MT-RewardTree: A Comprehensive Framework for Advancing LLM-Based Machine Translation via Reward Modeling

Anonymous ACL submission

Abstract

Process reward models (PRMs) have shown success in complex reasoning tasks for large language models (LLMs). However, their application to machine translation (MT) remains underexplored due to the lack of systematic methodologies and evaluation benchmarks. To address this gap, we introduce **MT-RewardTree**, a comprehensive framework for constructing, evaluating, and deploying process reward models in MT. Unlike traditional vanilla preference pair construction, we propose a novel method for automatically generating token-level preference pairs using approximate Monte Carlo Tree Search (MCTS), which mitigates the prohibitive cost of human annotation for fine-grained steps. Then, we establish the first MT-specific reward model benchmark and provide a systematic comparison of different reward modeling architectures, revealing that token-level supervision effectively captures fine-grained preferences. Experimental results demonstrate that our MT-PRM-Qwen-2.5-3B achieves state-of-the-art performance in both token-level and sequence-level evaluation given the same input prefix. Furthermore, we showcase practical applications where MT-PRMs successfully identify token-level translation differences and enable test-time alignment for LLMs without additional alignment training. Our work provides valuable insights into the role of reward models in MT research. Our code and data will be released.

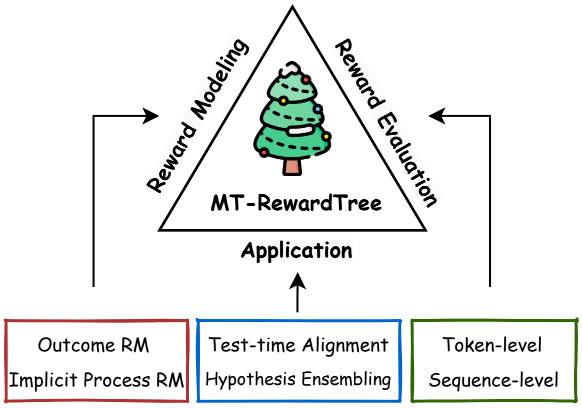

Figure 1: Components of MT-RewardTree.

1 Introduction

The next-token prediction process in large language models (LLMs) is often modeled as a Markov Decision Process (MDP) and has achieved remarkable success across various domains, largely attributed to reinforcement learning (RL) and the scaling of test-time compute (Snell et al., 2024; Zeng et al., 2024; DeepSeek-AI et al., 2025; Team, 2025; Xiang et al., 2025). Reward models are central to these advancements. Outcome Reward Models (ORMs), which are designed to evaluate full responses, have been widely adopted; however, due to the sparsity of outcome rewards, ORMs often yield suboptimal performance and struggle with stability and efficiency during RL training (Lightman et al., 2024; Cao et al., 2024; Chan et al., 2024). In contrast, Process Reward Models (PRMs) evaluate intermediate steps to provide fine-grained guidance during both training and inference. PRMs have proven particularly effective in tasks such as mathematics and coding by guiding stepwise decision-making (Wang et al., 2024a; Chen et al., 2024; Luo et al., 2024; Qi et al., 2024; Guan et al., 2025).

Machine translation (MT) naturally aligns with token-level MDP frameworks, as each translation decision corresponds directly to token generation. However, there is still a lack of systematic methodologies for constructing and evaluating PRMs in MT, which has hindered progress relative to advancements in general-domain LLMs.

Developing effective PRMs is challenging. Although Lightman et al. (2024) demonstrate that process supervision with human annotators improves PRM performance in mathematical tasks, this methodology requires domain-expert annotators, resulting in prohibitive costs and practical limitations for translation tasks. Recently, some studies suggest that a PRM can be automatically learned during Direct Preference Optimization (DPO) training (Rafailov et al., 2024b,a; Yuan et al., 2024). However, existing vanilla preference pair datasets provide only sequence-to-sequence preference data, rather than token-level preferences, which raises concerns about their applicability for token-level alignment. Additionally, evaluating PRMs remains a significant challenge. In mathematical tasks, evaluation is often done using a Best-of-N (BoN) sampling strategy—selecting the highest-scored response from N candidates based on a PRM (Lightman et al., 2024; Wang et al., 2024b; Luo et al., 2024)—or by having the PRM identify errors or verify correctness in the steps (Zheng et al., 2024; Zhang et al., 2025). Since each step in mathematics has a deterministic answer, these methods do not directly translate to PRM evaluation in MT.

In this paper, we introduce **MT-RewardTree**, a comprehensive framework for constructing, evaluating, and deploying PRMs in machine translation. We propose an approximate Monte Carlo Tree Search (MCTS) method (Kocsis and Szepesvári, 2006; Silver et al., 2016) to generate the token-level preference pair dataset. This dataset is then split into a training set for reward model development and a benchmark for reward evaluation. We provide a systematic comparison of different reward modeling methods and test on both token-level and sequence-level performance. Furthermore, we demonstrate two practical applications of PRMs, offering valuable insights for future MT research. Our main contributions are as follows:

- We introduce MT-RewardTree, a comprehensive framework for the construction, evaluation, and deployment of PRMs in MT. We establish the first dedicated reward benchmark - MT-PRMBench. Our experiments demonstrate that MT-PRMs achieve competitive performance on both token-level and sequence-level evaluations.

- Comprehensive experiments indicate that our token-level preference pairs, generated through an approximate MCTS method, significantly outperform vanilla preference pairs in process reward model training. Furthermore, our analysis validates that supervising PRMs using preference-based signals is more effective than direct supervision with absolute value estimates.

- We demonstrate that our MT-PRMs can directly identify token-level translation differences and facilitate test-time alignment for LLM-based MT without the need for additional alignment training, offering valuable practical insights for the application of reward models in MT.

