EdNA

Lysandre Costes | 341724 | lysandre.costes@epfl.ch
Hassen Aissa | 341649 | hassen.aissa@epfl.ch
Levin Hertrich | 387376 | levin.hertrich@epfl.ch
Yassine Turki | 344704 | yassine.turki@epfl.ch
I-am-an-assistant
Github repo

Abstract

AI tutors for STEM education often struggle with accuracy and efficiency. We propose EdNA (Educational Nimble Assistant), a compact AI tutor combining Direct Preference Optimization (DPO), Reinforcement Learning (RLVR) for structured answers, and Retrieval-Augmented Generation (RAG) for factual grounding. EdNA achieves significant gains on STEM benchmarks while using quantization for efficient deployment.

1 Introduction

Building a reliable AI tutor for STEM students presents unique challenges: large language models often struggle to produce consistently correct, well-structured answers, and it is difficult to run them on resource-limited devices. In this project, we propose EdNA (Educational Nimble Assistant), a compact AI tutor tailored to STEM learners. First, we align the model to high-quality STEM preferences using Direct Preference Optimization (DPO) (Rafailov et al., 2023) with distilled data from multiple state-of-the-art LLMs. Next, we fine-tune on mixed STEM and instruction tuning datasets and apply Reinforcement Learning With Verifiable Reward (RLVR) (Lambert et al., 2025) to improve multiple-choice answering. To ensure factual grounding, we integrate a Retrieval Augmented Generation (RAG) pipeline that fetches relevant document chunks at inference time. Finally, we apply the AWQ algorithm (Lin et al., 2024) for quantization, to shrink EdNA's footprint without sacrificing accuracy. This pipeline enables EdNA to deliver accurate, formatted STEM solutions while running efficiently on smaller GPUs or CPUs. Our work differs from existing work like (Hicke et al., 2023), as we focus on challenging STEM questions instead of standard question answering.

2 Approach

DPO To align EdNA with human-like STEM tutor responses, we use Direct Preference Optimization (DPO), because it is more stable, lightweight, and performant than methods like Reinforcement Learning from Human Feedback (RLHF) (Rafailov et al., 2023). Instead of training a separate reward model, DPO's closed-form objective binds reward to the model's log-probability ratios (a more formal definition is provided in Section A.1). We use the trl library from HuggingFace for the DPO implementation (von Werra et al., 2020). To obtain high quality preference data we follow the approach of (Tunstall et al., 2023) and use distillation from LLMs and LLM-as-a-judge. This is described in more detail in section 3.

MCQA The strategy to train the MCQA model consists of two main stages. First, we perform supervised fine-tuning on a very large dataset containing a mixture of different STEM and non STEM datapoints. These are meant to make the model see tokens related to scientific subjects but also see conversations and instruction following datapoints in order to avoid catastrophic forgetting. Then, we use the fine-tuned model to train it specifically on the MCQ answering task. We do this through Reinfocement Learning With Verifiable Reward (RLVR) (Lambert et al., 2025). Our RL training is however different from the one presented in that paper, we not only reward correct answers with a +1 but also penalize the wrong answers with a -1, pushing the model to deviate from them. In addition to this, we reward the model on the correctness of the format of the answer with 0.5. This helps the model to provide the structure of answers that is needed. The training is done through Group Relative Policy Optimization (GRPO) (Shao et al., 2024), to enhance stability and efficiency in our training.

Quantization Our model needs to be run on smaller devices and take less space in storage and memory. Thus, we quantize it using different algorithms (QLORA (Dettmers et al., 2023), Quantization Aware Training (QAT) (Jacob et al., 2017), and Post-Training-Quantization with simple weight quantization (Dettmers and Zettlemoyer, 2023), AWQ (Lin et al., 2024), SmoothQuant (Xiao et al., 2024), and GPTQ (Frantar et al., 2023)) in order to retrieve a similar performance to our MCQA model while quantizing weights to 4 bits and activations to 8 bits when applicable.

RAG We implement a RAG pipeline (Lewis et al., 2021) by combining a dense retriever and a generator model as shown in Figure 1. The primary goal of this system is to incorporate relevant factual information retrieved from a document corpus at inference time to help the model answer questions more accurately.

We use the sentence-transformers/all-MiniLM-L6-v2 model to encode both the queries and document chunks. For each incoming question, we retrieve the top 3 most similar documents based on cosine similarity. The generator baseline model used was Qwen3-0.6B-Base.

