Small But Mighty: Achieving High Accuracy with Small Language Models on scientific MCQ answering

BARGHORN Jérémy | 328403 | jeremy.barghorn@epfl.ch
MAIER Sebastian | 327504 | sebastianandreas.maier@epfl.ch
SCHIFFERLI Théo | 326468 | theo.schifferli@epfl.ch
Long-Short-Term-Midgets

Abstract

This paper explores how Small Language Models (SLMs) can be fine-tuned and optimized on consumer hardware to achieve high performance on specific downstream tasks. It dives into the finetuning, Direct Preference Optimization (DPO) training, and Retrieval-Augmented Generation (RAG) applied to the Phi-2 language model to enhance its performance in commonsense reasoning and multiple-choice question answering.

The finetuning process employed a variety of datasets, covering a wide range of instruction types from simple queries to complex language comprehension tasks. To further improve the model's mathematical and reasoning capabilities, additional scientific datasets were incorporated. DPO training was focused on datasets designed to improve concise response generation and structured mathematical dialogue. RAG implementation augmented the model with a robust knowledge base, enabling it to access contextually similar questions and enhance response accuracy and relevance.

The model was evaluated on diverse state-of-the-art benchmarks, including ARC Challenge, GSM8K, MMLU and HellaSwag, measuring performance improvements across various optimizations. The results indicated significant gains in model accuracy and reasoning ability post finetuning. Although the impact of DPO training was relatively modest, it still outperformed the fine-tuned model alone in some cases.

Overall, this report highlights the substantial advancements achieved through methodical finetuning and presents findings on the efficacy of DPO training and RAG in enhancing model performance. The combination of these techniques shows promise in enabling the model to perform well on tasks such as passing an EPFL exam by answering questions accurately.

1 Introduction

Advancements in natural language processing (NLP) have driven significant progress in language understanding tasks. Yet, challenges persist in areas like commonsense reasoning and multiple-choice question answering (MCQ) in scientific domains. These tasks require a nuanced understanding of context and complex reasoning, where current models often fall short, providing inconsistent and contextually inappropriate responses.

The Phi-2 language model aims to address these challenges by providing a non-restricted small language model (SLM) for the research community. This model facilitates exploration of critical safety challenges, such as reducing toxicity, understanding societal biases, and enhancing controllability (Yuanzhi Li). Our research enhances Phi-2's performance through three primary strategies: finetuning, Direct Preference Optimization (DPO) training, and Retrieval-Augmented Generation (RAG), specifically targeting scientific MCQs.

Previous work has shown the effectiveness of finetuning pre-trained models on specific tasks to improve performance. Howard and Ruder (Howard and Ruder, 2018) demonstrated significant improvements in text classification through finetuning language models on task-specific data. Similarly, Ziegler et al. (Ziegler et al., 2020) highlighted the benefits of domain-specific finetuning for various NLP applications.

We developed two primary models for testing these enhancements: **Instruct50k-Orca50k-GSM8k-LoRA-Phi2** and **DPO-M4AI-Instruct50k-Orca50k-GSM8k-LoRA-Phi2**. The finetuning process utilized diverse datasets, encompassing a range of instructional types from simple queries to complex language comprehension tasks. To enhance the model's mathematical reasoning capabilities, additional scientific datasets were incorporated.

DPO training refines the model's response generation, focusing on structured mathematical dialogue. This method, as described by Rafailov et al. (Rafailov et al., 2023a), aims to enhance the clarity, precision, and relevance of model outputs by training on datasets specifically designed for these purposes.

RAG, as introduced by Lewis et al. (Lewis et al., 2020), integrates a sophisticated retrieval mechanism, allowing the model to access and utilize a vast knowledge base. By referencing contextually similar questions and relevant information, RAG significantly improves the model's response accuracy and relevance, particularly for tasks requiring in-depth understanding and contextual awareness.

We rigorously evaluated these optimizations using state-of-the-art benchmarks, including ARC Challenge, GSM8K, MMLU, and HellaSwag. These evaluations measured improvements in accuracy and reasoning ability, providing a comprehensive assessment of the model's performance enhancements. finetuning demonstrated substantial gains in model accuracy and reasoning capability, while DPO training, though showing more modest improvements, still outperformed the fine-tuned model alone in several cases.

In summary, this paper presents a detailed examination of the methodologies employed to enhance the Phi-2 language model. By leveraging finetuning, DPO training, and RAG, our findings offer valuable insights into the efficacy of these approaches in addressing complex language understanding and reasoning tasks. This study highlights the potential of these techniques to enable SLMs to excel in specific tasks, such as passing an EPFL MCQ exam.

2 Related Work

2.1 LoRA and QLoRA

To fine-tune a Small Language Model (SLM) on consumer hardware efficiently and quickly, several techniques were employed. This approach follows the three main directives of the QLoRA (Dettmers et al., 2023) paper:

- quantizing the model in Normal-Float-4 and doing computations in bfloat16,
- finetuning the model with LoRA adapters instead of full weights (LoRA-**r** parameter and LoRA **alpha** both set to 32 in order to match the number of attention heads)
- Paged optimizers utilizing unified NVIDIA memory were employed to achieve automatic page transfers between the CPU and GPU, effectively avoiding significant memory spikes during the optimizer's update step for long inputs. This innovation is particularly important for consumer GPUs with limited memory, as these spikes can be proportionally more drastic.

With this technique the model achieved finetuning on 50k samples with a learning rate of 2e-5, weight updates every 16 samples in about 6 hours per epoch on a RTX 4070 Ti 16Go.

2.2 DPO

To integrate human preferences into our model, we utilized the Direct Preference Optimization (DPO) technique, inspired by the work of Rafailov et al. as previously cited (Rafailov et al., 2023b). Their research introduced an effective method for finetuning models to align with human preferences on two outputs. Before DPO, Reinforcement Learning from Human Feedback (RLHF), introduced by OpenAI (Ziegler et al., 2020), was the predominant method for calibrating models to human preferences. RLHF involves using a reinforcement learning (RL) model to assign scores to outputs, effectively acting as a loss function for the language model to learn from. However, RLHF can be complex as it necessitates training a separate RL model from scratch to learn human preferences. In contrast, DPO simplifies this process by leveraging the same model to evaluate outputs against human preferences, eliminating the need for a separate RL model. This approach, highlighted in the DPO paper, has proven effective in applying human preference adjustments to our chosen model without relying on RL.

2.3 RAG

To enhance model performance in MCQ answering the RAG method was used. This significantly improves the model's ability to retrieve context by building a robust knowledge base. The theoretical foundation for this approach was derived from the RAG paper by Lewis et al. (Lewis et al., 2020), while the practical implementation was facilitated using the Llama-Index library (Liu, 2022). Specifically, the indexing and retrieval methods were important in constructing and utilizing the knowledge base effectively. The **BAAI/bge-small-**

en-v1.5 model (Xiao et al., 2023) was used to generate embeddings and index the knowledge base. This model was chosen for its small size, which ensures that the total parameter count remains below the 2.9B limit, and for its high speed. Additionally, it is currently one of the top-performing models on Hugging Face for the selected task.

2.4 Evaluation and benchmarks

To evaluate the model's performance across different training strategies, the lm-evaluation-harness tool(Gao et al., 2023) was employed. This tool, currently the backend for the Hugging Face Open LLM Leaderboard, is used extensively in academic literature, demonstrating its reliability.

Initial benchmarks were conducted on the base **microsoft/phi-2** model to verify the reproducibility of its performance as reported online. The **ARC-Challenge** was used for commonsense reasoning, **HellaSwag** and **MMLU** for language understanding, and **GSM8K** for math and coding. The results were consistent with those reported online, confirming the model's baseline performance.

Subsequent benchmarks were performed on each of the ten created models. From these, the two best-performing models were selected. Two-shot prompting was primarily used for the evaluations. The **lm-evaluation-harness** tool provided valuable insights into the computation of benchmarks and scores, which were then incorporated into the final model for MCQ answering.

3 Approach

In this section the pipeline starting from the base **microsoft/phi-2** model to it's end result is explained in details. The datasets used, the finetuning pipeline, the DPO optimisation and the RAG augmentation will be discussed in order to show how a surprisly strong SLM can further be improved in order to achieve state of the art performance on scientific tasks.

3.1 M1 preference data and datasets

To efficiently collect data from the set of questions provided by EPFL courses, a small UI 5 was developed to prompt the GPT-wrapper. This interface allowed for fine-grained query customization for each question. Users could add a system prompt, a prompt to be prepended, and a prompt to be appended to the question 6, enabling detailed and task-specific queries. Additionally, sliders were provided to control parameters such as temperature, top-p, top-k, presence penalty, and frequency penalty for each question 7. This functionality allowed for the observation of how each query was impacted, facilitating the creation of high-quality preference pairs. The overall preference was determined by selecting the clearest, best-structured, and correctly formatted answers, free from ambiguities, contradictions, or factual errors.

The other datasets used in this study will be discussed in detail in subsequent sections. The main criteria for dataset selection were the use of high-quality, open-source datasets with scientific content and instructional data.

3.2 Finetuning

The model was fine-tuned on multiple datasets using the QLoRA technique (see Section 2.1). This process created several adapters, each with approximately 235 million parameters. In total, five adapters were developed using the following datasets: SciQ, tatsu-lab/alpaca, Open-Orca/OpenOrca, Grade School Math 8K and Open-Book QA discussed later 4.1.

