Adapting Open-Source Language Model into STEM-Focused AI Tutor

AHYA Team
Belghmi Amine | 311528 | amine.belghmi@epfl.ch
Morchid Hamza | 327085 | hamza.morchid@epfl.ch
Belghmi Youssef | 326601 | youssef.belghmi@epfl.ch
Othmane Idrissi Oudghiri | 326432 | mohamed.idrissioudghiri@epfl.ch

Abstract

In this project, we train a large language model to serve as an AI tutor for EPFL courses, focusing on STEM reasoning. Our aim is to adapt a general-purpose LLM into a more effective educational assistant using modern post-training methods like Direct Preference Optimization, Supervised Fine-Tuning, quantization, and Retrieval-Augmented Generation.

To align the model with academic standards, we generate preference data using ChatGPT on STEM questions and apply DPO for alignment. Simultaneously, we curate a large multiple-choice dataset from public STEM sources to train the model for precise answer selection. We improve deployment through quantization and enhance the model's accuracy with a custom RAG system retrieving scientific content.

The resulting models achieve strong performance on unseen evaluation sets, outperforming the base model and confirming that high quality educational assistants can be built from open source LLMs with limited resources.

1 Introduction

Large Language Models (LLMs) have revolutionized access to information by enabling fluent, general-purpose reasoning. In education, they offer promising opportunities for personalized AI tutors that answer student questions, support self-learning, and reduce teaching workload. However, adapting these models to academic domains remains challenging, especially for tasks like multiple-choice answering and domain-specific explanations.

This project builds an AI assistant specialized for EPFL STEM courses. We explore how the open-source LLM `Qwen3-0.6B-Base` can be adapted into a domain-specific educational model using targeted post-training techniques.

- **Direct Preference Optimization:** Aligning the model with human preferences via answer comparisons generated by ChatGPT.
- **Supervised Fine-Tuning:** Training the model on a curated set of multiple-choice STEM questions from public datasets.
- **Quantization:** Compressing the model to 4-bit precision for faster and lighter inference with minimal accuracy loss.
- **Retrieval-Augmented Generation:** Enriching prompts with relevant context retrieved from scientific documents.

This modular pipeline enables the creation of effective assistants that generalize well to unseen data, illustrating how fine-tuning, preference alignment, and retrieval can be synergistically combined, even under constrained computational resources.

2 Approach

2.1 Preference Alignment via DPO

While large language models acquire broad knowledge through unsupervised pretraining, guiding their behavior toward preferred outputs remains challenging. Alignment techniques such as Reinforcement Learning from Human Feedback address this by learning a reward model from human preferences, then optimizing the language model to maximize this reward. However, RLHF involves a complex and often unstable training loop.

To address these limitations, DPO (Rafailov et al., 2023) offers a simpler, more stable alternative. It aligns the model with preference-labeled data by comparing the log-likelihoods of preferred (y_w) and rejected (y_l) responses under the current policy π_θ and a reference policy π_{ref}:

$$\mathcal{L}_{\text{DPO}} = -\mathbb{E}\left[\log \sigma\left(\beta\left(\log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)\right)\right]$$

This loss encourages the model to increase the likelihood of preferred outputs relative to the baseline. Unlike RLHF, DPO does not require learning an explicit reward function, making it more sample-efficient and easier to implement in practice.

2.2 Supervised Fine-Tuning on MCQA Tasks

To adapt the model to academic reasoning, we applied SFT on an MCQA task (Abdellatif, 2025) using diverse STEM questions in a precise format.

To investigate the impact of prompting strategy, we experimented with two training formats:

- **Without rationale:** The model is prompted with the question and answer choices only, and must complete the prompt with the correct option letter: "The following is a multiple-choice question (with answers) about knowledge and skills in advanced master's-level STEM fields." "Select the correct answer by replying with the option letter only."
- **With rationale (support):** An explanatory sentence is prepended to help the model reason before choosing the correct option: "You will be provided with an explanation to help you understand the correct answer."

Each sample is a prompt-completion pair where the model predicts the correct answer letter. Including rationales during training may help the model internalize domain reasoning, possibly improving generalization despite their absence at test time.

