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

Anonymous ACL submission

Abstract

Large language models are increasingly used as evaluators in natural language generation tasks, offering scalability and interpretability advantages over traditional evaluation methods. However, current 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 introduce the Co-Eval framework, which employs a criteria planner model and optimized machine metric to improve scalability, fairness of LLM-based evaluation. Experimental results on both general and domain-specific tasks show that Co-Eval reduces biases across LLMs by up to 0.4903 in self-preference bias and improves alignment with human preferences by up to 0.324 in Spearman correlation.

1 Introduction

Evaluating natural language generation (NLG) quality is challenging, as these tasks often involve subjective judgments, and what constitutes high-quality output can vary depending on the specific context or audience. While human evaluation is a common method for assessing the quality of generated text, it is time-consuming. Recently, researchers (Liu et al., 2023; Chan et al., 2023; Zheng et al., 2023a) have started using large language models (LLMs) as evaluators, noting their impressive performance in aligning with human preferences when assessing generated text.

However, studies (Koo et al., 2023; Panickssery et al., 2024) have shown that LLMs exhibit certain biases, such as a preference for text generated by the models themselves, and factors like presentation order (Wang et al., 2023) and text length (Hu et al., 2024) can affect fairness as well. Moreover, general-purpose LLMs often fall short when it comes to evaluating natural language generation tasks within specific domains (Dorner et al., 2025).

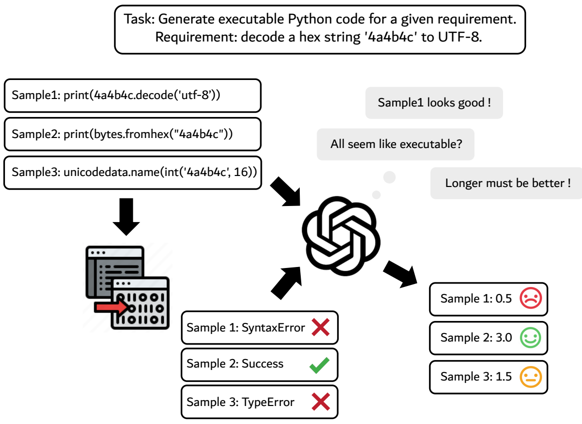


Figure 1: Machine metrics augment scalability and fairness of LLM-based evaluation.

Compared to LLM-based evaluators, machine metrics are more objective, providing precise assessments instead of the semantic evaluations typical of LLMs. Fine-tuned models can incorporate domain-specific knowledge, while rule-based metrics reflect human preferences embedded in rule design. For example, a compiler can definitively indicate if code runs, and BERTScore (Zhang et al., 2019) with CodeBERT can assess code similarity. Metrics like Cyclomatic Complexity (Watson et al., 1996) quantify code complexity by counting decision points. For fairer NLG evaluations and improved domain-specific LLM performance, machine metrics offer reliable benchmarks for consistent, human-aligned measurements.

In this paper, we introduce Co-Eval, a zero-shot reference-free LLM-based evaluation framework that enhances LLM-based evaluation through 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 are conducted across multiple tasks, including four general and four domain-specific tasks, demonstrating that Co-Eval framework enhances LLM-based evaluators, improving agreement with human preferences by up to 0.162 Spearman correlation in general generation tasks and up to 0.324 in domain-specific tasks, while reducing self-preference bias by up to 0.4903.

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

- 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 demonstrating 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.

- We conduct extensive experiments to demonstrate the effectiveness of the Co-Eval framework and, for the first time, explore LLM-based evaluation performance across domain-specific generation tasks.

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., 2024; 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 Sajjad, 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.