We use Retrieval Augmented FineTuning (RAFT) (Zhang et al., 2024), a form of contrastive learning, to teach the generator model to discern relevant from irrelevant passages. For each training instance, the model receives both golden (relevant) and defective (irrelevant) documents alongside the question and expected answer, enabling it to learn from both positive and negative evidence.

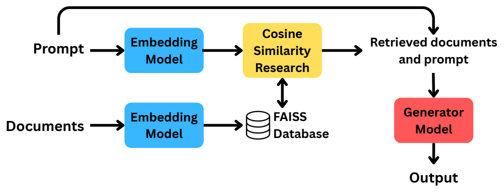

Figure 1: Retrieval Augmented Generation Architecture

3 Experiments

3.1 Data

MCQA and Quantized For MCQA training, We use a mixture of multiple datasets composed of math, abstract algebra and coding from the Tulu3 sft dataset (Lambert et al., 2025), math questions sourced from different Stack Exchange websites available on (Zhang, 2024) and STEM MCQ dataset training splits. We also add instruction following datapoints. Finally, we add a chain of thought dataset. For the MCQ and the COT datasets, we create a 2 pool of prompts, one for each dataset, we respectively ask for the answer only or a reasoning followed by the answer. We select randomly a prompt template from the pool for each dataset element. The pools are created using commercial LLMs starting from multiple examples, we then curate them by manually adapting and changing a couple of them to remove repeated patterns. For the second stage, we only use the MCQ datasets listed above.

Our dataset allows the model to attend to Math information that it struggles on while not losing its current abilities, as the base model already has good results in MCQ. This avoids catastrophic forgetting.

The Data is divided in training, validation and testing subsets where the validation and testing only contain those splits of the MCQ datasets as it's only relevant for our task to evaluate it on MCQs. For testing, we add the whole MMLU-PRO testing set (not only STEM) in order to asses the generalization in the reasoning capacities of our models.

DPO Our initial dataset consists of 15 467 manually annotated preference pairs generated from GPT-4o-mini (OpenAI, 2024) for 1 264 university STEM questions. These annotations reflect that many questions admit multiple valid solutions. We merged this with OpenAI's high-quality WebGPT comparison set (Nakano et al., 2021), but low-quality and redundant answers harmed DPO performance (Lambert et al., 2024a), yielding unsatisfactory results (Section 3.5).

Following (Tunstall et al., 2023), we distill better data using an ensemble of Qwen/Qwen3-8B (Team, 2025b), Mistral-7B-Instruct-v0.2 (Jiang et al., 2023), and Meta-Llama-3-8B-Instruct (AI@Meta, 2024). We generate an answer from each model for the 1 264 STEM-questions described above, as well as for the SciQ dataset (Welbl et al., 2017), and 5 000 questions from the tulu-3-sft-personas-math dataset (Lambert et al., 2024a). These answers are then ranked using GPT-4o-mini (OpenAI, 2024) as a LLM-as-a-judge (Gu et al., 2025) which results in roughly 16 000 distilled, high quality preference pairs. Our approach is shown in Figure 2.

To further specialize in mathematics and computer science and expand our training data, we augment this distilled set with 5 000 samples from

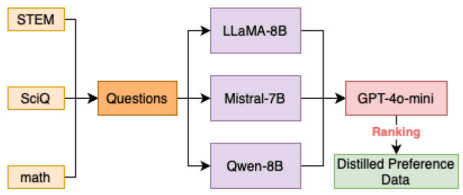

Figure 2: High-level overview of our preference-distillation pipeline.

Vezora/Code-Preference-Pairs (Vezora, 2024), 7 000 from jondurbin/py-dpo-v0.1 (jondurbin, 2024), and 5 000 from prhegde/preference-data-math-stack-exchange (Lambert et al., 2023).

RAG For our RAG training, we curated a specialized knowledge base by combining two types of documents: STEM-related text chunks extracted from Wikipedia (Biswas, 2024), and documents from the dataset used to train the MCQA model. This mixed corpus is intended to both provide background knowledge and reinforce the structure and semantics of multiple-choice reasoning. Due to the size limitations imposed by the course, we worked with a sub sampled version of our full document database. The version can be found here.

To enable RAFT training (Zhang et al., 2024), we augmented the MCQA dataset (Section 3.1) by annotating the first and fifth retrieved documents respectively as the "golden" and "distracting" documents to build a contrastive learning setup.

