Advancing Process Verification for Large Language Models via Tree-Based Preference Learning

Anonymous ACL submission

Abstract

Large Language Models (LLMs) have demonstrated remarkable potential in handling complex reasoning tasks by generating step-by-step rationales. Some methods have proven effective in boosting accuracy by introducing extra verifiers to assess these paths. However, existing verifiers, typically trained on binary-labeled reasoning paths, fail to fully utilize the relative merits of intermediate steps, thereby limiting the effectiveness of the feedback provided. To overcome this limitation, we propose **Tree**-based **P**reference **L**earning **V**erifier (Tree-PLV), a novel approach that constructs reasoning trees via a best-first search algorithm and collects step-level paired data for preference training. Compared to traditional binary classification, step-level preferences more finely capture the nuances between reasoning steps, allowing for a more precise evaluation of the complete reasoning path. We empirically evaluate Tree-PLV across a range of arithmetic and commonsense reasoning tasks, where it significantly outperforms existing benchmarks. For instance, Tree-PLV achieved substantial performance gains over the Mistral-7B self-consistency baseline on GSM8K (67.55% $\rightarrow$ 82.79%), MATH (17.00% $\rightarrow$ 26.80%), CSQA (68.14% $\rightarrow$ 72.97%), and StrategyQA (82.86% $\rightarrow$ 83.25%). Additionally, our study explores the appropriate granularity for applying preference learning, revealing that step-level guidance provides feedback that better aligns with the evaluation of the reasoning process.

1 Introduction

Large Language Models (LLMs) have demonstrated the ability to decompose complex questions into step-by-step problem-solving processes (Brown et al., 2020; Achiam et al., 2023; Liu et al., 2023; Frieder et al., 2024), achieving strong reasoning performance across a variety of tasks.

To enhance the reliability of reasoning paths, the best-of-N decoding strategy (Nakano et al., 2021; Askell et al., 2021; Cobbe et al., 2021) is employed, where N candidate solutions are generated by the LLMs, and the most plausible one is selected based on specific rules (Golovneva et al., 2022; Prasad et al., 2023), such as coherence, logical consistency, and alignment with known facts. Recently, some studies have introduced an auxiliary model, termed a verifier (Cobbe et al., 2021), to assess the quality of candidate solutions. The training of verifiers can be divided into outcome supervision (Cobbe et al., 2021; Yu et al., 2023a; Hosseini et al., 2024) and process supervision (Li et al., 2022; Lightman et al., 2023; Wang et al., 2023) based on the granularity of the supervision signal (see Figure 1). Outcome supervision labels the entire path based on the final result, while process supervision evaluates the correctness of each individual step.

Whether employing outcome supervision or process supervision, verifiers are typically trained using binary classification (Uesato et al., 2022; Lightman et al., 2023; Wang et al., 2023), which may not align ideally with the goals of optimizing verifiers. In the best-of-N decoding, verifiers are expected to rank candidate paths accurately. However, binary labels, offering rudimentary correct or incorrect signals, fail to capture the relative merits of different paths. Such coarse supervisory signals are insufficient to provide the detailed feedback necessary for verifiers to discern which steps are more effective, thereby limiting the potential for further improvements. Moreover, the annotations used in training, often derived from answers, inherently contain some degree of noise. Even if the final answer is correct, the reasoning process may not be entirely accurate. Unfaithful reasoning and spurious shortcuts can also lead to the correct answer (Creswell and Shanahan, 2022; Lyu et al., 2023; Turpin et al., 2024). Consequently, training verifiers using binary classification is particularly vulnerable to noisy labels, which constrains the verifier's capacity to precisely validate the steps.

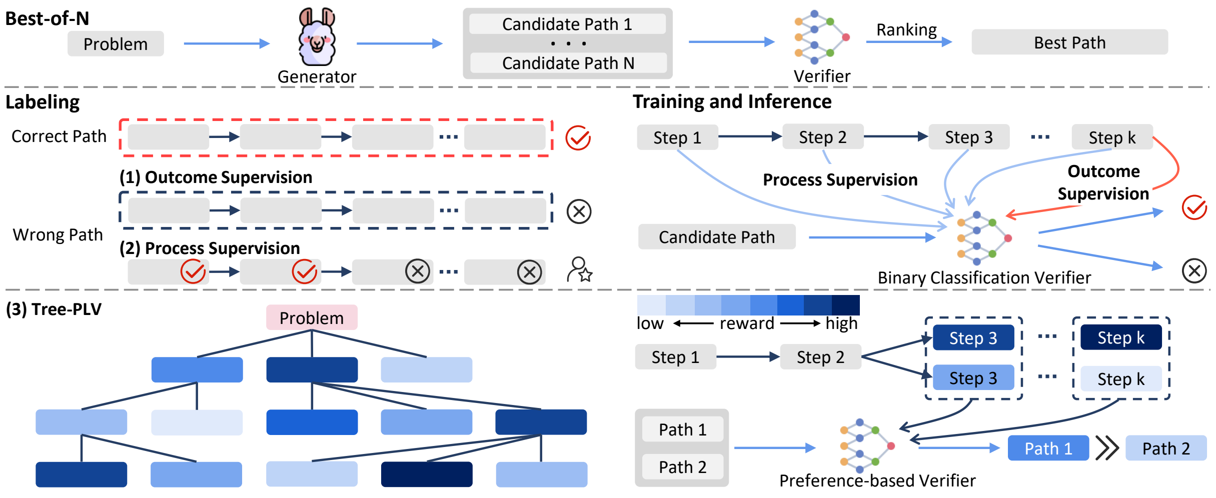

Figure 1: A comparison of different methods: Traditional verifiers rely on binary labels for outcome and process supervision, whereas Tree-PLV employs preferences instead of scalar values.

To tackle these challenges, we propose a shift from a binary to a preference-based verifier. Trained through preference learning, it ranks the relative merits of different reasoning paths, allowing for more nuanced partitioning than simply judging them as correct or incorrect. The advantages of adopting this step-level preference-based verifier for ranking the reasoning paths include:

- **Granular Validation at the Step Level:** Verifiers based on preference learning can capture subtle differences between steps, thereby providing more precise feedback.

- **Improving Verifier Robustness:** Focusing on ranking rather than binary judgments enhances the verifier's stability. As long as the relative ordering of steps is consistent, the training of the verifier remains robust against label noise.

- **Enhancing Model Explainability:** The detailed feedback provided by preference learning offers deeper insights into the reasoning process, moving beyond mere correctness on the final result.