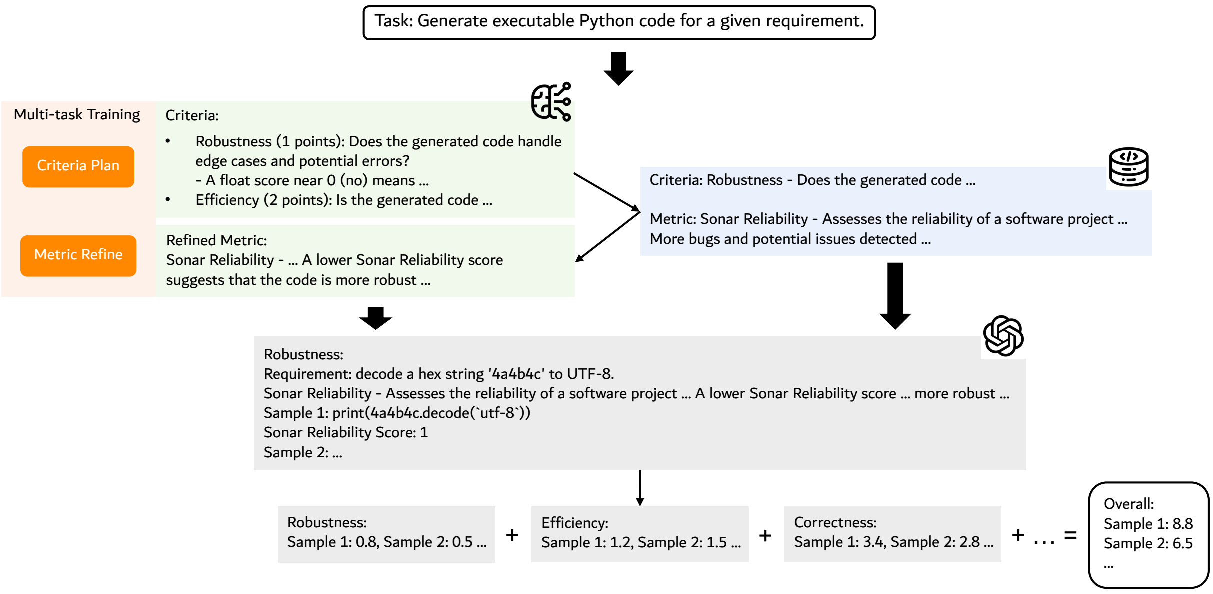


Figure 2: An overview of Co-Eval framework on executable Python code generation task. First, a fine-tuned criteria planner generates scoring criteria and corresponding weights for evaluating the task. Next, each criterion is matched with suitable machine metrics from a machine metric library based on 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.

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 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

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

data preparation is detailed in Appendix D.

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 for the machine metric library, which can be primarily divided into the following two categories:

Formula-based Metric relies on predefined rules and patterns to assess specific criteria in generated outputs, providing precise evaluations that LLMs may struggle to predict. For example, a syntax parser can accurately verify if generated code is syntactically correct and compilable, an assessment that may exceed the predictive capabilities of LLMs. Another key role of the formula-based metric is to guide the LLM-based evaluator toward aligning more closely with human preferences, which are often embedded within the metric's design. For instance, when evaluating text summarization, Information Density Formula can prioritize brevity and key information inclusion.

To theoretically validate our approach, we demonstrate the benefits of integrating Formula-based Metrics in the following proof:

Let $f(X)$ be the LLM-based evaluator's score based on sample X, and let $M(X)$ represent a formula-based metric score derived from X. Define $f(X, M(X))$ as the LLM-based evaluator's score that incorporates the formula-based metric score $M(X)$. Let $h(X)$ represent the human-assigned score. The error of the LLM-based evaluator relative to the human score is given by

$$\begin{aligned}\epsilon_f &= |h(X) - f(X)| \\ &= |h(X) - E_{s\sim p(s|X)}[s]|,\end{aligned} \quad (1)$$

where s denotes a potential scoring outcome, $p(s|X)$ is the probability distribution over scores s conditioned on the sample X, and $E_{s\sim p(s|X)}[s]$ represents the expected value of s under $p(s|X)$.

Similarly, the error of the LLM-based evaluator when incorporating the formula-based metric is given by

$$\begin{aligned}\epsilon_{f'} &= |h(X) - f(X, M)| \\ &= |h(X) - E_{s\sim p(s|X,M)}[s]|.\end{aligned} \quad (2)$$

According to Bayes' rule and the principle of maximum entropy, we have

$$p(s|X, M) \propto p(s|X)\cdot\exp(-\lambda(s-\beta M)^2), \quad (3)$$

where λ is a regularization parameter that controls the weight of the metric influence, and β is a scaling factor for the metric M.

For a distribution $p(s|X)$, $Var(p(s|X)$ quantifies how much scores s are expected to vary around their mean when conditioned on X alone. And by the properties of variance, we have

$$\begin{aligned}&Var(p(s|X, M)) = \\ &\frac{Var(p(s|X)) \cdot Var(\exp(-\lambda(s - \beta M(X))^2))}{Var(p(s|X)) + Var(\exp(-\lambda(s - \beta M)^2))} \\ &\qquad < Var(p(s|X)).\end{aligned} \quad (4)$$

This reduction implies that formula-based metric M can improve LLM-based evaluator to provide a more concentrated estimate around the target score.

