# Poly42: Final Project Report

Achache Khalil | 300350 | khalil.acheche@epfl.ch
Ben Mustapha Ali Raed | 300392 | ali.benmustapha@epfl.ch
Briki Farah | 300386 | farah.briki@epfl.ch
Tafrika

## Abstract

This project presents the development of Poly42, a novel AI system designed to effectively resolve multiple-choice questions (MCQs) by leveraging advanced machine learning techniques. The primary objective is to create an accurate, reliable, and efficient educational tool that enhances understanding and knowledge assessment for students and educators.

To achieve this, we fine-tuned Microsoft's Phi-3 Mini 4k Instruct model using Direct Preference Optimization (DPO), which improves the model's ability to select the best answer from a set of options. Additionally, we integrated Retrieval-Augmented Generation (RAG) techniques to enhance the model's capability to retrieve and utilize relevant information, ensuring contextually appropriate and well-informed answers.

Our approach combines these methodologies to deliver precise and reliable results across various subjects. Experimental results demonstrate that Poly42 shows significant improvement in exact match accuracy when using RAG with high-quality data sources such as Wikipedia. These outcomes indicate that Poly42 can substantially enhance learning outcomes by providing accurate and contextually appropriate answers.

This paper details our experimental setup, the methodologies employed, and the ethical considerations taken into account, demonstrating the potential of Poly42 to significantly enhance learning outcomes.

## 1 Introduction

The use of artificial intelligence (AI) in educational settings is transforming how students learn and how educators assess knowledge. AI systems offer capabilities such as instant feedback, personalized learning paths, and broad access to educational resources, making them invaluable tools in modern education. Multiple-choice questions (MCQs), a widely used assessment format, benefit significantly from AI advancements due to their ability to efficiently evaluate diverse areas of knowledge.

This project focuses on the development of Poly42, an AI system designed to tackle the challenge of accurately resolving MCQs. The goal is to enhance the learning experience by providing precise, contextually appropriate answers to MCQs, thus supporting both students in their study efforts and educators in their teaching objectives.

To achieve this, we utilize Microsoft's Phi-3 Mini 4k Instruct model as the foundation, chosen for its robust natural language processing capabilities. The model is fine-tuned using Direct Preference Optimization (DPO), a technique that aligns the model's outputs with human preferences, thereby improving its decision-making accuracy. Additionally, we incorporate Retrieval-Augmented Generation (RAG) techniques, which enable the model to retrieve relevant information from external sources, ensuring well-informed and contextually accurate answers.

## 2 Related Work

**Finetuning** In the landscape of natural language processing, the adaptation of large pre-trained language models to specific tasks has traditionally relied on methods such as full fine-tuning or the use of adapters. Low-Rank Adaptation of Large Language Models (LoRA) (Hu et al., 2021) introduces a novel approach that addresses the scalability and efficiency challenges posed by these traditional methods. LoRA applies low-rank matrix adaptations directly within the Transformer layers of models, significantly reducing the number of trainable parameters while avoiding the introduction of additional computational overhead during inference. This method stands out by allowing for parameter-efficient and computationally efficient fine-tuning of large models, contrasting with other approaches like adapter layers (Houlsby et al.,

2019) and prompt tuning (Li and Liang, 2021), which either add new parameters or modify input embeddings. By embedding the changes directly into the model's existing parameters through low-rank matrices, LoRA retains the model's original architecture and offers a seamless integration of fine-tuning, setting a new standard for efficient model adaptation in NLP.

**DPO** Direct Preference Optimization (DPO) (Rafailov et al., 2023) introduces an approach to training language models (LMs) to align with human preferences without the complexities associated with reinforcement learning (RL). Traditional methods, notably reinforcement learning from human feedback (RLHF), involve training a separate reward model from human preferences and then using RL to optimize the LM to maximize this reward. DPO simplifies this process by directly optimizing the LM using a binary classification loss, effectively embedding the reward model optimization within the LM training process itself. This method contrasts with existing techniques by eliminating the need for explicit reward model fitting or RL, offering a more straightforward and computationally efficient pathway to model fine-tuning. The approach leverages the theoretical framework of preference models like the Bradley-Terry model to recalibrate the LM's parameters directly based on human preference data, bypassing the traditional multi-stage RLHF process.

**RAG** Retrieval-Augmented Generation (Lewis et al., 2021) presents an innovative approach in the field of natural language processing by integrating retrieval-augmented generation (RAG) models. This method advances the use of non-parametric memory, such as a dense vector index of Wikipedia, alongside a pre-trained parametric seq2seq transformer model, to enhance language generation tasks. This hybrid model architecture effectively combines the depth of pre-trained language models with the dynamic and up-to-date knowledge retrieval from external databases, setting a new standard in the development of knowledge-intensive applications.

