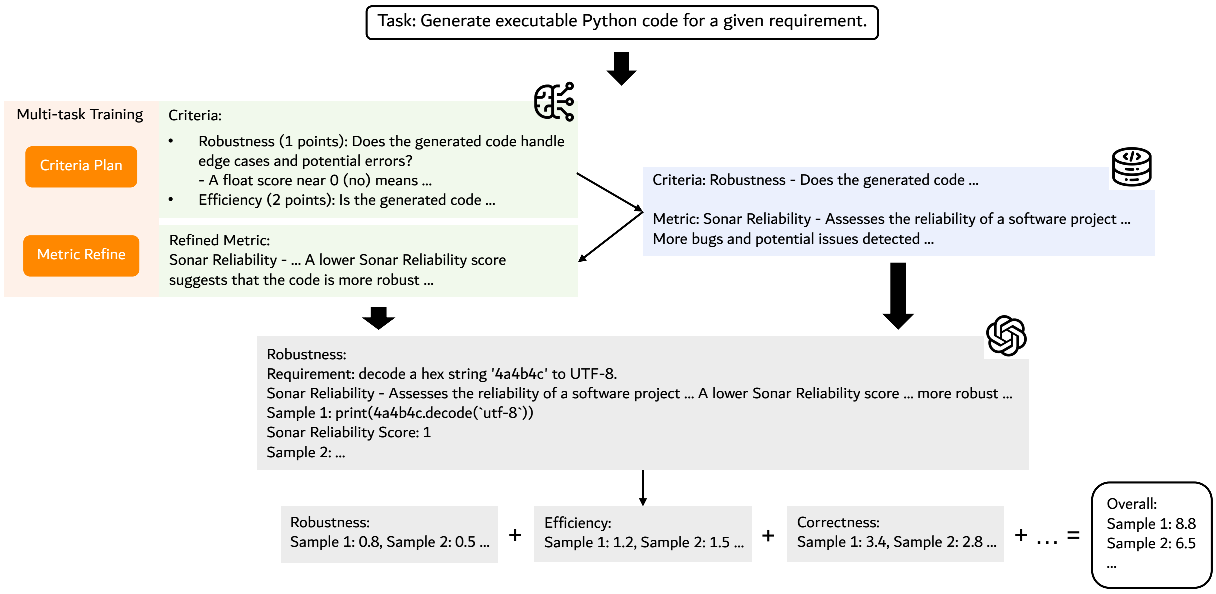


Figure 2: An overview of Co-Eval framework on executable Python code generation task. First, a fine-tuned criteria planner generates scoring criteria and corresponding weights for evaluating the task. Next, each criterion is matched with suitable machine metrics from a machine metric library based on semantic similarity between their descriptions. The chosen machine metrics are then refined by the criteria planner to specify how changes in their scores reflect the performance of the generated code against the criteria. Finally, the task description, original requirement, generated code, machine metric descriptions, and scores are input to a prompt-based evaluator to assign scores to each criterion. These scores are weighted and summed to produce the final evaluation score for each sample.

jad, 2024) have proven effective in mitigating self-preference bias. However, obtaining accurate models and references can be challenging for open-ended tasks. Additionally, swap-based methods (Raina et al., 2024; Wang et al., 2023) have been shown to effectively address position bias.

3 Methodology

To enhance the scalability and fairness of LLM-based evaluators, we propose the Co-Eval framework, outlined in Figure 2.

3.1 Criteria Planner

The main tasks of the criteria planner are to generate evaluation criteria and refine the descriptions of machine metrics.

For the criteria plan task, we recognize that machine metrics are suited for assessing well-defined criteria, which improves accuracy but limits scalability. Furthermore, criteria and their weights must be highly responsive to subtle differences across tasks, as even slight task variations can result in significant shifts in criteria and corresponding weights. Previous research (Kim et al., 2023) has also shown that using fine-grained criteria improves the performance of LLM-based evaluators. Therefore, a criteria planner is needed that can break down task criteria into fine-grained machine metrics and score-level descriptions, adjusting criteria and weights to capture nuanced task differences effectively.

For the metric refine task, we observe that machine metric descriptions tend to be straightforward, focusing mainly on the applicability of each metric rather than linking scores to criteria performance. To address this, we refine the machine metric descriptions to better reflect their relationship to the criteria being assessed, rather than using them directly in a prompt-based evaluation setting.

Data Preparation We constructed a multi-task supervised fine-tuning dataset comprising a total of 950 samples. For the criteria planning task, we developed a dataset with 500 task descriptions and corresponding criteria descriptions. Among these, 250 task descriptions were collected from agent platforms such as Coze[1] and GPT-Shop[2], while the remaining 250 were generated by GPT-4o following a consistent format to ensure diversity and coverage. For the metric refinement task, we used the 500 criteria produced in the criteria planning task. For 250 of these criteria, we searched a metric library to identify suitable metrics and had GPT-4o generate refined metric descriptions. For the

[1] https://www.coze.com
[2] https://chatgpt.com/gpts

remaining 250 criteria, GPT-4o was tasked with both generating suitable metrics and refining their descriptions. To ensure the quality and consistency of the dataset, we extracted the required information from the initial outputs, reorganized them into a standardized format, and filtered out 50 outputs with missing key information. The prompt used for data preparation is detailed in Appendix E.

Training Strategy Our primary objective is to distill GPT-4o's performance on criteria planning and metric description refinement tasks, as well as to correct the output format bias of the Llama-3.1-8B-Instruct-based planner, enhancing its suitability for downstream tasks. Given that our training data consists of no more than 1,000 samples and the target task aligns closely with the native capabilities of the Llama-3.1-8B-Instruct model, we employ LoRA (Hu et al., 2021) as our fine-tuning method.

3.2 Machine Metrics Library

We compiled approximately 50 machine metrics into a comprehensive library, broadly categorized into two types. The theoretical justification for our approach is provided in Appendix D.

Formula-based Metrics rely on deterministic rules and structured patterns to evaluate specific criteria in generated outputs. These metrics offer precise assessments that LLMs may find difficult to approximate. For instance, a syntax parser can reliably verify whether generated code is syntactically correct and compilable, an evaluation task that often exceeds the capabilities of LLMs. Additionally, formula-based metrics play a crucial role in guiding LLM-based evaluators toward better alignment with human preferences, which are often embedded in the design of these metrics. For example, in summarization tasks, an Information Density formula can prioritize brevity and the inclusion of key information.

Model-based Metrics utilize well-trained deep neural networks to assess domain-specific evaluation criteria. While LLMs excel at general-purpose generation, we emphasize smaller, domain-specific models trained on specialized corpora, which tend to be more robust in their respective domains. For instance, a BERT model fine-tuned on a financial corpus may more accurately assess contextual similarity within financial documents than a general-purpose LLM. As such, these metrics enhance the domain sensitivity of LLM-based evaluators.

These metrics are produced by compact, domain-specialized neural models. A BERT model fine-tuned on a financial corpus, for example, can judge topical coherence in corporate filings more reliably than a general-purpose LLM.

	Chinese	French	Spanish	Thai	Ukrainian	Vietnamese
Qwen-2.5-72B	0.3125	0.4601	0.5144	0.2674	0.3255	0.3767
Qwen-2.5-7B	0.2366	0.3978	0.4316	0.1884	0.2593	0.2827
Llama-3.1-70B	0.3128	0.5456	0.6252	0.3140	0.4244	0.4843
Llama-3.1-8B	0.1627	0.4041	0.5180	0.0861	0.1435	0.0576
Gemma-2-27B	0.2827	0.3531	0.5350	0.2718	0.3749	0.4494
Gemma-2-9B	0.2977	0.4533	0.4975	0.2429	0.3206	0.3351

Table 1: BLEU Scores for Back-translations.

Since conventional descriptions of machine metrics often fail to clarify which aspects of the data influence score changes, we aim to enhance interpretability by identifying the specific data features that impact each metric. To this end, we provide GPT-4o with pairwise evaluation samples for every metric, enabling it to generate more precise descriptions that reflect the features each metric effectively captures within context.

3.3 Prompt-based Evaluator

For the final LLM-based evaluator, we adopt the in-context learning and batchwise evaluation methods from BATCHEVAL (Yuan et al., 2023), along with its input and output formatting. To encourage the LLM to reason based on the provided metric scores, rather than blindly following them, we include an explicit instruction in the prompt. The full prompt template is provided in Appendix E.

