

# SELF-CONSISTENCY PREFERENCE OPTIMIZATION

Archiki Prasad<sup>1,2</sup> Weizhe Yuan<sup>1,3</sup> Richard Yuanzhe Pang<sup>1</sup> Jing Xu<sup>1</sup>  
 Maryam Fazel-Zarandi<sup>1</sup> Mohit Bansal<sup>2</sup> Sainbayar Sukhbaatar<sup>1</sup> Jason Weston<sup>1,3</sup> Jane Yu<sup>1</sup>

<sup>1</sup>Meta FAIR <sup>2</sup>UNC Chapel Hill <sup>3</sup>New York University

## ABSTRACT

Self-alignment, whereby models learn to improve themselves without human annotation, is a rapidly growing research area. However, existing techniques often fail to improve complex reasoning tasks due to the difficulty of assigning correct rewards. An orthogonal approach that is known to improve correctness is self-consistency, a method applied at inference time based on multiple sampling in order to find the most consistent answer. In this work, we extend the self-consistency concept to help *train models*. We thus introduce self-consistency preference optimization (SCPO), which iteratively trains consistent answers to be preferred over inconsistent ones on unsupervised new problems. We show SCPO leads to large improvements over conventional reward model training on reasoning tasks such as GSM8K and MATH, closing the gap with supervised training with gold answers or preferences, and that combining SCPO with standard supervised learning improves results even further. On ZebraLogic, SCPO finetunes Llama-3 8B to be superior to Llama-3 70B, Gemma-2 27B, and Claude-3 Haiku.

## 1 INTRODUCTION

Training large language models (LLMs) on human-annotated data has improved their performance on a wide array of tasks (Bai et al., 2022; Touvron et al., 2023). However, the size and quality of human data remains a major bottleneck as the data collection process is often resource-intensive in terms of cost, time, and expertise. To address this challenge, recent works focus on iteratively training from model-generated data via *self-training* (Yuan et al., 2024; Chen et al., 2024b). Notably, Yuan et al. (2024) propose a “self-rewarding” training pipeline for instruction-following, comprising two steps: (i) using the LLM to generate new queries and self-evaluating the generated responses for each query; and (ii) building preference pairs and training the LLM using iterative direct preference optimization loss (DPO; Rafailov et al., 2024; Xu et al., 2023). However, Huang et al. (2024) demonstrate that LLMs struggle at evaluating the correctness of their own responses on complex problem-solving tasks which have *an unambiguous correct answer*, thereby rendering Yuan et al.’s self-evaluation approach ineffective. Using an external reward model (RM) to rank responses can have similar problems; even if such models are trained on reasoning tasks they may still suffer on out-of-distribution problems (Casper et al., 2023; Zhang et al., 2024; Mahan et al., 2024).

To address this issue, we introduce *Self-consistency Preference Optimization* (SCPO). SCPO is an approach to self-train LLMs for complex problem-solving tasks without access to gold solutions or final answers in the training data. Our approach leverages the concept of self-consistency (Wang et al., 2023), an inference-time only approach that improves performance on reasoning tasks by generating multiple solutions using the LLM and choosing the most frequent final answer. More consistent answers are more likely to be correct because mistakes made by the model are often random, so incorrect solutions are unlikely to lead to the same answer multiple times (Fischler & Bolles, 1981; Chen et al., 2023). In SCPO, the self-consistency concept is instead applied *during unsupervised self-training*. The method consists of (i) *selecting* model-generated queries, (ii) *annotating* preference pairs using the most self-consistent response (winner) and least self-consistent response (loser), and (iii) *optimizing* a loss function that is weighted for each instance depending on the model’s confidence in the preference pair. Additionally, we propose a *semi-supervised* variant of SCPO that jointly trains LLMs on labeled and unlabeled instances, taking advantage of human

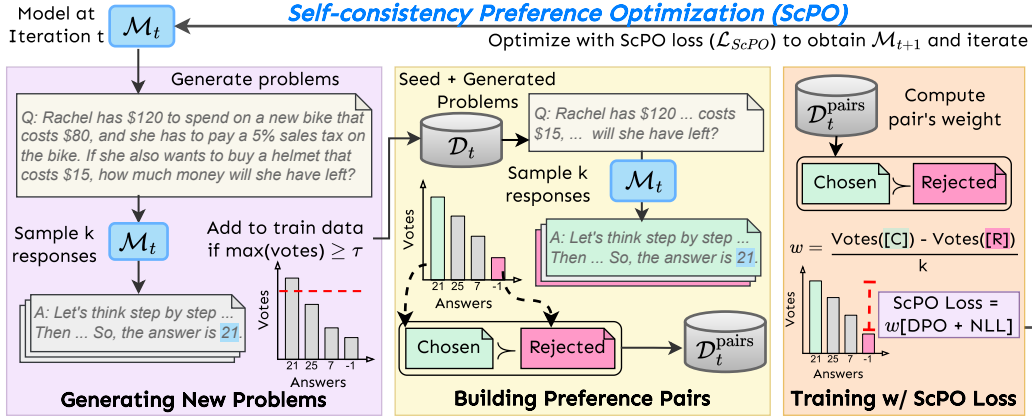


Figure 1: **Self-consistency Preference Optimization (ScPO)**. Given a query, we sample multiple responses from the current model  $\mathcal{M}_t$  and count the frequency of each answer (i.e., votes). We select the highest and lowest votes as chosen and rejected responses (middle), and use these preference pairs to train the model with weighted  $\mathcal{L}_{\text{ScPO}}$  loss (right). We employ a similar pipeline for generating new queries from the model itself (left), filtering out data where self-consistency is low.

annotations whenever available. Unlike self-consistency applied at inference time, SCPO does not increase inference-time compute, but they can also be combined together for better performance.

In our experiments using Llama-3 8B models (Dubey et al., 2024), we show that even without access to *any* gold answers during training, two iterations of unsupervised SCPO improves zero-shot accuracy of the base model by 22.74% and 5.26% (absolute) on GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021) respectively, closely matching the performance ( $< 1\%$  difference) of the *supervised* baseline from Pang et al. (2024). Moreover, when supplied with the gold labels in the training set and additional model-generated problems, semi-supervised SCPO improves GSM8K accuracy over the supervised baseline by 2.35%. On challenging logical puzzles in ZebraLogic (Dziri et al., 2024) – where only test puzzles (without solutions) are publicly available – training Llama-3 8B with SCPO improves puzzle accuracy by 6.5%, outperforming larger LLMs such as Llama-3 70B, Gemma-2 27B (Team et al., 2024), and Claude-3 Haiku (Anthropic, 2024).

## 2 SELF-CONSISTENCY PREFERENCE OPTIMIZATION

As depicted in Figure 1, SCPO is an unsupervised iterative training method that starts with a base language model. Each iteration makes use of existing training problems/queries (without labels) as well as newly generated problems. The self-consistency metric is used in both generating new problems and building preference pairs. We describe each step of SCPO’s iterative training setup below. All prompts for solution generation and new problem generation can be found in Appendix C.

