5 Appendix

5.1 Method Details

5.1.1 Strategy for Sentence Segmentation

SaT [7] provides a range of models for predicting sentence boundaries. We select the `sat-3l` model as our base segmenter, as it offers a favorable balance between segmentation quality and inference time. However, the output of the `sat-3l` segmenter may still include excessively long sentences. Following the approach in [10], we implement a supplementary step: we set a maximum character limit for a single sentence, and any segments exceeding this limit are re-segmented using a rule-based approach. The segmentation markers used are as follows: `["...","\n","!","?",";",":",".",",","\t"]`.

5.1.2 Case Studies of Sentence Segmentation in Code and Mathematics

In this work, we use the term "sentence" in an operational sense: it refers to a semantic unit produced by our segmentation pipeline rather than a strictly linguistic sentence boundary. In code, this unit is typically closer to a short executable step, such as variable initialization, a complete loop header, a state update, a conditional check, or a return statement. In mathematical reasoning, it is typically closer to a local derivation step, such as introducing notation, making an intermediate argument, performing one algebraic transformation, or drawing an intermediate conclusion.

This boundary ambiguity is also an important reason why rule-based segmentation is less effective in coding and reasoning scenarios. In code and mathematical expressions, semicolons, brackets, decimal points, member-access symbols, and LaTeX markers often serve syntactic or computational roles rather than semantic boundary indicators. We present a qualitative case study below.

- Coding Example.

Input code.

```
double ans = 0.0;
for (int i = 0; i < n; ++i) {
   ans += std::abs(a[i] - b[i]) /
      (i + 1.0);
   if (ans > 1e-6 &&
      checker.is_valid(ans)) return ans;
}
return ans;
```

Rule-based segmentation.

```
[1] double ans = 0.0;
[2] for (int i = 0;
[3] i < n;
[4] ++i) {
[5] ans += std::abs(a[i] - b[i]) /
   (i + 1.0);
[6] if (ans > 1e-6 &&
   checker.is_valid(ans)) return ans;
[7] }
[8] return ans;
```

sat-3l segmentation.

```
[1] double ans = 0.0;
[2] for (int i = 0; i < n; ++i) {
[3] ans += std::abs(a[i] - b[i]) /
   (i + 1.0);
[4] if (ans > 1e-6 &&
   checker.is_valid(ans)) return ans;
[5] }
[6] return ans;
```

Interpretation. In this example, the rule-based splitter mechanically breaks the `for (int i = 0; i < n; ++i)` loop header at semicolons, while `sat-3l` preserves it as a single coherent control unit, which better matches the local semantic structure of code.

5.1.3 Explorations for Sentence-level Reward Model

The idea of sentence-level reward model originates from our attempts to extend the Preference Transformer (PT) [11] approach to LLMs. Our initial exploration involves training a token-level reward model using a PT objective:

$$P(y^w > y^l|x) = \frac{\exp\left(\sum_{t=1}^{T} w_\phi(s_t^w, a_t^w) r_\phi(s_t^w, a_t^w)\right)}{\sum_{j\in\{w,l\}} \exp\left(\sum_{t=1}^{T} w_\phi(s_t^j, a_t^j) r_\phi(s_t^j, a_t^j)\right)},$$

where $w_\phi(s_t^j, a_t^j) = \frac{1}{T}\sum_{s=1}^{T} \mathrm{SoftMax}(q_t^j, k_s^j)$. $q_t^j = q_\phi(s_t^j, a_t^j)$ and $k_t^j = k_\phi(s_t^j, a_t^j)$ are obtained from the corresponding projection layer. While this token-level RM does not achieve superior performance compared to the response-level baseline, we make an interesting observation. Specifically, when analyzing the reward distribution of a PT-trained reward model, we find that tokens at sentence boundaries exhibit significantly higher weighted reward values, as is illustrated in Figure 3. This observation directly inspires us to attempt reward modeling at the sentence level.

5.1.4 RoPE for Attention-based Aggregate Function

To effectively model the positional relationships across different sentences, we integrate Rotary Positional Embeddings (RoPE, [12]) into the q_{n_c} and k_i. We provide a detailed explanation of the integration method of RoPE. As one of the most widely adopted positional encoding methods in LLMs, RoPE is implemented by applying a position-dependent rotation matrix to the query (q) and key (k) vectors within the attention block. This rotation matrix encodes the positional information directly into the representations. If a token x_i is located at position i in the sequence, and its corresponding query q_i and key k_i have a dimensionality of d, then the rotation matrix W_i^R to be applied is defined as follows:

$$W_i^R = \begin{bmatrix} \cos i\theta_0 & -\sin i\theta_0 & 0 & \cdots & 0 \\ \sin i\theta_0 & \cos i\theta_0 & 0 & \cdots & 0 \\ 0 & 0 & \cos i\theta_1 & -\sin i\theta_1 & \cdots \\ 0 & 0 & \sin i\theta_1 & \cos i\theta_1 & \cdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cos(i\theta_{d/2-1}) & -\sin(i\theta_{d/2-1}) \\ 0 & \cdots & 0 & \sin(i\theta_{d/2-1}) & \cos(i\theta_{d/2-1}) \end{bmatrix}$$

Here $\theta_t = 1000^{-\frac{2(t-1)}{d}}, t = 1, 2, \cdots, d/2$. The rotated $q_i^R = W_i^R q_i$ and $k_i^R = W_i^R k_i$ are then used to compute the dot product, yielding the corresponding attention weights. In our sentence-level reward model, two types of positional information are present: (1) the position of a sentence within the overall response, and (2) the position of sentence-boundary tokens within the input token sequence. We apply the RoPE operation based on the sentence-level positional information.

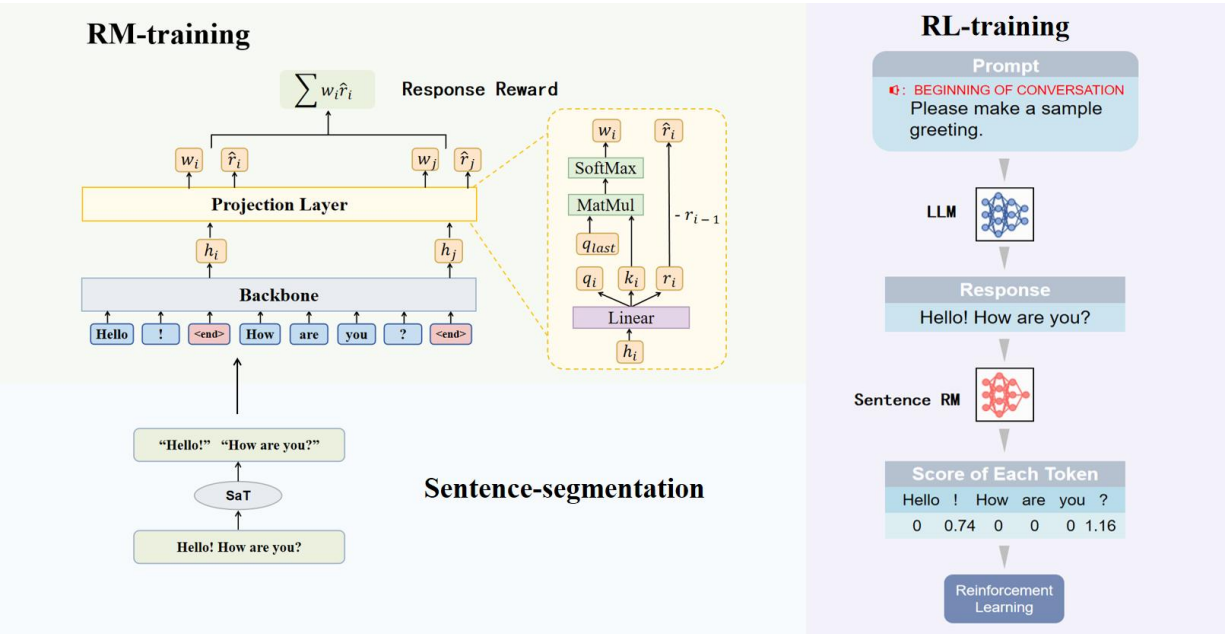


Figure 2 Framework of sentence-level reward modeling. Given an input response $(x, a_1, \cdots, a_T)$, a segmenter first identifies sentence boundaries. The projection layer then transforms the hidden states of these sentence boundary tokens into sentence-level rewards $\hat{r}_i$ and their corresponding weights w_i. The overall response-level reward is computed as a weighted sum of these sentence-level rewards. During RL training, the sentence-level reward model provides non-zero rewards at sentence boundaries, enabling the use of advanced RL algorithms for LLM optimization.

