Evaluating Large Language Model's in Educational Context

Kamel Charaf — Ivan Pavlov — Michele Smaldone
Department of Computer Science, EPFL Lausanne, Switzerland

I. Introduction

This report introduces an evaluation framework designed to assess the performance of large language models (LLMs) in educational contexts, with a focus on tasks related to STEM subjects. By using widely recognized benchmarks such as MATH, GSM8K, TutorEval, and MGSM, we measure model capabilities in problem-solving, reasoning, and explanation generation. The primary goal of this project is to identify the most suitable models for specific educational applications, including tutoring and grading. In addition, the introduction of the GraderEval dataset addresses the critical need for evaluating grading models, ensuring the fairness and reliability of their assessments.

II. The Datasets

The datasets were chosen for their relevance and popularity in benchmarking LLMs on STEM-related tasks. Each dataset focuses on a specific area of problem-solving:

- **GSM8K**: The GSM8K dataset contains over 8.5K high-quality linguistically diverse grade school mathematics problems. Since solutions are provided in natural language, this dataset is widely used to test stepwise reasoning ability [1].
- **MGSM**: The Multilingual Grade School Math Benchmark (MGSM) contains 250 problems from GSM8K, each of which is translated by human annotators into 10 different languages. In our evaluation, we used the German and French translations to test our multilingual models [2].
- **MATH**: The MATH dataset contains 12.5K challenging competition mathematics problems. Each problem is provided with step-by-step solutions and a final boxed answer. These problems are sorted into 7 different subjects and 5 difficulty levels (from 1 to 5) [3].
- **TutorEval**: This dataset is the most special, as this dataset is designed to simulate tutoring scenarios and is used to test the model's ability to provide clear and relevant explanations. The dataset contains relevant key points defined by human experts, which are guidelines about what a good response should contain [4].

III. Pipeline

The evaluation pipeline is designed to be highly modular and flexible, ensuring that it can easily accommodate new datasets, evaluation tasks, and models. This design allows new components to be added to the pipeline without requiring significant changes to the overall structure.

The pipeline consists of the following key components:

A. Datasets

Each dataset is implemented as a subclass of the `BaseDataset` class. This class defines the core functionality required for downloading, loading, and accessing datasets. It abstracts the common operations that are shared by different datasets, such as downloading the data from a remote source, organizing it, and loading it into memory for evaluation.

For example, datasets such as GSM8K, MATH, and MGSM each implement their specific behaviour for data loading and organization.

B. Evaluation Tasks

Evaluation tasks are implemented as subclasses of the `EvalTask` class, which is the base class for tasks that evaluate the performance of models on specific datasets. The `EvalTask` class provides the structure for performing the evaluation, including defining how to extract answers from model outputs and how to check that the generated answers are correct.

Each dataset can have multiple evaluation tasks, depending on the nature of the problem. The flexibility to define multiple tasks for a single dataset allows the pipeline to evaluate models in a variety of ways, such as few-shot or n-shot evaluations. For example, `GSM8KNShot` and `MATHFewShot`.

C. Models

The models in this pipeline are implemented as subclasses of the abstract `BaseModel` class. Each model class must implement the `generate` method, which is used to generate predictions based on the input data.

The `BaseModel` class provides an architecture where specific configurations or parameters, such as the model name or API key, can be passed to the model to adapt to different model-specific requirements. For example, the Swiss AI models are accessible through a special API endpoint via the `Ethel` model class. The `Ethel` class provides access to (`meta-llama/Llama-3.1-70B-Instruct`, `swissai/ethel-70b-pretrain`, `swissai/ethel-70b-magpie`).

The `Ollama` model class provides access to any locally running Ollama [5] model, for example, `llama3.2` [6]. The pipeline allows easy integration of additional models as long as they adhere to the `BaseModel` interface.

D. Evaluation Workflow

The pipeline runs in the following workflow:

1) **Dataset Selection**: The user specifies a dataset through command line arguments. The appropriate dataset class (e.g., `GSM8K`, `MATH`, `MGSM`) is instantiated, and the dataset is downloaded and loaded into memory.
2) **Task Selection**: Based on the selected dataset, an appropriate evaluation task class (e.g., `GSM8KNShot`, `MATHFewShot`, `MGSMNShot`) is selected. These task classes define the specific task to evaluate the model, such as few-shot or n-shot tasks, and provide the necessary logic to evaluate model performance.
3) **Model Selection**: A model is selected by specifying the model (e.g., `Ethel`, `Ollama`). If the exact model_name

is not specified, then the default model_name is used. The pipeline then loads the model and prepares it for evaluation.