**Initialization.** SCPO assumes access to an initial base model  $M_0$  and a small amount of (seed) high-quality unlabeled queries, which are typically complex reasoning problems. The model will be trained and updated at each training iteration resulting in models  $M_1, M_2, \dots, M_T$ , where  $T$  is the total number of iterations. Instead of gold labels (answers) for responses, SCPO uses the consistency of the model  $M_t$ , as measured by a real-valued vote function  $\mathcal{V}(\cdot)$  defined below, to rate and rank the quality of each response. Our vote function is based on *self-consistency* (Wang et al., 2023) of the model SCPO can also be used with *any* measure of model consistency such as internal consistency (Liang et al., 2024) or universal consistency (Chen et al., 2024a).

**Generating New Problems.** Following other self-alignment methods (Yuan et al., 2024; Yu et al., 2024), we use few-shot prompting to self-generate additional problems from the model. Using the seed set, multiple example problems are chosen at random and placed in context to generate a new problem. Note that some prior works are constrained to simultaneously generating both a new query along with its corresponding correct answer (Yu et al., 2024). In contrast, with SCPO, we do not rely on accurately generating the corresponding answer, allowing the model to generate

more diverse problems as long as the problems are *well-formed* and at least some are *answerable*. While the model may generate some unanswerable queries, these can be filtered out using the vote function  $\mathcal{V}(\cdot)$ . Specifically, we filter out query  $x$  if none of the responses generated by  $M_t$  have vote  $\geq \tau$  (shown in [Figure 1](#); left). At each iteration  $t$ , we augment the seed queries with the problems generated from  $M_t$  to obtain the training problems for the next iteration  $\mathcal{D}_{t+1}$ .

**Building Self-Consistency Preference Pairs.** For each problem  $x$  in the training data  $\mathcal{D}_t$ , we use temperature-based sampling with the current model  $M_t$  to generate  $k$  responses  $\bar{y}_x = \{y_1, y_2, \dots, y_k\}$  sampled from  $M_t(\cdot|x)$  including any rationales, e.g., chain-of-thought ([Wei et al., 2022](#)), followed by the final answer. Following [Wang et al. \(2023\)](#), the vote function  $\mathcal{V}(\cdot)$  extracts the final answer corresponding to each response  $y \in \bar{y}_x$  via  $\text{ans}(\cdot)$  and returns the relative frequency of the final answer, i.e.,  $\mathcal{V}(y) = \sum_{m=1}^k \mathbb{1}(\text{ans}(y_m) = \text{ans}(y))$ . As illustrated in [Figure 1](#) (middle), using the vote function, we create preference pairs  $\mathcal{D}_t^{\text{pairs}}$  by selecting the *most consistent* response as the chosen (winning) response and selecting the *least consistent* one as the rejected (losing) response, provided that the vote of the chosen response is greater than a threshold  $\tau$ .<sup>1</sup> In other words,

$$\mathcal{D}_t^{\text{pairs}} = \{(x, y^+, y^-) | x \in \mathcal{D}_t, y^+ = \arg \max_{y \in \bar{y}_x} \mathcal{V}(y), y^- = \arg \min_{y \in \bar{y}_x} \mathcal{V}(y), \text{ and } \mathcal{V}(y^+) \geq \tau\}.$$

**SCPO Loss Function.** SCPO operates under the assumption that when multiple responses sampled for problem  $x$  map to the same answer, then the predicted answer is likely to be correct, the same assumption as in [Wang et al. \(2023\)](#). Consequently, we use consistency via a vote function  $\mathcal{V}(\cdot)$  as a proxy to create preference pairs. However, at the same time, the number of votes attained by a response can also reflect the model’s *confidence* in the response ([Xiong et al., 2024](#); [Kabra et al., 2024](#)), implying that pairs where the vote margin – the difference in votes attained by the chosen vs. the rejected response – is larger, are of *higher quality* and vice-versa (refer to [Appendix A](#)). We model this in SCPO’s training by using an instance-level weight  $w(x)$  to the loss, i.e., for the preference pair  $(x, y^+, y^-) \in \mathcal{D}_t^{\text{pairs}}$ ,  $w(x) = (\mathcal{V}(y^+) - \mathcal{V}(y^-))/k$ , where  $k$  is the total number of responses generated for each question (total number of votes cast).<sup>2</sup> We thus use the following loss function:

$$\mathcal{L}_{\text{SCPO}}(y^+, y^- | x) = \underbrace{-w(x) \log \sigma \left( \beta \log \frac{M_\theta(y^+ | x)}{M_t(y^+ | x)} - \beta \log \frac{M_\theta(y^- | x)}{M_t(y^- | x)} \right)}_{\text{Weighted DPO Loss}} \underbrace{- \frac{\alpha w(x)}{|y^+|} \log M_\theta(y^+ | x)}_{\text{Weighted NLL Loss}}.$$

The loss includes a DPO and NLL term similar to the recently introduced supervised IRPO ([Pang et al., 2024](#)) loss, but in our case we have an unsupervised objective and use our introduced weighted loss. Here  $\sigma(\cdot)$  denotes the sigmoid function, and  $\alpha, \beta$  are hyperparameters of the loss function, and  $\theta$  represents the LLM parameters being trained in the current iteration. At the  $t^{\text{th}}$  iteration, we use the initialized model  $M_t$  as the reference model in the DPO loss ([Rafailov et al., 2024](#)). After training on this loss, the trained model is used to initialize the next iteration, i.e.,  $M_{t+1} \leftarrow M_\theta$ .

**Iterative Training.** Starting with an initial seed model  $M_0$ , we train a series of models  $M_1, M_2$ , i.e. for  $T = 2$  iterations (we justify this choice in [Appendix B](#)). Each model  $M_{t+1}$  is trained using  $\mathcal{L}_{\text{SCPO}}$  on  $\mathcal{D}_t^{\text{pairs}}$ , the data generated by the  $t^{\text{th}}$  model, defined as follows:

$M_0$ : Seed LLM, initialized with a pretrained LLM (need not be instruction-finetuned).

$M_1$ : Initialized with  $M_0$  to generate  $\mathcal{D}_0^{\text{pairs}}$  from  $\mathcal{D}_0$  (+ new problems) and trained using  $\mathcal{L}_{\text{SCPO}}$ .

$M_2$ : Initialized with  $M_1$  to generate  $\mathcal{D}_1^{\text{pairs}}$  from  $\mathcal{D}_1$  (+ new problems) and trained using  $\mathcal{L}_{\text{SCPO}}$ .

This approach is similar to the Self-Rewarding LM training loop ([Yuan et al., 2024](#)) except for the fact that we use the model’s self-consistency to score responses instead of using the same model as a judge to verify its own correctness, which [Huang et al. \(2024\)](#) show is often challenging. In contrast to other iterative bootstrapping techniques for reasoning ([Zelikman et al., 2022](#); [Pang et al., 2024](#)), SCPO does not require access to gold labels such as gold responses or final answers, allowing SCPO to scale beyond the problems from an existing training dataset.

