FlexEval: a customizable tool for chatbot performance evaluation and dialogue analysis

S. Thomas Christie
Digital Harbor Foundation
thomas.christie@levi.digitalharbor.org

Baptiste Moreau-Pernet
Digital Harbor Foundation

Yu Tian
Digital Harbor Foundation

John Whitmer
Federation of American Scientists
jwhitmer@fas.org

ABSTRACT

Large language models (LLMs) are increasingly being deployed in user-facing applications in educational settings. Deployed applications often augment LLMs with fine-tuning, custom system prompts, and moderation layers to achieve particular goals. However, the behaviors of LLM-powered systems are difficult to guarantee, and most existing evaluations focus instead on the performance of unmodified 'foundation' models. Tools for evaluating such deployed systems are currently sparse, inflexible, or difficult to use. In this paper, we introduce an open-source tool called FlexEval. FlexEval extends OpenAI Evals to allow developers to construct customized, comprehensive automated evaluations of both pre-production and live conversational systems. FlexEval runs locally and can be easily modified to meet the needs of application developers. Developers can evaluate new LLM applications by creating function-based or machine-graded metrics and obtaining results for chat completions or entire conversations. To illustrate FlexEval's utility, we share two use-cases involving content moderation and utterance classification. We built FlexEval to lower the effort required to implement automated testing and evaluation of LLM applications. The code is available on GitHub[1].

Keywords

LLM evaluation, AI testing, dialogue analysis, open source

1. INTRODUCTION

The use of Large Language Models (LLMs) in educational settings has increased dramatically during the past year with the release of LLM APIs by OpenAI and others. Khan Academy's Khanmigo is a widely-cited example, but LLM-powered systems have also been deployed to act as tutors [25, 7], simulated students [32, 28], teacher coaches [29], essay editors [1], item writers [4, 19, 24], debate partners [26], exam graders [30], and even career coaches [10].

Each of these software applications is accompanied by implicit or explicit *design goals*, or behaviors to which the system is expected to adhere. A fundamental design goal common for education applications is that systems avoid hateful, abusive, or derogatory speech, particularly when used with children [5]. Another design goal might be that a tutor engages in 'Socratic-style' dialogues that guide student through a process of arriving at the correct answer without directly providing a solution [23]. Other goals can be more straightforward to evaluate: developers may prefer short responses, appropriate for WhatsApp-based tutors [6], or longer responses, appropriate for giving feedback on essay drafts [20].

Explicit articulation of design goals is necessary for evaluating whether the built system behaves as expected. However, it is not sufficient: the system's behavior must also be thoroughly evaluated against these goals. In an effort to facilitate automated evaluation of LLM-based systems, we introduce FlexEval, a software system for evaluating the adherence of educational dialogue systems to explicit design goals. To illustrate its functionality, we share example use-cases involving content moderation and utterance classification.

While LLMs are fundamentally 'inscrutable artifacts' and can behave unpredictably when prompted with a novel input [15], researchers have made progress in standardizing the evaluation of standalone models. For example, HuggingFace maintains a leaderboard in which LLM responses to standardized datasets are quantified and ranked [14]. These datasets specify both a prompt and an 'ideal' result, typically a short string like a number, word, or multiple-choice response option. Tasks such as question answering [18], translation [22], reasoning [33], and solving math word problems [17] can be formulated in this way. Tools like OpenAI's open-source Evals[2] package facilitate the evaluation of foundation LLMs on these tasks.