Therefore, we introduce **Tree**-based **P**reference **L**earning **V**erifier (Tree-PLV), a novel method inspired by preference learning principles. Tree-PLV transcends traditional verifiers by modeling rewards based on comparisons between paths. Our method not only focuses on instance-level rewards derived from the outcomes but also emphasizes step-level optimization. This allows Tree-PLV to utilize intermediate steps to provide more finely grained feedback. Specifically, we employ a best-first search strategy during inference to construct a reasoning tree, with the initial problem statement as the root and each step as a node. Upon developing the tree, we construct our dataset by tracing paths from the root to each leaf node. At each level, we form pairs by conducting pairwise comparisons among child nodes, preferring those with higher rewards. This dataset serves to train our verifier using a ranking loss, greatly enhancing its ability to discern subtle nuances in reasoning sequences.

We conduct an empirical evaluation of Tree-PLV across diverse reasoning tasks, focusing on arithmetic reasoning with the GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021) datasets, and commonsense reasoning on the CSQA (Talmor et al., 2018) and StrategyQA (Geva et al., 2021) datasets. We benchmark Tree-PLV against existing verifiers, including self-consistency (Wang et al., 2022) as a strong baseline. Our results indicate substantial performance gains across all datasets. For instance, when compared to the Mistral-7B self-consistency baseline, our method showed the following increases in accuracy: GSM8K (67.55% $\rightarrow$ 82.79%), MATH (17.00% $\rightarrow$ 26.80%), CSQA (68.14% $\rightarrow$ 72.97%), and StrategyQA (82.86% $\rightarrow$ 83.25%). Notably, Tree-PLV, when trained with data from GSM8K, demonstrates robust generalization to the more challenging MATH dataset.

2 Tree-PLV

In this section, we introduce the Tree-PLV method, which leverages tree-based preference learning to advance verification for large language models in the context of stepwise reasoning processes. We begin by outlining the problem formulations (§ 2.1).

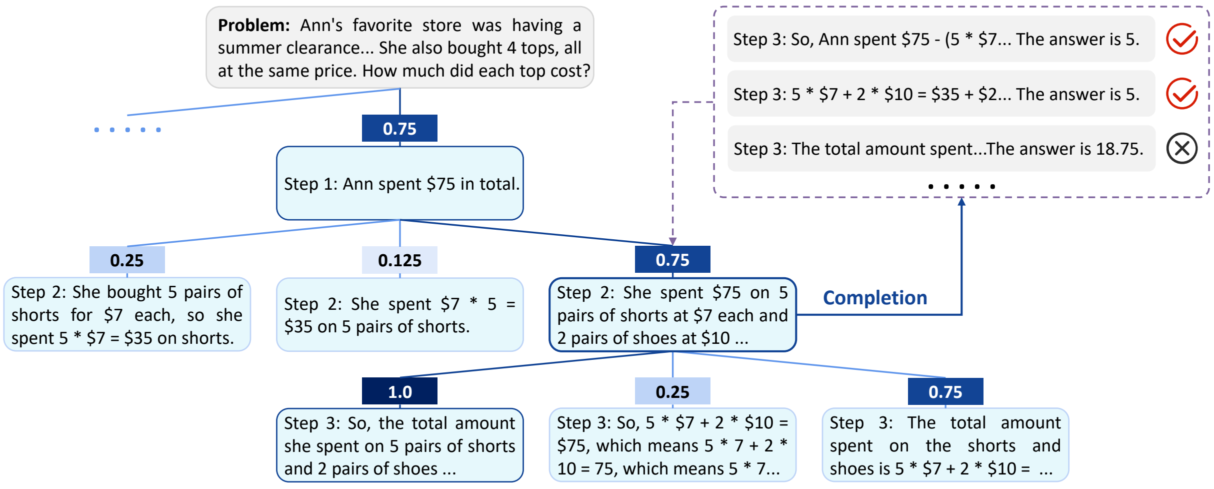
Problem: Ann's favorite store was having a summer clearance... She also bought 4 tops, all at the same price. How much did each top cost?
0.75
Step 1: Ann spent $75 in total.
0.25
Step 2: She bought 5 pairs of shorts for $7 each, so she spent 5 * $7 = $35 on shorts.
0.125
Step 2: She spent $7 * 5 = $35 on 5 pairs of shorts.
0.75
Step 2: She spent $75 on 5 pairs of shorts at $7 each and 2 pairs of shoes at $10 ...
Completion
Step 3: So, Ann spent $75 - (5 * $7... The answer is 5.
Step 3: 5 * $7 + 2 * $10 = $35 + $2... The answer is 5.
Step 3: The total amount spent...The answer is 18.75.
1.0
Step 3: So, the total amount she spent on 5 pairs of shorts and 2 pairs of shoes ...
0.25
Step 3: So, 5 * $7 + 2 * $10 = $75, which means 5 * 7 + 2 * 10 = 75, which means 5 * 7...
0.75
Step 3: The total amount spent on the shorts and shoes is 5 * $7 + 2 * $10 = ...

Figure 2: The construction process of the reasoning tree. Best-first search consistently selects the child node with highest reward for further expansion. To evaluate the quality of the i-th step, we sample N completions from it, denoted as $\mathcal{P}_i$. The reward is then calculated based on the proportion of these N paths that yield the correct answer.

Next, we detail how to construct a reasoning tree that represents reward preferences at each step (§ 2.2). Finally, we describe how we gather paired data for step-level preference learning and implement this into our verifier training (§ 2.3).

2.1 Problem Formulations

Following the best-of-N evaluation protocol proposed by Lightman et al. (2023), we generate N candidate solutions $\{y^{(1)}, y^{(2)}, \dots, y^{(N)}\}$ from the generator model for a given input x. Each solution y consists of a sequence of steps $\{y_1, y_2, \dots, y_n\}$. These solutions are then ranked by the verifier, and the highest-rated one is selected as the most plausible solution.

2.2 Reasoning Tree Construction

To provide precise step-level preference feedback, we implement a best-first tree search algorithm designed to generate paired data critical for preference learning. As Figure 2 depicts, our method constructs a reasoning tree step-by-step, where each node represents a reasoning step. Expansion starts from the root of the tree at each search iteration.

