Using THC to Improve Grades in Multiple Choice Questions of EPFL Courses

André Santo | 376762 | andre.loureiroespiritosanto@epfl.ch
Pedro Chaparro | 374339 | pedro.chaparro@epfl.ch
Pierre Høgenhaug | 385070 | pierre.hogenhaug@epfl.ch@epfl.ch
My Nice Long Penguin

Abstract

The recent boom in Large Language Models (LLMs) and conversational chatbots, mainly motivated by achievements such as ChatGPT, has caused many to be interested in and seek new ways to use such systems.

In parallel, an active problem that school and university students face is having their questions answered on time, at the most crucial times: days before a midterm or a final exam, when professors and TAs are the most occupied preparing the evaluation and ensuring every aspect is ready, students are intensively preparing for it by reading the material and, more importantly, practicing by solving exercises. It is at these times that students need help the most, but it is when less help can be provided.

Connecting both worlds, we present **TeachersHateChatbots (THC)**, a fine-tuned and direct-preference optimized model specialized in answering Multiple-Choice Questions (MCQ) of EPFL courses. This model is a result of intensive experimentation with different datasets and hyperparameters at every step of its development. Despite only achieving accuracies of 47%, it is an inexpensive, fast, and reliable model that achieves comparable accuracies in scientific questions to ChatGPT while having much fewer parameters. A quantized version, ideal for mobile applications and offline use, is also provided.

EPFL students can resort to this tool to, for now, provide quite accurate answers to MCQs, which should both ease the pressure put on the courses' staff as well as make students get an answer almost instantly.

1 Introduction

In recent years, there has been a very meaningful growth in the field of LLMs in multiple aspects: the investment made (Gupta), their sizes, as depicted in Figure 1, their capabilities, such as being able to generate very realistic videos (OpenAI, b) or realistic long verbal conversations (OpenAI, a), and the number of users, especially by interacting with chatbots such as ChatGPT (Mortensen).

Another present but older reality is the duality school and university students face during the days that precede important assessments: it is a time when much help is needed with the questions that are raised during the solving of exercises by the students, but also a time when professors and TAs are very occupied by preparing the evaluation. These situations put students under extreme stress, which is the exact opposite feeling they should be having, as it degrades their performance while studying and, consequently, during the test (Sue and Aziz, 2015).

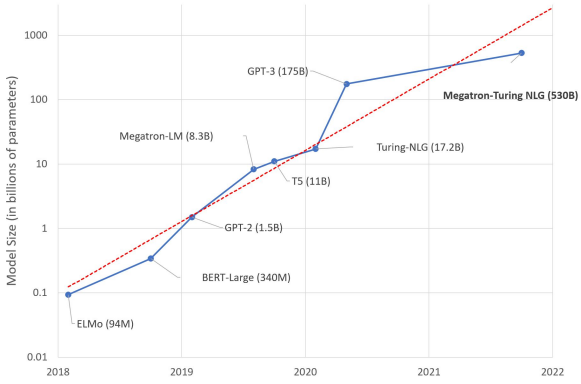


Figure 1: Growth of LLM size (Simon)

To the date of this report and to the best of our knowledge, no solution has existed to face these problems. Hence, we have used the knowledge learned from EPFL's CS-552 and developed **TeachersHateChatbots (THC)**, an LLM specialized in answering MCQs of EPFL courses. It was built on top of an already existing model, Phi2 Orange v2 (Jones, 2024), which featured 2.78B parameters and had already been instruction-tuned. To train it, we collected data from different sources, mostly of STEM knowledge from different fields. With it, we performed several benchmarks to elect the best datasets and hyperparameters to use and trained the model using DPO. Finally, to adhere to our goal, after another round of several benchmarks, we performed Supervised Fine-Tuning (SFT) so that the model's output would follow a specific format.

Our goal was also to allow the model to be used in one's computer, so we also performed a feasibility study on quantization, a compression technique that reduces the memory and computing needs of the model, analyzing how its performance would be affected. If it remained accurate, then more compact versions of the model could be used, allowing users to run the model offline on their personal computers, ideally including the ones without graphic cards, and even potentially on their smartphones.

THC is capable of answering EPFL courses' MCQs with an accuracy of 47%, succeeding in its task of helping stu-

dents study for their evaluations, while also easing the pressure on professors and TAs during the most intensive times of the semester. We consider the model to be inexpensive to run, fast to output, and still reliable, achieving comparable accuracies to ChatGPT while having much fewer parameters, while being open source. This makes the model very relevant despite its not outstanding accuracy.

The rest of this report is organized as follows. Section 2 presents related work that gives more context to this project. Then, section 3 details our approach to the problem and it is followed by section 4, where we present details about the data, evaluation method, baselines, experimental details, and results. We continue with section 5, where we provide a qualitative evaluation of our results. We move on with section 6, where we explain the ethical considerations of our work, and we conclude with section 7, where we summarize our findings, the learning experience and discuss future work.

2 Related Work

The topic of LLMs and NLP is a very broad and active research field and, as so, it is difficult to cover all the literature, as new papers are published daily. Given that, we interacted with many different papers on various topics.

Our interaction with research started on milestone 1, where we strategically chose papers to review that would benefit our work:

- Pedro reviewed the "**A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT**" (White et al., 2023) work, as a way to learn how to better interact with ChatGPT. This was meaningful for milestone 1, in order to craft the ideal prompt to maximize the quality of the produced answers to the EPFL questions.

- André chose the "**Disentangling Length from Quality in Direct Preference Optimization**" (Park et al., 2024) paper, which allowed him to learn a powerful technique to penalize lengthy outputs, which DPO is biased to produce, while also getting more knowledge of the workings of DPO, both useful for milestone 2. Although not intended, it was also relevant to better understand Robust DPO (Chowdhury et al., 2024), which was used to train the model.

- Pierre reviewed "**AlpacaFarm: A Simulation Framework for Methods that Learn from Human Feedback**" (Dubois et al., 2024), due to its potential applications in training LLMs more cost-effectively and efficiently. However, we opted not to use AlpacaFarm after realizing that our project demanded live feedback to tweak and tune model parameters, which AlpacaFarm's simulated feedback couldn't provide.

In addition, the original **DPO** paper (Rafailov et al., 2023) was needed to delve deep into the inner workings of the algorithm, so that we could understand it better and to be fully capable of understanding the expected implementation of functions of the model_dpo file.

Several pre-trained models are accompanied by research artifacts. So, to help us select the baseline model to train on, we also read the Flan-T5 (Chung et al., 2022) and Phi-1 and Phi-1.5 (Gunasekar et al., 2023; Li et al., 2023) papers.