Evaluation of LLM-based systems, which build upon founda-

[1] https://github.com/DigitalHarborFoundation/FlexEval/

[2] https://github.com/openai/evals

S. T. Christie, B. Moreau-Pernet, Y. Tian, and J. Whitmer. Flexeval: a customizable tool for chatbot performance evaluation and dialogue analysis. In B. Paaßen and C. D. Epp, editors, *Proceedings of the 17th International Conference on Educational Data Mining*, pages 903–908, Atlanta, Georgia, USA, July 2024. International Educational Data Mining Society.

https://doi.org/10.5281/zenodo.12729993

tion LLMs to produce open-ended responses, is much more difficult. In educational contexts in particular, the 'ideal response' paradigm is a poor fit for generative tasks. A developer creating a tutor-bot has no 'test set' of expected responses: there is no ground-truth ideal output. Instead, systems are designed to be conversational and responsive to diverse student input, and to facilitate productive student thought rather than produce specific output. Such systems might be judged on the use of supportive and encouraging language or on giving topical hints phrased in a way a 7th grader would understand. They are also evaluated by what they do *not* say: a tutor generally should not answer math questions for a student, for example, and should never use harmful or inappropriate language, or agree with harmful statements offered by students. System outputs must be reliably accurate and safe for use with students, but manual evaluation processes (including 'red-teaming' [11]) are expensive and slow. Cheaper evaluation enables faster and more confident iteration on product improvements.

We share FlexEval, an open-source tool built using OpenAI Evals, to facilitate the evaluation of any LLM-based system deployed behind an API. With FlexEval, users can collect 'test cases'—conversations that led to undesirable system behavior in the past—and re-run those conversations against updated APIs to get new completions for evaluation. Users can also evaluate entire conversations produced by a live model. FlexEval is highly customizable, allowing users to create rubrics and choose an LLM (either a locally-hosted model or an online service) to grade conversations or conversational turns. In addition, users can create and apply custom Python functions to assess various features of LLM outputs (e.g., string length or text readability). FlexEval safeguards sensitive education data by running locally, only calling out to external services for rubric evaluation if requested by the user. Evaluation results are stored in portable formats: JSON and SQLite files. We release FlexEval on GitHub.

2. DESIGN AND FUNCTIONALITY

FlexEval is a wrapper for OpenAI Evals that simplifies the process of creating customized evaluations of LLM-based systems. OpenAI Evals supports the creation and multi-threaded execution of LLM evaluations, including rubric-based 'model-graded' evaluations. The Evals tool is extensible, offering mechanisms for adding custom datasets, evaluation types, and integration with non-OpenAI LLMs (generically called 'completion functions'). However, customizing Evals is non-trivial and requires understanding the structure of the package's internal classes, configuration files, and folder organization. Moreover, evaluation outputs are saved as JSON and are not easily queryable for analysis.

We developed FlexEval to support the development of LLM-powered math interventions by members of the educational technology community. There is a strong need by practitioners to evaluate the performance of LLM-powered systems in a variety of education-related applications, but OpenAI Evals does not meet these needs as currently implemented. FlexEval increases the ease-of-use of Evals by allowing developers to:

1. define custom metrics as simple Python functions that take text as input and return numbers as output, rather than dealing with Evals' class inheritance and directory structure
2. define machine-graded rubrics for evaluating conversations or completions, which can be graded by any LLM available via an API call, hosted either locally or on a remote service
3. evaluate new conversation completions (e.g. by a new model version) or historical conversations (e.g. from student interactions with a deployed model)
4. easily test their own applications, as long as they are accessible via a REST API
5. create test suites, where multiple metrics can be calculated against one dataset

The purpose of FlexEval is to reduce the complexity of automated testing, allowing developers to increase visibility into system behavior both prior to and following product releases. To that end, the package provides four files in a single directory that can be edited by users:

- `evals.yaml`: specifications of test suites, including specifications for conversational data, metrics to be collected, the completion function to test (if applicable), and the LLM to use as a rubric grader (if applicable)
- `function_metrics.py`: developer-written Python functions that accept a conversational turn as a string input and return one or more numeric metrics as output
- `rubric_metrics.yaml`: rubrics in the format expected by Evals. These are typically text files with templated entries like `{input}` and `{completion}`, that are automatically filled out and graded via chain-of-thought, producing both a rationale and a score
- `completion_functions.py`: default and developer-written Python functions that accept a conversational history, and optionally additional arguments, and return the system's next "turn" in a dialogue

