SPARE: Single-Pass Annotation with Reference-Guided Evaluation for Automatic Process Supervision and Reward Modelling

Anonymous ACL submission

Abstract

Process or step-wise supervision has played a crucial role in advancing complex multi-step reasoning capabilities of Large Language Models (LLMs). However, efficient, high-quality automated process annotation remains a significant challenge. To address this, we introduce **S**ingle-**P**ass **A**nnotation with **R**eference-Guided **E**valuation (**SPARE**), a novel structured framework that enables single-pass, per-step annotation by aligning each solution step to one or multiple steps in a reference solution, accompanied by explicit reasoning for evaluation. We show that reference-guided step-level evaluation effectively facilitates process supervision on four datasets spanning three domains: mathematical reasoning, multi-hop compositional question answering, and spatial reasoning. We demonstrate that SPARE, when compared to baselines, improves reasoning performance when used for: (1) fine-tuning models in an offline RL setup for inference-time greedy-decoding, and (2) training reward models for ranking/aggregating multiple LLM-generated outputs. Additionally, SPARE achieves competitive performance on challenging mathematical datasets while offering 2.6 times greater efficiency, requiring only 38% of the runtime, compared to tree search-based automatic annotation. The codebase provided along with the submission will be made publicly available[1].

1 Introduction

While large language models (LLMs) have demonstrated strong performance across a broad range of tasks (Brown et al., 2020; Wei et al., 2022a,b; Chowdhery et al., 2023; Touvron et al., 2023; Srivastava et al., 2023), complex multi-step reasoning still remains a challenge for LLMs even when they are trained and finetuned with ground-truth chains of thoughts (Azerbayev et al., 2024; Yu et al., 2024b). Self-consistency (Wang et al., 2023) can improve performance by voting over multiple generations, only if the answers are correct in majority of them. To address this, reward models trained to assess output correctness have gained popularity. Outcome Reward Models (ORMs) (Cobbe et al., 2021; Yu et al., 2024a) are trained using outcome supervision relying on the correctness of the final answer, while Process Reward Models (PRMs) (Uesato et al., 2022; Lightman et al., 2024) use process supervision that relies on the correctness of individual reasoning steps.

PRMs achieve better performance due to the targeted step-level feedback but suffer from expensive and complex annotation requirements. Human-supervision (Uesato et al., 2022; Lightman et al., 2024) is very demanding in terms of highly skilled human evaluators, motivating efforts toward automatic process annotation largely driven by Monte Carlo Tree Search (MCTS)-based methods (Wang et al., 2024a,c; Luo et al., 2024; Zhang et al., 2024b). In MCTS-based approaches, models are initially trained on ground-truth reasoning traces and answers through supervised fine-tuning. However, during step evaluation, these methods overlook the valuable step-by-step information *already present* in the reference ground-truth rationales. Instead, they rely exclusively on final answer matching across multiple model rollouts, resulting in both computational inefficiency and *under-utilization* of the data already available at hand.

Parallel efforts aim to leverage valuable signals from reference reasoning traces, that are either *existing* ground-truth or *synthetically* generated rationales. For instance, Li et al. (2023); Khalifa et al. (2023) generated step-level annotations by decomposing candidate and reference solutions into individual steps, performing alignment using dataset-specific discriminative models with limited applicability to challenging datasets like MATH (Hendrycks et al., 2021), and annotat-

[1]The codebase can be accessed at `URL-placeholder`.

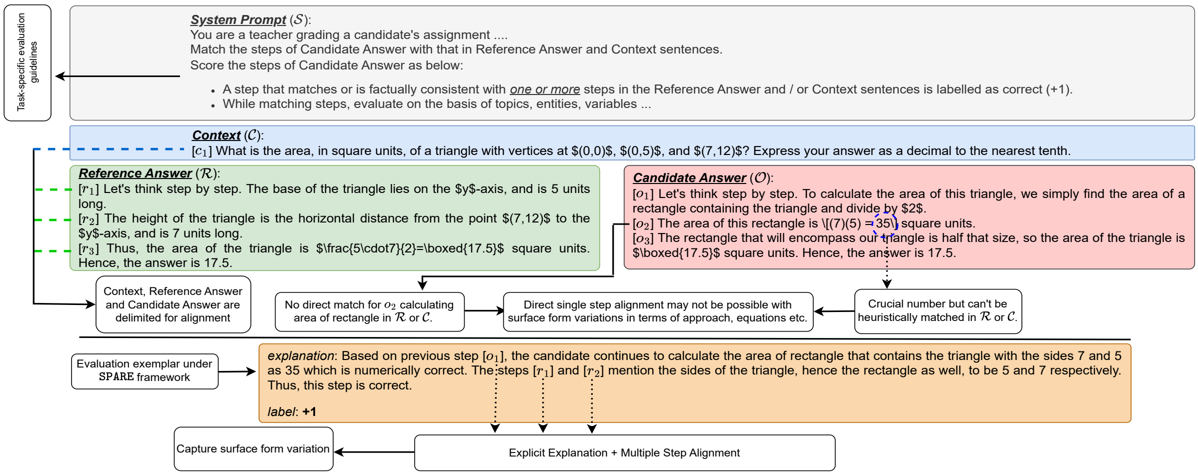


Figure 1: Determining the correctness of a step in a candidate solution against a given context and reference solution presents several challenges, including step alignment, surface-form variations, and heuristic limitations. We propose a unified, single-stage framework: **S**ingle **P**ass **A**nnotation with **R**eference-Guided **E**valuation (**SPARE**: $(\mathcal{S}, \mathcal{C}, \mathcal{R}, \mathcal{O}) \rightarrow \mathcal{E}$). SPARE produces an explanation-based step-by-step evaluation $\mathcal{E}$ of a candidate model output $\mathcal{O}$, grounded to a given context $\mathcal{C}$, reference reasoning $\mathcal{R}$, and system prompt $\mathcal{S}$ (Section 3.1).

ing steps in a restricted context where a candidate step is matched to a single reference step. AutoPRM (Chen et al., 2024) decomposes reference solutions into sub-questions and corresponding solutions to enable process supervision. However, their approach relies on an auxiliary model for data collection, which is trained using outputs from a more capable language model. More recently, GenRM (Zhang et al., 2025) employed reference-guided grading to train verifiers using synthetically generated rationales as references. However, GenRM is not a process supervision (PRM) model and relies on rationales generated by a more capable model than the one trained as the reward model. Similarly, ThinkPRM (Khalifa et al., 2025) utilized a more capable model to generate synthetic verification rationales, which were subsequently filtered to retain only those whose step annotations aligned with human-labeled steps in the PRM800K dataset (Lightman et al., 2024).

