Self-Refine Instruction-Tuning for Aligning Reasoning in Language Models

Leonardo Ranaldi (†), **Andrè Freitas**(†,∗)
(†) Idiap Research Institute, Martigny, Switzerland
(∗)Department of Computer Science, University of Manchester, UK
[name].[surname]@idiap.ch

Abstract

The alignments of reasoning abilities between smaller and larger Language Models are largely conducted via Supervised Fine-Tuning (SFT) using demonstrations generated from robust Large Language Models (LLMs). Although these approaches deliver more performant models, they do not show sufficiently strong generalization ability as the training only relies on the provided demonstrations.

In this paper, we propose the *Self-refine Instruction-tuning* method that elicits Smaller Language Models to self-refine their abilities. Our approach is based on a two-stage process, where reasoning abilities are first transferred between LLMs and Small Language Models (SLMs) via Instruction-tuning on demonstrations provided by LLMs, and then the instructed models Self-refine their abilities through preference optimization strategies.

In particular, the second phase operates refinement heuristics based on the Direct Preference Optimization algorithm, where the SLMs are elicited to deliver a series of reasoning paths by automatically sampling the generated responses and providing rewards using ground truths from the LLMs. Results obtained on commonsense and math reasoning tasks show that this approach significantly outperforms Instruction-tuning in both in-domain and out-domain scenarios, aligning the reasoning abilities of Smaller and Larger Language Models.

1 Introduction

Previous works have demonstrated that Chain-of-Thought (CoT) prompting can improve the Large Language Models (LLMs) [1] capacity to perform complex reasoning tasks by decomposing a reasoning task into a sequence of intermediate steps (Wei et al., 2022), where the generation of multi-step controlled reasoning can improve results in commonsense (Bubeck et al., 2023), symbolic and mathematical (Gaur and Saunshi, 2023; Liu et al., 2023) reasoning datasets.

Since the size of LLMs represents an adoption barrier for many use cases and smaller models do not seem to have the same emergent reasoning abilities as LLMs, several state-of-the-art alignment approaches for solving mathematical problems have emerged, where Supervised Fine-Tuning (SFT) has been used to train Small Language Models (SLMs) using CoT annotations. However, these annotations outline the intermediate reasoning steps for solving a given problem, which consists of a reasoning pathway generated by the LLM for the specific case. This phenomenon can lead to a relatively weak generalization capacity of tuned models that have a few and limited number of samples. Indeed, there are often multiple valid CoT annotations for the same question (Cobbe et al., 2021; Zhang et al., 2023), which underlines the need for a more general CoT-based fine-tuning approach.

In this paper, we propose *Self-refine Instruction-tuning*, which is a method to enable CoT reasoning over SLMs. Our approach starts by performing Instruction-tuning on SLMs via demonstrations delivered by LLMs and then applies preference optimization based on reinforcement learning (RL) heuristics to let the SLMs refine their abilities to solve a task in a step-wise manner. Hence, proposing a teacher-student alignment method, we investigate the impact of transferring Chain-of-Thought reasoning abilities through the support of Demonstrations "taught" by LLMs to SLMs as a warm-up to the Self-refine process. Therefore, to reinforce the Instruction-tuning phase, we analyze whether preference optimization methods could strengthen students' step-wise reasoning abilities.

Complementing the foundation work of (Wang et al., 2023c,d), we introduce Self-refinement based on reinforcement learning, and in contrast to (Uesato et al., 2022; Luo et al., 2023; Luong et al.,

[1] (e.g., with more than 60B parameters (Wei et al., 2023))

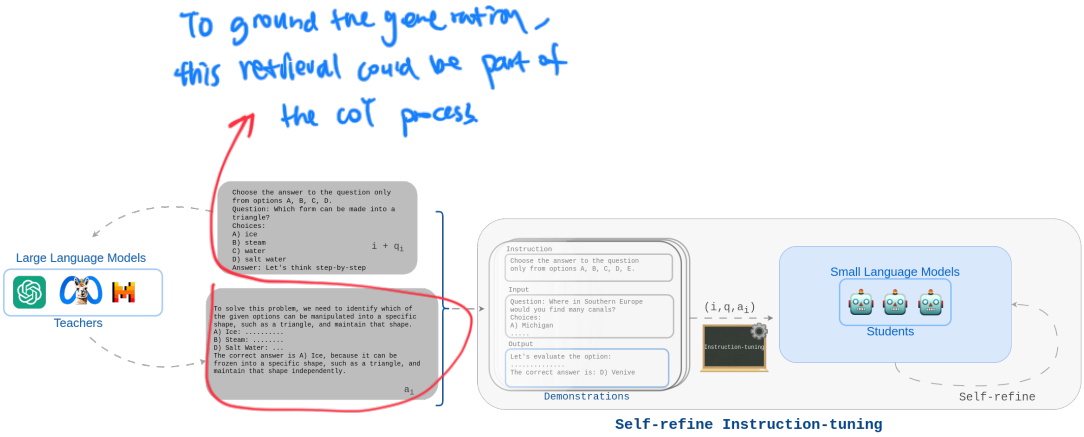

Figure 1: In Self-refine Instruction-tuning, the Demonstrations delivered by teacher models are used to align reasoning abilities in a teacher-student setting. Following the transference of step-wise reasoning knowledge via instruction tuning, the students Self-refine their abilities with the support of Direct Preference Optimization methods.

2024; Paul et al., 2024), we use an Instruction-tuning via Demonstrations approach (Ranaldi and Freitas, 2024) (i.e., a task-oriented specialization of Supervised Fine-Tuning) through which we instruct SLMs using Demonstrations delivered from different teachers prompted via a CoT mechanism.

This leads to the target research questions, which are the focus of this paper:

RQ1: How does Instruction-tuning via Demonstrations initialize the SLMs' reasoning abilities?

RQ2: What is the effect of the preference optimization algorithm on the alignment between teacher and student models?

RQ3: How much does the ability to solve tasks in a multi-step manner improve across different scenarios?

To answer these questions, we select three different SLMs: Llama2-7b, -13b (Touvron et al., 2023), Mistral-7b (Jiang et al., 2023); and three LLMs Llama2-70b, Mixtral (Jiang et al., 2024) and GPT-3.5 (OpenAI, 2023). In the teacher-student alignment phase, we use LLMs (teachers) to deliver Demonstrations at the core of the Instruction-tuning process (see Figure 1) used to instruct SLMs (students). In the Self-refine phase, the students improve their step-wise reasoning abilities via Direct Preference Optimization (DPO) (Rafailov et al., 2023). This allows the students to sample different reasoning paths and CoT Demonstrations and learn from them (Figure 1). Moreover, differently from previous works, preferences are self-generated, and there is no need for a separately trained reward model as in the previous approaches (Ouyang et al., 2022). We demonstrate the effectiveness of the proposed refinement technique in aligning teacher-student models (overcoming the differences highlighted by Ranaldi and Freitas (2024)) from the same family and in maximizing efficiency in in-domain and out-domain tasks.

Our contributions can be summarized as follows:

- We propose the Self-refined Instruction-tuning approach that is a task-oriented Supervised Fine-Tuning (SFT), which utilizes DPO heuristics to conduct a self-refinement process starting from instructed SLMs.
- We analyze the impact of different configurations of Instruction-tuning on the SLMs before and after the Self-refining phase by conducting in-depth experiments on mathematical problems and common sense question-answering tasks using Demonstrations delivered by teacher of the same family (in-family) or not (out-family). Hence, we show the downstream functionalities in both scenarios.
- Finally, we display the generalization abilities acquired via Self-refined Instruction-tuning through a systematic evaluation using Demonstrations provided by in-family and out-family teachers, both within in-domain and out-domain tasks.

2 Method

To transfer the step-wise reasoning properties from Large Language Models (LLMs) to Small Language Models (SLMs), we propose *Self-refine Instruction-tuning*, a two-step approach as shown in Figure 1. In the first phase, there is a transfer of step-wise (CoT) reasoning via Instruction-tuning, where LLMs systematically generate Demonstrations which are used by SLMs to initialize their step-wise (CoT) alignment (Section 2.1). In the second phase, the instructed SLMs Self-refine their internal CoT model via the preference optimization technique presented in Section 2.2.