<sup>1</sup>By design, several responses can share a final answer (but for example, their chain-of-thought may be different). So, we cluster the responses by final answer and pick a response at random among them.

<sup>2</sup>This normalization makes sure the weights are always between 0 and 1.

**Semi-Supervised Training with SCPO.** Although SCPO does not require access to gold labels, we can easily incorporate datasets with gold labels in conjunction with unlabeled datasets during SCPO training. To this end, we alter the preference pair creation strategy described in that case. When *gold labels are available* for a query  $x_{\text{gold}}$ , we sample  $k$  responses, and create pairs such that the chosen response  $y^+$  is *correct* and the rejected response  $y^-$  is *incorrect* (discarding queries where such pairs cannot be created). Since we already know these pairs are of high quality, we set the weight of annotated instances  $w(x_{\text{gold}}) = 1$ . For queries that do not have gold labels, we use our self-consistency criterion for pair creation and compute the weighted loss for those examples as before. A special case is that if all data is labeled, the loss reduces to the IRPO loss.

### 3 EXPERIMENTAL SETUP

**Datasets and Metrics.** We evaluate the effectiveness of SCPO on a range of mathematical and logical reasoning datasets:

- **GSM8K** (Cobbe et al., 2021) contains a train/test split of 7.5K/1.3K grade school math word problems. For the purpose of this work, we split the train set into a train/dev split with 6.7K/0.8K problems respectively. We use the dev split for hyperparameter tuning and checkpoint selection. The overall data split becomes 6.7K/0.8K/1.3K in the train/dev/test set, respectively. We report performance based on exact match accuracy of the final numeric answer on the test set.
- **MATH** (Hendrycks et al., 2021) is a dataset of challenging high-school math competitions that contains a train/test split of 7.5K/5K problems, respectively. Similar to GSM8K, we reserve 10% of samples from the train set to create a held-out dev set for model selection and hyperparameter tuning, resulting in our final train/dev/test splits with 6.7K/0.8K/5K problems, respectively. We report the accuracy of the final answer on the test set.
- **ZebraLogic** (Dziri et al., 2024) is a logical reasoning benchmark. It is a test set of 1K logic grid puzzles (or Einstein’s puzzles) designed as a constraint satisfaction problem (Prosser, 1993). Each puzzle is comprised of  $n$  houses with  $m$  unique features, resulting in an  $n \times m$  table. Given a list of clues, solving the puzzle requires deducing the correct (unique) assignment of values in the table, i.e., a unique value for each feature and house. Evaluation metrics for this dataset are: puzzle accuracy (overall, easy, and hard puzzles) as well as cell accuracy.

**Base Models.** For GSM8K and MATH, we use Llama-3 Base 8B (Dubey et al., 2024) as the seed model  $M_0$ . We note that the instruction-tuned version may have already been fine-tuned on the gold data from these tasks, so new experimental settings cannot be reliably tested in that case. For ZebraLogic, we use Llama-3 Instruct 8B (Dubey et al., 2024) as the seed model.

**Preference Training Data.** We use the Llama-3 Instruct 8B model to generate additional problems (queries). For GSM8K and MATH, we prompt the model to generate a problem similar to 4-shot examples of problems from the train set. Note that the prompt only requires valid human-written problems and *not* their corresponding answers. We filter out problems where  $\max_{i \leq k} \mathcal{V}(y_i) < 0.5k$  (or,  $\tau = 0.5k$ ) where  $k$  is the number of responses sampled or votes cast for each query. That is, where less than half of the votes go towards the majority answer, which we found to be a good threshold based on the dev set accuracy (see Section 5). Since  $M_1$  models tend to be more consistent than  $M_0$  (cf. Section 5), for  $M_2$  training data, we increase the filtering threshold  $\tau$  to  $0.7k$  and  $0.6k$  on GSM8K and MATH, respectively. For ZebraLogic, we prompt the model to rephrase or perturb features of a puzzle from the dataset in a one-shot manner. Then, we use the underlying model  $M_t$  to generate  $k = 16$  responses for each question and filter out questions where none of the responses accrue  $\tau = 2$  or more votes (exactly matching solutions) for  $M_1$  and set  $\tau = 0.5k$  for training  $M_2$ .

**Baselines.** We compare models trained with SCPO in unsupervised (denoted as SCPO<sub>Unsup.</sub>) and semi-supervised (denoted as SCPO<sub>Semi-Sup.</sub>) settings against the following baselines:

- **Seed model (Zero-shot CoT).** We compare against the seed model ( $M_0$ ) using zero-shot chain-of-thought prompting (Kojima et al., 2022) generated with greedy decoding and report results with or without inference-time self-consistency (SC; Wang et al., 2023).
- **Supervised Training with Gold Answers (IRPO<sub>Gold</sub>).** We use a strong supervised preference optimization method for reasoning tasks (Pang et al., 2024), to serve as an upper-bound on per-

Method	Train Data (K)		Test Acc. (%)	
	# Seed	Gen.	Greedy	SC (8-way)
<i>without access to gold labels</i>				
Seed model (zero-shot)	$M_0$	- -	41.17	51.80
IRPO <sub>RM</sub>	iter $M_1$	5.5 -	48.67	69.98
	iter $M_2$	4.4 -	50.11	61.25
SCPO <sub>Unsup.</sub>	iter $M_1$	5.3 -	<u>61.03</u>	<b>71.49</b>
	iter $M_2$	1.4 5.1	<b>63.91</b>	<u>71.11</u>
<i>with access to gold labels</i>				
IRPO <sub>Gold</sub>	iter $M_1$	4.4 <sup>†</sup> -	61.41	72.93
	iter $M_2$	5.7 <sup>†</sup> -	64.29	72.56
SCPO <sub>Semi-Sup.</sub>	iter $M_1$	4.4 <sup>†</sup> 1.9	63.61	<u>74.30</u>
	iter $M_2$	5.7 <sup>†</sup> 4.5	<b>66.64</b>	<b>74.75</b>

Table 1: **GSM8K zero-shot accuracy** after training Llama-3 Base 8B with SCPO and baselines, using greedy or self-consistency (SC)-based inference. The best performance is in *bold*, and second-best is *underlined*. We list train set sizes for each method: “Seed” corresponds to seed problems in the train set, whereas “Gen.” indicates additional problems generated by the model (without answers). IRPO<sub>Gold</sub>, and SCPO<sub>Semi-Sup.</sub>, highlighted in *green*, use the gold answers to create preference pairs (when available, indicated with <sup>†</sup>).

formance for unsupervised training as this uses *gold data* from the train set, which we compare to unsupervised and semi-supervised SCPO. For each query  $x$ , preference pairs are constructed such that chosen responses are correct and rejected responses are incorrect with  $w(x) = 1$ .