4) **Model Evaluation**: For each sample in the dataset, the model generates a response. The evaluation task extracts the answer generated by the model and checks its correctness. The results are recorded, and the evaluation continues for the entire dataset or until the specified iteration limit is reached.
5) **Recording Results**: Throughout the evaluation process, the results, including input data, target answers, generated responses, and correctness, are stored in an output file. Failed attempts are also recorded for further analysis and possible re-run.

E. TutorEval Evaluation

The `TutorEval` dataset represents a special case in our evaluation pipeline, where two models are used: a **Tutor Model** and a **Grader Model**. The scoring process follows the following steps:

1) **Closed-Book vs Open-Book Evaluation:** The assessment can be performed in either closed-book or open-book settings, depending on whether the tutor model has access to a passage from a textbook.
2) **Tutor Model Generation:** The tutor model takes the (possibly extended) question or problem from the `TutorEval` dataset and generates an answer to it.
3) **Grading by Grader Model:** The grader model evaluates the tutor's response given the human-written **key points** based on two criteria: **Presentation** and **Correctness**

Metrics: The evaluation focuses on two key metrics as defined by [7]:

- **Presentation:** How well is the response presented in terms of clarity and structure?
- **Correctness:** How accurate is the tutor's response?

IV. Grading the Grader: A Novel Contribution to TutorEval Evaluation

While the `TutorEval` dataset allows for the evaluation of a **Tutor Model** by assessing its responses to the defined key points, it does not directly address the quality of the **Grader Model**. To bridge this gap, we propose a novel approach for **grading the grader** — an evaluation methodology that focuses on assessing the quality of the grading model itself.

A. Dataset Creation for Grading the Grader

To evaluate the grader it is essential to define its primary purpose. The grader compares the tutor's response with the human-written **key points** and assigns a grade accordingly. The quality of the grader model is determined by how accurately it can detect the presence of the **key points** within a given response.

Our method uses the `TutorEval` dataset to evaluate the grader. Specifically, we consider only closed-book questions, removing the misleading ones (according to the labels in `TutorEval`). This leaves us with 322 samples, which we transform into a new synthetic dataset, called `GraderEval`.

Question: What is the easiest way to make a while loop that never stops executing?
Key Points:

A explains that while loop will never stop executing if the condition is always satisfied
B the easiest example is while (1 == 1), which is always true
C is an even easier example is while True, which is always true

Key Point Mask: [_, B, C]
Rephrased Answer: One way to create a while loop that never stops executing is by using `while (1 == 1)`, as this condition is always true. An even simpler example is `while True`, which is inherently always true and thus results in an infinite loop.

Figure 1. Example from `GraderEval` dataset

1) Key Point Shuffling and Masking: For each question in `TutorEval` multiple variants were created by randomly masking (i.e., removing) one key point from the list of key points. This creates a new version of the question where one of the important guidelines is missing. For this process, we select only the questions with multiple key points, leaving us with 267 samples from `TutorEval`. After masking each key point, we get **768** samples.

2) Key Point Rephrasing: For each question, we ask an advanced Large Language model (December 2024 `GPT-4o` in our case) to create an imitation of a student's answer. The masked key points are randomly shuffled and combined into a prompt that asks the advanced LLM model to reformulate them into a coherent answer. See the prompt for this part in the Appendix IX-A.

See the sample from the resulting dataset in Figure 1. We can see that only key points B and C are present in the rephrased answer.

B. Grading the Grader: Key Steps

1) Grading Rephrased Answers: The grader model is then tasked with reviewing the response generated by the student (based on the modified key points) and determining which of the original key points are covered in the answer. The grader model's ability to identify relevant key points in the student's response is the primary metric of evaluation.

A specific grader prompt is created that includes the key points (with labels), the question, and the student's answer. The grader model is instructed to return a list of key points from the answer in a labelled format. See the full prompt in the Appendix IX-B.