To address these gaps, we propose **S**ingle-**P**ass **A**nnotation with **R**eference-Guided **E**valuation (SPARE), a framework (Figure 1) that enables automatic process supervision through step-level evaluation of model responses by leveraging intermediate steps from reference reasoning traces. SPARE introduces a generic, structured evaluation scheme focused on (i) explicit reasoning for each step evaluation and (ii) multi-step aligned comparisons between the model output and reference. This design enables single-pass evaluation with additive scaling relative to the token lengths of the response and reference. Our experiments across three domains demonstrate that SPARE annotations outperform self-consistency and outcome supervision, while matching the performance of tree-search-based annotation methods with greater efficiency on the challenging MATH-500 dataset. In summary, our contributions are as follows:

- We introduce SPARE, a general, single-pass, and structured reference-guided evaluation framework for process annotation, emphasizing explicit annotation reasoning and multi-step alignment. Importantly, SPARE does not require additional reasoning traces but reuses the same reference solutions already available for standard supervised fine-tuning (SFT).

- We utilize SPARE annotations to improve reasoning capabilities of LLMs under two settings: (i) fine-tuning a model in an offline reinforcement learning (RL) setup for greedy-decoding during inference, and (ii) training Reward Models (RMs) for ranking and aggregating multiple generations.

- We extensively evaluate our framework on two mathematical reasoning datasets (GSM8K and MATH), a question-answering dataset (🎵 MuSiQue-Ans), and a spatial reasoning dataset (SpaRP), demonstrating improvements over baselines. Additionally, we show that our approach is competitive with tree search-based methods which are computationally expensive.

2 Related Work

Reasoning abilities of LLMs. Reasoning remains a challenging area for the Large Language Models (LLMs). Various prompting techniques, such as chain-of-thought, few-shot prompting and their variants (Wei et al., 2022b; Kojima et al., 2022; Yao et al., 2023; Hao et al., 2023; Bi et al., 2024) elicited reasoning capabilities in LLMs. Importance of individual steps while prompting (Fu et al., 2023; Zhou et al., 2023) was soon found to be crucial in successfully solving multi-step reasoning problems. While prompt-only techniques show promising results, their performances are constrained by and sensitive to prompt design and nature of tasks (Ye and Durrett, 2022). Consequently, explicitly finetuning with high-quality reasoning traces for improving LLM reasoning capabilities has become popular (Yu et al., 2024b; Luo et al., 2025).

Outcome and Process Supervision. Supervised finetuning quickly results in saturation, leading to the search for other advanced techniques and better supervision signals. Outcome supervision (Cobbe et al., 2021; Yu et al., 2024a) relies on signal based on the final answer, and hence, is easier to obtain. Process supervision offers advantages in the form of fine-grained feedback from individual reasoning steps, however, early work (Uesato et al., 2022; Pan et al., 2023; Lightman et al., 2024) relied on time-consuming and costly human annotation. To alleviate this problem, several recent approaches have emerged for automating process supervision. Monte-Carlo Tree Search (MCTS) based approaches (Wang et al., 2024a,c; Luo et al., 2024; Zhang et al., 2024a) target obtaining process annotation by several continuations from intermediate steps whose correctness are evaluated based on the final step. Parallel work has explored leveraging reference reasoning traces, either ground-truth or synthetic, for supervision. Prior approaches decompose solutions into steps for alignment (Li et al., 2023; Khalifa et al., 2023), often relying on dataset-specific models with limited generalizability. Others, like AutoPRM (Chen et al., 2024), use subquestion decomposition but depend on auxiliary models trained with outputs from stronger LLMs. GenRM (Zhang et al., 2025) and ThinkPRM (Khalifa et al., 2025) use synthetic rationales from more capable models for training verifiers, but do not provide a general-purpose, reference-guided process supervision framework. In contrast to these efforts, our work proposes a unified, single-pass, and structured evaluation framework for automatic process annotation, enabling flexible alignment and multi-step comparison with reference solutions. We further demonstrate its effectiveness across both fine-tuning and verification settings.

3 Our Approach

3.1 Single-Pass Annotation with Reference-Guided Evaluation (SPARE)

Consider a reference reasoning path $\mathcal{R} = \{r\}_{i=1}^{m}$ consisting of a sequence of m steps, a model generated output $\mathcal{O} = \{o\}_{i=1}^{n}$ consisting of n steps, and a sequence of s sentences as context with question $\mathcal{C} = \{c\}_{i=1}^{s}$. An answer or outcome annotation $y \in \mathbb{R}$ is a score indicating a measure of correctness of the model's output. Most commonly, $y = \mathbb{I}(o_n = r_m)$; i.e., the output's answer matches with the reference reasoning answer. In contrast, a process annotation $\mathcal{Y} = \{y \mid y \in \mathbb{R}\}_{i=1}^{n}$ is a sequence of scalar scores assigned to the corresponding steps $o_i \in \mathcal{O}$.

We propose **S**ingle-**P**ass **A**nnotation with **R**eference-Guided **E**valuation (**SPARE**: $(\mathcal{S}, \mathcal{C}, \mathcal{R}, \mathcal{O}) \rightarrow \mathcal{E}$) as a unified, single-pass framework that generates a step-by-step evaluation $\mathcal{E} = \{\varepsilon\}_{i=1}^{n}$ of a model output $\mathcal{O}$ with respect to a context $\mathcal{C}$, a reference reasoning $\mathcal{R}$, and a system prompt $\mathcal{S}$ that defines evaluation heuristics and guidelines. Each step o_i is evaluated in a structured format $\varepsilon = (e, c^+, o^+, r^+, \epsilon, y_i)$, where e provides an explicit explanation of the evaluation, ϵ is an optional list of error categories, and $y_i \in \{-1, +1\}$ is the assigned evaluation label. In addition to providing justification for the evaluation, the explanation e aligns the step o_i with potentially multiple reference steps ($o_i \mapsto c^+ \cup o^+ \cup r^+$), where $c^+ \subset \mathcal{C}$ represents a subset of relevant context sentences, $o^+ \subset \mathcal{O} \backslash \{o_i\}$ includes other related output steps, and $r^+ \subset \mathcal{R}$ consists of selected reference reasoning steps used for evaluation. The evaluation scheme enables efficient process annotation with an additive token complexity of $O(s + m + n)$, and allows for multiple alignment possibilities, particularly in cases where $m \neq n$:

1. *One-to-one* – Most simple alignment where one output step aligns directly and completely with *at most* one step, making it sufficient for evaluation. The alignment can take one of the forms: (i) a single reference reasoning step ($o_i \mapsto r_j$),

(ii) a single context sentence ($o_i \mapsto c_k$), (iii) follows directly from or complements another output step ($o_i \mapsto o_l$), or (iv) no alignment at all ($o_i \mapsto \varnothing$). In this case, $|c^+ \cup o^+ \cup r^+| \leq 1$.

2. *One-to-many* – An output step requires alignment with *at least* two steps for its evaluation. Such an alignment is *necessarily* required:

   i) When the model output step o_i is a composite step, either omitting minor intermediary steps or merging multiple steps into one. Its correctness must be evaluated against multiple reference steps r_j and c_k. In this case, $o^+ = \phi$ and $|c^+ \cup r^+| > 1$. This is likely when $n < m$ or $n < (m+s)$.

   ii) When the model output step o_i is a simple step while the reference steps are composite, its correctness must be evaluated in conjunction with at least one other output step o_l and at least one reference step r_j or context sentence c_k. In this case, $|o^+| \geq 1$ and $|c^+ \cup r^+| \geq 1$. This is likely when $n > m$ or $n > (m+s)$.