2 Background

2.1 Token-level Markov Decision Process

LLMs' autoregressive generation can be naturally formulated as a Markov Decision Process, where each token generation is treated as an action. At each time step t, an action a_t corresponds to the generation of a new token, and the state $\mathbf{s}_t$ is represented as the sequence of tokens generated up to that point. For tasks that do not involve interaction with an external environment—such as translation—the state is defined as

$$\mathbf{s}_t = (x_0, \ldots, x_L, y_0, \ldots, y_{t-1}),$$

where $(x_0, \ldots, x_L)$ represents the input prompt and $(y_0, \ldots, y_{t-1})$ is the sequence of generated tokens until time step $t-1$. The state transition function f is deterministic and updates the state by concatenating the newly generated token:

$$\mathbf{s}_{t+1} = f(\mathbf{s}_t, a_t) = \mathbf{s}_t \mid a_t,$$

with | denoting concatenation.

Within this token-level MDP framework, the reward function $r(\mathbf{s}_t, a_t)$ is typically designed to provide feedback only at the terminal time step T, reflecting the overall correctness of the generated sequence or the successful completion of the task. To optimize the policy π_θ based on this reward, Reinforcement Learning with Human Feedback (RLHF) (Ouyang et al., 2022) typically maximizes a KL-constrained objective:

$$\mathbb{E}_{(s_0,\ldots,s_T)\sim\rho_\pi}\left[\sum_{t=0}^{T}\left(r(s_t,a_t) - \beta \log \frac{\pi(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)}\right)\right], \quad (1)$$

where π_{ref} is a pre-trained reference policy, β controls the strength of the KL penalty and ρ_π denotes the trajectory distribution induced by policy π.

In practice, classical RLHF applies the reward solely at the terminal state. Specifically, the reward function used in Proximal Policy Optimization (PPO) (Schulman et al., 2017) is defined as:

$$r(s_t, a_t) = \begin{cases} \beta \log \pi_{\text{ref}}(a_t \mid s_t), & \text{if } s_{t+1} \text{ is non-terminal,} \\ r(x, y) + \beta \log \pi_{\text{ref}}(a_t \mid s_t), & \text{if } s_{t+1} \text{ is terminal.} \end{cases} \quad (2)$$

2.2 Reward Modeling in RLHF

Reward modeling is the cornerstone of RLHF, enabling LLMs to align their outputs with human preferences. In this section, we distinguish between typical (sequence-level) reward modeling and the more fine-grained token-level approach.

Sequence-level Reward Modeling. In classical RLHF, the reward function is learned from human feedback on prompt-response pairs $(\mathbf{x}, \mathbf{y})$. The reward model is formulated as a contextual bandit, where a scalar reward is assigned only at the terminal state—i.e., once the full response sequence has been generated. This formulation, known as Outcome Reward Modeling, follows the Bradley-Terry (Bradley and Terry, 1952) preference model to define the probability of preferring one response over another:

$$p^*(\mathbf{y}^w \succeq \mathbf{y}^l) = \frac{\exp\left(r_\phi(\mathbf{x},\mathbf{y}^w)\right)}{\exp\left(r_\phi(\mathbf{x},\mathbf{y}^w)\right)+\exp\left(r_\phi(\mathbf{x},\mathbf{y}^l)\right)}. \quad (3)$$

To train the reward model r_ϕ, we construct a preference dataset $\mathcal{D}$, where each prompt x is paired with two candidate responses, y and y'. Human annotators or heuristics determine the preferred response y_w and the rejected response y_l. The reward model is then optimized to maximize the likelihood of these human preferences:

$$\max_\phi \mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}} \left[\log \sigma(r_\phi(x, y_w) - r_\phi(x, y_l))\right], \quad (4)$$

where σ is the logistic function. By training r_ϕ in this manner, we ensure that the model assigns higher rewards to preferred responses, effectively capturing human-like quality judgments for sequence-level evaluation.

Token-level Reward Modeling. While the sequence-level approach treats the entire generated response as a single action, it fails to capture the fine-grained decision-making process inherent in token generation. Token-level reward modeling addresses this limitation by evaluating rewards at each token-generation step. This approach corresponds to a form of Process Reward Models. The cumulative reward for a trajectory τ is computed as the sum of per-token rewards, and the corresponding preference probability between two trajectories, τ^w and τ^l, is given by:

$$p^*(\tau^w \succeq \tau^l) = \frac{\exp\left(\sum_{i=1}^{N} r(\mathbf{s}_i^w, a_i^w)\right)}{\exp\left(\sum_{i=1}^{N} r(\mathbf{s}_i^w, a_i^w)\right)+\exp\left(\sum_{i=1}^{M} r(\mathbf{s}_i^l, a_i^l)\right)}. \quad (5)$$

Although token-level reward modeling offers finer-grained feedback, obtaining effective PRMs is more challenging to obtain and deploy (Lightman et al., 2024; Cao et al., 2024).

3 MT-RewardTree

In this section, we introduce the components of the MT-RewardTree. We first describe how we construct token-level preference pairs using an MCTS-based method. Next, we review several approaches employed for reward modeling.

3.1 Constructing Token-level Preference Pairs

Prior studies have investigated translation preference pair construction (Xu et al., 2024; Agrawal et al., 2024; Feng et al., 2024a), yet a standardized token-level preference pair dataset for PRMs in MT remains absent. MQM (Freitag et al., 2021) datasets depend on manual error annotation, which is both cost-prohibitive and incapable of producing granular token-level preference pairs.

Drawing inspiration from MCTS, we propose a token-centric approach that quantifies token quality based on its potential to contribute to higher-quality translations. This method aligns with Monte Carlo-based PRMs construction techniques in mathematics, where step-wise quality is determined by its incremental contribution to deriving correct answers (Wang et al., 2024b; Guan et al., 2025).

The MCTS process consists of four main steps (depicted in Figure 2): Selection, Expansion, Simulation (Evaluation), and Back-propagation.