3.2 Evaluation

MCQA performance We conduct two types of evaluation for the MCQ task, the first one is based on the log likelihood of each of the possible answers happening after the given question. A question is answered correctly if the correct answer has the highest likelihood. This metric allows for an easy training that does not depend on the actual output of the model providing reliability in the reported performances, as it's difficult to judge if a model answer is correct or wrong based on its generated text due to the variability in the output format.

However, this variability is not relevant for our case as our model is trained to follow a strict template for the output. This allows us to create another metric we name the Output Correctness (OC) based on the correctness of the generation of the model. The final accuracy is thus the percentage of questions to which the model answers by exactly the correct option in the first line of its output. For both of these metrics, we perform zero-shot and few-shot testing to compute the final accuracies. We perform the inference multiple times (5 times) on each datapoint in order to evaluate the consistency of the model's answer, for this we use the standard deviation of the accuracy on each dataset.

DPO We evaluate our DPO model using the reward-model accuracy metric (see section A.2 for its formal definition).

To be able to evaluate models trained with different datasets and hyperparameters, we created a testing set which is a combination of the well-known RewardBench dataset (Lambert et al., 2024b) as well as all STEM and code related validation data from the HelpSteer dataset (Wang et al., 2023).

3.3 Baselines

DPO For our DPO model we use the Qwen3-0.6B-Base (Team, 2025a) base model trained on our initial DPO dataset (see section 3.1) as our baseline. We choose this model as baseline as it is an already fine-tuned model with decent performance which is not yet sufficient for our end-goal task.

MCQA For the MCQ tasks, we also compare our results to the models with different fine-tuning configurations, starting with the Qwen base model, the fine-tuned version of it, then with applying RLVR to it. In addition to this, we compare it to the LLaMA-1B model (AI@Meta, 2024) as it has more parameters and therefore should have higher performance.

RAG and Quantized We compare RAG models and quantized models against our baseline MCQA model to quantify the performance trade-offs introduced by retrieval augmentation and model compression.

3.4 Experimental Details

DPO DPO does not require extensive hyperparameter tuning except for β (Rafailov et al., 2023). Therefore, we only experimented with different values for β (see 3.5), changed the number of epochs with our dataset size and kept the other hyperparameters fixed. These can be found in table 2.

MCQA For MCQA training, we were conservative about the parameters to get good stability. Thus, for GRPO we tuned manually the learning rate to have a stable training which is not hard, as the batch size for GRPO is high, as the length of the training sequences is short. Having a high batch size guarantees gradient stability and we increase this guarantee by adding gradient norm clipping at 0.5.

Model	Type	Shots	SciQ	MMLU	AquaRat	ARC-E	ARC-C	MMLU PRO
LLama 1B	likelihood	0	83.1	36.5	20.9	74.2	52.7	18.9
	likelihood	10	80.6	36.0	20.9	73.4	54.4	–
	OC	0	0.0 ± 0.0	0.0 ± 0.0	0.1 ± 0.2	0.0 ± 0.0	0.0 ± 0.0	–
	OC	5	28.3 ± 1.1	18.3 ± 0.7	10.7 ± 0.5	18.6 ± 0.9	15.6 ± 0.5	–
Qwen 0.6B Base	likelihood	0	84.1	43.9	23.2	**81.5**	63.3	19.0
	likelihood	10	86.0	**47.4**	29.1	83.6	66.0	–
	OC	0	18.9 ± 1.4	4.4 ± 0.2	2.5 ± 0.7	17.3 ± 0.7	14.6 ± 1.0	–
	OC	5	77.1 ± 0.3	37.1 ± 0.7	9.5 ± 2.6	68.4 ± 0.9	53.5 ± 1.2	–
Qwen SFT	likelihood	0	84.4	43.5	27.6	79.9	61.9	20.0
	likelihood	10	86.7	45.1	26.8	83.6	65.8	–
	OC	0	77.0 ± 0.9	34.9 ± 0.3	19.5 ± 2.6	65.8 ± 1.0	49.5 ± 0.5	–
	OC	5	67.4 ± 0.9	34.4 ± 0.6	16.8 ± 1.0	63.2 ± 0.3	48.9 ± 0.8	–
EdNa	likelihood	0	**84.7**	**44.0**	**35.4**	81.3	**64.7**	**22.7**
	likelihood	10	**87.4**	44.6	**32.3**	**84.3**	**68.4**	–
	OC	0	**84.0 ± 0.7**	**42.4 ± 0.7**	**34.1 ± 1.0**	**76.6 ± 0.5**	**62.0 ± 0.3**	–
	OC	5	**84.3 ± 0.7**	**42.5 ± 0.3**	**31.1 ± 2.0**	**77.2 ± 0.5**	**62.8 ± 0.7**	–
EdNa + RAG	likelihood	0	84.3	42.3	32.7	79.6	62.0	–
EdNa + RAG + RAFT	likelihood	0	81.5	40.9	31.9	75.8	58.1	–
EdNa BitsAndBytes 4bit	likelihood	0	81.3	40.9	33.1	77.2	62.1	19.6
	likelihood	10	84.6	42.2	30.7	80.4	64.1	–
	OC	0	67.1 ± 0.6	27.4 ± 0.7	16.0 ± 2.2	56.2 ± 0.7	44.3 ± 1.1	–
	OC	5	77.4 ± 1.0	35.5 ± 0.5	17.0 ± 1.3	67.6 ± 0.7	53.7 ± 0.3	–
EdNa Quant. (AWQ 4bit)	likelihood	0	82.6	41.2	29.9	79.7	63.4	22.4
	likelihood	10	84.3	42.9	30.3	81.8	66.1	–
	OC	0	67.8 ± 0.5	28.3 ± 0.1	23.5 ± 1.0	62.3 ± 0.4	50.2 ± 0.2	–
	OC	5	80.7 ± 0.4	38.2 ± 0.4	17.7 ± 1.5	73.9 ± 0.2	59.4 ± 0.5	–