In summary, our **SPARE** framework defines step correctness based on its alignment with one or more reference reasoning steps, other output steps or context sentences, with the reference reasoning representing a valid path to the final answer. The multi-step alignment, combined with explicit step evaluation and explanations, accommodates surface form variations such as different topical approaches (Figure 1) and expression formats. This ensures that the steps are properly contextualized within the broader reasoning structure, allowing for their more accurate evaluation.

We implement SPARE using LLM-based evaluation, where off-the-shelf LLMs are prompted with k-shot exemplars. A small, manually annotated set of diverse examples is used to capture SPARE's core evaluation principles. To generate automatic annotations for model-produced reasoning traces, we use a model—typically from the previous fine-tuning stage in an iterative setup—to generate multiple solutions per problem via temperature-based sampling. Each solution is then decomposed into individual steps, typically delineated by newline characters. Our **SPARE** framework subsequently evaluates the correctness of each step, enabling automatic process supervision.

3.2 Training Approach

3.2.1 SPARE–based Finetuning (SPARE–ORPO)

We propose SPARE-based fine-tuning of a model to enhance its reasoning capabilities. The step-by-step process annotations $\mathcal{Y} = \{y_i\}_{i=1}^n$, derived using the SPARE framework, can be effectively integrated with both online and offline Reinforcement Learning (RL). For ease of implementation, training stability, and resource efficiency, we employ Odds Ratio Preference Optimization (ORPO) (Hong et al., 2024) for preference training over *chosen* and *rejected* pairs ($\mathcal{O}_w$, $\mathcal{O}_l$).

In SPARE-ORPO, we compute a mean aggregation $\bar{y} = \frac{1}{n}\sum y_i$ of the reasoning step annotations to quantify reasoning trace correctness and combine this with the final answer correctness y. The tuple $(y, \bar{y})$ is used as the effective score for preference pair identification, where $y_w = 1$, $y_l = -1$, and $\bar{y}_w > \bar{y}_l$. Thus SPARE-ORPO employs a more comprehensive set of preference pairs, where the chosen solution demonstrates superiority over the rejected solution with respect to both reasoning quality and answer accuracy. In contrast, `Outcome`-ORPO is trained using preference pairs ($\mathcal{O}_w$, $\mathcal{O}_l$) that are determined solely by final answer accuracy, i.e., $y_w = 1$ and $y_l = -1$.

3.2.2 SPARE–based Process Reward Model (SPARE–PRM)

We utilize the step-level evaluations y_i obtained through SPARE as direct reward signals to train process reward models. The SPARE-PRM is trained in a stepwise classification setting, using the following cross-entropy loss:

$$\mathcal{L}_{PRM} = -\sum_{i=1}^{n}\Big(y_i \ \log \ \sigma(r_\theta(\mathcal{C}, o_{1:i})) + (1-y_i) \ \log \left(1-\sigma(r_\theta(\mathcal{C}, o_{1:i}))\right)\Big) \quad (1)$$

where $o_{1:i}$ is the sub-sequence of output $\mathcal{O}$ till the ith step. Unlike ORMs which predict a single solution score for $\mathcal{O}$, PRMs generate a probability sequence $\mathcal{P} = \{p_i\}_{i=1}^n$ for each step $o_i \in \mathcal{O}$. These step-wise probabilities are aggregated into a final correctness score using functions such as min, prod (Lightman et al., 2024; Wang et al., 2024a), last, and max (Wang et al., 2024c).

Ranking and Aggregation. We use reward models to score multiple generations during inference.

We then either do rank-and-select (e.g. Best-of-N sampling) or weighted aggregation as:

$$\hat{a} = \underset{a}{\operatorname{argmax}} \sum_{i=1}^{N} \mathbb{I}(a_i = a) \cdot f(\mathcal{C}, \mathcal{O}_i) \quad (2)$$

where each final answer a_i from $\mathcal{O}_i$ is grouped and weighted by a function $f(\cdot)$ over N solutions. In self-consistency (Wang et al., 2023), the weighting function f accounts for the presence of an answer a_i, meaning all occurrences of an answer are given equal weight., i.e., $f(\mathcal{C}, \mathcal{O}_i) = 1$. Alternatively, the probability scores from Reward Models are used as weights for aggregation, effectively enhancing reasoning accuracy by prioritizing solutions that exhibit both correct final answers and well-structured reasoning steps.

4 Experiment Results

4.1 Experimental Set-up

Datasets. We conduct extensive experiments over a suite of reasoning datasets[2]:

- **Mathematical Reasoning**. We use two mathematical datasets, GSM8K (Cobbe et al., 2021), which is a collection of grade school math word problems, and MATH (Hendrycks et al., 2021), which contains high school competition-level math problems across seven diverse topics. Following standard practice in the verification setting, we use the MATH-500 subset (Lightman et al., 2024) for test-time evaluation involving multiple generations.
- **Question-Answering**. We use MuSiQue-Ans dataset (Trivedi et al., 2022), a challenging multi-hop question-answering dataset constructed by composing six diverse reasoning graphs of sub-questions from five different sources.
- **Spatial Reasoning**. We use the small `SpaRP` (Rizvi et al., 2024), i.e., `SpaRP-S` dataset, which comprises four textual spatial reasoning sub-datasets covering various spatial characterizations. `SpaRP` requires spatial relation composition to deduce relations between two objects when their direct relation is not provided in the context.

[2]Since GSM8K, MATH, and MuSiQue-Ans do not provide dev-sets, we partition their train-sets to create train and dev splits. For GSM8K and MATH, we use a 90:10 split, while for MuSiQue-Ans, we adopt an 80:20 split.

Models. We conduct our primary experiments across all four datasets for both model fine-tuning and reward model training using the LLaMA-3 8B Instruct model. Due to computational constraints, we limit additional experiments to selected datasets. To assess generalization across model families, we report results on the most challenging MATH dataset using Qwen 2.5 models. For comparison with MCTS-based approaches, we evaluate on GSM8K and MATH using Mistral-7B models.

Metrics. We report[3] the accuracy for the GSM8K and MATH datasets, accuracy and F1 metric for the MuSiQue-Ans dataset, and the accuracy and macro-F1 for the `SpaRP` dataset.

Implementation Details and Baselines. We begin with a single-epoch supervised fine-tuning (SFT) on the training split. Next, for each problem in the training and dev-set, we generate $N = 20$ solutions from the fine-tuned model using a temperature of 1. These solutions are then annotated using final answers for outcome supervision and the `SPARE` framework for process supervision.