Meanwhile, given that M is designed based on human-defined criteria, we assume $Corr(h, M) = \rho$, where $Corr$ represents the correlation between the human-assigned score $h(X)$ and the formula-based metric $M(X)$. We assume $\rho > 0$ implies that $h(X)$ and $M(X)$ are positively correlated, if and only if M is suitable for evaluating X according to the defined criteria. This positive correlation ensures that $\beta > 0$ and that the expected value of s under $p(s|X, M)$ is closer to $h(X)$. Consequently,

$$\begin{aligned}&|h(X) - E_{s\sim p(s|X,M)}[s]| \\ &\qquad < |h(X) - E_{s\sim p(s|X)}[s]|,\end{aligned} \quad (5)$$

which implies

$$\epsilon_{f'} < \epsilon_f. \quad (6)$$

Model-based Metric leverages well-trained deep neural network models to assess specific criteria for generated outputs. While LLMs are generally effective for broad generation tasks, we focus on smaller, domain-specific models trained on specialized corpora, which are typically more robust in their respective domains compared to general-purpose LLMs. For instance, a BERT model trained on a financial corpus may better capture financial context similarities. This type of model-based metric can augment an LLM-based evaluator's domain-specific knowledge.

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 1: 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.

We also provide a theoretical justification for the benefits of integrating Model-based Metrics:

Let $D(X)$ represent a model-based metric score derived from X. Assuming that the domain-specific corpus aligns well with human preferences, we have

$$KL(p_d||p_h) \leq \epsilon_1, \quad (7)$$

where $p_d(x)$ denotes the distribution of the domain-specific corpus, $p_h(x)$ denotes the distribution implied by human preferences, and KL is Kullback-Leibler divergence. Since D is trained on the domain-specific corpus, it is optimized to minimize $min_D E_{x \sim p_d}[L(D(x), h(x))]$. After sufficient training, we assume

$$KL(p_D||p_d) \leq \epsilon_2, \quad (8)$$

where p_D is the distribution implied by D's scores.

By applying the triangle inequality for KL divergence, we obtain

$$\begin{aligned} KL(p_D||p_h) &\leq KL(p_D||p_d) + KL(p_d||p_h) \\ &\leq \epsilon_2 + \epsilon_1 = \epsilon, \quad (9) \end{aligned}$$

implying $Corr(h, D) = \rho > 0$. Therefore, the error of the LLM-based evaluator when incorporating the model-based metric is given by

$$\begin{aligned} \epsilon_{f''} &= |h(X) - f(X, D)| \\ &= |h(X) - E_{s \sim p(s|X,D)}[s]| \\ &< |h(X) - E_{s \sim p(s|X)}[s]| = \epsilon_f. \quad (10) \end{aligned}$$

Since typical descriptions of machine metrics sometimes fail to accurately reflect evaluation criteria, we aim to improve their precision by identifying the specific data features that influence changes in metric scores. To achieve this, we provide GPT-4o with pairwise evaluation samples for each metric, enabling it to generate more precise descriptions that highlight the specific features each machine metric effectively captures within its context.

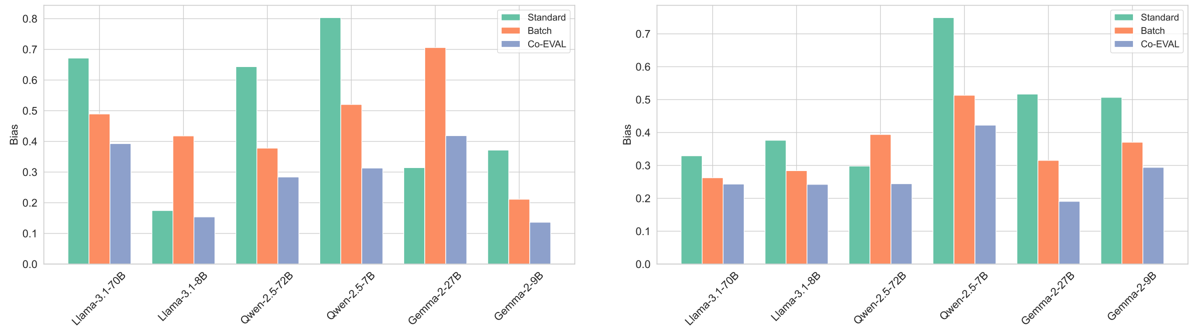
3.3 Prompt-based Evaluator

For the final LLM-based evaluator, we simply adopt the in-context learning and batchwise methods used in BATCHEVAL (Yuan et al., 2023), along with its input and output format. The prompt template is provided in the Appendix D.