RAG models contrast with and expand upon previous works like REALM (Guu et al., 2020) and ORQA (Lee et al., 2019), which also merge neural retrieval with language models but focus primarily on extractive tasks. RAG's unique contribution is its ability to perform well across a diverse set of generative tasks, providing state-of-the-art results on open-domain QA tasks and demonstrating significant improvements in generating diverse, specific, and factual language. This work exemplifies the potential of hybrid models to handle complex NLP challenges by leveraging both the generative capabilities of neural networks and the vast informational resources available externally.

**Prompt Engineering** The paper (Robinson and Wingate, 2023) on leveraging large language models (LLMs) for multiple choice question answering (MCQA) proposes a significant shift from the traditional cloze prompting method to what the authors term multiple choice prompting (MCP). This transition is rooted in the hypothesis that the common MCQA approach, which relies on generating and evaluating separate cloze statements, inherently conflates the grammaticality and commonality of text with the correctness of an answer. The authors argue that MCP, by presenting both the question and a set of answer options directly to the LLM, more effectively harnesses the model's capabilities.

This work aligns with the broader discourse on efficient utilization of LLMs, as seen in foundational model literature (Bommasani et al., 2022). It diverges from typical fine-tuning practices by adopting a direct, minimalistic training approach that could streamline the application of LLMs across varied MCQA tasks without extensive model retraining or customization. The paper's findings challenge the prevailing methodologies and suggest that LLMs' utility in MCQA has been underestimated.

## 3 Approach

### 3.1 Model Selection: Phi-3 Mini 4k Instruct

We selected Microsoft's Phi-3 Mini 4k Instruct model (Abdin et al., 2024) as the foundation for our project due to its robust capabilities in handling large-scale natural language processing tasks. This model is particularly suited for our preference learning task because of its extensive pre-training on diverse datasets, which equips it with a broad understanding of language and context. Additionally, its architecture supports efficient fine-tuning, allowing us to adapt it to specific tasks without requiring prohibitive computational resources.

### 3.2 Direct Preference Optimization (DPO)

**Motivation:** Direct Preference Optimization (DPO) (Rafailov et al., 2023) was chosen to fine-tune the model for preference learning. DPO directly optimizes the model based on preference judgments, enabling it to learn to distinguish between preferred and non-preferred responses. The three key components of our training approach are quantization, Low-Rank Adaptation (LoRA), and DPO.

**Quantization:** Quantization was implemented to reduce the model size and memory footprint, making it feasible to train on our available hardware. By representing the model's weights using 4-bit integers instead of the standard 32-bit floating-point representation, we achieved significant memory savings while maintaining a reasonable level of accuracy.

**Low-Rank Adaptation (LoRA):** LoRA (Hu et al., 2021) was employed to further optimize the model's adaptation process. It reduces the number of trainable parameters by decomposing the weight matrices into lower-rank matrices, thus making the fine-tuning process more efficient. Specifically, we targeted transformer modules such as *o_proj*, *qkv_proj*, *gate_up_proj*, and *down_proj*, and set the rank (r) to 16. This balance allowed effective learning while keeping the computational demands manageable.

**DPO Trainer:** We used the DPO trainer to implement the fine-tuning process. The DPO loss function is formulated as follows:

$$\mathcal{L}_{DPO} = -\sum_{i=1}^{N} \left[\log P_\theta(a_i^+|q_i) - \log P_\theta(a_i^-|q_i)\right]$$

where $q_i$ represents the input question, $a_i^+$ is the preferred (chosen) answer, $a_i^-$ is the non-preferred (rejected) answer, and $P_\theta$ denotes the probability assigned by the model with parameters $\theta$. The goal is to maximize the likelihood of the preferred answers while minimizing that of the non-preferred ones, thus training the model to generate outputs that align with the desired preferences.

### 3.3 Retrieval-Augmented Generation (RAG)

**Motivation for Using Llama-Index:** To enhance the model's ability to answer multiple-choice questions (MCQs), we integrated a Retrieval-Augmented Generation (RAG) system. The Llama-Index (Liu) was chosen for its efficiency and effectiveness in indexing and retrieving relevant context from large document collections, such as Wikipedia. This allows the model to access up-to-date and comprehensive information, which is crucial for generating accurate and relevant answers.

**Embedding Model and Index Creation:** To facilitate efficient retrieval, we used the BGE-Base embedding model (noa, b) for creating document embeddings. This model was selected for its high performance in generating dense vector representations of text, which are essential for accurate similarity search in the Llama-Index.

**RAG Pipeline:** The RAG pipeline involves several steps, each with a technical focus:

1. **Context Retrieval:** Using the Llama-Index, we retrieve a set of documents $\{d_i\}$ from the pre-indexed Wikipedia corpus that are relevant to the input question $q$. The retrieval process leverages cosine similarity between the question embedding and document embeddings to find the most relevant documents. We select the document with the highest score as the context.