We employ off-the-shelf pretrained models[4] for reference-guided step annotations using our `SPARE` framework. To account for problem diversity, we manually construct structured step-by-step evaluation exemplars per dataset—ranging from 6 for `SpaRP` to 56 for MATH—balanced for final answer correctness and covering all topics (MATH), sub-datasets (`SpaRP`), or reasoning graphs (MuSiQue-Ans). Each dataset is evaluated in a 5-shot setting using dataset-specific evaluation guidelines as system prompts. See Appendix A for an example.

In the *finetuning* scenario, we evaluate our `SPARE`-ORPO iteration trained on preference pairs formed using both outcome supervision and the mean reasoning scores of the step-by-step annotations (Section 3.2). We benchmark `SPARE`-ORPO against `Outcome`-ORPO and second iteration of Supervised Fine-Tuning (SFT) with an equivalent number of training instances. The training hyperparameter details are provided in Appendix C.

In the *verification* scenario, we use process annotations from the `SPARE` framework to train

[3]Exact Match for GSM8K. *competition_math* metric from the evaluate library for MATH. Accuracy and F1 for MuSiQue-Ans from their github repository. Accuracy and F1 for `SpaRP` from the scikit-learn library.

[4]Our framework could be further enhanced by specialized evaluator models such as Prometheus 2 (Kim et al., 2024), or by leveraging larger, more capable language models.

Training Method	Mathematical Reasoning		Question Reasoning	Spatial Reasoning
	GSM8K Acc. (↑)	MATH Acc. (↑)	♫ MuSiQue-Ans Acc. (↑) / F1 (↑)	SpaRP-S Acc. (↑) / F1 (↑)
SFT 1st Iteration	**70.43**	21.22	23.58 / 32.53	23.23 / 35.00
+ 2nd Iteration	**70.43**	22.08	26.31 / 35.12	39.90 / 47.13
+ Out.-ORPO	69.07	23.16	38.15 / 49.85	39.23 / 49.75
+ SPARE-ORPO	69.75	**23.42**	**38.89 / 50.53**	**40.13 / 50.96**

Table 1: Performance evaluations of Llama-3 8B Instruct model with *greedy decoding* under different training methods. Best values in **bold**, second best in underline.

Aggregation / Ranking	Mathematical Reasoning		Question Reasoning	Spatial Reasoning
	GSM8K Acc. (↑)	MATH-500 Acc. (↑)	♫ MuSiQue-Ans Acc. (↑) / F1 (↑)	SpaRP-S Acc. (↑) / F1 (↑)
SC	74.91	23.40	19.74 / 25.18	25.43 / 34.37
ORM	79.76	20.20	33.43 / 45.42	41.73 / 49.79
ORM + SC	79.83	23.80	34.80 / 44.45	41.70 / 49.78
SPARE-PRM	79.98	20.90	**34.84 / 45.52**	**43.73 / 50.08**
SPARE-PRM + SC	**80.29**	**24.10**	32.11 / 40.43	39.58 / 46.92

Table 2: Performance evaluations of aggregators and RM verifiers on $N = 20$ sample output generations from Llama-3 8B SFT 1st iteration. SC denotes Self-Consistency. RM only entries indicate Best-of-N (BoN) sampling based results. Best values in **bold**, second best in underline. Mean of metrics reported on 3 groups of sampling results.

Generator	Verifier	Agg./Rank.	Math–500
Qwen-2.5-3B	–	SC	31.4
	Llama-3-8B	SPARE-PRM	32.2
		SPARE-PRM + SC	34.2
	Qwen-2.5-3B	ORM	33.4
		ORM + SC	33.8
		SPARE-PRM	32.2
		SPARE-PRM + SC	**34.6**
Qwen-2.5-32B	–	SC	64.6
	Llama-3-8B	SPARE-PRM	55.0
		SPARE-PRM + SC	65.4
	Qwen-2.5-3B	ORM	57.6
		ORM + SC	65.6
		SPARE-PRM	58.6
		SPARE-PRM + SC	**66.0**

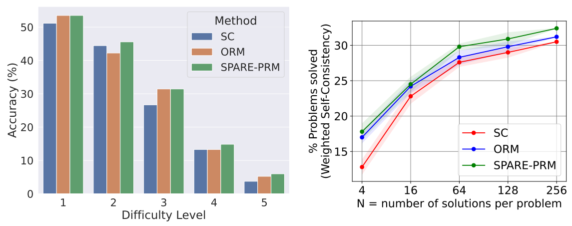
Table 3: Accuracy of SPARE-PRMs across model families and sizes on MATH-500 test-set with $N = 20$ generations. Mean accuracy reported on 3 groups of sampling results.

SPARE-PRMs as a standard language modeling task, predicting special tokens for correct and incorrect steps as a classification objective (Section 3.2.2) at special end-of-step ($\mathbb{EOS}$) tokens. We benchmark these models against outcome reward models (ORMs) and majority-voted self-consistency (Wang et al., 2023). To ensure balanced training, we randomly sample equal numbers of positive and negative examples. Evaluation metrics for both ORMs and PRMs are reported under two settings: (a) weighted aggregation (i.e., RM-weighted self-consistency) and (b) no aggregation, i.e., Best-of-N (BoN) sampling considering only the highest-scoring solution. Further training details and hyperparameters are provided in Appendix D.

4.2 Results and Discussion

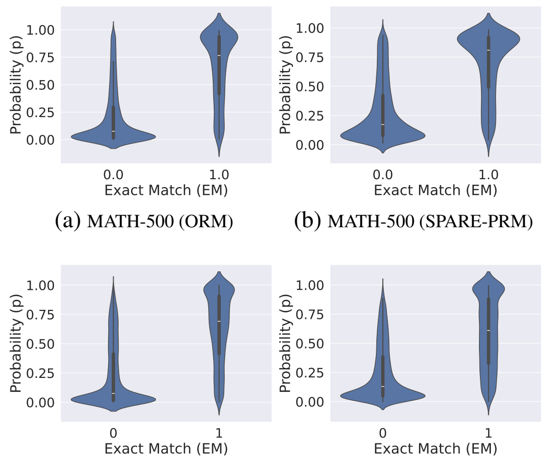
SPARE Helps in Fine tuning. We report the performance of fine tuning LLM followed by greedy decoding in Table 1. SPARE-ORPO achieves the best performance across three of the four datasets, with a maximum relative improvement of 2.43% in F1 score on the SpaRP-S dataset compared to the next-best Outcome-ORPO models. On the challenging MATH dataset, it attains a relative improvement of 1.12%, reaching an accuracy of 23.42%. The improvements of the SPARE models over Outcome baselines are statistically significant ($p < 0.05$) under one-tailed paired t-test. This underscores the effectiveness of SPARE in reasoning step annotation and identifying *superior* preference pairs than outcome-only preference pairs. Both these ORPO models significantly outperform the SFT models trained on the ground-truth reasoning traces, except for the GSM8K dataset. For GSM8K, the performance does not increase on ORPO-finetuned models and shows stagnation between the 1st and 2nd SFT iterations. We attribute this to the saturation of performance on this dataset, particularly as we used the same hyperparameters across all datasets (see details in Appendix C.1). For example, if we use different learning rates on GSM8K, we can observe a higher performance on SPARE. We did not include the result in Table 1 in order to ensure a fair and robust comparison (see details in Appendix C.2). Again, SPARE consistently outperforms other models in all the other setups in Table 1 and 2.

