

LaaJMeter: A Framework for LaaJ Evaluation

Gal Amram, Eitan Farchi, Shmulik Froimovich, Raviv Gal, and Avi Ziv

IBM Research, Israel

Abstract

Large Language Models (LLMs) are increasingly used as evaluators in natural language processing tasks, a paradigm known as *LLM-as-a-Judge* (LaaJ). While effective in general domains, LaaJs pose significant challenges in domain-specific contexts, where annotated data is scarce and expert evaluation is costly. In such cases, meta-evaluation is often performed using metrics that have not been validated for the specific domain in which they are applied. As a result, it becomes difficult to determine which metrics effectively identify LaaJ quality, and further, what threshold indicates sufficient evaluator performance.

In this work, we introduce *LaaJMeter*, a simulation-based framework for controlled meta-evaluation of LaaJs. LaaJMeter enables engineers to generate synthetic data representing virtual models and judges, allowing systematic analysis of evaluation metrics under realistic conditions. This helps practitioners validate and refine LaaJs for specific evaluation tasks: they can test whether their metrics correctly distinguish between better and worse (virtual) LaaJs, and estimate appropriate thresholds for evaluator adequacy.

We demonstrate the utility of LaaJMeter in a code translation task involving a legacy programming language, showing how different metrics vary in sensitivity to evaluator quality. Our results highlight the limitations of common metrics and the importance of principled metric selection. LaaJMeter provides a scalable and extensible solution for assessing LaaJs in low-resource settings, contributing to the broader effort to ensure trustworthy and reproducible evaluation in NLP.

1 Introduction

Large Language Models (LLMs) have dramatically expanded the capabilities of Natural Language Processing (NLP), making a wide range of tasks such as translation, summarization, question answering, and code generation not only feasible but scalable and accessible. Their ability to generalize across domains and adapt to diverse linguistic inputs has enabled the deployment of NLP tools in settings where traditional rule-based or supervised approaches were previously impractical.

As NLP systems become more pervasive, the need to evaluate their outputs reliably and efficiently becomes increasingly critical. Importantly, evaluating an NLP tool, whether AI-based or not, is itself an NLP task. It involves interpreting textual outputs, assessing their correctness, relevance, or fluency, and making nuanced judgments that often depend on domain-specific knowledge. This evaluative layer is essential for validating system behavior, guiding model development, and ensuring deployment safety [14].

LLMs are now widely used as evaluators, a paradigm known as *LLM as a Judge* (LaaJ), where the model is tasked with scoring, ranking, or otherwise assessing the outputs of other systems [17]. This approach has proven effective and is increasingly adopted in both research and production environments [5]. LaaJs are valued primarily for their scalability: they can provide context-aware evaluations across large datasets without the need for manual annotation. Beyond standalone evaluation, LaaJs are also integrated into AI-agent architectures, where they assess the outputs of other components and influence downstream decisions. In such systems, the judgment of a LaaJ may directly affect the agent’s actions, making the reliability of these judgments critical [16].

However, the growing reliance on LaaJs introduces a critical question: who evaluates the evaluator? This challenge, known as meta-evaluation, concerns the assessment of the judgments produced by an LLM acting

as a judge [5]. A robust meta-evaluation framework must determine whether these judgments are reliable, consistent, and meaningful across diverse tasks and inputs. A tempting but flawed solution is to use another LLM for this purpose. Yet, doing so merely shifts the problem up a level, creating a recursive loop with no independent ground truth. Without an external, model-agnostic mechanism for evaluation, the validity of the entire process is called into question. Thus, principled and reproducible approaches to meta-evaluation must break this dependency cycle and avoid relying on LLMs themselves.

One attempt to perform meta-evaluation involves using text-similarity metrics such as BLEU, ROUGE, METEOR, and BERTScore to compare LaaJ outputs against reference judgments [13]. While these metrics offer a convenient and automated way to assess textual alignment, they fall short in capturing the nuanced, context-dependent nature of evaluative tasks. For instance, BLEU and ROUGE rely heavily on n-gram overlap, which may penalize valid paraphrases or reward superficial similarity. BERTScore improves upon this by leveraging contextual embeddings, yet it still lacks the ability to assess reasoning quality, factual correctness, or domain-specific relevance. These limitations are particularly pronounced in tasks where multiple valid judgments exist or where subtle distinctions matter. For example, an instruction to write code in a programming language can yield multiple correct solutions, none of which may be textually similar. Therefore, for reliable meta-evaluation, comparison against expert human annotations remains the gold standard [14]. In practice, metrics such as agreement with expert rankings (e.g., via Kendall- τ or Spearman- ρ) and accuracy against gold-standard labels are commonly used to benchmark LaaJ performance against human judgment. Moreover, off-the-shelf benchmarks exist for many common tasks such as summarization (e.g., CNN/DailyMail), question answering (e.g., SQuAD), and code generation (e.g., HumanEval), providing standardized datasets and annotations for principled evaluation [1, 4].