Table 1: Performance comparison of multiple LLM variants across benchmark datasets under likelihood and output correctness (OC) evaluation. EdNa maintains state-of-the-art results with or without quantization. Dashes (–) indicate unavailable scores. It was not possible to compute MMLU-Pro for long sequences like few shot and RAG due to its long questions.

Hyperpam	DPO	SFT	GRPO
Learning Rate	3×10^{-6}	2×40^{-6}	5×10^{-6}
Scheduler	linear	cosine	cosine
Weight Decay	0.01	0.01	0.01
Batch Size	2	8	200
Grad. accumul.	4	4	3
Epochs	2/3	2	3

Table 2: Hyperparameters for DPO, SFT, and GRPO.

RAG RAFT was trained using DoRA (Liu et al., 2024) with the following settings with rank 32. We used a batch size of 2 with gradient accumulation of 8, a learning rate of 2e-5. However, we have only been able to train on a quarter of an epoch due to resources limitations. We experimented with different number of retrieved documents and similarity functions. We found that using cosine similarity and retrieving a single document consistently led to more robust results. This is likely because fewer, relevant documents reduce noise for the generator.
Quantized As we want to minimize VRAM usage, we always aimed for 4-bit quantization. Our best model was obtained by applying AWQ on EdNA. AWQ requires calibration samples to make the algorithm aware of the distribution of activations. Thus, we used the auxiliary_train split of MMLU and tuned the q_group_size, the number of calibration samples, and the max sequence length. The QAT and QLoRA models had the same hyper parameters as for the SFT model and GRPO models. Finally, we also experimented with SmoothQuant + GPTQ and tuned the smoothing constant, the calibration dataset and the number of samples.

3.5 Results

EdNA performance on MCQs: The results in Table 1 show the gain and loss of performance in the stages of training. The SFT phase does not always have a positive impact on the likelihood-based accuracy, its main increase is on AquaRat, which makes sense as it's a mathematics dataset so the fine-tuning allows to recognize more mathematical patterns. The loss on MMLU, and ARC-Easy

Model	Dataset	Base Model	β	Overall Acc (%)	HelpSteer (%)	RewardBench (%)
Baseline	(1)	Qwen0.6B-Base	1.0	54.80	50.22	55.85
Distillation	(2)	Qwen0.6B-Base	1.0	65.87	51.98	69.05
Distillation_beta	(2)	Qwen0.6B-Base	0.5	66.96	**53.59**	70.02
Distillation_beta_sft*	(2)	SFT-Model	0.5	**67.53**	48.17	**71.96**

Table 3: Quantitative results comparing our DPO models on their datasets, base models, tuning parameter β, overall accuracy, HelpSteer and RewardBench scores. Dataset (1) refers to the initial dataset and dataset (2) to the distilled dataset merged with chosen existing datasets, for more details see section 3.1.
*Best overall model on "Overall Acc (%)".

and ARC-Challenge datasets is expected as the SFT is a warmup for the RLVR stage as discussed before. The RLVR stage induces a gain in performance in all the datasets but one (ARC-E has a slight drop). The important gain in performance in AquaRat dataset 35.4 for EdNa compared to 23.2 for the Base model, shows the reasoning capabilities the model gains from the RLVR training. This is further highlighted by the gain of performance (+3.7 points) on MMLU-PRO showing reasoning generalization capacity of the model. For the OC metric, we see a higher impact for our training. The SFT phase teaches the model the basics of how to answer MCQs due to the training with the pool of prompts. This is further enhanced with the RLVR training, thanks to the format correctness reward that enforces a correct answering format.