At the point before starting training, we faced issues, first, related to the model size, and then to the speed. The model was too big to be trained in the only GPU we were given access to, so we needed to research ways to overcome that. We then found **LoRA** (Hu et al., 2021), a way to reduce the number of parameters that are trained while still retaining high quality and read both the paper and the HuggingFace documentation (Face) on how to implement it, successfully using it. Then, we were unhappy with the time the training would take, as we planned to sweep datasets and different values for the hyperparameters, which would incur in much time. To try to reduce it, we found **Unsloth** (Unsloth), a novel technique that severely boosts training time. At first, we wanted to use it but realized that, currently, it is only available to a small subset of the available models. A possible alternative was **Flash Attention 2** (Dao, 2023), which, like Unsloth, uses hardware optimizations to accelerate training and inference. However, also like Unsloth, we were unable to use it, as the GPUs of the cluster we were training on were not of the Ampere or of a more recent architecture. We did not find an alternative and accepted the long training times.

Another related work, used across all milestones of the project, was **Chain of Thought (CoT)** (Wei et al., 2023). The authors showed that the results of interactions with LLMs could be improved if they were requested to show the logic steps, e.g., by first providing an example and the thought process behind it, or just by adding the simple phrase "*Let's think step-by-step.*". These findings were used to both improve the responses of all the ChatGPT interactions as well as make the outputted MCQ answers of THC better.

As mentioned, the **Robust DPO work** (Chowdhury et al., 2024) was also relevant. As the quality of the data we collected could not be proven, we researched alternative loss functions and came across Robust DPO, which is more robust to random preference flips.

While doing some research, we came across the **Adafactor optimizer** (Shazeer and Stern, 2018), an optimization method that, upon some testing, provided better results, so it was used for the SFT training, replacing the commonly used AdamW optimizer (Loshchilov and Hutter, 2019).

To determine the best generation configuration, when choosing the final model, the works of Keskar et al. (Keskar et al., 2019) were studied, regarding the GenerationConfig 's repetition_penalty of HuggingFace (HuggingFace).

3 Approach

To achieve our goal of creating THC, we followed a plan that will be explained in the next sections, as well as our approaches to different parts of the project.

3.1 Model Selection and Strategy

Starting without a base model was a possibility, but given the short amount of time, it would have been extremely unfeasible, so we decided to use a pre-trained model.

We looked into multiple open-source pre-trained LLMs relevant to our project and compiled a list of those we

found most suitable, including models such as **Flan T5-XL** (Chung et al., 2022), due to its good performance and being already finetuned, and **LLama3** (AI@Meta, 2024), a family of models with different parameters that would allow us to better compare performance across sizes.

We ended up selecting **Phi-2** (Li et al., 2023), an autoregressive model from Microsoft, as it had a neat number of parameters, 2.7B, and showcased state-of-the-art performance in benchmarks of common sense, language understanding, and logical reasoning, which we believed aligned perfectly with our goal of answering STEM MCQs from EPFL courses. Additionally, it had not been finetuned yet, which would give us room to do so on the tasks we need.

With this baseline model, the roadmap had the following steps:

Instruction Tuning Phi-2's main categories are QA, chat, and code. So, we decided we should first instruction-tune it so that the model could learn to prioritize the pertinent information of the questions we pass and better understand and follow our MCQ format, improving its performance on that task.

DPO We would follow with DPO (Rafailov et al., 2023), where we would be aligning our model using the DPO algorithm. In this step, Phi-2 would be severely trained with pairs of data where the preference of one element over the other would be given, and it should learn from that how to better generate responses and hence select the correct choices.

SFT Finally, we would perform another round of fine-tuning, where this time we would specialize it to answer in a specific format, allowing us to easily extract the correct choice amidst the generated output.

However, we later realized it would be too ambitious to perform those three steps. So, we took a shortcut and instead used as our underlying model **Phi-2 Orange v2** (Jones, 2024), an already SFT'd version of Phi-2 that uses the ChatML template and where some of the datasets that were used for the SFT are of interest to us, such as Synthia-v1.3 (Tissera, 2023) and Verified-Camel (Wan, 2023). So, we kept the same template for training. Also, as it had been finetuned, we could immediately apply DPO.

This model uses the ChatML template (OpenAI, 2023) for interaction and, for the "user" role of the template, we have used an additional template that we have created. Example interactions with the templates can be seen on listings 9, 10, 11, and 12 of appendix D, which are further discussed on section 5.

A strategy that we employed during the final testing of our model when it generates answers to MCQs, is that, despite only outputting the correct option, THC still generates the explanation to benefit from CoT, as explained in section 2, and it is generated before the final answer is given, to also benefit from self-attention (Vaswani et al., 2023), given that the underlying model is of a transformer architecture and so the input is processed sequentially from left to right.

3.2 Data Collection

Large amounts of data were essential for both DPO and SFT training. We collected data from two different sources.

The first source actually originated from EPFL and the CS-442 (MNLP) students themselves (hence the target population of THC being EPFL students) - a set of 22k preference pairs of answers to 1.5k unique EPFL MCQs, from courses such as Machine Learning and Distributed Algorithms, was provided by the course staff. To generate those, students from the MNLP course worked together to get pairs of answers to those unique questions using ChatGPT and manually rated each question on different aspects, such as clarity and conciseness.

The second source, as that data itself was not sufficient, was open-source datasets. We carefully looked and chose a subset of the thousands of datasets available, considering our needs for STEM data and preferring more complete and well-structured datasets.

So, for DPO, we collected 53k examples from 5 different datasets - **MMLU-STEM** (Lab, 2020), **TheoremQA** (Lab, 2023), **MathQA** (Chen et al., 2019), **arxiv-physics-instruct-tune-30k** (ArtifactAI, 2023), and **Open Platypus** (Lee et al., 2023), as they were optimal for our project due to their diverse sources, covering different aspects of STEM, providing a wide range of contexts and problem-solving examples.

Then, for SFT, as we would be specializing our model to answer MCQ, we searched for datasets that were purely MCQ and ended up choosing **SciQ Dataset** (Welbl et al., 2017) and **Deepmind AQUA-RAT** (Ling et al., 2017). We chose these two specifically for their size (a combined 107k MCQ) and for the fact that they both have a distinct column that supports the correct answer, which we can leverage in the specialization of our model.

