

TaskEval: Synthesised Evaluation for Foundation-Model Tasks

Dilani Widanapathiranage, Scott Barnett, Stefanus Kurniawan, Wannita Takerngsaksiri

{s224731539, scott.barnett, stefanus.kurniawan, wannita.takerngsaksiri}@deakin.edu.au

Applied Artificial Intelligence Initiative, Deakin University

Geelong, Australia

Abstract

Hallucinations are a key concern when creating applications that rely on Foundation models (FMs). Understanding where and how these subtle failures occur in an application relies on evaluation methods known as *evals*. Prior work focuses on defining new eval methods or benchmark datasets for specific tasks. However, neither helps a software team with a task-specific FM application when there is no metric or dataset. The demand for both automated approaches and deep integration of human insight makes this a challenging problem. We address this gap by proposing an approach to synthesise a FM task-specific evaluator program that provides automation and a custom UI for capturing feedback. The core novelty of our approach lies in: (1) a task-agnostic meta-model that captures properties of any FM task, (2) an interaction protocol for efficient use of human feedback, and (3) an eval synthesiser that selects or generates an appropriate set of evals. We implement our approach in TaskEval and demonstrate the concept on two diverse FM tasks: chart data extraction and document question answering. A preliminary evaluation on the quality of our selected evals shows 93% and 90% accuracy respectively. Our research tackles a growing problem facing engineering teams, how to evaluate and review outputs from FM tasks.

Keywords

Evals, Evaluation, Foundation Models tasks, Software Engineering

ACM Reference Format:

Dilani Widanapathiranage, Scott Barnett, Stefanus Kurniawan, Wannita Takerngsaksiri. 2025. TaskEval: Synthesised Evaluation for Foundation-Model Tasks. In . ACM, New York, NY, USA, 5 pages. <https://doi.org/10.1145/nnnnnnn.nnnnnnn>

1 Introduction

Complex tasks for software have become ubiquitous through advances in foundation models (FM). Consider Figure 1 showing the task of converting an image of a chart to a table of records. Prior to foundation models, this task requires a) extensive data collection, preparation, and labelling, b) computer vision and machine learning expertise, and c) continuous training and evaluation of machine learning models. With foundation models the same effect is achieved with a simple prompt in a few seconds. However,

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Conference'17, Washington, DC, USA

© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM

<https://doi.org/10.1145/nnnnnnn.nnnnnnn>

Task Description:

Extract all the data points from the chart image and provide the output as a table.

Input Chart:

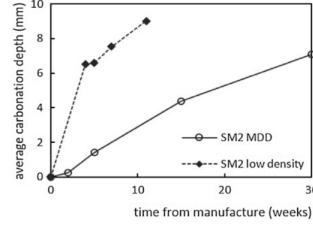


Fig. 5. Carbonation depth comparison of SM2 low density and MDD specimens.

Model Output (GPT-5):

time (weeks)	SM2 MDD (mm)	SM2 low-density (mm)
0	0.0	0.0
4	1.2	6.5
5	—	—
7	—	7.5
11	—	9.2
15	4.5	—
20	5.6	—
30	7.0	—

Figure 1: Chart-to-dataframe example [17] with GPT-5. Human review required to identify: incorrect values (red), spurious values (yellow), and missing values (purple).

foundation models are prone to hallucinate causing subtle and unpredictable errors in the output (i.e. missing values and wrong values in Figure 1). A recent finding indicates that this can be reduced with better evals [8] - metrics and processes to evaluate a foundation model on a task (an FM task). This problem now faces engineering teams who need domain and task-specific evals in order to build robust software.