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



(a) CoNaLa (b) Health Counseling

Figure 3: Self-preference bias on CoNaLa and Mental Health Counseling Conversations benchmarks.

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. Experimental results for the constrain are provided in Appendix E.3.

In the machine metric search, we select the top three metrics with embedding similarity scores exceeding 0.8, 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.

4.2 Agreement on Human Preference

For the Topical-Chat benchmark, as shown in Table 1, our proposed Co-Eval framework demonstrates remarkable improvements 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 and HANNA benchmarks, as shown in Table 3 and Figure 7, the Co-Eval framework, with its fine-tuned criteria planner and well-constructed machine metric library, achieves top correlations.

As shown in Table 2, Co-Eval outperforms standard and batch evaluation methods on both the CoNaLa and MATH benchmarks, achieving the highest correlations and even surpassing domain-specific evaluators and fine-tuned evaluation-enhanced models. Notably, on the CoNaLa benchmark, the LLaMA-3.1-70B-Instruct model under Co-Eval improves by up to 0.324 over standard methods.

Method	Model	CoNaLa ρ	CoNaLa τ	Model	MATH ρ	MATH τ
Standard	Prometheus-7B	.065	.063	Prometheus-7B	.113	.108
	Prometheus-8x7B	.256	.253	Prometheus-8x7B	.213	.211
	Llama-3.1-8B	.189	.194	Qwen-2.5-7B	.454	.415
	Llama-3.1-70B	.223	.205	Qwen-2.5-72B	.501	.470
Batch	Llama-3.1-8B	.322	.318	Qwen-2.5-7B	.397	.357
	CodeLlama-7B	.096	.109	Qwen-2.5-MATH-7B	.326	.302
	Llama-3.1-70B	.453	.419	Qwen-2.5-72B	.488	.466
	CodeLlama-70B	.259	.214	Qwen-2.5-MATH-72B	.391	.376
Co-Eval	Llama-3.1-8B	**.446**	**.420**	Qwen-2.5-7B	**.457**	**.423**
	Llama-3.1-70B	**.547**	**.492**	Qwen-2.5-72B	**.561**	**.535**

Table 2: Spearman (ρ) and Kendall (τ) correlations on CoNaLa and MATH benchmarks.

These results suggest that, whether for general or domain-specific generation tasks, the Co-Eval framework effectively aligns LLM-based evaluators with human preferences. This alignment is particularly beneficial in domain-specific tasks, where functional correctness is critical and general LLMs often struggle to assess accuracy reliably. In these cases, the Co-Eval framework can maximize evaluation effectiveness. In other words, Co-Eval framework can significantly **improve the scalability of LLM-based evaluation**.

4.3 Effectiveness on Bias Elimination

We demonstrate the effectiveness of the Co-Eval framework in eliminating three types of bias: self-preference bias, position bias, and verbosity bias.

Self-preference Bias We calculate the self-preference bias using the following equation:

$$Bias(i) = \frac{1}{N}\sum_{i=1}^{N} \max(0, R_o(i) - R_s(i)), \quad (11)$$

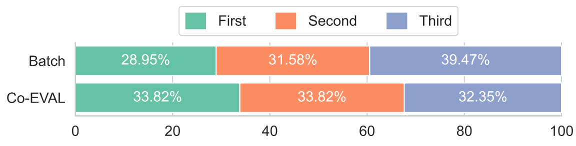


Figure 4: Top-ranking rate on MATH benchmark based on batch position.

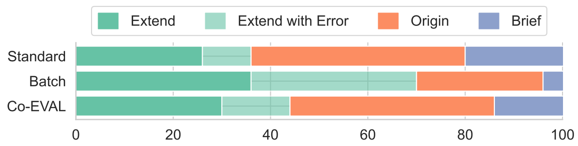


Figure 5: Top-ranking rate on FIQA benchmark based on verbosity degree.

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.

In the CoNaLa and Health Counseling benchmarks, as illustrated in Figure 3, the Co-Eval framework effectively reduces self-preference bias across all six LLM evaluators. Additionally, smaller LLMs exhibit greater shifts when aided by machine metric scores. The Qwen-2.5-72B-Instruct model achieves the most significant bias reduction compared to individual evaluation. Another notable observation is that certain models, such as Gemma-2-27B-Instruct and Qwen-2.5-72B-Instruct, show increased self-preference bias in batch evaluations. This suggests that while batch evaluation is an effective and straightforward method, it can sometimes amplify self-preference bias when an appropriate baseline is lacking.