For the datasets we have found online, we added a "Topic" column to all of the entries to capture that extra information, which would become relevant for one of the goals presented in section 3.3. While **arxiv-physics-instruct-tune-30k** only consists of physics-related questions, **MMLU-STEM**, **Open-Platypus** and the EPFL data are more of a combination of topics. The **Open Platypus** came with labeled data sources for each entry, e.g. "PRM800K" or "SciBench" helping us sort out data sources irrelevant to the desired topics. To populate the topic column, we prompted ChatGPT, to identify which topic each entry was, giving it a list of topics to choose from. Since not all questions fell into the listed topics, and considering that ChatGPT might be unsure, we also considered the option "Unknown". We manually processed all entries labeled "Unknown", to either assign it a topic or to drop it entirely.

Across the datasets, we also identified MCQs without explanations for their answers. For these questions, we prompted ChatGPT to see the correct answer and explain why that answer is correct. Before creating the combined datasets, we ensured that all datasets had identical column names, facilitating the merging.

Finally, for each dataset, we created an 80/20 train/test split. For the EPFL preference pair dataset, we ensured that for each unique question, all preference pairs had to be in either the train or the test split and never in both.

3.3 Experimental Objectives

We developed a structured plan for the experiments we wanted to conduct.

During both training phases of our project, we aimed to use the optimal data and hyperparameters. So, before starting the actual long training, we performed benchmarks with different datasets and hyperparameters. Only after identifying the most favorable configurations did we commence the actual training sessions. We also performed benchmarks with the generation configuration, as it also allowed for several tweaks and ways of generating new tokens, which greatly impacts the quality, correctness and conciseness of the answers. For the datasets specifically, after setting aside data for testing, we created unique datasets by combining different proportions of the collected data, with results detailed in section 4.

Furthermore, we wanted to experiment with our quantization. As such, we planned to test our best-performing model using different 4-bit and 8-bit configurations. This would help us balance between model size reduction and performance retention. We first tested 4-bit quantization, which is a more aggressive approach for maximum model size reduction, although it might degrade performance. The settings included various quantization types like floating point 4-bit or normalized floating 4-bit, with options for nested quantization and data types like torch.bfloat16. We then tried 8-bit quantization, which is less aggressive, retaining better model performance, with features like outlier detection in activation values, exclusion of specific model parts from quantization, and possibly maintaining 16-bit floating point for main weights. Our aim was to develop a model capable of running efficiently on both personal computers and smartphones. To encompass this, we considered the trade-offs between computational resources, inference speed, and accuracy. Furthermore, the model's efficiency, measured in tokens per second, is crucial for usability. For instance, a model operating at only one token per second would fall short of practical user expectations.

4 Experiments

Now, we present in detail the experiments we have performed and all the related properties and results.

4.1 Data

For DPO, we utilized the online datasets described in section 3.2, after having set aside data for testing, we created 4 unique datasets, of sizes **20k**, **30k**, **40k**, and **54k**, using different proportions of the gathered datasets. These generated datasets allowed us to partially train the same model and evaluate which proportion yielded the best results, using it to choose the final dataset used for DPO training.
More concretely, the datasets were created as follows: We chose a 12k subset of the EPFL preference pair data (all unique questions but only 10 preference pairs per question) and used it as a base for all datasets. We then added additional rows randomly sampled from other datasets until reaching specified size thresholds. These thresholds i.e. the total number of entries, define each dataset and we considered the previously mentioned sizes. Refer to table 1 of section 4.2 for the results of training on each dataset.
To give an idea of the distribution of topics and their sources, we provide a stacked bar chart in Figure 2, where each bar is the frequency of a topic divided into data sources.

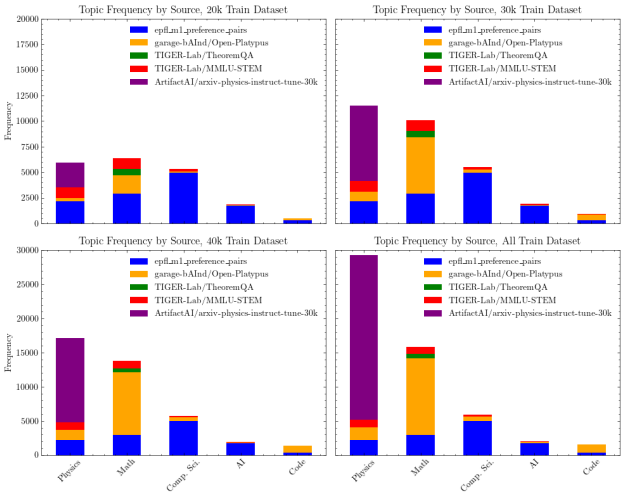


Figure 2: Train Dataset Source & Topic Split

To make it more clear, an example entry from a DPO dataset can be seen on listing 1 of appendix C.

Then, for SFT, similar to DPO, we decided to experiment with different proportions of the datasets and generate a final dataset ourselves. So, given that the SciQ dataset had significantly fewer rows compared to AQUA-RAT (approx. 13k vs. 94k), we followed a similar thought of DPO: we made an 80/20 train/test split and, for the training data, we constructed the datasets **2k**, **4k**, **6k** and **8k**, that, used 25%, 50%, 75% and 100% of the SciQ dataset, respectively, i.e., the names given to the datasets refer to the number of questions originating from SciQ. We then supplemented these with randomly sampled rows from AQUA-RAT until each dataset reached 50k entries. So, for instance, the 8k dataset contains 8k rows from the SciQ dataset and 42k from the AQUA-RAT dataset. An SFT entry can be seen on listing 2 of appendix C.

4.2 Evaluation Method

The evaluation methods varied when training THC and when comparing its final version to other models.

Evaluation during training stages

The strategies we present below were repeated for both DPO and SFT.

Initially, to select the best generated dataset, as discussed in section 4.1, we used the **Best Test Loss** i.e. the lowest loss recorded during the evaluation phase of training, which is available as *eval/loss* in Tensorboard, as well as the **Evaluation Loss**, the computed a posteriori loss of the evaluation dataset i.e. inference with the trained models was made and the loss between the generated and expected answers was calculated.

For hyperparameter fine-tuning, we relied on the **Best Test Loss** method along with a visual examination of the Tensorboard graphs. We excluded any models that demonstrated abrupt or irregular performance trends. We performed this lightweight evaluation as the number of models to compare was too high and it would be impracticable to deeply compare them. Furthermore, we deeply trained a DPO and an SFT model with the best hyperparameters found, but also trained additional models with slight varia-

tions in the hyperparameters. To choose between them and among their checkpoints, we looked at the **Accuracy** of the models, that is, how often a model correctly selects the chosen elements in the preference pairs. We also considered the evaluation loss, to confirm our observations, but did not present it in section 4.5.