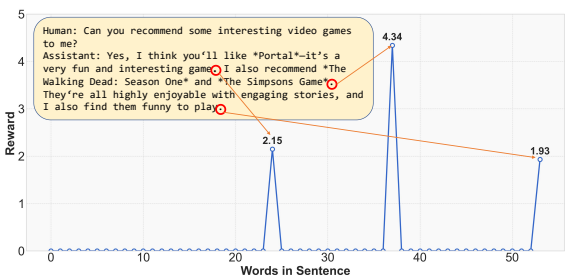


Figure 3 Reward distribution for token-level reward model trained with the PT objective. It is observed that tokens at sentence boundaries exhibit significantly higher weighted reward values.

5.1.5 Assigning Rewards to Sentence Boundary Tokens

Our sentence-level reward model operates on responses processed by the SaT model, which segments the text into sentences. After obtaining the segmentation results from SaT, we manually insert special `<END>` tokens to mark the boundaries between sentences. However, since the `<END>` tokens are not part of the policy's original output and are irrelevant to the optimization process, we need to identify the token positions immediately preceding each `<END>` token. These positions serve as the anchor points for reward assignments.

A straightforward approach is to segment the response and tokenize each sentence separately, allowing us to identify the last token as the sentence boundary. We then concatenate the tokenized sequences of all sentences as the final tokenization result for the entire response. However, we observe that under BPE-style tokenizer [13, 14], this approach leads to mismatches—specifically, the concatenation of individually tokenized sentences does not always match the result of tokenizing the entire sequence directly. A key challenge arises from the tokenizer's behavior of merging multiple characters into a single token. For example, multiple consecutive spaces may be encoded as a single token.

To address this, we leverage the tokenizer's `offset_mapping` feature, which maps each token back to its character-level position in the original text. We hope to construct a binary mask that aligns with the policy's token sequence. In this mask, `1` indicates a boundary position where rewards are assigned, while `0` indicates non-boundary positions with a zero reward.

The implementation involves the following two main steps:

1. **Tokenization and Boundary Identification**: The policy's original response (`source_text`) is tokenized using the tokenizer of policy model, with `offset_mapping` enabled to track character-level positions. Simultaneously, we feed the original response into the `sat-3l` for segmentation, and a special token `<END>` is appended to the end of each segmented sentence. These special tokens facilitate sentence-level RM inference and enable precise identification of sentence boundaries at the character level.

2. **Mask Generation**: Using the `offset_mapping`, we map the character-level boundary positions to their corresponding token indices in the policy's token sequence. A binary mask is then created, where `1` is assigned to tokens immediately preceding the identified boundaries, and `0` is assigned to all other tokens.

This approach ensures that the reward computation is accurately aligned with the sentence boundaries, even when the tokenizer merges multiple characters into a single token.

5.1.6 Reward Model Training without Introducing the Special Token

We introduce a special token to mark the end of a sentence, and the sentence-level reward model outputs the sentence reward at the special token's position. In fact, by using the method mentioned in Section 5.1.5, we can identify the sentence boundary token's position without introducing the special token. We train the sentence-level RM with this approach, then we follow the same RL experiment configurations as Section 3 to evaluate the performance. We report the win rate of the sentence-level reward model(without special token) compared to baselines and our method (with special token) in Table 2a. It was observed that, compared to the response-level and token-level reward model, it achieves a higher win rate against SFT model, indicating that sentence-level modeling can offer a performance over the response-level reward modeling. However, when compared to the model with the special token (<END>), its win rate show a slight drop, suggesting that the approach with the special token leads to better sentence-level reward model training.

Additionally, we also explore the impact of different aggregation functions on the performance of the reward model. We train the reward model using the aggregation functions described in Paragraph 5.4.4.1, with the key distinction that the <END> token is excluded in this experiment using the method metioned in Section 5.1.5. Notably, since using the aggregation function **DA** during training causes it to degrade into a response-level reward model, we don't conduct experiments for it. The trained reward models are then utilized for reinforcement learning. We evaluate the results on the AlpacaEval benchmark(judged by Deepseek-V3), computing the win rates by comparison with the Response. The detailed results are presented in Table 2b. The results reveals that the our proposed aggregation function (without <END>) achieves the best overall performance in terms of both win rate (WR) and length control win rate (LC).

Method	WR(%)	LC(%)
Response	71.55	55.97
Token	70.06	55.06
Ours (w.o. <END>)	74.66	60.31
Ours (w. <END>)	**78.39**	**60.32**

(a) Training sentence-level reward model (w.o. <END>), we evaluate the Win Rate (WR) and Length Control Win Rate (LC) against SFT. The performance of Sentence (w.o. <END>) surpasses that of Response and Token but falls behind Sentence (w. <END>).

Method	WR(%)	LC(%)
VA	73.91	57.44
VW	**75.78**	55.99
Ours (w.o. <END>)	74.66	**60.31**

(b) Using different aggregation functions (w.o. <END>), we evaluate the Win Rate (WR) and Length Control Win Rate (LC) compared to the Response. Our aggregation function achieves the best overall performance in terms of both WR and LC metrics.

5.1.7 Pseudo-code for Reward and Policy Training

A workflow is illustrated in Figure 2. The pseudo-code of sentence level RM and LLMs training is presented in Algorithm 1.

Algorithm 1 Reward and Policy Training

Input: Preference dataset $D_{\text{pref}} = \{(x^i, y^{w,i}, y^{l,i})\}_{i=1}^{N}$ for reward model training, a prompt dataset $D_{\text{prompt}} = \{x^i\}_{i=1}^{M}$ for policy training, a fine-tuned SFT model, the number of iterations M_{reward} for reward model training, the number of iterations M_{policy} for policy training, KL divergence coefficient β.

// Training the sentence reward model

Preprocess the preference dataset D_{pref} using the SaT model by inserting the <END> separator at the end of each segmented sentence and concatenating the sentences into complete responses. The processed dataset is denoted as D_{END}.

for iter $\in \{1, \ldots, M_{\text{reward}}\}$ **do**

  Sample a mini-batch $\mathcal{B} = \{(x^i, y^{w,i}, y^{l,i})\}_i \sim D_{\text{END}}$.

  Compute rewards using Equation (2).

  Optimize the reward model using Equation (3).

end for

// Training the policy with the sentence reward

for iter $\in \{1, \ldots, M_{\text{policy}}\}$ **do**

  Sample a mini-batch $\mathcal{B} = \{x^i\}_i \sim D_{\text{prompt}}$.

  Sample a response $y^i \sim \pi_\theta(\cdot|x^i)$ for each $x^i \in \mathcal{B}$.

  Process y^i using the SaT model to insert <END> separators, resulting in y^i_{split}.

  Compute sentence-level rewards $r_\phi(x^i, y^i_{\text{split}})$ using the reward model.

  Map the rewards of y^i_{split} to the corresponding tokens in y^i to obtain $\hat{r}(c_i)$ as described in Section 5.1.5.

  Optimize the policy model using the sentence-level rewards $\hat{r}(c_i)$ with Equation (4).

end for

5.2 Related Work

5.2.1 Reward Learning from Human Preference

In RL, learning reward models from human preference is a common approach to designing reward functions. [15] introduces the idea of learning a step-wise reward model from trajectory-level preference labels, and PT [11] employs a transformer-based architecture to construct a non-Markovian reward model.

With the widespread adoption of RLHF for LLMs alignment, significant research has focused on developing reward models for LLMs. These efforts can be broadly categorized into discriminative RMs and generative RMs [16]. Discriminative RMs are widely utilized in post-training stages of LLMs [17, 18], which project hidden states into reward scores with a linear head, and are trained through BT loss. To improve the performance of discriminative RMs, some studies have aimed at enhancing the quality of preference data [19, 20] or optimizing the model structure [21, 22]. Despite the popularity of discriminative RMs, generative RMs offer an alternative by leveraging the generative capabilities of the LLMs to evaluate the generated responses [23, 24]. The training process of generative reward models typically aligns with LLMs training, which allows it to improve the evaluation quality

through prompt design and Chain-of-Thought (CoT) [25] techniques. For instance, GenRM [26] combines the sequences to be evaluated with the question "`Is the answer right? (yes/no)`" as a prompt, using the probability of token `yes` as the reward score. Our work should be categorized as a discriminative reward model.