- **Unsupervised Training with External RM (IRPO<sub>RM</sub>).** We propose a new variant of IRPO that we also expect to be a strong baseline. Given the plethora of publicly-available reward models (RMs; Lambert et al., 2024), in the absence of gold labels, off-the-shelf RMs can be used to score a set of responses  $\bar{y} \sim M_t(\cdot|x)$  and create preference pairs such that chosen and rejected responses have the maximum and minimum reward, respectively, i.e.,  $y^+ = \arg \max_{y \in \bar{y}} \text{RM}(y|x)$  and  $y^- = \arg \min_{y \in \bar{y}} \text{RM}(y|x)$  with  $w(x) = 1$ . We use the strongly performing ArmoRM-Llama3-8B model (Wang et al., 2024a) as a reward model.<sup>3</sup>

**Hyperparameters.** When generating multiple response or new problems from the LLM, we sample with temperature of 0.7 and top- $p = 0.9$ . For GSM8K and MATH, we set  $k = 8$ . With every iteration of training, the models become more consistent due to the training objective (see Section 5), thereby, making picking the rejected response harder, i.e., none of the responses are incorrect or all the responses share the same final answer. Therefore, to sample rejected responses, we further generate 8 responses sampled with a higher temperature of 1.2 to encourage more diverse answers. On ZebraLogic, due to the complex nature of the response (an  $n \times m$  table), we find that sampling a response that gets multiple votes is relatively infrequent, so we set  $k = 16$  for this task. All models are trained for 10 epochs with a learning rate of  $5e-6$  (cosine scheduling), and effective batch size of 16. Lastly, we set DPO loss term hyperparameter  $\beta = 0.5$  and NLL regularization coefficient  $\alpha = 1$ . When a dev set is available (e.g., GSM8K and MATH), we use accuracy on the dev set for checkpoint selection (at every epoch). For ZebraLogic, which is similarly challenging to MATH and does not have a train or dev set, for each iteration, we train for the same number of epochs that performed best during MATH training.

## 4 MAIN RESULTS

### 4.1 MATH REASONING

**SCPO outperforms unsupervised baselines.** Comparing methods on GSM8K, in Table 1, we observe that training with only one iteration of SCPO outperforms the zero-shot seed model and IRPO<sub>RM</sub>, by 22.74% and 12.36%, respectively, using greedy decoding. Similarly, on MATH (cf.

<sup>3</sup>Wang et al. (2024a) use training splits of GSM8K and MATH to train ArmoRM, rendering these datasets highly in-distribution for the RM while ZebraLogic is out-of-distribution (further discussed in Section 5).



Method		Train Data (K)		Test Acc. (%)	
				Greedy	SC (8-way)
<i>without access to gold labels</i>					
Seed model (zero-shot)	$M_0$	-	-	14.46	18.20
IRPO <sub>RM</sub>	iter $M_1$	6.4	-	18.06	24.20
	iter $M_2$	6.5	-	<u>18.08</u>	22.64
SCPO <sub>Unsup.</sub>	iter $M_1$	0.6	1.2	17.36	<b>25.70</b>
	iter $M_2$	1.2	2.5	<b>19.72</b>	<u>24.58</u>
<i>with access to gold labels</i>					
IRPO <sub>Gold</sub>	iter $M_1$	2.7 <sup>†</sup>	-	18.64	26.88
	iter $M_2$	3.0 <sup>†</sup>	-	<u>20.32</u>	26.88
SCPO <sub>Semi-Sup.</sub>	iter $M_1$	2.7 <sup>†</sup>	1.2	19.88	<b>27.35</b>
	iter $M_2$	3.0 <sup>†</sup>	2.2	<b>20.48</b>	<u>26.92</u>

Table 2: **MATH zero-shot accuracy** after training Llama-3 Base 8B with SCPO and baselines, using greedy or self-consistency (SC)-based inference. Train data size: “Seed” corresponds to seed queries in the train set, “Gen.” are additional model-generated problems (without answers). IRPO<sub>Gold</sub> and SCPO<sub>Semi-Sup.</sub>, highlighted in green, use gold answers to train (indicated with <sup>†</sup>).

Table 2), two iterations of SCPO<sub>Unsup.</sub> yields an improvement of 5.26% and 1.64% respectively compared to the same two baselines. We further note that while IRPO<sub>RM</sub> is not given direct access to the gold labels, it uses the ArmoRM, which has been trained on human-annotated step-level data based on MATH’s train set (Lightman et al., 2023; Wang et al., 2024a). Hence, SCPO’s improvement over IRPO<sub>RM</sub> would likely be larger if the RM had not used in-domain gold labels during training. Overall, we find SCPO has the ability to outperform RMs, especially in out-of-distribution settings.

**Iterations of SCPO improve reasoning.** From Tables 1 and 2, we observe that two iterations of SCPO consistently improves the LLM’s performance when using greedy decoding in both unsupervised and semi-supervised settings compared to one iteration. On GSM8K, greedy test accuracy improves by 2.88%, and 3.03% when using SCPO for unsupervised and semi-supervised training, respectively. Similarly, on MATH, in Table 2, we find that  $M_2$  models with SCPO outperforms their  $M_1$  counterparts by up to 2.36% in greedy accuracy. This can be explained by models becoming more accurate and consistent after one round of SCPO training (shown in Section 5). Consequently, this allows us to bootstrap from additional problems in the original and generated training data, for which the  $M_0$  model did not have a consistent response. However, we find that the accuracy computed using 8-way self-consistency (SC) saturates after the first iteration, sometimes even resulting in a slight decrease compared to  $M_1$ . This may happen because now that the model is trained to be more consistent there is less benefit from applying self-consistency at inference time (see analysis in Section 5). We find that a third iteration of training also shows minimal gains, however if we utilize the (unlabeled) problems from the test set to build preference pairs, we find that we can obtain additional performance boosts on top of  $M_2$ , as discussed in Appendix B.

**Unsupervised SCPO is comparable to IRPO training with gold labels.** We can compare the unsupervised training of SCPO with the supervised training using gold labels of IRPO in Tables 1 and 2. The results show that SCPO<sub>Unsup.</sub> *without using any gold labels* can yield comparable accuracy to IRPO<sub>Gold</sub> on GSM8K and MATH with  $< 1\%$  gap in greedy performance and  $< 2\%$  gap in accuracy using 8-way self-consistency after two iterations of training ( $M_2$ ). This comparable performance of SCPO<sub>Unsup.</sub> is likely due to high correlation (0.8 across the datasets) between the vote shares and accuracy on the test set, as further discussed in Appendix A. Note that on tasks that are challenging for the seed model  $M_0$ , such as MATH, we can only bootstrap a small set of examples from the original set of training problem as compared to IRPO (i.e., only around a quarter of examples obtain a clear majority answer). However, we can offset this gap in training data by generating new problems using few-shot prompting (cf. Section 2) and creating preference pairs using our self-consistency method. This helps provide improvements during the second iteration.