2. **Integration with LLM:** The retrieved document $d_{\text{best}}$ is then used to create an augmented input $q_{\text{aug}}$ using a specific prompt template. The template incorporates the context into the question:

   **Prompt Template:**

   ```
   Context: {context}
   You will answer an MCQ question.
   You will be given the question,
   followed by the four possible
   answers (A, B, C, D). You will
   reply only with one character
   corresponding to the correct
   answer.
   Question: {question}
   Options:
   A. {option_1}
   B. {option_2}
   C. {option_3}
   D. {option_4}
   Answer:
   ```

3. **Final Answer Selection:** The model processes this augmented input to generate a response. To determine the final answer, we analyze the logits for the first generated token corresponding to the options (A, B, C, D).

The probability of each possible answer $a_i$ is calculated using the logits for the first generated token of each option. The answer with the highest probability is selected as the final answer:

$$a_{\text{final}} = \arg\max_{a_i} P_\theta(a_i | q_{\text{aug}}, \{a_i\})$$

Here, $P_\theta(a_i | q_{\text{aug}}, \{a_i\})$ represents the probability of each possible answer $a_i$ given the augmented input and the set of MCQ options. The model calculates the logits for the first generated token of each option to determine the most likely correct answer.

By combining these steps, our system effectively leverages both retrieval mechanisms and generative capabilities to enhance the performance on MCQs. The integration of Llama-Index with RAG allows the model to access and utilize relevant contextual information, leading to more accurate and contextually appropriate responses.

# 4 Experiments

## 4.1 Data

### 4.1.1 Data for Finetuning the Model

Various sources of preference data were used to finetune our model:

- M1 Preference Dataset: the M1 preference dataset consists of 21.5K samples where for each sample contains a question, a rejected answer and a chosen answer based on the overall scores given during the first milestone. These scores were given by Master students that are taking the MNLP course at EPFL.

- MathExchange: The preference data math stack exchange dataset (noa, c) consists of 18.7K pre-processed preference data that comes from the math Stack Exchange Data Dump. Mathematical questions were filtered using a regex based detector and chosen and rejected answers were extracted using a score corresponding to the Anthropic paper

- StackExchange: the data from this Stack Overflow Data Dump (noa, e) was pre-processed in a similar fashion to the previous dataset. Only data from forums related to subjects taught at EPFL were used (AI, Astronomy, Bioinformatics, Biology, Chemistry, Code review, CS, CS theory, Data science, Engineering, Mechanics, Physics, Robotics, Software engineering).

- GSM8K: the dataset GSM8K (Cobbe et al., 2021) consists of 8.5K grade school math word problems. These problems take between 2 and 8 reasoning steps to solve. Solutions primarily involve performing a sequence of elementary calculations using basic arithmetic operations (+ - / *) to reach the final answer. The reasoning steps followed by the final answers made up the chosen answers. Perturbations to the calculations which made the reasoning wrong and nonsensical were performed to produce the rejected answers.

- UltraFeedback: the dataset Ultrafeedback binarized (noa, 2023) consists of 63.6K preference pairs which are picked from the UltraFeedback dataset based on the mean of the 4 scores: instruction-following, truthfulness, honesty and helpfulness.

From each dataset (with the different Stack Exchange forums considered as such), 800 randomly drawn samples were taken to be in the training dataset and 200 randomly drawn samples were taken to be in the evaluation dataset. This resulted in an 18K dataset with a train/eval split of 80/20, which is uniformly drawn from the 18 available datasets.

### 4.1.2 Data for Retrieval-Augmented Generation

Two sources were scraped to constitute our RAG document database:

- Lecture Notes from MIT OpenCourseWare (noa, d): We scraped lecture notes from MIT OpenCourseWare, focusing on courses relevant to EPFL subjects. Each lecture slide was treated as an individual document after converting PDF slides into text and cleaning non-informative content. This resulted in a precise and granular retrieval database covering 603 courses with a total of 206K documents.

- Wikipedia (noa, a): we scarped Wikipedia articles in the English language that fall into categories or subcategories relevant to subjects taught at EPFL. To do so, we used this python wrapper (Majlis, 2024) for the Wikipedia API. We started from 32 categories related

to subjects taught at EPFL, and we looked at categories that are members of the original categories, and categories that are members of members of the original categories. We scraped pages that are members of this set of categories, and we split each section or subsection into its own document. This results in a total number of documents of 1.1M.

To evaluate our model's performance, Multiple Choice Questions (MCQs) data was used.

- MMLU: The dataset Massive Multitask Language Understanding (MMLU) (Hendrycks et al., 2021) is a benchmark designed to measure an LLM's multitask accuracy. The benchmark encompasses 57 subjects spanning STEM, humanities, social sciences, and additional fields. It includes questions of varying difficulty, from elementary to advanced professional levels, assessing both general knowledge and problem-solving skills. The subjects include traditional disciplines such as mathematics and history, as well as specialized areas like law and ethics. We only take samples which are part of subjects taught at EPFL. This results in a dataset of size 876.