These adapters were merged in various combinations to stay under the 2.9 billion parameter limit of the chosen category, resulting in different fine-tuned model variants. Over ten models were created and benchmarked using the same test suite to identify the best combination.

The test suite included automatic evaluation metrics such as BLEU (Saadany and Orasan, 2021), ROUGE (Lin, 2004), BERTScore (Zhang et al., 2020), and COMET (Rei et al., 2020), as well as human evaluation 8. Additionally, a suite of benchmarks was used, including ARC Challenge (Clark et al., 2018), HellaSwag (Zellers et al., 2019), MMLU (Hendrycks et al., 2021), GSM8K, and SciQ. Ultimately, only the highest-performing model was retained for further improvement through the DPO process. This model contained the merged adapters of the alpaca instruction dataset, the orca dataset and the

3.3 DPO and open question answering

Once the best fine-tuned model was selected, Direct Preference Optimization (DPO) runs were applied to it (see Section 2.2). Similar to the finetuning process, three adapters were created using the following datasets: 1. **M1 class dataset**: Containing preference pairs for EPFL questions. 2. **Intel/orca_dpo_pairs**: A dataset designed to train the model to be a better chatbot. 3. **M4-ai/prm_dpo_pairs_cleaned**: A dataset aimed at keeping responses concise. These three adapters were then merged to create fine-tuned models with preference optimization. These models were benchmarked and evaluated using the same methodology as for the finetuning. The M1 dataset was discarded due to its ambiguity and lack of improvement in model performance. The Orca dataset was also discarded because it caused the model to underperform in commonsense reasoning. Consequently, the only adapter retained was the one fine-tuned with the M4AI preference dataset. To achieve optimal performance, the DPO adapter was merged with a weight of 1, while all subsequent adapters from the finetuning phase were merged with a weight of 0.8. This weighting was intended to prioritize the DPO optimizations. At this point both of our models outperformed the base phi-2 version.

3.4 RAG and MCQ answering

For this step of the pipeline the best models from the previous steps were selected in order to first make them strong in answering MCQ questions and then augment their reasoning and answering skills by providing helpful context with RAG. In order to augment the model by adding meaningful context multiple scientific centred indices were created : 1. **stem_qa_m1_index**: An index of open book QA questions along with the M1 preference dataset. (300Mo) 2. **stem_open_question_index**: An index containing open questions and MCQs in math and coding. (500Mo) 3. **stem_all_index**: An index resulting in the merge of all the previous indicies. (1GB). The indices were created using the Llama-Index library, which facilitated an easy and cost-efficient implementation (Liu, 2022). To enable the model to predict letters for multiple-choice questions (MCQs), various strategies were tested, including multiple prompt templates, few-shot strategies, regular expression matching, and cosine similarity functions. However, these approaches yielded poor accuracy compared to benchmarks run on MMLU using the lm-evaluation-harness tool. To achieve the best performance, the prediction step was inspired by the methodology used in large dataset benchmarks (see Section 2.4). For each question, the four possible answers (A, B, C, or D) were concatenated with the question. The log probabilities of each concatenated sentence were computed, and the sentence with the highest probability was selected as the correct answer. This approach resulted in an accuracy of over 48% for the fine-tuned + DPO model and over 53% for the largest index in the Retrieval-Augmented Generation (RAG) method. This method is significantly faster than generating answers but has the downside of not allowing the model to build its own chain of thought, which could potentially lead to more reliable and well-constructed outputs 9.

4 Experiments

4.1 Data

In this section, the datasets used for finetuning, DPO training, and RAG knowledge base creation will be detailed. Each dataset will be briefly explained, highlighting the motivation for its selection. Additionally, the format of one template sentence for each dataset will be provided in the Appendix for reuse A.2. For the finetuning and DPO tasks, the datasets were preprocessed, and the following prompt template was applied: Instruct: prompt Output: chosen/rejected/answer. (The appended letters after the dataset descriptions indicate the tasks for which they were used: F for finetuning, D for DPO, and R for RAG.)

• SciQ : A scientific multiple-choice question-answering dataset used to test the finetuning capacities of the model. *F, R* • tatsu-lab/alpaca : An instruction dataset selected for improving the model's ability to follow and execute concise instructions. *F* • Open-Orca/OpenOrca : A subset of the Orca dataset used to provide a variety of complex and nuanced queries, aiding in the model's comprehension of longer queries and text summarizing tasks. *F* • Grade School Math 8K : A coding and math dataset aimed at enhancing the model's capabilities in mathematical problem-solving and logical reasoning. *F, D* • OpenBook QA : An open-question completion dataset selected to improve the model's ability to provide comprehensive and contextually relevant answers. *F, R* • M1 Class Dataset: Contains preference pairs for EPFL questions, focusing on improving the model's performance in an academic setting. *D, R* • intel/orca_dpo_pairs : A dataset designed to train the model to function as a better chatbot, enhancing conversational abilities. *D* • M4-ai/prm_dpo_pairs_cleaned : A dataset aimed at ensuring responses are concise and to the point. *D* • CodeAlpaca-20k : A dataset containing coding-related queries, chosen to enhance the model's programming and debugging skills in the form of short open questions. *R* • qwedsacf/competition_math : A competitive math dataset aimed at improving the model's performance in high-stakes mathematical problem-solving with longer answers. *R* • allenai/math_qa : A MCQ dataset focusing on math question answering, selected to boost the model's capabilities in mathematical reasoning and solution generation. *R*

4.2 Evaluation Methods

To evaluate the performance and various aspects of the different versions of the model, multiple evaluation methods were employed. These evaluations provided meaningful metrics that helped in understanding the model's capabilities comprehensively.

4.2.1 Metric-based evaluation

• **BLEU (Bilingual Evaluation Understudy)** BLEU-1: Measures the precision of unigrams (1-gram). BLEU-4: Measures the precision of up to 4-grams. **Reason**: BLEU is a popular metric for evaluating the quality of text generated by a model by comparing it to one or more reference texts. BLEU-1 captures the precision of individual words, while BLEU-4 captures longer sequences, providing a more comprehensive evaluation of fluency and coherence. • **ROUGE (Recall-Oriented Understudy for Gisting Evaluation)** ROUGE-1: Measures the overlap of unigrams between the generated and reference texts. ROUGE-2: Measures the overlap of bigrams. ROUGE-L: Measures the longest common subsequence between the generated and reference texts. **Reason**: ROUGE is widely used in summarization tasks. ROUGE-1 and ROUGE-2 evaluate the n-gram overlap, capturing both recall and precision aspects. ROUGE-L considers sequence overlap, which is crucial for evaluating the content and structure of summaries.

To summarize, the fore-mentioned metrics were chosen because they are standard metrics for evaluating the quality of text generation and summarization models, providing insights into different aspects of model performance, such as precision, recall, and fluency.

4.2.2 Model-based Evaluation

• **BERTScore:** Uses BERT embeddings to evaluate the similarity between the generated text and reference text at a semantic level. **Reason**: BERTScore captures the semantic similarity between texts, making it a robust metric for evaluating tasks like translation, summarization, and text generation, where semantic understanding is crucial. • **COMET (Commonsense Transformers):** Uses transformer-based models to generate and evaluate commonsense knowledge. **Reason**: COMET provides a model-based evaluation of the generated text by assessing its commonsense plausibility and coherence, which is essential for tasks that require understanding and generating human-like reasoning and inference.

BERTScore and COMET are chosen for their ability to evaluate the semantic and commonsense aspects of the generated text, providing a deeper understanding of model performance beyond surface-level n-gram matching.

4.2.3 Human evaluation

The model was evaluated on 50 questions and compared to the baseline to get an accuracy. The data was drawn from a random sampling of 4 different datasets. For each question, a preferred output was selected from the two models responses 8.

4.2.4 RAG metrics

During the testing of the RAG pipeline, three evaluation metrics were defined to search for improvements of the model. Firstly, time was a problem in the generation. Secondly, the size of the documents stored in the index was analysed to provide insights into the performance of the model. Thirdly, the Mean Reciprocal Rank (MRR) and the Hit Rate (HR) were calculated to evaluate how well the retriever model performs. Finally, the mechanic to extract the final answer was worked on to get better predictions from the generation of the model.

4.3 Baselines

Our proposed models were compared with the baseline Phi-2 model which has proven to be already quite powerful in mathematical and computer science reasoning.

4.4 Experimental details

To obtain a single letter output, several methods were attempted: zero-shot prompting by instructing the model to answer with a single letter, one-shot prompting with an example output, though this led to the model repeatedly outputting the same letter (e.g., always 'A' if 'A' was the example), and two-shot and three-shot prompting, which exhibited the same issue. Four-shot prompting with all possible outputs (A, B, C, or D) was also ineffective. These approaches yielded a single letter in approximately 80% of cases but were not flawless. Combining these methods with cosine similarity consistently produced a single letter answer, but the accuracy remained below 20%, which is worse than random guessing. Ultimately, a different approach was adopted, involving the computation of log probabilities for a set of four-element template sentences, such as ['A', 'B', 'C', 'D'] or ['The correct answer is A', 'The correct answer is B', 'The correct answer is C', 'The correct answer is D']. To assess these templates, two strategies were tested: 1. selecting the template with the highest log probabilities 2. taking the majority vote among five templates This technique achieved an accuracy of 48%, which is now significantly better than random. It is important to note that the most advanced approach tested was to let the model generate for some tokens (approx. 100) and then compute the log probabilities. These results are shown in table 1

For RAG some metrics were computed in order to measure the performance of the index, the first experiment measures the time it takes to generate an answer by increasing the prompt and the context length linearly. Then scores were compared for the 3 different indices that were created. Finally, the retriever model was evaluated with respect to the MRR and the HR, calculated on a top k = 2. To conduct the experiment, an LLM with better capabilities (**meta-llama/Meta-Llama-3-8B**) was required to reformulate the text found in the nodes of the index in order to create sentences that are close in the embedding dimension. Once the reformulations are generated, the goal is for the retriever to fetch the node for each created sentence and then see if it matches the correct context.