Final Model Evaluation

After training the last set of models, we employed a more exhaustive set of metrics to select the final model, as presented and defined below.

Accuracy The percentage of correct answers that a model achieved in our test dataset

BLEU (Papineni et al., 2002) Commonly used for translation, measures the precision of n-grams between generated and reference texts

ROUGE (Lin, 2004) The overlap between generated and reference texts, commonly used for evaluating summaries. We have used the **ROUGE-1**, **ROUGE-2**, and **ROUGE-L** variants that, respectively, measure the matching 1-gram, 2-gram, and longest common subsequence.

BERTScore (Zhang et al., 2020) Evaluates the text similarity using BERT embeddings i.e. instead of exact matches, contextual information using embeddings is used. We considered the three scores given, **BERTScore_P** (Precision), **BERTScore_R** (Recall), and **BERTScore_F1**.

Among these metrics, we prioritized **Accuracy** because it provides a clear and direct measure of performance. A model with higher accuracy is preferred as it answers correctly more often. To use the other metrics, as the model's final parsed output is just the correct option, we use its full output, which follows the template we trained it to follow and from where we extract the correct option, and compare it between models.

4.3 Baselines

Using the previously defined metrics, we compared the final version of THC with the following models.

Phi-2 Orange v2 We decided to compare the trained versions of our model with their predecessor, the baseline model. However, we faced difficulties when calculating the accuracy, so only the remaining metrics were calculated for 200 samples.

ChatGPT Using the ChatGPT wrapper, we prompted the model to answer the questions of our total SFT test dataset, totaling 21k questions, and measured its accuracy. We used two different prompt templates, a **direct prompt**, available on listing 13 of appendix E, and a template where **CoT** is used, shown on listing 14 of appendix E. With both, only accuracy was measured.

4.4 Experimental Details

We now present the configurations and other details of our experiments. Further details on hyperparameter finetuning appear in section 4.5, where the used values can be found.

The other relevant hyperparameters are: (1) for DPO, a maximum length of 2048 tokens, 1024 for the prompt and 1024 for the target, was used (2) and we also allowed the precomputing of the log probabilities of the reference model, to save GPU memory; (3) The evaluation was made every 500 (resp. 200) steps, instead of in epochs for DPO (resp. SFT) (4) The learning rate was scheduled using a cosine function (5) A warmup of 5 steps was performed (6) We defined a threshold value of 1.0 for the gradient norm (7) For DPO, as mentioned on section 2, the Robust DPO loss function was used, to minimize the impact of incoherent data, while SFT used the default loss (8) For the optimizer, we used the 32bit paged version of AdamW across DPO and until hyperparameter finetuning on SFT, where we found that AdaFactor performed better.

4.5 Results

After establishing the foundational concepts, we present our experimental results and discuss them, as each result motivates the subsequent experiment.

DPO Dataset Selection

Given the produced datasets presented in Section 4.1, to elect the best dataset to perform DPO, we trained the model with 1350 samples of each dataset for three epochs and obtained the results presented in Table 1. We calculated the **Evaluation Loss** using a fixed set of 5000 samples from the test dataset.

Dataset Size	Best Test Loss	Eval. Loss
20k	0.43	2.4830
30k	0.4113	2.4313
40k	**0.3409**	**2.4333**
54k	0.276	2.5349

Table 1: DPO Datasets Comparison

We believe the results were not ideal, as we observed both (1) increasing test loss and (2) signs of overfitting on *Tensorboard* for some datasets. Potential explanations include: the amount of training data could have been insufficient; A perhaps too high learning rate, $1e-4$, was used, possibly making the model converge to local minima; For the best test loss, it might not be directly comparable between sizes, as different test datasets were used, each being created from the dataset being measured.

Despite these issues, we selected the **40k dataset** as our primary dataset: both the 30k and 40k were good candidates as they had the lowest evaluation losses, so the lower best test loss of the 40k dataset was the deciding factor.

DPO Hyperparameter Finetuning

Having decided on the dataset, we proceeded with hyperparameter finetuning. We considered three parameters: the **Learning Rate**, the **Beta** parameter of DPO, and the **Label Smoothing** parameter of Robust DPO. For each, we tested four different values and, using 3000 samples of the 40k dataset, trained a total of 16 versions of the model for 3 epochs each. The results can be observed in Table 2.

The selected parameters are highlighted in bold on the table. For the learning rate, we chose $1e-5$ as, looking at *Tensorboard*, both the *eval/loss* and the *train/loss* go

L.r.	B.T. Loss	Beta	B.T. Loss	L. Sm.	B.T. Loss
5e-6	0.2121	0.05	-12.41	0	0.2848
1e-5	**0.0624**	0.1	-7.322	0.05	0.0537
3e-5	-7.786	**0.3**	**-59.12**	**0.1**	**-9.94**
5e-5	-0.0819	0.5	-4.293	0.25	-241.5

Table 2: DPO Hyperparameters Comparison

down in a steady and expected manner, unlike $3e-5$ and $5e-5$. For beta, we went with the lowest value and for label smoothing, we observed the same behavior of the learning rate and so chose a sub-optimal value.

DPO Training

Having selected the ideal parameters, we began the DPO Training. Nonetheless, we also trained two alternative versions: one with beta = 0.1, and another using the 20k dataset. The resulting accuracies, using the test dataset, for different checkpoints, can be observed in Table 3.

Additionally, we plotted the evolution of the evaluation loss and accuracy during training, shown in Figure 3. For loss, we see an expected behavior on all and can observe that 40k default performed better. For the accuracy, interestingly, it follows the inverse of what was expected until the 2000 steps checkpoint mark: instead of the accuracy increasing as the loss decreases, because theoretically it would become better, it loses some accuracy. Then, this behavior is reversed and shows a trend of increase.

40k default performed best and showed better signs of success, so we elected it to be used on SFT training.

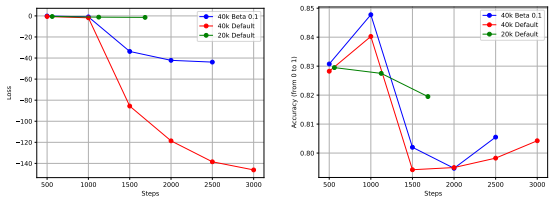
SFT Dataset Selection

Likewise to DPO, we trained our **40k default** model with the different datasets presented in Section 4.1. However, during some initial experiments, we detected that the model was barely generating EOS tokens, which meant it would be generating very large outputs, wandering to unrelated topics in the process. So, as an additional experiment, besides training our current model, we also trained the original base model, Phi-2 Orange v2 (i.e. without DPO) denoted as "*base*". Refer to the results in table 4.

