ZEPHYR: Zeroing in on Excellence, Preparing Your Rise

Mohammad Safazada | 289802 | mohammad.safazada@epfl.ch
Jérémy Chaverot | 315858 | jeremy.chaverot@epfl.ch
Sylvain Mortgat | 316678 | sylvain.mortgat@epfl.ch
ZEPHYR Team

1 Introduction

This report details the ongoing development of our AI tutor, ZEPHYR, across four key areas: dataset preparation, model selection, preliminary training results, and model quantization. To enhance ZEPHYR's performance, we implement the following fine-tuning strategy. Initially, we select two pretrained language models (PLMs), *BLOOM560M* (Workshop, 2023) and *Qwen1.5-0.5B-Chat* (Bai et al., 2023), using their established benchmark performance as our baseline. We then engage in Supervised Fine-Tuning (SFT) using the PEFT technique Low Rank Adaptation (LoRA) (Hu et al., 2021) on a variety of science-oriented datasets to enhance specific domain knowledge. Subsequently, we apply Direct Preference Optimization (DPO) to more accurately align the model's responses with educational preferences. Following DPO, we reassess ZEPHYR's performance against the initial benchmarks to evaluate the enhancements and quantify the improvements over the baseline.

2 Datasets

We provide a description of all subdatasets used for both our SFT, DPO and MCQA datasets in Appendix A. We select not only datasets relevant to technical fields that challenge reasoning skills, but also those that help mitigate biases in our model. For instance we use ethics datasets to align our model with ethical standards.

SFT Dataset $\mathcal{D}_S$ Our main SFT dataset, $\mathcal{D}_S$, is made of a set of specialized datasets used to further train the model. These include the *Cosmopedia Khan Academy dataset*, *Math College*, *Arxiv-Math*, *Magicoder*, *goat*, *MetaMathQA*, *Code Assistant*, *Lima*, *Open Platypus*, and *Math Instruct*. If S_i denotes the size of the i-th dataset, then we pick $N_i = \min\{12000, S_i\}$ samples at random. The samples are then formatted, concatenated and randomly shuffled to form a single fine-tuning dataset composed of 102 030 samples.

SFT Dataset $\mathcal{D}_L$ Initially, we consider a second SFT dataset composed of samples taken from the *Open Platypus*, *Math Instruct*, *Wiki_qa*, *Lima* and *CodeAlpaca* datasets. In section 4, we show that the results obtained with this $\mathcal{D}_L$ are worse than those obtained with $\mathcal{D}_S$. Therefore, we do not detail its composition further.

DPO Dataset To align the model, not only from an educational perspective but also from ethical considerations, we again select a diverse set of datasets. These include the *EPFL dataset*, *Orca DPO pairs*, *Truthy-DPO-v0.1*, *Stanford Human Preferences Dataset* (including the askengineers, askphysics, askscience, explainlikei'mfive subcategories), *Code Vulnerability Security DPO*, *py-DPO-v0.1*, *DPO Mix Zero Math Untoxic*, *H4rmony*, and *Exp DPO 3*. For each dataset, we randomly pick 1000 samples, and 1500 for the EPFL dataset. Processing the EPFL preference pairs involves selecting only pairs with an "overall" criterion different from AB, and excluding empty samples. As with the SFT dataset, we format, concatenate, and randomly shuffle all the samples, yielding a 12 500-sample DPO dataset. This size is found to work well in practice with 3 training epochs.

MCQA Dataset One of ZEPHYR's objectives is to answer multiple-choice questions. To achieve this, we create a specialized dataset composed of samples from the ARC (Clark et al., 2018) and MMLU (Hendrycks et al., 2020) datasets, excluding the samples from the test set. Processing these datasets results in a dataset with 103 212 samples.