While off-the-shelf benchmarks provide robust infrastructure for evaluating LaaJs on general tasks such as summarization, question answering, and code generation, they are often ill-suited for domain-specific variants of these tasks. For example, generating code in a legacy programming language, summarizing legal documents, or answering questions in a specialized scientific field may require domain expertise and context that standard benchmarks do not capture [11, 8]. These specialized tasks frequently suffer from data scarcity and limited access to expert annotations, due to the rarity and cost of qualified evaluators. This challenge is especially pronounced in low-budget projects, where the resources needed to curate domain-specific datasets or involve experts are not readily available. As a result, performing principled meta-evaluation in these contexts remains a significant hurdle, demanding creative solutions that can compensate for the lack of standardized resources.

In the absence of standardized benchmarks and expert annotations, meta-evaluation in domain-specific contexts often relies on creative, task-specific strategies. One illustrative approach is to use outputs of known or controlled quality, such as responses from models with varying capabilities, and verify that the LaaJ assigns higher scores to the better outputs. However, due to the specificity of the domain, there is often little to no literature guiding which metrics are effective or which experimental setups yield meaningful insights for meta-evaluation [7]. Even when a metric appears to correlate with human judgment (though this correlation is rarely verifiable), determining a threshold that signifies sufficient evaluator quality remains inherently difficult. Without clear standards, practitioners are left to define ad hoc criteria, which may vary widely across applications and compromise reproducibility.

To address this challenge, we propose *LaaJMeter*, a simulation-based framework for controlled meta-evaluation of LaaJs. LaaJMeter generates synthetic data representing virtual models and virtual LaaJs, enabling systematic analysis of evaluation metrics under realistic conditions. By simulating models of varying quality and LaaJs with different noise and bias profiles, we can test the sensitivity and robustness of metrics without relying on human annotation [10].

We demonstrate the utility of LaaJMeter through a use case in code translation, where an LLM translates legacy code into a modern programming language. In this context, we simulate virtual models and LaaJs, and evaluate several metrics: t -test, Kendall’s τ correlation, and an ordering experiment. Our results reveal that while some metrics (e.g., t -test) are insufficiently sensitive to LaaJ quality, others (e.g., τ -correlation) provide robust signals even when model distance estimates are imprecise. The ordering experiment, though effective, requires careful interpretation of model distance to ensure meaningful results.

In summary, LaaJMeter offers a principled and extensible framework for evaluating LLMs as judges, particularly in settings where annotated data is scarce or unreliable. It complements existing evaluation paradigms and provides practical guidance for metric selection, thresholding, and simulation-based benchmarking. Our work contributes to the growing body of research on LLM-based evaluation and opens new directions for scalable, data-efficient assessment of model judgment quality.

2 Related Work

Several prior works collectively underscore the importance of principled, adaptable meta-evaluation strategies for LaaJs, when expert data is limited or unavailable.

The paradigm of *LLM-as-a-Judge* (LaaJ) has gained traction as a scalable alternative to human evaluation, particularly in domains where expert annotations are scarce. While general-purpose benchmarks such as AlpacaEval and Chatbot Arena have been widely adopted, they often fail to capture the nuances of domain-specific tasks. The work [12] addresses this limitation by introducing a data pipeline for constructing domain-specific evaluation sets tailored for LaaJ frameworks. Their method combines manual curation, semi-supervised clustering, and stratified sampling to ensure balanced representation across domains such as law, medicine, and multilingual contexts.

In the legal domain, the work [3] demonstrates how LLMs can be leveraged for term extraction under limited annotated data conditions. Their study compares prompt-engineering and fine-tuning strategies across several models, showing that zero-shot and one-shot capabilities can significantly reduce reliance on expert-labeled data.