### 4.2 Evaluation Method

Our evaluation method consists of two main components: evaluating the Direct Preference Optimization (DPO) model and evaluating the Retrieval-Augmented Generation (RAG) model.

#### 4.2.1 Evaluation of the DPO Model

To evaluate the DPO model, we used the following metrics:

- **Reward Accuracy:** This metric measures the percentage of times the model's chosen answer matches the preferred answer in the evaluation dataset.

$$\text{Reward Accuracy} = \frac{\text{Number of correct preferences}}{\text{Total number of evaluations}}$$

- **Reward Margin:** This metric measures the average difference in the reward score between the chosen and rejected answers, indicating the model's ability to distinguish high-quality answers from low-quality ones.

$$\text{Reward Margin} = \frac{1}{N}\sum_{i=1}^{N}\left(\text{Reward}(a_i^+) - \text{Reward}(a_i^-)\right)$$

In our experiments, the DPO model achieved a Reward Accuracy of 73% and a Reward Margin of 5.58, demonstrating its effectiveness in preference learning tasks.

#### 4.2.2 Evaluation of the RAG Model

The RAG model was evaluated using Exact Match (EM) on the final answer. Exact Match is a strict metric that measures the percentage of predictions that match the ground truth answer exactly.

- **Exact Match (EM):** This metric is defined as the proportion of answers that are exactly correct.

$$\text{Exact Match} = \frac{\text{Number of exact matches}}{\text{Total number of questions}}$$

The RAG model's evaluation process involved generating responses to multiple-choice questions and selecting the most likely correct answer from the given options based on the retrieved context and initial response. The performance was then measured by comparing the selected answers with the ground truth answers using the Exact Match metric. This evaluation method helps us assess the model's ability to leverage retrieved context to produce accurate answers.

### 4.3 Experimental Setup

We set out to evaluate each added component's performance in the pipeline, as well as the choice of the documents database of the RAG component. We do so by evaluating the base model and our DPO finetuned model, with or without RAG, using either the Wikipedia documents base or the MIT Lecture Notes documents base. We evaluate the RAG models that use the MIT Lecture Notes documents base on the entire MMLU dataset, and we re-evaluate RAG Models using either document bases on a smaller database (7 categories $\times$ 10 samples) due to computational constraints when using the big Wikipedia documents base.

### 4.4 Experimental Details

#### 4.4.1 Direct Preference Optimization

We employed Direct Preference Optimization (DPO) to fine-tune Microsoft's Phi-3 Mini 4k Instruct model. The fine-tuning process utilized a Tesla V100 GPU with 32GB of memory from the IZAR SCITAS cluster. To optimize within our hardware constraints, we implemented Low-Rank Adaptation (LoRA), reducing trainable parameters from 3.8 billion to 25 million. Additionally, we applied quantization techniques to further decrease the model size.

We trained the model using the preference learning dataset. The following key hyperparameters were used:

- **Number of epochs:** 3
- **batch size:** 2
- **Learning Rate:** 5e-05 with a linear scheduler
- **LoRA Configuration:** Rank = 16, $\alpha$ = 32, Dropout = 0.05
- **Quantization Configuration:** 4-bit precision with bfloat16 compute dtype
- **Optimizer:** AdamW ($\beta_1$=0.9, $\beta_2$=0.999, $\varepsilon$=1e-8)

The model and tokenizer were initialized from the pretrained checkpoint "microsoft/Phi-3-mini-4k-instruct." The tokenizer was configured with a maximum sequence length of 2048 tokens and used the unknown token for padding.

**Intermediate Results** We observed a reward accuracy of 73% and a reward margin of 5.58 on the evaluation dataset, indicating the model's enhanced ability to distinguish between high-quality and low-quality answers.

These results demonstrate that by leveraging LoRA and quantization, we effectively fine-tuned the Phi-3 Mini model within limited hardware resources, achieving performance improvements in preference learning tasks.

#### 4.4.2 Retrieval Augmented Generation

For the Retrieval Augmented Generation (RAG) component of our experimental setup, we utilized Llama-Index with the following configuration:

**Chunking Configuration**

- **Chunk Size:** 1024 tokens
- **Chunk Overlap:** 20 tokens
- **Embedding model:** `BAAI/bge-base-en-v1.5`

### 4.5 Results

#### 4.5.1 Base Model vs DPO Finetuned Model

| | Base Model | DPO Finetuned Model |
|---|---|---|
| Accuracy | 47.37% | **47.49%** |

Table 1: Accuracy using the Exact Match metric of the base model and the finetuned DPO model without adding RAG on the full MMLU evaluation dataset.