SPARE Improves Reward Model Training and Scales with Inference-Time Compute. Table 2 shows that SPARE-PRM performs the best across all four datasets, outperforming both ORMs and the majority-voted Self-Consistency (SC). The improvements of the best SPARE-PRM ranked or aggregated strategy over the best Outcome baselines are statistically significant ($p < 0.05$) under one-tailed paired t-test, with a maximum *relative* improvement of 4.79% accuracy on SpaRP-S. On the challenging MATH-500 dataset, it attains a relative



(a) Level-wise Performance (b) Inference-time Scaling

Figure 2: Performance of Llama-3 8B Instruct model under different strategies–SC, ORM-weighted and PRM-weighted consistency–over difficulty levels and with candidate scaling on MATH-500 test set.



(a) MATH-500 (ORM) (b) MATH-500 (SPARE-PRM)

(c) SpaRP-S (ORM) (d) SpaRP-S (SPARE-PRM)

Figure 3: Distribution-plots of ORM and SPARE-PRM probabilities for correct and incorrect answers of Math-500 and SpaRP-S datasets.

improvement of 1.26%, reaching an accuracy of 24.10%. Notably, this improvement is consistent across difficulty levels and especially significant for more difficult problems (Figure 2a). We note, however, that all the reported performance corresponds to $N = 20$ generations. When scaled to the commonly reported setting of larger number of generations, e.g., $N = 256$ (Wang et al., 2024a), performance on MATH-500 improves to 35.4 with a Mistral-7B model (Table 4). Comparable results are also achieved with the Llama-3-8B model, as illustrated in Figure 2b, where SPARE-PRM consistently outperforms baselines as the number of generated solutions increases from 4 to 256.

Finally, even with $N = 20$ generations, we show that the Llama-3-8B PRM, and the SPARE framework more broadly, generalizes across model families. As shown in Table 3, it achieves 34.2% accuracy with a Qwen-2.5-3B generator, outperforming the corresponding ORM and only surpassed by the PRM from the same model family. For Qwen-2.5-32B generations, although the Llama-3-8B PRM is outperformed by the Qwen-2.5-3B ORM, the effectiveness of SPARE-PRM remains consistent when trained on the same Qwen-2.5-3B base model, achieving a peak accuracy of 66%.

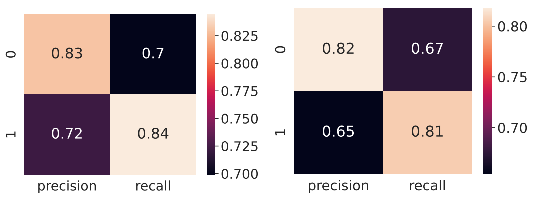
SPARE Adapts to Diverse Reasoning Traces. SPARE-PRM with Best-of-N (BoN) sampling achieves the best performance on datasets with limited reasoning diversity—♫ MuSiQue-Ans and SpaRP. In contrast, aggregating multiple outputs via self-consistency (SC) reduces the performance of SPARE-PRM on these datasets, whereas other reward models perform comparably or better. Combined with the fact that SPARE-PRM is trained on multiple outputs annotated against a reference reasoning trace, this suggests that SPARE-PRM may struggle to assess outputs with diverse reasoning forms. However, in challenging datasets with more diverse reasoning structures such as GSM8K and MATH-500, SPARE-PRM combined with aggregation (SC) outperforms both Best-of-N (BoN) sampling and other reward models.

Distribution plots of ORM and SPARE-PRM probabilities for correct and incorrect answers (Figure 3) on two representative datasets, Math-500 and SpaRP-S, further support this observation. On SpaRP-S, SPARE-PRM assigns lower mean probabilities to correct answers and higher mean probabilities to incorrect ones compared to ORMs. For correct responses, its probability distribution is more dispersed, whereas ORMs tend to skew toward higher probability values. These factors diminish the effectiveness of PRMs under aggregation (SC), making them better suited for ranked selection (BoN) strategies. However, on Math-500 dataset, although SPARE-PRM also assigns higher probabilities to incorrect answers, this is offset by a distribution skewed toward higher probability values for correct answers, resulting in a higher overall mean probability for correct responses.

SPARE is Computationally Efficient and Competitive with MCTS Methods. Monte Carlo Tree Search (MCTS)-based automatic annotations (Wang et al., 2024a,c; Zhang et al., 2024b) have recently proven effective for process supervision. To facilitate a comparison with MCTS-based methods, we report results on the GSM8K and Math-500 datasets using Math-Shepherd (Wang et al., 2024a) and SPARE-PRM in Table 4. For a

Aggregation	GSM8K Acc. (↑)	MATH-500 Acc. (↑)
Self-Consistency (SC)	83.1	33.6
+ ORM	86.7	35.1
+ Math-Shepherd	87.7	**35.4**
+ SPARE-PRM	**87.8**	**35.4**

Table 4: Performance comparison on mathematical datasets including Math-Shepherd (Wang et al., 2024a), a MCTS-based approach, over $N = 256$ generations. Mean accuracy reported on 3 groups of sampling results.



(a) Single-step (b) Multi-step

Figure 4: Precision and recall of SPARE relative to human annotations for ground-truth single-step and multi-step alignments.

direct comparison, we adopt the same experimental settings used in Math-Shepherd, employing the Mistral-7B: MetaMATH generator with 256 sampled outputs. Verifier performance is also reported for both Math-Shepherd and SPARE-PRM using the Mistral-7B model. With weighted aggregation, SPARE-PRM slightly outperforms Math-Shepherd on GSM8K and achieves comparable results on Math-500. Both methods outperform baselines such as self-consistency and ORM.

However, under identical compute setting (Appendix B), the MCTS-based annotation framework of Math-Shepherd is 2.6 times slower than SPARE, with SPARE completing the annotation process in just 38% of the time. This efficiency stems from SPARE's single-pass annotation process, in contrast to MCTS-based methods that require extensive search and repeated model inference at each step. Minimizing inference for annotation will be particularly advantageous when using commercially-hosted or proprietary models, where completion tokens incur higher costs than input tokens, making SPARE's single-pass, completion-light evaluation even more cost-efficient. As a result, SPARE is particularly well-suited for large-scale deployments and scenarios with constrained resources.

Comparison of SPARE with Manual Step Annotations. We evaluated the accuracy of SPARE annotations on 56 manually annotated examples from the MATH dataset, balanced for final answer correctness and covering all seven topics. We used 5-shot setting with the LLaMA-3-8B model, excluding the target example from in-context exemplars to avoid data leakage. For each instance, exemplars were randomly sampled, and annotations were repeated ten times. Comparing SPARE annotations to manual labels, we observed 76.9% accuracy for steps with single-step alignment and 73% for those requiring multi-step alignment. Despite a slight decrease for multi-step cases, these results demonstrate SPARE's strong capability in capturing complex reasoning and validate its robustness across topics and multiple runs.