2) Evaluating the Grader Model: The output of the grader model is compared with the correct key points, i.e. those that remained after the masking. To calculate the error of the grader model, we used the Jaccard-Distance metric (JD), which measures the dissimilarity between two sets. It is defined as:

$$\text{JD} = 1 - \text{Jaccard Index}$$

`Jaccard Index` is the similarity between the two sets calculated as:

$$\text{Jaccard Index} = \frac{|A \cup B|}{|A \cap B|},$$

where the first set is the key points identified by the grader, while the other is the ground truth. Each grader model is assigned to a `Jaccard-Distance` score for every prediction it makes and the grader's overall error is the average of the errors per question. The best grader model was **meta-llama/Meta-Llama-3.1-70B-Instruct**, so it was used as a grader model in the TutorEval evaluation pipeline.

C. *Dataset and Results*

The resulting `GraderEval` dataset containing 768 examples is open-sourced and available for download in our GitHub repository [8].

The novel `Grading the Grader` approach provides a framework for evaluating the quality of grader models in the context of TutorEval-like evaluations. By systematically masking key points and rephrasing prompts, we generate a dataset that challenges the grader model to accurately identify which key points are present in a student's answer. This process contributes to a better understanding of not only how well tutor models perform, but also how effectively grader models assess these responses.

V. Evaluation Results

This section presents the results of our evaluation pipeline on various datasets and tasks. Additionally, we discuss the outcomes of the `GraderEval` dataset evaluation for selecting the best grader model. See the Table I for the main results.

Specification of the models used:

1) **meta-llama/Meta- Llama-3.1-70B-Instruct** provided by SWISS AI API endpoint [6]
2) **meta-llama/Meta- Llama-3.3-70B-Instruct** provided by SWISS AI API endpoint [6]
3) **GPT-4o** December 2024 version, accessed by OpenAI API [9]
4) **SmolLM** by HuggingFace, run on Ollama `smollm:1.7b-base-v0.2-fp16`
5) **Llama 3.2 3b** run on Ollama [6]
6) **swissai/ethel-70b-magpie**, internal Swiss AI model, provided by Swiss AI
7) **swissai/ethel-70b-tutorchat**, internal Swiss AI model, provided by Swiss AI. The `Tutorchat` model is trained on top of the `Magpie` model checkpoint with additional data.

A. *MGSM Evaluation*

We performed n-shot evaluations on the MGSM dataset in two languages: French (FR) and German (DE) (those languages are especially interesting in the Swiss educational context). The accuracy metric is reported.

We present our results in Table V in the Appendix. We have compared **LLama-3.1-70B-Instruct** and **LLama-3.2** on the MGSM and found that the bigger model outperforms the smaller one. The models tend to perform worse for German, compared to French. Besides, while running evaluation for different n-shot constant, we observed that for German language, **Llama-3.2** with $8-$shot prompt could outperform the **LLama-3.1-70B-Instruct** with 0-shot configuration. This once again highlights the importance of prompting and power of n-shot approach. See the Figure 2 to observe how n, the number of shots, affects the performance.

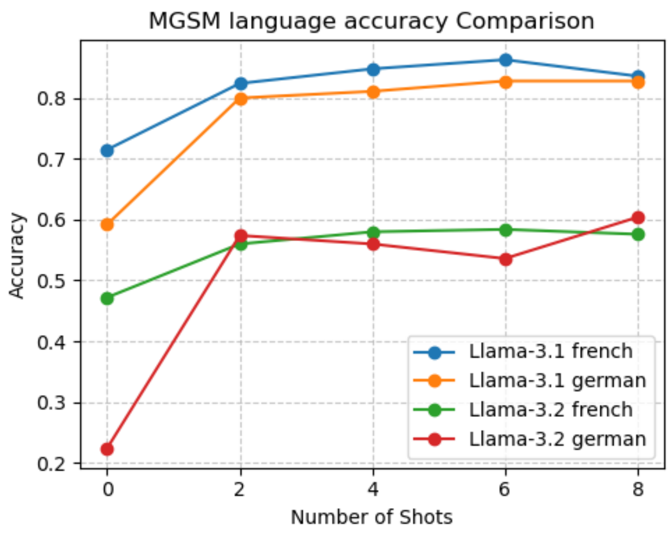

Figure 2. Llama 3.1 and Llama 3.2 N-shot learning for MGSM

B. *GSM8K Evaluation*

For the GSM8K dataset, we adopted an 8-shot prompting approach, following the methodology described in the Llama3 paper [6]. The Table I provides accurate results on GSM8K. We observe that **LLama-3.1-70B-Instruct** is the leader with an outstanding accuracy of 94.8%. Surprisingly, the SWISS AI 70B model **ethel-70b-magpie** shows poor performance, scoring lower than **Smol** model with 1B parameters. From the observation of the generated response, we can argue that ethel models are repetitive in their answers, looping over the same phrases and rarely providing a clear answer. We have communicated our findings to the model training team. At the same time, the 3B model **Llama-3.2** shows great results with 66.3% accuracy.