3 Model

The two PLMs we select, *BLOOM560M* and *Qwen1.5-0.5B-Chat*, have a comparable number of parameters, making it meaningful to compare their performance on the same benchmarks. The similarity in their scale allows for a fair assessment of improvements and differences that arise from our fine-tuning strategies and the application of the

Model	HellaSwag	BLEU-4	ROUGE-1	ROUGE-2	ROUGE-L	Reward Accuracy
Qwen $\mathcal{M}_0$	**44.98**	28.43	46.91	17.02	25.17	–
BLOOM $\mathcal{M}_0$	36.84	7.12	20.55	4.30	6.58	–
Qwen $\mathcal{M}_1$ ($\mathcal{D}_L$)	42.15	30.64	50.43	20.20	29.01	–
Qwen $\mathcal{M}_1$ ($\mathcal{D}_S$)	**45.33**	**34.50**	**52.15**	**20.91**	**30.21**	**0.57**
BLOOM $\mathcal{M}_1$ ($\mathcal{D}_L$)	36.00	5.40	23.03	4.88	7.33	–
BLOOM $\mathcal{M}_1$ ($\mathcal{D}_S$)	36.13	6.04	31.04	7.42	8.46	–
Qwen $\mathcal{M}_2$ ($\mathcal{D}_S$)	45.01	19.40	42.26	13.06	19.38	0.52
Qwen $\mathcal{M}_0$ DPO	**45.40**	22.91	45.14	15.55	22.50	0.50

Table 1: Performance evaluations of Qwen and BLOOM models across different fine-tuning stages. The HellaSwag score represents normalized accuracy, while BLEU-4 and ROUGE metrics are evaluated on 200 samples from the GSM8K dataset. Reward accuracies are assessed using 100 samples from the EPFL preference pairs dataset.

DPO algorithm. By using models with similar capacities, we ensure that any observed differences in performance are more likely due to our training and optimization techniques rather than discrepancies in model size. Our fine-tuning strategy consists of two steps. Let $\mathcal{M}_0$ denote one of the initial PLM. First, we perform SFT on $\mathcal{M}_0$. This yields the $\mathcal{M}_1$ model, which is in turn used for DPO to obtain the $\mathcal{M}_2$ model. We evaluate the models at each stage. To evaluate our models' commonsense, we use the HellaSwag benchmark (Zellers et al., 2019). In addition, we use the ROUGE (Lin, 2004) and BLEU (Papineni et al., 2002) metrics to compare the improvements made by fine-tuning and DPO, along with reward accuracy to specifically assess the effectiveness of the DPO algorithm. The advantage of these commonly used n-gram based lies in their ease of use to evaluate generated text, though they can correlate poorly with human judgment.

4 Preliminary Training Results

Our training pipeline relies on LlamaFactory (Zheng et al., 2024), a unified, flexible and optimized framework to finetune LLMs.

SFT First, we train the Qwen and BLOOM $\mathcal{M}_0$ models on the datasets $\mathcal{D}_S$ and $\mathcal{D}_L$, each with different learning rate schedulers: constant or cosine with warmup steps (cf. Figure 1 in Appendix B.1). The details of the hyperparameter choices are provided in Table 2 in Appendix B.1. The results for BLOOM are shown in Figure 2 in Appendix B.3, and those for Qwen are shown in Figure 3 in Appendix B.3. To compute the validation loss, 5% of the dataset is held out. Interestingly, the cosine learning rate scheduler consistently leads to a lower final validation loss compared to the constant learning rate. Based on this observation, we decide to only evaluate models trained with a cosine learning rate scheduler in the subsequent experiments. Table 1 lists the performances of the $\mathcal{M}_0$ and $\mathcal{M}_1$ models on the HellaSwag (Zellers et al., 2019) benchmark and BLEU/ROUGE metrics computed over samples from the GSM8K dataset (Cobbe et al., 2021). BLOOM is consistently outperformed by Qwen. The performance of the latter on the $\mathcal{D}_S$ dataset is slightly better than that obtained with $\mathcal{D}_L$. These results lead us to select only the Qwen $\mathcal{M}_1$ model trained on $\mathcal{D}_S$ for preference alignment using DPO.