2.1 Instruction-tuning Phase

A significant part of the state-of-the-art works employs standard Supervised Fine-Tuning (SFT) performed on annotations produced by a single LLM

(Large Language Model) as a mechanism to improve SLMs. In our contribution, we take a step further and use Instruction-tuning, which is a task-oriented specialization of SFT (Supervised Fine-Tuning), in coordination with a teacher-student alignment approach (detailed in Appendix A). In this phase, the SLM (student) is fine-tuned on a dataset produced by LLM (teacher) comprising a set of tuples in the form of (i, q, a_i), where i represents a specific instruction, q is the input question (e.g., math-word problem), and a_i is the expected output and CoT answers generated from the teacher in response to the instruction and input. This setup is intended to transfer to the student models foundational problem-solving abilities, emphasizing the generation of outputs that conform to the provided instructions. The CoT answer a_i is articulated as:

$$a_i = [w_1, w_2, \ldots, w_{l-1}, w_l]$$

with l indicating the sequence length. At each timestep t, the action w_t is derived from the policy $\pi_\theta(\cdot|s_t)$, where w_t can be any token from the models vocabulary, and the state s_t encapsulates the concatenation of all previously generated tokens and the optional input x if provided. The state transition is defined as:

$$s_{t+1} = \begin{cases} (x, i) & \text{if } t = 0 \\ [s_t, w_t] & \text{if } 1 \leq t \leq l \end{cases}$$

The Instruction-tuning loss function explicitly integrates the instruction i, aligning the models' learning process with the instructional context. This loss function is formulated as:

$$\mathcal{L}_{\text{inst}}(\theta) = -\mathbb{E}_{(i,q,a_i)\sim D}\left[\sum_{t=1}^{L} \log \pi_\theta(w_t|s_t, i)\right]$$

Here, π_θ is conditioned on both the state s_t, the input q, and the instruction i, ensuring that the model prioritizes instruction compliance in its output generation. This methodological shift from SFT to Instruction-tuning underlines the principle of enhancing the models' ability to accurately interpret and execute complex instructions.

2.2 Self-refinement Phase

In the second phase, the instructed SLMs (students) that have improved CoT properties via Instruction-tuning (Section 2.1) self-refine these properties with the support of Direct Preference Optimization (DPO) (Rafailov et al., 2023). This refinement can be conducted in an SFT style, relying exclusively on labeled preference data. The policy model, defined as π_θ, learns by repeatedly sampling the answers generated by teachers and students.

Direct Preference Optimization In the standard DPO approach (Rafailov et al., 2023), a human annotator ranks the outputs from a reference policy, labeling winning and losing pairs $y_w = \pi_{inst}(x)$ and $y_l = \pi_{inst}(x)$. However, we propose an optimization step via Self-generated annotation by the students π_{inst}, which, after Instruction-tuning, should have more robust performances and reliably follow the demands of the questions.

For each Demonstration (i, x, a_i), we prompt the students using the input $x = i + q$ (or $x_{CoT} = x +$ `"Let's think step by step"`) (blue block in Figure 1). Hence, for each instance within the Demonstrations we collect the **Answers** ($y_a = \pi_{\text{inst}}(x)$) that are the answers generated by the student given the input x, and the **CoT-Answers** ($y_{CoT} = \pi_{\text{inst}}(x_{CoT})$) are the answers that deliver CoT generated by the student elicited via CoT mechanism x_{CoT}.

In particular, assuming it is preferable for the model to generate responses that provide a CoT when elicited with x_{CoT} and responses when prompted with x just as the corresponding LLM teacher would do, we propose an alignment by exploiting DPO optimization. This aims to move the default style of our model (response generated by the student) towards the desired style (answers that deliver CoT). Different configurations are proposed depending on the desired result. Starting from the standard equation 1:

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{inst}}) = -\mathbb{E}_{(x,y_w,y_l)\sim D} \left[\log \sigma(M(x, y_w, y_l))\right] \quad (1)$$

where σ is the sigmoid function, and

$$M(x, y_w, y_l) = \beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{sft}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{sft}}(y_l|x)} \quad (2)$$

where β is a hyperparameter.

We propose the **Self-refine Instruction-tuning** that uses as optimization technique **DPO**$_{CoT}$ (described in details in Appendix B in Equation 3. In particular, in **DPO**$_{CoT}$ the answers that deliver a CoT response which is self-generated from the students are referred to as the preferred response.

3 Experimental Setup

In order to evaluate the proposed model, we use both commonsense and mathematical reasoning tasks (introduced in Section 3.1) that are generally used to assess the step-wise inference properties of Large Language Models (LLMs). Regarding the Self-refine Instruction-tuning on the Small Language Models (SLMs), we use the approach presented in Section 3.2.

3.1 Tasks & Datasets

In this paper, we selected different tasks that focus on reasoning tasks:

Commonsense Task We adopt two benchmarks to evaluate commonsense reasoning: CommonSenseQA (Talmor et al., 2019) (CSQA) and OpenBookQA (Mihaylov et al., 2018) (OBQA) are two multi-choice commonsense question-answering tasks.

Physical & Social Interaction Task We adopt two benchmarks to evaluate reasoning in the context of everyday situations, aiming to establish the most reasonable solution: Interaction Question Answering (PIQA) (Bisk et al., 2019) and Social Interaction Question Answering (SIQA) (Sap et al., 2019), which emphasizes people's actions and social implications.

Mathematical Task We use two math word problem benchmarks to evaluate the models of mathematical reasoning. MultiArith (Roy and Roth, 2015) covers a set of multi-step arithmetic reasoning tasks, while GSM8k (Cobbe et al., 2021) covers a set of primary school-level mathematical problems.

Additional benchmarks Finally, to evaluate the adaptability of our proposal, we conduct further analysis on two additional evaluation benchmarks: MATH (Hendrycks et al., 2021b), and MMLU (Hendrycks et al., 2021a).

Datasets Since the test split is not prescribed for all the benchmarks, we adopt the following strategy: for SIQA, PIQA, CSQA, and OBQA, we use 4000 examples with equally distributed target classes as training data and the validation versions found on huggingface as test data, while for GSM8K and MultiArith we use the full huggingface datasets. In Table 8, we report the descriptive statistics and splitting ratios, while in Table 7, we report one example for each benchmark. The supporting datasets are publicly accessible as described in Table 9.

3.2 Self-refine Instruction-tuning Pipeline

The Self-refine Instruction-tuning comprises the annotation process conducted by the LLMs teachers that are prompted in the zero-shot scenario (as shown in Table 6), as explained in Appendix A. We selected Llama-2-70 (Touvron et al., 2023), Mixtral7x8 (Jiang et al., 2024) and GPT-3.5 (OpenAI, 2023) as LLMs (teachers) and Llama2-7, -13 (Touvron et al., 2023) and Mistral-7 (Jiang et al., 2023) SMLs (students) models.

Hence, the students models are tuned, as proposed in (Taori et al., 2023) and evaluated with probing pipelines (detailed in Section 3.3). The students are instructed via Demonstrations that contain the answers generated by the teachers, as explained in Section 2.1. Downstream of the teacher-student CoT transference process, the optimization technique (proposed in Section 2.2 and detailed in Appendix B) is employed to improve alignment and self-refine the quality of the generation.

3.2.1 Models Setup

We conduct the Self-refined Instruction-tuning in two different phases. Firstly, we start with Instruction-tuning phase using QLoRA Dettmers et al. (2023). This approach allows Instruction-tuning to be performed while reducing memory usage. In particular, Dettmers et al. (2023) propose several techniques for tuning models with many parameters on GPUs with limited resources while preserving 16-bit tuning performance. We follow the training approach proposed in (Taori et al., 2023), setting four training epochs using a learning rate of 2e-5 with a 1e-4 weight decay. We use the cosine learning rate scheduler with a warm-up ratio of 0.03. Furthermore, we conduct the Self-refine phase following the approach proposed in (Rafailov et al., 2023). In particular, we use the huggingface $DPO_{trainer}$ to support its reproducibility. We follow the parameters proposed in (Rafailov et al., 2023). Hence, for the DPO policy, our work employs a learning rate of 1e-6, β set at 0.1, and a warm-up step count of 100. The batch size is configured to 128. The optimization process is capped at a maximum of 1000 steps, where we save the checkpoint corresponding to the lowest loss on the validation set. The experiments were conducted on a workstation equipped with four Nvidia RTX A6000 with 48GB of VRAM.