Position Bias As shown in Figure 4, we observe that placing the same generated answer in the last position within a batch increases its likelihood of achieving the top rank. However, with the Co-Eval framework, the LLM-based evaluator achieves a more balanced ranking rate, allowing the same answer to attain the top rank consistently, regardless of its position within the batch.

Verbosity Bias As shown in Figure 5, we observe that compared to standard individual methods, LLM-based evaluators using the batch method exhibit a pronounced preference for more verbose answers, even when these answers contain some functional errors. The Co-Eval framework, however, enhances the evaluator's ability to detect functional errors in generated responses, enabling the LLM-based evaluator to achieve a more balanced ranking across answers of varying verbosity.

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 of LLM-based evaluation**.

Model	Llama-3.1-70B		Qwen-2.5-72B	
	ρ	τ	ρ	τ
Batch	0.510	0.422	0.532	0.448
Pure	0.465	0.384	0.502	0.413
+ Fine-tuned Planner	0.515	0.431	0.537	0.449
+ GPT-4o as Planner	0.517	0.437	0.543	0.452
+ Metric Library	0.477	0.401	0.521	0.428
+ Planner and Library	**0.525**	**0.448**	**0.552**	**0.465**

Table 3: Spearman (ρ) and Kendall (τ) correlations on Summeval benchmark.

4.4 Ablation Study

We conducted ablation studies on the two main components: the fine-tuned criteria planner and the machine metrics library.

Fine-tuning for Criteria Planner As shown in Table 3 between the LLM-based evaluator and human preferences, increasing the Spearman correlation by up to 0.05. Moreover, the fine-tuned planner retains over 90% of the performance of the GPT-4o planner. We further analyzed cases before and after fine-tuning the criteria planner. As shown in Figure 6, 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.

Machine Metric Library As shown in Table 3, machine metrics effectively improve the alignment of LLM-based evaluations with human preferences. We further examined the influence of different machine metrics on the Kendall correlation of LLM-based evaluations. As shown in Figure 7, randomly selected metrics sometimes negatively impact agreement, while the top 1 most similar metric generally has a positive effect. Moreover, machine metric scores from the fine-tuned model have an even greater positive impact compared to

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 6: Case study for the fine-tuned criteria planner.

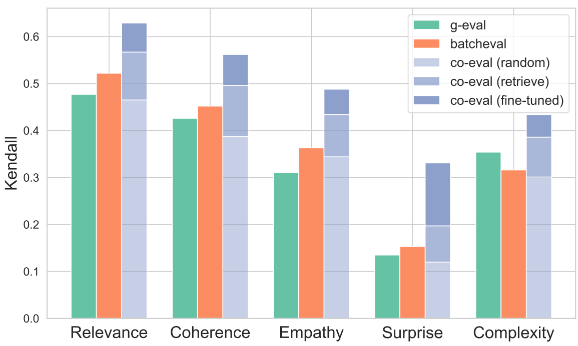


Figure 7: Kendall correlations on HANNA benchmark.

standard individual and batch methods. For criteria where LLM-based evaluations show the weakest performance, the appropriate machine metric and fine-tuned model scores achieve the most significant improvement compared to other criteria.

4.5 Error Analysis

Although we demonstrate the effectiveness of our proposed Co-Eval framework, some remaining errors in the process still need to be addressed:

Criteria planner sometimes fails. While using state-of-the-art models such as GPT-4o as a planner can be costly and inconsistent, fine-tuned smaller LLMs offer a more stable and cost-effective alternative while maintaining comparable performance. However, the generalization ability of fine-tuned smaller LLMs may not be sufficient, especially for long-tail tasks. Although we attempt to improve generalization by collecting data from real agent platforms, it is impossible to cover all real-world scenarios comprehensively. In such cases, using a state-of-the-art model is recommended.

Machine metric library sometimes fails. We rely on the semantic similarity to identify the most suitable machine metric. While we set a high threshold to ensure high precision and strive to make the machine metric descriptions as accurate as possible, semantic similarity does not always yield the best results. In some cases, the identified machine metric may be accurate but not more aligned with human preferences than the LLM itself, particularly for more general criteria. This can potentially misguide the evaluator.

Prompt-based evaluator sometimes fails. To counter occasional misguidance from the machine metric, we allow the final prompt-based evaluator to operate independently, without being strictly bound by these metrics. However, this approach also means that the evaluator may not always follow the instructions of the correct machine metric. Additionally, the limited format-following capability of some LLMs, particularly smaller models, can make parsing the final score more difficult.