At step i of the tree expansion, we have a partial solution $y_{1:i-1}$ consisting of the previous $i-1$ reasoning steps. We use a reward function $\mathcal{R}(y_i|x, y_{1:i-1})$ to evaluate the quality of the next potential step y_i, given the input x and the current partial solution $y_{1:i-1}$. The tree search proceeds by expanding the most promising node at each iteration, i.e., the node whose child (the next potential step) has the highest reward according to $\mathcal{R}$. This guided exploration allows us to construct high-quality reasoning paths through the tree, providing paired data for preference learning between competing steps.

The traditional approaches regard the correctness of a step as its quality, relying on metrics like perplexity (PPL) or self-evaluation by LLMs to design the reward function $\mathcal{R}$. However, recent studies have shown that LLMs frequently struggle to effectively recognize errors (Huang et al., 2023; Hong et al., 2023; Ren et al., 2023b), which can degrade performance. To address this, we leverage the model's look-ahead capability to assess a step's quality by its potential to lead to the correct conclusion. Specifically, to evaluate a candidate step y_i, we use the same model to simulate N subsequent reasoning trajectories starting from y_i, denoted as N completions $\mathcal{P}_i = \{P_i^1, P_i^2, \dots, P_i^N\}$. The quality of the step y_i is quantified by the proportion of trajectories reaching the correct answer:

$$\mathcal{R}(y_i) = \frac{\sum_{j=1}^{N} \mathbb{1}[a[P_i^j] = g]}{N} \tag{1}$$

where $a[P_i^j]$ is the outcome of the j-th trajectory P_i^j and g represents the golden answer.

After determining the node with the highest reward value according to $\mathcal{R}(y_i|x, y_{1:i-1})$, we expand the tree by generating new child nodes. To achieve this, we sample k potential subsequent reasoning steps $y_{i+1}^j \sim \pi_\theta(y_{i+1}|x, y_{1:i})$ for $j = 1, \dots, k$, where π_θ is the language model used for reasoning. Each of these candidate steps $\{y_{i+1}^j\}_{j=1}^k$ becomes a new child node connected to the previously selected node. If the selected node represents the last step, indicating the end of the reasoning chain, we omit the expansion phase, and this iteration concludes. Guided by the reward function $\mathcal{R}$, this approach ensures a systematic exploration and expansion of reasoning paths in the search tree.

2.3 Step-Level Pairwise Training

A reasoning tree illustrates all potential reasoning paths, starting from the root and branching out to various leaf nodes. Our objective is to create a dataset $\mathcal{D}_{\text{pair}}$ consisting of pairs that express preferences of reasoning paths. We generate this dataset by tracing each unique path from the root to the leaves of the tree. Within this dataset, each entry consists of a triplet in the form $\{(x, y^+, y^-)\}$, where x denotes the initial problem statement, y^+ is the preferred reasoning sequence that leads to an accurate solution, and y^- is a less desirable reasoning chain that results in an incorrect answer.

To collect the paired data $\{(x, y^+, y^-)\}$, we conduct pairwise comparisons between sibling nodes at each decision point along the tree. Sibling nodes are the various possible next steps in the reasoning process branching from the same prior context $y_{1:i-1}$. If the reward difference between a preferable step y_i^+ and a less preferable step y_i^- meets the minimum margin α, we create a pair (x, y^+, y^-) that includes the initial problem x, the superior reasoning sequence y^+, and the inferior alternative y^-. Specifically, for the preferable step y_i^+, we sample a path that leads to a correct solution among its completions $\mathcal{P}_i^+$. This constitutes the complete reasoning path y^+. Conversely, the less preferable step y_i^- is assembled with one of the subsequent paths $\mathcal{P}_i^-$ that begins from it to form y^-, which ultimately leads to incorrect outcomes. Based on the pairwise instance data $\{(x, y^+, y^-)\}$, we gather two types of step preferences during our training. The first type is derived from the reward function $\mathcal{R}$ for the current step y_i, which assesses the choices at each decision point. The second type of preference implicitly decomposes the outcome-guided instance preference into step-by-step comparisons, examining each potential path leading to possible outcomes. This breakdown allows us to allocate the instance preference to both the decision points and the completion steps in the reasoning process, ensuring that steps on the correct path are preferred over those on the incorrect path.

The verifier is built upon a large language model with an additional randomly initialized linear layer that outputs a scalar value. During evaluation, we take the value linked to the end token to represent the aggregate reasoning score.

We train the verifier using a step-level ranking loss function designed to promote the choice of the most effective solutions, defined as:

$$\mathcal{L} = -\sum_{i=d}^{n} \log \sigma \left(r_\phi(x, y_{1:i}^+) - r_\phi(x, y_{1:i}^-) \right) \quad (2)$$

Here, d is the index where the two reasoning paths diverge, and $r_\phi(x, y_{1:i})$ represents the output of our verifier for an input x and a sequence of steps $\{y_1, \ldots, y_i\}$ with parameters ϕ.

3 Experiments

3.1 Experimental Setup

Tasks and Datasets In our evaluation, we select benchmarks from two distinct reasoning categories: arithmetic reasoning and commonsense reasoning. These tasks encompass diverse reasoning paradigms, enabling a comprehensive assessment of our method's effectiveness. For arithmetic reasoning, we utilize GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021). GSM8K comprises grade school math problems, whereas MATH includes complex competition-level math problems. We use the entire GSM8K test set and a subset of 500 problems from MATH, which is identical to the test set of Lightman et al. (2023). For commonsense reasoning, we employ CSQA (Talmor et al., 2018) and StrategyQA (Geva et al., 2021). CSQA challenges the model with multiple-choice questions that often require reasoning based on complex semantics and prior knowledge. StrategyQA involves true-or-false questions that demand implicit multi-hop reasoning to derive answers.

Metrics Following the methodology suggested by Lightman et al. (2023), we adopt a best-of-N evaluation paradigm. For arithmetic reasoning, we generate 64 solutions for each problem. However, for commonsense reasoning, due to the relatively limited search space for answers in these tasks, we generate 10 solutions for each question.

Models Our experiments leverage a series of models to serve as the generator, including LLaMA2-7B/13B (Touvron et al., 2023) and Mistral-7B (Jiang et al., 2023). For arithmetic reasoning tasks, we additionally conduct experiments with two specialized models: a version of Mistral-7B fine-tuned on MetaMATH (Yu et al., 2023b), and WizardMath-7B (Luo et al., 2023).