A class-wise precision recall is also presented in Figure 4. We observe the recall of correct steps and precision of incorrect steps to be high (>80%) in both single-step and multi-step alignment scenarios. In comparison, the observed drop in precision for correct steps and recall for incorrect steps across both single- and multi-step scenarios indicates a more permissive and lenient annotation behavior by the LLM.

5 Conclusions

Achieving high-quality and efficient automatic process supervision is crucial for enhancing the complex multi-step reasoning abilities of Large Language Models (LLMs). To this end, we propose **S**ingle-**P**ass **A**nnotation with **R**eference-Guided **E**valuation (**SPARE**), a structured framework that enables per-step annotation in a single pass by evaluating each solution step against one or multiple reference steps with explicit reasoning. Our experimental results demonstrate that fine-tuning a base model and training a reward model with SPARE lead to improved reasoning performance under both greedy decoding and ranking/aggregation of multiple solutions. Furthermore, we observe consistent improvements across four datasets spanning mathematical reasoning, multi-hop compositional question answering, and spatial reasoning. SPARE achieves competitive performance with greater efficiency compared to tree search–based annotation methods, and shows strong alignment with human annotations. These findings highlight the potential of reference-guided automatic process supervision as a promising approach for enhancing LLM reasoning capabilities.

Limitations

SPARE and its associated models depend on the availability of reference reasoning chains to perform reference-guided step evaluations. However, we note that the reference reasoning traces used by SPARE are the same as those commonly used in Supervised Finetuning (SFT)—a foundational step in most finetuning methodologies including other automatic annotation methods. Thus, our approach does not introduce an additional annotation burden beyond what is typically required for training strong instruction-following models.

We also note that SPARE, like other LLM-based automatic processes, is susceptible to some degree of noise. Nevertheless, we find that the structured, reference-evaluated step annotations it provides are effective for training both Process Reward Models (PRMs) and base models, leading to improved reasoning performance. Despite its limitations, SPARE remains a practical, efficient and impactful approach for process supervision and model enhancement.

References

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen Marcus McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. 2024. Llemma: An open language model for mathematics. In *The Twelfth International Conference on Learning Representations*.

Zhen Bi, Ningyu Zhang, Yinuo Jiang, Shumin Deng, Guozhou Zheng, and Huajun Chen. 2024. When do program-of-thought works for reasoning? In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D 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 Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Zhaorun Chen, Zhuokai Zhao, Zhihong Zhu, Ruiqi Zhang, Xiang Li, Bhiksha Raj, and Huaxiu Yao. 2024. AutoPRM: Automating procedural supervision for multi-step reasoning via controllable question decomposition. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1346–1362, Mexico City, Mexico. Association for Computational Linguistics.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, and 48 others. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

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. *Preprint*, arXiv:2110.14168.

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. 2023. Complexity-based prompting for multi-step reasoning. In *The Eleventh International Conference on Learning Representations*.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8154–8173, Singapore. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1.

Jiwoo Hong, Noah Lee, and James Thorne. 2024. ORPO: Monolithic preference optimization without reference model. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11170–11189, Miami, Florida, USA. Association for Computational Linguistics.

Muhammad Khalifa, Rishabh Agarwal, Lajanugen Logeswaran, Jaekyeom Kim, Hao Peng, Moontae Lee, Honglak Lee, and Lu Wang. 2025. Process reward models that think. *Preprint*, arXiv:2504.16828.

Muhammad Khalifa, Lajanugen Logeswaran, Moontae Lee, Honglak Lee, and Lu Wang. 2023. GRACE: Discriminator-guided chain-of-thought reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 15299–15328, Singapore. Association for Computational Linguistics.

Seungone Kim, Juyoung Suk, Shayne Longpre, Bill Yuchen Lin, Jamin Shin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. 2024. Prometheus 2: An open source language model specialized in evaluating other language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 4334–4353, Miami, Florida, USA. Association for Computational Linguistics.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. 2023. 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.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Jie Liu, Gongye Liu, Jiajun Liang, Ziyang Yuan, Xiaokun Liu, Mingwu Zheng, Xiele Wu, Qiulin Wang, Wenyu Qin, Menghan Xia, Xintao Wang, Xiaohong Liu, Fei Yang, Pengfei Wan, Di Zhang, Kun Gai, Yujiu Yang, and Wanli Ouyang. 2025. Improving video generation with human feedback. *Preprint*, arXiv:2501.13918.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, Yansong Tang, and Dongmei Zhang. 2025. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *Preprint*, arXiv:2308.09583.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. 2024. Improve mathematical reasoning in language models by automated process supervision. *Preprint*, arXiv:2406.06592.

Sarah Pan, Vladislav Lialin, Sherin Muckatira, and Anna Rumshisky. 2023. Let's reinforce step by step. In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*.

Md Imbesat Rizvi, Xiaodan Zhu, and Iryna Gurevych. 2024. SpaRC and SpaRP: Spatial reasoning characterization and path generation for understanding spatial reasoning capability of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4750–4767, Bangkok, Thailand. Association for Computational Linguistics.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, Agnieszka Kluska, Aitor Lewkowycz, Akshat Agarwal, Alethea Power, Alex Ray, Alex Warstadt, Alexander W. Kocurek, Ali Safaya, Ali Tazarv, and 431 others. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*. Featured Certification.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. ♫ MuSiQue: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

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. *Preprint*, arXiv:2211.14275.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024a. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439, Bangkok, Thailand. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Zhilin Wang, Alexander Bukharin, Olivier Delalleau, Daniel Egert, Gerald Shen, Jiaqi Zeng, Oleksii Kuchaiev, and Yi Dong. 2024b. Helpsteer2-preference: Complementing ratings with preferences. *Preprint*, arXiv:2410.01257.

Zihan Wang, Yunxuan Li, Yuexin Wu, Liangchen Luo, Le Hou, Hongkun Yu, and Jingbo Shang. 2024c. Multi-step problem solving through a verifier: An empirical analysis on model-induced process supervision. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 7309–7319, Miami, Florida, USA. 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. 2022a. Emergent abilities of large language models. *Transactions on Machine Learning Research*. Survey Certification.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022b. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik R Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Xi Ye and Greg Durrett. 2022. The unreliability of explanations in few-shot prompting for textual reasoning. In *Advances in Neural Information Processing Systems*.

Fei Yu, Anningzhe Gao, and Benyou Wang. 2024a. OVM, outcome-supervised value models for planning in mathematical reasoning. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 858–875, Mexico City, Mexico. Association for Computational Linguistics.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng YU, Zhengying Liu, Yu Zhang, James Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2024b. Metamath: Bootstrap your own mathematical questions for large language models. In *The Twelfth International Conference on Learning Representations*.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024a. ReST-MCTS*: LLM self-training via process reward guided tree search. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024b. ReST-MCTS* : LLM self-training via process reward guided tree search. In *Advances in Neural Information Processing Systems*, volume 37, pages 64735–64772. Curran Associates, Inc.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. 2025. Generative verifiers: Reward modeling as next-token prediction. In *The Thirteenth International Conference on Learning Representations*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*.