A more detailed case study is presented in Appendix F.

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.

Cyril Chhun, Pierre Colombo, Chloé Clavel, and Fabian M Suchanek. 2022. Of human criteria and automatic metrics: A benchmark of the evaluation of story generation. *arXiv preprint arXiv:2208.11646*.

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*.

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*.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. 2022. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*.

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.

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*.

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.

Arthur Henry Watson, Dolores R Wallace, and Thomas J McCabe. 1996. *Structured testing: A testing methodology using the cyclomatic complexity metric*, volume 500. US Department of Commerce, Technology Administration, National Institute of ….

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

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.

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.

Prometheus (Kim et al., 2023, 2024) is a family of open-source language models designed specifically for evaluating other language models. Compared to the Prometheus 1 models, Prometheus 2 introduces support for switch modes by offering different input prompt formats and system prompts.

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.

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. Results are presented in Table 1.

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.

For the Flores benchmark, we examine the relationship between LLMs' familiarity with the target task and their preference bias. Six languages were selected for this study: French, Spanish, Chinese, Vietnamese, Ukrainian, and Thai. We used four LLMs: LLaMA-3.1-8B-Instruct, Qwen-2.5-7B-Instruct, Gemma-2-9B-Instruct, and GPT-4o-mini. Each model translated English text into these six languages. To measure each LLM's familiarity with the task, we followed previous work (Kadavath et al., 2022) that evaluates familiarity based on the self-consistency of LLMs in translation generation. Specifically, we selected ten samples, generated ten translations per sample with a temperature setting of 0.7, and computed the average token-level BLEU (Papineni et al., 2002) score across these translations. The results were ranked from 1 to 6, indicating each model's familiarity with the task, from most to least familiar. Results are presented in Figure 8.

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.

For the CoNaLa benchmark, we used six LLMs, including LLaMA-3.1-8B-Instruct, LLaMA-3.1-70B-Instruct, Qwen-2.5-7B-Instruct, Qwen-2.5-72B-Instruct, Gemma-2-9B-Instruct, and Gemma-2-27B-Instruct, to generate executable Python code based on specific requirements. The six responses were then randomly shuffled, and all six models served as LLM-based evaluators to examine their self-preference biases across three methods: the standard method, the batch method, and the Co-Eval framework. The results are displayed in Figure 3.

To further demonstrate that our proposed framework not only reduces bias but also aligns LLM-based evaluations with human preferences, we sampled the first 50 examples from the benchmark, manually scoring the code generated by LLaMA-3.1-8B-Instruct, Qwen-2.5-7B-Instruct, and Gemma-2-9B-Instruct. We invited three annotators. Each annotator with at least one year of Python coding experience was tasked with evaluating responses for correctness, readability, adherence to coding standards, and alignment with problem requirements. They were also encouraged to run the generated code to verify its functionality. The final human annotation score is calculated as the average of the scores provided by the three annotators. We then calculated the Spearman (ρ) and Kendall (τ) correlations between these models' scores and human preferences within the standard, batch, and Co-Eval frameworks, using the LLaMA-3.1-8B-Instruct and LLaMA-3.1-70B-Instruct models. Additionally, we applied domain-specific LLMs, CodeLLaMA-7B-Instruct and CodeLLaMA-70B-Instruct, using batch method. Results are shown in the Table 2.

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.

For the Health Counseling benchmark, similar to the CoNaLa benchmark, we used six LLMs as well, including LLaMA-3.1-8B-Instruct, LLaMA-3.1-70B-Instruct, Qwen-2.5-7B-Instruct, Qwen-2.5-72B-Instruct, Gemma-2-9B-Instruct, and Gemma-2-27B-Instruct, to generate responses to previous mental health dialogues. The six responses were then randomly shuffled, and all six models served as LLM-based evaluators to examine their self-preference biases across three methods: the standard method, the batch method, and the Co-Eval framework. The results are shown in Figure 3.

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.

For the MATH benchmark, we sampled the first 10 problems from each of the seven categories. Using LLaMA-3.1-8B-Instruct, Qwen-2.5-7B-Instruct, and Gemma-2-9B-Instruct, we generated answers for each question. We then organized the generated answers in three different orders, ensuring that each model's answer was evaluated in all positions in the batch. We used GPT-4o as an LLM-based evaluator to assess the generated answers across different orderings. We then calculated the rate at which each answer achieved the highest score at different positions, with results shown in Figure 4.