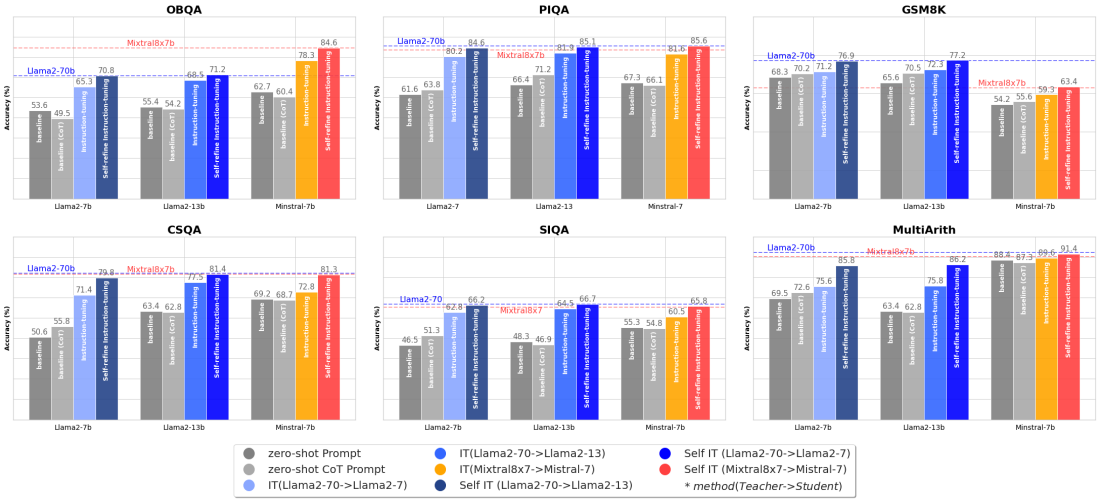

Figure 2: Accuracies (%) on benchmarks (Section 3.1) before Instruction-tuning (i.e., `Baselines` and `Baseline CoT`), after Instruction-tuning (IT) performed on Demonstrations delivering CoT and finally behind the `Self-refine Instruction-tuning` phase (Self IT). In particular, the models were instructed via Demonstrations delivered by in-family LLMs (as described in the legend, we use the notation *method(Teacher->Student)*).

3.3 Evaluation

The most commonly used evaluation methods for question-answering tasks are language-model probing, in which the option with the highest probability is selected (Brown et al., 2020), and multiple-choice probing, in which the models are asked to commit to an answer. The evaluation in the first case is performed with a function taking the *argmax* and, in the second case, with a direct string matching. The second method is more widely used in recent evaluations as it can be inclusive to the larger GPT family models(OpenAI, 2023), where probability values are not readily accessible. In the experiments, we chose the latter to have a comparable and scalable pipeline (Details provided in Appendix C.2). Finally, string matching is performed between the generated outputs and the target choice to evaluate the percentages of the correct answers.

4 Results & Discussion

The *Self-refine Instruction-tuning* improves the alignment between Large Language Models (LLMs) and Small (SLMs) in both in-family and out-family settings. These conclusions can be observed in Figure 2 and Figure 3, which reports the downstream accuracies without tuning (see the Baselines), with only the Instruction-tuning phase on Demonstrations and after the Self-refine phase. As discussed in Section 4.1, the models with only Instruction-tuning on Demonstrations (generated by LLMs) transfers the reasoning properties in a marginal way (see Instruction-tuned in Figures 2).

However, although teacher-student alignment via Instruction-tuning produces better students, an improved alignment is achieved through the Self-refine phase, as discussed in 4.2. In particular, the 'Self-refine Instruction-tuning' bars in Figure 2 show that the students self-refined outperformed the students tuned only with Instruction-tuning ('Instruction-tuning' bars on Figure 2). Furthermore, the alignment via Demonstrations generated by teachers outside the same family (out-family) delivers more robust students (see Figure 3 the Self-refine Instruction-tuning and (in-family) bars).

Finally, students models behind the self-refine phase outperformed others in both in-domain and out-domain tasks (discussed in Section 4.3). Hence, the self-refine mechanism effectively aligns teacher-student capabilities in out-domain tasks by enhancing performance even in the presence of fewer Demonstrations (Section 4.4).

4.1 The Instruction-tuning alignment

Instruction-tuning led by Larger Language Models (teachers models), which are able to deliver multi-step reasoned answers, induces this property within Smaller Language Models (students models). This can be seen in the experiments in Figure 2, Figure 3 and additional evaluations in Appendix I. The student models behind instruction-tuning on demonstrations produced by teacher models outperformed the baselines of the proposed benchmarks.

While one can observe consistent improvements

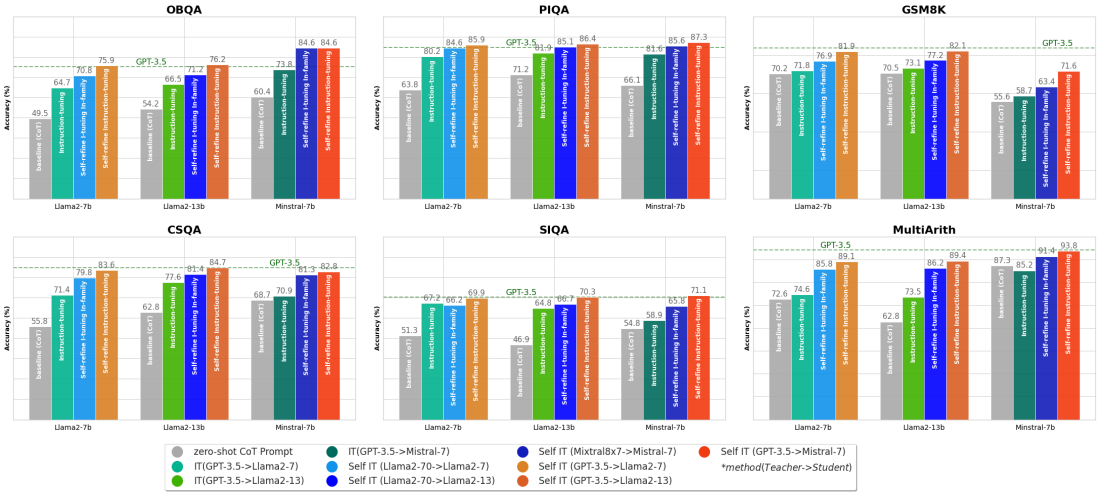

Figure 3: Accuracies (%) on benchmarks (Section 3.1) before Instruction-tuning (Baseline CoT), behind first phase performed on Demonstrations delivering CoT (i.e., Instruction-tuned (IT)) and finally behind the Self-refine phase (i.e., Self-refine IT). In particular, the models were instructed via Demonstrations delivered by out-family LLMs (as described in the legend, we use the notation *method(Teacher->Student)*).

in performance across the board, there are moderate variations across models and tasks. The teacher models that generate Demonstrations stem from different families and perform differently, as shown in Table 5. The consequence of this phenomenon can be seen in Figure 2 and Figure 3 (horizontal lines that are the reported performance of the teachers and bars 'Instruction-tuning' that are the performance of the students). Therefore, the teacher-student alignment is not complete as there is a gap between the performances of the teachers and the students tuned via Instruction-tuning (only phase presented in Section 2.1). In addition, it is possible to differentiate between in-family and out-family alignment. In the in-family, where students are instructed with Demonstrations delivered by the teachers of the same family, performances vary from 6.3 points on average in question-answering (QA) tasks and 8.2 points on average in math word problems (MWP) tasks. Meanwhile, in the out-family alignment, the performances vary by 8.5 on the QA and 8.7 on the MWP.

Hence, to improve the alignment both in-family and consistently out-family, we have proposed an optimization technique based on a self-refinement approach (introduced in Section 2.2), the results of which we discuss in Section 4.2.

4.2 The Self-refine Impact

The Self-refine process enables complete in-family student-teacher alignment by consistently increasing performance in out-family settings and improving the qualities of generated answers. The results obtained in Figure 2 show that the students (SLMs instructed with Self-refine Instruction-tuning) outperform the non-self-refined students and perform comparably to their teachers. The same behaviour can be observed from the out-family setting shown in Figure 3. In particular, the teacher `GPT-3.5` showed a more robust baseline performance (Table 5). Although Instruction-tuning alone transfers some of the abilities to the student models, they were significantly lower when compared to the out-family teacher models. In contrast, the teacher-student performances significantly converged after the self-refine phase, leading to the alignment completion. Finally, a positive impact can also be observed on the quality of students' generations, as shown in the additional experiment discussed in Appendix H.

The performances appear completely aligned, but the students were tested only for in-domain tasks. The proposed approach could cause students to over-specialize in in-domain tasks, running the risk of losing the ability to solve out-domain tasks. For this reason, we performed a set of assessments evaluating students on in-domain and out-domain tasks and discussed the results in Section 4.3.