1. **Selection**: The first phase involves selecting a portion of the existing tree that is most promising for further expansion. Starting from the root node, a standard approach would traverse the tree down to a leaf using the PUCT algorithm (Rosin, 2011; Silver et al., 2017). Since our goal is to construct token-level preference pairs rather than achieving global optimality, we automatically select the existing prompt and previously generated tokens as the prefix $y_{<t}$.

2. **Expansion**: If the selected leaf node is not an EOS (end-of-sentence) token—i.e. if it is not a terminal state—the node is expanded by generating k candidate children. This is achieved by decoding one additional step using the language model and selecting the top-k tokens as the new children. We select the top-2 candidate tokens a_{tj} (with $j \in \{1, 2\}$) that have the highest logits. Preliminary experiments demonstrate that tokens outside of the top-2 yield significantly lower translation quality during the **Simulation** phase. These top-2 tokens, sharing the same

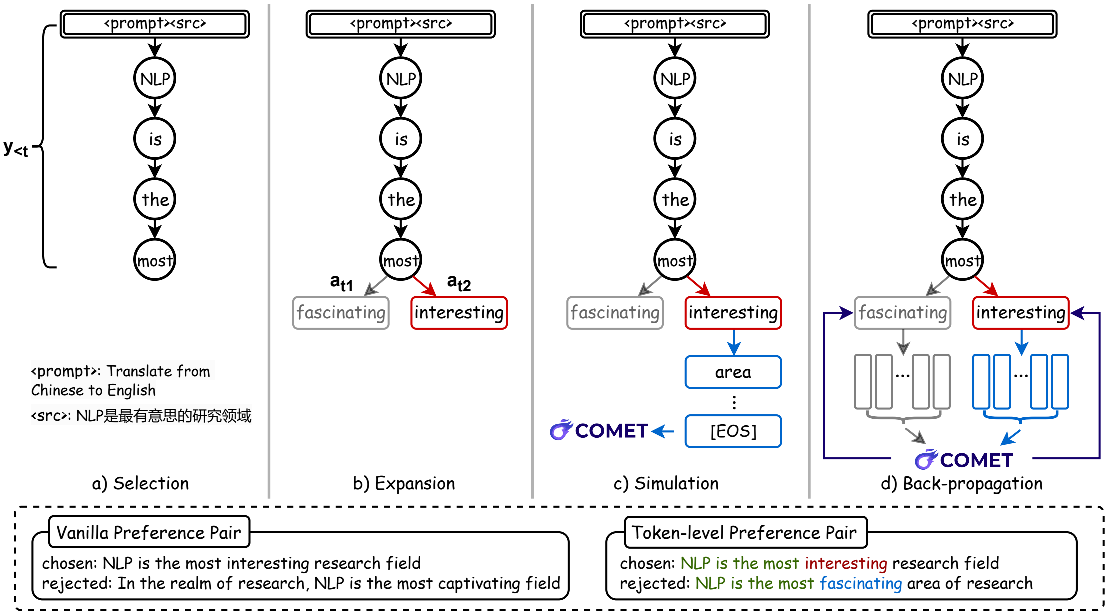

Figure 2: The construction process of token-level preference pairs. We utilize TowerInstruct-7B-v0.2 to generate candidate tokens. A *token-level* preference pair comprises two translations that share an identical prefix.

prefix $y_{<t}$, form the basis for our token-level preference pair.

3. **Simulation (Evaluation)**: From each expanded node a, we generate n complete translation rollouts until an EOS token is reached. We then evaluate the quality (or groundedness) of the full translation sequence, denoted by $g(y, n)$. In our framework, we use COMETKiwi (Rei et al., 2022) to estimate the quality of all n full rollouts. These scores are averaged and further assigned as the value of node a, i.e., $V(a)$.

4. **Back-propagation**: Since our objective is to construct token-level preference pairs, we compare the values $V(a_{t1})$ and $V(a_{t2})$ to determine which expanded token is superior. Finally, we retain the node with the higher V value. This node, along with its corresponding prefix $y_{<t}$, is then used as the starting point in the next simulation cycle, beginning again at Step 1.

These four steps are repeated until the EOS token appears during the Selection phase. We retain one rollout from the superior token and one from the inferior to construct our token-level preference pair. We use COMETKiwi to guarantee the score gap lies between 0.04 and 0.4 to control the quality.

3.2 Implicit Process Reward Modeling

Unlike ORMs, which assign a single reward to the entire response, PRMs aim to assign rewards at a finer granularity, such as at each step or token. However, traditional PRMs training requires step-level annotations, which are costly to obtain. Recent studies (Rafailov et al., 2024a; Zhong et al., 2024) show that ORMs can be trained with implicit reward modeling, enabling PRMs to emerge naturally without the need for explicit step labels.

Consider an ORM where the reward is parameterized by the log-likelihood ratio of two causal language models:

$$r_\theta(\mathbf{y}) := \beta \log \frac{\pi_\theta(\mathbf{y})}{\pi_{\text{ref}}(\mathbf{y})} \tag{6}$$

where π_θ represents the trained model's probability distribution, and π_{ref} is a reference model. We define the cumulative reward up to step t as:

$$q_\theta^t(\mathbf{y}_{<t}, y_t) := \sum_{i=1}^{t} \beta \log \frac{\pi_\theta(y_i|\mathbf{y}_{<i})}{\pi_{\text{ref}}(y_i|\mathbf{y}_{<i})} \tag{7}$$

which serves as an exponential moving average of r_θ across steps. The expected process reward at step t can then be expressed as:

$$q_\theta^t(\mathbf{y}_{<t}, y_t) = \beta \log \mathbb{E}_{\pi_{\text{ref}}(\mathbf{y}|\mathbf{y}_{\leq t})} \left[e^{\frac{1}{\beta} r_\theta(\mathbf{y})} \right] \tag{8}$$

This formulation shows that q_θ^t is an exact expectation of the outcome reward r_θ at step t, making it analogous to a Q-value in reinforcement learning.