5.2.2 Fine-grained Reward Model

We consider the granularity of reward modeling. A common approach is to assign a reward score to an entire response. This coarse-grained method simplifies the reward modeling, but as discussed in Section 1, it can lead to sparse reward challenges during RL training. To address the issue, many studies have explored assigning a reward score to a token, leading to fine-grained models. For instance, [4] propose computing response rewards by averaging the summed token-level rewards, and train the reward model using the BT loss. Moreover, [3] use the attention weights from the trained response-level reward model to distribute response-level rewards to tokens. [27] and [28] train a policy model with DPO [29] to derive token-level rewards. In addition, [30] and [31] leverage advanced AI systems to annotate token-level preferences. [32] train a language model to modify sequences and construct discrete token rewards based on whether tokens are rewritten.

Our work focuses on assigning reward scores to sentences. The most related approach is the Process Reward Model (PRM) [33], which assigns rewards to steps in the problem-solving process and is often used to enhance the reasoning capabilities of LLMs. However, constructing PRMs typically requires manual annotation [33] or tree search [18, 34] to label the correctness of each step. These approaches are both resource-intensive and challenging, especially for tasks where step correctness is ambiguous. In contrast, our work relies on the response-level preference labels.

5.2.3 Segment-level Reward Model

[9] proposes a segment-level reward modeling approach with a granularity similar to ours, but differing in key details. In conception, segment is implementation-driven and often breaks semantic units. Sentence is a semantic unit with a natural boundary and inherent completeness. Our Sentence RM utilizes this linguistic prior to ensure that the reward is assigned to a complete semantic block, thus balancing granularity and semantic coherence more effectively than heuristic segment. In practical implementation, first, it performs entropy-based segmentation by forwarding an additional LLM sharing the RM's tokenizer, introducing additional computational overhead. This also hinders its applicability in RL training for LLMs with different tokenizers, due to the need for the same tokenizer during segmentation. Moreover, the entropy-based method requires careful tuning of the segmentation threshold, which has a significant impact on segmentation quality. In contrast, our method leverages a lightweight 0.8B SAT model, enabling efficient and tokenizer-agnostic segmentation. Empirically, we also observe that sentences segmented by SaT are more semantically coherent, whereas entropy-based methods often produce fragmented segments consisting of isolated token. Due to the property of BPE-style tokenizer [14], the isolated token may lack any explicit semantic information. Second, while segment-level RM aggregates rewards by averaging over segments, we adopt an attention-based aggregation strategy, which is more flexible and expressive. Our approach consistently outperforms segment-level RM in RLHF experiments.

Hyperparameter	Value
Global Train Batch Size	256
Micro Train Batch Size	4
Train Epochs	1
Max Total Length	1024
DeepSpeed ZeRO stage	3
Optimizer	Adam
Learning Rate	3e-6
Weight Decay	0.2
Scheduler Type	Cosine
Gradient Clipping Norm	1.0
Dimension of q	256
Dimension of k	256
Offload	True

(a) The hyperparameter settings of SFT and reward model training.

Hyperparameter	Value
Global Train Batch Size	128
Micro Train Batch Size	8
Global Rollout Batch Size	1024
Micro Rollout Batch Size	16
Train Epochs	1
Max Total Length	1024
Max Prompt Length	512
Max Generation Length	512
DeepSpeed ZeRO Stage	2
Actor learning Rate	5e-7
Optimizer	Adam
Gradient Clipping Norm	1.0
KL Coefficient	0.01

(b) The hyperparameter settings of Reinforce++ in RL training.

5.3 Experiments Configurations

5.3.1 Detailed Experiment Settings

- Dataset

For reward model training, we use the Ultrafeedback [19] dataset, comprising 64k prompts and 128k responses with preference labels. For the alignment experiments, we utilize the HH-Golden dataset, a processed version of Anthropic's Helpful and Harmless (HH) dataset [35], comprising 42.5k high-quality prompts and corresponding responses. Specifically, following the methodology of DeepSpeed-Chat [36], the HH-Golden dataset is split into two equally sized subsets (5:5 ratio) used for Supervised Fine-Tuning (SFT) and RLHF, respectively. In the RL phase, only the prompts from the allocated subset are used for response generation.

- Models and Training

We use `sat-3l` [7] as the segment models. For reward model training, we use Llama3.1-8B [1] as the base model. For the RL experiments, we perform SFT on Llama3.2-3B, and then use the SFT model as the initial model. In addition, to verify the scaling effect, we also conduct RL experiments using Qwen2.5-3B [37], Llama3.1-Tulu-3-8B-SFT and Mistral-7B [38]. For the BoN experiments, we generate responses using Llama-3.1-8B-Instruct. The reward model training framework is based on DeepSpeed-Chat, and the RL training framework is built on Open-RLHF.

- Evaluation

We evaluate the performance of our reward model on RewardBench [39], RMBench [40], JudgeBench [41] and RewardBench2 [42]. These benchmarks are primarily composed of pairwise comparison datasets, where each instance consists of a prompt followed by two labeled candidate responses. To ensure a robust evaluation across diverse capabilities, these benchmarks categorize pairwise data into several distinct

domains. Reward model performance is measured by its accuracy in predicting the ground truth preference labels. For our sentence-level reward model, evaluation on RewardBench uses the aggregated response reward derived from the aggregation function in Equation (2).

To evaluate the alignment performance of LLMs fine-tuned with reward models, we utilize the AlpacaEval [43] and Arena-Hard [44] benchmarks. AlpacaEval comprises 805 prompts, and Arena-Hard provides 500 prompts. Responses generated are compared by a judge model. We use DeepSeek-V3 [45] and GPT-5 as the judge model and report the win rate of LLMs fine-tuned against the initial SFT model. In addition, we conduct Best-of-N (BoN, [46]), a common inference-time alignment method. We report the win rate of BoN policy on AlpacaEval and Arena-Hard against greedy decoding.

- Baselines

For comparison, we employ four reward models: the response-level reward model [46], the token-level reward model [4] the segment-level reward model [9] and the generative reward model [26]. For brevity, we refer to them as **Response**, **Token**, **Segment**, and **GenRM** respectively. While the first three models typically output scalar values based on specific granularities, GenRM utilizes a discrete evaluation approach. Specifically, we prompt the Llama3.1-8B [1] to perform a pairwise comparison between a response generated by the policy model and a reference response sampled from the initial policy model. If the model determines the policy-generated response is superior, it receives a reward of 1; otherwise, the reward is 0. In addition, we add DPO [29] on HH-Golden dataset as baseline for RL experiment. For the RL experiments, we define two additional baselines based on different reward signal granularities:

- Response to Sentence (**Res2Sent**): This baseline adapts a trained response-level reward model to provide sentence-level signals. The reward for sentence is computed as the difference between the model's output at the sentence end position and its output at the start position.
- Sentence to Response (**Sent2Res**): This baseline uses our trained sentence-level reward model to provide a single response-level score. The reward for response is aggregated by Equation (2).

5.3.2 Parameter Configurations

Our reward model training pipeline is initialized with the Llama3.1-8B model [1] as its foundation. The process commences with supervised fine-tuning (SFT) on the UltraFeedback dataset to tailor the base model for our specific task requirements. Following the SFT phase, we proceed with reward model training, using the fine-tuned SFT model as the starting point. The comprehensive hyperparameter configurations for both the SFT and reward model training stages are systematically presented in Table 3a.

For experiments about reward model training, they are conducted on a server outfitted with AMD-EPYC9374F-32-Core Processor, 8 GeForce RTX 4090 GPUs, running Pop OS 22.04 LTS. For experiments about RL training, they are conducted on a server outfitted with 13th GenIntel(R)Core(TM)i9-13900K CPU, 4 NVIDIA A100 GPUs, runing Ubuntu 22.04.

5.3.3 BoN

We implement Best-of-N (BoN) sampling to evaluate the performance of our reward models in inference-time alignment. Using the Llama-3.1-8B-Instruct model on the AlpacaEval benchmark [43], we generate response candidates with stochastic sampling parameters: temperature (τ = 0.6), top-k filtering (top-k = 50), and nucleus sampling (top-p = 0.9). For each prompt, we collect 32 independent responses. To analyze performance progression, we hierarchically select the best response by applying our trained reward model to subsets of $N \in \{2, 4, 8, 16, 32\}$ candidates, retaining the highest-scoring candidate for each N configuration.

The selected responses are compared against generations from the same model using greedy decoding (τ = 0, do_sample = False).

5.3.4 Prompt for Judge Model

We adopt the DeepSeek-V3 and GPT-5 as our automated judge model. The evaluation prompt is shown as below.