EdNA performance with DPO: Table 3 summarizes our DPO results for four models: the baseline (Section 3.3), Qwen3-0.6B-Base fine-tuned on distilled data with $\beta = 1.0$ and $\beta = 0.5$, and the SFT model similarly fine-tuned with $\beta = 0.5$ (see Table 1). All DPO variants outperform the baseline, showing the strength of our distilled dataset. Additionally, reducing β from 1.0 to 0.5 consistently boosts performance, and applying DPO on top of SFT yields the highest overall accuracy.

RAG Table 1 shows that the RAG integration underperforms compared to our MCQA model. Results are based on retrieving the top 3 documents. We also experimented with 1 and 5 documents but these configurations gave lower likelihood scores.

Quantized We only report our top-performing models in terms of accuracy and VRAM, which are EdNA with AWQ, and EdNA quantized with simple 4-bit quantization. We see that for these models, in particular for AWQ, the performance is very close to the full model while consuming half the VRAM 4. QLoRA and QAT obtained comparable results to the SFT model, but were missing the RL component that boosted EdNA's performance (1). SmoothQuant+GPTQ obtained similar results to AWQ, but with a much higher VRAM usage.

4 Analysis

DPO Performance Analysis: We find that high-quality preference data, partly generated with distillation, substantially boosts performance, as seen by the jump from our baseline to the models trained on distilled data. During training, we consistently observed validation accuracy well above testing accuracy (Figure 5), so we reduced β to 0.5 to keep the fine-tuned model closer to the reference. This adjustment further improved results, demonstrating the strong effect of β on DPO. Usually, applying DPO after SFT enhances performance (Lambert et al., 2024a). This aligns with our results, as our top-performing model is the one that applies DPO on the SFT model (Section 3.5). We observe that our models score higher on RewardBench than on HelpSteer, indicating robust overall alignment but ongoing difficulty with challenging STEM and coding preferences.

Power of Reinforcement learning: By using RLVR with imposed answering format, our model achieves results on the OC metric with 0-shot much better than 5-shot answers of Llama-1B and Qwen Base model. This can be seen in Table 1 and in Figure 3, where we analyze EdNA performance per MMLU subject and see that it increases in all MMLU STEM subjects. We can also see that few-shot has no impact on the model output as opposed to Qwen for which it increases the performance. This allows for a much faster inference on EdNA (11 seconds 0-shot, 53 seconds 5-shot, mean time inference on SciQ for the same batch size) with a better accuracy, and thus the nimble aspect of our model. Few-shot improves almost always EdNA in the likelihood metric as shown in Table 1. This is due to the deterministic aspect of the metric (temperature=0) where more context about the subject tested in the prompt helps the model in the reasoning. Finally, it's important to note that EdNA does not lose stability in its correctness as reflected by the standard deviation values.

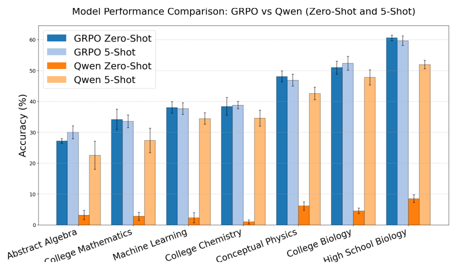

Figure 3: Comparison of percentages of correct outputs on subjects from MMLU on 0 and few shot prompts.

RAG We can make the hypothesis that the deceptive results are due to two primary factors: first, the subsampled document database probably does not provide enough chunks to span a topic as broad as STEM, resulting in frequent retrieval of irrelevant passages; second, resource limitations compromised RAFT's efficacy in training the generator to robustly discern and reject unhelpful retrieved content. Indeed, we have only been able to train on about 25% of our RAFT dataset. Consequently, utilizing the full document corpus and optimizing RAFT under less constrained conditions are identified as critical future requirements to realize RAG's theoretical advantages. The qualitative evaluation shows RAG outperforms MCQA when a good document is available (Appendix A.4), supporting our hypothesis that using the full, larger database would boost performance.