By defining the process reward r_θ^t as the difference between successive Q-values:

$$r_\theta^t := q_\theta^t - q_\theta^{t-1} = \beta \log \frac{\pi_\theta(y_t|\mathbf{y}_{<t})}{\pi_{\text{ref}}(y_t|\mathbf{y}_{<t})} \tag{9}$$

Model	MT-PRMBench					
	Sequence-level			Token-level		
	EN→XX	XX→EN	Avg.	EN→XX	XX→EN	Avg.
Baselines						
Skywork-Reward-LLaMA-3.1-8B	0.857	0.773	0.815	-	-	-
MT-Ranker-base	0.785	0.787	0.786	-	-	-
MT-Ranker-large	0.847	0.873	0.860	-	-	-
PRMs						
MT-PRM-LLaMA-3.2-3B	0.777	0.775	0.776	0.542	0.615	0.578
MT-PRM-Qwen-2.5-3B	0.867	0.858	0.863	0.637	0.685	0.660

Table 1: Accuracy results on MT-PRMBench. Skywork-Reward-LLaMA-3.1-8B is an advanced ORM for general domains, while MT-Ranker represents the SoTA non-metric reference-free translation quality estimation model.

Preference Pair Type	**Training Strategy**	**Avg.**
Token-level	DPO	0.660
Vanilla	DPO	0.574
Token-level	KTO	0.644
Vanilla	KTO	0.562

Table 2: Ablation study on the effect of training preference data and implicit reward training objectives. The backbone model is Qwen-2.5-3B-Instruct and we test these variants on MT-PRMBench (Token-level).

We see that PRMs can be derived directly from an ORM trained on response-level data, without requiring explicit step-wise labels. This insight suggests that training an ORM inherently leads to the learning of a Q-function, enabling step- or token-level reward modeling without requiring additional supervision. A typical example of this is DPO (Rafailov et al., 2024b), which optimizes the following objective:

$$L_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}} \log\sigma\left(\beta \log\frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta\log\frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right) \tag{10}$$

This formulation shows that optimizing π_θ implicitly optimizes a reward model, as described in Eq. 4. Moreover, Yuan et al. (2024) demonstrated that this approach is agnostic to the specific training objective (i.e., not limited to DPO). It can be instantiated using various training objectives (e.g., KTO (Ethayarajh et al., 2024)), with the only modification being the substitution of $r_\theta(\mathbf{y})$ with $\beta \log \frac{\pi_\theta(\mathbf{y})}{\pi_{\text{ref}}(\mathbf{y})}$.

Moreover, our implicit PRMs can seamlessly be converted into ORMs using weighted implicit rewards:

$$r_{\text{sequence}}(y_{1:T}) = \sum_{k=0}^{T-1} w_t \log\frac{\pi_\theta(y_t|y_{<t})}{\pi_{\text{ref}}(y_t|y_{<t})} \tag{11}$$

where the positional weights $w_t = \frac{1}{|y_{<t}|}$ are used to balance the contributions of each token.

4 Experiments

4.1 Experimental Setup

Datasets. We explore four languages—English (EN), German (DE), Chinese (ZH), and Russian (RU)—and six translation directions: EN→XX and XX→EN. Our raw corpus consists of test sets from WMT17 to WMT20, supplemented with development and test sets from the Flores (Costa-jussà et al., 2022). We use the TowerInstruct-7B-v0.2[1] model with a temperature of 0.95 and apply the MCTS-based approach described earlier. During the Simulation step, we sample three candidate hypotheses for each node. Our token-level preference pairs is divided into train and test set (MT-PRMBench). Detailed statistics are in Table 5.

Training Details. We take LLaMA-3.2-3B-Instruct and Qwen-2.5-3B-Instruct as the backbone models for training. For the DPO training, the higher-scored sentence is designated as the chosen response, while the lower-scored sentence is labeled as the rejected response. For the KTO training, the higher-scored sentence is treated as the positive sample, and the lower-scored sentence as the negative sample. We set β as 0.1.

Reward Evaluation. We evaluate reward models by framing the task as a classification problem, similar to prior work on reward model benchmarks in the general domain (Lambert et al., 2024; Liu et al., 2024). For sequence-level evaluation, given a tuple (x, y_c, y_r), where x is the prompt, y_c is the chosen response, and y_r is the rejected response, the reward model predicts whether y_c is better than y_r. If the reward model assigns a higher reward to y_c than to y_r, the prediction is correct; otherwise, it is incorrect. We use accuracy as the evaluation

[1]https://huggingface.co/Unbabel/TowerInstruct-7B-v0.2

metric, computed as follows:

$$\text{Accuracy} = \frac{1}{|D|} \sum_{(x,y_c,y_r)\in D} I[R_\theta(x, y_c) > R_\theta(x, y_r)] \quad (12)$$

where $I(\cdot)$ is the indicator function, and D denotes the evaluation dataset.

For token-level evaluation, we use tuples of the form $(x, y_{<t}+a_c, y_{<t}+a_r)$, where $y_{<t}$ is the generated tokens before, a_c is the next chosen token, and a_r is the rejected token. Similarly, we compute accuracy as the evaluation score: if the PRM assigns a higher reward to a_c than to a_r, the prediction is correct; otherwise, it is incorrect.

MT-PRMBench. MT-PRMBench comprises two distinct subsets for evaluation: *Token-level* and *Sequence-level*. The *Token-level* subset is designed for assessing preferences between immediate next-token candidates that follow an identical input prefix. In contrast, the *Sequence-level* facilitates the comparison of entire generated sequence completions that also originate from a shared input prefix.