Models	Verifiers	Arithmetic		Commonsense	
		GSM8K	MATH500	CSQA	StrategyQA
LLaMA2-7B	Self-Consistency	33.97	4.00	55.12	<u>76.82</u>
	ORM	55.72	5.80	56.27	61.95
	Self-Explore	51.40	<u>7.20</u>	57.82	72.47
	Math-Shepherd	<u>58.30</u>	6.00	<u>58.23</u>	70.78
	Tree-PLV (Ours)	**62.70**	**14.00**	**64.21**	**79.81**
LLaMA2-13B	Self-Consistency	54.51	3.40	64.62	<u>78.77</u>
	ORM	66.34	9.40	66.34	64.74
	Self-Explore	65.66	<u>9.60</u>	66.67	78.64
	Math-Shepherd	<u>68.01</u>	6.60	<u>67.89</u>	71.23
	Tree-PLV (Ours)	**76.12**	**18.20**	**71.66**	**83.64**
Mistral-7B	Self-Consistency	67.55	17.00	68.14	<u>82.86</u>
	ORM	70.74	14.40	63.96	67.14
	Self-Explore	72.18	20.00	67.49	77.99
	Math-Shepherd	<u>74.91</u>	<u>21.20</u>	<u>70.11</u>	79.22
	Tree-PLV (Ours)	**82.79**	**26.80**	**72.97**	**83.25**

Table 1: Results comparison (accuracy %) on arithmetic and commonsense reasoning tasks, with various generators. **Bold** indicates the best results and <u>underline</u> indicates the second best.

Method	GSM8K	MATH500
Mistral-7B: MetaMATH		
Self-Consistency	83.55	35.00
ORM	85.67	29.20
Self-Explore	86.05	34.80
Math-Shepherd	<u>87.11</u>	<u>35.40</u>
Tree-PLV (Ours)	**87.72**	**37.20**
WizardMath-7B		
Self-Consistency	88.93	38.80
ORM	84.31	24.00
Self-Explore	88.86	39.60
Math-Shepherd	<u>89.16</u>	<u>39.80</u>
Tree-PLV (Ours)	**90.14**	**40.20**

Table 2: Results (accuracy %) of the arithmetic reasoning task on generators with stronger capabilities.

Data Collection To construct the training dataset, we selected 6,000 problems from each of the GSM8K and CSQA training sets to generate paired data. We applied a margin threshold of $\alpha = 0.375$ to filter these pairs, resulting in approximately 100k and 120k valid pairs for GSM8K and CSQA, respectively. For additional evaluation, we sampled 750 questions from the StrategyQA training set, which yielded 15k pairs. The verifier then trains for an epoch on the corresponding dataset based on task types. A more detailed description of the training parameter settings is provided in Appendix B.

Baselines We conducted a comparative evaluation against several well-established verifiers. Specifically, we compared Tree-PLV with two verifiers trained using cross-entropy loss with binary labels: the outcome-supervised verifier, termed ORM (Lightman et al., 2023), and Math-Shepherd (Wang et al., 2023), a state-of-the-art process-supervised verifier that leverages automatically annotated data. Additionally, we adopted the self-consistency (Wang et al., 2022) as a strong baseline, following Lewkowycz et al. (2022). Regarding paired data generation, we also considered Self-Explore (Hwang et al., 2024) as a baseline method. It identifies the first erroneous step in a solution and uses the preceding steps to sample a correct path. These sequences are then paired to form training data. We evaluated the quality of data by comparing a verifier trained on this data with ours.

3.2 Main Results

Arithmetic Reasoning As shown in Tables 1, our proposed method significantly outperforms other methods across all scenarios within the GSM8K and MATH500 datasets. For the GSM8K dataset, Tree-PLV consistently surpasses competing approaches. For instance, using the LLaMA2-13B generator, Tree-PLV achieves an accuracy of 76.12%, outperforming the second-best method, Math-Shepherd, by 8.11%. The Tree-PLV, initially trained on the GSM8K dataset, yields effective results on the more complicated MATH500 dataset, illustrating its strong generalization capabilities. Its adaptability stems from the training approach of Tree-PLV, which focuses on comparing the quality of steps rather than simply identifying correct or incorrect paths. This strategy makes the verifier more versatile. Additionally, Tree-PLV requires only 22.7% of the training data size used by Math-

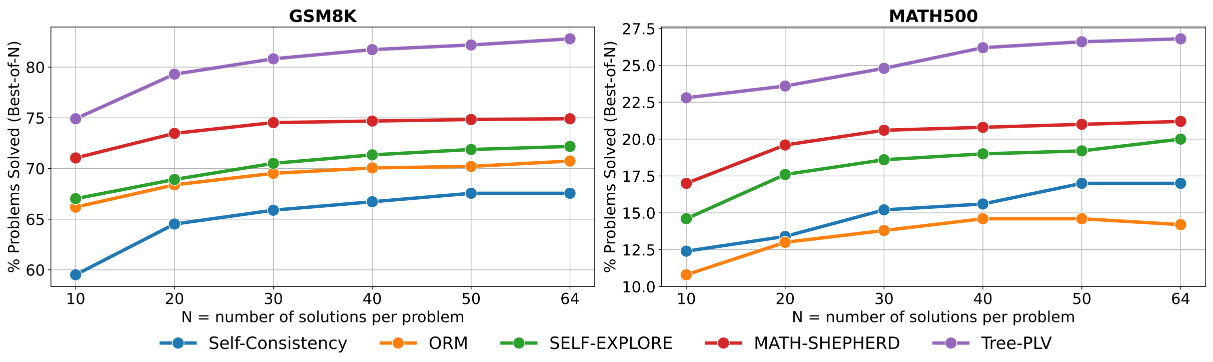

Figure 3: Performance of different verifiers across varying numbers of solution (N) generated by Mistral-7B.

Shepherd, significantly reducing data requirements.

Table 2 displays the results of arithmetic reasoning tasks using advanced generators, namely Mistral-7B: MetaMATH and WizardMath-7B, on the GSM8K and MATH500 datasets. In all scenarios, our method, Tree-PLV, consistently outperforms the others. With Mistral-7B: MetaMATH, it achieves accuracies of 87.72% on GSM8K and 37.20% on MATH500. Similarly, with WizardMath-7B, Tree-PLV delivers the highest accuracies of 90.14% and 40.20% on GSM8K and MATH500, respectively. These results underline Tree-PLV's robust performance and adaptability in tackling complex arithmetic reasoning tasks across various generator settings.