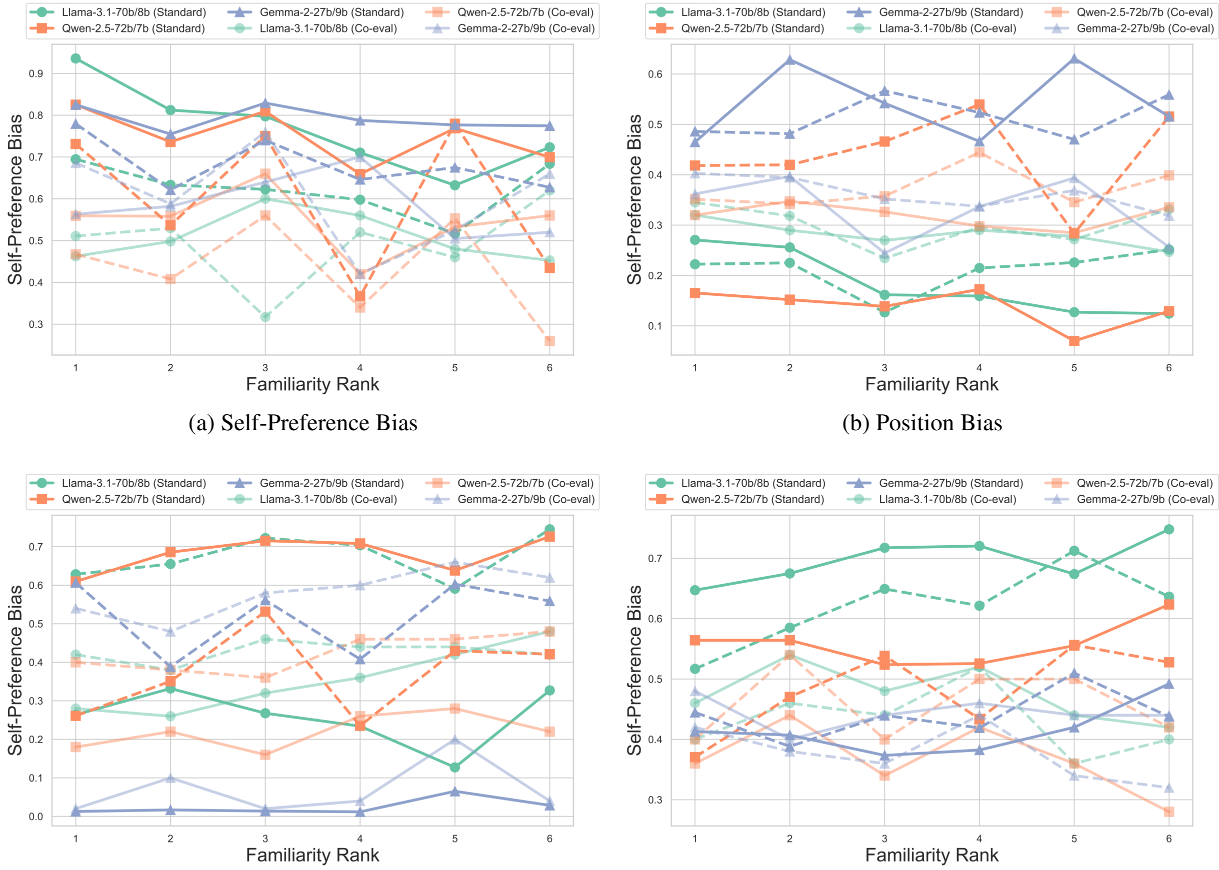
4 Experiment

4.1 Experimental Settings

The criteria planner model, based on the Llama-3.1-8B-Instruct model, was fine-tuned by LoRA (Hu et al., 2021) for 3 epochs with a learning rate of 1.0e-4, a cosine scheduler, and a warmup ratio of 0.1. We set a total score of 10 with a maximum of 5 evaluation criteria.

In the machine metric search, we select the top three metrics with embedding similarity scores exceeding 0.8, where cosine similarity is employed, averaging scores across five evaluation runs. Detailed descriptions of LLMs used as prompt-based evaluators and baselines are provided in Appendix A and Appendix B, respectively.

Experiments show that our Co-Eval framework enhances the scalability and fairness of LLM-based evaluation, especially in domain-specific tasks. Detailed experimental implementation information for each benchmark is provided in Appendix C.



(a) Self-Preference Bias

(b) Position Bias

(c) Verbosity Bias

(d) Format Bias

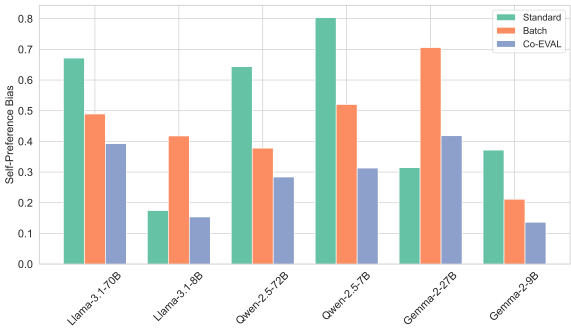
Figure 3: Impact of familiarity on four types of bias. In each subfigure, the horizontal axis represents the rank of the LLM's familiarity with six different languages. The higher the rank, the lower the model's familiarity with that language. For self-preference bias, the vertical axis shows the rate at which the LLM's own translation is ranked first compared to the gold reference. For position bias, the vertical axis indicates the rate at which a response is ranked first when it appears in the first position. For verbosity bias, the vertical axis reflects the rate at which more verbose responses are ranked first. For format bias, the vertical axis shows the rate at which formatted responses are ranked first. In each subfigure, solid lines represent results from the larger model in the LLM family, while dashed lines represent the smaller model.

4.2 Bias with LLM's Familiarity

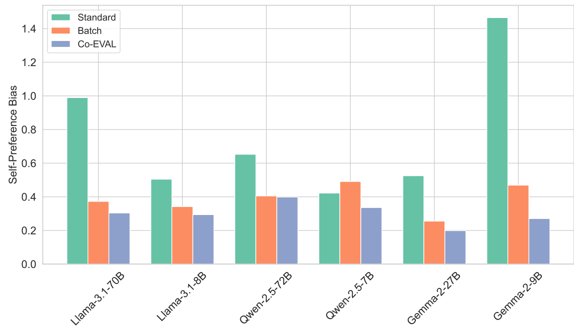
At the beginning of our experiment, we aim to investigate how an LLM's familiarity with a target task affects the performance of LLM-based evaluators. We choose translation as the target task because the input remains consistent across different target languages, allowing us to isolate the LLM's task familiarity based on its familiarity with the target language.

For our analysis, we use the FLORES benchmark (Costa-jussà et al., 2022) and three state-of-the-art LLMs as final prompt-based evaluators: Qwen-2.5-72B, Llama-3.1-70B, and Gemma-2-27B. These models translate English content into six target languages: Chinese, Thai, Spanish, French, Ukrainian, and Vietnamese. To estimate language familiarity, we follow Zhuo et al. (Zhuo et al., 2023). Each model performs translation and back-translation, and we compute BLEU scores between the original and back-translated English. Higher scores indicate greater familiarity. We use the first 100 FLORES samples, and average BLEU scores are shown in Table 1.

Figure 3 shows how language familiarity relates to self-preference, position, verbosity, and format biases. Self-preference bias is measured by how often the LLM prefers its own output over the gold reference. Position bias is assessed by rotating the same responses through positions 1–3 and recording how often each ranks first. Verbosity bias follows Zheng et al. (Zheng et al., 2023b). GPT-4o is used to generate extended and shortened versions

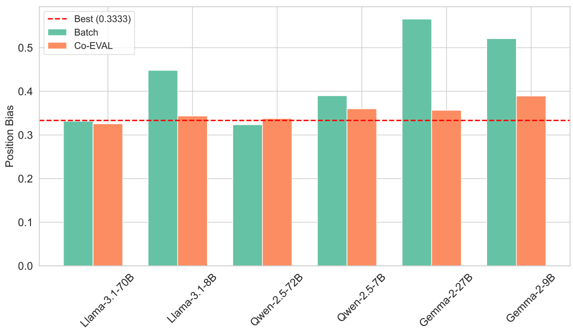


(a) CoNaLa

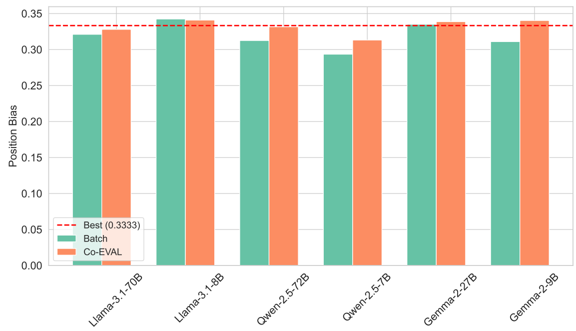


(b) MATH

Figure 4: Self-preference bias on CoNaLa and MATH benchmarks. The standard method refers to evaluating each response individually by sending it to the LLM-based evaluator one at a time. In contrast, the batch method involves presenting a group of responses together. For measuring self-preference bias, we batch the responses generated by the six LLMs and evaluate them collectively.



(a) CoNaLa



(b) MATH

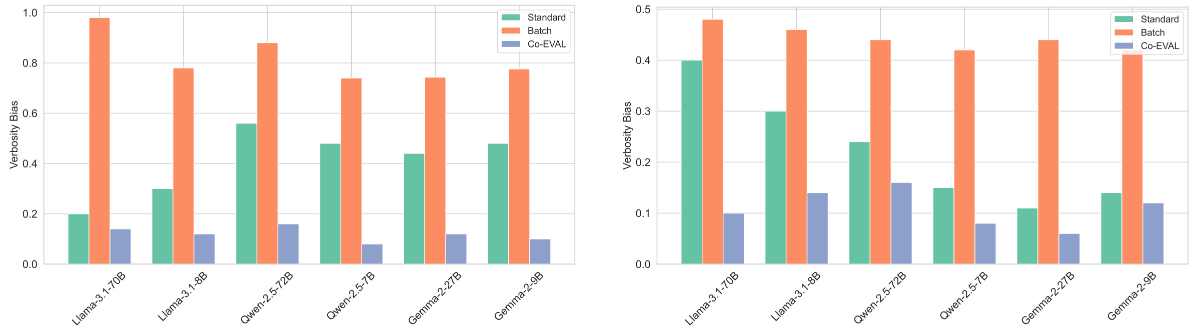
Figure 5: Position bias on CoNaLa and MATH benchmarks. To evaluate position bias, we batch the larger LLMs from three different families and rotate their positions. The red line represents the expected ideal rate, 0.3333, at which a response in position 1 is ranked first. This assumes that when the same response is randomly placed in positions 1 to 3, it has an equal probability of being ranked first, regardless of its position.

of Llama-3.1-70B's outputs, and we measure preference for longer responses. Format bias is tested by applying pre-defined formatting to Llama-3.1-70B outputs and measuring preference for the formatted version.

Analysis The results highlight key patterns in how model familiarity influences bias. First, as familiarity with the target language decreases, self-preference bias declines, while format bias rises. This suggests that when less confident, LLMs rely less on their own outputs and more on superficial cues like layout or structure. In contrast, position and verbosity biases remain stable, likely stemming from internal mechanisms such as reading preferences or decoding behavior that are less affected by content familiarity.