The DPO finetuned model shows a slight improvement in accuracy over the base model (47.49% vs. 47.37%). This indicates that Direct Preference Optimization (DPO) contributes to a marginal enhancement in the model's performance.

DPO might be fine-tuning aspects of the model that don't significantly impact overall accuracy but rather improve other metrics like precision, recall, or specific use-case performance. Thus, while there are enhancements, they might not substantially affect the accuracy metric.

#### 4.5.2 Models without vs with RAG

| Accuracy | Without RAG | With RAG |
|---|---|---|
| Base Model | 47.37 % | **49.09 %** |
| DPO Model | **47.49 %** | 47.37 % |

Table 2: Accuracy using the Exact Match metric of the base model and the finetuned DPO model with and without adding RAG on the full MMLU evaluation dataset.

Incorporating the Retrieval-Augmented Generation (RAG) system significantly improves the base model's accuracy (from 47.37% to 49.09%).

However, the DPO finetuned model's accuracy decreases slightly when RAG is added (from 47.49% to 47.37%). This suggests that while RAG enhances the base model's performance, it may not synergize as effectively with the DPO finetuned model.

One plausible reason for the different impacts of the Retrieval-Augmented Generation (RAG) system on the base and DPO finetuned models could be related to Model Compatibility. The base model

likely has a simpler or more general architecture that benefits from the external knowledge RAG provides, enhancing its output. In contrast, the DPO finetuned model may already be optimized for specific tasks or data, making it less receptive to or even disrupted by the integration of RAG's external information.

#### 4.5.3 Wikipedia vs MIT Lecture Notes

| Accuracy | MIT L.N. | Wikipedia |
|---|---|---|
| Base Model | 45.71 % | **48.57%** |
| DPO Model | 40 % | **47.14%** |

Table 3: Accuracy using the Exact Match metric of the base model and the finetuned DPO model using RAG with either document bases, on the small MMLU evaluation dataset.

The base model performs better with Wikipedia (48.57%) compared to MIT Lecture Notes (45.71%).

Similarly, the DPO model shows better performance with Wikipedia (47.14%) than with MIT Lecture Notes (40%).

This reinforces the finding that Wikipedia provides a more beneficial context for improving the model's accuracy in MCQA tasks compared to MIT Lecture Notes.

## 5 Analysis

Table 4 shows similar and unconclusive results when comparing between the base model (Phi3) and the DPO finetuned model (Poly42). However, the results how a strong advantage when using Wikipedia's documents, which was already the case in the average accuracy shown in table 3.

| Accuracy | MIT Poly42 | MIT Phi3 | WIKI Poly42 | WIKI Phi3 |
|---|---|---|---|---|
| Algebra | 20 % | **50 %** | 40 % | **50 %** |
| Physics | 30 % | 30 % | **40 %** | **40 %** |
| Electrical Eng. | 40% | 40% | **50%** | 40% |
| Comp. Science | 10% | 30% | **40%** | 30% |
| Mathematics | 30% | 30% | 30% | **50%** |
| Comp. Sec. | 70% | 70% | 70% | **80%** |
| ML | **80%** | 70% | 60% | 50% |

Table 4: Accuracies of the various RAG models on the different subjects of the small MMLU dataset.

A notable difference exists between the average accuracy of the different categories; we notice that subjects that demand more fact-basd and knowledge-based answers, like machine learning and computer security, perform better that the subjects where reasoning and calculations need to be performed to get the answer, like algebra.

Annex C shows some examples of the questions and their corresponding context from both sources. Overall we can see that Wikipedia's pages are more structured and constitute better context, although it does not necessarily constitute the complete context needed by the model. For example, in the second example, Yann LeCun's wikipedia page, which would have been a great context in this case, was the second page retrieved, so it was not passed to the model. It is a careful balance to be made to give just the right amount of context, as increasing the number of documents to pass would result in redundant or irrelevant documents in most cases.

## 6 Ethical considerations

### 6.1 Impact on Learning and Accessibility

As we introduce AI technology into educational settings, particularly with the goal of providing correct final answers, it's important to consider the potential impact on learning. There is a concern that having direct answers might limit how deeply students engage with their studies and affect their ability to think critically. To address this, we recommend integrating the AI system in a way that supplements traditional learning rather than replacing it. Educators can use the AI to provide hints or supplementary information, encouraging students to explore topics more deeply and engage in critical thinking.

### 6.2 Reliability of Sources

Using publicly editable sources like Wikipedia introduces concerns about the accuracy of the information our AI might use. This reliance could potentially lead to the propagation of incorrect or misleading content. To mitigate this, it is crucial that the AI references and cites verified data where possible. We suggest implementing a system that cross-references information with multiple sources, prioritizing peer-reviewed and authoritative references to ensure the accuracy and reliability of the educational content generated by the AI.