Evals have been extensively studied in the literature [5, 7, 10, 15, 19, 22, 27]. What is clear from this research is the importance of evals and the consideration of limitations of popular approaches such as LLM-as-a-judge – where large language models (a type of foundation model) evaluate the output of foundation models [23, 25]. Most work on evals focuses on either presenting a new eval [11, 14] or creating a benchmark dataset [12, 16, 26, 28] to assist with evaluating foundation models. Neither are helpful to software engineering teams who have a task-specific application for foundation models where there is no eval or available dataset. Existing work also does not consider the monitoring needs of continuous evaluation where a label is not available instantly [4, 11]. The gap in the literature is there is no one technique that is a) task-agnostic, b) support no labelled examples, and c) supports operational requirements such as monitoring. We address this gap.

Our core idea is to synthesise an FM task-specific set of evals and UI for labelling examples. The novelty of our approach lies in 1) specifying a task-agnostic meta-model, 2) an interaction protocol for extracting task knowledge, and 3) an eval synthesiser that creates a custom eval or selects from a set of known evals. We have

demonstrated the core elements of our approach in TaskEval and conducted a preliminary evaluation of our approach on two diverse FM tasks (extracting data from a chart and question and answer use case). The output from TaskEval includes an evaluator UI for labelling data and viewing eval results and an evaluator API for operational use cases such as CI. TaskEval is designed to help teams bootstrap tooling for inspecting, labelling and evaluating their FM tasks. The research is guided by the following questions:

- **RQ1:** How do we model FM tasks in a task-agnostic way?
- **RQ2:** How can we extract FM task details in a task-agnostic way?
- **RQ3:** How can we synthesise task-specific evals for automatic evaluation and manual inspection?

2 Motivation

Our research is motivated by a recent collaboration with a material scientist to use an FM to extract tabular data from charts. The following scenario highlights the core challenges. Consider Sarah, a software developer, tasked with creating a system to extract data from charts for the purpose of analyzing crystalline structures and thermal properties. See Figure 1 for an example.

Sarah begins by using an FM for the task by writing a simple prompt to extract data from a chart. To evaluate the output Sarah needs a dataset with labels. As she is just starting, a dataset is not available so she needs to either create one herself with support of available tools (i.e. WebPlotDigitizer [1]) or find a relevant open-source dataset (which is not always available). Sarah does not find a suitable dataset and has to create one herself. She also finds that existing evaluation frameworks are task-specific and don't capture the nuances of chart data extraction.

After running the evaluation and comparing the extracted results against the labels Sarah finds the following issues: missing values, misidentified coordinates, and incorrect scales. Sarah is glad for her labelled dataset but realises that these errors are always going to occur. Sarah realises that even if her results are promising the material scientist will need to validate the output for correctness, full automation is only possible if the errors are acceptable.

Sarah decides to create a tool to help the material scientist inspect the output from the chart data extraction task. This tool uses visualisations and a modified eval to help speed up the review process. This strikes a happy medium between full automation and human inspection. However, Sarah realises i) existing evals are task-specific, ii) evaluation strategies must be manually designed for each use case, and iii) each task requires custom tool and visualisations.

To help Sarah, we propose TaskEval, a framework that automatically synthesises a task-specific evaluator to support evaluation and human labelling of datasets.

3 Vision

Our goal is to help engineering teams improve the quality of their evals of FM tasks. We achieve this by synthesising a task-specific evaluator and interface that supports human inspection. We implemented our ideas in TaskEval to demonstrate the core elements. Our approach combines meta-modelling [2] from model-driven engineering (MDE) [3] with domain-modelling. An overview of

the core elements of TaskEval are shown in Figure 2. Below, we describe the core components of our proposed solution.

3.1 Task-agnostic Meta-model

RQ1. How do we model FM tasks in a task-agnostic way?

The **task-agnostic Meta-model** represents the essential properties of an FM task and elements required for creating an eval. TaskEval accepts as input a description of the task and input for the task. Based on this information an instance of the task-agnostic meta-model is created and used to select appropriate evals. We anticipate needing to come up with criteria (and potentially new data fields) to adequately represent the context of an FM task. We plan to investigate multiple facets including: (i) input/output specification (modalities and formats), (ii) task type, (iii) evidence and grounding requirements, (iv) constraints (schema, units, ranges), (v) reasoning mode and answer multiplicity, (vi) evaluation objectives, and (vii) reference sources. The meta-model will define the relationships among these concepts. It is unlikely that a user will include this information on first pass, we plan to use an interaction protocol to iteratively populate the meta-model.