Similar to CoNaLa benchmark, we also manually scored 70 examples with answers generated by all three models. We invited three annotators as well. Each annotators who had completed at least one mathematics course was instructed to assess responses for accuracy, clarity, logical reasoning, and adherence to problem-solving approaches. The final human annotation score is calculated as the average of the scores provided by the three annotators. We then calculated the Spearman (ρ) and Kendall (τ) correlations between the models' scores and human preferences across the standard, batch, and Co-Eval frameworks, using Qwen-2.5-7B-Instruct and Qwen-2.5-72B-Instruct. Domain-specific LLMs, Qwen-2.5-MATH-7B-Instruct and Qwen-2.5-MATH-72B-Instruct, were also applied using the batch method. Results are presented in Table 2.

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.

For the FIQA benchmark, we sampled the first 50 examples and used LLaMA-3.1-8B-Instruct to generate answers for each question. GPT-4o was then used to create both a brief and an extended version of each answer. From the extended versions, we sampled 25 examples and manually introduced errors, such as adding incorrect information, reversing the meaning of some sentences, and making calculation mistakes. We then organized the brief, original, and extended versions, both with and without errors, into a single batch, shuffling the presentation order. Using GPT-4o as an LLM-based evaluator, we calculated the rate at which each version received the highest score across the standard, batch, and Co-Eval frameworks. The results are shown in Figure 5.

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.

For the Summeval benchmark, we conducted an ablation experiment for the two main components of the Co-Eval framework: the fine-tuned criteria planner and the machine metric library. We selected the first 6 generated responses from the initial 50 samples and evaluated the Spearman (ρ) and Kendall (τ) correlations of these samples against human preferences, using the LLaMA-3.1-70B-Instruct and Qwen-2.5-72B-Instruct models as LLM-based evaluators. The evaluation employed both the batch method and the Co-Eval framework across four configurations: (1) with a non-fine-tuned criteria planner and no machine metric, (2) with only a fine-tuned criteria planner and no machine metric, (3) with GPT-4o as criteria planner and no machine metric, (4) with a non-fine-tuned criteria planner and machine metric, and (5) with both a fine-tuned criteria planner and machine metric. Results are presented in Table 3, with the complete leaderboard for each criterion shown in Table 4.

C.8 HANNA

HANNA (Chhun et al., 2022) is a large-scale, annotated benchmark designed for evaluating story generation models. It includes human-written and model-generated narratives with detailed annotations for five key aspects: coherence, relevance, empathy, surprise, and engagement.

For the HANNA benchmark, we investigated the impact of machine metric alignment with human preferences on the agreement of LLM-based evaluators with human preferences. We skipped the criteria planning step, using the original criteria descriptions instead. For each criterion, we applied three types of machine metrics: (1) a randomly selected metric from the top 10 retrieved metrics in the machine metric library, (2) the top 1 metric retrieved from the machine metric library, and (3) BERTScore using our fine-tuned BERT model. We then used the LLaMA-3.1-70B-Instruct model as an LLM-based evaluator across the five key aspects using the standard, batch, and Co-Eval frameworks on the first five generated stories of the initial 30 samples. The Spearman (τ) correlations of the evaluation results against human preferences are shown in Figure 7.

In our training setup for the BERT model, we allocated 50 of the remaining 70 samples for training and 20 for validation. The Adam optimizer is used with a learning rate of 1e-5, and training runs for a maximum of 30 epochs. We employ a pairwise ranking loss on batches generated from the same prompt, with early stopping applied if the Kendall correlation does not improve on the validation set for 5 consecutive epochs.

D Prompts

D.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.

D.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}}

D.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 numer-

ous 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 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.

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}}

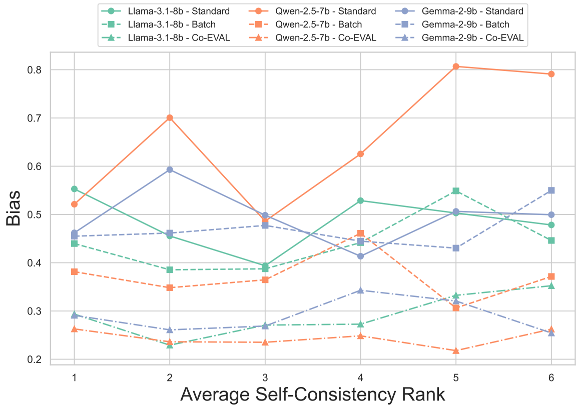


Figure 8: Self-preference bias on Flores benchmark.