### 6.3 Potential for Errors and Misinformation

Despite our best efforts to ensure accuracy, our AI model can produce incorrect answers and potentially lead to the dissemination of false information. This risk underscores the importance of using the AI as a supplementary tool rather than a sole source of truth. Users should be encouraged to verify the AI-generated information with reliable sources and exercise critical thinking.

### 6.4 Transparency and Accountability

Transparency and accountability are key to building trust in AI systems. While our current system does not include mechanisms for transparency or accountability, it is important to consider these aspects in future iterations. Providing users with insights into how answers are generated instead of giving them a final answer and establishing a feedback mechanism to report errors or provide suggestions will help ensure that the AI operates ethically and responsibly.

## 7 Conclusion

In this project, we explored the enhancement of large-scale natural language processing (NLP) models for multiple-choice question answering (MCQA) tasks by leveraging Direct Preference Optimization (DPO) and Retrieval-Augmented Generation (RAG) techniques.

Our results indicate that while DPO provides modest improvements, the integration of RAG with high-quality external data sources such as Wikipedia can substantially enhance the performance of NLP models on MCQA tasks. Future work could focus on refining the synergy between DPO and RAG, exploring additional datasets, and further optimizing the fine-tuning process to maximize model efficiency and accuracy. These advancements hold promise for developing robust, contextually aware AI systems capable of supporting educational and knowledge-intensive applications.

## References


a. API:Main page.

b. BAAI/bge-base-en · Hugging Face.

c. prhegde/preference-data-math-stack-exchange · Datasets at Hugging Face.

d. Search | MIT OpenCourseWare | Free Online Course Materials.

e. Stack Exchange Data Dump : Stack Exchange, Inc. : Free Download, Borrow, and Streaming.

2023. argilla/ultrafeedback-binarized-preferences · Datasets at Hugging Face.

Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Qin Cai, Martin Cai, Caio César Teodoro Mendes, Weizhu Chen, Vishrav Chaudhary, Dong Chen, Dongdong Chen, Yen-Chun Chen, Yi-Ling Chen, Parul Chopra, Xiyang Dai, Allie Del Giorno, Gustavo de Rosa, Matthew Dixon, Ronen Eldan, Victor Fragoso, Dan Iter, Mei Gao, Min Gao, Jianfeng Gao, Amit Garg, Abhishek Goswami, Suriya Gunasekar, Emman Haider, Junheng Hao, Russell J. Hewett, Jamie Huynh, Mojan Javaheripi, Xin Jin, Piero Kauffmann, Nikos Karampatziakis, Dongwoo Kim, Mahoud Khademi, Lev Kurilenko, James R. Lee, Yin Tat Lee, Yuanzhi Li, Yunsheng Li, Chen Liang, Lars Liden, Ce Liu, Mengchen Liu, Weishung Liu, Eric Lin, Zeqi Lin, Chong Luo, Piyush Madan, Matt Mazzola, Arindam Mitra, Hardik Modi, Anh Nguyen, Brandon Norick, Barun Patra, Daniel Perez-Becker, Thomas Portet, Reid Pryzant, Heyang Qin, Marko Radmilac, Corby Rosset, Sambudha Roy, Olatunji Ruwase, Olli Saarikivi, Amin Saied, Adil Salim, Michael Santacroce, Shital Shah, Ning Shang, Hiteshi Sharma, Swadheen Shukla, Xia Song, Masahiro Tanaka, Andrea Tupini, Xin Wang, Lijuan Wang, Chunyu Wang, Yu Wang, Rachel Ward, Guanhua Wang, Philipp Witte, Haiping Wu, Michael Wyatt, Bin Xiao, Can Xu, Jiahang Xu, Weijian Xu, Sonali Yadav, Fan Yang, Jianwei Yang, Ziyi Yang, Yifan Yang, Donghan Yu, Lu Yuan, Chengruidong Zhang, Cyril Zhang, Jianwen Zhang, Li Lyna Zhang, Yi Zhang, Yue Zhang, Yunan Zhang, and Xiren Zhou. 2024. Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone. ArXiv:2404.14219 [cs].

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle

Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher Ré, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishnan Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. 2022. On the Opportunities and Risks of Foundation Models. ArXiv:2108.07258 [cs].

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training Verifiers to Solve Math Word Problems. ArXiv:2110.14168 [cs].

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. REALM: Retrieval-Augmented Language Model Pre-Training. ArXiv:2002.08909 [cs].

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring Massive Multitask Language Understanding. ArXiv:2009.03300 [cs].

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-Efficient Transfer Learning for NLP. ArXiv:1902.00751 [cs, stat].

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-Rank Adaptation of Large Language Models. ArXiv:2106.09685 [cs].

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent Retrieval for Weakly Supervised Open Domain Question Answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096, Florence, Italy. Association for Computational Linguistics.