Prompt used in AlpacaEval

```
<|im_start|>system
You are a helpful assistant, that ranks models
 by the quality of their answers.
<|im_end|>

<|im_start|>user
I want you to create a leaderboard of
different of large-language models. To do so,
I will give you the instructions (prompts)
given to the models, and the responses of
two models. Please rank the models based on
which responses would be preferred by
humans. All inputs and outputs should be
python dictionaries.

Here is the prompt:
{
   "instruction": """{instruction}""",
}

Here are the outputs of the models:
[
   {
      "model": "model_1",
      "answer": """{output_1}"""
   },
   {
      "model": "model_2",
      "answer": """{output_2}"""
   }
]

Now please rank the models by the quality of
their answers, so that the model with
rank 1 has the best output. Then return a list
 of the model names and ranks, i.e.,
produce the following output:
[
```

```
    {'model': <model-name>, 'rank': <model-rank
    >},
    {'model': <model-name>, 'rank': <model-rank
    >}
]

Your response must be a valid Python
dictionary and should contain nothing else
because we will directly execute it in Python.
 Please provide the ranking that the
majority of humans would give.
<|im_end|>
```

5.3.5 RL Training

In our reinforcement learning (RL) experiments, we maintain reproducibility by fixing the random seed to 1234 across all trials. The comprehensive hyperparameter configurations for RL training are systematically detailed in Table 3b.

We employ the AdamW optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.95$, augmented by three advanced memory optimization techniques: (1) ZeRO-2 stage optimization [47] for distributed parameter management, (2) flash attention mechanisms [48] for efficient attention computation, and (3) gradient checkpointing with CPU offloading to maximize GPU memory utilization.

The RL training protocol is configured with a fixed learning rate of 5×10^{-7}, micro batch sizes of 8 (training) and 16 (rollout), global batch sizes of 128 (training) and 1024 (rollout), and a uniform KL divergence penalty coefficient of 0.01 across all reward models.

For evaluation on the AlpacaEval benchmark [43], we use generation parameters including a temperature of 1.0, top-p sampling of 1.0, disabled top-k sampling, and a maximum of 512 new tokens.

5.4 More Experiment Results

5.4.1 Main Results

Figure 4 presents the performance of our sentence-level reward model and baseline models on four reward model benchmarks. Across the subtasks of rewardbench, the sentence-level reward model consistently outperforms the token-level and response-level reward models. On average, our sentence-level model surpasses the second-best baseline (segment-level) by 1.8%. Particularly, on the reasoning subtask of rewardbench, our sentence-level reward model significantly outperforms other reward models, leading the response-level and token-level models by 7.1% and 4.4%, respectively. Under identical conditions regarding the base model and training data, the above results demonstrate the improved generalization capability of our proposed sentence-level reward model to unseen preference datasets.

Table 4 (Judged by Deepseek-V3) and Table 6 (Judged by GPT-5) show the performance of LLMs fine-tuned with various reward models on AlpacaEval and Arena-Hard. All win rates are reported relative to the initial SFT model. Based on them, we have the following key observations:

(1) **RL training can benefit from sentence-level reward signals**. For the same sentence-level reward model, training using direct sentence-level rewards yields higher performance compared to training with the aggregated response score (Sent2Res). And LLMs trained with sentence-level reward model consistently outperform those trained with the response-level reward model. This indicates that incorporating sentence-level reward signals significantly enhances the effectiveness of RL training.

(2) **The sentence-level reward model exhibits greater robustness against length bias**. Both sentence-level and token-level reward models offer fine-grained signals for RL training. However, LLMs trained with token-level rewards are highly susceptible to length hacking — they tend to generate significantly longer responses than other methods and exhibit a noticeable decline in win rates under length-controlled evaluations on AlpacaEval. In contrast, our sentence-level reward model effectively guides the model to enhance generation quality while preserving controlled response length.

(3) **The response-level reward model struggles to provide accurate reward signals for sentences**. We applying similar differential operations for the response-level reward model to derive sentence-level rewards (Res2Sent). The performance of LLMs trained with Res2Sent lags behind those trained with the direct response reward. It's inferred response-level reward models provide inaccurate reward signals for sentences, making it difficult for RL training to benefit from it.

Figure 6a and Figure 6b illustrate the performance of various reward models with BoN. We vary N from 2 to 32 and report the win rate against greedy decoding. The sentence-level reward model consistently outperforms the token-level, segment-level and response-level reward models across different candidate responses. Notably, when $N = 32$, our method achieves a 7% performance improvement over the baselines on AlpacalEval. These results highlight the potential of utilizing sentence-level reward models to enhance inference-time alignment strategies. Besides, we provide the detailed reward model performance in Table 7.

5.4.2 Training Efficiency Improvements

We investigate whether sentence-level reward signals improve RL training efficiency. Figure 6c presents RL training curves of Llama-3.1-Tulu3-8B-SFT comparing two approaches that utilize signals from the same sentence-level reward model: one uses the aggregated response score for coarse-grained training, and the other uses direct sentence-level rewards for fine-grained training. Performance is evaluated using the aggregated response reward in Equation (2). As depicted in Figure 6c, training with direct sentence-level rewards yields substantially higher evaluation scores within the same training duration. This indicates that fine-grained sentence-level reward signals facilitate more efficient sample utilization, leading to improved training efficiency.

5.4.3 Reward visualization

Figure 5 illustrates the reward values assigned by different reward models. Color intensity indicates reward magnitude, with darker shades representing higher values. As shown, the response-level reward model assigns a uniformly high score across the entire generated response, even for segments with relatively weak relevance to the prompt(e.g., the third sentence). In contrast, the sentence-level reward model demonstrates a more refined understanding of semantics. It can effectively

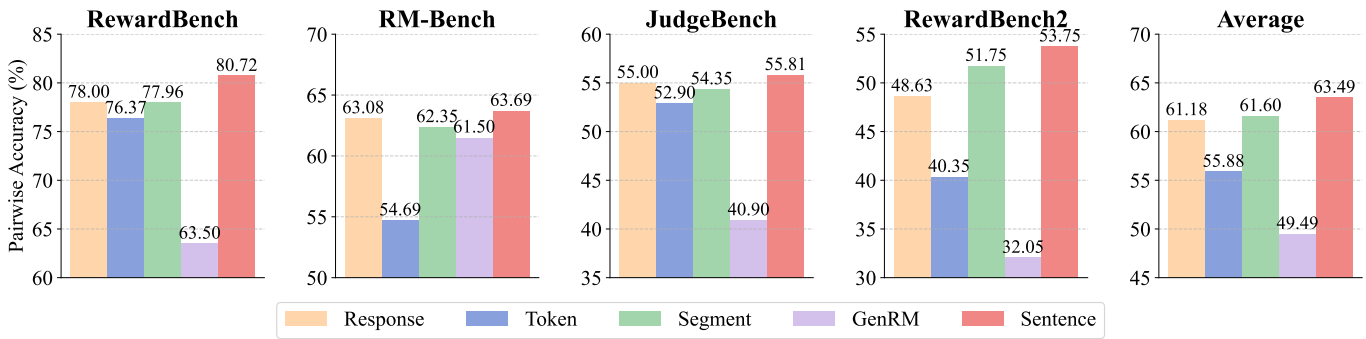


Figure 4 **The prediction accuracy of the reward models on Reward Model Benchmark**. Our sentence level reward model outperforms the token-level, response-level, segment-level and generative reward models on all benchmarks.

Prompt: I'm struggling with my college courses and considering dropping out. What should I do?

Sentence	Identify the courses causing youtrouble first . Meeting with your advisor andprofessors can lead to effective solutions . Still , school often feels dull andfrustrating with its tedious routines .
Token	Identify the courses causing youtrouble first . Meeting with your advisor andprofessors can lead to effective solutions . Still , school often feels dull andfrustrating with its tedious routines .
Response	Identify the courses causing youtrouble first . Meeting with your advisor andprofessors can lead to effective solutions . Still , school often feels dull andfrustrating with its tedious routines .
Segment	Identify the courses causing youtrouble first. Meeting with your advisor andprofessors can lead to effective solutions . Still, school often feels dull andfrustrating with its tedious routines.