4.2 Evaluation Results

Token-level Performance. From Table 1, we can observe that our MT-PRM-LLaMA-3.2-3B and MT-PRM-Qwen-2.5-3B models achieved accuracies of 0.578 and 0.66 respectively on the token-level MT-PRMBench. As shown in Table 2, we systematically compare models trained with vanilla sequence-level preference pairs versus our token-level preference pairs, while evaluating both DPO and KTO training objectives. The results demonstrate that token-level preference pairs significantly improve discrimination accuracy: implicit PRMs trained with token-level preference pairs outperform vanilla sequence-level baselines by +8.6% (DPO) and +11.5% (KTO). This performance gap highlights the critical advantage of token-level preference pairs in helping capture fine-grained translation quality distinctions.

Sequence-level Performance. We also convert our PRMs to sequence-level scoring through weighted DPO rewards (as shown in Eq. 11). We can observe that our MT-PRM-Qwen-2.5-3B achieves the highest performance among all models in the Prefixed set, with an average score of 0.863, outperforming both Skywork-Reward-LLaMA-3.1-8B [2] and the MT-Ranker (Moosa et al., 2024) variants. This demonstrates the effectiveness of our token-level supervision framework even when adapted to sequence-level scoring.

[2] https://huggingface.co/Skywork/Skywork-Reward-Llama-3.1-8B

5 Analysis and Practical Insight

5.1 Modeling Advantage versus Value as the PRM Training Signal

We have explored the impact of vanilla preference pairs versus MCTS-generated token-level preference pairs on the performance of implicit PRMs in the previous experiments (Table 2). This section shifts focus to the nature of the supervisory signal used for training PRMs. Specifically, we analyze our choice of implicit reward modeling—derived from token-level preference pairs (termed "Supervised by Preference (SP)")—against the alternative of directly using MCTS-backpropagated values $V(a)$ as a dense supervisory signal (termed "Supervised by Value (SV)").

Table 3 demonstrates that SP yields markedly superior performance in token-level discrimination, where SP achieved an average accuracy of 0.66, while SV achieved 0.52. This significantly higher accuracy underscores the effectiveness of our implicit PRM when trained with preference-based supervision. The SV approach, directly regressing on MCTS-backpropagated $V(a)$ values, tasks the PRM with learning a value function. However, $V(a)$ as a dense, token-level reward faces limitations: Monte Carlo estimates can be noisy, and the absolute value of a partial translation may offer an indirect and insufficiently discriminative signal for the most recent token's quality, especially in complex MT scenarios or with imperfect rollouts—a recognized challenge in process supervision literature (Lightman et al., 2024). We also find that these $V(a)$ values often clustered within a narrow numerical range (e.g., 0.7-0.8) further illustrates why this SV approach struggles. Such clustering severely hampers a regression model's ability to discern fine-grained distinctions, as the supervisory signal becomes subtle.

In contrast, the SP approach, using DPO/KTO objectives, excels by directly learning a representation reflecting the *advantage* of one token choice over another. This implicitly shapes a reward function where the per-step process reward, $r_\theta^t = \beta \log \frac{\pi_\theta(y_t|y_{<t})}{\pi_{ref}(y_t|y_{<t})}$ (Eq. 9), quantifies the localized, step-wise *advantage* of selecting token y_t. This preference-based optimization strategy is well-supported by prior work (Yuan et al., 2024; Rafailov et al., 2024a; Wang et al., 2024a). Crucially, DPO's mechanism, focusing on the log-

MT-PRMBench (Token-level)	DE-EN	RU-EN	ZH-EN	EN-DE	EN-RU	EN-ZH	Avg.
Supervised by Value (SV)	0.52	0.56	0.48	0.50	0.51	0.55	0.52
Supervised by Preference (SP)	0.64	0.74	0.68	0.58	0.68	0.66	0.66

Table 3: Experimental results of two training methods on the Token-level Benchmark

Source	在橘红色背景墙映衬下格外鲜亮的广告牌上，它们各自标出了不菲的价码。
Reference	On the bright advertising board against a tangerine colored wall, they listed their respective exorbitant price tags.
Translation A	Against an orange background, the **bright** billboards listed their respective, **hefty** prices.
Translation B	Against an orange background, the **dark** billboards listed their respective, **hefty** prices.
Translation C	Against an orange background, the **bright** billboards listed their respective, **bargain** prices.

Translation A	'Against'	' an'	…	' the'	**' bright'**	' bill'	…	**' hefty'**	' prices'	**Weighted Implicit Rewards(↑)**	**COMETKiwi(↑)**
Reward	-2.3	-2.3	…	0.07	1.28	-2.3	…	0.03	0.09	-3.43	0.80
Translation B	'Against'	' an'	…	' the'	**' dark'**	' bill'	…	' hefty'	' prices'	**Weighted Implicit Rewards(↑)**	**COMETKiwi(↑)**
Reward	-2.3	-2.3	…	0.07	-2.3	-2.3	…	0.09	0.09	-4.13	0.73
Translation C	'Against'	' an'	…	' the'	' bright'	' bill'	…	**' bargain'**	' prices'	**Weighted Implicit Rewards(↑)**	**COMETKiwi(↑)**
Reward	-2.3	-2.3	…	0.07	1.28	-2.3	…	-0.95	-2.3	-3.99	0.78

Table 4: Case study illustrating token-level credit assignment by our Qwen-PRM.

probability ratio between sequences (Eq. 4), is inherently more sensitive to relative differences than absolute value regression. This makes it adept at capturing preferences even when underlying $V(a)$ scores from the SV context are close, explaining its superior performance in training PRMs for nuanced token-level discrimination in our MT setting.

5.2 Per-token Credit Assignment

Our PRM can identify token-level translation differences, with the per-step process reward defined as Eq. 9 quantifying the reward for generating token y_t at step t. These individual token rewards are then aggregated into a final weighted sequence score (Eq. 11), allowing for a comprehensive evaluation that originates from fine-grained assessments.