Second, while both small and large LLMs follow similar trends in most biases, format bias diverges sharply. Smaller models behave similarly to each other, as do larger models, but the two groups differ in formatting tendencies. This is due to differences in post-training exposure: larger models typically undergo more instruction tuning and structured-output training (e.g., JSON, XML), leading to more standardized formatting, while smaller models retain more varied styles.

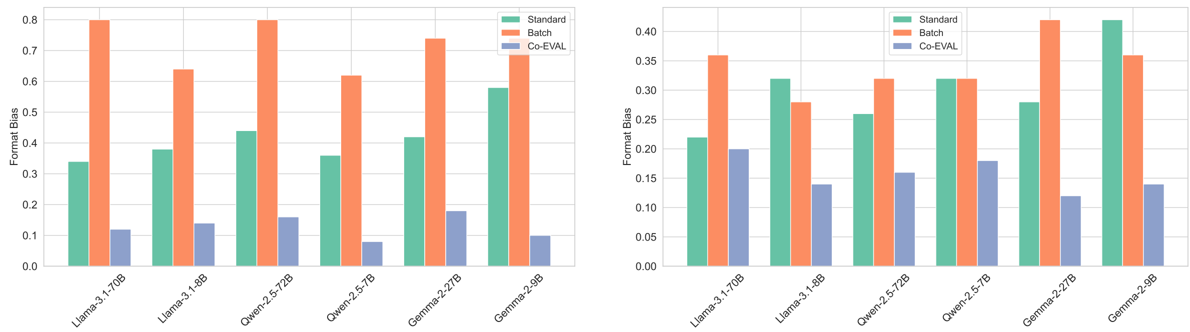
Finally, self-preference bias increases with model size, reflecting better recognition of self-generated text and alignment with reward signals. No clear size-related trends emerge for position, verbosity, or format bias. Position bias stems from the transformer's attention mechanics, verbosity bias reflects early-learned length-helpfulness associations, and format bias is shaped more by fine-tuning data than model scale.



(a) CoNaLa

(b) MATH

Figure 6: Verbosity bias on CoNaLa and MATH benchmarks. To evaluate verbosity bias, we batch the original responses generated by Llama-3.1-70B-Instruct together with their extended versions that contain intentional errors.



(a) CoNaLa

(b) MATH

Figure 7: Format bias on CoNaLa and MATH benchmarks. To assess format bias, we batch the original responses generated by Llama-3.1-70B-Instruct together with their formatted counterparts that contain intentional errors.

4.3 Effectiveness on Bias Elimination

We demonstrate the effectiveness of the Co-Eval framework in mitigating four types of bias—self-preference bias, position bias, verbosity bias, and format bias, across four domain-specific tasks: CoNaLa (Yin et al., 2018) for Python code generation, MATH for mathematical problem solving, FIQA for financial question answering, and Health Counseling for open-ended medical advice and patient guidance. We employ six state-of-the-art LLMs to generate responses for each benchmark and also use them as prompt-based evaluators in the final assessment.

Self-preference Bias For some domain-specific tasks, the ground truth may guarantee correctness but does not necessarily represent the best or most informative answer. For example, in open-ended medical consultations, a ground truth response might list accurate symptoms but omit critical diagnostic insights that a more helpful answer would include. To quantify the tendency of LLM-based evaluators to favor their own outputs, we define self-preference bias as the degree to which an LLM ranks its own responses higher than others do. Specifically, we measure it using the following equation:

$$Bias(i) = \frac{1}{N}\sum_{i=1}^{N} \max(0, R_o(i) - R_s(i)), \quad (1)$$

where $R_s(i)$ is the rank assigned by the LLM-based evaluator to its self-generated result for instance i, $R_o(i)$ is the average rank assigned by other evaluators, N is the total number of instances.

As shown in the results on the four benchmarks in Figures 4 and Figure 8, the Co-Eval framework effectively reduces self-preference bias across all six LLM evaluators. Notably, smaller LLMs exhibit more significant shifts when guided by machine metric scores.

Position Bias As shown in Figures 5 and 9, the Co-Eval framework brings the ranking rate of a response placed in position 1 much closer to the ideal expectation. This indicates that the LLM-

Method	Model	Human Align		Self-Prefer	Position	Verbosity	Format	Tokens/items	
		ρ	τ					Input	Output
Standard	Llama-3.1-70B	0.223	0.205	0.671	-	0.219	0.338	563	891
Batch	Llama-3.1-70B	0.453	0.419	0.489	0.331	0.981	0.797	267	729
	CodeLlama-70B	0.259	0.214	-	-	-	-	-	-
Co-Eval	Llama-3.1-70B	0.547	0.492	0.392	0.325	0.143	0.119	604	1208
	- Criteria Planner	0.443	0.406	0.402	0.321	0.247	0.318	-	-
	- Planner Fine-tuning	0.465	0.420	0.431	0.337	0.232	0.194	-	-
	- Metric Refinement	0.528	0.488	0.411	0.330	0.159	0.125	-	-
	- Metric Library	0.489	0.413	0.476	0.343	0.878	0.691	-	-
	+ GPT-4o as Planner	0.551	0.521	0.387	0.322	0.148	0.203	-	-
	+ Random Metric	0.438	0.392	0.463	0.356	0.349	0.474	-	-

Table 2: Human Alignment and Ablation Study on CoNaLa benchmarks.

based evaluator achieves a more balanced ranking behavior, allowing the same response to consistently attain the top rank regardless of its position within the batch.

Verbosity Bias and Format Bias We argue that simply measuring how often extended or well-formatted responses are ranked first does not reliably indicate bias, since length and format can sometimes correlate with higher quality. To better reveal potential bias, we not only extend or reformat responses but also introduce subtle factual or functional errors into the modified content. In doing so, we uncover the hallucination tendencies of LLMs during evaluation. As shown in Figures 6, 10, 7, and 11, we observe significant hallucinations in LLM-based evaluations. Compared to the standard method, evaluators using the batch method show a stronger preference for more verbose responses, even when these contain functional errors. The Co-Eval framework mitigates this issue by improving the evaluator's ability to detect such errors, resulting in more balanced rankings across responses of varying verbosity and format.

Based on the results above, the Co-Eval framework demonstrates outstanding effectiveness in mitigating self-preference bias, position bias, and verbosity bias. In summary, Co-Eval framework can significantly **improves the fairness and scalability of LLM-based evaluation**.

4.4 Human Alignment and Ablation Study

For human alignment on CoNaLa benchmark, we recruited three annotators had at least one year of Python experience. They assessed generated responses based on correctness, readability, coding standards, and alignment with task requirements. They were encouraged to run the code when needed. Final scores were averaged across three annotators.

As shown in Table 2, Co-Eval outperforms standard and batch evaluation methods on both the CoNaLa benchmarks, achieving the highest correlations. and with the state-of-the-art model as criteria planner, the performance of Co-Eval framework can further improvement.

In the ablation study, we find that the effectiveness of bias reduction primarily arises from the incorporation of retrieved machine metrics, particularly functional metrics such as compilers, which inject domain-specific knowledge into the evaluation process and promote consistency across different LLMs. The observed improvements in alignment with human preferences are largely attributable to the relevance of the retrieved metrics. Notably, Co-Eval with retrieved metrics achieves over a 10-point performance gain compared to using randomly selected metrics, which can even impair model judgment. Similarly, applying an inappropriate evaluation criterion can significantly degrade performance. A detailed case study is provided in Appendix G, including a comparison before and after fine-tuning the criteria planner, an overview of machine metrics, and error analysis.

5 Conclusion

In this paper, we present Co-Eval, a zero-shot LLM-based evaluation framework that enhances scalability and fairness. The Co-Eval framework integrates machine metrics into the prompt-based evaluator by utilizing a fine-tuned criteria planner and a comprehensive library of metrics. This approach addresses limitations such as bias and misalignment, which arise from inaccurate recognition of functional correctness and gaps in domain-specific knowledge.

Limitations

Although we demonstrate the effectiveness of our proposed Co-Eval framework, several limitations remain:

- While we have collected machine metrics for natural language generation tasks across a diverse set of domains, including general, code, mathematical, health, and financial, it remains challenging to cover all potential metrics. There is considerable room for expanding the range of machine metrics to enhance coverage.
- Our metric retrieval algorithm currently depends on semantic similarity between criteria descriptions and metric descriptions. However, this approach lacks adaptability, and mismatches in metric selection may mislead the LLM-based evaluator.
- The Co-Eval framework is primarily designed to support LLM-based evaluation, meaning its overall effectiveness largely relies on the capabilities of the LLM, which serves as a prompt-based evaluator. This factor lies beyond the scope of this paper.

References

Amod. 2024. Mental health counseling conversations (revision 9015341).

Zahra Ashktorab, Michael Desmond, Qian Pan, James M. Johnson, Martin Santillan Cooper, Elizabeth M. Daly, Rahul Nair, Tejaswini Pedapati, Swapnaja Achintalwar, and Werner Geyer. 2024. Aligning human and llm judgments: Insights from evalassist on task-specific evaluations and ai-assisted assessment strategy preferences. *Preprint*, arXiv:2410.00873.

Sher Badshah and Hassan Sajjad. 2024. Reference-guided verdict: Llms-as-judges in automatic evaluation of free-form text. *Preprint*, arXiv:2408.09235.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Anna Bavaresco, Raffaella Bernardi, Leonardo Bertolazzi, Desmond Elliott, Raquel Fernández, Albert Gatt, Esam Ghaleb, Mario Giulianelli, Michael Hanna, Alexander Koller, André F. T. Martins, Philipp Mondorf, Vera Neplenbroek, Sandro Pezzelle, Barbara Plank, David Schlangen, Alessandro Suglia, Aditya K Surikuchi, Ece Takmaz, and Alberto Testoni. 2024. Llms instead of human judges? a large scale empirical study across 20 nlp evaluation tasks. *Preprint*, arXiv:2406.18403.

Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. 2023. Chateval: Towards better llm-based evaluators through multi-agent debate. *arXiv preprint arXiv:2308.07201*.

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

Cheng-Han Chiang and Hung-yi Lee. 2023. A closer look into using large language models for automatic evaluation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 8928–8942, Singapore. Association for Computational Linguistics.

Marta R Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, et al. 2022. No language left behind: Scaling human-centered machine translation. *arXiv preprint arXiv:2207.04672*.

Arash Gholami Davoodi, Seyed Pouyan Mousavi Davoudi, and Pouya Pezeshkpour. 2025. LLMs are not intelligent thinkers: Introducing mathematical topic tree benchmark for comprehensive evaluation of LLMs. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3127–3140, Albuquerque, New Mexico. Association for Computational Linguistics.

Mahesh Deshwal and Apoorva Chawla. 2024. Phudge: Phi-3 as scalable judge. *Preprint*, arXiv:2405.08029.

Jacob Devlin. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Florian E. Dorner, Vivian Y. Nastl, and Moritz Hardt. 2025. Limits to scalable evaluation at the frontier: Llm as judge won't beat twice the data. *Preprint*, arXiv:2410.13341.

Alexander R Fabbri, Wojciech Kryściński, Bryan McCann, Caiming Xiong, Richard Socher, and Dragomir Radev. 2021. Summeval: Re-evaluating summarization evaluation. *Transactions of the Association for Computational Linguistics*, 9:391–409.

Rudolf Flesch. 1943. Marks of readable style; a study in adult education. *Teachers College Contributions to Education*.

Jinlan Fu, See-Kiong Ng, Zhengbao Jiang, and Pengfei Liu. 2023. Gptscore: Evaluate as you desire. *arXiv preprint arXiv:2302.04166*.

Jinlan Fu, See-Kiong Ng, Zhengbao Jiang, and Pengfei Liu. 2024. GPTScore: Evaluate as you desire. 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 6556–6576, Mexico City, Mexico. Association for Computational Linguistics.

Karthik Gopalakrishnan, Behnam Hedayatnia, Qinlang Chen, Anna Gottardi, Sanjeev Kwatra, Anu Venkatesh, Raefer Gabriel, and Dilek Hakkani-Tur. 2023. Topical-chat: Towards knowledge-grounded open-domain conversations. *arXiv preprint arXiv:2308.11995*.

Hosein Hasanbeig, Hiteshi Sharma, Leo Betthauser, Felipe Vieira Frujeri, and Ida Momennejad. 2023. Allure: Auditing and improving llm-based evaluation of text using iterative in-context-learning. *Preprint*, arXiv:2309.13701.

Yuanqin He, Yan Kang, Lixin Fan, and Qiang Yang. 2024. Fedeval-llm: Federated evaluation of large language models on downstream tasks with collective wisdom. *Preprint*, arXiv:2404.12273.

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*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Zhengyu Hu, Linxin Song, Jieyu Zhang, Zheyuan Xiao, Jingang Wang, Zhenyu Chen, Jieyu Zhao, and Hui Xiong. 2024. Rethinking llm-based preference evaluation. *arXiv preprint arXiv:2407.01085*.

Sameer Jain, Vaishakh Keshava, Swarnashree Mysore Sathyendra, Patrick Fernandes, Pengfei Liu, Graham Neubig, and Chunting Zhou. 2023. Multi-dimensional evaluation of text summarization with in-context learning. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8487–8495, Toronto, Canada. Association for Computational Linguistics.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Tong Jiao, Jian Zhang, Kui Xu, Rui Li, Xi Du, Shangqi Wang, and Zhenbo Song. 2024. Enhancing fairness in llm evaluations: Unveiling and mitigating biases in standard-answer-based evaluations. *Proceedings of the AAAI Symposium Series*, 4(1):56–59.

Pei Ke, Bosi Wen, Andrew Feng, Xiao Liu, Xuanyu Lei, Jiale Cheng, Shengyuan Wang, Aohan Zeng, Yuxiao Dong, Hongning Wang, Jie Tang, and Minlie Huang. 2024. CritiqueLLM: Towards an informative critique generation model for evaluation of large language model generation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13034–13054, Bangkok, Thailand. Association for Computational Linguistics.

Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, et al. 2023. Prometheus: Inducing fine-grained evaluation capability in language models. In *The Twelfth International Conference on Learning Representations*.

Seungone Kim, Juyoung Suk, Shayne Longpre, Bill Yuchen Lin, Jamin Shin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. 2024. 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, Miami, Florida, USA. Association for Computational Linguistics.

Ryan Koo, Minhwa Lee, Vipul Raheja, Jong Inn Park, Zae Myung Kim, and Dongyeop Kang. 2023. Benchmarking cognitive biases in large language models as evaluators. *arXiv preprint arXiv:2309.17012*.

Ruosen Li, Teerth Patel, and Xinya Du. 2024. Prd: Peer rank and discussion improve large language model based evaluations. *Preprint*, arXiv:2307.02762.

Sirui Liang, Baoli Zhang, Jun Zhao, and Kang Liu. 2024. ABSEval: An agent-based framework for script evaluation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 12418–12434, Miami, Florida, USA. Association for Computational Linguistics.

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

Yen-Ting Lin and Yun-Nung Chen. 2023. LLM-eval: Unified multi-dimensional automatic evaluation for open-domain conversations with large language models. In *Proceedings of the 5th Workshop on NLP for Conversational AI (NLP4ConvAI 2023)*, pages 47–58, Toronto, Canada. Association for Computational Linguistics.

Minqian Liu, Ying Shen, Zhiyang Xu, Yixin Cao, Eunah Cho, Vaibhav Kumar, Reza Ghanadan, and Lifu Huang. 2024a. X-eval: Generalizable multi-aspect text evaluation via augmented instruction tuning with auxiliary evaluation aspects. 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 8560–8579, Mexico City, Mexico. Association for Computational Linguistics.

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

Yuxuan Liu, Tianchi Yang, Shaohan Huang, Zihan Zhang, Haizhen Huang, Furu Wei, Weiwei Deng, Feng Sun, and Qi Zhang. 2024b. Calibrating LLM-based evaluator. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 2638–2656, Torino, Italia. ELRA and ICCL.

Qingyu Lu, Liang Ding, Liping Xie, Kanjian Zhang, Derek F. Wong, and Dacheng Tao. 2023. Toward human-like evaluation for natural language generation with error analysis. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5892–5907, Toronto, Canada. Association for Computational Linguistics.

Shikib Mehri and Maxine Eskenazi. 2020. Usr: An unsupervised and reference free evaluation metric for dialog generation. *arXiv preprint arXiv:2005.00456*.

Behrad Moniri, Hamed Hassani, and Edgar Dobriban. 2025. Evaluating the performance of large language models via debates. *Preprint*, arXiv:2406.11044.

Arjun Panickssery, Samuel R Bowman, and Shi Feng. 2024. Llm evaluators recognize and favor their own generations. *arXiv preprint arXiv:2404.13076*.

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.

Alec Radford. 2018. Improving language understanding by generative pre-training.

Vyas Raina, Adian Liusie, and Mark Gales. 2024. Is LLM-as-a-judge robust? investigating universal adversarial attacks on zero-shot LLM assessment. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7499–7517, Miami, Florida, USA. Association for Computational Linguistics.

Swarnadeep Saha, Omer Levy, Asli Celikyilmaz, Mohit Bansal, Jason Weston, and Xian Li. 2024. Branch-solve-merge improves large language model evaluation and generation. 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 8352–8370, Mexico City, Mexico. Association for Computational Linguistics.

Lin Shi, Chiyu Ma, Wenhua Liang, Weicheng Ma, and Soroush Vosoughi. 2024. Judging the judges: A systematic study of position bias in llm-as-a-judge. *Preprint*, arXiv:2406.07791.

Hwanjun Song, Hang Su, Igor Shalyminov, Jason Cai, and Saab Mansour. 2024a. FineSurE: Fine-grained summarization evaluation using LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 906–922, Bangkok, Thailand. Association for Computational Linguistics.

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

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

Weixi Tong and Tianyi Zhang. 2024. CodeJudge: Evaluating code generation with large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 20032–20051, Miami, Florida, USA. Association for Computational Linguistics.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Yu-Min Tseng, Wei-Lin Chen, Chung-Chi Chen, and Hsin-Hsi Chen. 2024. Are expert-level language models expert-level annotators? *Preprint*, arXiv:2410.03254.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. 2023. Large language models are not fair evaluators. *arXiv preprint arXiv:2305.17926*.

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

Shuying Xu, Junjie Hu, and Ming Jiang. 2024a. Large language models are active critics in nlg evaluation. *Preprint*, arXiv:2410.10724.

Wenda Xu, Guanglei Zhu, Xuandong Zhao, Liangming Pan, Lei Li, and William Wang. 2024b. Pride and prejudice: Llm amplifies self-bias in self-refinement. In *Proceedings of the 62nd Annual Meeting of the*

Association for Computational Linguistics (Volume 1: Long Papers), pages 15474–15492.

Hongyang Yang, Xiao-Yang Liu, and Christina Dan Wang. 2023. Fingpt: Open-source financial large language models. *arXiv preprint arXiv:2306.06031*.

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

Ziyi Ye, Xiangsheng Li, Qiuchi Li, Qingyao Ai, Yujia Zhou, Wei Shen, Dong Yan, and Yiqun Liu. 2024b. Beyond scalar reward model: Learning generative judge from preference data. *Preprint*, arXiv:2410.03742.