4.3 In-Domain and Out-Domain

The Self-refine Instruction-tuning approach complements student-teacher alignment and improves students' generalization abilities in out-domain tasks. These results can be observed in Table 1 with

Trained on	Teacher	Evaluated on					
		OBQA	CSQA	PIQA	SIQA	GMS8K	MultiArith
Baseline	-	53.6±.2	50.6±.4	61.6±.1	46.5±.3	68.2±.5	69.5±.2
Baseline CoT	-	49.5±.4	55.8±.3	63.8±.1	51.3±.5	71.3±.2	72.6±.4
OBQA	Instruction-tuning	65.3±.3	65.4±.2	66.3±.4	59.2±.2	61.4±.2	60.2±.3
	+ Self-refine	**70.8**±.3	73.2±.2	75.3±.1	62.6±.3	68.7±.4	69.8±.3
	Cross Self-refine	-	78.4±.1	78.3±.5	64.5±.3	74.4±.4	83.2±.2
CSQA	Instruction-tuning	57.8±.1	71.4±.3	65.5±.4	61.8±.2	60.1±.5	59.3±.1
	+ Self-refine	69.5±.5	**79.8**±.3	74.2±.1	66.3±.2	61.2±.3	60.3±.3
	Cross Self-refine	68.7±.4	-	78.4±.2	64.1±.3	72.1±.4	73.4±.2
PIQA	Instruction-tuning	56.9±.1	64.3±.2	80.2±.3	57.3±.3	58.3±.1	59.1±.3
	+ Self-refine	68.2±.4	67.3±.5	**84.6**±.3	63.4±.2	67.8±.1	66.9±.3
	Cross Self-refine	68.2±.3	71.3±.3	-	64.2±.1	68.7±.4	67.6±.1
SIQA	Instruction-tuning	58.9±.2	62.8±.5	63.2±.1	62.8±.3	59.6±.1	60.2±.3
	+ Self-refine	68.3±.3	68.5±.2	78.3±.3	**66.2**±.4	61.3±.5	60.9±.4
	Cross Self-refine	69.4±.2	68.5±.2	77.9±.3	-	65.1±.3	64.7±.2
GSM8K	Instruction-tuning	53.2±.4	54.9±.5	63.7±.1	52.5±.2	71.2±.3	70.3±.2
	+ Self-refine	58.6±.3	61.7±.4	62.3±.2	52.4±.3	**76.9**±.1	74.3±.2
	Cross Self-refine	64.6±.5	64.3±.2	77.6±.4	60.3±.2	-	75.3±.3
MultiArith	Instruction-tuning	53.6±.2	55.7±.3	53.8±.3	51.5±.3	69.3±.1	75.6±.2
	+ Self-refine	59.1±.2	63.2±.5	58.3±.3	58.6±.1	70.2±.4	**85.8**±.2
	Cross Self-refine	65.3±.4	61.3±.1	62.1±.2	60.7±.5	73.4±.3	-

Table 1: Evaluation of Llama-2-7 Instruction-tuned (`Instruction-tuned`) and with completely Self-refine Instruction-tuning (`+ Self-refine Instruction-tuned`) on Demonstrations using different test sets. We evaluate in-domain (QA vs QA) and out-domain (QA vs math-word problem) benchmarks. "Baselines" are referred to the non-instructed model. Results colored in green indicate the in-domain benchmark, blue the out-domain benchmark, and orange the same benchmark on which perform the evaluation phase. Moreover, we propose Self-refine Instruction-tuning in cross-setting scenario where we optimize the model on the training set related to the evaluated task.

`Llama2-7` as students and `Llama2-70` as teachers (in Appendix Table 10 with Llama2-13 Table 11 with Mistral-7). In particular, behind the evaluations performed on in-domain and out-domain tasks, the students Self-refine Instruction-tuned outperform the baselines and the Instruction-tuned models. Furthermore, to observe the impact of the optimization phase (introduced in Section 2.2) on the downstream performance, we conducted a further experiment by fixing the Instruction-tuning phase and switching the Self-refine ones across different evaluation tasks (e.g., we instructed a student on OBQA and then optimized via self-refine approach on CSQA). As shown in lines `Cross Self-refine` of Table 1, students warmed up on tasks other than those they are optimized, outperformed the others, and obtained similar performances to those obtained from in-domain models. This shows that optimization positively impacts the alignment of generalization abilities in out-domain tasks. Finally, following evaluations in out-domain tasks and across scenarios, we evaluate the performance of the proposed approach by reducing the number of demonstrations available for alignment in Section 4.4.

4.4 Low-resource Optimization

Self-refine Instruction-tuning achieves sustainable performances in low-resource settings. In fact, in Figure 4, it is possible to observe that the performance achieved by the self-refined students consistently outperforms that of the non-self-refined students (where only phase 1 described in Section 2.1 was performed) (technical details on the breakdown can be found in Appendix C.1). Although it emerges that only the optimization process via DPO is more performant than the instruction-tuning process alone, the combination of the two phases achieves the best results in both in-family and out-family alignment in each proposed splitting that are described in Appendix C.1.

5 Related Work

5.1 Multi-step Reasoning

Previous works focus on Chain-of-Thought (CoT) prompting techniques, studying the impact of prompting design and engineering, proposing specialized interventions to improve CoT generalization and fine-grained multi-step reasoning properties (Wei et al., 2022; Fu et al., 2023).

On the prompting design side, Gao et al. (2023)

proposed using Python programs as a CoT prompt, demonstrating more accurate reasoning steps and significant improvements behind CoT prompting (Wei et al., 2022). Zhou et al. (2023) introduced a code generation approach to verify the intermediate reasoning step (OpenAI, 2023).

In parallel, there have been improvements in the accessibility of lower-parameter versions of Large Language Models (LLMs), which we define as Small Language Models (SLMs), on which previous CoT improvements cannot be fully observed (Shridhar et al., 2023; Ho et al., 2023). Therefore, several works are emerging at this gap, aiming to transfer LLM reasoning properties to SLMs. Pioneering proposals in this direction proposed teacher-student alignment methods through a series of approaches geared towards the distillation of the knowledge generated by the teacher for the fine-tuning of the student (Li et al., 2023b; Magister et al., 2023; Shridhar et al., 2023). Later, Yue et al. (2023) proposed specialized Instruction-tuning using Alpaca-like style demonstrations (Taori et al., 2023) specialized for mathematical tasks, while Luo et al. (2023); Xu et al. (2023) proposed supervised fine-tuning reinforced with rewarding algorithms.

5.2 Reinforcement Learning (RL)

A significant component that promotes the generative reasoning delivering CoT is provided by refinement via RL methods. Recent work that applies Proximal Policy Optimization (PPO) (Schulman et al., 2017) for aligning human preferences (Ouyang et al., 2022). Several methods have been proposed to improve the efficiency of alignment (Azar et al., 2023), including Direct Preference Optimization (DPO) (Rafailov et al., 2023).

In this work, we adopt RL to refine performance over conventional SFT. For mathematical problem solving, Uesato et al. (2022) trained an outcome- or process-based reward model to perform re-ranking (Cobbe et al., 2021), achieving better performance than SFT and majority voting (Wang et al., 2023b). (Luong et al., 2024) adopted reinforcement learning as an extension of traditional supervised tuning. We adopt DPO and automate the reward process in a teacher-student context. We focus on the transfer of CoT-style, step-wise reasoning and propose a refinement technique applied to models downstream of the instruction-tuning phase.

5.3 Self-refined Instruction-tuning

Complementing and enhancing foundational approaches (Magister et al., 2023; Uesato et al., 2022; Li et al., 2023a; Ho et al., 2023), several papers have been published simultaneously Wang et al. (2023d); Luo et al. (2023); Wang et al. (2023a); Paul et al. (2024); Luong et al. (2024); Ranaldi and Freitas (2024) (Table 15 summarises the main features). These works prove the effect of supervised fine-tuning to transfer the ability to produce multi-step reasoned answers from larger to smaller models, as described in Section 5.2. Our work goes beyond the state-of-the-art by:

- proposing a method for aligning CoT abilities by introducing Instruction-tuning via Demonstrations produced by answers generated by different LLMs, decentralizing the unique teacher model (in many cases GPT-3.5,4).
- analyzing the alignment performance between in-family and out-family models on different tasks related to commonsense and math reasoning, identifying crucial alignment factors that arise between teachers and students.
- investigating the impact of teacher-student alignment by adapting and promoting DPO (Rafailov et al., 2023) as a cornerstone method for eliminating performance gaps.

6 Conclusion

This paper proposes a novel approach for aligning multi-step CoT reasoning between teacher Large Language Models (LLMs) and student Smaller LMs (SLMs). In particular, our Self-refine Instruction-tuning is framed as an instruction tuning via Chain-of-Thought Demonstrations method based on explanations delivered by LLMs prompted by the CoT mechanism, which is then reinforced via the Self-refine phase that uses Direct Preference Optimization. We also contrast the impact of in-family and out-family alignment across teacher and student models. The results highlight the impact of teacher-student Instruction-tuning interventions as a mechanism to improve the multi-wise reasoning properties of smaller language models and promote the self-refinement abilities of instructed models to complete the alignment.