The model is trained to predict the correct answer by minimizing the cross-entropy loss between its output and the ground truth. Given a prompt x and correct answer token y, the loss is defined as $\mathcal{L}_{\text{SFT}} = -\log p_\theta(y \mid x)$, where p_θ is the model's predicted probability for y. At inference, the option with the highest log-probability is selected.

2.3 Quantization for Efficient Inference

To reduce memory and compute requirements during inference, we applied post-training quantization (Ouyang et al., 2022) to the fine-tuned `Qwen3-0.6B` model using the `BitsAndBytes` library. Quantization enables efficient deployment on resource-constrained devices while preserving framework compatibility. This method compresses model weights into lower-precision formats (8-bit or 4-bit), enabling faster, lighter inference without modifying the model architecture or retraining.

We tested several settings: 8-bit quantization via `load_in_8bit`, and 4-bit quantization using NF4 and FP4 formats via `load_in_4bit`. These were combined with double quantization, which adds a second quantization step on quantizer parameters to better model weight distributions in low-bit setups.

Quantized models were loaded from pretrained checkpoints without altering the training pipeline or data. Using `BitsAndBytesConfig` (Hugging Face, 2024) at load time ensured seamless integration. Despite reduced precision, the models maintain full functionality while lowering memory and storage needs, enabling deployment on lightweight devices without sacrificing reasoning quality.

2.4 Retrieval-Augmented Generation Pipeline

To improve factual accuracy, we implemented a RAG pipeline that lets the model use external scientific content during inference (Lewis et al., 2020). This is useful for STEM queries needing specific knowledge not stored in the model's parameters.

Chunks were embedded with `Alibaba-NLP/-gte-multilingual-base` (Alibaba NLP, 2024), a lightweight multilingual model with strong retrieval performance. Embeddings were normalized (Euclidean norm) and stored with their metadata.

For fast, scalable retrieval, we built a FAISS index (`IndexFlatIP`) (Johnson et al., 2024) over the normalized embeddings to enable cosine similarity search. At query time, input is encoded with the same model, normalized, and compared to the index. The top-$k = 5$ matches ($k = 5$) are returned as tuples with content and metadata.

The retrieved passages are then inserted into the prompt provided to the model. The prompt begins with a short instruction indicating that the question concerns advanced STEM topics, followed by the retrieved context chunks, each labeled sequentially. After presenting the supporting information, the actual question is included, and the model is instructed to answer by selecting only the correct option letter (A, B, C, or D).

This augmented prompting strategy allows the model to leverage relevant external knowledge dynamically retrieved, improving factual accuracy.

3 Experiments

3.1 Datasets

3.1.1 Data for the Reward Model

To train our reward model, we collected preference-labeled response pairs through a collaborative in-class annotation. Each student received 100 STEM questions from EPFL courses and used ChatGPT to generate two responses per question, applying different prompting strategies (Kojima et al., 2023) to ensure subtle quality differences without making one answer obviously superior.

To maintain this balance, we applied various prompting strategies when generating answer pairs:

- **Zero-shot vs. Chain-of-Thought:** Compare a direct answer with one generated using step-by-step reasoning ("Let's think step by step").
- **Few-shot vs. Few-shot CoT:** Contrast few-shot answers with and without explicit step-by-step rationales in the demonstrations.
- **Single-turn vs. Multi-turn Refinement:** Generate a first answer, then improve it with a follow-up prompt asking for refinement.
- **Neutral vs. Role Prompting:** Assess whether assigning a role (e.g., "You are a physicist") improves clarity, precision, or tone.

Preferred answers were chosen for correctness, completeness, clarity, capturing human preferences to learn subtle distinctions.

To strengthen training, we incorporated external data from the `argilla/ultrafeedback-binarized-preferences-cleaned` dataset (Bartolome et al., 2023), which includes approximately 60k high-quality prompt-response pairs labeled as "chosen" or "rejected". Although rich in alignment signals, it is not strictly STEM-focused.

We therefore applied an additional filtering step. Using the zero-shot `facebook/bart-large-mnli` classifier (Lewis et al., 2019), we categorized each prompt into EPFL-relevant disciplines (e.g., computer science, physics, etc.). Samples classified as "other" were discarded, yielding a filtered STEM-focused dataset of roughly 15,000 samples. This version is publicly available at `tocico28/ultrafeedback_stem_preprocessed`.

