EvalAssist: LLM-as-a-Judge Simplified

Michael Desmond, Zahra Ashktorab, Werner Geyer, Elizabeth M. Daly,
Martín Santillán Cooper, Qian Pan, Rahul Nair, Nico Wagner, Tejaswini Pedapati

IBM Research

Abstract

We present *EvalAssist*, a framework that simplifies the LLM-as-a-judge workflow. The system provides an online criteria development environment, where users can interactively build, test, and share custom evaluation criteria in a structured and portable format. A library of LLM based evaluators is made available that incorporates various algorithmic innovations such as token-probability based judgement, positional bias checking, and certainty estimation that help to engender trust in the evaluation process. We have computed extensive benchmarks and also deployed the system internally in our organization with several hundreds of users.

Introduction

Human evaluation of Large Language Models (LLMs) is common practice and still considered gold standard. However, given cost and time constraints, LLMs are increasingly being used to judge the output of other LLMs. LLM-as-a-judge is attractive as it can accommodate use case specific needs through custom criteria, is easy-to-understand by non-technical users, does not require reference data, and can significantly reduce human evaluation effort. Empirical studies have reported high agreement between LLM and human ratings. For example (Zheng et al. 2023) report more than 80% agreement and (Kim et al. 2023) report that fine tuned evaluator models have high correlation with human judgments. More recently, evaluator ensembles have been shown to be effective (Verga et al. 2024).

While LLM-as-a-judge has become popular, several challenges remain to make the approach effective, trustworthy and aligned with user needs. (Doddapaneni et al. 2024) report on evaluator LLMs that failed to identify synthetic quality drops in half the cases, suggesting that evaluators did not understand the task. (Bavaresco et al. 2024) urge caution after empirical analysis of several language tasks and conclude that LLMs are not yet ready to systematically replace human judges. Lastly, issues with bias (Liu et al. 2023; Li et al. 2023) and prompt sensitivity (Wang et al. 2023) have been identified that prohibit naive application of LLM-as-a-judge. These issues need to be addressed for practical application.

In this paper, we introduce `EvalAssist`[1], a comprehensive LLM-as-a-judge solution that provides a criteria development and test environment, and an intuitive Python toolkit focused on robustness and scale-ability. Users can iteratively test and refine evaluation criteria until they are confident that it works well and aligns with expectations. The criteria can then be applied to a larger dataset using the toolkit. Algorithmic innovations such as token log-probability option selection, positional bias checking, and certainty estimation helps to engender trust in the evaluation process. We have extensively tested our approach through various benchmarks, and we plan to open source EvalAssist as part of the Unitxt framework (Bandel et al. 2024).

EvalAssist

User Experience

`EvalAssist` simplifies the process of developing, refining and applying LLM-as-a-judge. Users work with three core constructs when developing an evaluation: task context, response(s) to evaluate, and evaluation criteria.

Task context, Fig. 1, represented as a dictionary of key-value pairs, is used to incorporate information that is pertinent to the evaluation task. For example when evaluating the quality of an answer in a Q&A task, the task context may include the question and source document, allowing the LLM evaluator to take this information into account.

Figure 1: An example task context that contains a question.

The responses to evaluate, Fig. 2, are simply text elements that are subject to evaluation. This is generally the output of an LLM, but could come from any source. When developing an evaluation users can test with multiple responses to increase coverage.

Finally, the evaluation criteria, Fig. 3, describes the dimension that the LLM considers when performing the evaluation. `EvalAssist` supports both direct assessment based on a rubric, and pairwise comparison. Direct assessment criteria includes a criteria description (often a specific question

[1]Demonstration link: https://youtu.be/Fb70IJ-vcx0

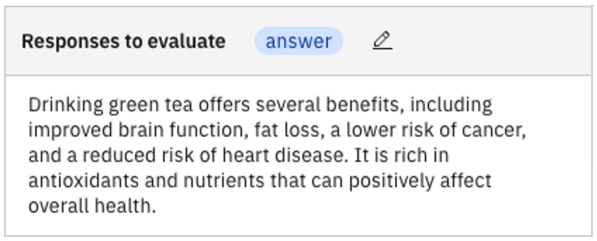

Figure 2: A response to evaluate. When developing criteria users can address the response directly, or rename the response using an alias. In this example the response has been renamed to "answer", to better reflect the semantics of the evaluation.