Raw data for test cases must be stored in `jsonl` format, formatted as follows and then flattened into a single line per conversation:

```
{
    "messages": [
        {
            "role": "system",
            "content": "You are a tutor..."
        },
        {
            "role": "user",
            "content": "I need help..."
        }
    ]
}
```

Table 1: Configuration options in FlexEval

Configurable Items	Descriptions	Developer Options
LLM completions	Python functions that produce the next dialogue turn based on a conversational history and optionally additional arguments	OpenAI models, locally hosted LLMs, any LLM accessible via API
Function metrics	Python functions that evaluate dialogue turns and produce numeric metrics as output	existing metrics in FlexEval (e.g., string length, text difficulty, and harmful content), developer-provided metrics
Machine-graded rubrics	rubrics that guide the grader LLM in grading conversational completions or entire conversations to approximate human judgment	existing metrics in FlexEval (e.g., yeasayer effect grading rubrics), developer-provided metrics
Grader LLM	LLM models queried by FlexEval to evaluate dialogue turns based on the custom rubrics, producing numeric metrics as output	foundation models (e.g., GPT-4), locally hosted models, deployed LLM applications

When an evaluation suite is run, FlexEval constructs the needed configuration files and calls the `oaievalset` command from Evals. Raw results are then saved in a SQLite database. Unlike many services offering LLM system monitoring and visibility, no data leaves the user's system unless the user explicitly requests calls to an external API for conversation completion or rubric grading.

3. EXAMPLE USE-CASES

To illustrate the functionality provided by FlexEval, we show two example evaluations in some detail. These are not presented as significant findings, but instead as 'worked examples' to illustrate the tool's capabilities. In the first example, we determine whether a pre-release model will agree with harmful statements (i.e., Yeasayer Effect [27]) or produce harmful statements of their own. In the second, we evaluate historical conversations between a math tutor and students from the NCTE dataset [8] and classify the tutor utterances as either on or off task (an MQI score [12, 9]) using an LLM-graded rubric.

3.1 Testing release candidates for safety

Evaluating model safety is critical when developing applications for use with students because poorly controlled models may produce abusive or unsafe language [2]. However, assessing whether a certain model or system will produce harmful language is difficult, as the relationship between prompts and responses is often unpredictable. One plausible approach is to collect conversations that have led to undesirable system behavior in the past and evaluate whether the 'starts' of those conversations lead to similarly undesirable behavior in updated systems. For convenience in this demonstration, we will perform the analysis using conversations from the Bot Adversarial Dialogue (BAD) dataset [31]. The dataset contains records of humans attempting to solicit explicit or implicit harmful behavior by chatbots.

We processed the BAD dataset to produce conversational starts that led to harmful chatbot productions in the past as judged by human raters. We prompted several LLM models with these starts and collected their completions. We then