4.5 Results

The benchmarks results on the three models can be observed in Figure 1 and on MMLU in Figure 2.

From the results of the experiments of RAG, it is visible that some improvements have been made. To begin with the time of inference of the RAG model, from the Figure 3.

Then, the impact of the size of the index was measured in table 1. All the models were tested with and without RAG on a MCQA benchmark, with varying indices and inference methods.

After, the MRR and the HR were computed on the three indices with varying contents. The results are given in the table 4. The results for the QA index is relevant in the analysis of the performance. Furthermore, with top k = 2, it can be observed that either the retriever finds the correct answer and places it at the top, or it does not appear in the top 2.

5 Analysis

The M1 dataset, generated by the class, posed challenges in applying human preference using DPO. Consensus revealed significant variability in preferences among students, such as favoring either lengthy or concise explanations, bullet points or paragraphs, and other factors. Consequently, this variability caused the gradient to oscillate during each batch process. To address this issue, the number of forward passes before each backpropagation step was increased to approximately 50. While this approach resulted in a gradual reduction in evaluation loss, it did not achieve a satisfactory level. As a result, we decided to discard the dataset and opted for more structured datasets containing human preference pairs.

We tested different types and sizes of indices to optimize performance on benchmarks based on their content. From Table 1, comparing the first two indices shows that the smaller 'stem_qa_m1_index' (300MB) outperforms the larger 'stem_open_question_index' (500MB). This is because the former contains more relevant content for MCQA benchmarks, specifically focusing on MCQA with answers. Combining these two indices yields better scores than using either index individually. This improvement suggests that having more documents to choose from increases the likelihood of finding embeddings closer to the query prompt. A denser embedding space enhances the chances of retrieving top k = 2 results that closely match the query embedding. Hence, a larger index size is advantageous.

Another point to be made from the table 1, is that the simply taking the log probabilities, as explained in the section 4.4, outperforms making the model generate a response and then taking the log probabilities. The most probable reason is that after some meticulous observations on the output of the rag model, the model would have two examples from the prompt and will continue giving further question it comes up with itself (hallucination). These extra questions might influence of the log proba-

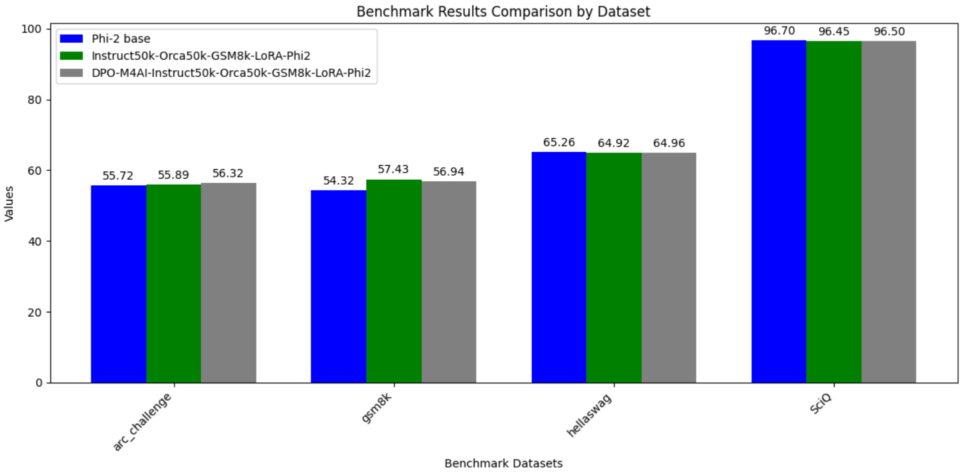


Figure 1: Evaluation results of different models across selected benchmark datasets

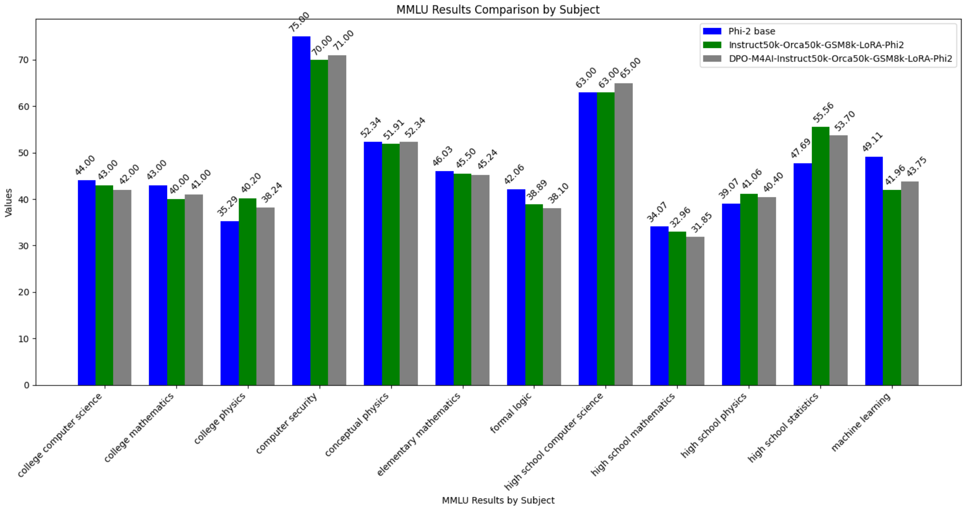


Figure 2: Performance of different models on a range of MMLU subjects

		Phi-2 base	Instruct50k-Orca50k-GSM8k-LoRA-Phi2	DPO-M4AI-Instruct50k-Orca50k-GSM8k-LoRA-Phi2
	Base MCQA performance	49.4	48.6	48.6
stem_qa_m1_index (300Mo)	Generation and logprobs	46.9	42.1	42.8
	Logprobs only	53.0	50.6	50.3
stem_open_question_index (500Mo)	Generation and logprobs	46.3	48.9	48.6
	Logprobs only	52.8	53.4	53.4
stem_all_index (1GB)	Generation and logprobs	47.4	46.6	46.3
	Logprobs only	**54.2**	**53.7**	**53.7**

Table 1: Table showing the accuracy of our model on MCQA only and then on MCQA with two different generation methods and on multiple indices.

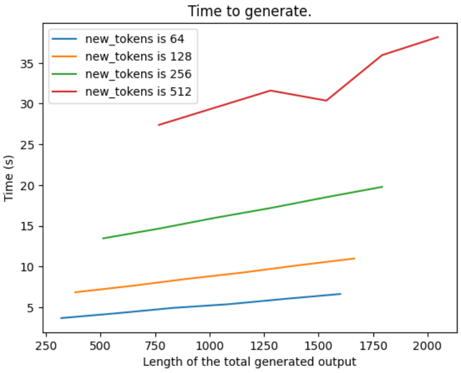


Figure 3: Time taken to generate tokens with respect to the total length of context and different output tokens

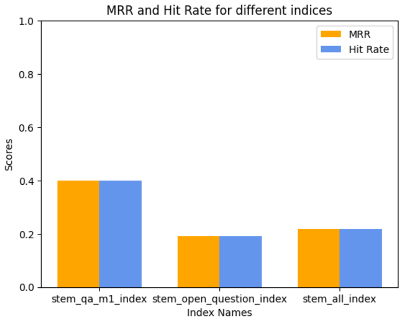


Figure 4: MRR and HR with different Indices

bilities at the end, since the output gives different questions with different final answers. Thus, simply taking the log probabilities of the documents retrieved and the question to answer is more likely to give the same answer. This discovery is slightly disappointing, because the LLMs perform better when using Chain of Thought, or when they are asked to go through the problem step by step.

6 Ethical considerations

The project prioritizes privacy, data protection, fairness, transparency, and accountability, adhering to regulations like GDPR to prevent unauthorized access or breaches. To prevent misuse, such as cheating, measures like activity monitoring, policy development, and engagement with educational institutions will be implemented.

Ethical considerations include ensuring model's accessibility for users who communicate using signed language, highlighted in the guest lecture on May 2nd, where the need for inclusivity and accessibility for users who communicate using signed language was emphacized. Steps to support this include:

An important aspect of our ethical considerations is the model's accessibility, including its ability to interact with users in signed language. This perspective, highlighted in the guest lecture on May 2nd, emphasizes the need for inclusivity and accessibility for users who communicate using signed language. We will take the following steps to adapt our language model to support signed language interactions: 1. **Integration with Sign Language Recognition Systems:** Collaborate with experts in sign language recognition technology to integrate these systems with our language model. 2. **Curation of a Sign Language Dataset:** Compile a diverse dataset of signed language interactions, encompassing various sign languages and contexts to ensure broad applicability and accuracy. 3. **Training and finetuning**: Use the sign language dataset to fine-tune our language model, adapting existing algorithms to handle the unique structure and grammar of sign languages. 4. **User Interface and Interaction Design:** Develop a user interface that supports seamless interaction between users and the model, including video input/output capabilities for real-time translation of sign language. 5. **Accessibility and Inclusivity Testing:** Conducting extensive testing to ensure effectiveness and user-friendliness.