C. *MATH Evaluation*

For the MATH dataset evaluation we use the 4-shot problem made available by Lewkowycz et al. [10]. As MATH evaluation is heavily computationally intensive, we have compared two **Llama** models. Expectedly, **LLama-3.1-70B-Instruct** outperforming the **LLama-3.2-3B**. We have also categorised evaluation results to a different topic, as annotation is provided in the dataset. As shown in Figure 3, Pre-Algebra and Algebra are the easiest topics for the LLMs that we have evaluated, while Precalculus and Intermediate Algebra appear to be the toughest. For the exact number, check the Table IV in the Appendix.

D. *GraderEval Evaluation*

The `GraderEval` dataset was used to evaluate multiple models for selecting the best grader model. Jaccard-distance is reported.

We present our results in Table II. The best-performing grader model was **GPT-4o**, followed by **Meta-Llama-3.1-70B-Instruct**. These models demonstrated high accuracy in detecting key points, making them reliable choices for the grader role in further evaluations.

E. *TutorEval Evaluation*

Using the **GPT-4o** grader model, we performed evaluations on the `TutorEval` dataset. The evaluation included both Pre-

Table I
ALL DATASETS – ACCURACY

Model	**MGSM_FR (8)**	**MGSM_DE (8)**	**MATH**	**GSM8K**
ethel-70b-magpie	-	-	-	0.009
Llama-3.1-70B	0.836	0.828	0.568	0.948
Smol	-	-	-	0.029
Llama-3.2	0.576	0.604	0.249	0.663

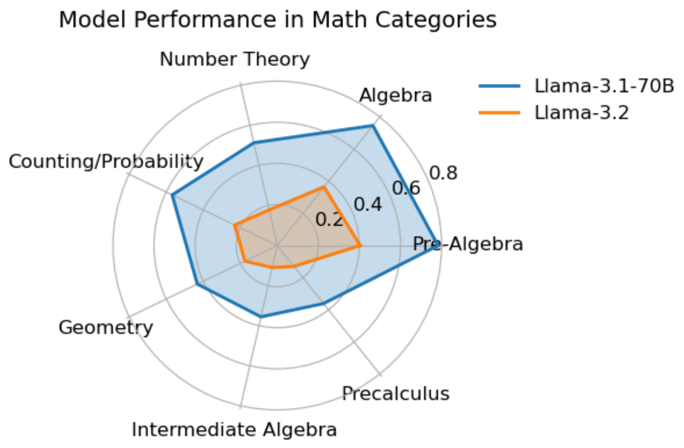

Figure 3. Llama 3.1 and Llama 3.2 compared in all MATH categories

Table II
GRADERS LEADERBOARD

Grader-models	**Position**	**Jaccard-dist**
gpt-4o	1.	0.133
Llama-3.1-70B-Instruct	2.	0.204
Llama-3.3-70B-Instruct	3.	0.222
Llama-3.2	4.	0.401

sentation and Correctness metrics. The results are presented in Table III. Once again, the SWISS AI **ethel-70b-tutorchat** model is outperformed by other models, including significantly smaller ones. Notably, **GPT-4o** emerges as the leading model, with **Llama-3.1-70B** achieving a close second place in performance.

Table III
TUTOREVAL – WITH LLAMA 3.1-70B-INSTRUCT AS GRADER

Grader-models	**Presentation**	**Correctness**
gpt-4o	2.735	2.829
Llama-3.1-70B-Instruct	2.585	2.6
ethel-70b-tutorchat	0.970	0.289
Llama-3.2	2.017	1.6

VI. DISCUSSION

The evaluated models demonstrate strong capabilities in solving problems step-by-step, working across multiple languages, and providing clear explanations. However, there are issues to consider, such as the risk of bias, difficulty in understanding how they make decisions, and the chance that students might rely too much on them instead of developing their skills. The 'Grading the Grader' method adds an important layer by checking not just how well tutor models work but also how fairly and accurately the grading models perform. This ensures the system is reliable and avoids spreading mistakes or unfair judgments. Lastly, the report highlights the need to follow ethical principles, making sure these tools are fair, transparent, and helpful for students, while also being mindful of their environmental impact by improving efficiency.