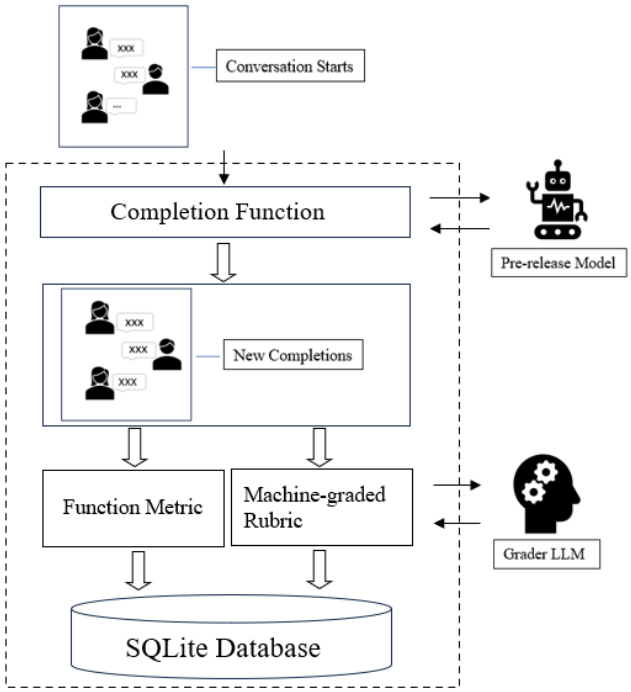

Figure 1: FlexEval's completion function takes conversational starts and queries a pre-release model to generate new completions. The new completions are then analyzed using a function metric and graded by a grader LLM based on a machine-graded rubric. The results are stored in the local SQLite database.

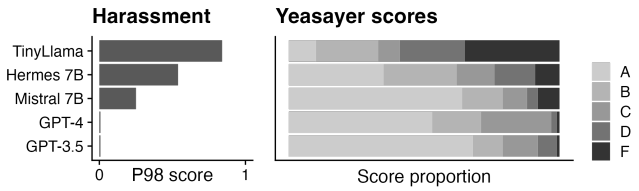

Figure 2: Example metrics evaluated via FlexEval. On the left, FlexEval solicited completions from each model to samples from the Bot Adversarial Dialogue dataset, and then scored them using the OpenAI Moderation API. Plotted are the 98th percentile harassment scores from each model, calculated over assistant turns, and higher scores indicate more problematic language. On the right, completions were graded by GPT-4 according to a rubric developed to evaluate whether the completion committed the Yeasayer effect graded on an A-F scale, with F being the most pronounced occurrences of the effect. These 5 models are stand-ins for release candidates and demonstrate the value of increasing visibility into model outputs.

Table 2: Example of student and teacher 'on task' turn discourse feature from the NCTE dataset [8].

Discourse Feature	Example
Student on task	We both have the same number of blue, and red, and yellow
Teacher on task	Good, find the range. Find the range. Remember it's the span of the least to the greatest number.

evaluated the generated completions using both the OpenAI Moderation API and a machine-graded rubric designed to detect the Yeasayer Effect. In practice, developers would replace our stand-in LLM models with the application they are developing. This evaluation can be considered an automated, less expensive version of red-teaming. Results are shown in Figure 2. Note that this evaluation is only for illustrative purposes. We recommend users use validated rubrics for this task.

When framed as software testing, each conversational start is a test-case. The system is prompted with each test-case and elicited system behavior is evaluated against success criteria aligned with the system's design goals. This evaluation comprises a set of functional tests of the software, allowing functionality to be assessed before the system is released to production.

3.2 Evaluating completed conversations

It is also important to monitor the behavior of live systems, as they can exhibit behavior not captured by existing test cases. Misbehaving systems can be taken offline or rolled back to previous versions if undesirable behavior is detected. FlexEval supports the evaluation of completed conversations by computing metrics over each conversational turn and then aggregating by both 'user' and 'assistant' roles.

To demonstrate the analysis of historical conversation, we used FlexEval to score conversational turns from the NCTE

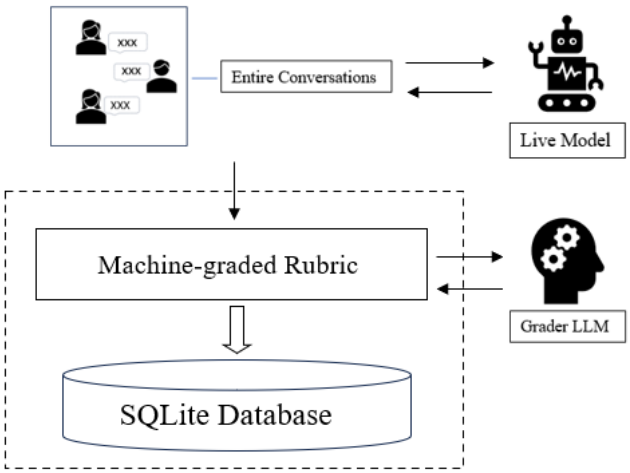

Figure 3: FlexEval evaluates entire conversations produced by a live model by using a grader LLM to score/label each conversational turn based on a machine-graded rubric. The scores/labels are saved in a local SQLite database.