**Semi-supervised training with SCPO outperforms IRPO.** Lastly, in Tables 1 and 2, we evaluate the semi-supervised version of SCPO *combined with using gold labels*. We find that on GSM8K, SCPO<sub>Semi-Sup.</sub> improves the greedy accuracy by 2.35% and SC accuracy by 2.19% in comparison

Method	Train Data (K) # Seed / Gen.	Puzzle Acc. (%)			Cell Acc. (%)
		Overall	Easy	Hard	
Llama-3 Instruct 70B	- / -	<u>17.2</u>	52.1	<b>3.6</b>	42.9
Gemma-2 27B IT*	- / -	16.3	50.7	<u>2.9</u>	41.2
Claude-3 Haiku*	- / -	14.3	47.9	<u>1.2</u>	37.9
$M_0$ (Llama-3 Instruct 8B)	- / -	11.6	40.0	0.4	39.1
$M_1$ w/ IRPO <sub>RM</sub>	1.0 / -	11.3	37.9	1.0	42.1
$M_1$ w/ SCPO <sub>Unsup.</sub>	0.4 / 1.0	17.0	<u>54.3</u>	2.5	<b>47.6</b>
$M_2$ w/ SCPO <sub>Unsup.</sub>	0.4 / 2.2	<b>18.1</b>	<b>58.2</b>	2.5	<u>45.2</u>

Table 3: **ZebraLogic test performance** after unsupervised training of Llama-3 Instruct 8B with SCPO, compared to baselines. “Seed” corresponds to original puzzles in the test set, whereas “Gen.” indicates additional puzzles generated. \*Model’s performance reported from the [Leaderboard](#).

to IRPO<sub>Gold</sub>. Similar trends hold on the MATH dataset, where one iteration of SCPO<sub>Semi-Sup.</sub> outperforms IRPO<sub>Gold</sub> by 1.24% using greedy decoding. These results show the utility of using SCPO to bootstrap from model-generated problems even given access to a labeled training set.

#### 4.2 ZEBRALOGIC: A CHALLENGING LOGICAL REASONING TASK

**SCPO outperforms unsupervised baselines.** Table 3 reports performance on ZebraLogic of SCPO and various baselines, using greedy decoding. We observe large improvements over the seed model, Llama-3 Instruct 8B ( $M_0$ ) with one iteration of unsupervised SCPO ( $M_1$ ), improving performance by 5.4% and 8.5% in overall puzzle accuracy (exact match of tables) and cell accuracy (match of each cell in the table), respectively. In contrast, unsupervised training of IRPO<sub>RM</sub> yields only mild gains over the seed model by 3% in cell accuracy and even a slight drop in puzzle accuracy (11.6% to 11.3%). This can be attributed to ZebraLogic puzzles being out-of-distribution for the ArmoRM (cf. Section 5), thus trailing behind one iteration of SCPO by 5.7% in puzzle accuracy and 5.5% in cell accuracy. Overall, training with SCPO for two iterations improves the performance of the seed model by 8 positions on the leaderboard (from 38<sup>th</sup> to 30<sup>th</sup>) with a 6.5% boost in puzzle accuracy and, to the best of our knowledge, is the best 8B-scale LLM on ZebraLogic.

**8B LLM trained with SCPO outperforms larger models.** Comparison of SCPO-trained models to other models in Table 3 demonstrates that SCPO-training after two iterations ( $M_2$ ) outperforms significantly larger models such as Llama-3 Instruct 70B, Gemma-2 27B, and Claude-3 Haiku by 0.9%, 1.8%, and 3.8% in overall puzzle accuracy, respectively. Additionally, we find that models trained using SCPO also yield the highest cell accuracy. We attribute these gains over larger models to the substantial improvement in solving easy puzzles with SCPO (up to 10.3%).

### 5 ABLATIONS AND ANALYSIS

#### Importance of weighted SCPO loss.

While the results in Section 4 are obtained using the weighted  $\mathcal{L}_{\text{SCPO}}$  loss that is a function of consistency, here we compare SCPO using an unweighted loss. More specifically, we train using the same preference dataset created based on self-consistency of responses, but with  $w(x) = 1$  in the  $\mathcal{L}_{\text{SCPO}}$  loss. In Table 4, we observe that across *datasets* and *iterations*, the weighted loss consistently outperforms the unweighted version. The improvement in accuracy is even more pronounced for the first iteration of training  $M_1$ , yielding an improvement of 2.5% in accuracy on GSM8K and 1.44% on MATH with

	Method	Train (K) # Seed / Gen.	Test Acc. (%)	
			Greedy	SC (8-way)
GSM8K	$M_1$ w/ $w(x)=1$	5.3 / -	58.53	69.07
	$M_2$ w/ $w(x)=1$	1.4 / 5.1	62.62	69.90
	$M_1$ w/ SCPO <sub>Unsup.</sub>	5.3 / -	61.03	71.49
	$M_2$ w/ SCPO <sub>Unsup.</sub>	1.4 / 5.1	63.91	71.11
MATH	$M_1$ w/ $w(x)=1$	0.6 / 1.2	15.92	25.34
	$M_2$ w/ $w(x)=1$	1.2 / 2.5	18.74	25.58
	$M_1$ w/ SCPO <sub>Unsup.</sub>	0.6 / 1.2	17.36	25.70
	$M_2$ w/ SCPO <sub>Unsup.</sub>	1.2 / 2.5	19.72	24.58

Table 4: Ablation comparing unweighted loss ( $w(x) = 1$ ) to the proposed weighted loss used in SCPO. SCPO outperforms the unweighted loss in all cases.

greedy inference. Even in the second iteration,  $M_2$  models trained with SCPO outperform their unweighted counterparts by roughly 1% on both GSM8K and MATH. This indicates that it is better to take the amount of votes into account when optimizing for consistency, as this indicates confidence in the chosen and rejected labeling.

#### Models become more consistent across iterations.

In Figure 2, we analyze how the degree of model consistency varies across iterations. To this end, we measure the vote share  $\mathcal{V}(y^+)/k$  of the most consistent response, i.e., chosen response in self-consistency of models trained using unsupervised SCPO. From Figure 2, we conclude that SCPO training increases the consistency of models with each training iteration across different tasks. We suspect this finding stems from three contributing factors: (i) with increasing iterations models become more accurate (Section 4); (ii) additional rounds of preference-optimization decreases model diversity (Kirk et al., 2024); and (iii) training with SCPO effectively distills the SC distribution into the model’s single-sample distribution. Additionally, we find that models are more consistent on tasks with higher test accuracy, i.e., on GSM8K the LLM is most consistent and accurate whereas on ZebraLogic it is the least consistent and accurate.

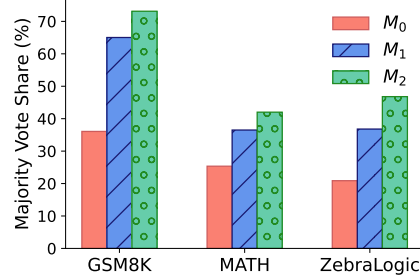


Figure 2: Vote share (%) of the most consistent response:  $\mathcal{V}(y^+)/k$  increases with iterations across all datasets.