3.2 Task Interaction Protocol

RQ2. How can we extract FM task details in a task-agnostic way?

The **Interaction Protocol** enables TaskEval to incrementally instantiate the meta-model and to capture feedback. We present our protocol following guidelines from the literature [9]. It formalises how task knowledge is elicited, validated, and compiled into a runnable procedure, supporting: (i) *validation of potential errors*; (ii) *validation or override of proposed strategies*; and (iii) *refinement of evaluation requirements* via add/delete/edit operations.

Message Types — Request types include *Validate Errors*, *Update Evaluation Objective*, and *Propose Strategies*; approval flow via *Approve Plan*; execution via *Run Evaluation*; and iteration via *Provide Feedback*. Responses return structured feedback to *approve*, *reject*, or *amend*.

Syntax — the syntax of our protocol is rooted in markdown that is both human and machine readable. Includes: *task type*, *io* (input/output), *constraints*, *objectives* (evaluation criteria), *Potential errors*, and *proposed strategies*.

Field Semantics — This includes specifies what the task output must align. *Task type* selects evaluator templates; *io* fixes modalities and formats; *constraints* (schema/units/ranges) enable reference-free checks; *objectives* specify what outputs must satisfy; *potential errors* describe hypothesised failure modes; *proposed strategies* bind descriptors to executable checks.

Behavioural Rules — The protocol follows a four-stage loop: Elicit (validate errors → feedback), Map (validate strategy mappings → feedback/objective update), Run (execute evaluation → results/error report), Refine (apply edits → update descriptor/plan). The loop repeats until the evaluation plan is finalised.

3.3 Eval Synthesiser

RQ3. How can we synthesise task-specific evals for automatic evaluation and manual inspection?

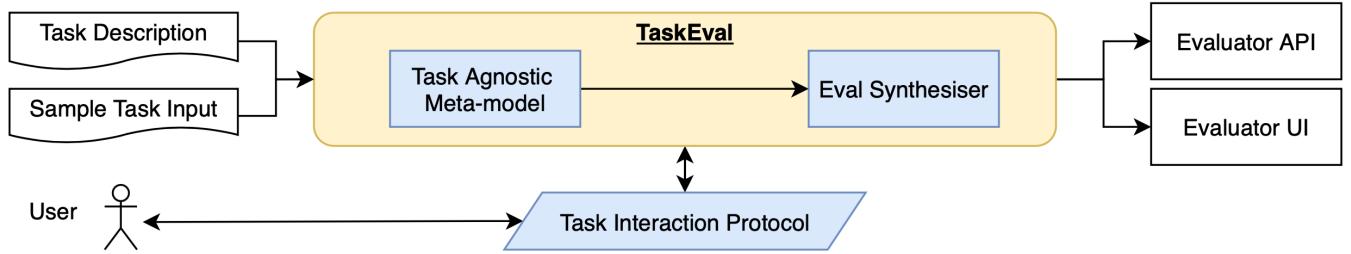


Figure 2: TASKEVAL system overview. Blue boxes show the novel elements of our solution: a) Task Interaction protocol, b) task-agnostic meta-model, and c) Eval Synthesiser.

The **Eval Synthesiser** is responsible for synthesising and/or selecting an appropriate set of evals and determining the UI components required for displaying these evals. Choosing to create a custom eval or use an existing eval is based on the task and dimensions of evaluation that are expressed by the user. This approach enables our evals to include multiple dimensions. An important aspect of the UI generation is the consideration of how user feedback is captured. For example, in the table from chart example from Figure 1 a UI would need to be able to edit the extracted data when the FM makes an error. We design the Eval Synthesiser to ensure that labelled data is collected as people interact with the system. Our demonstrator includes a set of eval strategies that are categorised as a) summarize, b) visualize, c) Judge, and d) logic program – custom code generation. Each of these approaches includes pre-defined evals from the literature (i.e. rubric-based assessment such as G-Eval [13]). The Eval Synthesiser produces the Evaluator API and the Evaluator UI. The API is designed for integration with operational infrastructure such as continuous integration pipelines.