3.1.2 Data for the MCQA Model

To train our multiple-choice question answering model, we built a unified dataset of 30,000 questions by combining six public sources: SciQ, OpenBookQA, MathQA, ARC-Easy, ARC-Challenge, and MedMCQA. These span multiple STEM fields.

Compared to Milestone II, we refined the dataset by removing narrow, underrepresented sources that limited generalization. We replaced them with more clinically focused MCQA data.

- **SciQ** (11,679): factual science questions with distractors, ideal for basic STEM grounding.
- **OpenBookQA** (4,957): multi-hop questions requiring external commonsense reasoning.
- **MathQA** (6,000): subset of quantitative math problems derived from AQuA-RAT, focused on numerical reasoning and manipulation.
- **ARC-Easy** (2,140) & **-Challenge** (1,094): train sets from the AI2_ARC benchmark, with elementary and advanced science exam mcq.
- **MedMCQA** (5,000): specialized clinical questions testing advanced medical expertise.

All samples were reformatted into a consistent structure: question, 4 answer choices, and correct answer. A support field was added when available, with brief explanations or rationales, to test if training with context improves performance (Beurer-Kellner et al., 2024). This enabled prompt format experiments with and without rationales.

3.1.3 Data Used for the RAG Pipeline

We built the RAG knowledge base from a large corpus of domain-specific scientific sources:

- **Wikipedia:** Using the MediaWiki API, we crawled all pages under 22 seed categories (e.g., "Biology", "Chemistry", "Physics", "Computer Science", "Mathematics", "Engineering") up to depth 2. Each page was segmented into contiguous passages of 300 words and stored as JSON records with fields `metadata:{page_id, title}` and `text`.
- **Augmented Data from DPO Pairs:** We enriched the corpus using questions from the preference pairs dataset collected in class. Each question was issued as a Google query, and the top three URLs were retrieved. The content of these pages was extracted via `r.jina.ai`, tokenized into 300-token chunks, deduplicated, and stored with fields such as `question_id`, `from_dpo`, and `text`.

Our retrieval corpus contained around 617,000 unique passages. To meet the 52M token submission cap, we first filtered for passages containing at least one term from a curated STEM vocabulary, reducing the set to approximately 555,000 chunks. We then applied k-means clustering ($k = 170{,}000$) to the embeddings and selected, from each cluster, the chunk nearest to its centroid. This two-stage procedure reduced the corpus to the desired size.

3.2 Evaluation Methods

3.2.1 Evaluation Benchmarks

We evaluate our models on public benchmarks for DPO and MCQs, focusing on realistic STEM scenarios. We begin with synthetic demo and evaluation sets from the `zechen-nlp` namespace on Hugging Face, which are useful but limited in scope.

To further assess our reward model, we use a filtered version of the RewardBench dataset (Lambert

et al., 2024), originally designed to benchmark reward models across domains like chat, safety, and reasoning. Each entry includes a prompt and two responses labeled as "win" or "lose." Since our goal is to improve academic reasoning, we focus only on the "reasoning" subset. The final evaluation set includes 1.43k samples, available on Hugging face at `tocico28/rewardbench_filtered`.

For the MCQA component, we use a filtered subset of the MMLU dataset (Hendrycks et al., 2021), which consists of multiple-choice questions across 57 academic disciplines. Since our objective is to develop a model specialized in STEM reasoning, we select only the 21 STEM-related subjects from the test split of `cais/mmlu`. The resulting benchmark, comprising over 3.7k questions in a consistent MCQA format, is released under `youssefbelghmi/MNLP_M3_mcqa_benchmark`.

3.2.2 Evaluation Protocol

For DPO model, evaluation consists in verifying whether the model assigns a higher relative likelihood to the preferred (chosen) response than to the rejected (reject) one, compared to a reference policy (`Qwen/Qwen3-0.6B-Base`). For each prompt x, rewards are computed as log-likelihood differences between current policy π_θ and reference π_{ref}.

$$r_{\text{chosen}} = \log \pi_\theta(y_{\text{chosen}} \mid x) - \log \pi_{\text{ref}}(y_{\text{chosen}} \mid x)$$
$$r_{\text{reject}} = \log \pi_\theta(y_{\text{reject}} \mid x) - \log \pi_{\text{ref}}(y_{\text{reject}} \mid x)$$

The model is correct when $r_{\text{chosen}} > r_{\text{reject}}$, indicating alignment with the preferred answer.