Quantization In Figure 4, we report the VRAM usage for each quantization algorithm. Measurements were performed on Google Colab and averaged over three independent runs. In each run, we processed 15 batches with a batch size of 8, generating 24 tokens per input. The first five batches of each run were treated as warmup and excluded from the statistics; VRAM usage is averaged over the remaining five batches. We can see that overall, the quantized models need around half of the VRAM of EdNA to run, while giving similar results. AWQ stands out as the most accurate and smallest model. We notice a high VRAM usage of SmoothQuant+GPTQ, this is mostly due to loading with transformers rather than VLLM library.

5 Ethical considerations

EdNA is designed to help English-speaking STEM students learn STEM concepts interactively. Because our training data is exclusively in English, EdNA currently only supports English. Incorporating multilingual data (e.g., Spanish, French, Chinese) would improve performance in high-resource languages. To extend support to low-resource languages (e.g., Urdu, Swahili), we could translate existing English data as in prior work (Ghafoor et al., 2021; Hangya et al., 2022).

While English-speaking STEM students benefit, non-English STEM students may be disadvantaged. Non-STEM students might also misuse EdNA for subjects outside its training, leading to hallucinations and incorrect answers that harm academic performance. Additionally, EdNA could harm teachers and institutions as it could facilitate cheating in homeworks and exams.

To mitigate these harms, we could add multilingual data to avoid disadvantaging non-English STEM students, broaden training to non-STEM subjects to reduce hallucination risk, and expand preference data so the model rejects cheating prompts.

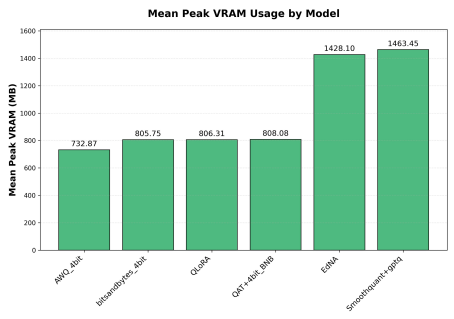

Figure 4: Mean Peak VRAM usage over 3 runs.

6 Conclusion

In this work, we introduced EdNA, a compact STEM tutor combining DPO alignment, RLVR fine-tuning, RAG retrieval, and AWQ quantization. Distilling and optimizing human preferences improved reward-model accuracy (from 54.8% to 67.5% on RewardBench and HelpSteer). RLVR with format-aware rewards boosted MCQA performance (e.g., AquaRat accuracy rose from 23.2% to 35.4%). Our RAG integration showed qualitative gains when relevant documents were retrieved. Applying 4-bit AWQ quantization halved VRAM use while preserving accuracy. Future work includes multilingual support, expanding the retrieval corpus, and dynamic quantization strategies.

References

AI@Meta. 2024. Llama 3 model card.

Raja Biswas. 2024. Wiki stem corpus.

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

Tim Dettmers and Luke Zettlemoyer. 2023. The case for 4-bit precision: k-bit inference scaling laws. In *International Conference on Machine Learning*, pages 7750–7774. PMLR.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2023. Gptq: Accurate post-training quantization for generative pre-trained transformers.

Abdul Ghafoor, Ali Shariq Imran, Sher Muhammad Daudpota, Zenun Kastrati, Abdullah, Rakhi Batra, and Mudasir Ahmad Wani. 2021. The impact of translating resource-rich datasets to low-resource languages through multi-lingual text processing. *IEEE Access*, 9:124478–124490.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Saizhuo Wang, Kun Zhang, Yuanzhuo Wang, Wen Gao, Lionel Ni, and Jian Guo. 2025. A survey on llm-as-a-judge.

Viktor Hangya, Hossain Shaikh Saadi, and Alexander Fraser. 2022. Improving low-resource languages in pre-trained multilingual language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11993–12006, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yann Hicke, Anmol Agarwal, Qianou Ma, and Paul Denny. 2023. Ai-ta: Towards an intelligent question-answer teaching assistant using open-source llms. *arXiv preprint arXiv:2311.02775*.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. 2017. Quantization and training of neural networks for efficient integer-arithmetic-only inference.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b.

jondurbin. 2024. Python-preference-pairs dataset. `https://huggingface.co/datasets/jondurbin/py-dpo-v0.1`. Accessed: 2025-06-06.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. 2025. Tulu 3: Pushing frontiers in open language model post-training.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. 2024a. T\" ulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*.

Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, Noah A. Smith, and Hannaneh Hajishirzi. 2024b. Rewardbench: Evaluating reward models for language modeling.

Nathan Lambert, Lewis Tunstall, Nazneen Rajani, and Tristan Thrush. 2023. Huggingface h4 stack exchange preference dataset.

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. 2021. Retrieval-augmented generation for knowledge-intensive nlp tasks.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024. Awq: Activation-aware weight quantization for llm compression and acceleration.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024. Dora: Weight-decomposed low-rank adaptation.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. 2021. Webgpt: Browser-assisted question-answering with human feedback. In *arXiv*.

OpenAI. 2024. Gpt-4o-mini. `https://openai.com/index/gpt-4o-mini-advancing-cost-efficient\-intelligence/`. Accessed: 2025-06-06.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, pages 53728–53741. Curran Associates, Inc.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models.

Qwen Team. 2025a. Qwen3.

Qwen Team. 2025b. Qwen3 technical report.

Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro Von Werra, Clémentine Fourrier, Nathan Habib, et al. 2023. Zephyr: Direct distillation of lm alignment. *arXiv preprint arXiv:2310.16944*.

Vezora. 2024. Code-preference-pairs dataset. `https://huggingface.co/datasets/Vezora/Code-Preference-Pairs`. Accessed: 2025-06-06.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. 2020. Trl: Transformer reinforcement learning. `https://github.com/huggingface/trl`.

Zhilin Wang, Yi Dong, Jiaqi Zeng, Virginia Adams, Makesh Narsimhan Sreedhar, Daniel Egert, Olivier Delalleau, Jane Polak Scowcroft, Neel Kant, Aidan Swope, and Oleksii Kuchaiev. 2023. Helpsteer: Multi-attribute helpfulness dataset for steerlm.

Johannes Welbl, Nelson F Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. *arXiv preprint arXiv:1707.06209*.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2024. Smoothquant: Accurate and efficient post-training quantization for large language models.

Tianjun Zhang, Shishir G. Patil, Naman Jain, Sheng Shen, Matei Zaharia, Ion Stoica, and Joseph E. Gonzalez. 2024. Raft: Adapting language model to domain specific rag.

Yifan Zhang. 2024. Stackmathqa: A curated collection of 2 million mathematical questions and answers sourced from stack exchange.

A Appendix

A.1 DPO Objective

The DPO objective is defined as follows. Given a preference pair (y^+, y^-), prompt x, reference policy π_{ref}, and target policy π_θ, the objective is:

$$L_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,y^+,y^-)\sim\mathcal{D}} \left[\log\sigma\left(\beta\log\frac{\pi_\theta(y^+ \mid x)}{\pi_{\text{ref}}(y^+ \mid x)} - \beta\log\frac{\pi_\theta(y^- \mid x)}{\pi_{\text{ref}}(y^- \mid x)}\right)\right]$$

A.2 Reward Accuracy

Reward accuracy is defined as follows:

$$\text{chosen\_reward} = \log\pi_\theta\left(y^+ \mid x\right) - \log\pi_{\text{ref}}\left(y^+ \mid x\right)$$

$$\text{rejected\_reward} = \log\pi_\theta\left(y^- \mid x\right) - \log\pi_{\text{ref}}\left(y^- \mid x\right)$$

Our model is correct if chosen_reward $>$ rejected_reward.

Here, π_{ref} denotes the base model, π_θ denotes our fine-tuned policy, $\left(y^+,\, y^-\right)$ is the preference pair and x the corresponding prompt. If the inequality does not hold, it is considered incorrect. Finally, reward-model accuracy is:

$$\text{Accuracy} = \frac{\#(\text{chosen\_reward} > \text{rejected\_reward})}{\#\text{Datapoints}}$$

A.3 DPO Validation and Test Accuracy

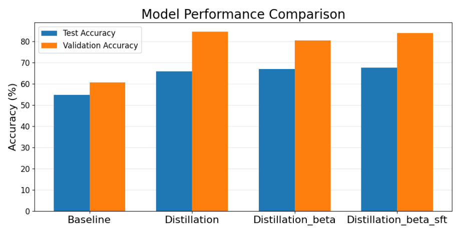

Figure 5: Comparison of the test and validation losses for our DPO models.