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:

E Additional Experiment Results

E.1 Self-preference on Flores Benchmark

For the Flores benchmark, we attempt to explore the relationship between self-preference bias and LLMs' familiarity with different languages. Unfortunately, as shown in Figure 8, our results indicate that self-preference bias does not exhibit a clear correlation with language familiarity. This may be due to variations in language familiarity affecting the accuracy of self-preference bias calculations based on average rank. Nevertheless, regardless of the direction of these variations, batch evaluations help reduce self-preference across models and languages, with the Co-Eval framework further minimizing bias to near-uniform levels across languages.

E.2 Complete Summeval Leaderboard

We present complete experimental results on the Summeval benchmark, a meta-benchmark with fine-grained labels. The results are summarized

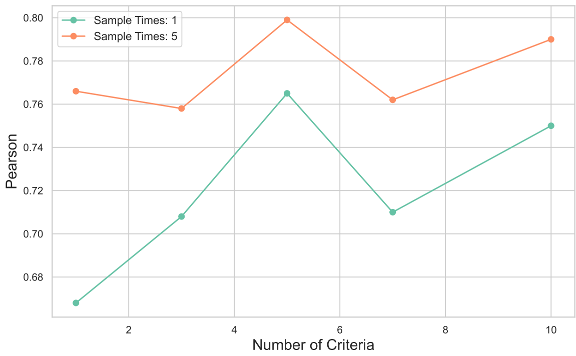


Figure 9: Pearson correlations on Topical-chat benchmark.

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.

E.3 Criteria Number and Sample Times

We evaluate the impact of the number of criteria and sample times on the Pearson correlations using the Topical-Chat benchmark in relation to our proposed Co-Eval framework. Specifically, we assess the performance of the Co-Eval framework with criteria numbers of 1, 3, 5, 7, and 10, and sample times of 1 and 5. As shown in Figure 9, the performance is more consistently aligned with human preferences when we sample 5 times and take the average score, compared to sampling only once, which is consistent with the findings reported in prior work (Yuan et al., 2023).

Regarding the number of criteria, the Pearson correlation shows an increasing trend from 1 (equivalent to the batch method) to 5. However, when the number of criteria exceeds 5, the Pearson correlation begins to decrease, indicating that 5 criteria is the most suitable choice for common generation tasks. Too few criteria fail to provide a comprehensive evaluation of the task, while too many criteria can lead to diminishing returns, potentially introducing redundant or conflicting evaluation metrics that compromise the accuracy and coherence of the overall assessment.

E.4 Impact of Temperature

We evaluate the impact of temperature on self-preference bias, position bias, and verbosity bias by testing temperatures of 0.0, 0.3, 0.5, 0.7, and 1.0, and reproducing the experiments for each type of bias.

As shown in Figure 10, while the effect of temperature on self-preference bias varies across models, our proposed Co-Eval framework consistently enables the LLM-based evaluator to achieve the lowest self-preference bias. Furthermore, for position bias and verbosity bias, GPT-4o, when used as an LLM-based evaluator with the Co-Eval framework, consistently maintains a balanced top-ranking rate while being less influenced by the position and verbosity of each response.

F Detailed Case Study

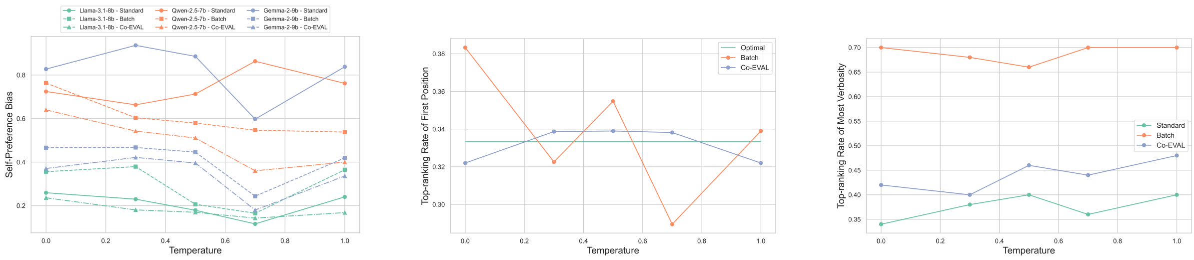
We further analyze the cases throughout the entire process:

Case 1: 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 following 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 2: 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

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



(a) Self-Preference Bias (b) Position Bias (c) Verbosity Bias

Figure 10: Impact of temperature on three kinds of bias.

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 3: 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 1 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.

Case 4: 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 5: 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 6: 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.