For the MCQA model, evaluation selects the answer with the highest log-probability. A prediction is correct if it matches the ground-truth answer. This metric assesses the model's ability to reason among distractors. The same benchmark and evaluation are used for both the quantized and RAG-enhanced variants to ensure consistent comparison.

3.2.3 Baseline Model

Our baseline is `Qwen/Qwen3-0.6B-Base`, a dense model from the latest Qwen3 series of LLMs. It features 0.6 billion parameters and supports long-context reasoning with a maximum sequence length of 32,768 tokens (Team, 2025).

3.3 Experimental Setup

DPO Model. We trained the reward model using the `SFTTrainer` from the `trl` library (von Werra et al., 2020), which simplifies preference-based alignment. Training was done on the filtered `argilla/ultrafeedback` dataset, with 13.4k training and 1.49k validation samples, over three epochs using a learning rate of `5e-6` and a 10% warmup ratio for stable optimization. Due to hardware limits, we used a batch size of 2 per device with gradient accumulation over 8 steps (effective batch size 16), enabled gradient checkpointing to save memory, and used the default sigmoid loss from the DPO formulation. Training took around 3 hours on EPFL's izar GPU cluster.

MCQA Model. We fine-tuned the base model on our MCQA dataset using `SFTTrainer` from `trl`, with a 70/15/15 train/val/test split to balance evaluation and maximize training data. Training was limited to one epoch due to signs of early overfitting. A learning rate of `2e-5`, selected after testing multiple values, provided the best accuracy. We used a batch size of 4 with gradient accumulation over 4 steps (effective batch size 16) to train efficiently under memory constraints. Gradient checkpointing was enabled, and evaluation was run every 100 steps. The model was trained to complete prompt-answer pairs by predicting the correct letter.

RAG Model. Training followed the MCQA setup, with retrieved context chunks added to each prompt. This tested whether external scientific content improves answer accuracy and reasoning. We applied the approach to various base models.

4 Results and Analysis

4.1 Impact of DPO Alignment

We now examine the impact of preference-based fine-tuning via DPO by analyzing training loss trends, reward accuracy, and the transferability of preference-aligned models to MCQA tasks.

The figure below shows training and validation losses for the DPO model. Training loss decreases steadily, and evaluation loss follows, dropping from 0.65 to 0.52 across epochs, indicating strong generalization without overfitting.

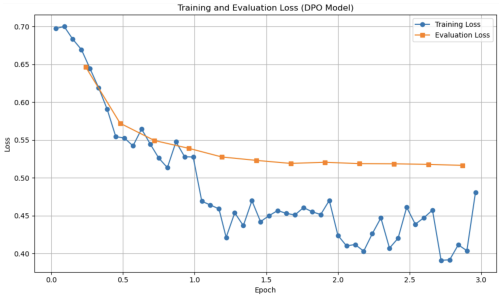

Figure 1: Training and validation loss curves of DPO

We evaluated the DPO-aligned model using the LightEval framework on the filtered `allenai/rewardbench` dataset. It achieved a reward accuracy of **0.8246** across 1,431 samples, correctly preferring the "chosen" answer in over 82% of cases, significantly outperforming the `Qwen/Qwen3-0.6B-Base` model. This high score confirms generalization of the preference signal.

The model was also evaluated on the synthetic `zechen-nlp/MNLP_dpo_evals` dataset, reaching a reward accuracy of **0.7936** on 528 samples. This close performance suggests strong alignment between our training data and these benchmarks.

To assess downstream gains, we tested the DPO-aligned model on two MCQA benchmarks: our filtered MMLU split and `zechen-nlp/MNLP_STEM_mcqa_evals` (770). In both, we compared it against the base model.

Model	Zechen	MMLU
Qwen3-0.6B Base	0.2481	0.2545
Fine-Tuned (DPO)	**0.3325**	**0.3431**

Table 1: Comparing Base and DPO Models on MCQA

The DPO model significantly outperforms the base model, confirming that preference tuning not only improves alignment but also benefits structured reasoning tasks like MCQA in STEM.