Limitations

In this paper, we analyzed the impact of Answers delivered by Large Language Models using them as Demonstrations to reinforce the abilities of Small Language Models. Although we proposed an extensive study, there are several limitations:

- only English-language prompting methods and tasks are considered. The understanding of these methods across different languages still needs to be established.
- dependence on Large Language Models, where the supporting training sets are not always fully known. Although the characteristics of the corpora are reported in the system reports. Consequently, contextualising the differences in pre-training data between models is not fully possible, where the analysis is constrained to observing the outputs in natural language.

In conclusion, learning from and with Demonstrations carries some specific risks associated with automation. Although a model may generalize its predictions using a seemingly consistent series of natural language steps, even if the prediction is ultimately correct, there is no guarantee that the predicted output comes from a process represented by the generalization. A end-user might be overconfident in the model based on the CoT mechanism.

Ethical Statement

Although this research enhances the reasoning abilities of Smaller Language Models, they still need to be made sufficiently robust to be applied within more critical domains. Further safety and out-of-distribution generalisation mechanisms needs to be developed in tandem with the application of the methods described in this paper, in order to establish the robustness of the described mechanisms.

References

Mohammad Gheshlaghi Azar, Mark Rowland, Bilal Piot, Daniel Guo, Daniele Calandriello, Michal Valko, and Rémi Munos. 2023. A general theoretical paradigm to understand learning from human preferences.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2019. Piqa: Reasoning about physical commonsense in natural language.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4.

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.

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. 2023. Complexity-based prompting for multi-step reasoning.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: Program-aided language models.

Vedant Gaur and Nikunj Saunshi. 2023. Reasoning in large language models through symbolic math word problems. In Findings of the Association for Computational Linguistics: ACL 2023, pages 5889–5903, Toronto, Canada. Association for Computational Linguistics.

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

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the math dataset.

Namgyu Ho, Laura Schmid, and Se-Young Yun. 2023. Large language models are reasoning teachers. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 14852–14882, Toronto, Canada. Association for Computational Linguistics.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego

de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of experts.

Liunian Harold Li, Jack Hessel, Youngjae Yu, Xiang Ren, Kai-Wei Chang, and Yejin Choi. 2023a. Symbolic chain-of-thought distillation: Small models can also "think" step-by-step. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 2665–2679, Toronto, Canada. Association for Computational Linguistics.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. 2023b. Making language models better reasoners with step-aware verifier. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 5315–5333, Toronto, Canada. Association for Computational Linguistics.

Hanmeng Liu, Ruoxi Ning, Zhiyang Teng, Jian Liu, Qiji Zhou, and Yue Zhang. 2023. Evaluating the logical reasoning ability of chatgpt and gpt-4.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. 2023. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct.

Trung Quoc Luong, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. 2024. Reft: Reasoning with reinforced fine-tuning.

Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. 2023. Teaching small language models to reason. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 1773–1781, Toronto, Canada. Association for Computational Linguistics.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering.

OpenAI. 2023. Gpt-4 technical report.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback.

Debjit Paul, Mete Ismayilzada, Maxime Peyrard, Beatriz Borges, Antoine Bosselut, Robert West, and Boi Faltings. 2024. Refiner: Reasoning feedback on intermediate representations.

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.

Leonardo Ranaldi and Andre Freitas. 2024. Aligning large and small language models via chain-of-thought reasoning. In Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1812–1827, St. Julian's, Malta. Association for Computational Linguistics.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1743–1752, Lisbon, Portugal. Association for Computational Linguistics.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. Social IQa: Commonsense reasoning about social interactions. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 4463–4473, Hong Kong, China. Association for Computational Linguistics.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms.

Kumar Shridhar, Alessandro Stolfo, and Mrinmaya Sachan. 2023. Distilling reasoning capabilities into smaller language models. In Findings of the Association for Computational Linguistics: ACL 2023, pages 7059–7073, Toronto, Canada. Association for Computational Linguistics.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang,

and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process- and outcome-based feedback.

Peiyi Wang, Lei Li, Liang Chen, Feifan Song, Binghuai Lin, Yunbo Cao, Tianyu Liu, and Zhifang Sui. 2023a. Making large language models better reasoners with alignment.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023c. Self-instruct: Aligning language models with self-generated instructions. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 13484–13508, Toronto, Canada. Association for Computational Linguistics.

Zhaoyang Wang, Shaohan Huang, Yuxuan Liu, Jiahai Wang, Minghui Song, Zihan Zhang, Haizhen Huang, Furu Wei, Weiwei Deng, Feng Sun, and Qi Zhang. 2023d. Democratizing reasoning ability: Tailored learning from large language model. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pages 1948–1966, Singapore. Association for Computational Linguistics.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent abilities of large language models.

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.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions.

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.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. 2022. Star: Bootstrapping reasoning with reasoning.

Mengxue Zhang, Zichao Wang, Zhichao Yang, Weiqi Feng, and Andrew Lan. 2023. Interpretable math word problem solution generation via step-by-step planning.

Aojun Zhou, Ke Wang, Zimu Lu, Weikang Shi, Sichun Luo, Zipeng Qin, Shaoqing Lu, Anya Jia, Linqi Song, Mingjie Zhan, and Hongsheng Li. 2023. Solving challenging math word problems using gpt-4 code interpreter with code-based self-verification.

A Instruction-tuning

The Instruction-tuning proposed in our contribution follows the pipeline proposed in (Ranaldi and Freitas, 2024) to achieving teacher-student alignment comprises two steps: annotation and knowledge transfer. In the annotation phase, Large Language Models (teachers) are prompted with questions (see Table 3). The answers are collated and form the Demonstrations (see Table 6). They then move on to the Instruction-tuning phase, conducted using what was proposed in (Taori et al., 2023). In particular, the Demonstrations are constructed with triples formed by the instruction (a pattern to guide the generation related to the task), the input, which is the question related to the mathematical problem or the desired question, and the output the prompted LLM generated. Note that instruction and input can oftentimes be concatenated, but this depends on the basic configurations of the patterns and the type of task to be solved. The instruction-tuning process, a specialization of task-oriented fine-tuning, is similar to the latter and can be described in Section 2.1.

B Self-refine Instruction-tuning

In order to refine Small Language Models (students) instructed via Demonstrations delivered by Large Language Models (teachers) we propose the Self-refine phase (introduced in Section 2.2). In particular, this is based on a variant of the DPO optimization algorithm (Rafailov et al., 2023).
Starting from the Demonstrations defined as $\mathcal{D} = (i_i, q_i, a_i)$ where $i \in \mathcal{D}$ (note that a_i are generated using CoT prompt as showed in Appendix D), we prompt the students using the input $x_i = i_i + q$ (and $\hat{x}_i = x_i +$ `"Let's think step by step"`) $\forall i \in \mathcal{D}$.
Hence, for each element in Demonstrations, we collect the **Answers** ($y_i = \pi_{\text{inst}}(x_i)$) that are the answers generated by the student given the input x_i, and the **CoT-Answers** ($\hat{y}_{CoT} = \pi_{\text{inst}}(\hat{x}_i)$) are the answers that deliver CoT generated by the student elicited via CoT mechanism $\hat{x}_i$.
Hence, we introduce:

- **Oracle or Target** t_i that is the target answer given the input x_i.
- **Demonstration Answer** $\hat{a}_i$ and a_i: that are target answer given the input x_i or $\hat{x}_i$.
- **Answer** $y_i = \pi_{\text{inst}}(x)$: is the answer generated by the student given the input x (without CoT prompt).
- **CoT Answer** $y_{CoT} = \pi_{\text{inst}}(x_{CoT})$: is the answer that delivers CoT generated by the student elicited via CoT mechanism x_{CoT}.

In the following lines, we formalize the structuring of **DPO**$_{CoT}$, **DPO**$_{answer}$ and other configurations.

DPO$_{CoT}$ We propose **DPO**$_{CoT}$ where the answers that deliver correct CoT are referred to as the preferred response, while the others are the answers without CoT defined as:

$$\mathcal{L}_{\text{DPO}_{CoT}}(\pi_\theta; \pi_{\text{inst}}) = -\mathbb{E}_{(x_{CoT}, y_w, y_l)\sim D}\left[\log \sigma(M(x_{CoT}, y_w, y_l))\right] \tag{3}$$

Where $\mathcal{L}_{\text{DPO}_{CoT}}(\pi_{theta}; \pi_{textinst})$ the same $\mathcal{L}_{\text{DPO}}$ introduced in Section 2.2 but in particular to elicit preferred generations the y_w and y_l components are defined as follows, $\forall i \in \mathcal{D}$:

$$y_w = \begin{cases} \hat{y}_i & \text{if } t_i \in \hat{y}_i \\ \hat{a}_i \end{cases} \tag{4}$$

while the discouraged answers are y_l that are y_i $\forall i \in \mathcal{D}$.