Figure 5 **Reward distribution analysis** for different models on an educational advice response. Our sentence-level model (top) selectively rewards actionable suggestions (sentences 1-2, dark shading) while penalizing unconstructive complaints (sentence 3). Token-level rewards (middle) lack clear semantic interoperability and response-level scoring (bottom) provides uniform evaluation. Color intensity corresponds to reward magnitude.

assess the relevance of each sentence to the prompt, assigning higher rewards to strongly aligned content and penalizing sentences that diverge from the intended meaning. We believe this contributes to the sentence-level reward model's greater robustness against length bias. While the token-level reward model also provides fine-grained signals, our analysis suggests these signals often lack a robust and consistent correlation with the underlying semantic meaning, potentially leading to inaccurate reward signals.

5.4.4 Ablation Studies

- The Impact of Aggregation Functions

The aggregation function employed in our method introduces an inductive bias into the training process, which has a significant impact on the overall performance. We investigate the impact of difference aggregation functions on performance. The specific forms of the aggregation functions are listed below:

- **VW** : $r_\phi(x, y) = \sum_{i=1}^{n_c} w_i r_\phi(x, c_1, \cdots, c_i)$,
- **VA** : $r_\phi(x, y) = \frac{1}{n_c} \sum_{i=1}^{n_c} r_\phi(x, c_1, \cdots, c_i)$,
- **DA** : $r_\phi(x, y) = \frac{1}{n_c} \sum_{i=1}^{n_c} \left(r_\phi(x, \cdots, c_i) - r_\phi(x, \cdots, c_{i-1})\right)$.

Performance is reported in Table 8a and Table 8b, measured by RewardBench accuracy and AlpacaEval win rate. As shown, variants without differential operations for sentence rewards (VW, VA) and without weighted summation for aggregation (VA, DA) generally perform worse. Specifically, When we use value head's output as sentence rewards, using weighted summation as the aggregation function consistently outperforms average aggregation, demonstrating its superior capacity to capture the varying importance of individual sentence rewards. The DA variant performs worst on both reward modeling and alignment. All results highlights the effectiveness of the proposed differential operation and weighted summation in modeling sentence rewards and aggregating them into a response score.

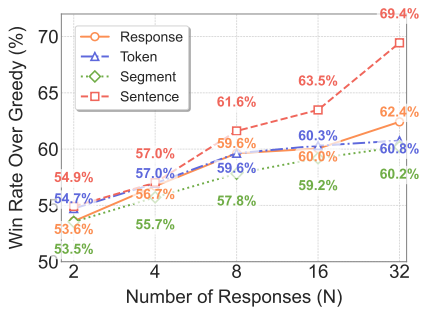
- The Impact of Segment Models

To investigate the impact of using different SaT model variants, we experiment on Llama3.2-3B with rule-based methods and five segment models (`sat-1l`, `sat-3l`, `sat-6l`, `sat-9l` and `sat-12l`). The numbers denote the number of layers in the sat model, the larger the number of layers lead to the better the segmentation performance and higher inference overhead. We report the corresponding win rates on AlpacaEval and RLHF training time in Figure 1. Figure 7 also presents the average inference latency of the reward models across different response lengths during the RLHF training process. From the perspective of latency, compared with response-level reward model, the sentence-level reward model trained with sat-3l exhibits acceptable latency, with the average processing time staying below the 50ms threshold for sequences within a 4,096-token context window. From the perspective of performance, rule-based methods perform poorly due to their limited capacity for complex text segmentation, particularly when encountering specialized domains such as math and code. As illustrated in Table 5, the reward model trained with rule-based segmentation shows a marked degradation in reasoning tasks across reward model benchmarks, with a notable performance drop specifically within the math subtask of rewardbench2. It's observed that the `sat-1l` model performs the worst in all model-based segment methods, likely due to its limited segmentation performance. While `sat-9l` achieves similar alignment performance to `sat-3l`, it results in a 19.7% increase in training time. In Table 9, we show the detail results using three different SaT models for segmentation. We observe a significant increase in win rates on AlpacaEval and Arena-Hard when replacing `sat-1l`

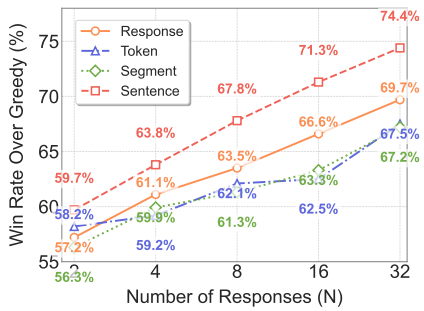
Method	Llama3.2-3B-SFT					Llama-3.1-Tulu-3-8B-SFT				
	AlpacaEval 2.0			Arena-Hard		AlpacaEval 2.0			Arena-Hard	
	LC (%)	WR (%)	Len.	WR (%)	Len.	LC (%)	WR (%)	Len.	WR (%)	Len.
Response	55.97	71.55	467	74.90	481	59.73	69.25	1587	58.70	2044
Token	55.06	70.06	2190	74.70	2016	60.88	73.60	2105	63.90	2278
Segment	58.14	74.79	613	74.80	673	68.82	74.81	1576	60.40	2054
Res2Sent	57.90	54.91	219	48.80	220	65.03	68.76	1437	61.40	1936
Sent2Res	60.32	77.27	623	74.40	583	63.41	70.87	1550	58.70	2063
DPO	50.80	58.07	328	58.60	335	58.52	64.60	1045	59.40	1733
GenRM	50.07	64.04	450	66.80	438	62.05	70.06	1223	63.60	1879
Sentence (Reinforce++)	60.32	78.39	558	76.00	638	**71.52**	75.65	1447	63.00	2021
Sentence (PPO)	**60.48**	**82.24**	812	**81.10**	762	66.86	**76.40**	1381	**68.20**	1974

Method	Qwen2.5-3B					Mistral-7B				
	AlpacaEval 2.0			Arena-Hard		AlpacaEval 2.0			Arena-Hard	
	LC (%)	WR (%)	Len.	WR (%)	Len.	LC (%)	WR (%)	Len.	WR (%)	Len.
Response	61.36	61.68	1165	59.00	1898	64.66	58.45	810	59.00	1173
Token	60.87	71.37	2149	61.80	2214	50.06	58.14	1157	57.40	1506
Segment	60.10	67.33	1703	59.60	2087	62.61	61.99	973	61.60	1371
Res2Sent	52.34	50.12	1038	56.00	1711	53.43	58.76	1108	55.30	1382
Sent2Res	63.94	69.94	1605	60.80	2054	61.81	61.68	966	65.40	1328
DPO	54.55	55.09	2371	53.50	2227	56.90	59.63	1102	59.00	1346
GenRM	59.36	59.69	2379	57.00	2247	60.97	62.28	1271	64.20	1478
Sentence (Reinforce++)	64.69	71.64	1698	61.00	2070	65.01	63.73	907	67.00	1376
Sentence (PPO)	**72.08**	**72.42**	1803	**65.50**	2036	**71.59**	**76.27**	1360	**72.30**	1526

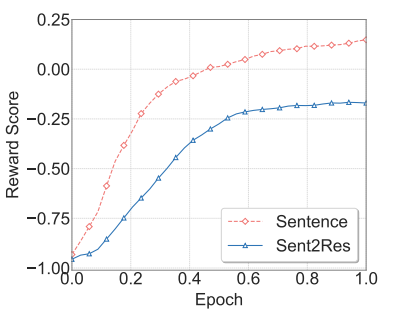
Table 4 The Win Rates (WR) and Length-Control Win Rates (LC) of different models compared to SFT on AlpacaEval2 and Arena-Hard benchmarks (Judged by Deepseek-V3). Results demonstrate the superiority of our method and its favorable trade-off between quality and response length.



(a) BoN results on AlpacaEval. N varies from 2 to 32. Win rate vs greedy decoding.



(b) BoN results on Arena-Hard. N varies from 2 to 32. Win rate vs greedy decoding.



(c) Reward curves during RL training. Sentence-level reward leads to improved training efficiency.

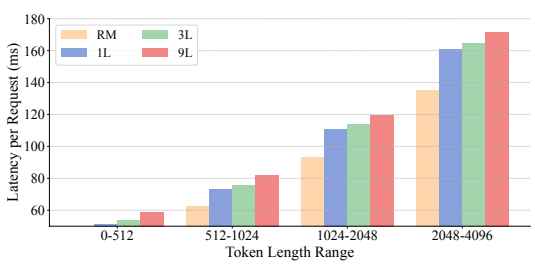


Figure 7 Average inference latency of the reward models across different response lengths during the RLHF training process.

with `sat-3l`, suggesting that improved segmentation quality can lead to better alignment performance. While `sat-12l` exhibits the best overall performance, it significantly increases the RL training time. `sat-3l`, in contrast, provides a better trade-off between performance and training time.