The work [2] explores the use of open-source LLMs for text annotation in political science, highlighting the effectiveness of fine-tuning in low-resource settings. Their findings suggest that modest quantities of annotated data can yield competitive performance, offering a practical path for domain-specific evaluation.

From a broader perspective, the work [9] provides a systematic survey of LLM evaluation methodologies, identifying key challenges in reproducibility and reliability. Their recommendations emphasize the need for standardized protocols, especially in specialized domains where evaluation setups vary widely.

Finally, the work [15] proposes a Bayesian evaluation framework to assess LLMs’ capabilities as functional approximators. Their work offers insights into how prior domain knowledge influences model behavior, which is particularly relevant for designing evaluators in domain-specific contexts.

3 Problem Description

Domain-specific NLP tasks, particularly in low-budget projects, often suffer from a lack of high-quality human-labeled data. In such scenarios, evaluating the performance of a LaaJ becomes challenging due to the limited availability of reliable ground truth.

In the absence of a comprehensive expert evaluation, one way to assess LaaJ quality is by applying it to outputs of known, varying quality. The evaluated LaaJ is expected to correctly distinguish between different outputs to the same input, by assigning higher scores to better outputs. This form of meta-evaluation enables comparison between different LaaJs based on their ability to rank outputs in accordance with their true quality. See [6] for an example of such an evaluation. However, even when relative comparisons are possible, it remains difficult to define a satisfactory threshold for what constitutes acceptable LaaJ quality.

Another common scenario involves benchmarks annotated by a single expert. While such data can be used to compare LaaJs, for instance, by measuring their accuracy relative to the expert’s scores, it remains unclear what level of agreement should be expected from a high-quality LaaJ. This stands in contrast to cases where multiple expert annotations are available, allowing for the calibration of LaaJ performance against inter-annotator agreement among human graders. In the absence of such comparative data, it becomes difficult to establish meaningful thresholds for acceptable LaaJ quality.

The core problem, therefore, is that when meta-evaluation data is scarce, it becomes difficult to determine:

- Is a given metric suitable for evaluating LaaJ quality?

- What threshold should be considered acceptable for that metric?

4 LaaJMeter, a General Description

To address the problem described above, we propose the creation of synthetic data representing virtual models and virtual LaaJs. This data is designed to simulate the behavior of models and LaaJs on a specific task, enabling controlled meta-evaluation. The key design principle is to ensure that the generated data closely resembles a realistic distribution and the behaviors observed in the evaluated NLP task.

4.1 Virtual Points

We begin by defining a set of virtual points, which serve as inputs for the virtual models. These points are not instantiated with actual content; rather, we specify only the size of the set. A reasonable choice is to match the size of a typical benchmark used for the task under evaluation. The number of virtual points can influence the stability of metric calculations. For instance, larger sets tend to reduce the variance in the mean scores of evaluation metrics.

4.2 Virtual Models

Once the number of virtual points is determined, we define virtual models. Conceptually, a virtual model produces inferences for each virtual point, which are then assigned ground-truth scores. In practice, we do not generate actual inferences, but rather assign scores directly. Formally, a virtual model is defined as a function $M : \{0, \dots, n-1\} \rightarrow [a, b]$, where n is the number of virtual points and $[a, b]$ is the range of possible scores. A good design choice is to ensure that the score distribution reflects typical patterns observed in the evaluated task.

4.3 Additional Virtual Models

To simulate models of varying quality, we generate additional virtual models by systematically modifying the scores of an existing model. For example, given a model M , we can define a new model M' such that $M'(k) = M(k) + b_k$, where b_k is sampled from a distribution. If the distribution tends to be positive, M' represents a model of higher quality than M . Importantly, this approach maintains a pointwise relationship between models, ensuring that M' performs better than M in expectation across all points, mimicking the behavior of improved model versions.

4.4 Virtual LaaJs

With virtual models defined, we proceed to construct virtual LaaJs. Intuitively, a virtual LaaJ evaluates the inferences produced by virtual models. Formally, a virtual LaaJ L is a mapping $L(i, k) = s$, where i is the index of a virtual model, k is the index of a virtual point, and s is the score assigned by the LaaJ to the model’s inference on that point. The quality of a virtual LaaJ is determined by the expected deviation between its scores and the ground-truth scores of the virtual models. By generating LaaJs of varying quality, we can evaluate which metrics are effective in distinguishing between them.