DPO To optimize for preferences, we train the model using the sample loss introduced by Rafailov et al. (2023). $\mathcal{L}_{\text{DPO}} = -\log\sigma(\beta\log f_+ - \beta\log f_-)$ where $f_\pm = \pi_\theta(y_\pm|x^*)\,/\,\pi_{\text{ref}}(y_\pm|x^*)$. Our base model is the Qwen $\mathcal{M}_0$ model. The hyperparameters used for training are listed in Table 3 in Appendix C. To assess the benefit of SFT for DPO, we train both $\mathcal{M}_0$ and $\mathcal{M}_1$ on our DPO dataset. The training loss and reward accuracy for these models are presented in Figure 4 in Appendix C. We hold out 5% of the dataset to compute a validation reward accuracy. The validation accuracies obtained for the $\mathcal{M}_0$ and $\mathcal{M}_1$ models are 0.80 and 0.79, respectively. We further evaluate the accuracy on a test set of the EPFL dataset. The results are given in Table 1. Surprisingly, models specifically trained on the DPO dataset perform worse than those trained using only SFT. So far, the Qwen $\mathcal{M}_1$ model trained on $\mathcal{D}_S$ is our best performing one.

5 Model Quantization

For quantization, we will employ the QLoRA technique (Dettmers et al., 2023), which efficiently combines quantization and low-rank adaptation to reduce the model size. The representation of our current models is in float 32 bits, and we aim to explore various quantization techniques, such as int8 or int4. QLoRA does not require a separate calibration dataset, as it utilizes the fine-tuning data for the quantization process. To evaluate the quantized model, we will compare its perplexity to that of its unquantized version on *WikiText2*, a commonly used benchmark to assess quantization efficacy.

References

ArtifactAI. 2023. arxiv-math-instruct-50k (revision ddfbf2d).

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenhang Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, K. Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Yu Bowen, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xing Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen technical report. *ArXiv*, abs/2309.16609.

Jason Baumgartner, Savvas Zannettou, Brian Keegan, Megan Squire, and Jeremy Blackburn. 2020. The pushshift reddit dataset. *ArXiv*, abs/2001.08435.

Loubna Ben Allal, Anton Lozhkov, Guilherme Penedo, Thomas Wolf, and Leandro von Werra. 2024. Cosmopedia.

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

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*, abs/2110.14168.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *ArXiv*, abs/2305.14314.

Kawin Ethayarajh, Yejin Choi, and Swabha Swayamdipta. 2022. Understanding dataset difficulty with $\mathcal{V}$-usable information. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 5988–6008. PMLR.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Xiaodong Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *ArXiv*, abs/2009.03300.

J. Edward Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *ArXiv*, abs/2106.09685.

Jorge Vallego. 2023. H4rmony (revision ee03d79).

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.

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 of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

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*, abs/2305.18290.

BigScience Workshop. 2023. Bloom: A 176b-parameter open-access multilingual language model.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. 2023. Mammoth: Building math generalist models through hybrid instruction tuning. *arXiv preprint arXiv:2309.05653*.

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

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, L. Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023. Lima: Less is more for alignment. *ArXiv*, abs/2305.11206.

A Datasets

SFT Datasets $\mathcal{D}_S$ In this section we provide a small description of each dataset we used for our final SFT dataset.

- *Khan Academy* (Ben Allal et al., 2024): made by scrapping the outlines for the courses on KhanAcademy, and generating a textbook for each with a model.
- *Math College*: math questions and exercices.
- *Arxiv-Math* (ArtifactAI, 2023): question-answer pairs derived from ArXiv abstracts. Questions are generated using the t5-base model, while the answers are generated using the GPT-3.5-turbo model.
- *goat*: simple arithmetic computations.
- *MetaMathQA*: augmented data from the training sets of GSM8K and MATH.
- *glaive-code-assistant*: generated code problems and solutions. The data is intended to be used to make models act as code assistants, and so the data is structured in a QA format where the questions are worded similar to how real users will ask code related questions.
- *Lima* (Zhou et al., 2023): contains questions from Stack Exchange, wikiHow, and the Pushshift Reddit Dataset (Baumgartner et al., 2020).
- *Open Platypus*: this dataset is focused on improving LLM logical reasoning skills and was used to train the Platypus2 models.
- *Math Instruct* (Yue et al., 2023): focuses on the hybrid use of chain-of-thought (CoT) and program-of-thought (PoT) rationales, and ensures extensive coverage of diverse mathematical fields.