Seungjun Yi, Jaeyoung Lim, and Juyong Yoon. 2024. Protocollm: Automatic evaluation framework of llms on domain-specific scientific protocol formulation tasks. *Preprint*, arXiv:2410.04601.

Pengcheng Yin, Bowen Deng, Edgar Chen, Bogdan Vasilescu, and Graham Neubig. 2018. Learning to mine aligned code and natural language pairs from stack overflow. In *Proceedings of the 15th international conference on mining software repositories*, pages 476–486.

Peiwen Yuan, Shaoxiong Feng, Yiwei Li, Xinglin Wang, Boyuan Pan, Heda Wang, and Kan Li. 2023. Batcheval: Towards human-like text evaluation. *arXiv preprint arXiv:2401.00437*.

Weizhe Yuan, Graham Neubig, and Pengfei Liu. 2021. Bartscore: Evaluating generated text as text generation. *Advances in Neural Information Processing Systems*, 34:27263–27277.

Xiang Yue, Boshi Wang, Ziru Chen, Kai Zhang, Yu Su, and Huan Sun. 2023. Automatic evaluation of attribution by large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 4615–4635, Singapore. Association for Computational Linguistics.

Kaiqi Zhang, Shuai Yuan, and Honghan Zhao. 2024. Talec: Teach your llm to evaluate in specific domain with in-house criteria by criteria division and zero-shot plus few-shot. *Preprint*, arXiv:2407.10999.

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

Ruochen Zhao, Wenxuan Zhang, Yew Ken Chia, Weiwen Xu, Deli Zhao, and Lidong Bing. 2024a. Auto-arena: Automating llm evaluations with agent peer battles and committee discussions. *Preprint*, arXiv:2405.20267.

Xiutian Zhao, Ke Wang, and Wei Peng. 2024b. Measuring the inconsistency of large language models in preferential ranking. In *Proceedings of the 1st Workshop on Towards Knowledgeable Language Models (KnowLLM 2024)*, pages 171–176, Bangkok, Thailand. Association for Computational Linguistics.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E Gonzalez, and Ion Stoica. 2023a. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623. Curran Associates, Inc.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023b. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623.

Ming Zhong, Yang Liu, Da Yin, Yuning Mao, Yizhu Jiao, Pengfei Liu, Chenguang Zhu, Heng Ji, and Jiawei Han. 2022. Towards a unified multi-dimensional evaluator for text generation. *arXiv preprint arXiv:2210.07197*.

Terry Yue Zhuo. 2024. ICE-score: Instructing large language models to evaluate code. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 2232–2242, St. Julian's, Malta. Association for Computational Linguistics.

Terry Yue Zhuo, Qiongkai Xu, Xuanli He, and Trevor Cohn. 2023. Rethinking round-trip translation for machine translation evaluation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 319–337, Toronto, Canada. Association for Computational Linguistics.

A Large Language Models

GPT Family (Radford, 2018), developed by OpenAI, is a series of large language models designed to understand and generate human-like text. Built on transformer architecture and pre-trained on extensive datasets, these models primarily excel in natural language generation tasks.

Llama Family (Touvron et al., 2023), developed by Meta, comprises a series of advanced open-source language models. Included within this family is CodeLlama, a domain-specific model focused on code generation. CodeLlama is trained on a substantial amount of code data, building on the foundation of the general LLaMA models to enhance its capabilities in software development tasks.

Qwen Family (Bai et al., 2023), developed by Alibaba Cloud, is distinguished by its targeted optimization for conversational AI and information retrieval. Additionally, it offers the Qwen-Math series, which enhances the mathematical performance of the general Qwen models.

Gemma Family (Team et al., 2024), developed by EleutherAI, focuses on lightweight, state-of-the-art open models, with the largest model containing 27 billion parameters.

Mixtral Family (Jiang et al., 2024), developed by Mistral AI, comprises a series of advanced open-source language models, with its notable feature being the implementation of Sparse Mixture of Experts (SMoE) architecture.

B Baselines

B.1 Formula-based

BLEU (Papineni et al., 2002) is an automated metric for evaluating the quality of machine-translated text against one or more human reference translations. In this study, since we focus on zero-shot reference-free evaluation performance of each baseline method, we calculate the BLEU score between the generated response and the source conversation concatenated with knowledge-based content from the Topical-Chat benchmark.

ROUGE (Lin, 2004) measures the overlap of n-grams, word sequences, and word pairs between a generated summary and reference summaries. Similar to BLEU, we calculate the ROUGE-L score between the generated response and the source conversation concatenated with knowledge-based content from the Topical-Chat benchmark.

B.2 Embedding-based

BERTScore (Zhang et al., 2019) leverages pre-trained BERT embeddings to capture semantic similarity between tokens in the generated and reference texts. For our evaluation, we use the source conversation concatenated with knowledge-based content as the reference text for each generated response in the Topical-Chat benchmark.

BARTScore (Yuan et al., 2021) measures the likelihood of a generated text relative to a reference text using the BART model, treating the evaluation as a text generation task itself. We also use the source conversation concatenated with knowledge-based content as the reference text for each generated response in the Topical-Chat benchmark.

B.3 Learning-based

USR (Mehri and Eskenazi, 2020) is a reference-free metric and leverages pre-trained language models and unsupervised learning techniques to estimate how well a generated response aligns with context and meets conversational quality standards.

UNIEVAL (Zhong et al., 2022) is a unified, reference-free evaluation framework designed for assessing text generation quality. It leverages pre-trained language models to assess these qualities, enabling it to handle a diverse range of text generation tasks with a consistent, robust methodology.

B.4 LLM-based

G-EVAL (Liu et al., 2023) is a generative evaluation framework for assessing the quality of generated text. It employs LLMs to directly evaluate generated text based on criteria across a variety of text generation tasks.

BATCHEVAL (Yuan et al., 2023) is a large-scale, automated evaluation framework designed to assess the quality of text generation models in batch settings. It leverages LLMs and customizable evaluation criteria, allowing it to assess aspects across diverse tasks.

C Experimental Implementation

C.1 Topical-Chat

Topical-Chat (Gopalakrishnan et al., 2023) is a large-scale open-domain conversational benchmark containing crowd-sourced conversations on diverse topics, grounded in factual knowledge, and includes human evaluation scores for generated responses across five key criteria: naturalness, coherence, engagingness, groundedness, and understandability.

C.2 Flores

Flores (Costa-jussà et al., 2022) is a benchmark designed to provide high-quality human translations of standardized sentences, enabling the evaluation of translation accuracy across low-resource and diverse linguistic settings.

C.3 CoNaLa

CoNaLa (Yin et al., 2018) is a large-scale benchmark designed for research in code generation and understanding from natural language. It includes manually curated examples of Python code paired with corresponding natural language intents.

C.4 Mental Health Counseling Conversations

Mental Health Counseling Conversations (Amod, 2024) is a comprehensive collection of conversational data designed to support research and development in the field of mental health counseling. It consists of real-world dialogues between mental health professionals and their clients, focusing on therapeutic interactions aimed at addressing various psychological issues.

C.5 MATH

MATH (Hendrycks et al., 2021) is a large-scale benchmark designed to assess mathematical reasoning abilities, featuring problems that span a wide range of topics from middle school to high school mathematics, including algebra, geometry, calculus, and more. Each problem is accompanied by a detailed step-by-step solution.

C.6 FIQA

FIQA (Yang et al., 2023) is a benchmark designed for research in financial question-answering tasks. It contains a collection of financial questions paired with corresponding answers, covering a wide range of topics such as stock markets, investments, and economic policies.

C.7 Summeval

Summeval (Fabbri et al., 2021) is a comprehensive benchmark for evaluating abstractive summarization models, featuring human evaluations of machine-generated summaries based on four key criteria: coherence, consistency, fluency, and relevance.

D Theorem Proof

D.1 Formula-based Metrics

To theoretically validate the utility of formula-based metrics, consider the following framework.

Let S denote the *true* human preference score for an output generated from a prompt X. Define the baseline evaluator as $\hat{f}(X) = \mathbb{E}[S \mid X]$. We propose to augment this evaluator using an auxiliary metric

$$M = M(X, \eta), \quad \text{where } \eta \sim \mathcal{N}(0, 1). \tag{2}$$

The explicit inclusion of noise η ensures that, conditioned on X, the random variable M still contains information about S. If M were fully deterministic in X, the arguments below would collapse into equalities without meaningful gain.

We define the augmented evaluator as

$$\hat{f}'(X) = \mathbb{E}[S \mid X, M]. \tag{3}$$

By the law of total variance,

$$\mathrm{Var}(S \mid X) = \mathbb{E}_M\big[\mathrm{Var}(S \mid X, M)\big] + \mathrm{Var}_M\big(\mathbb{E}[S \mid X, M]\big) \geq \mathrm{Var}_M(\hat{f}'(X)). \tag{4}$$

The inequality is strict whenever M is not measurable with respect to $\sigma(X)$; that is, when M provides information beyond what is already captured by X.

To operationalize this in practice, define the residual $r = h - \hat{f}$ on a calibration set with human annotations h. We project r onto the one-dimensional space spanned by M:

$$\hat{f}'(X) = \hat{f}(X) + \gamma M,$$
$$\gamma = \arg\min_c \mathbb{E}[(r - cM)^2] = \frac{\mathrm{Cov}(r, M)}{\mathrm{Var}(M)}. \tag{5}$$

Consequently,

$$\mathbb{E}[(h - \hat{f}')^2] = \big(1 - \rho_{hM}^2\big)\, \mathbb{E}[(h - \hat{f})^2], \quad \rho_{hM} = \mathrm{Corr}(h, M). \tag{6}$$

This equality holds under two conditions: (1) the correction is linear in M, and (2) the coefficient γ is optimal. For arbitrary γ, the result becomes an inequality:

$$\mathbb{E}[(h - \hat{f}')^2] \leq \big(1 - \rho_{hM}^2\big)\, \mathbb{E}[(h - \hat{f})^2]. \tag{7}$$

D.2 Model-based Metrics

Let $D = D(X)$ denote a model-based metric.