5.4.5 Empirical Analysis of Training Stability

The training of our Sentence-level Reward Model incorporates sentence-wise differences. To verify the stability of this approach for long responses, we conduct additional experiments using models trained on UltraFeedback [19] and HelpSteer3 [49]. We analyze the variance of sentence-level rewards across responses with varying sentence counts,

Method	RewardBench2	JudgeBench			Average
	Math	Reasoning	Math	Coding	
Rule	45.64	56.38	59.82	**52.14**	53.49
`sat-3l`	**51.91**	**58.22**	**66.07**	48.72	**56.23**

Table 5 Impact of different segmentation methods on reward model performance in reasoning tasks. The `sat-3l` model demonstrates superior performance in reasoning-related benchmarks.

with results illustrated in Figure 8. Notably, the reward variance does not exhibit a significant upward trend as the number of sentences increases, demonstrating that our design is robust and scales effectively.

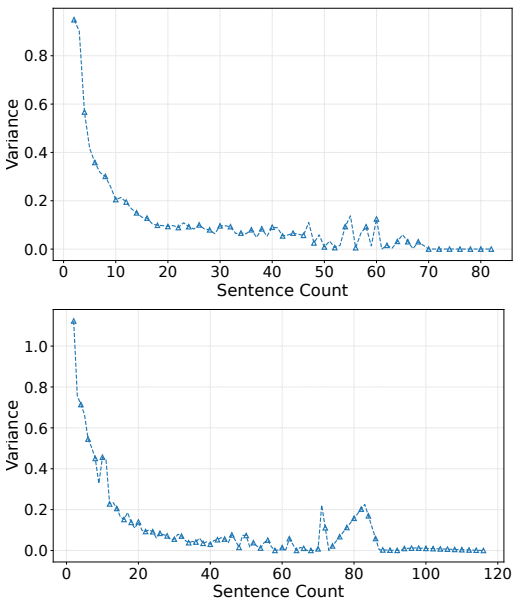


Figure 8 The variance of sentence-level rewards across responses with varying sentence numbers. Reward Models are trained with different preference dataset. The reward variance is stable as the number of sentences increases.**Upper** : Ultrafeedback **Lower** : Helpsteeer3

5.5 Examples of Generated Response

In this section, we study cases selected from AlpacaEval, from which we observe distinct behavioral patterns across reward modeling approaches. The sentence-level reward model yields responses that strike a balance between depth and conciseness: compared to response-level optimization (which produces overly generic outputs), it incorporates richer contextual details, while avoiding the verbosity and factual inconsistencies prevalent in token-level optimization. Notably, it also surpasses segment-level rewards, which often suffer from low information density. In Figure 9, Figure 10 and Figure 11, we present specific examples that illustrate these differences, highlighting the superiority of the sentence-level reward model in enhancing the final policy's training performance.

Method	Llama3.2-3B-SFT					Llama-3.1-Tulu-3-8B-SFT				
	AlpacaEval 2.0			Arena-Hard		AlpacaEval 2.0			Arena-Hard	
	LC (%)	WR (%)	Len.	WR (%)	Len.	LC (%)	WR (%)	Len.	WR (%)	Len.
Response	55.91	73.72	467	69.40	481	63.78	67.88	1587	61.20	2044
Token	53.03	73.90	2190	76.20	2016	60.82	69.61	2105	69.20	2278
Segment	56.82	76.46	613	71.20	673	65.01	69.94	1576	65.40	2054
Res2Sent	58.13	54.49	219	51.50	220	56.94	60.66	1437	62.60	1936
Sent2Res	58.91	78.55	623	78.33	583	63.33	70.99	1550	67.10	2063
DPO	51.59	59.02	328	60.29	335	58.02	63.42	1045	58.00	1733
GenRM	54.95	60.90	450	69.63	438	62.35	70.71	1423	63.60	1879
Sentence (Reinforce++)	60.44	81.25	558	82.25	638	**69.83**	72.33	1447	**71.00**	2021
Sentence (PPO)	**60.53**	**82.40**	812	**82.90**	762	65.90	**76.92**	1381	69.80	1974

Method	Qwen2.5-3B					Mistral-7B				
	AlpacaEval 2.0			Arena-Hard		AlpacaEval 2.0			Arena-Hard	
	LC (%)	WR (%)	Len.	WR (%)	Len.	LC (%)	WR (%)	Len.	WR (%)	Len.
Response	62.00	62.09	1165	57.80	1898	65.69	60.60	810	57.20	1173
Token	54.46	66.52	2149	63.40	2214	49.75	58.01	1157	63.80	1506
Segment	61.05	64.53	1703	60.60	2087	63.19	61.85	973	63.40	1371
Res2Sent	51.18	48.82	1038	51.80	1711	51.86	57.70	1108	55.20	1382
Sent2Res	62.75	69.29	1605	58.40	2054	60.88	60.51	966	66.60	1328
DPO	58.02	63.42	2371	58.00	2227	59.17	62.69	1102	58.00	1346
GenRM	56.63	56.97	2379	56.00	2247	60.10	61.15	1271	56.00	1478
Sentence (Reinforce++)	63.36	**70.50**	1698	58.20	2070	**66.12**	**64.67**	907	68.40	1376
Sentence (PPO)	**69.36**	68.79	1803	**65.50**	2036	62.39	63.72	1360	**69.00**	1526

Table 6 The win rates (WR) and Length-Control win rates (LC) of different models compared to SFT on AlpacaEval2 and Arena-Hard benchmarks(Judged by GPT-5). Results demonstrate the superiority of our method and its favorable trade-off between quality and response length.

Method	Chat	Chat-Hard	Safety	Reasoning	Average
Response	92.18	60.96	77.03	81.83	78.00
Token	94.69	55.70	70.68	84.42	76.37
Segment	91.00	60.09	76.62	83.22	77.96
GenRM	80.17	43.42	61.50	68.92	63.50
Sentence	94.97	61.84	77.16	88.89	80.72

Method	Safety	IF	Focus	Math	Factuality	Ties	Average
Response	55.44	39.37	64.24	37.16	58.63	36.91	48.63
Token	54.89	37.50	45.76	41.80	54.32	7.85	40.35
Segment	55.78	38.80	69.14	43.24	55.60	47.94	51.75
GenRM	53.22	28.28	31.51	45.63	33.68	-0.07	32.05
Sentence	56.22	32.50	72.93	51.91	58.63	50.31	53.75

Method	Knowledge	Reasoning	Math	Coding	Average
Response	49.86	59.06	68.75	47.86	55.00
Token	46.69	60.40	57.14	52.14	52.90
Segment	51.30	54.08	53.57	58.65	54.35
GenRM	38.30	45.90	44.60	33.30	40.90
Sentence	52.07	58.22	66.07	48.72	55.81

Method	Safety	Chat	Math	Code	Average
Response	75.71	67.70	57.87	51.07	63.08
Token	56.39	51.25	54.59	53.56	54.69
Segment	74.90	67.70	55.60	51.30	62.35
GenRM	80.42	60.12	53.83	51.64	61.50
Sentence	75.93	70.11	56.06	52.68	63.69

Table 7 Detailed performance of the Reward Models across various benchmarks. From top to bottom: **RewardBench**, **RewardBench2**, **JudgeBench**, **RM-Bench**.

Method	Chat	Chat-Hard	Safety	Reasoning	Average
DA	92.18	60.96	77.03	81.83	78.00
VA	93.02	61.40	75.00	82.95	78.09
VW	94.69	59.43	**81.49**	83.58	79.80
ours	**94.97**	**61.84**	77.16	**88.89**	**80.72**

(a) Pairwise accuracy on RewardBench for RMs Trained with different aggregation functions.

Method	AlpacaEval		
	LC (%)	WR (%)	Len.
DA	57.90	54.91	219
VA	59.53	70.75	388
VW	59.33	75.53	501
Ours	**60.32**	**78.39**	558

(b) Comparison of aggregation methods on AlpacaEval with the Llama-3.2-3B-SFT model.

Table 8 The Impact of different aggregation methods on trained reward models

Method	AlpacaEval			Arena-Hard	
	LC (%)	WR (%)	Len.	WR (%)	Len.
`sat-1l`	**65.86**	71.43	314	70.90	371
`sat-6l`	58.79	79.36	651	77.55	649
`sat-9l`	61.79	77.64	554	77.00	525
`sat-12l`	60.54	**81.29**	705	**80.28**	608
`sat-3l`	58.77	78.39	623	76.20	638

Table 9 The Win Rates(WR) and Length-Control Win Rates(LC) of models trained with different segment models on AlpacaEval2 and Arena-Hard benchmarks.