DPO$_{answer}$ In contrast, we propose **DPO**$_{answer}$ and where the answers without CoT are referred to as the preferred.

$$\mathcal{L}_{\text{DPO}_{answer}}(\pi_\theta; \pi_{\text{inst}}) = -\mathbb{E}_{(x, y_p, y_{CoT})\sim D}\left[\log \sigma(M(x, y_p, y_{CoT}))\right] \tag{5}$$

However, since our contribution is focused on CoT in the main work, we only consider **DPO**$_{CoT}$. In the Table 4, we have reported **DPO**$_{answer}$ results.

C Experimental Details

C.1 Data Splitting

In order to observe the impact of the Demonstrations, we produced a series of experiments by systematically decreasing the Instruction-tuning data. In particular, we chose three sub-sets with 75%, 50%, and 25% from the total number of demonstrations. In detail, the Self-refine Instruction phases on the number of equal Demonstrations are performed by taking about 3000 examples in splitting 100%, 2250 in splitting 50%, 1500 in splitting 50%, and 750 in splitting 25%. We chose the value 3000 because it has the smallest CoT Demonstrations available. For the total Demonstrations, we selected random samples. Using these splitting, we performed the evaluations incrementally as the demonstrations used to do Instruction-tuning, to do Self-refine, and to do Self-refine Instruction-tuning.

C.2 Parameters

The annotation phase that the Teachers performed was done on the training set. The evaluation phase of both the basic models and the Students and the Teachers was done on the test splitting. The evaluation, described in Section 3.3, was done with question probing and string matching of the generated answers. More specifically:

Teachers We performed the annotation phase for each benchmark by delivering to `GPT-3.5-turbo`, `Mixtral7x8` and `Llama-2-70-chat` the prompts structured as shown in Table 2 and Table 3 (customized for each benchmark). We set the temperatures to 0.7 for `GPT-3.5-turbo` and 0.1 for `Llama-2-70-chat` as recommended in technical reports. Moreover, we kept all the other parameters as default. All parameters are shown in our code .

Baseline & Students We evaluated the performance of the Small Language Models (`Llama-2-7-chat`, `Llama-2-13-chat`, `Mistral-7b`) by prompting them with the same format used for the Teachers. For both the baselines and the instructed models, we set the temperature to 0.1 and kept all the other parameters as default. The evaluation pipelines and generation parameters are available in our code.

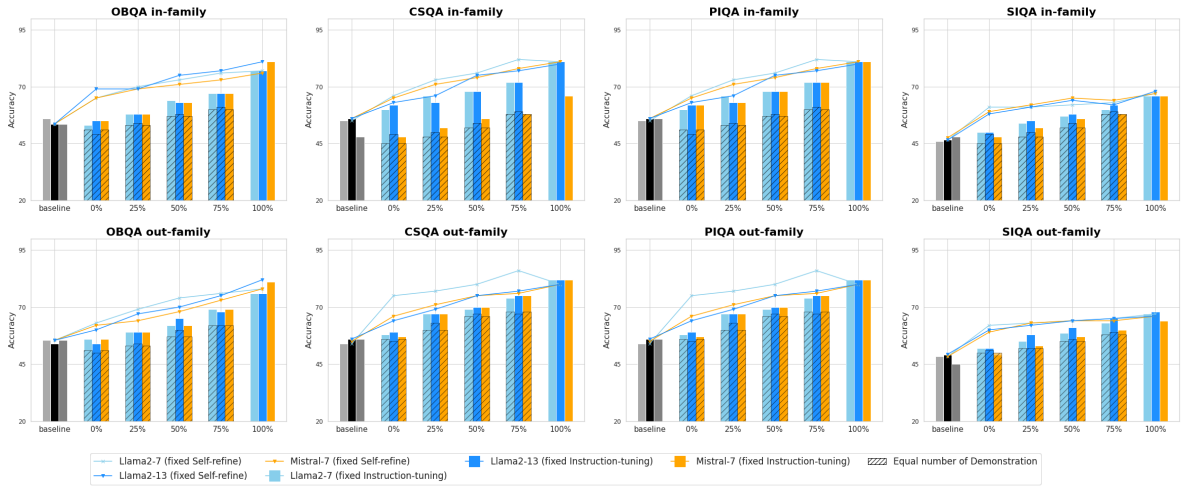

Figure 4: Acciracies (%) on the test set of benchmarks. The Self-refine Instruction-tuning performed on different splits (see Appendix C.1 for major details).

D Prompting Approaches

Prompt for task: OBQA, CSQA, PIQA, SIQA
Choose the answer to the question only from options A, B, C, [...]. **Question:** <Question> **Choices:** A) <Option1> B) <Option2> C) <Option3> .... **Answer:**

Prompt for task: GSM8k, MultiArith
Answer the following mathematical question with numerical solution. **Question:** <Question> **Answer:**

Table 2: Example of input-prompt for multiple-choices (left) and mathematical (right) question-answering benchmarks.

Prompt for task: OBQA, CSQA, PIQA, SIQA
Choose the answer to the question only from options A, B, C, [...]. **Question:** <Question> **Choices:** A) <Option1> B) <Option2> C) <Option3> .... **Answer: Let's think step by step**

Prompt for task: GSM8k, MultiArith
Answer the following mathematical question with numerical solution. **Question:** <Question> **Answer: Let's think step by step**

Table 3: Example **Zero-shot CoT** of input-prompt for multiple-choices (left) and mathematical (right) question-answering benchmarks (approach used in this work).

E Models

Model	Version
Llama-2-7-chat	meta-llama/Llama-2-7b
Llama-2-13-chat	meta-llama/Llama-2-13b
Llama-2-70-chat	meta-llama/Llama-2-70b
Mistral-7	mistralai/Mistral-7B-Instruct-v0.1
Mixtral7x8	mistralai/Mixtral-8x7B-v0.1

Table 4: List and specific versions of the models proposed in this work, which can be found on huggingface.co. For each model we used all the default configurations proposed in the repositories.

F Accuracy of LLMs on different Benchhmark

Benchmarks	Llama-2-70		GPT-3.5		Mixtral7x8	
	Baseline	CoT	Baseline	CoT	Baseline	CoT
Training						
OpenBook QA	$65.6_{\pm.3}$	$71.3_{\pm.1}$	$66.2_{\pm.2}$	$75.4_{\pm.4}$	$77.9_{\pm.3}$	$\mathbf{81.2}_{\pm.1}$
CommonSesnse QA	$74.2_{\pm.1}$	$79.6_{\pm.3}$	$79.3_{\pm.4}$	$\mathbf{84.8}_{\pm.1}$	$78.2_{\pm.2}$	$82.3_{\pm.3}$
Social Interaction QA	$65.4_{\pm.2}$	$67.5_{\pm.3}$	$67.6_{\pm.5}$	$\mathbf{70.3}_{\pm.4}$	$65.5_{\pm.2}$	$68.2_{\pm.3}$
Physical Interaction QA	$82.6_{\pm.2}$	$\mathbf{85.8}_{\pm.2\pm.3}$	$83.5_{\pm.3}$	$85.3_{\pm.1}$	$80.2_{\pm.3}$	$84.1_{\pm.3}$
GSM8K	$74.6_{\pm.1}$	$77.2_{\pm.2}$	$83.2_{\pm.2}$	$\mathbf{86.5}_{\pm.1}$	$65.6_{\pm.4}$	$67.9_{\pm.2}$
MultiArith	$88.6_{\pm.4}$	$90.8_{\pm.3}$	$94.9_{\pm.4}$	$\mathbf{96.7}_{\pm.1}$	$89.3_{\pm.1}$	$91.5_{\pm.4}$
Testing						
OpenBook QA	$65.9_{\pm.2}$	$70.8_{\pm.1}$	$67.8_{\pm.1}$	$74.6_{\pm.4}$	$78.4_{\pm.3}$	$\mathbf{84.6}_{\pm.2}$
CommonSesnse QA	$73.4_{\pm.2}$	$81.8_{\pm.3}$	$80.2_{\pm.2}$	$\mathbf{83.7}_{\pm.1}$	$77.6_{\pm.3}$	$81.5_{\pm.1}$
Social Interaction QA	$64.2_{\pm.2}$	$66.9_{\pm.4}$	66.9	$\mathbf{71.3}_{\pm.3}$	$64.3_{\pm.3}$	$65.4_{\pm.2}$
Physical Interaction QA	$82.6_{\pm.3}$	$85.6_{\pm.5}$	$84.3_{\pm.2}$	$\mathbf{85.8}_{\pm.5}$	$79.9_{\pm.3}$	$83.5_{\pm.1}$
GSM8K	$75.2_{\pm.5}$	$77.8_{\pm.5}$	$82.8_{\pm.2}$	$\mathbf{84.6}_{\pm.4}$	$63.4_{\pm.3}$	$62.8_{\pm.5}$
MultiArith	$89.2_{\pm.1}$	$92.3_{\pm.2}$	$95.6_{\pm.2}$	$\mathbf{97.4}_{\pm3}$	$88.9_{\pm.1}$	$90.2_{\pm.3}$

Table 5: Accuracy (%) of Llama-2-70, GPT-3.5 and Mixtral7x8 (teachers) on training and testing data with CoT prompt (CoT) and with the standard prompt (Baseline).