Finally, By adhering to these aforementioned ethical considerations, we aim to develop a responsible AI system that upholds the highest standards of ethics and integrity. Our commitment to these principles will help ensure that our project benefits all involved parties and respects individual rights.

7 Conclusion

In conclusion, the Phi-2 model performances were enhanced for scientific MCQ answering. By leveraging finetuning, Direct Preference Optimization (DPO), and Retrieval-Augmented Generation (RAG), significant improvements in model performance were achieved across various benchmarks. finetuning provided the most substantial boost, while DPO and RAG further enhanced accuracy and relevance. These methods demonstrate that targeted optimizations can enable SLMs to compete against much bigger models in complex tasks. Future work could explore more diverse datasets and optimization techniques to expand these findings.

References

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2023. A framework for few-shot language model evaluation.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:9459–9474.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Jerry Liu. 2022. LlamaIndex.

Rafael Rafailov, Sergey Serebryakov, Linxi Fan, Yuhuai Wu, Ofir Press, Mikel Artetxe, and Marc'Aurelio Ranzato. 2023a. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2023b. Direct preference optimization: Your language model is secretly a reward model.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. Comet: A neural framework for mt evaluation.

Hadeel Saadany and Constantin Orasan. 2021. BLEU, METEOR, BERTScore: Evaluation of metrics performance in assessing critical translation errors in sentiment-oriented text. In *Proceedings of the Translation and Interpreting Technology Online Conference*, pages 48–56, Held Online. INCOMA Ltd.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-pack: Packaged resources to advance general chinese embedding.

Ronen Eldan Allie Del Giorno Suriya Gunasekar Yin Tat Lee Yuanzhi Li, Sébastien Bubeck. Phi 2. Available online.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2020. Fine-tuning language models from human preferences.

A Appendix

A.1 Team contribution

BARGHORN Jérémy : M1 : UI / website for the preference collection M2 : Finetuning the models and create the adapters, merge the adapters, benchmark the models on the lm-eval, quantize the models so that finetuning works on a consumer GPU, hyperparameters tuning for LoRA, UI for human preferences, implementation of the model_dpo.py M3 : RAG implementation and tests, benchmarking the models on RAG, implementation of the logprobs for MCQA, implementation of the model_dpo.py, UI for the RAG models
MAIER Sebastian : M2: DPO implementatiom, DPO hyperparameters tuning and dataset testing, implementation of log probs scores and forward pass of model_dpo.py M3: RAG index/Vector database creation, testing generation times, testing hit rates and mrr on the retriever model with different sizes of indices, regenerating M1 dataset for more shorter and concise answers on correct answers for RAG with the GPT wrapper, checking length of token lists in each node of index, prompt engineering for RAG formatting
SCHIFFERLI Théo : M1: report writing M2: Data Processing, Evaluation pipeline implementation, parameters tuning for evaluation, data post-processing, report writing M3: MCQ model output research : Prompt Engineering, few shot prompting, cosine similarity, log probabilities, answer templates research and tuning, testing model configurations, report writing

A.2 Datasets format

SciQ
Instruct : What phenomenon makes global winds blow northeast to southwest or the reverse in the northern hemisphere and northwest to southeast or the reverse in the southern hemisphere?
A. coriolis effect
B. muon effect
C. tropical effect
D. centrifugal effect

Output : coriolis effect

tatsu-lab/alpaca
Instruct : Create a classification task by clustering the given list of items. Apples, oranges, bananas, strawberries, pineapples

Output : Class 1: Apples, Oranges Class 2: Bananas, Strawberries Class 3: Pineapples

Open-Orca/OpenOrca
Instruct : Write a question about the following article: Coming off their home win over the Buccaneers, the Packers flew to Ford Field for a Week 12 Thanksgiving duel with their NFC North foe, the Detroit Lions. After a scoreless first quarter, Green Bay delivered the game's opening punch in the second quarter with quarterback Aaron Rodgers finding wide receiver Greg Jennings on a 3-yard touchdown pass. The Packers added to their lead in the third quarter with a 1-yard touchdown run from fullback John Kuhn, followed by Rodgers connecting with wide receiver James Jones on a 65-yard touchdown pass and a 35-yard field goal from kicker Mason Crosby. The Lions answered in the fourth quarter with a 16-yard touchdown run by running back Keiland Williams and a two-point conversion pass from quarterback Matthew Stafford to wide receiver Titus Young), yet Green Bay pulled away with Crosby nailing a 32-yard field goal. Detroit closed out the game with Stafford completing a 3-yard touchdown pass to wide receiver Calvin Johnson. With the win, the Packers acquired their first 11-0 start in franchise history, beating the 1962 team which started 10-0 and finished 14-1 including postseason play. Rodgers (22/32 for 307 yards, 2 TDs) was named NFL on FOX's 2011 Galloping Gobbler Award Winner. Question about the article:

Output : What happened during the Thanksgiving football game between the Packers and the Lions, and what record did the Packers achieve with their win?

Grade School Math 8K

Instruct : Natalia sold clips to 48 of her friends in April, and then she sold half as many clips in May. How many clips did Natalia sell altogether in April and May?

Output : Natalia sold 48/2 = «48/2=24»24 clips in May. Natalia sold 48+24 = «48+24=72»72 clips altogether in April and May.72

OpenBook QA

Instruct : The sun is responsible for
A. puppies learning new tricks
B. children growing up and getting old
C. flowers wilting in a vase
D. plants sprouting, blooming and wilting

Output : D

---

intel/orca_dpo_pairs

Instruct : You will be given a definition of a task first, then some input of the task. This task is about using the specified sentence and converting the sentence to Resource Description Framework (RDF) triplets of the form (subject, predicate object). The RDF triplets generated must be such that the triplets accurately capture the structure and semantics of the input sentence. The input is a sentence and the output is a list of triplets of the form [subject, predicate, object] that capture the relationships present in the sentence. When a sentence has more than 1 RDF triplet possible, the output must contain all of them. AFC Ajax (amateurs)'s ground is Sportpark De Toekomst where Ajax Youth Academy also play. Output:

Chosen : [["AFC Ajax (amateurs)", "has ground", "Sportpark De Toekomst"], ["Ajax Youth Academy", "plays at", "Sportpark De Toekomst"]]

Rejected : Sure, I'd be happy to help! Here are the RDF triplets for the input sentence: [AFC Ajax (amateurs), hasGround, Sportpark De Toekomst] [Ajax Youth Academy, playsAt, Sportpark De Toekomst] Explanation: * AFC Ajax (amateurs) is the subject of the first triplet, and hasGround is the predicate that describes the relationship between AFC Ajax (amateurs) and Sportpark De Toekomst. * Ajax Youth Academy is the subject of the second triplet, and playsAt is the predicate that describes the relationship between Ajax Youth Academy and Sportpark De Toekomst. Note that there may be other possible RDF triplets that could be derived from the input sentence, but the above triplets capture the main relationships present in the sentence.

M4-ai/prm_dpo_pairs_cleaned

Instruct : Find the units digit of the following within the indicated number base: $52_7 + 62_7$

Chosen : To find the units digit of a sum, I can just add the units digits of the addends and then convert to the base if needed. The units digits of 52_7 and 62_7 are both 2_7.So the units digit of the sum is $2_7 + 2_7$.In base 7, adding two 2's gives 4, so the units digit is 4_7. No conversion is needed, since the base is already 7. Therefore, the units digit of $52_7 + 62_7$ is 4_7. Answer4_7

Rejected : Since $2_7 + 2_7 = 4_7$, the units digit is 4.

CodeAlpaca-20k

Instruct : Create an array of length 5 which contains all even numbers between 1 and 10.

Output : arr = [2, 4, 6, 8, 10]

---

qwedsacf/competition_math

Instruct : Sam is hired for a 20-day period. On days that he works, he earns \$60. For each day that he does not work, \$30 is subtracted from his earnings. At the end of the 20-day period, he received \$660.

How many days did he not work?

Output : Call x the number of days Sam works and y the number of days he does not. We can set up the following system of equations to represent the given information:

$$
\begin{aligned}
x + y &= 20 \\
60x - 30y &= 660
\end{aligned}
$$

The first equation represents the total number of days Sam works, and the second equation represents his total profit. Solving for x in the first equation yields $x = 20 - y$. Substituting into the second equation gives $60(20 - y) - 30y = 660$. Canceling a factor of 10 and multiplying out gives $120 - 6y - 3y = 66$. This simplifies to $-9y = -54$, or $y = 6$. Thus, Sam did not work for $\boxed{6}$ days.