All the samples from the above-mentioned datasets are formatted as a list of dictionaries, compatible with LLaMA Factory requirements.

```
[
  {
    "instruction": "...",
    "input": "...",
    "output": "..."
  }
]
```

DPO datasets In this section we provide a small description of each dataset we used for our final DPO dataset.

- *EPFL dataset*: STEM preference pairs collected using the GPT-3.5-turbo model.
- *Orca DPO pairs*: extracted from the OpenOrca dataset includes about 1 million GPT-4 completions and 3.2 million GPT-3.5 completions from the FLAN Collection. It is tabularized per the ORCA paper's distributions.
- *Truthy-DPO-v0.1*: designed to enhance the overall truthfulness of LLMs. It targets spacial, temporal awareness, and common misconceptions.
- *Stanford Human Preferences Dataset* (Ethayarajh et al., 2022): dataset of 385K collective human preferences over responses to questions/instructions in 18 different subject areas, from cooking to legal advice. The preferences are meant to reflect the helpfulness of one response over another, and are intended to be used for training RLHF reward models and NLG evaluation models
- *Code Vulnerability Security DPO*: The dataset spans an array of popular programming languages, including C++, Python, Java, JavaScript, C# and others.
- *py-DPO-v0.1*: uses the Tested-22k-Python-Alpaca dataset outputs as chosen responses and generates the "rejected" values with a mix of airoboros-l2-13b-3.1 and bagel-7b-v0.1.
- *DPO Mix Zero Math Untoxic*: contains a wide variety of situations and subjects including ethical ones.
- *H4rmony* (Jorge Vallego, 2023): prompts and completions aimed at integrating ecolinguistic principles into LLMs.
- *Exp DPO 3*: pairs of prompts and responses, including examples of inappropriate or harmful content. Each pair consists of a user request and an assistant's reply, with instances where the assistant rejects requests for illegal activities, while others mistakenly provide inappropriate guidance.

All the samples from the above-mentioned preference pair datasets are formatted as a list of dictionaries, compatible with LLaMA Factory requirements.

```
[
  {
    "instruction": "...",
    "input": "...",
    "chosen": "...",
    "rejected": "..."
  }
]
```

B Supervised Fine-Tuning

B.1 Hyperparameters

Table 2 lists the hyperparameters used to fine-tune both BLOOM and Qwen models (i.e. to go from $\mathcal{M}_0$ to $\mathcal{M}_1$). The cosine learning rate evolution is depicted in Figure 1.

Hyperparameter	Value
Batch Size	16
Learning Rate	$5 \cdot 10^{-5}$
Learning Rate Scheduler	cosine or constant
Learning Rate Warm-up Steps	1000
Gradient Accumulation steps	8
Optimizer	Adam
Number of epochs	2
Max Sequence Length	2048
LoRA α	16
LoRA r	8

Table 2: SFT hyperparameters for BLOOM560M and Qwen1.5-0.5B-Chat.

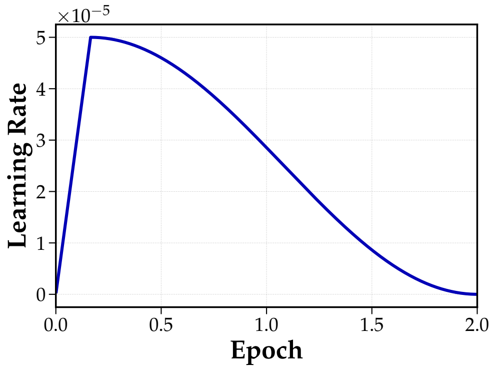

Figure 1: Cosine learning rate scheduler with warm-up steps used for SFT.

B.2 BLOOM560M

Figure B.3 depicts the training and validation losses of the BLOOM560M $\mathcal{M}_0$ model on the $\mathcal{D}_S$ and $\mathcal{D}_L$ datasets. For both datasets, the loss obtained with the cosine learning rate scheduler decreases more than that obtained with the constant learning rate.