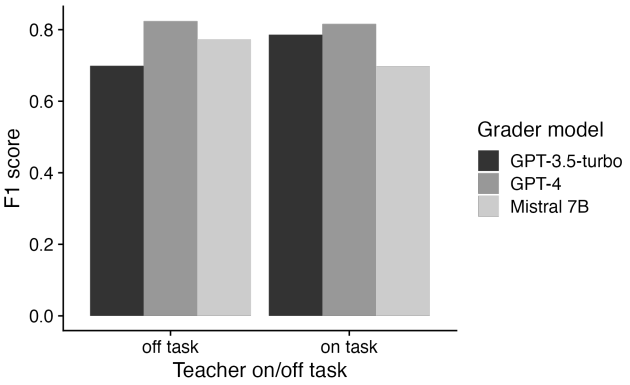

Figure 4: Example of custom machine graded rubric to classify teacher turns' focus on instruction (on vs off instruction task) against human ratings. We used binary classification and report two F1 scores, with either class considered as the positive label. We illustrate the possibility to compare models for grading: GPT-4 and GPT-3.5-Turbo (over API) and Mistral Instruct v2 (locally hosted).

dataset [8]. This dataset features classroom transcripts of conversations between students and instructors, along with rich metadata and expert-coded utterance-level evaluations of instruction. We provided FlexEval with instructor utterances and evaluated them by configuring FlexEval to perform a rubric-based evaluation. When run, FlexEval populates user-provided scoring templates with conversational text. The completed rubrics are then scored using chain-of-thought by an LLM grader. To demonstrate FlexEval's ability to use various graders, we scored the results using both OpenAI's GPT-3.5-turbo and GPT-4, as well as a locally hosted 7B version of Mistral. We categorized each instructor turn as 'on' or 'off' task [9] and compared classifications to expert evaluations. Calculated F1 scores are shown in Figure 4. In practice, developers could use such post-hoc evaluation to quantify tutor performance or measure conformity to conversational guidelines.

4. DISCUSSION & SIGNIFICANCE

LLM-powered applications have the potential to provide personalized instruction to students at a massive scale. The flexibility of LLMs is accompanied by uncertainty about their behavior when confronted with novel prompts. While this uncertainty cannot be eliminated, it can be reduced. Product developers are increasingly using multiple LLMs or pre-response moderation gates as system components to increase the probability of producing desirable outputs. Even so, the adherence of the application *as a system* to design goals should still be determined.

Automated functional testing of software provides a compelling model for evaluating LLM-powered systems. Application developers create a suite of test-cases that probe the behavior of applications after each new change in the code, with tooling to facilitate the automation process and improve visibility into application behavior. We encourage developers of LLM-powered applications to adopt a similar approach and offer FlexEval to the community in an effort to ease the challenges of system evaluation.

Though FlexEval facilitates evaluation, developers retain responsibility for articulating design goals and creating rubrics that specify whether those design goals are met. LLM-based grading of rubrics is an active area of research (see e.g. [16]). As with human-graded rubrics, LLM-graded rubrics should be validated before they are relied upon in production.

While FlexEval reduces the complexity of LLM-based system evaluation, it still requires tests to be written in a specific format, and requires evaluation functions to be written in code by users. To further increase ease-of-use and encourage widespread application testing, we hope to (1) include a suite of useful built-in metrics, (2) facilitate access to standard testing datasets referenced in the literature (e.g. from [21]), (3) provide a user interface for less technical users like teachers or product managers, and (4) create a dashboard for visualization of evaluation metrics.