To compare the implied distributions, we use total variation (TV) distance, which satisfies the triangle inequality. Suppose

$$\mathrm{TV}(p_d, p_h) \leq \varepsilon_1, \quad \mathrm{TV}(p_D, p_d) \leq \varepsilon_2, \tag{8}$$

where p_h is the empirical distribution of human annotations, p_d is the in-domain distribution, and p_D is the distribution induced by D. Then, by the triangle inequality,

$$\mathrm{TV}(p_D, p_h) \leq \varepsilon_1 + \varepsilon_2. \tag{9}$$

For any bounded function $g : [0, 1] \to \mathbb{R}$, it follows that

$$\big|\mathbb{E}_{p_D}[g] - \mathbb{E}_{p_h}[g]\big| \leq 2(\varepsilon_1 + \varepsilon_2), \tag{10}$$

by the standard bound $|\mathbb{E}_P[g] - \mathbb{E}_Q[g]| \leq 2\|g\|_\infty \operatorname{TV}(P, Q)$.

Following the same linear projection logic as before, define

$$\hat{f}''(X) = \mathbb{E}[S \mid X, D],$$
$$\mathbb{E}[(h - \hat{f}'')^2] = \left(1 - \rho_{hD}^2\right) \mathbb{E}[(h - \hat{f})^2], \quad (11)$$

where $\rho_{hD} = \operatorname{Corr}(h, D) > 0$ on the calibration set. This equality holds only under an optimal linear projection. Otherwise, it becomes an inequality.

E Prompts

E.1 Criteria Plan

Default for Fine-tuned Criteria Planner

Please provide the evaluation criteria for this task, including the weight of each criterion. The total score should be 10 points.

Task: {{task description}}

Default for Data Preparation

Task: {{task description}}

Instruction: Please provide the evaluation criteria for this task, including the weight of each criterion. The total score should be 10 points, with no more than 5 criteria in total. Present the information in the following format:

No. Criterion Name (Weight in points) - Description of what this criterion evaluates. Provide clear guidance on how this aspect of the response will be assessed.

An Example:

1. Efficiency (2 points): Is the generated code optimized in terms of time and space complexity?

- A float score near 0 (no) means the code is inefficient and has significant room for optimization.

- A float score near 1 (somewhat) means the code has a moderate level of efficiency but could be improved.

- A float score near 2 (yes) means the code is highly optimized in both time and space complexity.

Return the complete list. Note: Efficiency is included as an example and is not required to be part of the final list.

E.2 Machine Metric Refinement

Default for Fine-tuned Criteria Planner

Please provide a detailed metric description that clearly explains how the metric reflects and aligns with the corresponding criterion.

Criteria: {{criteria name}} - {{criteria description}}

Machine Metric: {{machine metric name}} - {{machine metric description}}

Default for Data Preparation

Instruction: First, generate the most suitable machine metric for the given criterion with metric description. Then, provide a detailed metric description that clearly explains how the metric reflects and aligns with the corresponding criterion.

An Example:

Criteria: Coherence – Measures how logically the summary flows, ensuring clarity and consistency in the ideas presented.

Machine Metric: BERTScore – Evaluates the semantic similarity between two pieces of text.

Detailed Machine Metric: BERTScore – Evaluates the semantic similarity between two pieces of text. A higher BERTScore reflects a greater degree of coherence, indicating that the summary aligns more closely with the logical flow and meaning of the original content.

Criteria: {{criteria name}} - {{criteria description}}

Machine Metric: {{machine metric name}} - {{machine metric description}}

E.3 Evaluation

Example of Standard Individual Evaluation

You will be given a sample, containing a generated code for given requirement.

Your task is to assign a float score to the response on one metric.

You should carefully horizontally compare the given samples in order to assign a suitable float score to each sample.

Please make sure you read and understand these instructions carefully. Please keep this document open while reviewing, and refer to it as needed.

Evaluation Criteria:

Overall (floating point numbers within the interval [1,5]): What is your overall impression of the quality of the generated code?

- A float score near 1 (very poor): The generated code is of very low quality. It contains significant

errors or does not run at all, lacks any meaningful structure, and does not meet the requirements in any substantial way. The code might be difficult or impossible to salvage for further use.

- A float score near 2 (poor): The code runs but is largely incorrect or ineffective. There are numerous logical errors or missing functionality, and it does not align well with the provided requirements. The code may also suffer from poor readability or lack of proper structure, making it difficult to understand or maintain.

- A float score near 3 (neutral): The code is functional but unremarkable. It may have some errors or areas for improvement but generally follows the basic requirements and runs with acceptable results. The code is neither highly readable nor efficient, but it's not overly difficult to understand or extend.

- A float score near 4 (good): The generated code is of good quality, meeting most of the requirements with only minor issues. It runs correctly for the majority of test cases and is fairly easy to read and maintain. The code could be improved, but any changes would be enhancements rather than necessary fixes.

- A float score near 5 (excellent): The code is of very high quality, demonstrating strong adherence to all requirements. It is free from significant errors, highly readable, well-structured, efficient, and maintainable. The code is clear, concise, and easy to understand, with well-considered logic and style. There are no significant flaws or areas for improvement.

Generated code and given requirement:
Source: {{requirement source}}
System Response: {{response output}}

Evaluation Form (scores ONLY):
- Overall:

Example of Batch Evaluation

You will be given a batch of 8 samples. Each sample contains a generated code for given requirement.

Your task is to assign a float score to the response on one metric.

You should carefully horizontally compare the given samples in order to assign a suitable float score to each sample.

Please make sure you read and understand these instructions carefully. Please keep this document open while reviewing, and refer to it as needed.

Evaluation Criteria:

Overall (floating point numbers within the interval [1,5]): What is your overall impression of the quality of the generated code?

- A float score near 1 (very poor): The generated code is of very low quality. It contains significant errors or does not run at all, lacks any meaningful structure, and does not meet the requirements in any substantial way. The code might be difficult or impossible to salvage for further use.

- A float score near 2 (poor): The code runs but is largely incorrect or ineffective. There are numerous logical errors or missing functionality, and it does not align well with the provided requirements. The code may also suffer from poor readability or lack of proper structure, making it difficult to understand or maintain.

- A float score near 3 (neutral): The code is functional but unremarkable. It may have some errors or areas for improvement but generally follows the basic requirements and runs with acceptable results. The code is neither highly readable nor efficient, but it's not overly difficult to understand or extend.

- A float score near 4 (good): The generated code is of good quality, meeting most of the requirements with only minor issues. It runs correctly for the majority of test cases and is fairly easy to read and maintain. The code could be improved, but any changes would be enhancements rather than necessary fixes.

- A float score near 5 (excellent): The code is of very high quality, demonstrating strong adherence to all requirements. It is free from significant errors, highly readable, well-structured, efficient, and maintainable. The code is clear, concise, and easy to understand, with well-considered logic and style. There are no significant flaws or areas for improvement.

Generated code and given requirement:
Source: {{requirement source}}
Sample 1:
System Response: {{sample 1 response output}}
Sample 2:
System Response: {{sample 2 response output}}
...
Sample 6:
System Response: {{sample 6 response output}}

Evaluation Form (Answer by starting with "Analysis:" to analyze the given samples regarding the *evaluation criteria and offer insights derived from the machine metric scores as concise as possible (Attention: Don't give your scores during this step). After analyzing all the samples, please give all the float scores in order following the template "Float Scores: [Sample1:score of Sample1, Sample2:score of Sample2, Sample3:score of Sample3, Sample4:score of Sample4, Sample5:score of Sample5, Sample6:score of Sample6]".*

Example of Co-Eval Evaluation

You will be given a batch of 6 samples. Each sample contains a generated code for given requirement.

Your task is to assign a float score to the response on one metric.

You should carefully horizontally compare the given samples in order to assign a suitable float score to each sample.

You can refer to the machine metric scores of each sample if you are not confidence.

Please make sure you read and understand these instructions carefully. Please keep this document open while reviewing, and refer to it as needed.

Evaluation Criteria:

Robustness (floating point numbers within the interval [0,2]): Does the generated code handle edge cases and potential errors gracefully?

- A float score near 0 (no) means the code fails to handle edge cases or crashes on invalid inputs.

- A float score near 1 (somewhat) means the code handles some edge cases but misses others or lacks comprehensive error handling.

- A float score near 2 (yes) means the code effectively handles all edge cases and includes comprehensive error handling.

Given Content and potentially useful Machine Metric Score:

Source: {{requirement source}}

Sonar Reliability - Assesses the robustness and fault-tolerance of software code, focusing on its potential to contain bugs or defects that could lead to malfunctions in production. The lower the numerical score, the better the reliability of the code, indicating fewer bugs and a lower risk of defects impacting the software's functionality.

Sample 1:
System Response: {{sample 1 response output}}
Score: {{sample 1 sonar reliability score}}
Sample 2:
System Response: {{sample 2 response output}}
Score: {{sample 2 sonar reliability score}}
...
Sample 6:
System Response: {{sample 6 response output}}
Score: {{sample 6 sonar reliability score}}

Evaluation Form (Answer by starting with "Analysis:" to analyze the given samples regarding the evaluation criteria and offer insights derived from the machine metric scores as concise as possible (Attention: Don't give your scores during this step). After analyzing all the samples, please give all the float scores in order following the template "Float Scores: [Sample1:score of Sample1, Sample2:score of Sample2, Sample3:score of Sample3, Sample4:score of Sample4, Sample5:score of Sample5, Sample6:score of Sample6]".