5 Code Translation Use Case

We now describe a concrete instantiation of the technique outlined above, applied in the context of evaluating an LLM designed to perform code translation from a legacy programming language to a modern one.

5.1 Virtual Models

We began by constructing a virtual base model $M_0 : \{0, \dots, 99\} \rightarrow [0, 30]$. This model was created by assigning ground-truth scores to point indices, sampled from the score distribution of a LaaJ we developed for the code translation task. Accordingly, we chose the range $[0, 30]$ to match the scoring scale used by that LaaJ. The number of points, 100, is typical to the benchmarks we use for model evaluation.

Following the construction of M_0 , we generated additional virtual models M_1, \dots, M_{20} . Each model M_{i+1} was derived from M_i by modifying its scores as follows: for each point k , with probability p , we set $M_{i+1}(k) = \min\{M_i(k) + 1, 30\}$, incrementing the score unless it exceeds the maximum value of 30. With probability $1 - p$, we set $M_{i+1}(k) = \max\{M_i(k) - 1, 0\}$, decrementing the score unless it falls below 0. The value of p was chosen such that the expected mean score of M_{i+1} is expected to be 0.5 points higher than that of M_i , accounting for the number of scores at the boundaries (0 and 30).

We then constructed models with lower scores than M_0 , denoted M_{-1}, \dots, M_{-20} , using the same procedure in reverse. For each M_{i-1} , the value of p was selected so that its expected mean score is 0.5 points lower than that of M_i . In total, we created 41 virtual models: M_{-20}, \dots, M_{20} , with incrementally varying quality. For any pair M_i and M_j with $-20 \leq i < j \leq 20$, the mean score of M_j exceeds that of M_i by $\frac{j-i}{2}$. We refer to the value $j - i$ as the *distance* between M_i and M_j .

5.2 LaaJ Simulation

We now describe the construction of virtual LaaJs, informed by our experience evaluating LaaJs in the code modernization task. In practice, benchmarks often contain a mix of straightforward and challenging examples. A competent LaaJ typically performs well on simple cases, while its performance may vary on examples involving specific features, such as uncommon commands, that introduce evaluation difficulty. Some LaaJs may handle certain features effectively but struggle with others.

To simulate this behavior, we designated 20 out of the 100 virtual points as “simple points”, which all virtual LaaJs are expected to evaluate accurately, regardless of their quality. The remaining 80 points were divided into 10 disjoint sets of 8 points each, referred to as *featured sets*. We then constructed virtual LaaJs L_1, \dots, L_{10} as follows: for each LaaJ L_j , we randomly selected j featured sets on which it performs poorly. On these sets, L_j exhibits bias and high noise in its scoring; on all other points, it exhibits only low noise.

Formally, for a point k not in the selected featured sets for L_j , the score is computed as $L_j(i, k) = M_i(k) + l_{i,k}$, where $l_{i,k} \sim \mathcal{N}(0, 1)$ represents low noise. Recall that $M_i(k)$ is the ground-truth score for model M_i on point k .

If k belongs to one of the featured sets selected for L_j , we sample a fixed bias $b_S \sim \mathcal{N}(0, 2)$ for the entire set S , and compute the score as $L_j(i, k) = M_i(k) + b_S + h_{i,k}$, where $h_{i,k} \sim \mathcal{N}(0, 5)$ represents high noise. This setup allows us to simulate LaaJs with varying sensitivity to specific features, enabling robust evaluation of metric effectiveness. Note that for $i < j$, L_i outperforms L_j , as L_j introduces both bias and high noise over an additional $j - i$ featured sets, corresponding to $8 \times (j - i)$ more points, compared to L_i .

6 Code Translation Use Case: Metric Simulation and Findings

We present simulations for various metrics used in the meta-evaluation of LaaJs.

6.1 *t*-Test

We begin with a negative result, one that demonstrates the limitations of a particular metric for meta-evaluation. Specifically, we assess LaaJ quality using a *t*-test applied to LaaJ scores over model inferences of varying quality.

The idea is as follows: we select two models (or model versions), where one is known to be better than the other. We apply a LaaJ to evaluate the inferences of both models and compute the *p*-value for the resulting score vectors. The expectation is that a high-quality LaaJ will yield a statistically significant

result, indicating that the scores for the better model are higher. A poor LaaJ, however, may fail to detect this difference.