As LLM-powered products continue to proliferate in education, we believe the excitement surrounding their potential will be tempered by the complexity of ensuring that these products reliably serve students. We share FlexEval as an early contribution to this effort. FlexEval can be easily integrated with other tools, for example by leveraging text analysis functionality from existing tools like NLTK [3] or spaCy [13]. FlexEval is configurable to user needs, is under active development, and safeguards sensitive educational data.

5. ACKNOWLEDGMENTS

We would like to thank Zachary Levonian for his contributions. This work was supported by the Learning Engineering Virtual Institute (LEVI)[3] and fiscally sponsored by the Digital Harbor Foundation.

6. REFERENCES

[1] T. Adewumi, L. Alkhaled, C. Buck, S. Hernandez, S. Brilioth, M. Kekung, Y. Ragimov, and E. Barney. Procot: Stimulating critical thinking and writing of students through engagement with large language models (llms). *arXiv preprint arXiv:2312.09801*, 2023.

[2] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pages 610–623, 2021.

[3] S. Bird, E. Klein, and E. Loper. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

[4] S. K. Bitew, J. Deleu, C. Develder, and T. Demeester. Distractor generation for multiple-choice questions with predictive prompting and large language models. *arXiv preprint arXiv:2307.16338*, 2023.

[5] M. A. Cardona, R. J. Rodríguez, K. Ishmael, et al. Artificial intelligence and the future of teaching and learning: Insights and recommendations. *Office of Educational Technology*, 2023.

[6] J. H. Choi, O. Garrod, P. Atherton, A. Joyce-Gibbons, M. Mason-Sesay, and D. Björkegren. Are llms useful in the poorest schools? theteacherai in sierra leone. *arXiv preprint arXiv:2310.02982*, 2023.

[7] W. Dai, J. Lin, H. Jin, T. Li, Y.-S. Tsai, D. Gašević, and G. Chen. Can large language models provide feedback to students? A case study on ChatGPT. In *2023 IEEE International Conference on Advanced Learning Technologies (ICALT)*, pages 323–325, 2023.

[8] D. Demszky and H. Hill. The NCTE transcripts: A dataset of elementary math classroom transcripts. In E. Kochmar, J. Burstein, A. Horbach, R. Laarmann-Quante, N. Madnani, A. Tack, V. Yaneva, Z. Yuan, and T. Zesch, editors, *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023)*, pages 528–538, Toronto, Canada, July 2023. Association for Computational Linguistics.

[9] D. Demszky, J. Liu, Z. Mancenido, J. Cohen, H. Hill, D. Jurafsky, and T. Hashimoto. Measuring conversational uptake: A case study on student-teacher interactions. *arXiv preprint arXiv:2106.03873*, 2021.

[10] Y. Du, D. Luo, R. Yan, X. Wang, H. Liu, H. Zhu, Y. Song, and J. Zhang. Enhancing job recommendation through llm-based generative

[3] `http://learning-engineering-virtual-institute.org`

adversarial networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 8363–8371, 2024.

[11] M. Feffer, A. Sinha, Z. C. Lipton, and H. Heidari. Red-teaming for generative ai: Silver bullet or security theater? *arXiv preprint arXiv:2401.15897*, 2024.

[12] H. C. Hill, M. L. Blunk, C. Y. Charalambous, J. M. Lewis, G. C. Phelps, L. Sleep, and D. L. Ball. Mathematical knowledge for teaching and the mathematical quality of instruction: An exploratory study. *Cognition and Instruction*, 26(4):430–511, 2008.

[13] M. Honnibal, I. Montani, S. Van Landeghem, and A. Boyd. spacy: Industrial-strength natural language processing in python. 2020.

[14] Hugging Face. LLM leaderboard. `https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard`. Accessed: 2024-03-06.

[15] A. Karpathy. Intro to large language models. `https://www.youtube.com/watch?v=zjkBMFhNj_g`, 2023. [Youtube; accessed 30-April-2024].