#### Impact of consistency-based filtering on constructing preferences.

In Section 3, when generating self-consistency preference data for GSM8K and MATH, we filter out instances where fewer than half of the votes go towards the majority answer, i.e.,  $\tau = 0.5k$ . The choice of this threshold presents a trade-off between the *number of preference pairs* available for training and the *quality of the training data*, and affects the difference (margin) in accuracy of the chosen and the rejected response. Assuming access to the gold answers to measure quality of preference data, in Table 5, we analyze this trade-off on MATH. As the vote threshold increases from  $\tau = 0.1k$  to  $\tau = 0.7k$ , the quality of training preference pairs increases, with the accuracy margin increasing from 18% to 68%. On the other hand, the size of the training data decreases from 6.7K pairs to fewer than 700 pairs. Interestingly, Table 5 shows that as we vary the threshold, the performance of the trained model increases till  $\tau = 0.5k$  and then decreases. In other words, from  $\tau = 0.1k$  to  $\tau = 0.5k$  the quality of the preference data (or the accuracy margin) takes precedence over the quantity, improving downstream performance by 1.92%. However, when we set  $\tau = 0.7k$ , we end up with fewer than 700 pairs to train which we suspect is insufficient (in terms of both data size and diversity) to train a model with 8B parameters.

Setting	Margin	# Train	Test Acc.
$M_0$	-	-	14.46
$M_1$ ( $\tau = 0.1k$ )	18%	6.7K	15.44
$M_1$ ( $\tau = 0.3k$ )	44%	2.4K	16.34
$M_1$ ( $\tau = 0.5k$ )	57%	1.8K	17.36
$M_1$ ( $\tau = 0.7k$ )	68%	0.7K	14.76

Table 5: Impact of using different thresholds on majority vote to filter training data on MATH. Margin (%) denotes the difference in accuracy of the chosen and rejected response.

#### Comparison of self-consistency to RMs.

Our results in Section 4 show that models trained with unsupervised SCPO outperform models trained with IRPO using ArmoRM to build preference pairs. To study this further, we conduct additional analysis by measuring the ability of the two methods to distinguish between correct and incorrect responses, comparing the methods to gold labels. Results are given in Figure 3. We find that ArmoRM consistently has more incorrect orderings of pairwise preferences (the chosen is incorrect and the rejected is correct) than SCPO across all three datasets (shown in red). This added noise in training may be a major factor as to why IRPO<sub>RM</sub>

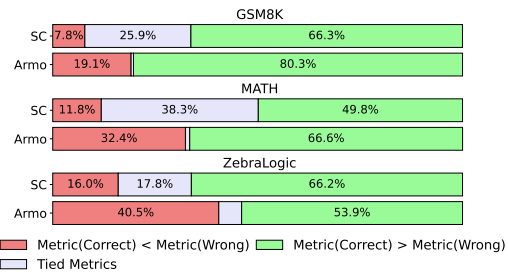


Figure 3: Comparing the quality of metrics: self-consistency (SC) and ArmoRM to distinguish between correct and incorrect responses.



performs poorly compared to  $\text{SCPO}_{\text{Unsup}}$ . On the other hand, self-consistency results in a greater number of ties, i.e., when the chosen and rejected answers get the same number of votes; these are ignored in SCPO’s loss since  $w(x) = 0$ . Lastly, we find that in out-of-distribution settings like ZebraLogic, self-consistency outperforms ArmoRM with 12.3% higher correct orderings of pairwise preferences (shown in green in Figure 3).

## 6 RELATED WORK

**Iterative Training of LLMs.** Iterative training or self-training has shown meaningful improvements in a number of domains such as safety (Bai et al., 2022), multilingual reasoning (She et al., 2024), and evaluation (Wang et al., 2024b). Because LLMs often struggle with both generating and validating solutions to complex reasoning tasks, prior works on training LLMs for complex problem-solving tasks largely rely on human-annotated (gold) final answers (Zelikman et al., 2022; Chen et al., 2024b; Pang et al., 2024) or access to an external reward model that performs well on the underlying task (Singh et al., 2024; Dong et al., 2023). However, both these classes of approaches suffer from their own shortcomings. Firstly, manually annotating or verifying the final answer requires working through the solution step-by-step, making it especially resource-intensive for complex multi-step problems. Training strong reward models for such reasoning and problem-solving tasks also often requires human judgements of LLM generations (Cobbe et al., 2021; Uesato et al., 2022; Lightman et al., 2023), making it similarly expensive. Our work focuses on the setting *without access to gold solutions or final answers*, which remains largely unaddressed. While other works such as She et al. (2024); Yuan et al. (2024); Rosset et al. (2024); Tran et al. (2023) geared towards general instruction following tasks (as opposed to reasoning tasks specifically) circumvent the need for human-annotated labels in the dataset by using the model itself to score the responses, these works demonstrate only modest gains in the context of reasoning tasks.

**Consistency in LLMs.** Self-consistency (Wang et al., 2023) relies upon the intuition that sampling several responses, some of which lead to the same answer, lends higher certainty that the consistent answer is the correct one. Application of self-consistency *at inference time* has enabled performance improvements in a number of domains like math (Wang et al., 2023), code generation (Shi et al., 2022; Li et al., 2022; Chen et al., 2018), and even open-ended tasks like summarization and question answering (Chen et al., 2024a). In this work, we explore using self-consistency *at training time* for reasoning tasks, constructing preference pairs according to the self-consistent final answer. We employ a preference optimization loss function that is weighted according to the consistency of an answer. Intuitively, the consistency of an answer is a reflection of the model confidence, and several prior works have demonstrated that leveraging model uncertainty can lead to faster convergence and improved performance (Gal & Ghahramani, 2016; Krishnan & Tickoo, 2020; Corbière et al., 2019).

## 7 CONCLUSION

In this paper, we introduced Self-Consistency Preference Optimization (SCPO). SCPO leverages the concept of self-consistency, usually employed only at inference time, to improve the self-training of large language models. By iteratively optimizing to prefer consistent answers to inconsistent ones, SCPO achieves significant improvements over traditional reward model training without the need for additional gold labels. Our experiments demonstrate the efficacy of SCPO on various reasoning tasks, including GSM8K, MATH, and ZebraLogic, where in the latter it outperforms several larger state-of-the-art language models. We also showed that SCPO works well in semi-supervised setups with access to some gold labels, in addition to unlabeled inputs – improving performance further. These results highlight the potential of SCPO to improve self-alignment across reasoning tasks – a domain that prior self-alignment methods still struggle with. Future work could extend SCPO to tasks where a single final answer cannot be easily parsed (e.g., summarization) through universal self-consistency (Chen et al., 2024a), which leverages an LLM to select the most consistent answer among multiple samples. While we explore consistency in this work according to one model class (Llama-3 8B Base and Instruct), future work could also investigate consistency according to a suite of other models and tasks.