4 Why is it new?

Task-agnostic and modality independent. Our approach works for any task including multi-dimensional tasks, objective or subjective, and FM or machine learning outputs. This is to support the next generation of FM Systems that use agents and techniques such as ML to solve problems. Unlike existing evaluation frameworks designed for specific domains or tasks, TaskEval operates through a meta-model that captures task-agnostic descriptors, enabling synthesis of evaluations across diverse applications. This generalizability is essential as FM systems increasingly handle complex, multi-step problems where evaluation requirements vary significantly.

Generate evaluation strategy and labelling UI

The oracle problem in FM evaluation, deciding correctness without ground truth remains unresolved. Existing approaches depend on costly human inspection [6, 24] or on LLM-as-a-Judge methods that are inconsistent and themselves require evaluation. Prior tools are tied to benchmarks or labelled data, limiting reuse across tasks. TaskEval is new because it provides a ground-truth independent evaluation framework. It combines structured summaries, visualisations, automated judging, and evaluation logic in a unified UI. Unlike prior work, it supports continuous evaluation at inference time where labelled data is unavailable. Its reusable, project-agnostic

design allows evaluation components to be adapted across domains without rebuilding logic.

Evaluator API for use in development and application
TaskEval generates an evaluation service API that is used by the interface to support engineering teams in evaluation. However, the API is also available for creating end user evaluation functionality enabling teams to focus on the application and not on the evaluation requirements. To the best of our knowledge this is the first approach that dual guides evaluation in development and with end user.

5 TaskEval Demonstration

To demonstrate the framework, we implemented a proof-of-concept TaskEval for two different tasks: (i) chart data extraction and (ii) question answering over documents. We conducted a preliminary evaluation of the ability for generating task-specific outputs and to assess the effort in evaluating the evals. Our tool synthesises task-specific evaluations based on the description of the task and a sample input. Figure 3 illustrates the generated UI for the two tasks and examples of evals.

Task-specific generation of evaluations: TaskEval uses a single pipeline with a shared meta-model. The meta-model captures essential elements of a task based on the task description and sample input. TaskEval then selects relevant failure modes and binds them to reusable strategy templates. Figure 3 shows TaskEval synthesising different evaluators for two FM tasks by adapting key attributes. **Task input:** *Chart data extraction*: task description and a chart image, *Document QA*: task description, source document, and expected answer(optional). **Task-specific eval:** *Chart data extraction*: Visualise strategy that redraws the chart from the extracted points for a side-by-side comparison with the original chart image. *Document QA*: combines Visualise and Judge strategies, highlighting source spans that support answer claims and computing evaluation metrics (e.g., answer relevancy, faithfulness) via DeepEval¹. **Task-specific UI:** *Chart data extraction*: side-by-side original and regenerated chart images with the extracted data table for inspection. *Document QA*: displays highlighted evidence alongside the evaluation metrics. **task-specific label:** *Chart data extraction*: numeric cell edits in the extracted table (e.g., correcting incorrect values from the motivating example in Figure 1). *Document QA*: binary approvals on provided sources to confirm whether the answer is supported.