VII. SUMMARY

This report presented methods for evaluating large language models (LLMs) in an educational context, focusing on their performance across datasets such as GSM8K, MGSM, MATH, and TutorEval. The evaluation pipeline was designed with modularity and flexibility, enabling the assessment of various models and tasks, including multilingual and few-shot learning capabilities. A novel approach, termed 'Grading the Grader,' is introduced to assess the reliability of grading models themselves. Ethical considerations, including fairness, privacy, and sustainability, are highlighted to ensure the responsible deployment of LLMs in educational settings. Additionally, the introduction of the GraderEval dataset provides a valuable tool for advancing research in evaluating grading models effectively. Future work should aim to extend these evaluations by applying more computational resources to complete missing data points for better analysis and comparison across various model classes.

VIII. Ethical risk assessment

The motivation for this project comes from the idea of using LLMs in educational contexts, specifically as potential replacements for teaching assistants. While such applications are scalable and accessible, they also introduce significant ethical risks. Using automated systems to assess and assist people can potentially introduce biases and unfairness.

A. Identified Risks

Risk Description: The primary risk lies in deploying LLMs as TAs or tutors without ensuring their fairness and accuracy. If these systems generate biased or incorrect judgments, they could unfairly impact students.

Impacted Stakeholders:

1) Students: They risk receiving biased evaluations.
2) Teachers: Teachers and institutions may face reputational

Negative Impacts: - Unfair treatment of students based on incomplete or biased assessments. - Lack of trust in educational systems using AI.

Risk Likelihood and Importance: This risk is highly significant, given the promising role of AI in education of the future. The likelihood of occurrence depends on how well AI systems are evaluated before deploying.

B. Evaluation of Risks

We evaluated this risk by: - Measuring model performance using metrics such as correctness and presentation, as well as fairness indicators like Jaccard-distance in the GraderEval dataset.

C. Mitigation Steps

GraderEval Dataset: As a step towards addressing these risks, we introduced the GraderEval dataset, designed to evaluate the ability of models to identify key points accurately and fairly. This is one step to control the bias and fairness of the ”Grader” model. We have also shared the GraderEval dataset with the Swiss AI community so that future researchers can adapt our methodology and build on that.

Barriers: - Limited computational resources restricted the ability to conduct exhaustive evaluations across all potential scenarios.

Additionally, We provide ethical risks and possible mitigations according to the Digital Ethics Canvas in the Appendix IX-C.

IX. Appendix

A. Prompt for imitating answer for `GraderEval`

```
def rephrase_prompt(question: str, key_points: list)
    -> str:
    return f"""
    You are given the following key points: {'\n'.join
        (key_points)}

    Those are the key points to answer the question: {
        question}

    You are NOT allowed to answer the question by
        yourself. Use only the information from key
        points.

    Reformulate the given key points to imitate the
        answer to the question.

    Here is the answer with only provided key points:
    """
```

B. Prompt for grading answer for `GraderEval`

```
def generate_grader_prompt(self, answer: str,
    key_points: List[str]) -> str:
    instruction = """Your task is to decide which of
        key points are present in the answer provided
        and return the letter of the key points that
        are present in the answer. Here are the key
        points:\n""" +"\n".join([f"{chr(65+i)}.␣{kp}"
        for i, kp in enumerate(key_points)])

    instruction+= f"""\nAnd then the answer is: \n {
        answer} \n\n Return the letters separated by
        commas after ####"""

    return instruction
```

C. Digital Ethics Canvas

1) Welfare: **Risk:** Misleading responses, harmful outputs, or inaccurate grading may negatively impact student learning.

Mitigation: Deployed systems should have human-in-the-loop oversight, robust testing, and safeguards to ensure reliable outputs.

2) Fairness: **Risk:** Biases in LLMs can lead to unfair evaluations or unequal accessibility for marginalized groups.

Mitigation: Future work must audit models for bias.