A LLM-based Grading Example

An example system prompt with evaluation guidelines, user request and process annotation as graded output for MATH dataset under our SPARE framework is provided in Table 5 and Table 11. Similarly, other datasets have their own domain and subject specific evaluation heuristics included in the system prompt.

B Computing Infrastructure

The experiments were conducted using 8 NVIDIA A100 GPUs, each with 40GB of memory.

C Details of Finetuning

C.1 Implementation detail common across all datasets.

We used Huggingface's TRL library and QLoRA for parameter-efficient finetuning of all ORPO models across datasets (Table 1), using a fixed set of hyperparameters for consistency (Table 7). This setup performed well across all datasets except GSM8K, where we observed performance saturation after the first iteration and degradation in ORPO models. To address this, we conducted a learning rate search specifically for GSM8K, leading to consistent improvements as detailed in Section C.2.

C.2 GSM8K Hyperparameter Tuning.

Due to training saturation observed on GSM8K with the common hyperparameters (Table 7), we conducted a targeted search over learning rates $\{10^{-6}, 5\times10^{-6}, 10^{-5}, 5\times10^{-5}, 10^{-4}\}$. Using $lr = 10^{-5}$ for the first iteration and $lr = 5\times10^{-6}$ for the second on SPARE produced the best results on the validation/development dataset. The corresponding test set performance is reported in Table 9, where the SPARE-ORPO model achieved an accuracy of 72.25%. We did not include it in Table 1 in order to ensure a robust comparison by using the same hyperparameters across all datasets.

D Details of Reward Model (RM) Training

The number N of individual positive and negative samples (i.e. N pairs) for Reward Model training are presented in Table 8. Hence, for example, the total effective outcome or process supervision dataset used for RM training for mathemtical datasets was ≈40K. This is still significantly smaller than other work (Lightman et al., 2024; Wang et al., 2024a).

While prior work (Lightman et al., 2024; Wang et al., 2024a; Luo et al., 2024) have shown the min or prod aggregation to be the better performing aggregation strategies, other work (Wang et al.,

<table>
<tr><th>Role</th><th>Content</th></tr>
<tr><td>System</td><td>You are a teacher grading a student's assignment. You are given a QUESTION, its ground-truth correct REFERENCE ANSWER and a STUDENT'S ANSWER. You are asked to match the steps of STUDENT'S ANSWER with that in the REFERENCE ANSWER and in the context of the given QUESTION. You are required to score the steps of STUDENT'S ANSWER as below:

A step in the STUDENT'S ANSWER that matches or is factually consistent with one or more steps in the REFERENCE ANSWER and in the context of the sentences provided in the QUESTION is labelled as CORRECT. While matching steps, evaluate on the basis of:

(a) whether the topic, entities, variables and the intended result of the step are correct or not, and
(b) whether the expressions, equations, and / or numerical computations in a step are correct or not.

A step in the STUDENT'S ANSWER that doesn't match or is factually incorrect with respect to the provided REFERENCE ANSWER and the QUESTION is labelled as INCORRECT.

You need to evaluate ALL the steps of the STUDENT'S ANSWER. Provide your evaluation ONLY and ONLY in JSON format as a list of dictionaries whose keys and their intended purposes are:

"student_step": The current step number of the STUDENT'S ANSWER.

"reasoning": The reasoning expanding upon why or what part of the current 'student_step' of the STUDENT'S ANSWER, either DIRECTLY and ENTIRELY in itself or probably in combination with other steps in the STUDENT'S ANSWER, is correct or incorrect in reference to one or more REFERENCE ANSWER steps and the QUESTION sentences.

"question_sentences": A list of sentences in the QUESTION based on which the correctness or the incorrectness of the current 'student_step' is reasoned and arrived at. If the number of steps in the STUDENT'S ANSWER is less than that in the REFERENCE ANSWER or the topic and the intended goal of a 'student_step' doesn't match with any steps in the REFERENCE ANSWER, then it is useful to look for QUESTION sentences to assess the 'student_step'. More than one QUESTION sentences can be of use for this evaluation. Leave it as an empty list if the current 'student_step' DIRECTLY and ENTIRELY matches with one or more steps in the REFERENCE ANSWER.

"student_combining_steps": A list of previous 'student_step' that is necessary in evaluating the current 'student_step' because of restatements or transformations, or when combined with the current 'student_step' will be part or whole of one or more steps in the REFERENCE ANSWER. Leave it as an empty list if the current 'student_step' DIRECTLY and ENTIRELY matches with one or more steps in the REFERENCE ANSWER. If the number of steps in the STUDENT'S ANSWER is more than that in the REFERENCE ANSWER, then a single step in REFERENCE ANSWER may correspond to multiple steps in the STUDENT'S ANSWER and this list will be non-empty for some of the 'student_step'.

"matching_reference_steps": A list of steps in the REFERENCE ANSWER based on which the correctness or the incorrectness of the current 'student_step' is reasoned and arrived at. If the number of steps in the STUDENT'S ANSWER is less than that in the REFERENCE ANSWER, then multiple steps in the REFERENCE ANSWER may correspond to a single step in the STUDENT'S ANSWER. Leave this empty if there are no matching steps in the REFERENCE ANSWER.

"error_category": A list of type of errors that caused the current 'student_step' to be partially or fully incorrect. Report from "COMPREHENSION" when the student misunderstands and misapplies some concept, "NUMERIC" when the numeric values don't match, "CALCULATION" when the computations done are incorrect, "TRANSFORMATION" when the algebraic or trigonometric rearrangements or substitutions are incorrect, "PROPAGATION" when the incorrectness gets carried forward directly from the previous 'student_step', "RESTATEMENT" when the student makes mistake in restating from its own or question sentences, and "NO STEP MATCH". Leave it as an empty list if the current 'student_step' is completely correct.

"label": binary score of the current 'student_step' as either CORRECT or INCORRECT.</td></tr>
<tr><td>User</td><td>QUESTION:
[1] My three friends and I have dinner together every weekend.
[2] Each weekend, two of us cook and the other two clean up afterwards.
[3] How many different ways are there for us to choose who cooks and who cleans?

REFERENCE ANSWER:
[1] Let's think step by step.\nThere are four ways to choose the first cook and three ways to choose the second cook, but this counts every pair of cooks twice since order doesn't matter.
[2] Once the cooks are chosen, the two people left over are the cleaners.
[3] So, there are $(4 \cdot 3)/2 = \boxed{6}$ ways for us to choose who cooks and who cleans.\nHence, the answer is 6.

STUDENT'S ANSWER:
[1] Let's think step by step.\nWe have four choices for who cooks, and then we have three choices for who cleans.
[2] Since each order cooks, it doesn't matter who cleans; only the total cleaning pair counts.
[3] There are $\binom{4}{2} \cdot \binom{2}{1} = \boxed{12}$ ways in total.\nHence, the answer is 12.</td></tr>
</table>

Table 5: An example system prompt with evaluation guidelines and evaluation task as user prompt for MATH dataset under our SPARE framework. Other datasets have their own domain and subject specific evaluation details included in the system prompt.