Instruction: Are there any weird Christmas traditions?

Response: There are indeed many unique Christmas traditions in various regions across the world. Some might be more well-known, like the German tradition of leaving out nuts and hay for mice, (incorrect, not a known Christmas tradition) while others might be less common, like the Scottish custom of singing carols while carrying a broom! While these customs are interesting, it's important to remember that all traditions should foster a spirit of warmth, unity, and goodwill.

Token: Yes, many cultures have their unique traditions associated with Christmas, some of these traditions may sound somewhat odd to us, such as the Christmas stocking exchange originated in England, which involves adults filling up their own stockings at the Christmas celebration. (incorrect, stockings are typically filled for children, not adults) While these traditions may not be everyone's personal favorites, they certainly add a layer of cultural diversity and charm to the festive season. Remember, traditions aren't about the materialism of gifts or decorations, but about passing down values, emotions, and memories that hold sentimental significance to the people involved. Some traditions, while not quite as well-known, are quite interesting, including the tradition of leaving cookies for jolly old Saint Nicholas in the hope of receiving gifts in return, or putting up the Christmas tree after Saint Nicholas Day on December 6th! These traditions showcase the rich human experiences and perspectives around Christmas celebrations worldwide. So while some traditions might appear a bit unusual to us, they're also valuable in contributing to the diverse tapestry of holiday festivities around the world. Next time around, why not explore more about these unexpected yet meaningful holiday customs, they might surprise and delight you. For more insight, a recommended book to explore is "Christmas Around the World" by Martha Day Zschock, which showcases the multifaceted traditions of various regions and cultures.

Segment: Christmas traditions vary greatly around the world, with some involving behaviors or rituals that might seem unusual or 'weird' to someone from a different cultural background. However, what might be perceived as 'weird' can often be traced back to the cultural or historical reasons behind them. For example, some cultures have traditions involving unique foods, special dances, specific gift-giving customs, or unique decorations. (This sentence is vague and lacks concrete examples) It's important to remember that what's considered 'weird' is often a matter of perspective and varies greatly from culture to culture. Would you be interested in learning about a specific country's Christmas traditions, or perhaps a lesser-known tradition from around the world? I'd be happy to provide more details on any of these topics.(defers the answer rather than directly addressing the question)

Sentence: Yes, many cultures have their unique traditions associated with Christmas, some of these traditions may sound somewhat odd to us, such as the Christmas stocking exchange originated in England, which involves adults filling up their own stockings at the Christmas celebration. There are also numerous traditions associated with Christmas tree decorating which can vary from place to place. For instance, Germany has a distinct tradition of hanging wreaths adorned with candles called Lichterketten on their Christmas trees, (accurate and specific example) while in Scandinavia, they typically wear masks during their Christmas parties. In Ireland, it's common to serve mulled wine, (culturally accurate detail) while in Portugal, the use of a nativity scene called a presépe is widespread. Different versions of The Twelve Days of Christmas are also very famous traditions in some places. These are just some examples of unique and distinctive Christmas traditions across geographical locations around the world. Remember, it's all about celebrating the spirit of Christmas with love and unity. How about starting an engaging conversation about these traditions with a family member, a friend, or a colleague today? (engaging and informative conclusion)

Figure 9 Comparison of responses to a question about unusual Christmas traditions. In the examples, correct and accurate information is explained in dark green, while incorrect or misleading details are marked in red. **Sentence** provides a detailed, accurate, and engaging overview of unique traditions across cultures, such as Germany's Lichterketten and Ireland's mulled wine, demonstrating its superiority. In contrast, **Response** incorrectly describes a German tradition of leaving nuts and hay for mice, and **Token** inaccurately states that adults fill their own stockings in England. **Segment** defers the answer rather than directly addressing the question and lacks informative examples. **Sentence**'s response is both informative and culturally precise, making it the most reliable and comprehensive answer.

Instruction: Write a C++ function that takes a reference to a std::string containing markdown formatted text and returns a std::string containing html formatted text.

Response: Here's some sample code that accomplishes what you're asking for:

```
std::string escapeMarkdown(const std::string& markdown_code) {
  return std::string(markdown_code.begin(), markdown_code.end());
}
```

(Fails to convert Markdown to HTML)

```
int main() {
  std::string markdown_body = "# Welcome To My Website\n\nThis website was "
                              "created as a part of a course. It mainly "
                              "features content related to my course projects "
                              "and technical skills acquired during the course. "
                              "Let's dive in!";
  std::string html_rendered_body = escapeMarkdown(markdown_body);
  return 0;
}
```

Token: Here's a possible implementation of a function that takes a markdown formatted string as input and returns its html counterpart:

```
#include <iostream>
#include <string>
#include <cstring>
#include <cctype>
std::string markdown2html(const std::string& md) {
    std::ostringstream oss;

    // Replace asterisks with actual bold text
    std::string re = "[b][u]([0-9a-z]*)[/u][/b]";
    std::string final_re = ".*?";
    oss << "<b><u>";
    while (md.substr(0).find_first_of(" *", final_re, re)!= std::string::npos)
    {
        int idx = md.find_first_of(" *", final_re, re);
        const std::string chunk = md.substr(0, idx);
        const std::string newChunk = chunk.replace(chunk.begin(), chunk.end(), "<b><u>");
        md = md.substr(idx + 1); // Remove the original character(s) from the md string
        oss << newChunk << " ";
    }
    // Replace underscores with actual italicized text
    oss << "<sub>[i]";
    re = "[i]";
    final_re = ".*?";
    oss << "<u>*([0-9a-z]*)*</u>";
    oss << "</sub>";
    while (md.substr(0).find_first_of(" *", final_re, re)!= std::string::npos)
    {
        int idx = md.find_first_of(" *", final_re, re);
        const std::string chunk = md.substr(0, idx);
        const std::string newChunk = chunk.replace(chunk.begin(), chunk.end(), "<b><u>");
        md = md.substr(idx + 1); // Remove the original character(s) from the md string
        oss << newChunk << " ";
    }
    return oss.str();
}
```

(Incorrect implementation and overly complex)

Figure 10 Comparison of responses to a C++ Markdown-to-HTML conversion task (Part a). **Response** fails to convert Markdown, while **Token** provides an incorrect and over-engineered solution.

Segment: This problem can be solved using C++ programming language, but note that C++ isn't typically used for parsing markdown to HTML. Usually, this parsing is done on the server-side in a language like Python, Ruby, or JavaScript. Parsing markdown can be very complex due to the variety of ways in which markdown can be formatted.
However, for the sake of responding to this request, here's a simple demonstration of how this could be done. In our code, we first include the relevant header files, define the function, iterate over each word of the input string, and replace markdown with HTML tags.
But this is over simplifying the problem and it won't cover all the edge cases, nested tags, complex markdown syntax, etc.

```
#include <iostream>
#include <string>
#include <sstream>
using namespace std;

string convertMarkdownToHTML(string &markdown) {
    // This is oversimplified for demonstration purposes, a real-world solution would require a much more complex parsing logic
    // for markdown.
    string html = "";

    stringstream ss(markdown);
    string word;

    while (getline(ss, word,'')) {
        if (word == "This") {
            html += "<p>";
        } else if (word == "is") {
            html += "</p><p>";
        } else if (word == "an") {
            html += "<strong>";
        } else if (word == "example") {
            html += "</strong>";
        } else {
            html += word + " ";
        }
    }

    return html;
}

int main() {
    string markdown = "This is an example";
    string html = convertMarkdownToHTML(markdown);
    cout << html << endl;
    return 0;
}
```

This will print: `<p><strong>This is an example</strong></p>`.
Again, it has to be noted this is a very simplistic implementation, a real-world solution would need to handle much more complex cases, for example by using a library that is built to parse Markdown, or a language specifically designed for text processing, like Python.
(getline(ss, word, '') is invalid, so the code wouldn't output expected result.)

Sentence: Sure, here is an example of how we can create a generic function that converts Markdown to HTML:

```
std::string convert_md_to_html(const std::string& input_text) {
   std::stringstream html_stream;
   html_stream << "<p>" << input_text << "</p>";
   return html_stream.str();
}
```

(A concise and simple implementation)

Figure 11 Comparison of responses to a C++ Markdown-to-HTML conversion task (Part b). **Segment** provides a realistic discussion with a toy implementation, while **Sentence** offers a clean, minimal, and functional solution.