Commonsense Reasoning The results in Table 1 highlights Tree-PLV's strong performance in commonsense reasoning tasks. Regardless of the dataset or the generator used, Tree-PLV consistently outperforms other verifiers, demonstrating its versatility and robustness. In the CSQA dataset, specifically, Tree-PLV improves accuracy by up to 5.98%, 3.77%, and 3.03%, when compared with the second-best model. This high level of accuracy becomes even more pronounced with the StrategyQA dataset, confirming Tree-PLV's strong capabilities in handling intricate reasoning tasks. In contrast, both ORM and Math-Shepherd, which are trained using a binary supervision strategy, fail to show similar improvement. This underperformance could indicate the limitations inherent to binary supervision, especially in tasks that require complex reasoning. It is more prone to introducing noise into the binary supervision signal, which could consequently limit the verifier's capabilities.

We attribute these improvements to the following factors: 1) The training method of our verifier utilizes step-level preference learning, allowing for a nuanced evaluation of step quality that is better aligned with the best-of-N ranking paradigm. 2) The reward function, in our approach, is used to compare sibling steps instead of annotating them, which minimizes the impact of label noise. 3) Our approach improves the diversity of the data by incorporating comparisons among similar steps, thereby enriching the dataset and improving the robustness of our method.

4 Analysis

4.1 Different Amounts of Candidate Solutions

The number of candidate solutions impacts the performance of verification. As the number of solutions increases, more potentially superior candidates are introduced. However, there's a trade-off, as more solutions mean higher computational overhead and diminishing performance gains. For our study, we set a maximum of 64 solutions. As shown in Figure 3, we found that all methods show an enhancement in performance as the number of solutions increases and finally stabilizes at 64, consistent with Wang et al. (2023)'s findings. Throughout the study, the Tree-PLV method consistently outperforms the others across all quantities of solutions. Remarkably, as the number of solutions rises, Tree-PLV continues to widen the performance gap, underscoring its robustness and superior capacity to leverage a greater number of solutions.

4.2 Granularity of Preference Learning

We further analyze the appropriate granularity for applying preference learning, aiming to determine the most effective level of feedback for Tree-PLV. Specifically, we compare our step-level preference with instance-level and conventional token-level training methods (Christiano et al., 2017; Ouyang et al., 2022). Additionally, we include a verifier

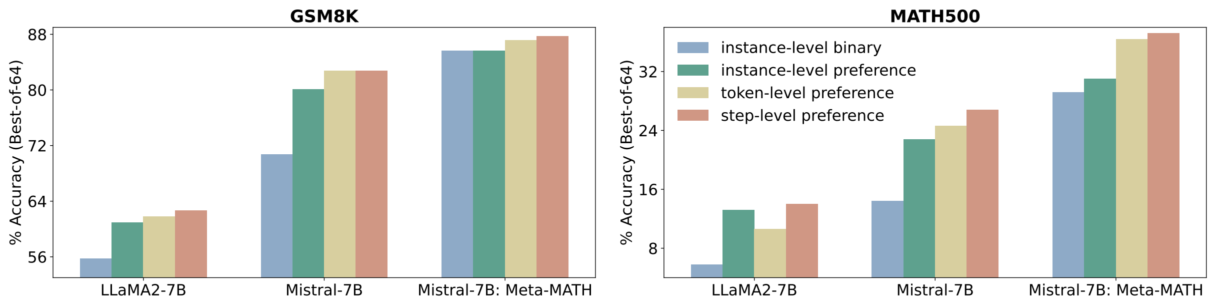

Figure 4: A performance comparison of verifiers trained with different levels of feedback granularity.

trained using instance-level binary classification in our comparison for a comprehensive analysis. The results presented in Figure 4 demonstrate that verifiers trained with preference learning consistently outperform those trained using binary classification across all levels of granularity. This finding suggests that preference learning is better aligned with the ranking evaluation pattern. Furthermore, step-level guidance exhibits the best performance, indicating that it represents the most suitable granularity. In contrast to instance-level sparse supervision, it provides more detailed and informative feedback. Moreover, in step-by-step inference scenarios, stepwise preference proves to be more reasonable than token-level supervision, and such step-level alignment is also more consistent with the intuitive human process of evaluating reasoning.

4.3 Impact on Reasoning Path Distribution

Recent studies have highlighted the difficulties large language models encounter in accurately identifying their errors (Huang et al., 2023; Hong et al., 2023). To assess our verifier's effectiveness, we generate reasoning paths using greedy decoding, which are then scored by the original model's generation confidence and by Tree-PLV. We compare the score distributions from both methods. Figure 6 shows Tree-PLV's capability to separate correct and incorrect reasoning paths. In contrast, generation confidence scores produce overlapping distributions for both categories, highlighting their limited differentiation capacity. Specifically, Tree-PLV increases the scores of correct paths by 35%, whereas the confidence scores show only a marginal 2% higher score for them. This pronounced disparity between the two methods not only confirms the superior performance of Tree-PLV in evaluating the quality of reasoning paths but also emphasizes the critical need for an effective verifier to accurately assess model outputs.

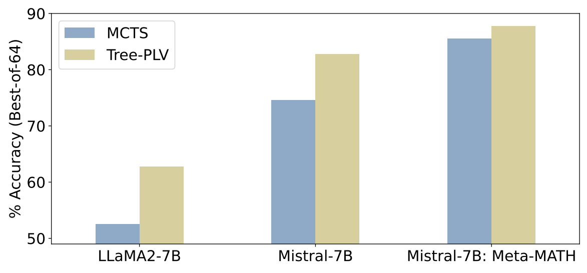

Figure 5: Performance comparison of MCTS and Tree-PLV across different generators on GSM8K.

4.4 Reward Design

Our proposed reward function $\mathcal{R}$ evaluates the quality of the current step by assessing the proportion of its latent complete paths that reach accurate conclusions. Some research demonstrates the advantages of integrating models' self-evaluation capabilities into inference to enhance accuracy (Hao et al., 2023; Ren et al., 2023a; Xie et al., 2023). Inspired by this, we also developed a reward function that relies on self-evaluation. As per Hao et al. (2023), we first construct a reasoning tree based on Monte Carlo Tree Search (MCTS) that incorporates self-evaluation. We then collect preference data from each step of this tree to train a verifier under the same setting. The results, as shown in Figure 5, reveal that the self-evaluation-based reward does not perform as effectively as our proposed reward function. This trend is consistent across different generators, where the performance of the verifier trained with self-evaluation rewards significantly lagged behind our Tree-PLV. Our comparative analysis reveals that relying solely on the model's intrinsic ability to evaluate reasoning steps falls short in reliability. Additional results on the MATH500 dataset are provided in the Appendix C.