Analyzing the results, we see that, for **Best Test Loss**, the value decreases the more SciQ data we use in the dataset, which can be explained by the evaluation dataset being a sample of the training dataset (10%) so, since the training dataset has less variability the less SciQ is used, the evaluation dataset will too and hence the loss will be smaller. If that is the case, then this is not a good indicator of performance. For the **Evaluation Loss**, we can observe a rather unexpected behavior: it decreased between 2k and 4k, then unexpectedly increased on 6k, and went down again on 8k. We expected it to continuously decrease and, coincidentally,

Model	Chkpt	Acc
40k default	final	0.744
40k default	1000	0.6306
40k beta 0.1	2500	0.743
40k beta 0.1	1000	0.6136
20k default	final	0.6292

Table 3: Checkpoints and accuracy on DPO policy reward



(a) Loss (b) Reward Accuracy

Figure 3: Evaluation Loss and Accuracy using 10% of `combined_40k_train.jsonl`

Dataset Size	Best Test Loss	Eval. Loss
2k	1.254	1.6583
2k (base)	1.242	1.2608
4k	**1.266**	**1.6496**
4k (base)	**1.256**	**1.2593**
6k	1.302	1.7484
6k (base)	1.279	1.2975
8k	1.329	1.7457
8k (base)	1.305	1.2979

Table 4: SFT Datasets Comparison

the first two and the last two were trained on different clusters. Suspecting a problem on the clusters or in any other configuration, as we strongly believed the loss behavior should be a linear decrease, not only moved on with **4k** as our main dataset but also performed training with **8k** as a backup dataset, in case our suspicions were correct.

Also, for the DPO analysis, it was clear that the versions without DPO outperformed the ones trained on top of it.

SFT Hyperparameter Finetuning

Despite our additional findings explained on the last chapter, we used the DPO'd version to perform finetuning, as we theorized that, later, as this version contained EPFL information, the model would have a higher performance. As hyperparameters to finetune, we considered the **Learning Rate**, the **Optimizer**, and the **Maximum Gradient Norm** and, as we found out it was faster to train for SFT than DPO, used a total of 10k samples from the **4k** dataset. The results are shown in table 5.

L.r.	B.T. Loss	Optimizer	B.T. Loss	Max. G. N.	B.T. Loss
1e-6	2.194	**AdaFactor**	**1.251**	0.6	1.265
1e-5	1.264	AdamW	1.264	**1**	**1.264**
5e-5	**1.185**	AdamW Torch Fused	1.273	1.5	1.27
1e-4	1.148	-	-	2	1.271
5e-4	1.069	-	-	-	-

Table 5: SFT Hyperparameters Comparison

For the learning rate, we decided to increase the value we were using to **5e-5**, as, likewise to section 4.5, the results on *Tensorboard* have a more predictable behavior than the ones of 1e-4 and 5e-4. We used **AdaFactor** for the optimizer, as it performed better and aligned well with our research (discussed in section 2). For the maximum gradient norm, we maintained it as **1**, as it showed the best performance.

SFT Training

With the observations in mind, we trained four versions. We also continued to explore the **40k default** configuration, and

trained it using both the **4k** and **8k** dataset, using the learning rate $5e-5$, to which we've called **sft_final_model_4k** and **sft_final_model_8k**, respectively. We trained on top of the base model, **Phi-2 Orange v2**, using also both the **4k** and **8k** datasets and learning rate $5e-5$, naming them, resp., **sft_final_model_4k_base** and **sft_final_model_8k_base**, to be a direct comparison to the previous non-base models.

Now, using the metrics discussed in section 4.2 and the baselines presented in section 4.3, we more thoroughly compare the final models, using 200 samples. The results are shown in table 6.

Following the trend, we can observe that the "base" models achieved higher accuracy than the others, and, for the other metrics, the models that underwent training with a dataset containing fewer SciQ questions (4k) performed better. However, for all metrics, the differences were slight.

So, we considered as best the **sft_final_model_4k** model, given that its accuracy is just barely lower than the "base" versions and that this discrepancy might be explained by the limited representativeness of the collected MCQ datasets, which consist of only two datasets primarily comprising questions below the college level in math and physics, which can mean that, once applied to EPFL MCQs, the model trained on top of the DPO adapters will outperform the base models. Therefore, the sft_final_model_4k model, highlighted in table 6, was elected the final version of **TeachersHateChatbots**.

When comparing it to the baseline models, we can observe that, for Phi-2 Orange v2, it performed overall just slightly better in the non-accuracy metrics. However, for ChatGPT we observe some interesting results: if ChatGPT is interacted using a simple prompt where only the correct choice is asked, THC outperforms it as its performance is equivalent to a random choice. Then, if we ask for an explanation before the answer is provided, the accuracy jumps and surpasses THC's by over 17%, which again confirms that THC's performance was not stellar, considering that it was finetuned to specifically answer these questions. In any case, this validates our decision to use CoT and prompt THC to explain before showing the final answer.

Generation Configuration

Having ultimately decided on the final model, it was time to explore more the generation part of it. Different hyperparameters can be used when generating and so we decided to try different generation configurations, to maximize the accuracy metric. Using **THC**, we tried six different configurations, presented on appendix C, and show the resulting accuracies over 100 samples on table 7.

We can see that the configurations that use beam-search clearly yield better results for answering MCQ than other sampling methods. Given the draw on the highest accuracy, we moved on with the configuration on listing 3, as it is less memory intensive by having fewer beams.

Quantization Configuration

For the quantization, we measured the performance of the 3 variations presented in Section 3.3, also available on table 6, and decided to choose the 4-bit fp4 quantization over nf4 due to its superior performance, as indicated by the higher quantized policy accuracy (0.47 vs. 0.42). FP4 proved more effective than NF4, which is tailored for normally distributed weights. Additionally, the decision to utilize 4-bit quantization instead of 8-bit was driven by the significant model size reduction achieved ($32/4 = 8$ times), which was crucial for deployment constraints despite the slight performance trade-off. We also used nested quantization and bfloat16 to ensure computational efficiency. Our model's parameters were originally stored in 32-bit format, so it roughly occupied about 10 GB. Quantization significantly reduced this size: 8-bit quantization lowered it to 2.7 GB, while 4-bit quantization compressed it further to 1.35 GB. Although 4-bit quantization fits within modern smartphones' memory limits, often 4 to 8 GB, the computational demand of running the model remained a challenge. While working on our project, we learned that Microsoft tested their newly released Phi3-mini, a 3.8 billion parameter model, on a smartphone to demonstrate its feasibility for local deployment. The phi-3-mini was successfully run on an iPhone 14 using a 4-bit quantized version that only occupied approximately 1.8 GB of memory. This test showed that the model, despite its large size, could perform efficiently on a mobile device, generating over 12 tokens per second operating entirely offline (Abdin et al., 2024). Their achievement is an inspiration, underscoring the possibility of deploying advanced AI models on everyday devices.