¹<https://github.com/confident-ai/deepeval>

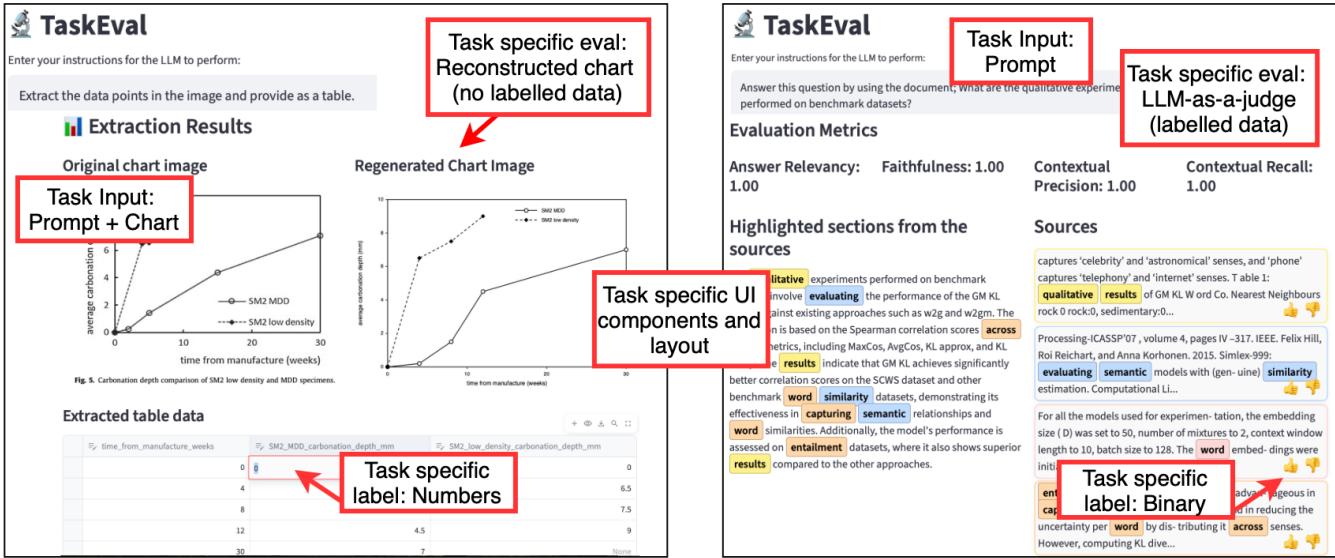


Figure 3: Output from TaskEval for two tasks: chart data extraction (left) and document QA (right). Depending on task description, availability of labels and input data, TaskEval selects task-specific a) evals (custom visualisation vs LLM-as-a-judge), b) human labels, and c) UI components and layout.

Effort to evaluate the evals: TaskEval generates code and other output using FMs which creates a problem of evaluating the evals. To evaluate the significance of this problem we performed a small experiment for each use case. For chart data extraction, we checked whether the regenerated plot matched the extracted points. On 30 charts, 28 were correct (93%). For document QA, we verified whether the answer was mainly sourced from the passage with the highest count of highlighted words. Across 30 document QA tested, 27 met this criterion (90%).

6 Future plans

Experimental evaluation of the generalisability of TaskEval. We plan to extend TaskEval by supporting additional tasks including objective tasks (i.e. evaluation can be confirmed as right/wrong) and subjective tasks (i.e. multiple right answers). This will help us to discover the essential elements of a task and domain that needed to be modelled. We will use this information to update and refine our hierarchical task-agnostic meta-model (Section 3.1). Our evaluation plan is to compare our task-agnostic approach to other task *specific* evaluation methods. This will help us discover the strengths and limitations of generalising across tasks. The outcome of this research will be an updated meta-model.

Evaluating the protocol with industry case studies. The next stage of our research is to evaluate the protocol (Section 3.2) through a series of industry case studies. We picked case studies as we want to understand how to improve the way practitioners evaluate FM tasks [18]. A case study will also enable us to gauge the preferences and benefits of our approach compared to manually creating evals and supporting tools. From collaborating with our industry partners we have found a wider range of FM tasks than are reported in the literature which will further provide insight into

how we model the domain. The outcome from this evaluation will be refined protocol.

Optimise the creation of applications with FM tasks. The final stage of our research plan is to address a growing problem of *how do we evaluate large amounts of generated code* [21]. We will do this by extending our approach to optimise the end-to-end (i.e. design, implementation and testing) of any application with an FM task. TaskEval will be extended with a) data synthesis capabilities based on prior work [20], b) include FM task generation and optimisation, and c) include a generate, test, and optimise loop. By focusing on evaluation first, our intention is to speed up the design and implementation of FM tasks providing a human guided system for application generation. The evaluation will focus on comparing human implemented applications with generated applications against multiple dimensions .