Table 6 provides a case study illustrating these capabilities. For instance, in Translation B, the token "dark", which semantically contradicts the source "鲜亮" (bright), receives a significantly negative reward (-2.3) from our PRM. In Translation C, the incorrect token "bargain", used where "不菲的" (exorbitant/costly) is implied, is substantially penalized (-0.95). In contrast, contextually appropriate tokens in Translation A, such as "bright" (1.28) and "hefty" (0.03), secure relatively higher rewards. Furthermore, the final weighted sequence scores computed by our PRM demonstrate strong alignment with automatic metrics like COMETKiwi. Translation A, the highest quality hypothesis (COMETKiwi 0.80), also achieves the most favorable weighted PRM score (-3.43). This case study thus substantiates our PRM's effective token-level credit assignment and the consistency of its fine-grained assessments with established sequence-level quality metrics.

5.3 Test-time Alignment

Test-time alignment, also known as decoding-time alignment (Huang et al., 2024; Rashid et al., 2024), refers to the process of adjusting an LM's output during inference to better align with human preferences, without additional training or fine-tuning. Its application in MT remains underexplored.

In the context of MT, given the prior context $s_{<t}$ and timestamp t, we define the reward-guided scoring function for a candidate token a as:

$$s(a, s_{<t}) = \mathrm{LM}(a \mid s_{<t}) + w \cdot P(r([s_{<t}, a])) \tag{13}$$

where $\mathrm{LM}(a \mid s_{<t})$ represents the LM's predicted probability for token a given the preceding context $s_{<t}$. $r([s_{<t}, a])$ denotes the reward signal for token a, conditioned on the prior context $s_{<t}$. The softmax function is applied over the reward signal $r([s_{<t}, a])$, computed over the top k candidate tokens (with k being a window size), normalizing the reward value, which we label as $P(r([s_{<t}, \mathrm{a}]))$. The scaling factor w adjusts the relative weight of the reward signal, allowing it to contribute effectively without overpowering the LM's probability. Compared to standard decoding strategies, this approach offers a more refined scoring function, as it encourages the generated text to: 1) Maintain semantic coherence and relevance with the prior context, and 2) Align more closely with reward-based criteria and human preferences. Test-time alignment also substantially reduces the need for the extensive resources typically required for LM alignment training.

We use Qwen2.5-14B-Instruct[3] for generating tokens and leverage MT-PRM-LLaMA-3.2-3B and

[3] https://huggingface.co/Qwen/Qwen2.5-14B-Instruct

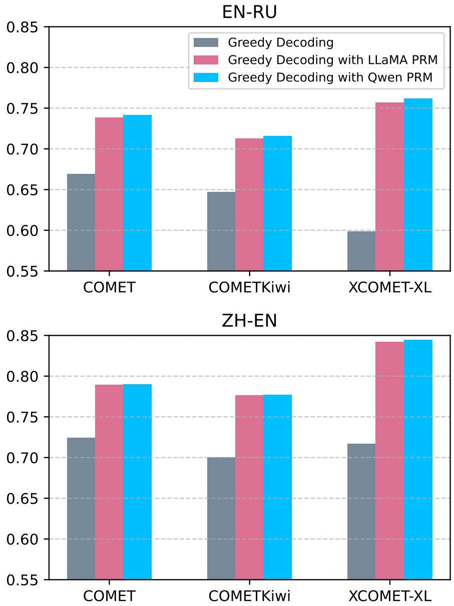

Figure 3: Results of test-time alignment across WMT 23 ZH-EN and EN-RU. MT-PRMs with less parameters can assist in aligning Qwen-2.5-14B-Instruct.

MT-PRM-Qwen-2.5-3B as the models for providing token-level rewards. We randomly sample 500 cases from the WMT 2023 testset. As shown in Figure 3, the reward-guided decoding methods outperform the standard greedy decoding in both EN-RU and ZH-EN translation tasks, evaluated by the COMET (Rei et al., 2020), COMETKiwi (Rei et al., 2022), and XCOMET-XL (Guerreiro et al., 2024) metrics. For instance, using the XCOMET-XL metric, LLaMA PRM and Qwen PRM outperform the standard greedy decoding by 17.5% and 17.9% in the EN-RU task respectively. Additionally, Qwen PRM slightly outperforms LLaMA PRM in both translation tasks and across all metrics, which aligns with the results in Table 1, where Qwen PRM achieves better token-level reward performance. These findings highlight the effectiveness of reward-guided decoding strategies in improving MT outcomes.

6 Related Work

Token-Level Feedback Mechanisms. Fine-grained feedback has been recognized for its ability to help models capture potential errors more precisely (Lightman et al., 2024). In the context of mathematical reasoning, process supervision using Monte Carlo methods has shown significant promise (Wang et al., 2024b; Qi et al., 2024; Guan et al., 2025). Furthermore, developments in general-domain have demonstrated that DPO can implicitly learn token-level rewards through policy optimization, a process referred to as implicit reward learning (Rafailov et al., 2024a; Wang et al., 2024a; Yuan et al., 2024). Despite these advancements, these approaches have yet to be tested in the context of MT. The translation community has long acknowledged the value of granular feedback, with early attempts relying on binary error markings from human annotations (Kreutzer et al., 2020), reference-based heuristics (Petrushkov et al., 2018), or LLM (Feng et al., 2024b).

Alignment Paradigms in Machine Translation. Alignment techniques in neural machine translation have evolved from Minimum Risk Training (Shen et al., 2015) to more sophisticated reinforcement learning approaches (Dang et al., 2024). While PPO-based RLHF has achieved success in general-domain alignment, its application to MT presents unique challenges, particularly due to the need for fine-grained quality signals rather than the bandit reward. Recent works like He et al. (2024) and Xu et al. (2024) have investigated the use of automatic metrics to select better translations or construct preference pairs to improve the LLM, while Zhao et al. (2024) explored scaling test-time compute to further enhance translation performance. Recently, Ramos et al. (2024) pioneered the use of xCOMET as a dynamic reward signal during RL training. However, these methods remain limited to sequence-level guidance or binary approximations of the reward process, failing to provide the fine-grained token-level feedback required for more accurate translation alignment.