Instruction

```
Choose the answer to the question only from options A, B, C, D.
```

Input

```
Question: Which form can be made into a triangle?
Choices:
A) ice
B) steam
C) water
D) salt water
Answer: Let's think step-by-step
```

Output (that is the Answer produced by teacher model)

```
Answer:
A) ice: Ice is a solid form of water. In solid form, substances can be molded or shaped into
various forms, including triangles. If you had a block of ice, you could carve or mold it
into a triangular shape.
B) steam: Steam is the gaseous form of water. Being a gas, it doesn't have a fixed shape. You
can't shape steam into a triangle without containing it within some form of solid boundary,
and even then, the steam itself isn't triangular - the container is.
C) water: Water is a liquid that takes the shape of its container. You can pour water into
a triangular container, and while the volume inside would be triangular, the water itself
isn't inherently triangular.
D) salt water: This is essentially the same as option C, but with salt dissolved in it. Being
a liquid, it takes the shape of its container.
Given these considerations, the only substance among the options that can be shaped into a
triangle by itself is A) Ice.
```

Table 6: Example of Demonstration produced by the GPT-3.5 answer on an Open Book Question Answering benchmark instance. The structure is composed by: **Instruction**, **Input** and **Output**.

G Description of proposed Benchmark

Dataset	Example
Open Book Question Answering (OBQA) (Mihaylov et al., 2018)	*When birds migrate south for the winter, they do it because* **A) they are genetically called to.** B) their children ask them to. C) it is important to their happiness. D) they decide to each.
Common Sense Question Answering (CSQA) (Talmor et al., 2019)	*Aside from water and nourishment what does your dog need?* A) bone. B) charm. C) petted. **D) lots of attention.** E) walked.
Physical Interaction Question Answering (PIQA) (Bisk et al., 2019)	*How do you attach toilet paper to a glass jar?* **A) Press a piece of double-sided tape to the glass jar and then press the toilet paper onto the tape.** B) Spread mayonnaise all over the jar with your palms and then roll the jar in toilet paper.
Social Interaction Question Answering (SIQA) (Sap et al., 2019)	*Taylor gave help to a friend who was having trouble keeping up with their bills. What will their friend want to do next?* A) Help the friend find a higher paying job. **B) Thank Taylor for the generosity.** C) pay some of their late employees.
(GSM8K) (Cobbe et al., 2021)	Tina makes $18.00 an hour. If she works more than 8 hours per shift, she is eligible for overtime, which is paid by your wage + 1/2 your hourly hourly wage. If she works 10 hours every day for 5 days, how much money does she make?
(MultiArith) (Roy and Roth, 2015)	Chloe was playing a video game where she scores 9 points for each treasure she finds. If she found 6 treasures on the first level and 3 on the second, what would her score be?

Table 7: Examples of the benchmarks used in this paper.

	OBQA	CSQA	PIQA	SIQA	GSM8K	MultiArith
classes	4	5	2	3	-	-
Training # examples for each class	1000	800	2000	1330	4000	420
Test # examples for each class	125* (± 8)	235* (± 11)	924* (± 18)	640* (± 19)	1318	180

Table 8: Characteristics Training and Test set of benchmarks proposed in Section 3.1. The * indicates that the number of examples are not perfect balanced, but the difference from the average is marginal. GMS8K e MultiArith are not closed-ended question answering; they only have a question and a numerical solution.

Name	Repository
CommonSenseQA (Talmor et al., 2019)	huggingface.co/datasets/commonsense_qa
OpenBookQA (Mihaylov et al., 2018)	huggingface.co/datasets/openbookqa
StrategyQA ()	huggingface.co/datasets/voidful/StrategyQA
PIQA (Bisk et al., 2019)	huggingface.co/datasets/piqa
SIQA (Sap et al., 2019)	huggingface.co/datasets/social_i_qa
GSM8K (Cobbe et al., 2021)	huggingface.co/datasets/gsm8k
MultiArith (Roy and Roth, 2015)	huggingface.co/datasets/ChilleD/MultiArith

Table 9: In this table, we list the versions of the benchmark proposed in this work, which can be found on huggingface.co.

Trained on	Teacher	Evaluated on					
		OBQA	CSQA	PIQA	SIQA	GMS8K	MultiArith
Baseline	-	55.4±.2	63.4±.3	66.4±.2	48.3±.2	65.6±.4	63.4±.2
Baseline CoT	-	54.2±.2	62.8±.4	71.2±.3	46.9±.5	70.5±.1	62.8±.2
OBQA	Instruction-tuning	68.5±.4	67.5±.3	69.4±.1	60.1±.2	62.3±.4	61.5±.5
	+ Self-refine	**71.2**±.4	74.1±.2	76.2±.3	63.4±.3	69.9±.4	70.7±.2
	Cross Self-refine	-	79.2±.1	79.5±.2	65.6±.3	75.2±.4	84.3±.5
CSQA	Instruction-tuning	58.4±.4	77.5±.2	66.4±.2	61.8±.3	62.4±.4	60.2±.2
	+ Self-refine	69.5±.5	**81.4**±.2	74.2±.5	67.9±.1	62.1±.3	61.4±.4
	Cross Self-refine	70.2±.4	-	79.5±.3	65.2±.1	73.3±.3	75.3±.5
PIQA	Instruction-tuning	57.8±.2	65.2±.3	81.9±.4	58.5±.4	59.2±.4	60.3±.3
	+ Self-refine	69.6±.2	68.2±.4	**85.1**±.5	64.3±.1	69.3±.2	68.1±.3
	Cross Self-refine	69.9±.1	71.3±.1	-	65.3±.1	69.6±.4	69.2±.2
SIQA	Instruction-tuning	59.6±.1	63.9±.4	67.1±.2	64.5±.3	60.3±.4	61.3±.2
	+ Self-refine	69.2±.2	69.4±.1	79.2±.4	**66.7**±.3	62.4±.4	61.8±.2
	Cross Self-refine	71.2±.2	69.2±.1	80.4±.2	-	66.5±.1	66.7±.2
GSM8K	Instruction-tuning	54.3±.2	55.8±.3	64.3±.4	53.2±.3	72.3±.3	71.6±.2
	+ Self-refine	59.3±.4	62.2±.2	63.5±.3	53.5±.5	**77.2**±.4	75.2±.3
	Cross Self-refine	65.7±.1	65.2±.5	78.1±.3	61.6±.4	-	76.2±.2
MultiArith	Instruction-tuning	54.7±.2	56.6±.3	54.5±.3	52.4±.3	70.2±.1	75.8±.2
	+ Self-refine	60.3±.2	64.1±.4	59.4±.3	59.7±.1	72.1±.4	**86.2**±.3
	Cross Self-refine	66.2±.3	62.4±.1	63.2±.3	61.5±.4	73.9±.2	-

Table 10: Evaluation of Llama-2-13 Instruction-tuned (`Instruction-tuned`) and with completely Self-refine Instruction-tuning (`+ Self-refine Instruction-tuned`) on Demonstrations using different test sets. We evaluate in-domain (QA vs QA) and out-domain (QA vs math-word problem) benchmarks. "Baselines" are referred to the non-instructed model. Results colored in green indicate the in-domain benchmark, blue the out-domain benchmark, and orange the same benchmark on which the evaluation phase is performed. Moreover, we propose Self-refine Instruction-tuning in cross-setting scenarios where we optimize the model on the training set related to the evaluated task.