7 Risk

Modelled task context does not generalise. When creating TaskEval we realised the challenge of capturing the context required for generating the evaluator programs. Without capturing sufficient context our approach will not generalise well to unseen FM tasks. To mitigate this risk our intention is to extend TaskEval to many other tasks that include a) different input/output modalities, b) complexities, and c) types.

Variation of tasks is too vast to model. For our approach to work we need to identify the essentials of multiple tasks to form a meta-model that enables the creation of tasks specific evaluation. This will require careful design of an agent that is able to work with engineering teams together to create evaluators. We believe this is an essential task moving forward as software increasingly writes the code the effort will shift to evaluation of the program.

Evaluating the evaluator. Our approach uses FMs to create the evaluator which means the evaluator also needs to be evaluated! We plan to design a workflow and process that guides developers on how to create and evaluate the evaluator incrementally to minimise the effort required to evaluate the evaluator.

References

- [1] Automeris. 2025. Automeris Digitization Tools. <https://automeris.io/>. Accessed: 2025-09-18.
- [2] Benoit Baudry, Clementine Nebut, and Yves Le Traon. 2007. Model-driven engineering for requirements analysis. In *11th IEEE international enterprise distributed object computing conference (EDOC 2007)*, IEEE, 459–459.
- [3] Jean Bézivin. 2005. Model driven engineering: An emerging technical space. In *International Summer School on Generative and Transformational Techniques in Software Engineering*. Springer, 36–64.
- [4] Ding Chen, Qingchen Yu, Pengyuan Wang, Wentao Zhang, Bo Tang, Feiyu Xiong, Xinch Li, Minchuan Yang, and Zhiyi Li. 2025. xverify: Efficient answer verifier for reasoning model evaluations. *arXiv preprint arXiv:2504.10481* (2025).
- [5] Michael Desmond, Zahra Ashktorab, Qian Pan, Casey Dugan, and James M Johnson. 2024. EvaluLLM: LLM assisted evaluation of generative outputs. In *Companion proceedings of the 29th international conference on intelligent user interfaces*. 30–32.
- [6] Kehua Feng, Keyan Ding, Hongzhi Tan, Kede Ma, Zhihua Wang, Shuangquan Guo, Yuzhou Cheng, Ge Sun, Guozhong Zheng, Qiang Zhang, et al. 2024. Sample-efficient human evaluation of large language models via maximum discrepancy competition. *arXiv preprint arXiv:2404.08008* (2024).
- [7] Sangwoo Heo, Sungwook Son, and Hyunwoo Park. 2025. HaluCheck: Explainable and verifiable automation for detecting hallucinations in LLM responses. *Expert Systems with Applications* 272 (2025), 126712.
- [8] Adam Tauman Kalai, Ofir Nachum, Santosh S Vempala, and Edwin Zhang. 2025. Why language models hallucinate. *arXiv preprint arXiv:2509.04664* (2025).
- [9] James Kurose and Keith Ross. 2010. Computer networks: A top down approach featuring the internet.
- [10] Yuran Li, Jama Huseein Mohamud, Chongren Sun, Di Wu, and Benoit Boulet. 2025. Leveraging llms as meta-judges: A multi-agent framework for evaluating llm judgments. *arXiv preprint arXiv:2504.17087* (2025).
- [11] Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. 2022. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110* (2022).
- [12] Zhenwen Liang, Ye Liu, Tong Niu, Xiangliang Zhang, Yingbo Zhou, and Semih Yavuz. 2024. Improving llm reasoning through scaling inference computation with collaborative verification. *arXiv preprint arXiv:2410.05318* (2024).