4.5 Margin Value for Preference Collection

We delve deeper into the analysis by applying varying margins to filter data. As shown in Figure 7,

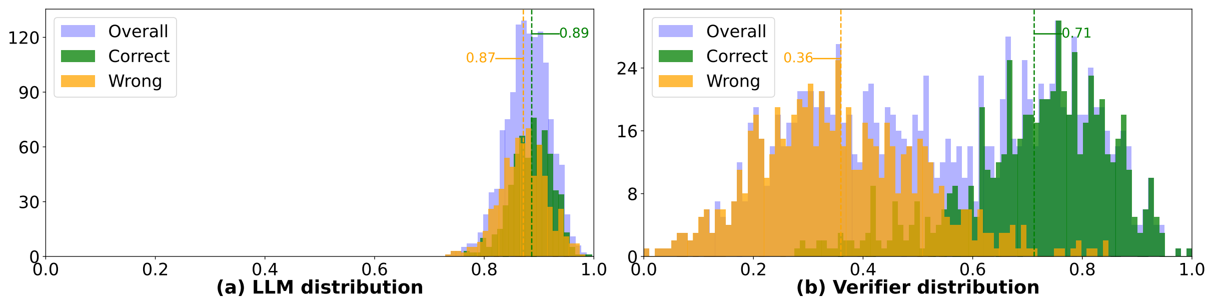

Figure 6: Distributions of the LLM's generation confidence and verifier score on correct/incorrect reasoning paths. The median scores for correct and wrong paths are highlighted with lines of the same respective colors.

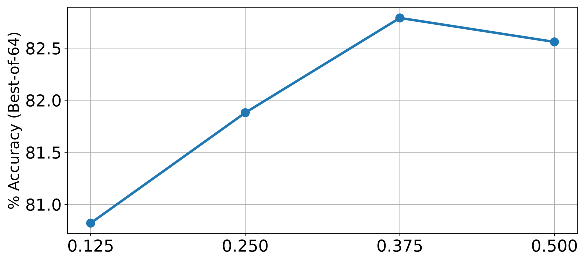

Figure 7: Results on GSM8K, showing verifiers trained with data sampled using different margins.

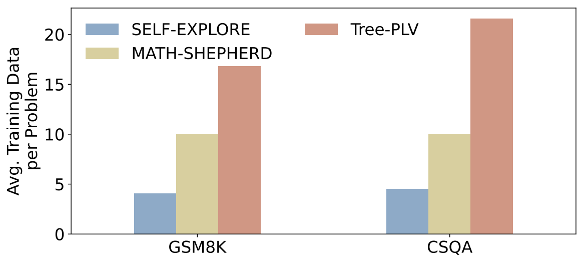

Figure 8: A comparison of methods' efficiency in generating training data.

increasing the margin value raises the proportion of paired data with more noticeable path differences, thereby reducing noise and improving the performance of the verifier. However, when the margin value reaches 0.5, the verifier's performance declines, indicating that excessively high margins can diminish the comparison of similar paths and reduce the richness of the data. Results on the MATH500 dataset are provided in the Appendix C.

4.6 Efficiency of Data Generation

We further analyze our tree-based framework across the dimensions of data generation efficiency. We compare the average number of training data per problem generated by different methods over 10 iterations, as shown in Figure 8. Math-Shepherd consistently labels ten paths. In contrast, Self-Explore struggles with datasets where the model already performs well, as it inherently relies on incorrect paths to form pairs. Our framework enhances the utilization of single paths through the branching structure of a tree, demonstrating significant scalability and efficiency in data generation.

5 Related Work

Recent studies (Lightman et al., 2023; Cobbe et al., 2021; Yu et al., 2023a) have demonstrated that incorporating a verifier can significantly enhance performance in reasoning tasks. Primarily, there are two methods for training verifiers: outcome supervision and process supervision. Unlike these approaches, which rely on supervised learning with binary labels, our methodology employs preference learning to achieve step-level alignment. We provide a detailed discussion of additional related work in Appendix A.

6 Conclusion

In this paper, we propose Tree-PLV, an innovative verifier developed through step-level preference learning. Utilizing a best-first tree search framework to construct reasoning trees, this method creates a diverse and robust dataset, ensuring a rich pool of paired data for preference training. By leveraging step-level preferences instead of conventional binary labels, our methodology provides more granular and detailed feedback, enhancing the verification of reasoning paths. Our empirical findings highlight the efficacy of Tree-PLV, showing significant performance improvements over existing benchmarks across a variety of arithmetic and commonsense reasoning tasks. In future work, we aim to delve deeper into integrating our verifier within the reasoning process by combining it with various search algorithms, thereby refining the verification of intermediate reasoning steps.

Limitations

While Tree-PLV is effective at scoring complete solutions, its potential to provide feedback that assists and refines the reasoning process during the inference phase has not been explored in this work. Additionally, aside from assisting in reasoning to encourage successful outcomes, some studies treat verifiers as reward models in reinforcement learning scenarios. However, our current research focuses on developing highly reliable verifiers, rather than employing reinforcement learning to enhance the generator. Addressing these aspects will be the focus of our future research.

Ethics Statement

The development of Tree-PLV aims to improve the accuracy and reliability of reasoning assessments in Large Language Models. Although our method poses no immediate ethical concerns, we acknowledge the potential for misuse if applied in sensitive areas such as automated decision-making. We recommend rigorous evaluation and oversight to prevent bias and ensure data privacy in all applications. It is crucial to maintain transparency and adhere to ethical standards in the deployment of such technologies.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Benjamin Mann, Nova Dassarma, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, John Kernion, Kamal Ndousse, Catherine Olsson, Dario Amodei, Tom B. Brown, Jack Clark, Sam McCandlish, Christopher Olah, and Jared Kaplan. 2021. A general language assistant as a laboratory for alignment. *ArXiv*, abs/2112.00861.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Antonia Creswell and Murray Shanahan. 2022. Faithful reasoning using large language models. *arXiv preprint arXiv:2208.14271*.

Simon Frieder, Luca Pinchetti, Ryan-Rhys Griffiths, Tommaso Salvatori, Thomas Lukasiewicz, Philipp Petersen, and Julius Berner. 2024. Mathematical capabilities of chatgpt. *Advances in Neural Information Processing Systems*, 36.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.

Olga Golovneva, Moya Chen, Spencer Poff, Martin Corredor, Luke Zettlemoyer, Maryam Fazel-Zarandi, and Asli Celikyilmaz. 2022. Roscoe: A suite of metrics for scoring step-by-step reasoning. *arXiv preprint arXiv:2212.07919*.