Importantly, the distance between the models also plays a critical role. If the distance is too large, even a poor LaaJ may detect the difference, leading to a significant p -value. To explore this, we simulated t -tests on virtual LaaJ scores across pairs of virtual models with varying distances. The resulting p -values are shown in Table 1.

Distance/LaaJ	L_1	L_2	L_3	L_4	L_5	L_6	L_7	L_8	L_9	L_{10}
1	0.07	0.08	0.11	0.19	0.20	0.20	0.21	0.25	0.24	0.29
2	0.00	0.00	0.02	0.04	0.04	0.06	0.05	0.09	0.09	0.13
3	0.00	0.00	0.00	0.01	0.01	0.01	0.02	0.01	0.02	0.05
4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01
5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
6	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
8	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 1: p -value results

6.1.1 t -Test Result Analysis and Conclusions

As shown in Table 1, all LaaJs yield statistically significant p -values for model distances greater than 2. This indicates that even a poor LaaJ, such as L_{10} , can detect differences between models, even when the distance is merely notable. Consequently, the t -test lacks sensitivity to LaaJ quality and fails to distinguish between LaaJs of varying performance. We conclude that the t -test is not suitable for meta-evaluation in this context.

6.2 Kendall- τ rank correlation

We now present a positive example, where our simulation supports the usefulness of a metric. The Kendall- τ rank correlation coefficient (also known as τ -correlation) measures the agreement between two ranked vectors, with values ranging from -1 (complete disagreement) to 1 (perfect agreement).

For meta-evaluation, we use τ -correlation as follows: we select two models (or model versions) and apply them to the same benchmark. Then, we apply a LaaJ to evaluate the inferences and compute the τ -correlation between the resulting score vectors. Since inference quality also depends on input complexity, we expect the ranking of scores to be preserved to some extent. A high-quality LaaJ should produce scores that maintain this ranking, resulting in a high τ -correlation. In contrast, a poor LaaJ may fail to detect issues in lower-quality model outputs, leading to lower correlation.

Moreover, we expect the difference in τ -correlation between good and poor LaaJs to be more pronounced when comparing models with small distances. In such cases, subtle differences in inference quality are harder to detect, and only better LaaJs are expected to capture them effectively.

The results of our simulation are presented in Table 2.

6.2.1 τ -Correlation Result Analysis and Conclusions

The results align with our expectations. Better LaaJs consistently yield higher τ -correlation scores, especially at smaller model distances. Notably, the metric is more sensitive to LaaJ quality than to model distance, making it particularly useful for meta-evaluation. This sensitivity allows practitioners to use τ -correlation effectively without requiring precise knowledge of model distance.

Distance/LaaJ	L_1	L_2	L_3	L_4	L_5	L_6	L_7	L_8	L_9	L_{10}
1	0.79	0.76	0.71	0.67	0.65	0.60	0.57	0.56	0.53	0.48
2	0.78	0.74	0.70	0.67	0.63	0.60	0.58	0.55	0.53	0.48
3	0.76	0.73	0.69	0.65	0.63	0.59	0.56	0.56	0.52	0.47
4	0.75	0.72	0.67	0.64	0.61	0.58	0.55	0.55	0.52	0.48
5	0.74	0.71	0.67	0.63	0.62	0.57	0.55	0.55	0.52	0.46
6	0.73	0.70	0.66	0.63	0.61	0.57	0.54	0.53	0.52	0.46
7	0.72	0.69	0.65	0.62	0.61	0.56	0.53	0.53	0.50	0.46
8	0.70	0.68	0.65	0.60	0.60	0.56	0.53	0.53	0.51	0.46
9	0.69	0.67	0.63	0.60	0.59	0.55	0.53	0.53	0.50	0.44
10	0.68	0.67	0.62	0.59	0.58	0.54	0.52	0.50	0.49	0.45

Table 2: Kendall- τ rank correlation coefficient results

Our findings suggest that a τ -correlation score of approximately 0.70 is sufficient to distinguish between high- and low-quality LaaJs, even when model distance is only roughly estimated. This insight supports the use of τ -correlation as a reliable metric and helps establish a practical threshold for LaaJ evaluation. We conclude that τ -correlation is an effective and robust metric for LaaJ meta-evaluation.