- Robustness:

F Additional Experiment Results

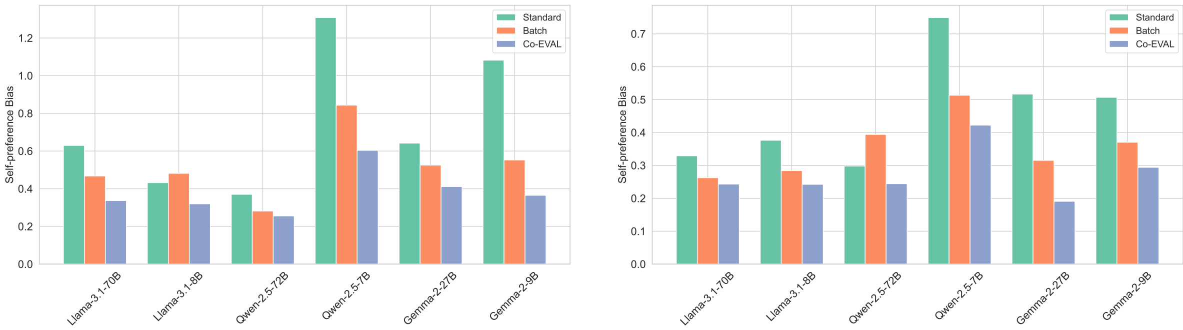
F.1 More Bias Elimination Effectiveness

We further demonstrate the effectiveness of the Co-Eval framework in mitigating bias on the FIQA and Health Counseling Conversations benchmarks, as shown in Figures 8, 9, 10, and 11.

F.2 Human Alignment on General Task

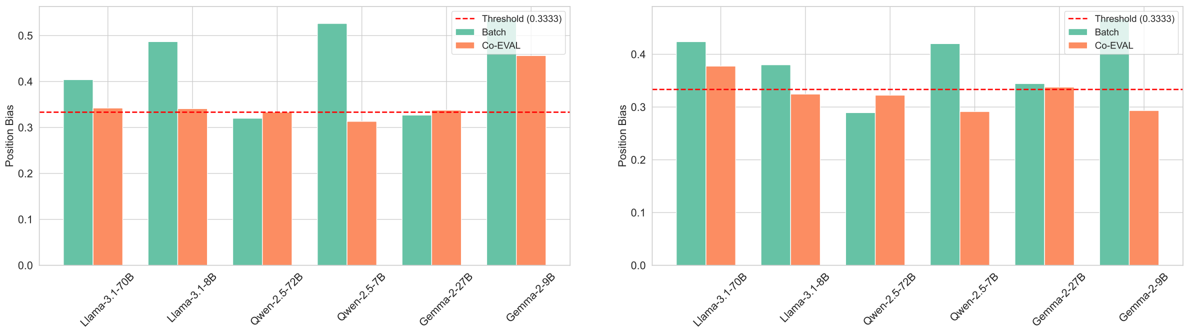
In our work with the Topical-Chat benchmark, we adhere to the original six evaluation criteria: understanding, naturalness, coherence, engagingness, groundedness, and overall quality. Since Topical-Chat is a multi-turn conversation benchmark, we follow previous studies (Liu et al., 2023; Yuan et al., 2023) and use turn-level correlations, assessing alignment between generated evaluations and human judgments by computing both Spearman (ρ) and Kendall (τ) correlations for each turn response, then averaging the scores to obtain the final evaluation. For the first five criteria, we adopt the descriptions provided by BATCHEVAL (Yuan et al., 2023) and select relevant metrics from the machine metric library. To evaluate overall quality, we implement the full Co-Eval pipeline. Additionally, in our analysis of G-Eval (Liu et al., 2023), we focus on the zero-shot evaluation capability of the LLM-based evaluator, conducting assessments without any pre-existing evaluation samples.

As shown in Table 3, our proposed Co-Eval framework demonstrates remarkable improvements



(a) FIQA (b) Health

Figure 8: Self-preference bias on FIQA and Mental Health Counseling Conversations benchmarks.



(a) FIQA (b) Health

Figure 9: Position bias on FIQA and Mental Health Counseling Conversations.

in Spearman and Kendall correlations across all three models and five original criteria. Even for GPT-4o, the use of suitable machine metrics improve groundedness assessment by up to 0.141 compared to BATCHEVAL, while the Co-Eval framework consistently surpasses baselines in overall quality evaluation. Similarly, on the Summeval, as shown in Table 4.

The results on the Summeval benchmark with fine-grained labels exhibit a trend similar to that of the Topical-Chat benchmark. While G-EVAL and BATCHEVAL outperform in certain criteria, our proposed Co-Eval framework consistently achieves the best performance on the "Overall" criteria.

G Detailed Case Study

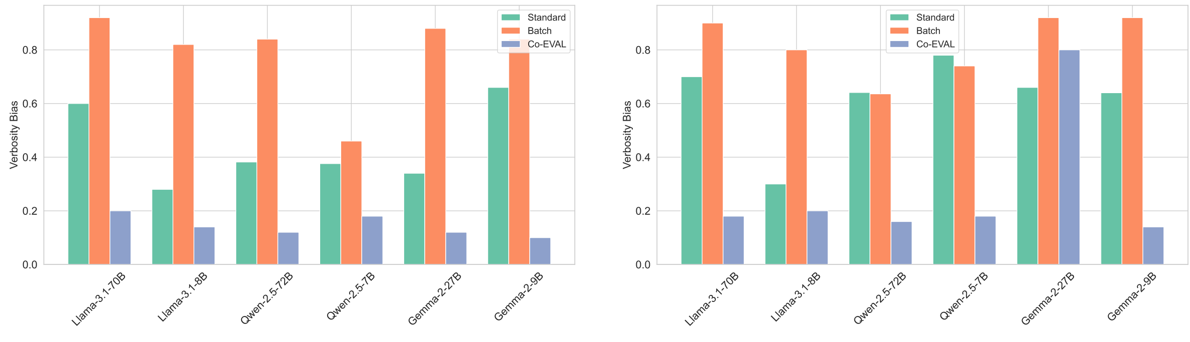
We further analyze the cases throughout the entire process:

Case 1: As shown in Figure 12, compared to the original LLaMA-3.1-8B-Instruct model, the fine-tuned planner provides more detailed criteria descriptions and assigns weights more aligned with human preferences. Simple errors, such as incorrect total scores, are also corrected. Additionally, the fine-tuned planner better captures subtle feature differences between tasks. For instance, it identifies "Structure" as essential criteria for "structured outline" task, but not for "summarization" task.

Case 2:

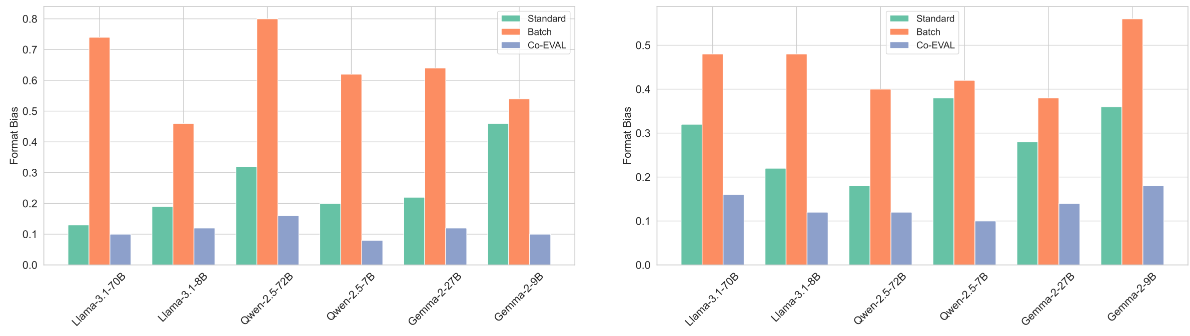
We present a subset of the machine metrics used in our experiments, as summarized in Table 5. These metrics span a range of tasks, including code generation, financial reasoning, mathematical problem solving, and empathetic dialogue, and include both semantic similarity measures and rule-based functional evaluations. Among them, functional metrics derived from rule-based tools, such as compilers and static analyzers, are particularly effective in mitigating hallucinations produced by LLM-based evaluators and in improving consistency both within a single LLM and across different LLMs.

Case 3: For some long-tail tasks, the generalization ability of the fine-tuned criteria planner is insufficient to generate a comprehensive set of evaluation criteria. For example, consider the task: Generate architectural drawings for a supermarket. The fine-tuned criteria planner accounts for the follow-



(a) FIQA (b) Health

Figure 10: Verbosity bias on FIQA and Mental Health Counseling Conversations.



(a) FIQA (b) Health

Figure 11: Format bias on FIQA and Mental Health Counseling Conversations.

ing aspects: Accuracy of Store Layout, Adherence to Building Codes and Regulations, Effective Use of Space, Aesthetic Appeal and Brand Identity, and Technical Quality and Presentation. However, all five criteria are equally weighted, each contributing 2 points to the total 10-point score. In contrast, human preferences suggest that Regulations and Store Layout should carry the most weight, making the evaluation misaligned with human judgment. Additionally, compared to the GPT-4o, budget considerations and branding alignment, both critical factors in supermarket architectural design, are missing from the criteria set. This gap further highlights the planner's limitations in capturing human-centric evaluation priorities.

Case 4: For some criteria descriptions, the machine metric with the highest semantic similarity score does not necessarily align best with human preferences. For example, in the Fluency criterion of the SummEval benchmark, perplexity is the machine metric whose description is most semantically similar to the criterion description. However, BARTScore exhibits a significantly higher Spearman correlation with human judgment. This misalignment leads to lower performance when Llama-3.1-70B-Instruct serves as the final prompt-based evaluator within the Co-Eval framework. The mistake arises despite regenerating machine metric descriptions via sampling to better reflect the specific aspects each metric evaluates. However, human evaluation does not always have clearly defined boundaries between different criteria—especially for closely related aspects. As a result, scores for Coherence can inadvertently influence the evaluation of Fluency, leading to discrepancies in alignment.