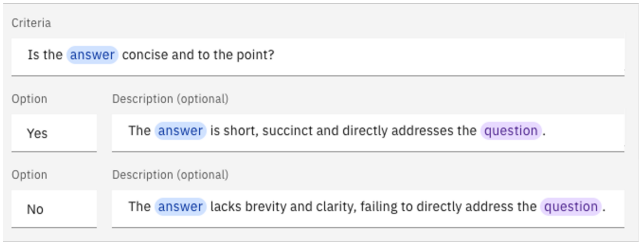

Figure 3: A direct assessment criteria that references a value from the task context (question) and the response via the answer alias. EvalAssist applies syntax highlighting to help the user to accurately reference information in the criteria.

or assertion) and a set of options, each with their own descriptive qualifier. Pairwise evaluation includes a description only, that is used by the evaluator when choosing between a pair of responses.

`EvalAssist` criteria are represented in a standard JSON format for portability. Once a criteria developer is satisfied with their criterion, they can export the JSON representation from the developer interface and use it in the Python toolkit for bulk evaluations.

Evaluators

`EvalAssist` provides a library of evaluators based on Granite (Granite 2024), Llama2 (Touvron et al. 2023), Mixtral (Jiang et al. 2024), Prometheus2 (Kim et al. 2024), and GPT4 (Achiam et al. 2023).

Evaluation is implemented as a chained prompting process, see Fig. 4. **Make Assessment**, this step is loosely inspired by Chain-of-Thought prompting (Wei et al. 2022), and involves prompting the evaluator LLM to produce an assessment of the response subject to the task context and evaluation criteria. **Generate Explanation**, the assessment is summarized to create a user-facing explanation of the evaluation rationale. **Choose Judgement**, the final step is option selection using log probabilities which is chained with the original evaluation prompt and the generated assessment. In the direct assessment modality, each of the available rubric options are presented to the evaluator as the selected option, and the token probabilities of that option are computed. The relative first token probability of each option is then compared, and the most probable option is the final judgement. The same process is applied for pairwise evaluation, except that the options map to the first and second response.

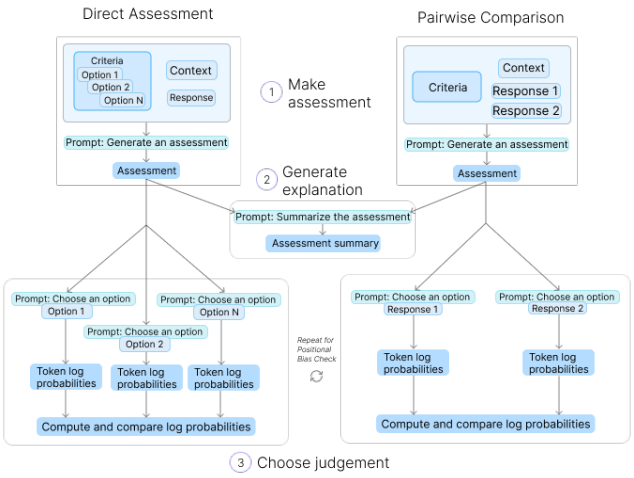

Figure 4: EvalAssist prompting workflow.

Using token log probability to extract a final judgement is more expensive than direct generation, but significantly improves robustness and makes it possible to approximate certainty. Instead of relying on the LLM to generate an option, a process prone to hallucination and formatting errors, the answer is extracted from the set of predefined completions. `EvalAssist` also supports positional bias checking and certainty estimation. Positional bias checking is implemented by shuffling the order in which options are presented to the evaluator LLM and checking for output consistency.

Benchmarks

We ran EvalAssist evaluators on several datasets and benchmarks (Bavaresco et al. 2024), where human annotations are available. We found no clear choice for an evaluator that outperforms on all tasks, suggesting that users need to experiment on which evaluator to use. LLM evaluators perform better at simpler criteria than those requiring more nuanced understanding. For more complex criteria, LLM evaluators tend to be pessimistic in their scoring relative to humans. In these cases, agreement between humans also tends to be low. Finally, evaluators generally perform better at pairwise assessments compared to direct assessments.