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. ArXiv:2005.11401 [cs].

Xiang Lisa Li and Percy Liang. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. ArXiv:2101.00190 [cs].

Jerry Liu. llama-index: Interface between LLMs and your data.

Martin Majlis. 2024. martin-majlis/Wikipedia-API. Original-date: 2017-12-11T09:54:02Z.

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:2305.18290 [cs].

Joshua Robinson and David Wingate. 2023. LEVERAGING LARGE LANGUAGE MODELS FOR MULTIPLE CHOICE QUESTION ANSWERING.

# A AI Usage Appendix

In compliance with the project's AI policy, we outline the use of AI-based tools in our work. Specifically, we utilized two AI tools:

1. GitHub Copilot
2. ChatGPT (versions 3.5 and 4o)

### GitHub Copilot

We used GitHub Copilot to help with coding smaller tasks within the project. Copilot generated code snippets based on contextual comments and prompts. For example, we would input comments such as:

```
# List all files in the directory,
# create it if it does not exist
```

We verified the correctness of the suggested code snippets by testing them within our development environment and making necessary adjustments.

### ChatGPT (Versions 3.5 and 4.0)

ChatGPT was primarily used for paraphrasing sections of our report to enhance clarity and structure. The process involved drafting initial versions, inputting these drafts into ChatGPT for improved wording and organization, and then refining the outputs through a back-and-forth exchange. We ensured the accuracy of the paraphrased content by reviewing the outputs against our original drafts and making necessary adjustments.

### Additional Pertinent Information

The integration of AI tools like GitHub Copilot and ChatGPT was guided by a commitment to maintain the quality and originality of our work. While these tools facilitated efficiency and improved the readability of our report, all AI-generated outputs were reviewed and validated by our team to ensure compliance with academic standards and project requirements.

## B Contributions

- DPO processing and collecting datasets :**Farah**
- initial Phi3 inference and adaptation:**Raed**
- policy model implementation:**Raed & Khalil**
- Lora and Quantization:**Khalil**
- DPO trainer implementation:**Raed**
- Evaluation and Experimentation implementation:**Farah & Khalil**
- Writing M2 Report:**Raed & Khalil & Farah**
- preparing evaluation data for RAG:**Farah**
- MIT documents scraping:**Khalil**
- Wikipedia documents scraping:**Khalil & Farah**
- Implementation of RAG:**Raed**
- MCQA Prediction:**Khalil**
- Running experiments:**Khalil & Farah & Raed**
- Writing Final Report:**Farah & Raed & Khalil**

## C Model Output Examples

### C.1 Example 1: College Computer Science

**Question:** Consider a computer design in which multiple processors, each with a private cache memory, share global memory using a single bus. This bus is the critical system resource. Each processor can execute one instruction every 500 nanoseconds as long as memory references are satisfied by its local cache. When a cache miss occurs, the processor is delayed for an additional 2,000 nanoseconds. During half of this additional delay, the bus is dedicated to serving the cache miss. During the other half, the processor cannot continue, but the bus is free to service requests from other processors. On average, each instruction requires 2 memory references. On average, cache misses occur on 1 percent of references. What proportion of the capacity of the bus would a single processor consume, ignoring delays due to competition from other processors?
Options:
A. 1/50
B. 1/27
C. 1/25
D. 2/27

**Wikipedia Context:** Computer architecture, CPU cache, Cache miss
A cache miss is a failed attempt to read or write a piece of data in the cache, which results in a main memory access with much longer latency. There are three kinds of cache misses: instruction read miss, data read miss, and data write miss.
Cache read misses from an instruction cache generally cause the largest delay, because the processor, or at least the thread of execution, has to wait (stall) until the instruction is fetched from main memory. Cache read misses from a data cache usually cause a smaller delay, because instructions not dependent on the cache read can be issued and continue execution until the data is returned from main memory, and the dependent instructions can resume execution. Cache write misses to a data cache generally cause the shortest delay, because the write can be queued and there are few limitations on the execution of subsequent instructions; the processor can continue until the queue is full. For a detailed introduction to the types of misses, see cache performance measurement and metric.