5 Analysis

To qualitatively evaluate our model, we generated multiple answers using hand-picked questions we found relevant and manually inspected them. Some samples are available on listings 9, 10, 11, and 12 of appendix D. In them, we can see some of the explanations are not very clear, such as listing 12 where the model seems to repeat the term "constant" on the right-hand side of the presented equalities. In any case, they all seem plausible and show signs of CoT.

Upon further analysis, we have verified that, sometimes, in the explanation of the answer, the model hallucinates and talks about unrelated content. However, in all of our manual tests, the chosen answer mentioned in the explanation is always the final answer given by the model i.e. it never selects an answer it had previously considered wrong. We believe this positive behavior is due to the self-attention mechanism and asking for the explanation before the answer, seen in section 3.1, which should enforce more coherence.

In addition, we can see that it shows some signs of deep math understanding as, on listing 10, the model incorrectly modifies an expression ending in $4(x^2+x-1)$ but correctly deduces that the expression is divisible by four. However, no further explanation is given on why that is, which would be a welcoming addition to future work.

In any case, despite our efforts to perfect the model, it only attained an accuracy of 47%, making it a "slightly better than random" model, as random sampling would achieve an average accuracy of 25%, assuming four options. That result, when compared to a naive interaction with ChatGPT, is significantly better but, if the prompt is formulated to include a CoT process, it performs worse. Despite this, we achieved a meaningful reduction in size through quantization while maintaining the same level of accuracy.

Model	Accuracy	BLEU	ROUGE-1	ROUGE-2	ROUGE-L	BERTScore_P	BERTScore_R	BERTScore_F1
sft_final_model_4k	**0.4786**	**0.5718**	**0.7383**	**0.6602**	**0.7161**	**0.8611**	**0.8519**	**0.8551**
sft_final_model_4k_base	0.4838	0.5689	0.7395	0.6628	0.7177	0.8617	0.8513	0.8548
sft_final_model_8k	0.4762	0.5605	0.7273	0.6484	0.7061	0.8562	0.8514	0.8519
sft_final_model_8k_base	0.4804	0.5585	0.7289	0.6482	0.7072	0.8569	0.8505	0.8519
Phi-2 Orange v2	-	0.5593	0.7235	0.6420	0.6980	0.8523	0.8489	0.8493
ChatGPT Direct	0.22	-	-	-	-	-	-	-
ChatGPT CoT	0.65	-	-	-	-	-	-	-
4bit_quant_model	0.47	0.5444	0.7141	0.6370	0.6924	0.8526	0.8445	0.8464
4bit_nf4_quant_model	0.42	0.5645	0.7240	0.6426	0.6981	0.8528	0.8487	0.8491
8bit_quant_model	0.42	0.5673	0.7371	0.6556	0.7143	0.8644	0.8525	0.8569

Table 6: Final models comparisons

Configuration	Accuracy
Listing 3	0.47
Listing 4	0.24
Listing 5	0.33
Listing 6	0.47
Listing 7	0.36
Listing 8	0.42

Table 7: Accuracies for different generation configurations

6 Ethical considerations

With a great model, comes great ethical responsibility.

Overall, we believe that this work does not face many ethical problems mainly because only a letter is displayed to the user. However, the base model was trained on real data, and as so, human biases are present in the datasets and therefore in the model. No adversarial training was performed, so adversarial or malicious input can trigger unintended behavior from the model but it is worth noting that the model will **always** output **exactly one** letter from the options given, so if all options are adversarial, one will always be chosen, as the model does not abstain from answering questions, answering, in the worst case, randomly. In any case, the data used for our training was all STEM-based, which means that it was more based on science facts and logic and not on "opinion topics" such as politics; This was also confirmed by the multiple checks we have made while processing the data. Then, we are not planning to retrain the model with user inputs (as prompts), so any private data that could intentionally or mistakenly be used in the prompts would not be memorized and later used by the model. However, as mentioned in section 4, we used outputs of ChatGPT, which is known to have some concerns related to ethics (Guleria et al., 2023). In any case, we do not believe it to be a concern, as (1) it was only asked either to answer STEM questions or choose between answers (2) as mentioned, we were attentive to the data while processed.

However, we have identified some possible problems that are worth discussing. A serious problem to tackle is, sadly, cheating. Despite being a fantastic tool to aid students in their studies, it could be maliciously used during evaluations to obtain the answers of the questions during, for instance, exams that allow the use of PCs. To tackle this, we suggest building a system on top of THC that integrates both the model and EPFL data. With it, users would be required to log in to use the model and they would use their institutional e-mail. This e-mail would be linked to their evaluation periods and usage of the model would be restricted during that time. Of course, circumventions to this measure would be possible, such as asking a friend who is not undergoing evaluation at that time. To detect this, a monitoring system could be imposed, where prompts would be analyzed before being processed and questions that should not be asked, such as questions of an exam happening at the time, would be detected. This extra solution would however not be perfect, as the friend could potentially reword the question and bypass the filtering, but it would severely slow down the cheating process until it would become unfeasible. However, this problem was out of scope and is left as future work.

Additionally, we did not explore, as it was also out of scope, the case where a malicious actor would use the model i.e. would pass unexpected queries, such as asking for illegal information or asking for potentially controversial information (e.g. "*Which is the best political party?*") and just the data from the underneath model. If this happened to be a concern, we would need to either reconsider the base model, as the unethical knowledge would have come from its training data, or find a way to overwrite that information, which we believe is difficult. In any case, additional training would need to be performed to fix this issue.

7 Conclusion

With this work, we've developed an academic chatbot to answer EPFL MCQs, which can mainly be used during the troublesome times of evaluations to help the students study. We enhanced a baseline model by applying DPO and SFT techniques with carefully selected and processed datasets and tested different variations along the different stages to ensure we were following the best configuration. Despite this, we only achieved a 47% accuracy, which was below what was expected, as it is not meaningfully superior to random guessing the options.