4.2 How Does the MCQA Model Perform

We evaluated several MCQA model variants to study the impact of fine-tuning strategies. The first model was fine-tuned from `Qwen/Qwen3-0.6B-Base` on our MCQA dataset without the support field (rationale). While better than the base, its performance remained modest.

To enhance reasoning, we introduced rationales by including support explanations in training prompts. This boosted performance, showing that the model benefits from explanatory supervision, even when explanations are absent at inference.

We then fine-tuned the DPO-aligned model (already trained on preference-labeled STEM responses) on the same MCQA dataset with rationale. Treated as a pre-fine-tuned model, this version combined alignment and structured task training. Final results confirmed best generalization.

Below, we show training and validation loss for the best model (DPO + MCQA with support), trained for 1 epoch to avoid overfitting. The loss curves show smooth, consistent convergence.

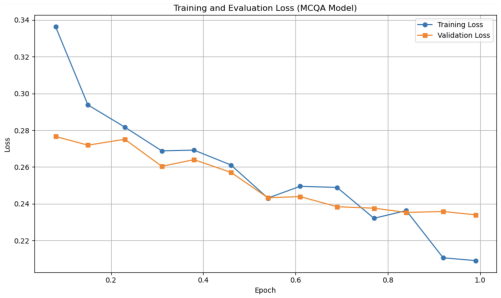

Figure 2: Training and validation loss curves of MCQA

The table below summarizes the performance of all model variants evaluated with LightEval. Benchmarks include `zechen-nlp` demo/eval datasets and our MMLU STEM benchmark.

Model	Demo	Eval	MMLU
Qwen3-0.6B Base	0.30	0.2494	0.2553
MCQA	0.36	0.316	0.3236
MCQA (support)	**0.44**	0.3519	0.3674
DPO + MCQA	0.42	**0.3818**	**0.4057**

Table 2: Accuracy of MCQA Model Variants

Results confirm that adding rationales improves accuracy, and initializing from a DPO-aligned model further boosts generalization. The DPO + MCQA (with support) variant performs best across representative benchmarks and is selected as our final MCQA model for GitHub release.

4.3 Evaluating the Quantized Model

We evaluated quantized versions of our multiple choice question answering to analyze their impact on accuracy, model size, and memory efficiency. Memory was measured during inference on the validation set of the MCQA dataset, and accuracy on the `zechen-nlp` evals dataset.

Quantization reduced memory and storage. The 4-bit NF4 model halved size (1.11GB to 506.88MB) and slightly outperformed FP16 (**0.3532** vs. **0.3519**). The 8-bit model had similar accuracy (**0.3506**) with size down to 716.88MB. All models used `BitsAndBytes` without fine-tuning. The 4-bit FP4 variant had lower accuracy due to greater quantization noise.

We also computed an efficiency score (accuracy ÷ peak VRAM), with 4-bit NF4 variant achieving the best trade-off. This confirms NF4 post-training quantization as an effective strategy for deploying educational assistants under resource constraints (Dettmers et al., 2023).

4.4 RAG-Augmented Model Improvements

We evaluated four RAG-based strategies to improve MCQA performance:

1. **Prompt Augmentation with Base Model:** Query the base Qwen3 model using the augmented prompt (retrieved context prepended).
2. **Prompt Augmentation with MCQA Fine-Tuned Model:** Query the version of `Qwen3-0.6B` already fine-tuned on MCQA, using the same RAG-augmented prompt.
3. **RAFT on Base Model:** Fine-tune the base model directly on the MCQA dataset with retrieved context included in the prompt.
4. **RAFT on MCQA Fine-Tuned Model:** Apply a 2nd fine-tuning step on the MCQA model using augmented prompts with support.

Model	Eval Accuracy
Qwen3-0.6B Base	0.2494
Strategy 1	0.30
Strategy 2	**0.40**
Strategy 3	0.30
Strategy 4	0.30

Table 3: Accuracy of RAG-Enhanced Strategies

The table above reports accuracy on the `zechen-nlp` evals dataset. The top non-retrieval model was DPO + MCQA, while the best RAG result came from combining retrieval with the support MCQA model. In this setup, retrieved context complements the rationale, achieving the highest accuracy. Loss curves confirm stable convergence.