**MIT L.M. Context:** Context: 6.826—Principles of Computer Systems

2002 Handout 10. Performance 5 With considerable care this performance can be achieved. On a parallel machine you can do perhaps 30 times better.3 Here are some examples of parameters that might determine the performance of a system to first order: cache hit rate, fragmentation, block size, message overhead, message latency, peak message bandwidth, working set size, ratio of disk reference time to message time. Modeling Once you have chosen the right scale, you have to break down the work at that scale into its component parts. The reason this is useful is the following principle: If a task x has parts a and b, the cost of x is the cost of a plus the cost of b, plus a system effect (caused by contention for resources) which is usually small. Most people who have been to school in the last 20 years seem not to believe this. They think the system effect is so large that knowing the cost of a and b doesn't help at all in understanding the cost of x. But they are wrong. Your goal should be to

break down the work into a small number of parts, between two and ten. Adding up the cost of the parts should give a result within 10the measured cost for the whole. If it doesn't then either you got the parts wrong (very likely), or there actually is an important system effect. This is not common, but it does happen. Such effects are always caused by contention for resources, but this takes two rather different forms: • Thrashing in a cache, because the sum of the working sets of the parts exceeds the size of the cache. The important parameter is the cache miss rate. If this is large, then the cache miss time and the working set are the things to look at. For example, SQL server on Windows NT running on a DEC Alpha 21164 in 1997 executes .25 instructions/cycle, even though the processor chip is capable of 2 instructions/cycle. The reason turns out to be that the instruction working set is much larger than the instruction cache, so that essentially every block of 4 instructions (16 bytes or one cache line) causes a cache miss, and the miss takes 64 ns, which is 16 4 ns cycles, or 4 cycles/instruction. • Clashing or queuing for a resource that serves one customer at a time (unlike a cache, which can take away the resource before the customer is done). The important parameter is the queue length. It's important to realize that a resource need not be a physical object like a CPU, a memory block, a disk drive, or a printer. Any lock in the system is a resource on which queuing can occur. Typically the physical resources are instrumented so that it's fairly easy to find the contention, but this is often not true for locks. In the Alta Vista web search engine, for example, CPU and disk utilization were fairly low but the system was saturated. It turned out that queries were acquiring a lock and then page faulting; during the page fault time lots of other queries would pile up waiting for the lock and unable to make progress. In the section on techniques we discuss how to analyze both of these situations.

3 Andrea Arpaci-Dusseau et al., High-performance sorting on networks of workstations. SigMod 97, Tucson, Arizona, May, 1999, http://now.cs.berkeley.edu/NowSort/nowSort.ps . 6.826—Principles of Computer Systems

2002 Handout 10. Performance 6 Measuring The basic strategy for measuring is to count the number of times things happen and observe how long they take. This can be done by sampling (what most profiling tools do) or by logging significant events such as procedure entries and exits. Once you have collected the data, you can use statistics or graphs to present it, or you can formulate a model of how it should be (for example, time in this procedure is a linear function of the first parameter) and look for disagreements between the model and reality.4 The latter technique is especially valuable for continuous monitoring of a running system. Without it, when a system starts performing badly in service it's very difficult to find out why.

### C.2 Example 2: Yann LeCun

**Question:** In Yann LeCun's cake, the cherry on top is
Options:
A. reinforcement learning
B. self-supervised learning
C. unsupervised learning
D. supervised learning

**Wikipedia Context:** Artificial intelligence, Machine learning, Approaches
Machine learning approaches are traditionally divided into three broad categories, which correspond to learning paradigms, depending on the nature of the "signal" or "feedback" available to the learning system:
Supervised learning: The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs.
Unsupervised learning: No labels are given to the learning algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning).
Reinforcement learning: A computer program interacts with a dynamic environment in which it must perform a certain goal (such as driving a vehicle or playing a game against an opponent). As it navigates its problem space, the program is provided feedback that's analogous to rewards, which it tries to maximize. Although each algorithm has advantages and limitations, no single algorithm works for all problems.

**Note:** The second retrieved context is Yann LeCun's Wikipedia page.

**MIT Context:** Context: 6.034 Artificial Intelligence. Copyright © 2004 by Massachusetts Institute of Technology. Slide 4.1.7 When viewed

technically, there are lots of different kinds of machine learning problems. We'll just sketch them out here, so you get an idea of their range. Slide 4.1.8 Supervised learning is the most common learning problem. Let's say you are given the weights and lengths of a bunch of individual salmon fish, and the weights and lengths of a bunch of individual tuna fish. The job of a supervised learning system would be to find a predictive rule that, given the weight and length of a fish, would predict whether it was a salmon or a tuna. Slide 4.1.9 Another, somewhat less well-specified, learning problem is clustering. Now you're given the descriptions of a bunch of different individual animals (or stars, or documents) in terms of a set of features (weight, number of legs, presence of hair, etc), and your job is to divide them into groups, or possibly into a hierarchy of groups that "makes sense". What makes this different from supervised learning is that you're not told in advance what groups the animals should be put into; just that you should find a natural grouping. Slide 4.1.10 Another learning problem, familiar to most of us, is learning motor skills, like riding a bike. We call this reinforcement learning. It's different from supervised learning because no-one explicitly tells you the right thing to do; you just have to try things and see what makes you fall over and what keeps you upright. Supervised learning is the most straightforward, prevalent, and well-studied version of the learning problem, so we are going to start with it, and spend most of our time on it. Most of the fundamental insights into machine learning can be seen in the supervised case.