Trained on	Teacher	Evaluated on					
		OBQA	CSQA	PIQA	SIQA	GMS8K	MultiArith
Baseline	-	62.7±.3	69.2±.4	67.3±.1	55.3±.2	54.2±.2	88.4±.1
Baseline CoT	-	60.4±.3	68.7±.2	66.1±.2	54.8±.4	55.6±.3	87.3±.2
OBQA	Instruction-tuning	78.3±.2	65.4±.2	67.2±.3	59.2±.1	64.2±.2	62.1±.3
	+ Self-refine	**87.6**±.2	73.1±.2	76.1±.1	63.3±.3	69.1±.4	70.7±.3
	Cross Self-refine	-	79.4±.1	80.1±.2	68.2±.4	75.2±.4	81.3±.1
CSQA	Instruction-tuning	58.9±.1	73.1±.4	65.8±.2	62.1±.1	62.2±.3	60.2±.2
	+ Self-refine	69.5±.5	**81.3**±.1	75.1±.1	66.5±.2	61.1±.4	62.4±.1
	Cross Self-refine	69.2±.2	-	79.3±.1	65.2±.4	72.8±.4	74.4±.2
PIQA	Instruction-tuning	58.6±.2	64.8±.2	81.6±.2	59.2±.4	60.2±.2	60.3±.4
	+ Self-refine	68.2±.4	68.2±.5	**85.6**±.2	63.8±.2	67.9±.2	67.2±.4
	Cross Self-refine	69.2±.3	71.9±.3	-	63.2±.1	68.4±.5	69.6±.1
SIQA	Instruction-tuning	59.3±.2	66.8±.2	63.2±.4	61.5±.2	60.2±.1	61.3±.3
	+ Self-refine	68.3±.3	68.5±.2	78.3±.3	**65.8**±.4	62.4±.5	61.3±.4
	Cross Self-refine	71.3±.4	69.2±.2	78.1±.2	-	65.6±.3	68.3±.1
GSM8K	Instruction-tuning	52.4±.1	54.9±.5	58.7±.1	51.8±.3	56.1±.1	65.2±.
	+ Self-refine	57.6±.3	58.7±.4	59.3±.2	51.4±.2	**63.4**±.1	60.3±.1
	Cross Self-refine	61.3±.5	64.3±.2	70.1±.4	58.2±.1	-	70.5±.3
MultiArith	Instruction-tuning	57.9±.2	59.2±.3	53.8±.4	51.5±.3	69.3±.2	89.6±.4
	+ Self-refine	59.1±.2	63.2±.4	59.4±.5	59.9±.2	68.2±.1	**91.4**±.3
	Cross Self-refine	64.7±.4	65.8±.2	64.1±.4	61.5±.4	70.1±.3	-

Table 11: Evaluation of Mistral-7 Instruction-tuned (`Instruction-tuned`) and with completely Self-refine Instruction-tuning (`+ Self-refine Instruction-tuned`) on Demonstrations using different test sets. We evaluate in-domain (QA vs QA) and out-domain (QA vs math-word problem) benchmarks. "Baselines" are referred to the non-instructed model. Results colored in green indicate the in-domain benchmark, blue the out-domain benchmark, and orange the same benchmark on which perform the evaluation phase. Moreover, we propose Self-refine Instruction-tuning in cross-setting scenario where we optimize the model on the training set related to the evaluated task.

H Quality of Generations

To demonstrate the quality of the demonstrations generated by the teachers and students, we propose annotating the responses provided by the teacher and student models automatically. In particular, we sampled 300 questions (50 questions for each task from the testing set split). Hence, we systematically prompt both the teacher LLMs and students. Finally, we estimated the quality of the responses generated by systematically prompting a judge LLM (we chose `GPT-4` as it is not among the models used in this work).

```
Please act as an impartial judge and evaluate the quality of the response
provided by an AI assistant to the user instruction displayed below. Your
evaluation should consider factors such as quality, accuracy, depth, and
level of detail. Begin your assessment with a short explanation. Be as
objective as possible. After providing your explanation, please rate the
response on a scale of 1 to 3 strictly following this format:“[[rating]]”,
for example: “Rating: [[2]]”.
[question]
${question}
[AI assistant’s response]
${response}
```

Table 12: Using this prompt, we systematically query GPT-4 to note the answers' quality.

Model	**Llama2-70b**	**Mixtral8x7b**	**GPT-3.5**
Baseline	1.63	1.34	1.68
Baseline CoT	2.72	2.56	**2.89**
Target Answers	1	1	1

Table 13: Averages quality scores obtained by LLMs' answers by using GPT-4 as judge (see Table H).

	Model	**Llama2-7b**	**Llama2-13b**	**Mistral-7b**
	Baseline	1.26	1.39	1.16
	Baseline CoT	1.47	1.56	1.21
in-family	Instruction-tuning	2.43	2.66	2.36
	Self-refine Instruction-tuning	2.75	**2.83**	2.54
out-family (GPT-3.5)	Instruction-tuning	1.99	2.17	1.76
	Self-refine Instruction-tuning	**2.86**	2.79	**2.82**

Table 14: Averages quality scores obtained by students' answers by using GPT-4 as judge (see Table H).

work	approach	teacher/s	students/s	tasks
(Zelikman et al., 2022)	Self-SFT	-	GPT-J, LaMDA	GSM8k, CSQA
(Magister et al., 2023)	SFT	PaLM GPT-3.5	T5-small, -medium T5-large, -xxl	GSM8k, StrategyQA, MArith
(Li et al., 2023a)	SFT	GPT-3 175B	OPT-1.3b	CSQA, OBQA, QARel
(Shridhar et al., 2023)	SFT	GPT-3 175B	GPT-2	GSM8k, StrategyQA SVAMP
(Ho et al., 2023)	SFT	InstructGPT (text-davinci-002)	GPT-3 (ada,babbage,curie)	GSM8k, StrategyQA, MArith, SVAMP, AddSub
(Wang et al., 2023d)	IT+RL	GPT-3	GPT-J	GSM8K, MultiArith, SVAMP CSQA, StrategyQA
(Luong et al., 2024)	SFT+RL	GPT-3.5	Galactica, CodeLlama	GSM8k SVAMP MathQA
(Ranaldi and Freitas, 2024)	IT	GPT-3.5, Llama2-70	Llama2-7,13, Mistral-7	GSM8k, PIQA, MathQA CSQA, OBQA, SIQA
(Wang et al., 2023a)	SFT+RL	GPT-3.5	Llama2-7,13	GSM8k, EAQA
(Paul et al., 2024)	SFT	GPT-3.5	CodeT5 s,m	GSM8k, SVAMP, MArith
Ours	IT+RL (DPO) (in-family vs out-family)	GPT-3.5, Llama2-70 Mixtral8x7	Llama2-7,Llama2-13, Mistral-7	GSM8k, CSQA, OBQA PIQA, SIQA, MArith MATH, MMLU

Table 15: Summary of methods, teacher and student models of previous work, we indicate Supervised Fine-tuning as (SFT), Instruction-tuning as (IT), and Reinforcement Learning (RL). *note that previous works do not use DPO (Rafailov et al., 2023)

I Additional Evaluations

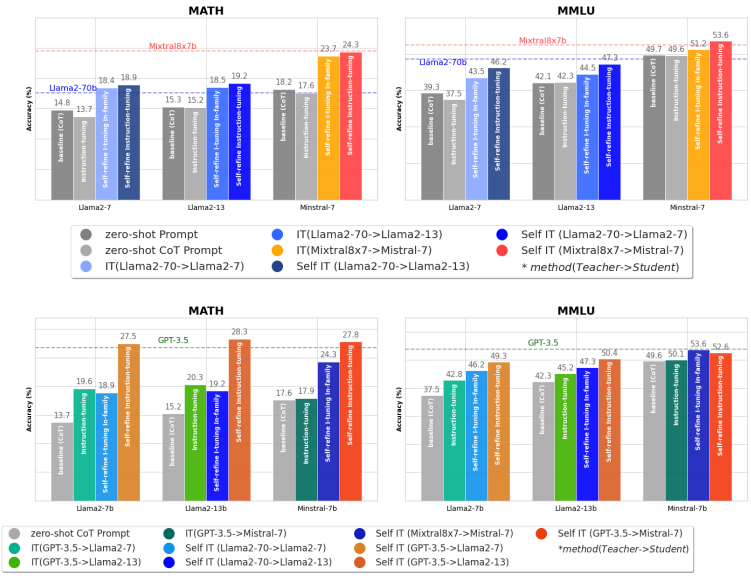

Figure 5: Accuracies (%) additional benchmarks as described in Section 3.1. Applying the same pipeline proposed in Section 2 and the same experimental set-up (Section 3) as the experiments shown in Figure 2 and Figure 3. In this experiment, we showed that the approach proposed in Section 2 is also scalable on multi-task benchmarks such as MATH (Hendrycks et al., 2021b) and MMLU (Hendrycks et al., 2021a). (Self-refine Instruction-tuning phase performed using 25% as the training set and omitted in the evaluation phase) (as described in the legend, we use the notation *method(Teacher->Student)*).