## REFERENCES

- Anthropic. The claude 3 model family: Opus, sonnet, haiku. 2024. URL <https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model.Card.Claude.3.pdf>.
- Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.
- Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, Jérémy Scheurer, Javier Rando, Rachel Freedman, Tomasz Korbak, David Lindner, Pedro Freire, et al. Open problems and fundamental limitations of reinforcement learning from human feedback. *arXiv preprint arXiv:2307.15217*, 2023.
- Bei Chen, Fengji Zhang, Anh Nguyen, Daoguang Zan, Zeqi Lin, Jian-Guang Lou, and Weizhu Chen. Codet: Code generation with generated tests. In *The Eleventh International Conference on Learning Representations*, 2023.
- Xinyun Chen, Chang Liu, and Dawn Song. Execution-guided neural program synthesis. In *International Conference on Learning Representations*, 2018.
- Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. Universal self-consistency for large language models. In *ICML 2024 Workshop on In-Context Learning*, 2024a. URL <https://openreview.net/forum?id=LjsjHF7nAN>.
- Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. Self-play fine-tuning converts weak language models to strong language models. In *Forty-first International Conference on Machine Learning*, 2024b. URL <https://openreview.net/forum?id=04cHTxW9BS>.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- Charles Corbière, Nicolas Thome, Avner Bar-Hen, Matthieu Cord, and Patrick Pérez. Addressing failure prediction by learning model confidence. *Advances in Neural Information Processing Systems*, 32, 2019.
- Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, KaShun Shum, and Tong Zhang. RAFT: Reward ranked finetuning for generative foundation model alignment. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL <https://openreview.net/forum?id=m7p507zbly>.
- Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.
- Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, et al. Faith and fate: Limits of transformers on compositionality. *Advances in Neural Information Processing Systems*, 36, 2024.
- Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.
- Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Maria Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1050–1059, New York, New York, USA, 20–22 Jun 2016. PMLR. URL <https://proceedings.mlr.press/v48/gal16.html>.

- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.
- Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=Ikmd3fKBPQ>.
- Anubha Kbra, Sanketh Rangreji, Yash Mathur, Aman Madaan, Emmy Liu, and Graham Neubig. Program-aided reasoners (better) know what they know. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 2262–2278, 2024.
- Robert Kirk, Ishita Mediratta, Christoforos Nalmpantis, Jelena Luketina, Eric Hambro, Edward Grefenstette, and Roberta Raileanu. Understanding the effects of rlhf on llm generalisation and diversity. In *The Twelfth International Conference on Learning Representations*, 2024.
- Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.
- Ranganath Krishnan and Omesh Tickoo. Improving model calibration with accuracy versus uncertainty optimization. *Advances in Neural Information Processing Systems*, 33:18237–18248, 2020.
- Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, et al. Rewardbench: Evaluating reward models for language modeling. *arXiv preprint arXiv:2403.13787*, 2024.
- Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022.
- Xun Liang, Shichao Song, Zifan Zheng, Hanyu Wang, Qingchen Yu, Xunkai Li, Rong-Hua Li, Feiyu Xiong, and Zhiyu Li. Internal consistency and self-feedback in large language models: A survey. *arXiv preprint arXiv:2407.14507*, 2024.
- Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let’s verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.
- Dakota Mahan, Duy Van Phung, Rafael Rafailov, Chase Blagden, Nathan Lile, Louis Castricato, Jan-Philipp Fränken, Chelsea Finn, and Alon Albalak. Generative reward models. *arXiv preprint arXiv:2410.12832*, 2024.
- Richard Yuanzhe Pang, Weizhe Yuan, Kyunghyun Cho, He He, Sainbayar Sukhbaatar, and Jason Weston. Iterative reasoning preference optimization. *arXiv preprint arXiv:2404.19733*, 2024.
- Patrick Prosser. Hybrid algorithms for the constraint satisfaction problem. *Computational intelligence*, 9(3):268–299, 1993.
- Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024.
- Corby Rosset, Ching-An Cheng, Arindam Mitra, Michael Santacrose, Ahmed Awadallah, and Tengyang Xie. Direct nash optimization: Teaching language models to self-improve with general preferences. *arXiv preprint arXiv:2404.03715*, 2024.
- Shuaijie She, Shujian Huang, Wei Zou, Wenhao Zhu, Xiang Liu, Xiang Geng, and Jiajun Chen. MAPO: Advancing multilingual reasoning through multilingual alignment-as-preference optimization. *arXiv preprint arXiv:2401.06838*, 2024.

- Freda Shi, Daniel Fried, Marjan Ghazvininejad, Luke Zettlemoyer, and Sida I Wang. Natural language to code translation with execution. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 3533–3546, 2022.
- Avi Singh, John D Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J Liu, James Harrison, Jaehoon Lee, Kelvin Xu, Aaron T Parisi, Abhishek Kumar, Alexander A Alemi, Alex Rizkowsky, Azade Nova, Ben Adlam, Bernd Bohnet, Gamaleldin Fathy Elsayed, Hanie Sedghi, Igor Mordatch, Isabelle Simpson, Izzeddin Gur, Jasper Snoek, Jeffrey Pennington, Jiri Hron, Kathleen Kenealy, Kevin Swersky, Kshiteej Mahajan, Laura A Culp, Lechao Xiao, Maxwell Bileschi, Noah Constant, Roman Novak, Rosanne Liu, Tris Warkentin, Yamini Bansal, Ethan Dyer, Behnam Neyshabur, Jascha Sohl-Dickstein, and Noah Fiedel. Beyond human data: Scaling self-training for problem-solving with language models. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL <https://openreview.net/forum?id=1NAyUngGFK>. Expert Certification.
- Robert H Somers. A new asymmetric measure of association for ordinal variables. *American sociological review*, pp. 799–811, 1962.
- Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.
- 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. Llama 2: Open foundation and fine-tuned chat models, 2023. URL <https://arxiv.org/abs/2307.09288>.
- Hoang Tran, Chris Glaze, and Braden Hancock. Iterative DPO alignment. Technical report, Snorkel AI, 2023.
- Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.
- Haoxiang Wang, Wei Xiong, Tengyang Xie, Han Zhao, and Tong Zhang. Interpretable preferences via multi-objective reward modeling and mixture-of-experts. *arXiv preprint arXiv:2406.12845*, 2024a.
- Tianlu Wang, Ilia Kulikov, Olga Golovneva, Ping Yu, Weizhe Yuan, Jane Dwivedi-Yu, Richard Yuanzhe Pang, Maryam Fazel-Zarandi, Jason Weston, and Xian Li. Self-taught evaluators. *arXiv preprint arXiv:2408.02666*, 2024b.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL <https://openreview.net/forum?id=1PL1NIMMrw>.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.
- Miao Xiong, Zhiyuan Hu, Xinyang Lu, YIFEI LI, Jie Fu, Junxian He, and Bryan Hooi. Can LLMs express their uncertainty? an empirical evaluation of confidence elicitation in LLMs. In *The Twelfth International Conference on Learning Representations*, 2024.

- Jing Xu, Andrew Lee, Sainbayar Sukhbaatar, and Jason Weston. Some things are more cringe than others: Preference optimization with the pairwise cringe loss. *arXiv preprint arXiv:2312.16682*, 2023.
- Longhui Yu, Weisen Jiang, Han Shi, Jincheng YU, Zhengying Liu, Yu Zhang, James Kwok, Zhengguo Li, Adrian Weller, and Weiyang Liu. MetaMath: Bootstrap your own mathematical questions for large language models. In *The Twelfth International Conference on Learning Representations*, 2024.
- Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason E Weston. Self-rewarding language models. In *Forty-first International Conference on Machine Learning*, 2024. URL <https://openreview.net/forum?id=0NphYcmgua>.
- Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. STaR: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.
- Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction. *arXiv preprint arXiv:2408.15240*, 2024.