3) Autonomy: **Risk:** Over-reliance on LLMs may reduce student critical thinking and agency.

Mitigation: Clearly communicate model limitations and design systems to support, not replace, learning.

4) Privacy: **Risk:** Sensitive student data may be misused, leaked, or exposed during evaluation.

Mitigation: Deployed system must follow data protection standards (e.g., GDPR), anonymize data, and ensure encryption.

5) Sustainability: **Risk:** High computational costs of LLMs contribute to environmental concerns.

Mitigation: Optimize model efficiency, use carbon-neutral infrastructure, and limit resource-intensive processes.

6) Grading the Grader: **Risk:** Using LLMs to evaluate other LLMs can propagate biases and errors.

Mitigation: Future work must incorporate human-in-the-loop grading as well as “Grading the Grader” to ensure fairness and accuracy.

References

[1] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman, “Training verifiers to solve math word problems,” 2021. [Online]. Available: https://arxiv.org/abs/2110.14168

[2] F. Shi, M. Suzgun, M. Freitag, X. Wang, S. Srivats, S. Vosoughi, H. W. Chung, Y. Tay, S. Ruder, D. Zhou, D. Das, and J. Wei, “Language models are multilingual chain-of-thought reasoners,” 2022. [Online]. Available: https://arxiv.org/abs/2210.03057

[3] D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt, “Measuring mathematical problem solving with the math dataset,” 2021. [Online]. Available: https://arxiv.org/abs/2103.03874

[4] A. Chevalier, J. Geng, A. Wettig, H. Chen, S. Mizera, T. Annala, M. J. Aragon, A. R. Fanlo, S. Frieder, S. Machado, A. Prabhakar, E. Thieu, J. T. Wang, Z. Wang, X. Wu, M. Xia, W. Xia, J. Yu, J.-J. Zhu, Z. J. Ren, S. Arora, and D. Chen, “Language models as science tutors,” 2024. [Online]. Available: https://arxiv.org/abs/2402.11111

Table IV
MATH – ACCURACY

Model \n-shot	Prealgebra	Algebra	Number-theory	Counting Probability	Geometry	Intermediate Algebra	Precalculus
Llama-3.1-70B-Instruct	0.783	0.747	0.513	0.568	0.432	0.356	0.361
Llama-3.2	0.404	0.365	0.178	0.230	0.175	0.110	0.130

Table V
MGSM – ACCURACY – FRENCH AND GERMAN

Model \n-shot	0-shot	2-shot	4-shot	6-shot	8-shot	Language
Llama-3.1-70B-Instruct	0.715	0.824	0.848	0.863	0.836	French
	0.592	0.80	0.811	0.828	0.828	German
Llama-3.2	0.472	0.56	0.58	0.584	0.576	French
	0.224	0.572	0.56	0.536	0.604	German

[5] Ollama, “Ollama: A conversational ai model,” 2024, accessed: 2024-12-16. [Online]. Available: https://github.com/ollama/ollama

[6] A. P. Abhimanyu Dubey, Abhinav Jauhri *et al.*, “The llama 3 herd of models,” https://arxiv.org/abs/2407.21783, 2024, accessed: 2024-12-16.

[7] A. Chevalier, J. Geng, A. Wettig, H. Chen, S. Mizera, T. Annala, M. J. Aragon, A. R. Fanlo, S. Frieder, S. Machado, A. Prabhakar, E. Thieu, J. T. Wang, Z. Wang, X. Wu, M. Xia, W. Jia, J. Yu, J.-J. Zhu, Z. J. Ren, S. Arora, and D. Chen, “Language models as science tutors,” 2024.

[8] M. S. Kamel Charaf, Ivan Pavlov, “ethel-tutor-eval,” https://github.com/swiss-ai/ethel-tutor-eval, 2024.

[9] OpenAI, “Gpt-4: Technical report,” OpenAI, 2024. [Online]. Available: https://openai.com/research/gpt-4

[10] A. Lewkowycz, A. Andreassen, D. Dohan, E. Dyer, H. Michalewski, V. Ramasesh, A. Slone, C. Anil, I. Schlag, T. Gutman-Solo, Y. Wu, B. Neyshabur, G. Gur-Ari, and V. Misra, “Solving quantitative reasoning problems with language models,” 2022. [Online]. Available: https://arxiv.org/abs/2206.14858