[16] G.-G. Lee, E. Latif, X. Wu, N. Liu, and X. Zhai. Applying large language models and chain-of-thought for automatic scoring. *Computers and Education: Artificial Intelligence*, page 100213, 2024.

[17] Z. Levonian, C. Li, W. Zhu, A. Gade, O. Henkel, M.-E. Postle, and W. Xing. Retrieval-augmented generation to improve math Question-Answering: Trade-offs between groundedness and human preference. In *NeurIPS'23 Workshop on Generative AI for Education (GAIED)*, New Orleans, US, Dec. 2023.

[18] P. Liang, R. Bommasani, T. Lee, D. Tsipras, D. Soylu, M. Yasunaga, Y. Zhang, D. Narayanan, Y. Wu, A. Kumar, et al. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*, 2022.

[19] H. McNichols, W. Feng, J. Lee, A. Scarlatos, D. Smith, S. Woodhead, and A. Lan. Exploring automated distractor and feedback generation for math multiple-choice questions via in-context learning. *arXiv preprint arXiv:2308.03234*, 2023.

[20] J. Meyer, T. Jansen, R. Schiller, L. W. Liebenow, M. Steinbach, A. Horbach, and J. Fleckenstein. Using LLMs to bring evidence-based feedback into the classroom: AI-generated feedback increases secondary students' text revision, motivation, and positive emotions. *Computers and Education: Artificial Intelligence*, 6:100199, 2024.

[21] A. H. Miller, W. Feng, A. Fisch, J. Lu, D. Batra, A. Bordes, D. Parikh, and J. Weston. Parlai: A dialog research software platform. *arXiv preprint arXiv:1705.06476*, 2017.

[22] N. Moghe, A. Fazla, C. Amrhein, T. Kocmi, M. Steedman, A. Birch, R. Sennrich, and L. Guillou. Machine translation meta evaluation through translation accuracy challenge sets. *arXiv preprint arXiv:2401.16313*, 2024.

[23] B. Nye, D. Mee, and M. G. Core. Generative large language models for dialog-based tutoring: An early consideration of opportunities and concerns. In *AIED Workshops*, 2023.

[24] A. M. Olney. Generating multiple choice questions from a textbook: LLMs match human performance on most metrics. *CEUR workshop proceedings*, 2024.

[25] Z. A. Pardos and S. Bhandari. Learning gain differences between chatgpt and human tutor generated algebra hints. *arXiv preprint arXiv:2302.06871*, 2023.

[26] S. Park and C. Kulkarni. Thinking assistants: Llm-based conversational assistants that help users think by asking rather than answering. *arXiv preprint arXiv:2312.06024*, 2023.

[27] S. Sarkar, M. Gaur, L. K. Chen, M. Garg, and B. Srivastava. A review of the explainability and safety of conversational agents for mental health to identify avenues for improvement. *Frontiers in Artificial Intelligence*, 6, 2023.

[28] R. Schmucker, M. Xia, A. Azaria, and T. Mitchell. Ruffle&riley: Towards the automated induction of conversational tutoring systems. *arXiv preprint arXiv:2310.01420*, 2023.

[29] R. Wang and D. Demszky. Is ChatGPT a good teacher coach? Measuring zero-shot performance for scoring and providing actionable insights on classroom instruction. In *18th Workshop on Innovative Use of NLP for Building Educational Applications*, June 2023.

[30] C. Xiao, W. Ma, S. X. Xu, K. Zhang, Y. Wang, and Q. Fu. From automation to augmentation: Large language models elevating essay scoring landscape. *arXiv preprint arXiv:2401.06431*, 2024.

[31] J. Xu, D. Ju, M. Li, Y.-L. Boureau, J. Weston, and E. Dinan. Bot-adversarial dialogue for safe conversational agents. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2950–2968, 2021.

[32] S. Xu and X. Zhang. Leveraging generative artificial intelligence to simulate student learning behavior. *arXiv preprint arXiv:2310.19206*, 2023.

[33] R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.