7 Conclusion

In this work, we propose MT-RewardTree, a comprehensive framework for constructing, evaluating, and deploying process reward models in MT. Our framework leverages an automatic token-level preference pair generation approach inspired by approximate Monte Carlo Tree Search, effectively addressing the challenge of large-scale fine-grained supervision annotation. Extensive experiments on both sequence-level and token-level benchmarks demonstrate that our MT-PRM achieves advanced performance in reward modeling in MT, surpassing traditional sequence-level preference pairs. Our exploration of token credit assignment and test-time alignment provide valuable insights for the application of reward models in MT.

Limitations

Although we have developed the first comprehensive framework for process reward models in the field of machine translation, several important challenges remain to be addressed. Our work primarily focuses on synthesizing token-level data to leverage its fine-grained benefits. However, methods like Token-level DPO, RTO which optimize training algorithms, also show promise in further improving PRM performance. Additionally, our current framework includes only a limited set of high-resource languages, and expanding to multilingual settings, especially for low-resource languages, is a crucial direction for future work. While we have demonstrated the potential applications of reward models in test-time alignment and hypothesis ensembling, their integration into reinforcement learning training remains an important area for exploration.

References

Sweta Agrawal, José G. C. De Souza, Ricardo Rei, António Farinhas, Gonçalo Faria, Patrick Fernandes, Nuno M Guerreiro, and Andre Martins. 2024. Modeling user preferences with automatic metrics: Creating a high-quality preference dataset for machine translation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 14503–14519, Miami, Florida, USA. Association for Computational Linguistics.

Ralph Allan Bradley and Milton E. Terry. 1952. Rank analysis of incomplete block designs. *Biometrika*.

Meng Cao, Lei Shu, Lei Yu, Yun Zhu, Nevan Wichers, Yinxiao Liu, and Lei Meng. 2024. Enhancing reinforcement learning with dense rewards from language model critic. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9119–9138, Miami, Florida, USA. Association for Computational Linguistics.

Alex James Chan, Hao Sun, Samuel Holt, and Mihaela van der Schaar. 2024. Dense reward for free in reinforcement learning from human feedback. In *Forty-first International Conference on Machine Learning*.

Ruizhe Chen, Xiaotian Zhang, Meng Luo, Wenhao Chai, and Zuozhu Liu. 2024. Pad: Personalized alignment of llms at decoding-time. *arXiv preprint arXiv:2410.04070*.

Marta R Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, et al. 2022. No language left behind: Scaling human-centered machine translation. *arXiv preprint arXiv:2207.04672*.

John Dang, Arash Ahmadian, Kelly Marchisio, Julia Kreutzer, Ahmet Üstün, and Sara Hooker. 2024. RLHF can speak many languages: Unlocking multilingual preference optimization for LLMs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 13134–13156, Miami, Florida, USA. Association for Computational Linguistics.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*.

Zhaopeng Feng, Ruizhe Chen, Yan Zhang, Zijie Meng, and Zuozhu Liu. 2024a. Ladder: A model-agnostic framework boosting LLM-based machine translation to the next level. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 15377–15393, Miami, Florida, USA. Association for Computational Linguistics.

Zhaopeng Feng, Yan Zhang, Hao Li, Wenqiang Liu, Jun Lang, Yang Feng, Jian Wu, and Zuozhu Liu. 2024b. Improving llm-based machine translation with systematic self-correction. *arXiv preprint arXiv:2402.16379*.

Markus Freitag, George Foster, David Grangier, Viresh Ratnakar, Qijun Tan, and Wolfgang Macherey. 2021. Experts, errors, and context: A large-scale study of human evaluation for machine translation. *Transactions of the Association for Computational Linguistics*, 9:1460–1474.

Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. 2025. rstar-math: Small llms can master math reasoning with self-evolved deep thinking. *arXiv preprint arXiv:2501.04519*.

Nuno M Guerreiro, Ricardo Rei, Daan van Stigt, Luisa Coheur, Pierre Colombo, and André FT Martins. 2024. xcomet: Transparent machine translation evaluation through fine-grained error detection. *Transactions of the Association for Computational Linguistics*, 12:979–995.

Zhiwei He, Xing Wang, Wenxiang Jiao, Zhuosheng Zhang, Rui Wang, Shuming Shi, and Zhaopeng Tu. 2024. Improving machine translation with human feedback: An exploration of quality estimation as a reward model. *arXiv preprint arXiv:2401.12873*.

James Y Huang, Sailik Sengupta, Daniele Bonadiman, Yi-an Lai, Arshit Gupta, Nikolaos Pappas, Saab Mansour, Katrin Kirchhoff, and Dan Roth. 2024. Deal: Decoding-time alignment for large language models. *arXiv preprint arXiv:2402.06147*.

Levente Kocsis and Csaba Szepesvári. 2006. Bandit based monte-carlo planning. In *Machine Learning: ECML 2006*, pages 282–293, Berlin, Heidelberg. Springer Berlin Heidelberg.

Julia Kreutzer, Nathaniel Berger, and Stefan Riezler. 2020. Correct me if you can: Learning from error corrections and markings. *arXiv preprint arXiv:2004.11222*.

Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, et al. 2024. Rewardbench: Evaluating reward models for language modeling. *arXiv preprint arXiv:2403.13787*.

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*.

Yantao Liu, Zijun Yao, Rui Min, Yixin Cao, Lei Hou, and Juanzi Li. 2024. Rm-bench: Benchmarking reward models of language models with subtlety and style. *arXiv preprint arXiv:2410.16184*.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, et al. 2024. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint arXiv:2406.06592*.