## A RELATIONSHIP BETWEEN CONSISTENCY AND ACCURACY

### Level of consistency or vote share correlates with accuracy.

We observe that the degree of consistency, or vote share, is positively and strongly correlated with accuracy. This relationship is evidenced in Table 6 by a high rank order correlation for all three datasets, as determined by Somer’s D (Somers, 1962), which measures the degree of association between two possibly dependent variables. This association is lowest for MATH, likely because the challenging nature of this task makes it difficult for the model to produce consistent answers.

Dataset	Somers’ D
GSM8K	0.80
MATH	0.68
ZebraLogic	0.92

Table 6: Somers’ D computed between  $\text{Acc}(y)$  and  $\mathcal{V}(y)$  for  $y \in \{y^+, y^-\}$  on test set.

Method	GSM8K Acc.		MATH Acc.	
	Greedy	SC (8-way)	Greedy	SC (8-way)
$M_0$	41.17	58.80	14.46	18.20
$M_1$ w/ SCPO <sub>Unsup.</sub>	61.03	<b>71.49</b>	17.36	25.70
$M_2$ w/ SCPO <sub>Unsup.</sub>	63.91	71.11	19.72	24.58
$M_3$ w/ SCPO <sub>Unsup.</sub>	64.21	70.81	19.76	24.66
$M_3$ w/ SCPO <sub>Unsup.</sub> on test queries	<b>65.35</b>	70.96	<b>20.00</b>	<b>25.84</b>

Table 7: Additional iteration of training ( $M_3$ ) by bootstrapping from questions in the train and test set. On GSM8K, we bootstrap 8.7K pairs using train problems and 5.8K pairs with the test problems. On MATH, we build 4.4K, and 4.2K preference pairs using train and test problems, respectively.

## B TRANSDUCTION DURING INFERENCE

**Bootstrapping preference pairs from test queries further boosts performance.** In our primary experiments, we report results for two rounds of iterative training. However, as shown in Table 7, introducing a third round of SCPO yields only marginal improvements, with gains of less than 1% over the second round. To address this saturation, we explore generating new problems and building preference pairs using the queries from test split as exemplars instead of the train split. This strategy results in more substantial improvements (+1.44% for GSM8K), as it enables the model to better adapt to the unique characteristics of the test set. For MATH, we see more substantial improvements when using SC accuracy, resulting in an improvement bump of 1.26%. We note that ZebraLogic is excluded from this analysis, as it only provides test samples.

## C PROMPTS

In this section, we provide all task-specific prompts used for both generating new problems and for generating candidate solutions.

### Response Generation: ZebraLogic

#### Example Puzzle:

There are 3 houses, numbered 1 to 3 from left to right, as seen from across the street. Each house is occupied by a different person. Each house has a unique attribute for each of the following characteristics:

- Each person has a unique name: 'Peter', 'Eric', 'Arnold'.
- Each person has a unique favorite drink: 'tea', 'water', 'milk'

#### ## Clues:

1. Peter is in the second house.
2. Arnold is directly left of the one who only drinks water.
3. The one who only drinks water is directly left of the person who likes milk.

#### Answer to the Example Puzzle:

```
{
  "reasoning": "Given Clue 1, we know Peter is in House 2. According to Clue 2, Arnold is directly left of the one who only drinks water. The person in House 3 cannot be on the left of anyone, so Arnold must be in House 1. Thus, Peter drinks water, and Eric lives in House 3. Then, according to Clue 3, Eric drinks milk. Therefore, Arnold drinks tea.",
  "solution": {
    "House 1": { "Name": "Arnold", "Drink": "tea" },
    "House 2": { "Name": "Peter", "Drink": "water" },
    "House 3": { "Name": "Eric", "Drink": "milk" }
  }
}
```

#### Puzzle to Solve: {puzzle}

**Prompt:** Now please solve the above puzzle. Present your reasoning and solution in the following json format: {json template}

### Response Generation: GSM8K

**Prompt:** Answer the following question step-by-step. When you are ready, place the final answer in a new line as ##### < number >.

**Q:** {question}

**A:** Let's think step by step.

### Response Generation: MATH

**Prompt:** Answer the following question step-by-step. When you are ready, place the final answer in a new line as: The final answer is  $\boxed{< \text{your answer}>}$

**Q:** {question}

**A:** Let's think step by step.

### Query Generation: ZebraLogic

#### Example Puzzle:

*Attributes to Change:* ["Name", "Drink"]

“ There are 3 houses, numbered 1 to 3 from left to right, as seen from across the street. Each house is occupied by a different person. Each house has a unique attribute for each of the following characteristics:

- Each person has a unique name: 'Peter', 'Eric', 'Arnold'.
- Each person has a unique favorite drink: 'tea', 'water', 'milk'

## Clues:

1. Peter is in the second house.
2. Arnold is directly left of the one who only drinks water.
3. The one who only drinks water is directly left of the person who likes milk.

#### Answer:

Let's change the "Name" and "Drink" attributes of the given puzzle to create a new puzzle. There are 3 names and drinks involved. Mentions of "Name" changes from 'Peter', 'Eric', 'Arnold' to mentions of "Name": 'Molly', 'Shannon', 'Kelly' respectively. Instead of "Drink" as the attribute, let's their "Food" preferences as the attribute. So mentions of "Drink" changes from 'tea', 'water', 'milk' to mentions of "Food": 'pizza', 'burgers', 'fries' respectively. Now, changing the language of the puzzle and clues we get,

*New Attribute Map:* {"Name": "Name", "Drink": "Food"}

*Puzzle:*

” There are 3 houses, numbered 1 to 3 from left to right, as seen from across the street. Each house is occupied by a different person. Each house has a unique attribute for each of the following characteristics:

- Each person has a unique name: 'Molly', 'Shannon', 'Kelly'.
- Each person has a unique favorite food: 'pizza', 'burgers', 'fries'

## Clues:

1. Molly is in the second house.
2. Kelly is directly left of the one who only eats burgers.
3. The one who only eats burgers is directly left of the person who likes fries.

#### Puzzle to rephrase:

*Attributes to Change:* {attributes dict}

“ {input puzzle} ”

**Prompt:** Rephrase the above puzzle by changing only the attributes above. ALWAYS mention the "New Attribute Map" and enclose the new puzzle within “ ”. Aside from these attributes keep the logic of the puzzle as similar as possible. Similar to the example above, give your reasoning before rephrasing the puzzle.

### Query Generation: GSM8K and MATH

**Q:** {few-shot question 1}

**Q:** {few-shot question 2}

**Q:** {few-shot question 3}

**Q:** {few-shot question 4}

**Prompt:** Based on the examples above, generate ONE solvable math word problem with similar difficulty. Note that all the information needed to solve the problem should be included in the question. Output the question and nothing else.

**Q:**