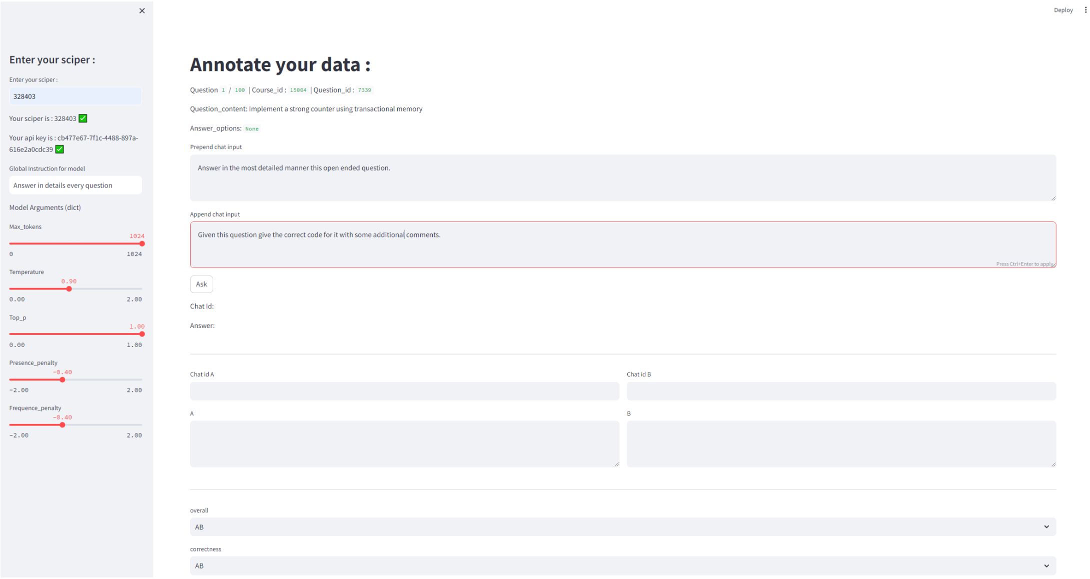
---

allenai/math_qa

Instruct : a multiple choice test consists of 4 questions , and each question has 5 answer choices . in how many r ways can the test be completed if every question is unanswered ? a) 24 , b) 120 , c) 625 , d) 720 , e) 1024

Output : c

A.3 Screenshots



Enter your sciper :

Enter your sciper :

328403

Your sciper is : 328403 ✅

Your api key is : cb477e67-7f1c-4488-897a-616e2a0cdc39 ✅

Global Instruction for model

Answer in details every question

Model Arguments (dict)

Max_tokens

1024

0 1024

Temperature

0.90

0.00 2.00

Top_p

1.00

0.00 1.00

Presence_penalty

-0.40

-2.00 2.00

Frequence_penalty

-0.40

-2.00 2.00

Annotate your data :

Question 1 / 100 | Course_id : 15004 | Question_id : 7339

Question_content: Implement a strong counter using transactional memory

Answer_options: None

Prepend chat input

Answer in the most detailed manner this open ended question.

Append chat input

Given this question give the correct code for it with some additional comments.

Press Ctrl+Enter to apply

Chat Id:

Answer:

Chat id A

Chat id B

A

B

overall

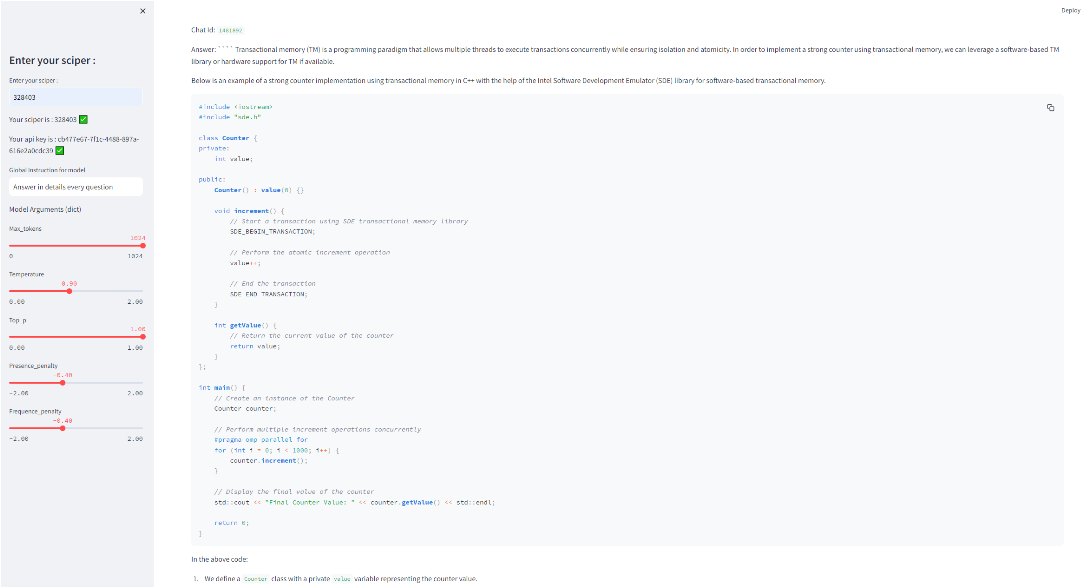
AB

correctness

AB

Figure 5: Screenshot showcasing the main view of the interface used to query the model wrapper and create the preference pairs

A.4 Tables



Enter your sciper :

Enter your sciper :

328403

Your sciper is : 328403 ✅

Your api key is : cb477e67-7f1c-4488-897a-616e2a0cdc39 ✅

Global Instruction for model

Answer in details every question

Model Arguments (dict)

Max_tokens
1024
0 1024

Temperature
0.90
0.00 2.00

Top_p
1.00
0.00 1.00

Presence_penalty
-0.40
-2.00 2.00

Frequence_penalty
-0.40
-2.00 2.00

Chat Id: 1481892

Answer: ```` Transactional memory (TM) is a programming paradigm that allows multiple threads to execute transactions concurrently while ensuring isolation and atomicity. In order to implement a strong counter using transactional memory, we can leverage a software-based TM library or hardware support for TM if available.

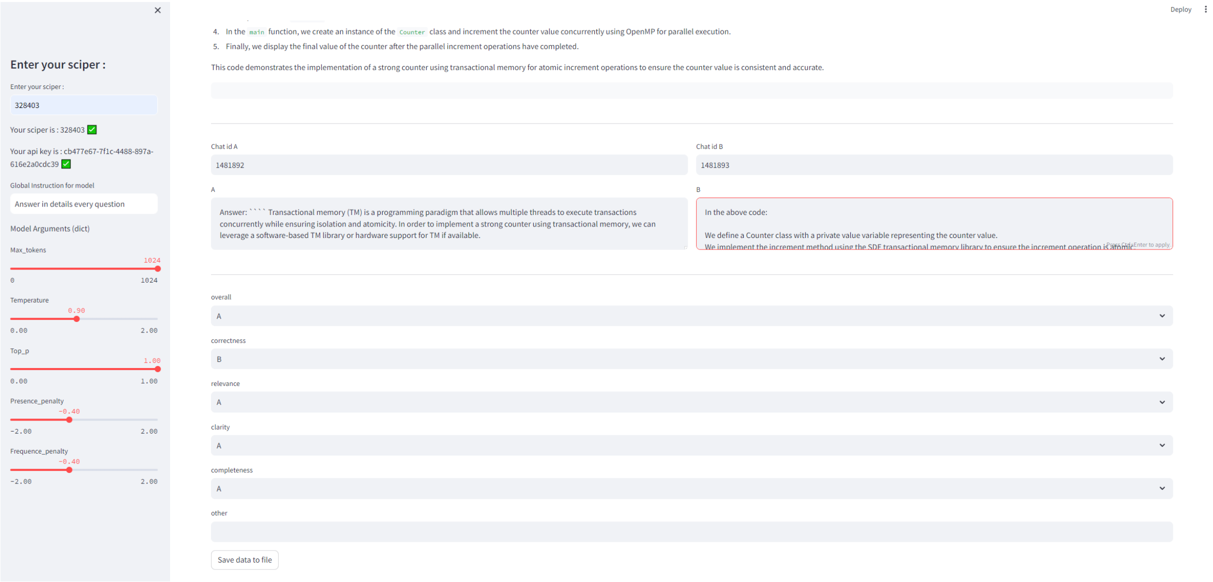
Below is an example of a strong counter implementation using transactional memory in C++ with the help of the Intel Software Development Emulator (SDE) library for software-based transactional memory.

```cpp
#include <iostream>
#include "sde.h"

class Counter {
private:
    int value;

public:
    Counter() : value(0) {}

    void increment() {
        // Start a transaction using SDE transactional memory library
        SDE_BEGIN_TRANSACTION;

        // Perform the atomic increment operation
        value++;

        // End the transaction
        SDE_END_TRANSACTION;
    }

    int getValue() {
        // Return the current value of the counter
        return value;
    }
};

int main() {
    // Create an instance of the Counter
    Counter counter;

    // Perform multiple increment operations concurrently
    #pragma omp parallel for
    for (int i = 0; i < 1000; i++) {
        counter.increment();
    }

    // Display the final value of the counter
    std::cout << "Final Counter Value: " << counter.getValue() << std::endl;

    return 0;
}
```

In the above code:

1. We define a `Counter` class with a private `value` variable representing the counter value.

Figure 6: Screenshot showcasing the correctly formatted output of a response, allowing for easier analysis of the results in a human-readable format.



Enter your sciper :

Enter your sciper :

328403

Your sciper is : 328403 ✅

Your api key is : cb477e67-7f1c-4488-897a-616e2a0cdc39 ✅

Global Instruction for model

Answer in details every question

Model Arguments (dict)

Max_tokens
1024
0 1024

Temperature
0.90
0.00 2.00

Top_p
1.00
0.00 1.00

Presence_penalty
-0.40
-2.00 2.00

Frequence_penalty
-0.40
-2.00 2.00

4. In the `main` function, we create an instance of the `Counter` class and increment the counter value concurrently using OpenMP for parallel execution.
5. Finally, we display the final value of the counter after the parallel increment operations have completed.