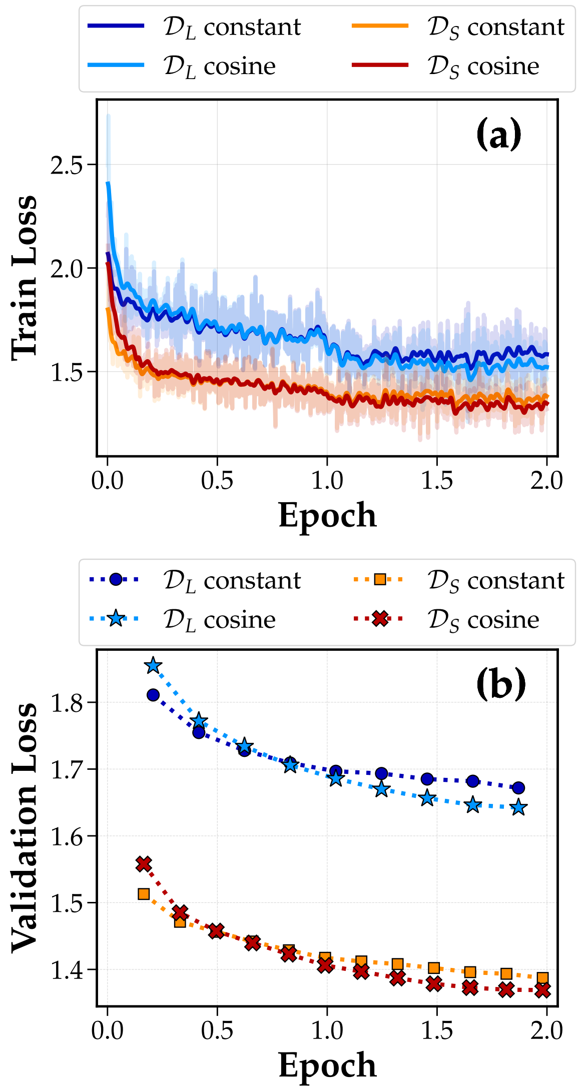

Figure 2: (a) Train and (b) validation losses of the BLOOM560M model on the $\mathcal{D}_S$ and $\mathcal{D}_L$ SFT datasets with constant and cosine learning rates.

B.3 Qwen1.5-0.5B-Chat

Figure 3 depicts the training and validation losses of the Qwen $\mathcal{M}_0$ model on the $\mathcal{D}_S$ and $\mathcal{D}_L$ datasets. For both datasets, the loss obtained with the cosine learning rate scheduler decreases more than that obtained with the constant learning rate.

In comparison with BLOOM560M models, the loss is lower with the $\mathcal{D}_L$ dataset.