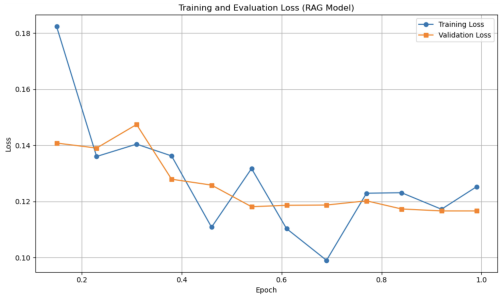

Figure 3: Training and validation loss of Strat. 2

We initially planned to evaluate RAG on MMLU but couldn't due to LightEval's high compute cost.

4.5 Failed Attempts

Class Preference Pairs. Train the DPO model on in-class preference pairs failed to yield learning. Loss stayed around 0.69, suggesting random guess. Responses were likely too similar or inconsistently labeled, resulting in near-zero reward margins.

Pre-train on Open QA. We pretrained the MCQA model on `garage-bAInd/Open-Platypus` to boost STEM reasoning, but gains were under 2% on `zechen-nlp` benchmarks, showing limited value from generic QA data without MCQA.

RAG Retrieval Limitations. Our RAG corpus lacked sources like GeeksforGeeks and Stack Overflow due to access issues. These would have provided key support for coding and math questions.

5 Ethical Considerations

Our system is trained in English and targets STEM reasoning. While adaptable to high-resource languages like French, extending to low-resource languages requires more data and multilingual alignment. Retrieval-based methods or multilingual pretraining may help bridge this gap.

When used responsibly, the system supports learning by providing clear explanations on complex topics. But risks emerge if it outputs incorrect, outdated, or biased answers, especially in sensitive fields like medicine. These risks increase when used beyond its intended domain or language.

To mitigate harm, we recommend stating model limitations, involving humans for critical outputs, and tracking confidence or citation data. As with any LLM trained on web or synthetic data, bias is a risk. Without safeguards, the model may reinforce stereotypes or exclude underrepresented groups. Fair access, regular evaluation, and careful data curation are key to avoiding educational inequality.

6 Conclusion

We built an AI assistant for STEM education by fine-tuning an open-source model via a modular pipeline. Using DPO, SFT, RAG, and quantization, we developed models that outperform the base LLM on reasoning and MCQA tasks. The best results came from combining preference-aligned training with MCQA rationale prompts and RAG.

Our work shows lightweight models can specialize effectively with limited compute, though challenges remain in RAG scaling, data quality, and multilingual generalization.

Future work could explore multilingual fine-tuning, improved retrieval filtering, and human-in-the-loop feedback to enhance alignment and accessibility in diverse learning settings.

References

Mohamed Hisham Abdellatif. 2025. Fine-tuning phi-3 for multiple-choice question answering: Methodology, results, and challenges. *arXiv preprint arXiv:2501.01588*.

Alibaba NLP. 2024. gte-multilingual-base. https://huggingface.co/Alibaba-NLP/gte-multilingual-base. Model card on Hugging Face, accessed 2025-06-11.

Alvaro Bartolome, Gabriel Martin, and Daniel Vila. 2023. Notus. https://github.com/argilla-io/notus.

Lukas Beurer-Kellner, Martina Dell, and Christian Pöhlmann. 2024. Evaluation of preference optimization methods for alignment of large language models. https://easychair.org/publications/preprint/J6XHr/open.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding.

Hugging Face. 2024. Quantization with bitsandbytes. https://huggingface.co/docs/transformers/en/quantization/bitsandbytes.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2024. Faiss: Facebook ai similarity search - indexflatip api documentation. https://faiss.ai/cpp_api/struct/structfaiss_1_1IndexFlatIP.html.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2023. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*.

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. 2024. Rewardbench: Evaluating reward models for language modeling.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2019. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *CoRR*, abs/1910.13461.

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. *arXiv preprint arXiv:2005.11401*.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training a helpful and harmless assistant with rlhf. *arXiv preprint arXiv:2112.06126*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*.

Qwen Team. 2025. Qwen3 technical report.

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.