This code demonstrates the implementation of a strong counter using transactional memory for atomic increment operations to ensure the counter value is consistent and accurate.

Chat id A
1481892

Chat id B
1481893

A

Answer: ```` Transactional memory (TM) is a programming paradigm that allows multiple threads to execute transactions concurrently while ensuring isolation and atomicity. In order to implement a strong counter using transactional memory, we can leverage a software-based TM library or hardware support for TM if available.

B

In the above code:

We define a Counter class with a private value variable representing the counter value.
We implement the increment method using the SDE transactional memory library to ensure the increment operation is atomic.

overall
A

correctness
B

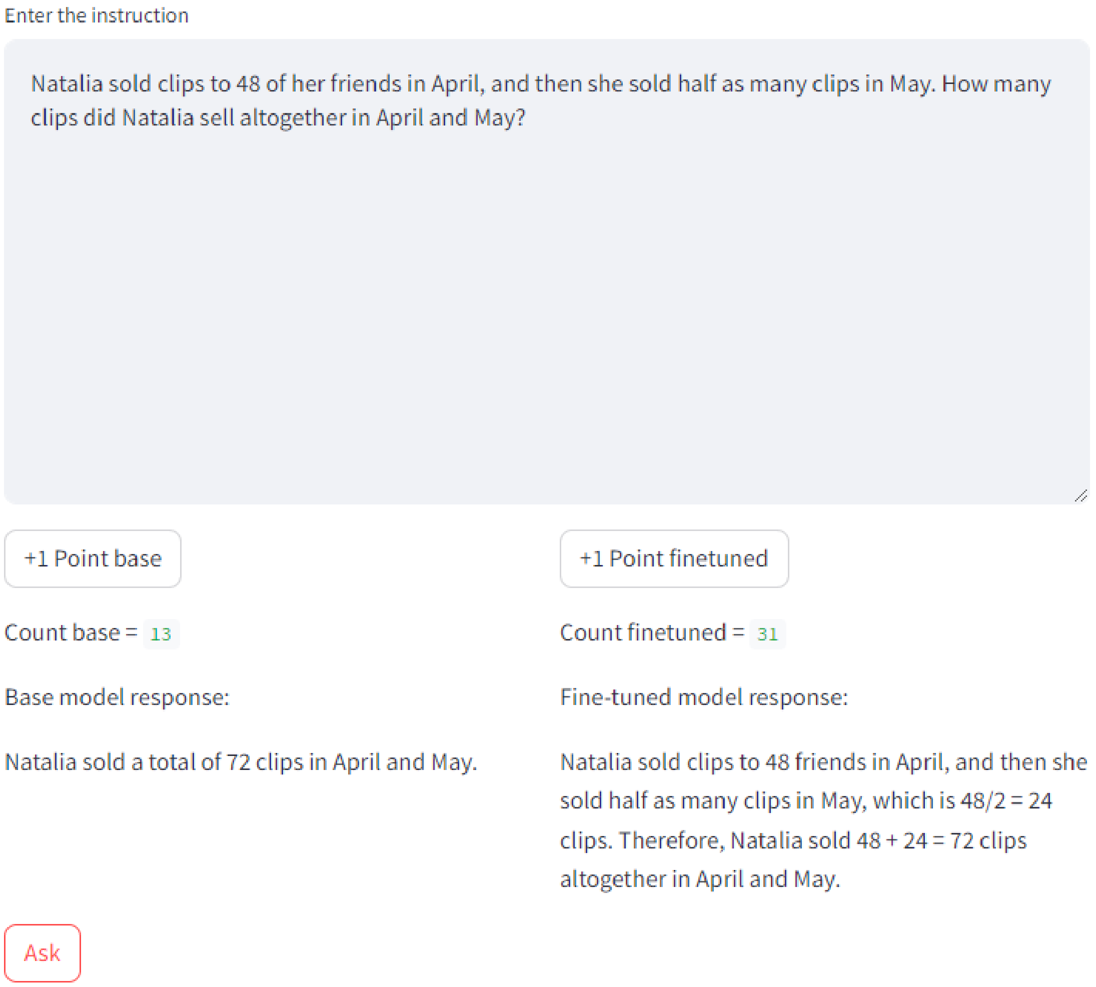
relevance
A

clarity
A

completeness
A

other

Figure 7: Screenshot showcasing the sliders for the model parameters, along with the fields that evaluators need to complete for data annotation.



Enter the instruction

Natalia sold clips to 48 of her friends in April, and then she sold half as many clips in May. How many clips did Natalia sell altogether in April and May?

Count base = 13

Count finetuned = 31

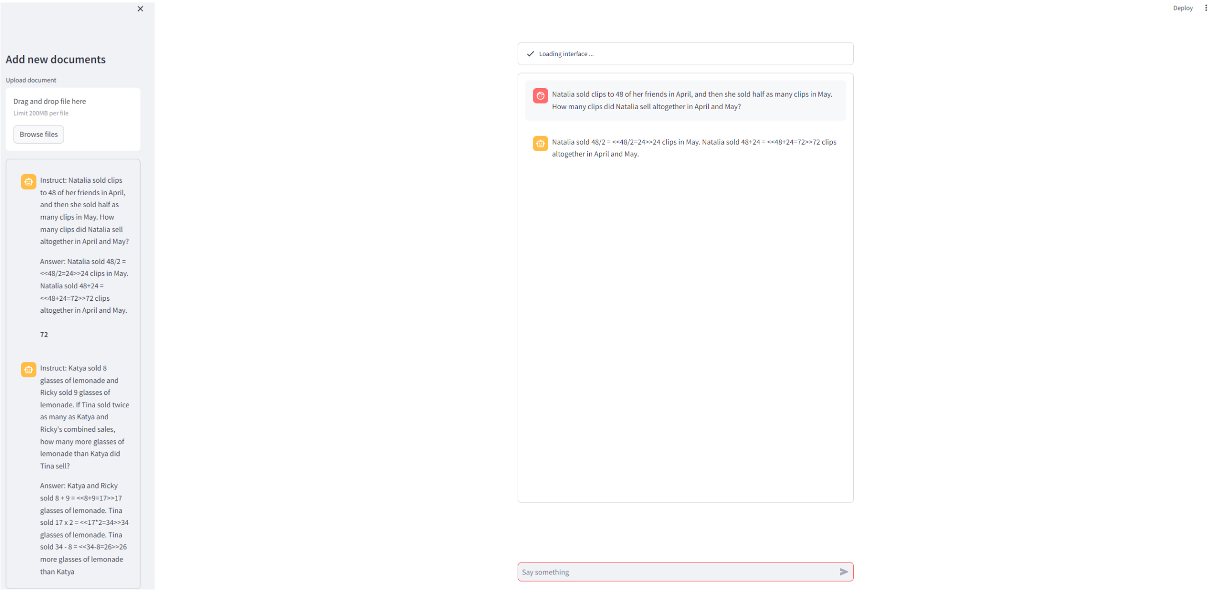
Base model response:

Natalia sold a total of 72 clips in April and May.

Fine-tuned model response:

Natalia sold clips to 48 friends in April, and then she sold half as many clips in May, which is 48/2 = 24 clips. Therefore, Natalia sold 48 + 24 = 72 clips altogether in April and May.

Figure 8: This screenshot showcases the interface used to query both the base and DPO models. It allows users to compare their outputs and assign points accordingly.



Instruct: Natalia sold clips to 48 of her friends in April, and then she sold half as many clips in May. How many clips did Natalia sell altogether in April and May?

Answer: Natalia sold 48/2 = <<48/2=24>>24 clips in May. Natalia sold 48+24 = <<48+24=72>>72 clips altogether in April and May.

72

Instruct: Katya sold 8 glasses of lemonade and Ricky sold 9 glasses of lemonade. If Tina sold twice as many as Katya and Ricky's combined sales, how many more glasses of lemonade than Katya did Tina sell?

Answer: Katya and Ricky sold 8 + 9 = <<8+9=17>>17 glasses of lemonade. Tina sold 17 x 2 = <<17*2=34>>34 glasses of lemonade. Tina sold 34 - 8 = <<34-8=26>>26 more glasses of lemonade than Katya

Natalia sold clips to 48 of her friends in April, and then she sold half as many clips in May. How many clips did Natalia sell altogether in April and May?

Natalia sold 48/2 = <<48/2=24>>24 clips in May. Natalia sold 48+24 = <<48+24=72>>72 clips altogether in April and May.

Figure 9: This screenshot displays the web interface for querying the RAG model. It features a left panel where sources retrieved from the RAG index are visible. Additionally, the interface allows users to upload files to expand the information stored in the index.