Ibraheem Muhammad Moosa, Rui Zhang, and Wenpeng Yin. 2024. Mt-ranker: Reference-free machine translation evaluation by inter-system ranking. *arXiv preprint arXiv:2401.17099*.

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.

Pavel Petrushkov, Shahram Khadivi, and Evgeny Matusov. 2018. Learning from chunk-based feedback in neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 326–331, Melbourne, Australia. Association for Computational Linguistics.

Zhenting Qi, Mingyuan Ma, Jiahang Xu, Li Lyna Zhang, Fan Yang, and Mao Yang. 2024. Mutual reasoning makes smaller llms stronger problem-solvers. *arXiv preprint arXiv:2408.06195*.

Rafael Rafailov, Joey Hejna, Ryan Park, and Chelsea Finn. 2024a. From r to q^*: Your language model is secretly a q-function. In *First Conference on Language Modeling*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024b. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

Miguel Moura Ramos, Tomás Almeida, Daniel Vareta, Filipe Azevedo, Sweta Agrawal, Patrick Fernandes, and André FT Martins. 2024. Fine-grained reward optimization for machine translation using error severity mappings. *arXiv preprint arXiv:2411.05986*.

Ahmad Rashid, Ruotian Wu, Julia Grosse, Agustinus Kristiadi, and Pascal Poupart. 2024. A critical look at tokenwise reward-guided text generation. *arXiv preprint arXiv:2406.07780*.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. Comet: A neural framework for mt evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702.

Ricardo Rei, Marcos Treviso, Nuno M Guerreiro, Chrysoula Zerva, Ana C Farinha, Christine Maroti, José GC De Souza, Taisiya Glushkova, Duarte Alves, Luísa Coheur, et al. 2022. Cometkiwi: Ist-unbabel 2022 submission for the quality estimation shared task. In *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 634–645.

Christopher D. Rosin. 2011. Multi-armed bandits with episode context. *Annals of Mathematics and Artificial Intelligence*, 61(3):203–230.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2015. Minimum risk training for neural machine translation. *arXiv preprint arXiv:1512.02433*.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, L. Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. 2017. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*.

Kimi Team. 2025. Kimi k1.5: Scaling reinforcement learning with llms.

Huaijie Wang, Shibo Hao, Hanze Dong, Shenao Zhang, Yilin Bao, Ziran Yang, and Yi Wu. 2024a. Offline reinforcement learning for llm multi-step reasoning. *arXiv preprint arXiv:2412.16145*.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024b. 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.

Violet Xiang, Charlie Snell, Kanishk Gandhi, Alon Albalak, Anikait Singh, Chase Blagden, Duy Phung, Rafael Rafailov, Nathan Lile, Dakota Mahan, et al. 2025. Towards system 2 reasoning in llms: Learning how to think with meta chain-of-though. *arXiv preprint arXiv:2501.04682*.

Haoran Xu, Amr Sharaf, Yunmo Chen, Weiting Tan, Lingfeng Shen, Benjamin Van Durme, Kenton Murray, and Young Jin Kim. 2024. Contrastive preference optimization: Pushing the boundaries of LLM performance in machine translation. In *Forty-first International Conference on Machine Learning*.

Lifan Yuan, Wendi Li, Huayu Chen, Ganqu Cui, Ning Ding, Kaiyan Zhang, Bowen Zhou, Zhiyuan Liu, and Hao Peng. 2024. Free process rewards without process labels. *arXiv preprint arXiv:2412.01981*.

Zhiyuan Zeng, Qinyuan Cheng, Zhangyue Yin, Bo Wang, Shimin Li, Yunhua Zhou, Qipeng Guo, Xuanjing Huang, and Xipeng Qiu. 2024. Scaling of search and learning: A roadmap to reproduce o1 from reinforcement learning perspective. *arXiv preprint arXiv:2412.14135*.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025. The lessons of developing process reward models in mathematical reasoning. *arXiv preprint arXiv:2501.07301*.

Yu Zhao, Huifeng Yin, Bo Zeng, Hao Wang, Tianqi Shi, Chenyang Lyu, Longyue Wang, Weihua Luo, and Kaifu Zhang. 2024. Marco-o1: Towards open reasoning models for open-ended solutions. *Preprint*, arXiv:2411.14405.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2024. Processbench: Identifying process errors in mathematical reasoning. *arXiv preprint arXiv:2412.06559*.

Han Zhong, Guhao Feng, Wei Xiong, Xinle Cheng, Li Zhao, Di He, Jiang Bian, and Liwei Wang. 2024. Dpo meets ppo: Reinforced token optimization for rlhf. *arXiv preprint arXiv:2404.18922*.

A Case Study of Test-Time Alignment

The case in Table 6 compares a ZH-EN translation using Greedy Decoding (GD), GD with LLaMA PRM, and GD with Qwen PRM. The standard GD translation, "The mechanic said the oil was clear, indeed good oil", conveying the basic meaning but lacking the important "changing oil" context. Both reward-guided decoding methods improve the translation to a more complete, high-quality version, achieving a COMETKiwi score improvement of 4.97%.

B Data Statistics

Translation Direction	Token-level Preference Pairs	
	Train	MT-PRMBench
DE-EN	1,255	200
EN-DE	2,059	200
RU-EN	1,219	200
EN-RU	1,711	200
ZH-EN	1,232	200
EN-ZH	1,176	200

Table 5: Data Statistics.

Source	换油的师傅说油品清亮，确实是好油。
Reference	The oil was changed by the master, who claimed that it was clean and good oil.
Greedy Decoding (GD)	The mechanic said the oil was clear, indeed good oil. COMETKiwi: 0.7779
GD with LLaMA PRM	The mechanic who changed the oil said that the oil is clear, indeed it is good oil. COMETKiwi: 0.8165
GD with Qwen PRM	The mechanic who changed the oil said that the oil is clear, indeed it is good oil. COMETKiwi: 0.8165

Table 6: Case study of test-time alignment.