We also recognize other limitations that can be improved as a new iteration of this work. The model can be improved to also display explanations, as they are used for CoT (Wei et al., 2023) reasoning but not displayed as sometimes lack quality, so the model would require further training and tuning. Another interesting aspect to explore is to consider content from other schools and universities, providing both a greater boost to the current users of this chatbot, as it would have absorbed more knowledge, and allowing more users from other institutes to also use it.

References

Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat Behl, et al. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*.

AI@Meta. 2024. Llama 3 model card.

ArtifactAI. 2023. Arxic physics instruct tune 30k model card.

Wenhu Chen, Ming Yin, Max Ku, Pan Lu, Yixin Wan, Xueguang Ma, Jianyu Xu, Xinyi Wang, and Tony Xia. 2019. Mathqa model card.

Sayak Ray Chowdhury, Anush Kini, and Nagarajan Natarajan. 2024. Provably robust dpo: Aligning language models with noisy feedback.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models.

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning.

Yann Dubois, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2024. Alpacafarm: A simulation framework for methods that learn from human feedback.

Hugging Face. Lora.

Ankita Guleria, Kewal Krishan, Vishal Sharma, and Tanuj Kanchan. 2023. Chatgpt: ethical concerns and challenges in academics and research. *The Journal of Infection in Developing Countries*, 17:1292–1299.

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Harkirat Singh Behl, Xin Wang, Sébastien Bubeck, Ronen Eldan, Adam Tauman Kalai, Yin Tat Lee, and Yuanzhi Li. 2023. Textbooks are all you need.

Piyush Gupta. How venture capital is investing in ai in the top five global economies — and shaping the ai ecosystem.

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.

HuggingFace. Text generation.

Rhys Jones. 2024. Phi-2 orange v2 model card.

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation.

TIGER Lab. 2020. Mmlu-stem model card.

TIGER Lab. 2023. Theoremqa model card.

Ariel N. Lee, Cole J. Hunter, and Nataniel Ruiz. 2023. Open platypus model card.

Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. 2023. Textbooks are all you need ii: phi-1.5 technical report.

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

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *arXiv preprint arXiv:1705.04146*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization.

Oskar Mortensen. Number of users of chatgpt.

OpenAI. a. Openai's gpt-4o.

OpenAI. b. Openai's sora.

OpenAI. 2023. Chatml documentation.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, page 311–318, USA. Association for Computational Linguistics.

Ryan Park, Rafael Rafailov, Stefano Ermon, and Chelsea Finn. 2024. Disentangling length from quality in direct preference optimization.

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.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost.

Julien Simon. Large langual models growth.

Hui Sue and Zoriah Aziz. 2015. Assessing stress among undergraduate pharmacy students in university of malaya. *Indian Journal of Pharmaceutical Education and Research*, 49:99–105.

Migel Tissera. 2023. Synthia v1.3 dataset card.

Unsloth. Unsloth.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need.

Fanqi Wan. 2023. Verified camel dataset card.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. *CoRR*, abs/1707.06209.

Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C. Schmidt. 2023. A prompt pattern catalog to enhance prompt engineering with chatgpt.

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

Appendix

A AI Usage

AI-based tools were sometimes used during the development of this project.

One use case was, at some points, asking for clarification on documentation or methods to use. At times, we found some potentially interesting parameters to pass to objects of libraries we used, however, upon checking the documentation, it was sometimes too scarce or confusing. To clarify it, we first researched online but, in some cases, we did not find clarifying information so asked ChatGPT to provide more context on it. We would not ask directly for an explanation, as sometimes ChatGPT could provide wrong but true-looking answers: instead, we would ask for information like "*Where is this method from?*", or "*Is this parameter inherited from another class?*", in hopes that we could find pointers to more reliable information. Only then, if we did not get a positive answer, would we ask ChatGPT for an explanation and a guide on how to use the method or parameter. Then, we would test it before actually using it.

Additionally, we sometimes used ChatGPT to point us in the right direction when researching topics relevant to the project. We would then use this information to research more about the topic and read relevant research or documentation. As an example, at one point during the SFT stage, we wondered if using a different optimizer would be beneficial. We then asked ChatGPT which other alternatives we could use, given our current situation, and it provided some other optimizers. We used that knowledge to research more about them and ended up verifying that two provided optimizers could be valid alternatives. Then, we performed hyperparameter finetuning with both and ended up selecting one as the new optimizer to use.

Finally, all members of the group used the GitHub Copilot extension **solely** to accelerate the writing of boilerplate code or documentation. All the auto-completion suggestions regarding the implementation of logic blocks were either discarded or only their ideas, after being thoroughly tested, were used (i.e., the suggested code would be discarded, but the sequence of actions represented by the code would be considered, investigated, and only then manually implemented).

B Example Dataset Entries

Listing 1: DPO dataset example

```
{
    "prompt": "Question: When Renaud broke
        the world record <...> did his
        center of mass also go over his
        height?",
    "chosen": "Yes, Renaud Lavillenie's
        center of mass did go over <...> as
         <...>",
    "rejected": "In the case of Renaud
        breaking the <...>, it can be
        assumed that his center <...>"
}
```

Listing 2: SFT dataset example

```
{
    "question": "Question: During <...>,
        who proposed that the sun, <...>
        system?\n\nOptions:\nA. janus\nB.
        Newton\nC. Galileo\nD. copernicus\n
        \nAnswer:",
    "answer": "D",
    "explanation": "The Scientific
        Revolution <...>",
}
```

C Generation Configurations

Listing 3: First Generation Configuration

```
generation_config = GenerationConfig(
    eos_token_id = tokenizer.eos_token_id,
    pad_token_id = tokenizer.pad_token_id,
    max_new_tokens = 1023,
    num_beams = 10,
    num_beam_groups = 5,
    diversity_penalty = 0.5,
    repetition_penalty = 1.8,
    early_stopping = True,
    no_repeat_ngram_size = 5
)
```

Listing 4: Second Generation Configuration

```
generation_config = GenerationConfig(
    eos_token_id = tokenizer.eos_token_id,
    pad_token_id = tokenizer.pad_token_id,
    max_new_tokens = 1023,
    do_sample = True,
    temperature = 0.7,
    top_p = 0.80,
    repetition_penalty = 1.8,
    no_repeat_ngram_size = 5
)
```

Listing 5: Third Generation Configuration

```
generation_config = GenerationConfig(
    eos_token_id = tokenizer.eos_token_id,
    pad_token_id = tokenizer.pad_token_id,
    max_new_tokens = 1023,
    do_sample = True,
    temperature = 0.6,
    top_p = 0.40,
    repetition_penalty = 1.2,
    no_repeat_ngram_size = 5
)
```