User Evaluation

`EvalAssist` is deployed internally and has had over 700 users so far. Our user research (Ashktorab et al. 2024) has revealed that direct assessment evaluation is preferred when users are seeking greater control, while Pairwise assessment is favored when working with more subjective tasks. We also discovered that users tended to over-fit their evaluation criteria when working with a single task context and a small set of responses. We plan to address this phenomenon by allowing users to work with a wider set of data, while maintaining the ability to quickly iterate and refine their criteria.

References

Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Ashktorab, Z.; Pan, Q.; Desmond, M.; Johnson, J. M.; Cooper, M. S.; Daly, E. M.; Nair, R.; Pedapati, T.; Achintalwar, S.; and Geyer, W. 2024. Aligning Human and LLM Judgments: Insights from EvalAssist on Task-Specific Evaluations and AI-assisted Assessment Strategy Preferences. https://arxiv.org/abs/2410.00873. Under review.

Bandel, E.; Perlitz, Y.; Venezian, E.; Friedman-Melamed, R.; Arviv, O.; Orbach, M.; Don-Yehyia, S.; Sheinwald, D.; Gera, A.; Choshen, L.; et al. 2024. Unitxt: Flexible, shareable and reusable data preparation and evaluation for generative ai. *arXiv preprint arXiv:2401.14019*.

Bavaresco, A.; Bernardi, R.; Bertolazzi, L.; Elliott, D.; Fernández, R.; Gatt, A.; Ghaleb, E.; Giulianelli, M.; Hanna, M.; Koller, A.; et al. 2024. Llms instead of human judges? a large scale empirical study across 20 nlp evaluation tasks. *arXiv preprint arXiv:2406.18403*.

Doddapaneni, S.; Khan, M. S. U. R.; Verma, S.; and Khapra, M. M. 2024. Finding Blind Spots in Evaluator LLMs with Interpretable Checklists. *arXiv preprint arXiv:2406.13439*.

Granite, I. 2024. Granite 3.0 Language Models.

Jiang, A. Q.; Sablayrolles, A.; Roux, A.; Mensch, A.; Savary, B.; Bamford, C.; Chaplot, D. S.; Casas, D. d. l.; Hanna, E. B.; Bressand, F.; et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Kim, S.; Shin, J.; Cho, Y.; Jang, J.; Longpre, S.; Lee, H.; Yun, S.; Shin, S.; Kim, S.; Thorne, J.; et al. 2023. Prometheus: Inducing fine-grained evaluation capability in language models. *arXiv preprint arXiv:2310.08491*.

Kim, S.; Suk, J.; Longpre, S.; Lin, B. Y.; Shin, J.; Welleck, S.; Neubig, G.; Lee, M.; Lee, K.; and Seo, M. 2024. Prometheus 2: An open source language model specialized in evaluating other language models. *arXiv preprint arXiv:2405.01535*.

Li, X.; Zhang, T.; Dubois, Y.; Taori, R.; Gulrajani, I.; Guestrin, C.; Liang, P.; and Hashimoto, T. B. 2023. Alpacaeval: An automatic evaluator of instruction-following models.

Liu, Y.; Iter, D.; Xu, Y.; Wang, S.; Xu, R.; and Zhu, C. 2023. G-Eval: NLG Evaluation using GPT-4 with Better Human Alignment, May 2023. *arXiv preprint arXiv:2303.16634*.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Verga, P.; Hofstatter, S.; Althammer, S.; Su, Y.; Piktus, A.; Arkhangorodsky, A.; Xu, M.; White, N.; and Lewis, P. 2024. Replacing Judges with Juries: Evaluating LLM Generations with a Panel of Diverse Models. *arXiv preprint arXiv:2404.18796*.

Wang, J.; Liang, Y.; Meng, F.; Shi, H.; Li, Z.; Xu, J.; Qu, J.; and Zhou, J. 2023. Is chatgpt a good nlg evaluator? a preliminary study. *arXiv preprint arXiv:2303.04048*.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q. V.; Zhou, D.; et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35: 24824–24837.

Zheng, C.; Zhou, H.; Meng, F.; Zhou, J.; and Huang, M. 2023. Large language models are not robust multiple choice selectors. In *The Twelfth International Conference on Learning Representations*.