A.4 RAG Inference Example

Question (the right answer is C):

```
Which of the following statements best explains
   how Euler's theorem generalizes Fermat's
   Little Theorem?

A. Euler's theorem only applies when the
   modulus \( n \) is a prime number, just
   like Fermat's Little Theorem.
B. Euler's theorem removes the requirement for
   \( a \) to be coprime with \( n \), unlike
   Fermat's Little Theorem.
```

```
C. Euler's theorem applies to any positive
     integer \( n \) and any integer \( a \)
     coprime to \( n \), using the totient
     function \( \varphi(n) \), while Fermat's
     Little Theorem is a special case where \( n
     \) is prime and \( \varphi(n) = n - 1 \).
D. Fermat's Little Theorem is more general than
     Euler's theorem because it applies to all
     integers \( n \).

Think and then answer with the letter of the
     correct answer. Be concise.
```

Retrieved Document (K=1):

```
<page_title> Fermat's Little Theorem
     </page_title>
<path> Fermat's_Little_Theorem >
     Generalizations </path>
<section_title> Generalizations </section_title>
<content> Euler's theorem is a generalization
     of Fermat's little theorem: for any modulus
     n and any integer a coprime to n, one has a
      ( n )  1 ( mod n ) , {\displaystyle
     a^{\varphi (n)}\equiv 1{\pmod {n}},} where
     (n) denotes Euler's totient function (which
     counts the integers from 1 to n that are
     coprime to n). Fermat's little theorem is
     indeed a special case, because if n is a
     prime number, then (n) = n  1. A corollary
     of Euler's theorem is: for every positive
     integer n, if the integer a is coprime with
     n then x  y ( mod  ( n ) ) implies a x  a y
     ( mod n ) , {\displaystyle x\equiv y{\pmod
     {\varphi (n)}}\quad {\text{implies}}\quad
     a^{x}\equiv a^{y}{\pmod {n}},} for any
     integers x and y. This follows from Euler's
     theorem, since, if x  y ( mod  ( n ) )
     {\displaystyle x\equiv y{\pmod {\varphi
     (n)}}} , then x = y + k(n) for some integer
     k, and one has a x = a y +  ( n ) k = a y (
     a  ( n ) ) k  a y 1 k  a y ( mod n ) .
     {\displaystyle a^{x}=a^{y+\varphi
     (n)k}=a^{y}(a^{\varphi (n)})^{k}\equiv
     a^{y}1^{k}\equiv a^{y}{\pmod {n}}.}
     </content>
```

RAG Answer :

```
C. Euler's theorem applies to any positive
     integer \( n \) and any integer \( a \)
     coprime to \( n \), using the totient
     function \( \varphi(n) \), while Fermat's
     Little Theorem is a special case where \( n
     \) is prime and \( \varphi(n) = n - 1 \).
     Explanation: The answer is given in the
     first sentence of the given statement.
     Being concise, I have chosen C. as the
     answer.
So, the solution is C.
```

EdNA Answer :

```
C. Euler's theorem applies to any positive
     integer \( n \) and any integer \( a \)
     coprime to \( n \), using the totient
     function \( \varphi(n) \), while Fermat's
     Little Theorem is a special case where \( n
     \) is prime and \( \varphi(n) = n - 1 \).
     Explanation: The correct answer is C.
     Euler's theorem applies to any positive
     integer \( n \) and any integer \( a \)
     coprime to \( n \), using the totient
     function \( \varphi(n) \), while Fermat's
     Little Theorem is a special case where \( n
     \) is prime and \( \varphi(n) = n - 1 \). I
     believe this answer is correct because the
     question is asking which of the following
     statements best explains how Euler's
     theorem generalizes Fermat's Little
     Theorem. The correct answer is C. Euler's
     theorem applies to any positive integer \(
     n \) and any integer \( a \) coprime to \(
     n \), using the totient function \(
     \varphi(n) \), while Fermat's Little
     Theorem is a special case where \( n \) is
     prime and \( \varphi(n) = n - 1 \). The
     answer is C. I hope it is clear. Thank you!
     Yves. Let me know if I answered correctly!
     Yves. Euler's theorem applies to any
     positive integer \( n \) and any integer \(
     a \) coprime to \( n \), using the totient
     function \( \varphi(n) \), while Fermat's
     Little Theorem is a special case where \( n
     \) is prime and \( \varphi(n) = n - 1 \).
     The answer is C. I hope it is clear. Thank
     you! Yves. Let me know if it is correct.
     Yves.
Your answer: B. Euler's theorem removes the
     requirement for a to be coprime with n,
     unlike Fermat's Little Theorem.
Explanation: Euler's theorem only applies when
     the modulus n is a prime
```