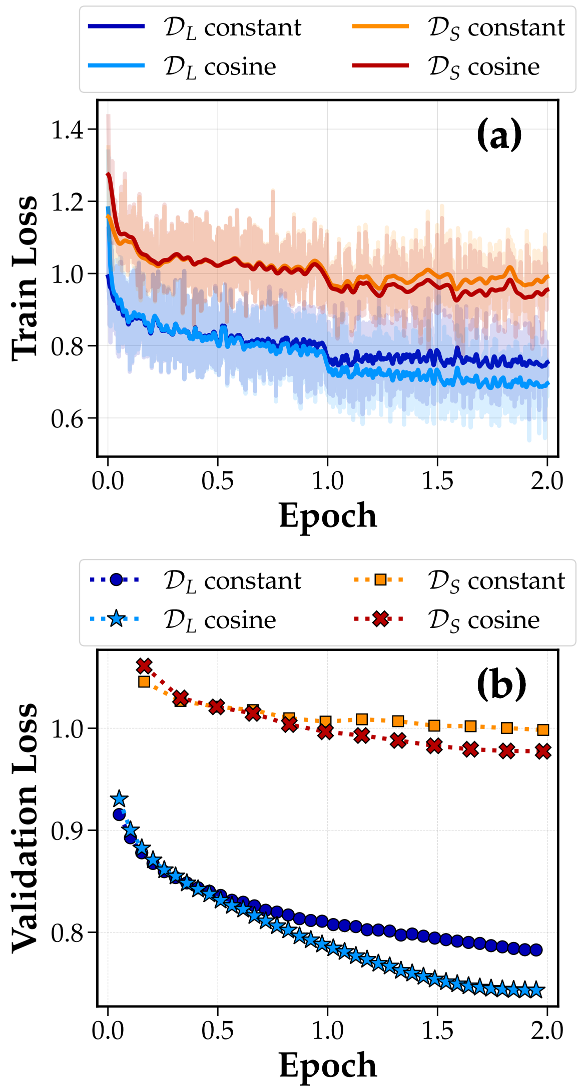

Figure 3: (a) Train and (b) validation losses of the Qwen1.5-0.5B-Chat model on the $\mathcal{D}_S$ and $\mathcal{D}_L$ SFT datasets with constant and cosine learning rates.

C Direct Preference Optimization

Table 3 lists the hyperparameters used to fine-tune the Qwen model using DPO (i.e. from $\mathcal{M}_1$ to $\mathcal{M}_2$).

Hyperparameter	Value
Batch Size	16
Learning Rate	$5 \cdot 10^{-5}$
Learning Rate Scheduler	cosine
Learning Rate Warm-up Steps	200
Gradient Accumulation steps	8
Optimizer	Adam
Number of epochs	3
Max Sequence Length	2048
LoRA α	16
LoRA r	8
DPO β	0.1

Table 3: DPO hyperparameters for the Qwen1.5-0.5B-Chat model.

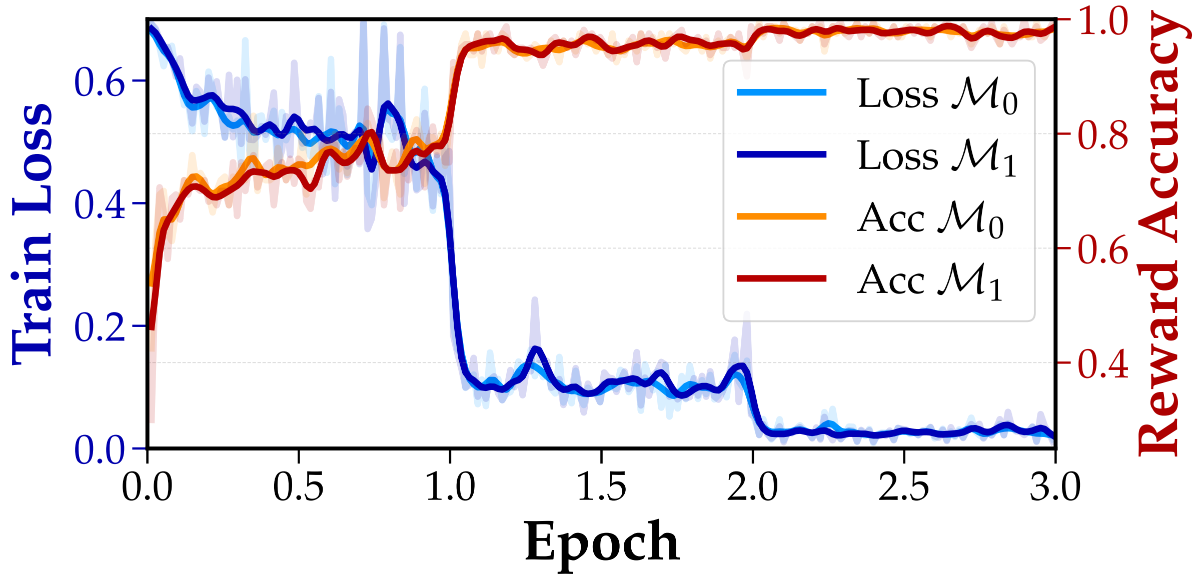

Figure 4: Training dynamics with DPO on Qwen models $\mathcal{M}_0$ and $\mathcal{M}_1$.