6.3 Ordering Experiment

In our final example, we demonstrate sensitivity to distance and thereby learn to apply a given metric appropriately. A test we use for meta-evaluation is the ability of a LaaJ to correctly identify the better model among two candidates. To this end, we select two models of known, comparable quality, apply them to the same benchmarks, and compute the percentage of points for which the LaaJ (weakly) prefers the better model.

It is important to note that the better model does not necessarily produce superior inferences for all points. However, we expect that a more accurate LaaJ will yield higher results in this test, as it is less noisy, makes fewer mistakes, and is thus more likely to be biased toward the better model. We conducted a simulation of the ordering experiment using virtual models at various distances. The results are presented in Table 3.

Distance/LaaJ	L_1	L_2	L_3	L_4	L_5	L_6	L_7	L_8	L_9	L_{10}
1	64.8%	64.6%	62.5%	62.4%	61.6%	61.1%	59.6%	59.5%	58.8%	57.2%
2	74.3%	71.5%	70.6%	68.4%	67.1%	67.3%	65.3%	64.5%	63.4%	61.1%
3	78.2%	77.0%	77.2%	74.7%	71.8%	70.9%	69.9%	68.3%	67.5%	63.9%
4	83.2%	80.4%	79.7%	77.5%	76.3%	74.1%	73.1%	72.0%	70.8%	67.5%
5	86.2%	84.6%	82.5%	81.1%	78.9%	76.7%	75.8%	73.9%	72.8%	69.8%
6	88.4%	86.9%	85.5%	83.5%	82.3%	79.4%	78.5%	76.9%	75.7%	71.8%
7	90.3%	88.9%	86.9%	86.0%	83.9%	81.7%	79.9%	78.6%	77.1%	74.4%
8	91.6%	90.2%	89.2%	86.8%	86.3%	83.6%	82.3%	81.5%	81.1%	76.0%
9	92.9%	91.3%	90.6%	88.4%	87.1%	85.1%	84.5%	82.0%	82.2%	77.6%
10	94.4%	92.7%	92.2%	89.8%	88.4%	86.9%	85.5%	84.2%	83.7%	79.9%

Table 3: Ordering experiment results

6.3.1 Ordering Experiment Result Analysis and Conclusions

Our results show that the ordering experiment is sensitive to LaaJ quality and is therefore a useful metric for LaaJ meta-evaluation. However, the metric is also sensitive to model distance, which complicates the

interpretation of results. A careful estimation of model distance is necessary to set a meaningful threshold for evaluation.

For example, a result of approximately 80% is achieved by L_3 at distance 4, and also by the noisier L_{10} at distance 10. This illustrates that without accounting for model distance, the metric may fail to distinguish between LaaJs of different quality.

In summary, the ordering experiment is an effective metric, but its application requires careful handling. Specifically, an understanding of the relative distance between models.

7 Conclusion

In this work, we addressed the challenge of evaluating Large Language Models as Judges (LaaJs) in domain-specific NLP tasks where high-quality annotated data is scarce. We introduced the LaaJMeter framework, which enables controlled meta-evaluation through the construction of synthetic data representing virtual models and LaaJs. This approach allows for a systematic analysis of evaluation metrics under realistic conditions.

Through a detailed use case in code translation, we demonstrated how LaaJMeter can simulate nuanced model and LaaJ behaviors. We evaluated several metrics, t -test, Kendall- τ correlation, and an ordering experiment, highlighting their strengths and limitations. Our findings show that while the t -test lacks sensitivity to LaaJ quality, Kendall- τ correlation provides a robust signal even with imprecise model distance estimates. The ordering experiment, although effective, requires careful calibration based on model distance.

In this paper, we focused on simulating evaluation metrics that do not rely on human-annotated data. However, our framework can be extended to incorporate such data. This extension would involve simulating expert scores. One straightforward approach is to define the ground-truth scores as human evaluations. Alternatively, noise can be added to the ground-truth scores to simulate variability in expert judgment. This enables the evaluation of human-referenced metrics, such as mean squared error. We leave the exploration of this direction for future work.