Role	Content

Assistant	```json
	[
	{
	“student_step”: 1,
	“reasoning”: The step [1] of the STUDENT’S ANSWER mentions that there are 3 choices for who cleans. Sentence [2] of QUESTION and step [2] of REFERENCE ANSWER mention that there two people who can clean. Hence this is a mismatch and the student’s assertion is wrong.,
	“question_sentences”: [2],
	“student_combining_steps”: [],
	“matching_reference_steps”: [2],
	“error_category”: [“COMPREHENSION”],
	“label”: [“INCORRECT”],
	}
	{
	“student_step”: 2,
	“reasoning”: The step [2] of the STUDENT’S ANSWER is a generic restatement of the fact that the count of cleaners follows automatically from the count of cooks.,
	“question_sentences”: [],
	“student_combining_steps”: [],
	“matching_reference_steps”: [],
	“error_category”: [],
	“label”: [“CORRECT”],
	}
	{
	“student_step”: 3,
	“reasoning”: The step calculates the total ways or count to be 12 which is twice the ground truth ways of 6 as reported in [3] of the REFERENCE ANSWER. This could be because of the overuse of the factor 2 in doing the calculations due to the misunderstandings in the previous steps.,
	“question_sentences”: [],
	“student_combining_steps”: [],
	“matching_reference_steps”: [3],
	“error_category”: [“COMPREHENSION”, “CALCULATION”],
	“label”: [“INCORRECT”],
	}
	]
	``

Table 6: A graded example for MATH dataset under our SPARE framework.

Parameter Name	**Value**
QLoRA:	
α	16
Dropout	0.1
r	64
bias	None
task_type	CAUSAL_LM
Training Arguments:	
Effective Batch Size	32
lr	$1.0e-4$
weight decay	0.001
max_grad_norm	0.3
warm up ratio	0.03
lr_scheduler	cosine

Table 7: Values of the parameters and hyperparameters used while ORPO finetuning.

Dataset	N
GSM8K	20175
MATH	20,250
MuSiQue	5,000
SpaRP	8,000

Table 8: Training data sizes for Reward Models

2024c) have reported these to underperform ORM when the annotation process differs. For their annotation process, they reported last aggregation strategy, among others, to outperform ORM. We also found the min and prod aggregation strategies to sometimes underperform the ORMs, while last aggregation strategy performing the best. Hence, all the metrics are reported using the last aggregation strategy for PRMs.

E Pointwise vs Pairwise ORMs.

While *pairwise*-loss RM training is generally considered more effective than *pointwise*-loss RMs (Liu et al., 2025), empirical evidence remains divided. For instance, even accounting for differ-

Training Method	`GSM8K` Acc. (↑)
SFT 1st Iteration	68.61
+ 2nd Iteration	70.36
+ `Out.`-ORPO	<u>71.27</u>
+ SPARE-ORPO	**72.25**

Table 9: Performance evaluations of Llama-3 8B Instruct model with *greedy decoding* under different training methods on the GSM8K dataset with $lr = 10^{-5}$ for 1st SFT. followed by $lr = 5 \times 10^{-6}$ for other iterations. Best values in **bold**, second best in <u>underline</u>.

	Mathematical Reasoning		**Question Reasoning**	**Spatial Reasoning**
Aggregation / Ranking	`GSM8K` Acc. (↑)	`MATH-500` Acc. (↑)	♫ `MuSiQue-Ans` Acc. (↑) / F1 (↑)	`SpaRP-S` Acc. (↑) / F1 (↑)
pair.-ORM	78.54	16.30	30.45 / 42.87	31.65 / 40.78
pair.-ORM + SC	79.45	21.00	34.13 / 43.82	31.63 / 40.77
SPARE-ORM	79.15	19.00	34.67 / 45.11	32.7 / 41.95
SPARE-ORM + SC	79.22	20.60	35.29 / 45.52	32.63 / 41.90
point.-ORM	79.76	20.20	33.43 / 45.42	41.73 / 49.79
point.-ORM + SC	79.83	23.80	34.80 / 44.45	41.70 / 49.78

Table 10: Performance evaluations of different class of ORMs on $N = 20$ sample output generations from Llama-3 8B SFT 1st iteration. SC denotes Self-Consistency. RM only entries indicate Best-of-N (BoN) sampling based results. Mean of metrics reported on 3 groups of sampling results.

ences in annotation guidelines and human expectations, Liu et al.(2025) found pairwise RM training superior, whereas Wang et al.(2024b) reported better results with pointwise RM training. Our study adds to this debate with empirical evidence showing *pointwise*-ORM outperforming *pairwise*-ORM, significantly so on the MATH and SpaRP datasets (Table 10). Both models are trained on a balanced set of positive and negative instances based on final answer outcomes, with *pairwise*-ORM forming pairs in reference to given contexts. Furthermore, SPARE-ORM also underperforms *pointwise*-ORM, despite incorporating *superior* pairs selected based on both outcome and mean aggregated reasoning scores.

F Use of AI Assistance

We used generative AI tools exclusively for grammar and language refinement. All content was subsequently reviewed and revised by the author(s), assuming full responsibility for the final version of the manuscript and its submission.

Dataset	Error Categories
MATH	1. **Comprehension** – The student misunderstands and misapplies some concept. 2. **Numeric** – The numeric values don't match. 3. **Calculation** – The computations done are incorrect. 4. **Transformation** – The algebraic or trigonometric rearrangements or substitutions are incorrect. 5. **Propagation** – The incorrectness gets carried forward directly from previous student steps. 6. **Restatement** – The student makes mistake in restating from its own or question sentences. 7. **No Step Match** – The step doesn't match with any reference step nor it can be inferred from any context / question sentences.
GSM8K	1. **Comprehension** – The student misunderstands and misapplies some concept. 2. **Numeric** – The numeric values don't match. 3. **Calculation** – The computations done are incorrect. 4. **No Step Match** – The step doesn't match with any reference step nor it can be inferred from any context / question sentences.
♫ MuSiQue-Ans	1. **Document Name** – The document name in the step doesn't match or exists. 2. **Entity Name** – The entity name in the step doesn't match or exists. 3. **Numeric** – The numbers mentioned don't match. 4. **Intended Category** – The category differs from what is required e.g. Date required but Country discussed. 5. **Semantic Relation** – The relation between entities, to be compared semantically e.g. local language and native language in reference to a person and a place is semantically same. 6. **No Step Match** – The step doesn't match with any reference step nor it can be inferred from any context / question sentences.
SpaRP	1. **Entity Name** – The name of the entity in the step doesn't match or exist. 2. **Incorrect Relation** – The reported spatial relation in the step doesn't match or exist. 3. **No Step Match** – The step doesn't match with any reference step nor it can be inferred from any context / question sentences.

Table 11: Dataset specific error categories.