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. *arXiv preprint arXiv:2103.03874*.

Ruixin Hong, Hongming Zhang, Xinyu Pang, Dong Yu, and Changshui Zhang. 2023. A closer look at the self-verification abilities of large language models in logical reasoning. *arXiv preprint arXiv:2311.07954*.

Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron Courville, Alessandro Sordoni, and Rishabh Agarwal. 2024. V-star: Training verifiers for self-taught reasoners. *arXiv preprint arXiv:2402.06457*.

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

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2023. Large language models cannot self-correct reasoning yet. *arXiv preprint arXiv:2310.01798*.

Hyeonbin Hwang, Doyoung Kim, Seungone Kim, Seonghyeon Ye, and Minjoon Seo. 2024. Self-explore to avoid the pit: Improving the reasoning capabilities of language models with fine-grained rewards. *arXiv preprint arXiv:2404.10346*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Jean Kaddour, Joshua Harris, Maximilian Mozes, Herbie Bradley, Roberta Raileanu, and Robert McHardy. 2023. Challenges and applications of large language models. *arXiv preprint arXiv:2307.10169*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. 2022. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, B. Chen, Jian-Guang Lou, and Weizhu Chen. 2022. Making language models better reasoners with step-aware verifier. In *Annual Meeting of the Association for Computational Linguistics*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Hanmeng Liu, Ruoxi Ning, Zhiyang Teng, Jian Liu, Qiji Zhou, and Yue Zhang. 2023. Evaluating the logical reasoning ability of chatgpt and gpt-4. *arXiv preprint arXiv:2304.03439*.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. 2023. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*.

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. 2023. Faithful chain-of-thought reasoning. *arXiv preprint arXiv:2301.13379*.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*.

Ansong Ni, Jeevana Priya Inala, Chenglong Wang, Oleksandr Polozov, Christopher Meek, Dragomir Radev, and Jianfeng Gao. 2022. Learning math reasoning from self-sampled correct and partially-correct solutions. *arXiv preprint arXiv:2205.14318*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Archiki Prasad, Swarnadeep Saha, Xiang Zhou, and Mohit Bansal. 2023. Receval: Evaluating reasoning chains via correctness and informativeness. *arXiv preprint arXiv:2304.10703*.

Jie Ren, Yao Zhao, Tu Vu, Peter J Liu, and Balaji Lakshminarayanan. 2023a. Self-evaluation improves selective generation in large language models. *arXiv preprint arXiv:2312.09300*.

Ruiyang Ren, Yuhao Wang, Yingqi Qu, Wayne Xin Zhao, Jing Liu, Hao Tian, Hua Wu, Ji-Rong Wen, and Haifeng Wang. 2023b. Investigating the factual knowledge boundary of large language models with retrieval augmentation. *arXiv preprint arXiv:2307.11019*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2018. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel Bowman. 2024. Language models don't always say what they think: unfaithful explanations in chain-of-thought prompting. *Advances in Neural Information Processing Systems*, 36.

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. *arXiv preprint arXiv:2211.14275*.

Peiyi Wang, Lei Li, Zhihong Shao, RX Xu, Damai Dai, Yifei Li, Deli Chen, Y Wu, and Zhifang Sui. 2023. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *CoRR, abs/2312.08935*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, Xu Zhao, MingSung Kan, Junxian He, and Qizhe Xie. 2023. Self-evaluation guided beam search for reasoning. In *Neural Information Processing Systems*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.

Fei Yu, Anningzhe Gao, and Benyou Wang. 2023a. Outcome-supervised verifiers for planning in mathematical reasoning. *arXiv preprint arXiv:2311.09724*.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023b. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Chuanqi Tan, and Chang Zhou. 2023. Scaling relationship on learning mathematical reasoning with large language models. *arXiv preprint arXiv:2308.01825*.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488.

Xinyu Zhu, Junjie Wang, Lin Zhang, Yuxiang Zhang, Yongfeng Huang, Ruyi Gan, Jiaxing Zhang, and Yujiu Yang. 2022. Solving math word problems via cooperative reasoning induced language models. *arXiv preprint arXiv:2210.16257*.

A Related Work

Multi-step Reasoning Challenging reasoning tasks have spurred innovative research in large language models (LLMs), which are essential for handling complex queries (Kaddour et al., 2023; Lightman et al., 2023; Huang et al., 2023). A prominent strategy employed is the Chain-of-Thought (CoT) prompting technique (Wei et al., 2022), along with its derivatives (Kojima et al., 2022; Wang et al., 2022; Yao et al., 2024). These methods decompose the reasoning process into sequential steps, systematically approaching problem-solving by mimicking human-like reasoning. To further enhance the accuracy of these intermediate steps, recent studies leverage extensive synthetic datasets, which are either distilled from cutting-edge models (Yu et al., 2023b; Luo et al., 2023) or composed of self-generated rationales (Zelikman et al., 2022; Yuan et al., 2023; Ni et al., 2022), to fine-tune the LLMs. Such training effectively sharpens the models' ability to produce CoT reasoning that leads to correct answers.

In addition, there is an increasing emphasis on test-time verification, a process that generates multiple solutions and ranks them through a separate verifier (Cobbe et al., 2021) to select the most accurate one. The DIVERSE framework (Li et al., 2022) demonstrates the effectiveness of employing a variety of CoT prompts in conjunction with a verifier to tackle reasoning challenges. CoRe (Zhu et al., 2022) implements a dual-process system by fine-tuning both the generator and the verifier, thereby enhancing the performance of LLMs on math word problems.

Training Verifiers The training process for verifiers is typically approached as a binary classification task. In closely related work, Uesato et al. (2022) introduces two distinct training methodologies for verifiers, differentiated by the granularity of the supervision signal: outcome supervision (Cobbe et al., 2021; Yu et al., 2023a; Hosseini et al., 2024) and process supervision (Li et al., 2022; Uesato et al., 2022; Lightman et al., 2023; Wang et al., 2023). Verifiers trained under outcome supervision utilize only the final result of a reasoning path for learning, while those trained under process supervision benefit from incremental feedback at each step within the CoT.