Listing 6: Forth Generation Configuration

```
generation_config = GenerationConfig(
    eos_token_id = tokenizer.eos_token_id,
    pad_token_id = tokenizer.pad_token_id,
    max_new_tokens = 1023,
    num_beams = 20,
    num_beam_groups = 10,
    diversity_penalty = 1.0,
    repetition_penalty = 1.2,
    early_stopping = False,
    no_repeat_ngram_size = 5
)
```

Listing 7: Fifth Generation Configuration

```
generation_config = GenerationConfig(
    eos_token_id = tokenizer.eos_token_id,
    pad_token_id = tokenizer.pad_token_id,
    max_new_tokens = 1023,
    do_sample = True,
    temperature = 0.5,
    top_p = 0.30,
    repetition_penalty = 1.2,
    no_repeat_ngram_size = 5
)
```

Listing 8: Sixth Generation Configuration

```
generation_config = GenerationConfig(
    eos_token_id = tokenizer.eos_token_id,
    pad_token_id = tokenizer.pad_token_id,
    max_new_tokens = 1023,
    num_beams = 20,
    num_beam_groups = 10,
    diversity_penalty = 1.5,
    repetition_penalty = 1.2,
    early_stopping = False,
    no_repeat_ngram_size = 5
)
```

D Final Model Interactions

Listing 9: First example of model interactions

```
<|im_start|>system
You are a helpful EPFL chatbot.<|im_end|>
<|im_start|>user
### QUESTION
My 5 sons are enrolled in an army's platoon
     of 80 men. The General is choosing
    randomly 2 men to a suicide mission.
    What is the probability of both of the
    selected men will be two of my sons?

###OPTIONS
A:  1/80
B:  2/80
C:  5/80
D:  1/316
E:  2/316
<|im_end|>
<|im_start|>assistant
### EXPLANATION
Total number of ways to select 2 men out of
     80 = 80C2 = 3040
Number of ways to select two sons out of 5
    = 5C2 = 10
Required probability = 10/3040 = 1/316
Answer: D

### ANSWER
D<|im_end|>
```

Listing 10: Second example of model interactions

```
<|im_start|>system
You are a helpful EPFL chatbot.<|im_end|>
<|im_start|>user
### QUESTION
The difference of the squares of two
    consecutive even integers is divisible
    by which of the following integers?

###OPTIONS
A: 3
```

```
B: 4
C: 6
D: 7
E: 9
<|im_end|>
<|im_start|>assistant
### EXPLANATION
Let the two consecutive even integers be 2x
     and 2x + 2.
Then, (2x + 2)2 - (2x)2 = 4x2 + 4x - 4 = 4(
    x2 + x - 1).
This is divisible by 4.
ANSWER:B

### ANSWER
B<|im_end|>
```

Listing 11: Third example of model interactions

```
<|im_start|>system
You are a helpful EPFL chatbot.<|im_end|>
<|im_start|>user
### QUESTION
If x^2 + 9/x^2 = 42, what is the value of x
     - 3/x

###OPTIONS
A: 6
B: 25
C: 9
D: 5
E: 3
<|im_end|>
<|im_start|>assistant
### EXPLANATION
x^2+9/x^2=42
x^2(x+9)=42x^2
x^2-42x+9=0
(x-3)(x-3)=0
x=3
x-3/x=3-3/3=3-1=2
The answer is E.

### ANSWER
E<|im_end|>
```

Listing 12: Forth example of model interactions

```
<|im_start|>system
You are a helpful EPFL chatbot.<|im_end|>
<|im_start|>user
### QUESTION
A certain musical scale has has 13 notes,
    each having a different frequency,
    measured in cycles per second. In the
    scale, the notes are ordered by
    increasing frequency, and the highest
    frequency is twice the lowest. For each
     of the 12 lower frequencies, the ratio
     of a frequency to the next higher
    frequency is a fixed constant. If the
    lowest frequency is 110 cycles per
    second, then the frequency of the 7th
    note in the scale is how many cycles
    per second?

###OPTIONS
A: 110 * sqrt 2
B: 110 * sqrt (2^7)
C: 110 * sqrt (2^12)
D: 110 * the twelfth root of (2^7)
E: 110 * the seventh root of (2^12)
<|im_end|>
<|im_start|>assistant
### EXPLANATION
The lowest frequency is 110.
The highest frequency is 2*110 = 220.
The ratio of a frequency f1 to a frequency
    f2 is constant.
f1/f2 = constant
f1/220 = constant
f2/f1 = constant
220/110 = constant
2/1 = constant
2^7 = constant
The frequency of the seventh note is
    220*(2^7)/110 = 110*sqrt(2^7).
The answer is B.

### ANSWER
B<|im_end|>
```

E Baseline Model Prompts

Listing 13: Direct Prompt Template

```
Given the following question and choices,
    output the correct choice and only the
    correct choice.

{question}
```

Listing 14: CoT Prompt Template

```
Given the following question and choices,
    give an answer where you think step by
    step and then output the correct choice
     on a new line.

{question}
```

F Contributions

Stage 1

For the project proposal, a meeting was scheduled were the sections of the report were uniformly divided between the members of the group. Each member performed research on the given topic (e.g. models, datasets, ...) and presented it in a later meeting.

For the literature reviews, each member worked individually on its review and peer-reviewed the other member's reviews.

For the preference data, Pedro created a script to automate the interaction with ChatGPT, which was later improved by André. Each member rated the assigned 100 questions.

Stage 2

- André wrote the necessary functions and code in the 'model_dpo.py' file for this stage.
- Pedro wrote a script to perform SFT training, that was later improved by André. This would be used to perform two round of SFT in the model, in this stage and the next, but this stage's SFT was later discarded.
- André wrote a script to perform DPO training.
- Pierre performed the data collection, cleaning and processing for the DPO datasets.

- Pedro and André performed DPO training.
- Pierre performed the data collection, cleaning and processing for the SFT datasets, to be used in Stage 3 but needed to be available on this stage.

For the progress report, all members of the group wrote about their findings and progress, and discussing the other sections before dividing them uniformly.

Stage 3

- André wrote the remaining functions and code in the 'model_dpo.py' file.
- Pedro created a helper script to perform the evaluation of the model.
- André performed SFT training and made part of the evaluation of the models.
- Pierre performed quantization and made the remaining part of the evaluation of the models.
- Pedro, except for the quantization parts, fully wrote the report.
- Pierre wrote the quantization part of the report.