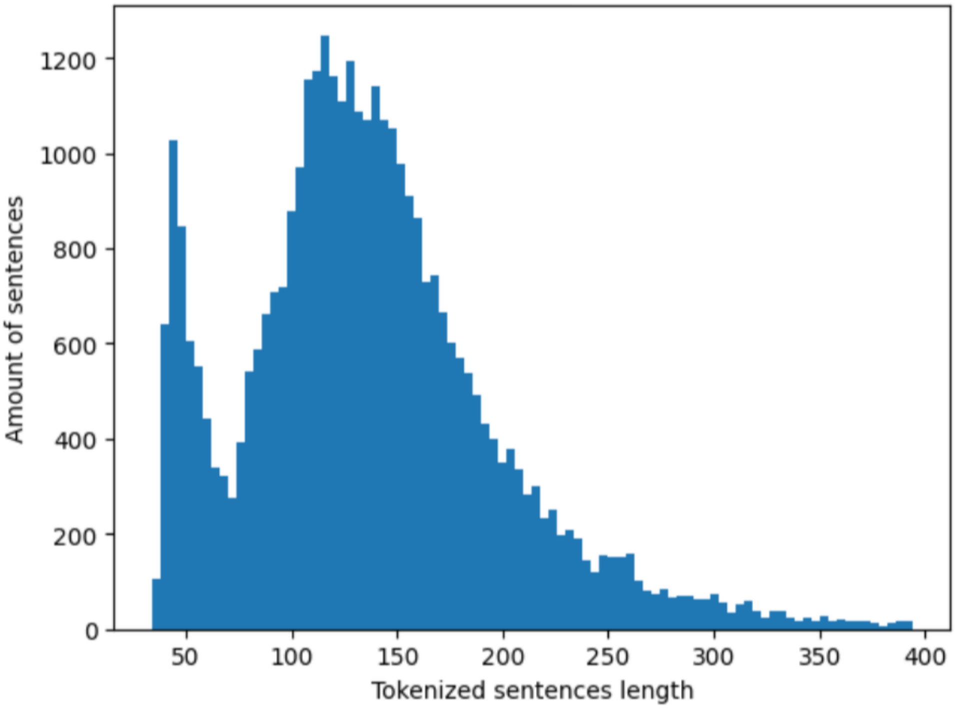


Figure 10: Length of the questions in the stem_qa_m1_index

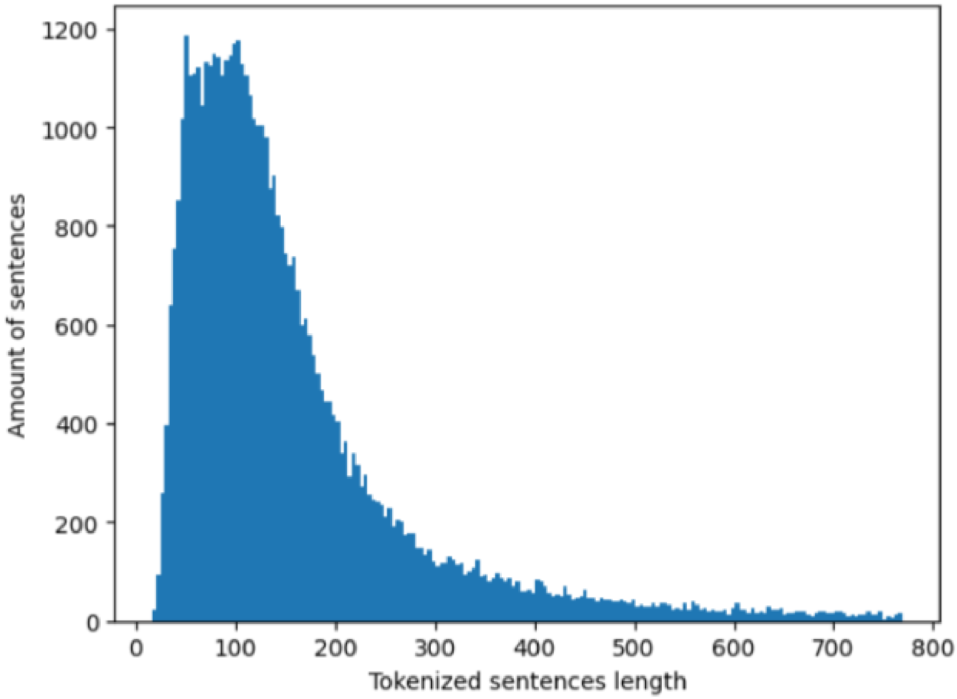


Figure 11: Length of the questions in the stem_open_question_index

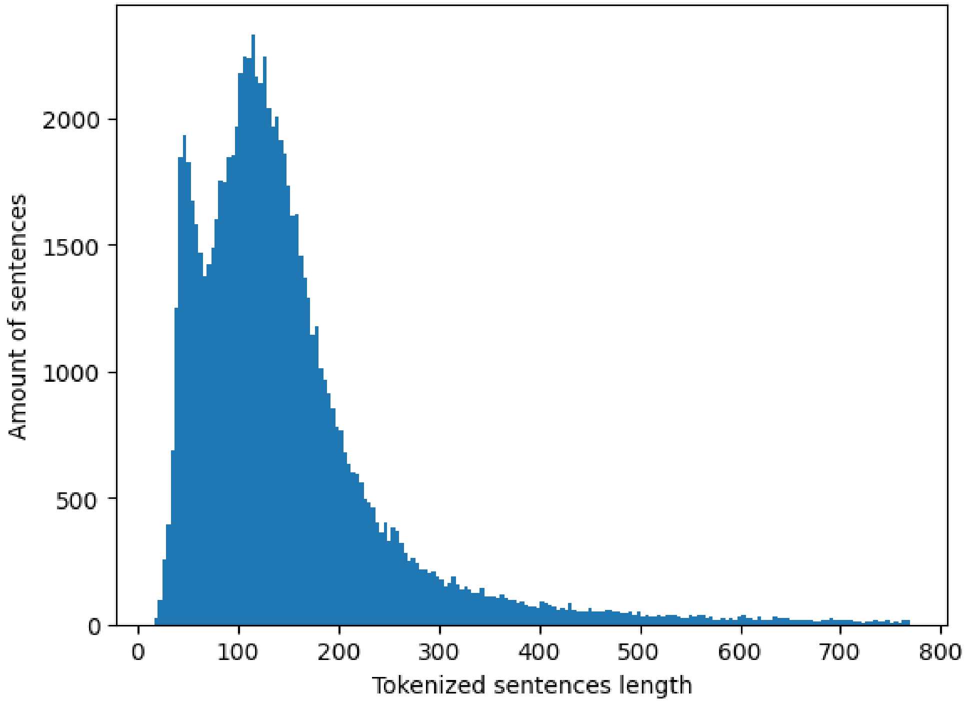
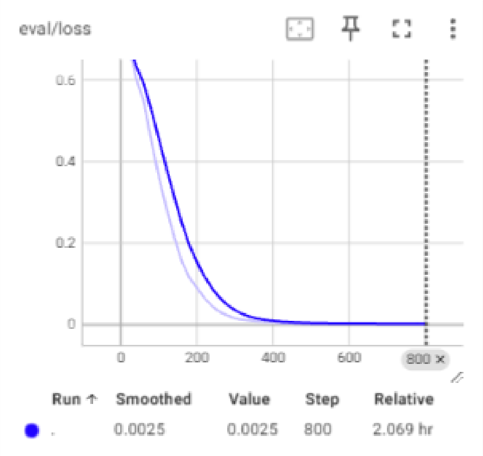


Figure 12: Length of the questions in the stem_all_index

Figure 13: (a) QA (b) Open questions (c) All

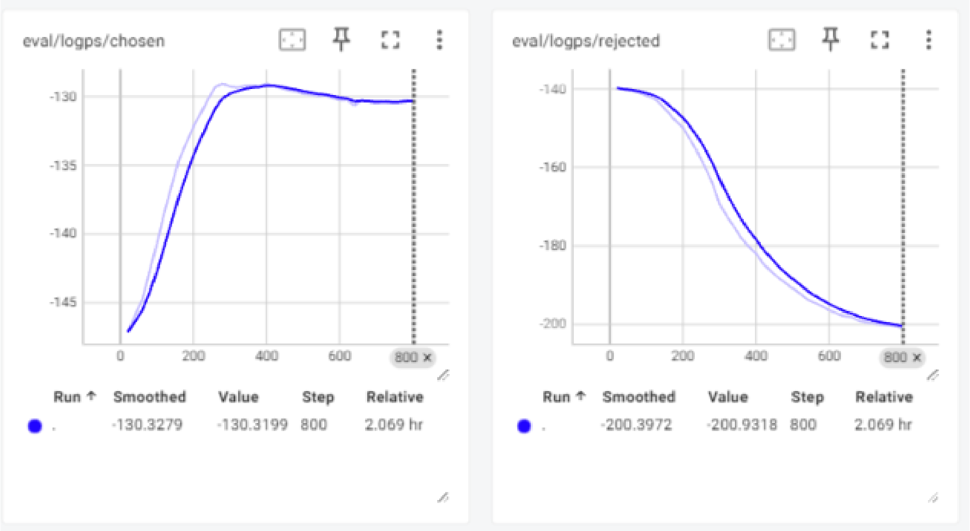
Dataset	Metric	Model		
		Phi-2	Instruct50k-Orca50k-GSM9k-LoRA-Phi2	DPO-M4AI-Instruct50k-Orca50k-GSM8k-LoRA-Phi2
M1_preference	bleu	0.1267	**0.1517**	0.1334
	bleu-1	0.2992	**0.3270**	0.3007
	bleu-4	0.0654	**0.0834**	0.0703
	rouge-1	0.2868	**0.3362**	0.3250
	rouge-2	0.1287	**0.1578**	0.1531
	rouge-l	0.1952	**0.2293**	0.2239
	bertscore-precision	0.0	**0.8563**	0.8344
	bertscore-recall	0.0	**0.7653**	0.7523
	bertscore-f1	0.0	**0.8082**	0.7912
	comet	0.2506	**0.8642**	0.7979
SciQ	bleu	0.0	0.0	0.0
	bleu-1	0.1997	0.2760	**0.2983**
	bleu-4	0.0	0.0	0.0
	rouge-1	0.5219	0.6156	**0.6199**
	rouge-2	0.1537	0.1856	**0.1910**
	rouge-l	0.5221	0.6143	**0.6181**
	bertscore-precision	1.0	1.0	1.0
	bertscore-recall	1.0	1.0	1.0
	bertscore-f1	1.0	1.0	1.0
	comet	0.6654	0.6588	**0.6654**
MMLU	bleu	**0.1139**	0.0761	0.0665
	bleu-1	**0.1761**	0.1221	0.1141
	bleu-4	**0.0801**	0.0525	0.0434
	rouge-1	**0.3278**	0.1987	0.1951
	rouge-2	**0.1929**	0.1171	0.1174
	rouge-l	**0.3141**	0.1886	0.1858
	bertscore-precision	0.4393	**0.5098**	0.4524
	bertscore-recall	0.4588	**0.8803**	0.5098
	bertscore-f1	0.4488	**0.6457**	0.4794
	comet	0.4640	**0.5824**	0.5236