The training objective for this binary classification does not align well with its intended application in ranking scenarios. The binary labels provide a coarse supervisory signal, which fails to differentiate the efficacy of various steps, thus capping potential enhancements in performance. Addressing these issues is not straightforward, as accurately assessing the quality of each step involves subjective judgments that are difficult to standardize, even with human annotations. Math-Shepherd (Wang et al., 2023) employs heuristic rules to annotate individual steps; however, the verifier trained on these refined labels does not show improvement compared to its binary version. Our method, Tree-PLV, fundamentally rethinks the verifier's role by incorporating step-level preference learning. This approach more effectively captures subtle differences among paths and better aligns with the ranking evaluation paradigm.

B Experimental Details

B.1 Dataset Description

Table 3 presents the reasoning benchmarks used in this paper, along with examples. We utilize the same test sets as Wang et al. (2023) for GSM8K and MATH500, and as Wei et al. (2022) for CSQA.

For StrategyQA, there are 2,290 test cases (i.e., questions paired with TRUE/FALSE labels). However, there are no additional cases that can be leveraged by Tree-PLC to construct the training set. To address this issue, we randomly sampled 750 test cases to create the training data, leaving the remaining 1,540 cases for testing.

B.2 Training Data

To construct the training set for Tree-PLV, we select 6,000 problems each from the GSM8K and CSQA training datasets to generate paired data. We sample 8 subsequent trajectories (N=8) to calculate the reward for each step. By applying a margin threshold of $\alpha = 0.375$, we filter these pairs, resulting in approximately 100k valid pairs from GSM8K and 120k from CSQA. Additionally, we sampled 750 questions from the StrategyQA training set, yielding 15k pairs.

B.3 Training Details

Considering our limited computational resources, we use Mistral-7B as the backbone to train our verifier via the LoRA (Hu et al., 2021) training method for one epoch. We choose a learning rate of 1e-6, implemented with a cosine learning rate scheduler. Specifically for arithmetic reasoning, the base model is fine-tuned on MetaMATH, fol-

Dataset	N	Example Problem
GSM8K	1319	James decides to run 3 sprints 3 times a week. He runs 60 meters each sprint. How many total meters does he run a week?
MATH500	500	A regular hexagon can be divided into six equilateral triangles. If the perimeter of one of the triangles is 21 inches, what is the perimeter, in inches, of the regular hexagon?
CSQA	1221	Where would you find magazines along side many other printed works? Answer Choices: (A) doctor (B) bookstore (C) market (D) train station (E) mortuary
StrategyQA	1540	Could a silverfish reach the top of the Empire State Building?

Table 3: Reasoning benchmarks we use in this paper with examples. N: number of test cases.

lowing Wang et al. (2023). Furthermore, ORM, Self-Explore, and Math-Shepherd are trained under settings identical to those we use.

B.4 Prompt Examples

We present a prompt example for generating solutions and performing reasoning. We employ few-shot learning to guide the model towards producing outputs in the step-by-step format.

Solve the questions step by step. Make each step clear and logical, leading to the next. If a step directly leads to an answer, end it with "The answer is: ". If not, proceed to the next step until you find the solution.

Four years ago, Kody was only half as old as Mohamed. If Mohamed is currently twice as 30 years old, how old is Kody?
Step 1: If Mohamed is currently twice as old as 30 years old, then Mohamed is currently 2 * 30 = 60 years old.
Step 2: Four years ago, Mohamed was 60 - 4 = 56 years old.
Step 3: Four years ago, Kody was only half as old as Mohamed, so Kody was (1/2) * 56 = 28 years old.
Step 4: Therefore, Kody is currently 28 + 4 = 32 years old. The answer is: 32.

<more examples>

B.5 Preference Pair

We present an example of preference pairs $\{(x, y^+, y^-)\}$ used for training Tree-PLV, where x represents the problem statement, y^+ denotes the preferred reasoning sequence, and y^- is a less desirable path.

x: One pie costs $4 for a piece. Each pie is having 3 pieces. During one hour the bakery can make 12 pies. Creating one pie costs the bakery $0.5. Considering the bakery would be able to sell all pie pieces, how much money would it make?

y^+:
Step 1: Each pie is having 3 pieces.
Step 2: 12 pies is equal to 3 * 12 = 36 pieces.
Step 3: Each piece costs $4, so the bakery would earn 36 * $4 = $144 if it could sell all of its pieces.
Step 4: The cost of making one pie is $0.5, so the cost of making 12 pies is 12 * $0.5 = $6.
Step 5: The bakery would make 144−6 = $138. The answer is: 138.

y^-:
Step 1: Each pie is having 3 pieces.
Step 2: 12 pies is equal to 3 * 12 = 36 pieces.
Step 3: The bakery can make 36 pieces during one hour.
Step 4: One piece costs $4 and a pie costs $0.5, so it costs the bakery (1/3) * $4 + (2/3) * $0.5 = $1.5 to create one pie.
Step 5: The bakery would make 12 * $1.5 = $18 during one hour. The answer is: 18.

C Additional Experiment Results

C.1 Additional Results of Reward Design

In Figure 9, we show the additional results on the MATH500 dataset for verifiers in Section 4.4. The results of the experiment are consistent with our pa-

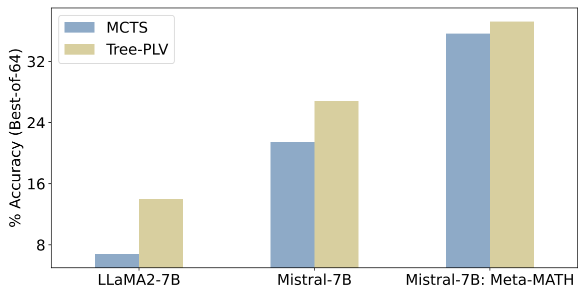

Figure 9: Performance comparison of MCTS and Tree-PLV across different generators on MATH500.

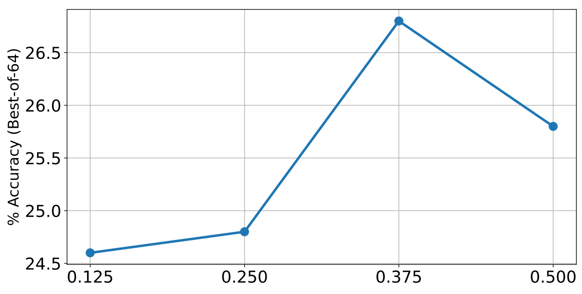

Figure 10: Results on MATH500, showing verifiers trained with data sampled using different margins.

per's analysis; our reward design is comparatively more effective.

C.2 Additional Results on Different Margins

In this section, we provide the experimental results of different verifiers on MATH500 in Section 4.5. As shown in Figure 10, the performance trend of the verifiers with different margin values is consistent with the trend on GSM8K.