References

- [1] Confident AI. Squad — deepeval - the open-source llm evaluation framework, 2025. Accessed August 2025.
- [2] Meysam Alizadeh, Maël Kubli, Zeynab Samei, Shirin Dehghani, Mohammadmasiha Zahedivafa, Juan D. Bermeo, Maria Korobeynikova, and Fabrizio Gilardi. Open-source llms for text annotation: A practical guide for model setting and fine-tuning. *Journal of Computational Social Science*, 8(17), 2025.
- [3] Julien Breton, Mokhtar Mokhtar Billami, Max Chevalier, Ha Thanh Nguyen, Ken Satoh, Cassia Trojahn, and May Myo Zin. Leveraging llms for legal terms extraction with limited annotated data. *Artificial Intelligence and Law*, 2025.
- [4] Mark Chen, Jerry Tworek, Heewoo Jun, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.
- [5] Steffi Chern, Ethan Chern, Graham Neubig, and Pengfei Liu. Can large language models be trusted for evaluation? scalable meta-evaluation of llms as evaluators via agent debate. *arXiv preprint arXiv:2401.16788*, 2024.
- [6] Ora Nova Fandina, Eitan Farchi, Shmulik Froimovich, Rami Katan, Alice Podolsky, Orna Raz, and Avi Ziv. Automated validation of llm-based evaluators for software engineering artifacts, 2025.
- [7] Rishav Hada, Varun Gumma, Mohamed Ahmed, Kalika Bali, and Sunayana Sitaram. METAL: towards multilingual meta-evaluation. In Kevin Duh, Helena Gómez-Adorno, and Steven Bethard, editors,

Findings of the Association for Computational Linguistics: NAACL 2024, Mexico City, Mexico, June 16-21, 2024, pages 2280–2298. Association for Computational Linguistics, 2024.

- [8] Swayambhoo Jain, Ravi Raju, Bo Li, Jonathan Li, and Urmish Thakker. Constructing domain-specific evaluation sets for llm-as-a-judge. *arXiv preprint arXiv:2408.08808*, 2024.
- [9] Md Tahmid Rahman Laskar, Sawsan Alqahtani, M Saiful Bari, Mizanur Rahman, Mohammad Abdullah Matin Khan, Haidar Khan, Israt Jahan, Amran Bhuiyan, Chee Wei Tan, Md Rizwan Parvez, Enamul Hoque, Shafiq Joty, and Jimmy Huang. A systematic survey and critical review on evaluating large language models: Challenges, limitations, and recommendations. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 13785–13816. Association for Computational Linguistics, 2024.
- [10] Yuxuan Lu, Jing Huang, Yan Han, Bingsheng Yao, Sisong Bei, Jiri Gesi, Yaochen Xie, Zheshen Wang, Qi He, and Dakuo Wang. Prompting is not all you need! evaluating llm agent simulation methodologies with real-world online customer behavior data. *arXiv preprint arXiv:2503.20749*, 2025.
- [11] Joel Niklaus, Veton Matoshi, Pooja Rani, Andrea Galassi, Matthias Stürmer, and Ilias Chalkidis. Lex-treme: A multi-lingual and multi-task benchmark for the legal domain. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3016–3054. Association for Computational Linguistics, 2023.
- [12] Ravi Raju, Swayambhoo Jain, Bo Li, Jonathan Li, and Urmish Thakker. Constructing domain-specific evaluation sets for llm-as-a-judge. In *Proceedings of the 1st Workshop on Customizable NLP (CustomNLP4U)*, pages 167–181. Association for Computational Linguistics, 2024.
- [13] Hadeel Saadany and Constantin Orasan. Bleu, meteor, bertscore: Evaluation of metrics performance in assessing critical translation errors in sentiment-oriented text. In *Proceedings of TRITON (Translation and Interpreting Technology Online)*, pages 48–56, 2021.
- [14] Hendrik Schuff, Lindsey Vanderlyn, Heike Adel, and Ngoc Thang Vu. How to do human evaluation: A brief introduction to user studies in nlp. *Natural Language Engineering*, 29(5):1199–1222, 2023.
- [15] Shoaib Ahmed Siddiqui, Yanzhi Chen, Juyeon Heo, Menglin Xia, and Adrian Weller. On evaluating llms’ capabilities as functional approximators: A bayesian evaluation framework. In *Proceedings of the 31st International Conference on Computational Linguistics (COLING)*, pages 5826–5835. Association for Computational Linguistics, 2025.
- [16] Huilin Wang and Lei Yu. Can we employ llm to meta-evaluate llm-based evaluators? a preliminary study. In De-Shuang Huang, Bo Li, Haiming Chen, and Chuanlei Zhang, editors, *Advanced Intelligent Computing Technology and Applications*, pages 161–172, Singapore, 2025. Springer Nature Singapore.
- [17] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.