Table 2: Results of different datasets for the models Phi-2, Instruct50k-Orca50k-GSM9k-LoRA-Phi2, and DPO-M4AI-Instruct50k-Orca50k-GSM8k-LoRA-Phi2 in zero shot prompting



Run ↑	Smoothed	Value	Step	Relative
● .	0.0025	0.0025	800	2.069 hr

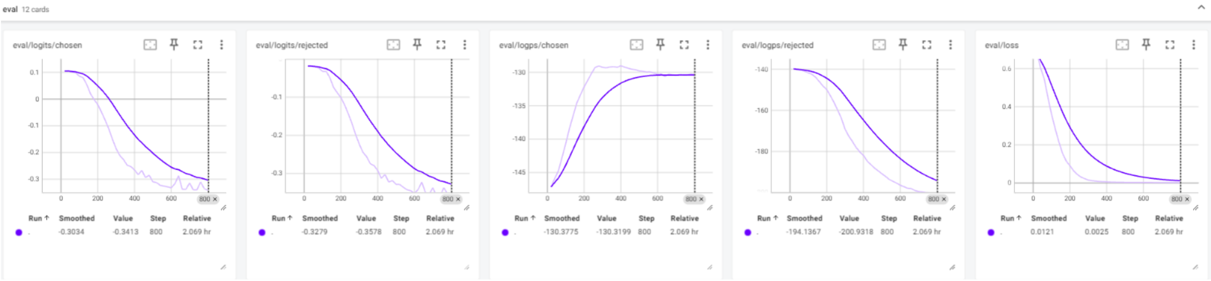
Figure 14: Evaluation loss on the M4-AI dataset.



Run ↑	Smoothed	Value	Step	Relative
● .	-130.3279	-130.3199	800	2.069 hr

Run ↑	Smoothed	Value	Step	Relative
● .	-200.3972	-200.9318	800	2.069 hr

Figure 15: Comparison between the log probabilities between the chosen and rejected outputs.



Run ↑	Smoothed	Value	Step	Relative
● .	-0.3034	-0.3413	800	2.069 hr

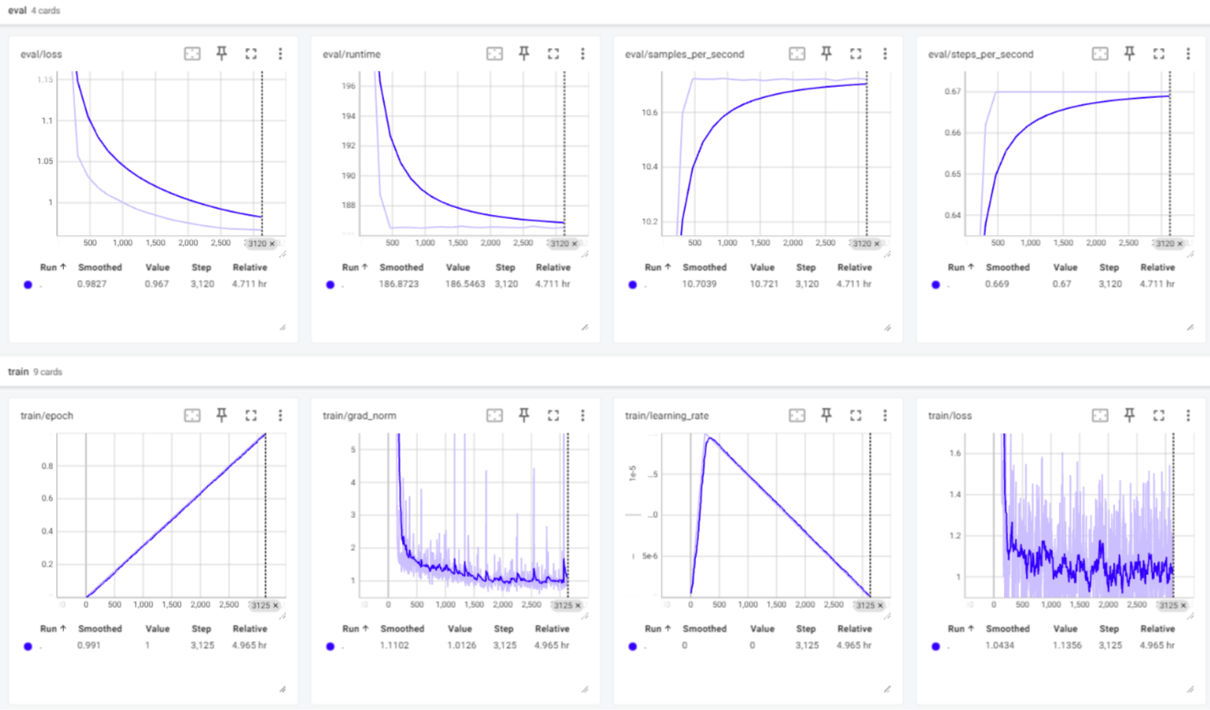
Run ↑	Smoothed	Value	Step	Relative
● .	-0.3279	-0.3578	800	2.069 hr

Run ↑	Smoothed	Value	Step	Relative
● .	-130.3775	-130.3199	800	2.069 hr

Run ↑	Smoothed	Value	Step	Relative
● .	-194.1367	-200.9318	800	2.069 hr

Run ↑	Smoothed	Value	Step	Relative
● .	0.0121	0.0025	800	2.069 hr

Figure 16: Screenshot showing the training for the DPO-M4AI dataset



eval/loss

Run ↑	Smoothed	Value	Step	Relative
● .	0.9827	0.967	3,120	4.711 hr

eval/runtime

Run ↑	Smoothed	Value	Step	Relative
● .	186.8723	186.5463	3,120	4.711 hr

eval/samples_per_second

Run ↑	Smoothed	Value	Step	Relative
● .	10.7039	10.721	3,120	4.711 hr

eval/steps_per_second

Run ↑	Smoothed	Value	Step	Relative
● .	0.669	0.67	3,120	4.711 hr

train/epoch

Run ↑	Smoothed	Value	Step	Relative
● .	0.991	1	3,125	4.965 hr

train/grad_norm

Run ↑	Smoothed	Value	Step	Relative
● .	1.1102	1.0126	3,125	4.965 hr

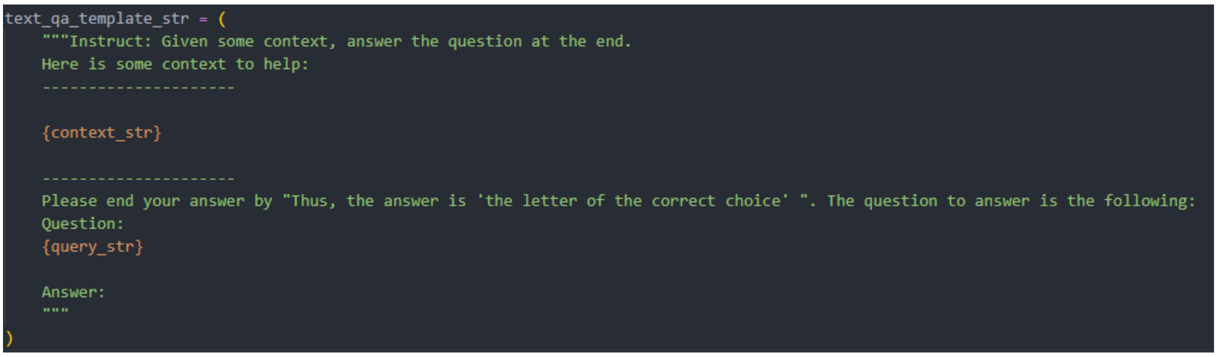
train/learning_rate

Run ↑	Smoothed	Value	Step	Relative
● .	0	0	3,125	4.965 hr

train/loss

Run ↑	Smoothed	Value	Step	Relative
● .	1.0434	1.1356	3,125	4.965 hr

Figure 17: Screenshot showing the training metrics for our finetuning on Orca



```
text_qa_template_str = (
    """Instruct: Given some context, answer the question at the end.
    Here is some context to help:
    ---------------------

    {context_str}

    ---------------------
    Please end your answer by "Thus, the answer is 'the letter of the correct choice' ". The question to answer is the following:
    Question:
    {query_str}

    Answer:
    """
)
```

Figure 18: Prompt used to query the RAG model in order to collect the best letter for the MCQ