Case 5: For some general tasks, the machine metric score is less aligned with human preferences than the LLM itself. For example, as shown in the results in Table 3 and 4, LLM-based evaluation achieves the highest scores in some criteria using the batch method, even when the standard method is used without a machine metric. This is true even when the reference machine metric is suitable, particularly for criteria that are more subjective and dependent on the evaluator. In such cases, the machine metric may interfere with the prompt-based evaluator to some extent.

Metrics	Model	Understand		Natural		Coherence		Engaging		Grounded		Overall	
		ρ	τ	ρ	τ	ρ	τ	ρ	τ	ρ	τ	ρ	τ
Formula-based Evaluators													
BLEU-4	-	.033	.025	.130	.100	.277	.219	.386	.316	.446	.396	.280	.223
ROUGE-L	-	.052	.040	.132	.095	.206	.163	.321	.267	.461	.405	.249	.193
Embedding-based Evaluators													
BERTScore	-	.105	.080	.140	.101	.228	.184	.334	.275	.450	.395	.267	.213
BARTScore	-	.061	.039	.158	.124	.232	.188	.300	.237	.489	.422	.272	.215
Learning-based Evaluators													
USR	-	.322	.266	.346	.280	.354	.299	.392	.330	.551	.476	.438	.365
UNIEVAL	-	.467	.360	.513	.373	.612	.465	.608	.458	.574	.451	.662	.486
LLM-based Evaluators													
G-EVAL	GPT-4o	.679	.598	.618	.535	.570	.484	.707	.602	.726	.650	.692	.596
	Llama-3.1-70B	.472	.404	**.535**	**.443**	.515	.431	.615	.521	.628	.553	.650	.559
	Qwen-2.5-72B	.571	.486	.618	**.531**	.590	.505	**.744**	**.663**	.696	.621	.689	.592
BATCHEVAL	GPT-4o	.680	.591	.664	.562	.601	.514	.704	.607	.595	.525	.736	**.651**
	Llama-3.1-70B	.502	.433	.466	.391	.438	.376	.593	.499	.595	.522	.532	.450
	Qwen-2.5-72B	.500	.434	.488	.409	.455	.390	**.662**	.569	.530	.459	.551	.474
Co-Eval	GPT-4o	**.683**	**.594**	**.673**	**.579**	**.628**	**.547**	**.708**	**.607**	**.736**	**.656**	**.745**	.650
	Llama-3.1-70B	**.598**	**.508**	.530	.437	**.602**	**.512**	**.617**	**.522**	**.733**	**.646**	**.694**	**.593**
	Qwen-2.5-72B	**.594**	**.510**	**.622**	.523	**.616**	**.532**	.660	.572	**.722**	**.642**	**.698**	**.609**

Table 3: Turn-level Spearman (ρ) and Kendall (τ) correlations on Topical-Chat benchmark. The bold scores represent the highest score generated by each LLM as the final prompt-based evaluator, while the grey scores indicate the highest score across the entire column.

Case 6: The prompt-based evaluator demonstrates critical thinking when assessing the reference machine metric score. For example, "Upon reviewing the samples, it is evident that the machine metric scores do not directly reflect the readability of the code... However, analyzing the samples based on readability, we find that..." This capability strengthens the robustness of our proposed Co-Eval framework against unsuitable machine metric scores. However, it also introduces the possibility that the prompt-based evaluator may resist following the instructions of the augmented machine metric. As shown in the experiment on verbosity bias, an 8% extended response containing error information still achieved the highest score, even though the machine metric detected the error.

Case 7: Some LLMs, particularly smaller models, exhibit weak format-following capabilities. For example, when LLaMA-3.1-8B-Instruct is used as the final prompt-based evaluator, it may present scores in inconsistent formats such as: "Float Scores: Sample1: [3], Sample2: [2], Sample3: [3], Sample4: [4]" and "Float Scores: [4.5: Sample1, 2: Sample2, 4: Sample3, 4.5: Sample4]", whereas the expected standard format is: "Float Scores: [Sample1: 2.5, Sample2: 2.5, Sample3: 4, Sample4: 4]". These inconsistencies complicate score parsing and may lead to misinterpretations of evaluation results.

Case 8: Compared to the diversity of tasks, the coverage of machine metrics is limited. As a result, some criteria lack suitable machine metrics, such as the "Completeness" criteria in the MATH benchmark. Determining whether a solution step is both complete and reasonable remains an open question. In our experiment, we design a metric to evaluate completeness using the BERTScore between consecutive steps in a solution. A higher average score across all solution steps indicates a more complete and detailed response. Additionally, the Co-Eval framework makes it easy to incorporate new and useful machine metrics into the evaluation process, improving adaptability and coverage.

Metrics	Model	Coherence		Consistency		Fluency		Relevance		Overall	
		ρ	τ	ρ	τ	ρ	τ	ρ	τ	ρ	τ
G-EVAL	Llama-3.1-70B	.542	.454	**.550**	**.486**	.423	.366	.395	.338	.517	.423
	Qwen-2.5-72B	.509	.425	**.624**	**.563**	.529	.469	.413	.349	.474	.399
BATCHEVAL	Llama-3.1-70B	.444	.366	.547	.483	**.427**	**.372**	.421	.354	.510	.422
	Qwen-2.5-72B	**.514**	**.424**	.552	.497	.430	.373	.407	.343	.532	.448
Co-Eval	Llama-3.1-70B	**.548**	**.502**	.452	.413	.391	.355	**.464**	**.427**	**.525**	**.448**
	Qwen-2.5-72B	.483	.415	.592	.544	**.558**	**.511**	**.457**	**.391**	**.552**	**.465**

Table 4: Spearman (ρ) and Kendall (τ) correlations on Summeval benchmark.

Task	LLaMA-3.1-8B-Instruct	Fine-tuned Criteria Planner
Generate a summarization for a given content.	1. **Accuracy** (3 points): Does the summarization capture the essential information and key findings? 2. **Completeness** (2 points): Are all the relevant topics and material included in the summarization? 3. **Clarity** (2 points): Is the language simple and easy to understand? 4. **Organization** (2 points): Are the main ideas and supporting details presented in a logical order? 5. **Brevity** (1 points): Is the summary free from redundancy and unnecessary information?	1. **Accuracy** (4 points): Does the summary capture the main ideas and key details of the original content? 2. **Clarity** (2 points): Is the summary written in a clear, coherent, and easily understandable manner? 3. **Brevity** (2 points): Does the summary condense the content while maintaining all essential information? 4. **Relevance** (2 points): Does the summary focus only on the most important points and avoid unnecessary tangents?
Create a structured outline for a given content.	1. **Content Coverage** (2 points): Comprehensively cover the given content, addressing all key points and subtopics. 2. **Organization and Hierarchy** (2 points): Have a clear, logical structure, with suitable headings, subheadings, and bullet points. 3. **Relevance and Clarity** (2 points): Be easy to follow, with concise and non-ambiguous headings and subheadings. 4. **Mechanics and Consistency** (2 points): Follow a consistent structure, and use proper grammar and punctuation. 5. **Adaptability to Content Requirements** (4 points): Be adaptable to the specific requirements of the content.	1. **Structure** (4 points): Does the outline follow a clear, logical progression of ideas? 2. **Comprehensiveness** (3 points): Does the outline cover all the key points and important sections of the content? 3. **Clarity** (2 points): Is the outline clear and easy to understand, with appropriate labeling of sections and subsections? 4. **Brevity** (1 points): Is the outline concise, without including unnecessary details or extraneous information?

Figure 12: Case study for the fine-tuned criteria planner.

Metric Name	Description	Task Usage
COMET	Evaluates translation quality by comparing semantic similarity between generated texts using multilingual embeddings... Higher scores indicate ...	Flores
Grammarly	Assesses grammar, clarity, and fluency by detecting language issues... Higher scores reflect cleaner, more polished writing.	Flores
codeBERTScore	Measures semantic similarity of code using CodeBERT embeddings... Higher scores indicate better alignment with reference code.	CoNaLa
Cyclomatic Complexity	Calculates the number of independent paths in code... Higher scores suggest greater complexity and lower maintainability.	CoNaLa
Sonar Maintainability	Evaluates maintainability via code duplication, complexity, and smells... Lower scores indicate cleaner and easier-to-maintain code.	CoNaLa
Sonar Reliability	Identifies potential bugs and risky patterns... Higher scores signal more reliability issues.	CoNaLa
Python Compiler	Checks if Python code compiles correctly... A score of 1 means success; 0 indicates syntax errors.	CoNaLa
finBERTScore	Evaluates financial text similarity or sentiment using FinBERT embeddings... Higher scores indicate stronger semantic or sentiment alignment.	FIQA
Perplexity	Measures how well a language model predicts the text... Lower scores indicate higher fluency and coherence.	FIQA, Health
FactCC	Checks factual consistency between a statement and its context... Higher scores reflect greater factual accuracy.	FIQA
mathBERTScore	Evaluates relevance of mathematical expressions using MathBERT... Higher scores indicate stronger semantic similarity.	MATH
Completeness	Assesses coherence in multi-step reasoning by comparing step-wise similarity... Higher scores suggest more logically complete responses.	MATH
Calculator	Verifies arithmetic correctness... A score of 1 means exact match; 0 indicates a mismatch.	MATH
Sentiment Analysis	Evaluates emotional tone by estimating sentiment polarity... Higher scores imply more positive or emotionally aligned content.	Health
Empathy	Assesses empathetic expression via emotional and relational cues... Higher scores indicate stronger empathetic resonance.	Health

Table 5: Part of Descriptions of Evaluation Metrics