- [13] Tianyi Liu et al. 2023. G-Eval: NLG Evaluation using GPT-4 with Better Human Alignment. In *EMNLP*. <https://aclanthology.org/2023.emnlp-main.153.pdf>
- [14] 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* (2023).
- [15] Potsawee Manakul, Adian Liusie, and Mark JF Gales. 2023. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. *arXiv preprint arXiv:2303.08896* (2023).
- [16] Jinjun Peng, Leyi Cui, Kele Huang, Junfeng Yang, and Baishakhi Ray. 2025. Ceweal: Outcome-driven evaluation of functionality and security of llm code generation. In *2025 IEEE/ACM International Workshop on Large Language Models for Code (LLM4Code)*. IEEE, 33–40.
- [17] Maciej P. Polak and Dane Morgan. 2025. Leveraging Vision Capabilities of Multimodal LLMs for Automated Data Extraction from Plots. *arXiv* (2025). arXiv:2503.12326 <https://arxiv.org/abs/2503.12326>
- [18] Per Runeson and Martin Höst. 2009. Guidelines for Conducting and Reporting Case Study Research in Software Engineering. *Empirical Software Engineering* 14, 2 (April 2009), 131–164. doi:10.1007/s10664-008-9102-8
- [19] Shreya Shankar, JD Zamfirescu-Pereira, Björn Hartmann, Aditya Parameswaran, and Ian Arawja. 2024. Who validates the validators? aligning llm-assisted evaluation of llm outputs with human preferences. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*. 1–14.
- [20] Shangeetha Sivasothy, Scott Barnett, Stefanus Kurniawan, Zafaryab Rasool, and Rajesh Vasa. 2024. RAGProbe: An automated approach for evaluating RAG applications. *arXiv preprint arXiv:2409.19019* (2024).
- [21] Claudio Spiess, David Gros, Kunal Suresh Pai, Michael Pradel, Md Rafiqul Islam Rabin, Amin Alipour, Susmit Jha, Prem Devanbu, and Toufique Ahmed. 2024. Calibration and correctness of language models for code. *arXiv preprint arXiv:2402.02047* (2024).
- [22] Gaurang Sriramanan, Siddhant Bharti, Vinu Sankar Sadasivan, Shoumik Saha, Priyatham Kattakinda, and Soheil Feizi. 2024. Llm-check: Investigating detection of hallucinations in large language models. *Advances in Neural Information Processing Systems* 37 (2024), 34188–34216.
- [23] Aman Singh Thakur, Kartik Choudhary, Venkat Srinik Ramayapally, Sankaran Vaidyanathan, and Dieuwke Hupkes. 2024. Judging the judges: Evaluating alignment and vulnerabilities in llms-as-judges. *arXiv preprint arXiv:2406.12624* (2024).
- [24] Ruiqi Wang, Jiyu Guo, Cuiyun Gao, Guodong Fan, Chun Yong Chong, and Xin Xia. 2025. Can llms replace human evaluators? an empirical study of llm-as-a-judge in software engineering. *Proceedings of the ACM on Software Engineering* 2, ISSTA (2025), 1955–1977.
- [25] Hui Wei, Shenghua He, Tian Xia, Fei Liu, Andy Wong, Jingyang Lin, and Mei Han. 2024. Systematic evaluation of llm-as-a-judge in llm alignment tasks: Explainable metrics and diverse prompt templates. *arXiv preprint arXiv:2408.13006* (2024).
- [26] Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Ben Feuer, Siddhartha Jain, Ravid Schwartz-Ziv, Neel Jain, Khalid Saifullah, Siddartha Naidu, et al. 2024. Livebench: A challenging, contamination-free llm benchmark. *arXiv preprint arXiv:2406.19314* 4 (2024).
- [27] Zhenyu Wu, Qingkai Zeng, Zhihan Zhang, Zhaoxuan Tan, Chao Shen, and Meng Jiang. 2024. Large language models can self-correct with key condition verification. *arXiv preprint arXiv:2405.14092* (2024).
- [28] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems* 36 (2023), 46595–46623.