References

[1] Dubey A, Jauhri A, Pandey A, Kadian A, Al-Dahle A, Letman A, Mathur A, Schelten A, Yang A, Fan A, others . The Llama 3 herd of models. arXiv preprint arXiv:2407.21783, 2024

[2] Ouyang L, Wu J, Jiang X, Almeida D, Wainwright C, Mishkin P, Zhang C, Agarwal S, Slama K, Ray A, others . Training language models to follow instructions with human feedback. In: Advances in Neural Information Processing Systems. 2022, 27730–27744

[3] Chan A J, Sun H, Holt S, Schaar v. d M. Dense reward for free in reinforcement learning from human feedback. In: International Conference on Machine Learning. 2024, 6136–6154

[4] Yang S, Zhang S, Xia C, Feng Y, Xiong C, Zhou M. Preference-grounded token-level guidance for language model fine-tuning. In: Advances in Neural Information Processing Systems. 2023, 24466–24496

[5] Guo G, Zhao R, Tang T, Zhao X, Wen J R. Beyond imitation: Leveraging fine-grained quality signals for alignment. In: International Conference on Learning Representations. 2024

[6] Bradley R A, Terry M E. Rank analysis of incomplete block designs: I. the method of paired comparisons. Biometrika, 1952, 39(3/4): 324–345

[7] Frohmann M, Sterner I, Vulić I, Minixhofer B, Schedl M. Segment any text: A universal approach for robust, efficient and adaptable sentence segmentation. In: Conference on Empirical Methods in Natural Language Processing. 2024, 11908–11941

[8] Hu J. REINFORCE++: A simple and efficient approach for aligning large language models. arXiv preprint:2501.03262, 2025

[9] Yin Y, Yang S, Xie Y, Yang Z, Sun Y, Awadalla H, Chen W, Zhou M. Segmenting text and learning their rewards for improved rlhf in language model. arXiv preprint arXiv:2501.02790, 2025

[10] LCM , Barrault L, Duquenne P A, Elbayad M, Kozhevnikov A, Alastruey B, Andrews P, Coria M, Couairon G, Costa-jussà M R, others . Large concept models: Language modeling in a sentence representation space. arXiv preprint arXiv:2412.08821, 2024

[11] Kim C, Park J, Shin J, Lee H, Abbeel P, Lee K. Preference Transformer: Modeling human preferences using transformers for RL. In: International Conference on Learning Representations. 2023

[12] Su J, Ahmed M, Lu Y, Pan S, Bo W, Liu Y. Roformer: Enhanced transformer with rotary position embedding. Neurocomputing, 2024, 568: 127063

[13] Gage P. A new algorithm for data compression. The C Users Journal, 1994, 12(2): 23–38

[14] Sennrich R. Neural machine translation of rare words with subword units. In: Annual Meeting of the Association for Computational Linguistics. 2016, 1715–1725

[15] Christiano P F, Leike J, Brown T, Martic M, Legg S, Amodei D. Deep reinforcement learning from human preferences. In: Advances in Neural Information Processing Systems. 2017, 4299–4307

[16] Liu C Y, Zeng L, Liu J, Yan R, He J, Wang C, Yan S, Liu Y, Zhou Y. Skywork-reward: Bag of tricks for reward modeling in LLMs. arXiv preprint arXiv:2410.18451, 2024

[17] Yang A, Zhang B, Hui B, Gao B, Yu B, Li C, Liu D, Tu J, Zhou J, Lin J, others . Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. arXiv preprint arXiv:2409.12122, 2024

[18] Wang P, Li L, Shao Z, Xu R, Dai D, Li Y, Chen D, Wu Y, Sui Z. Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations. In: Annual Meeting of the Association for Computational Linguistics. 2024, 9426–9439

[19] Cui G, Yuan L, Ding N, Yao G, He B, Zhu W, Ni Y, Xie G, Xie R, Lin Y, others . UltraFeedback: Boosting language models with scaled AI feedback. In: International Conference on Machine Learning. 2024, 9722–9744

[20] Cai Z, Cao M, Chen H, Chen K, Chen K, Chen X, Chen X, Chen Z, Chen Z, Chu P, others . InternLM2 technical report. arXiv preprint arXiv:2403.17297, 2024

[21] Wang H, Xiong W, Xie T, Zhao H, Zhang T. Interpretable preferences via multi-objective reward modeling and mixture-of-experts. arXiv preprint arXiv:2406.12845, 2024

[22] Chen L, Zhu C, Chen J, Soselia D, Zhou T, Goldstein T, Huang H, Shoeybi M, Catanzaro B. ODIN: Disentangled reward mitigates hacking in RLHF. In: International Conference on Machine Learning. 2024, 7935–7952

[23] Zheng L, Chiang W L, Sheng Y, Zhuang S, Wu Z, Zhuang Y, Lin Z, Li Z, Li D, Xing E, others . Judging LLM-as-a-judge with MT-bench and Chatbot Arena. In: Advances in Neural Information Processing Systems. 2023, 46595–46623

[24] Mahan D, Van Phung D, Rafailov R, Blagden C, Lile N, Castricato L, Fränken J P, Finn C, Albalak A. Generative reward models. arXiv preprint arXiv:2410.12832, 2024

[25] Wei J, Wang X, Schuurmans D, Bosma M, Xia F, Chi E, Le Q V, Zhou D, others . Chain-of-thought prompting elicits reasoning in large language models. In: Advances in Neural Information Processing Systems. 2022, 24824–24837

[26] Zhang L, Hosseini A, Bansal H, Kazemi M, Kumar A, Agarwal R. Generative verifiers: Reward modeling as next-token prediction. In: The 4th Workshop on Mathematical Reasoning and AI at NeurIPS'24. 2024

[27] Zhong H, Feng G, Xiong W, Cheng X, Zhao L, He D, Bian J, Wang L. DPO meets PPO: Reinforced token optimization for RLHF. arXiv preprint arXiv:2404.18922, 2024

[28] Rafailov R, Hejna J, Park R, Finn C. From r to Q^*: Your language model is secretly a Q-function. arXiv preprint arXiv:2404.12358, 2024

[29] Rafailov R, Sharma A, Mitchell E, Manning C D, Ermon S, Finn C. Direct preference optimization: Your language model is secretly a reward model. In: Advances in Neural Information Processing Systems. 2023, 53728–53741

[30] Yoon E, Yoon H S, Eom S, Han G, Nam D W, Jo D, On K W, Hasegawa-Johnson M, Kim S, Yoo C D. TLCR: Token-level continuous reward for fine-grained reinforcement learning from human feedback. In: Findings of the Association for Computational Linguistics. 2024, 14969–14981

[31] Cao M, Shu L, Yu L, Zhu Y, Wichers N, Liu Y, Meng L. DRLC: Reinforcement learning with dense rewards from LLM critic. arXiv preprint arXiv:2401.07382, 2024

[32] Zhou J, Ji J, Dai J, Yang Y. Sequence to sequence reward modeling: Improving RLHF by language feedback. arXiv preprint arXiv:2409.00162, 2024

[33] Lightman H, Kosaraju V, Burda Y, Edwards H, Baker B, Lee T, Leike J, Schulman J, Sutskever I, Cobbe K. Let's verify step by step. In: International Conference on Learning Representations. 2024

[34] Luo L, Liu Y, Liu R, Phatale S, Lara H, Li Y, Shu L, Zhu Y, Meng L, Sun J, others . Improve mathematical reasoning in language models by automated process supervision. arXiv preprint arXiv:2406.06592, 2024

[35] Bai Y, Jones A, Ndousse K, Askell A, Chen A, DasSarma N, Drain D, Fort S, Ganguli D, Henighan T, others . Training a helpful and harmless assistant with reinforcement learning from human feedback. arXiv preprint arXiv:2204.05862, 2022

[36] Yao Z, Aminabadi R Y, Ruwase O, Rajbhandari S, Wu X, Awan A A, Rasley J, Zhang M, Li C, Holmes C, others . DeepSpeed-Chat: Easy, fast and affordable RLHF training of chatgpt-like models at all scales. arXiv preprint arXiv:2308.01320, 2023

[37] Yang A, Yang B, Zhang B, Hui B, Zheng B, Yu B, Li C, Liu D, Huang F, Wei H, Lin H, Yang J, Tu J, Zhang J, Yang J, Yang J, Zhou J, Lin J, Dang K, Lu K, Bao K, Yang K, Yu L, Li M, Xue M, Zhang P, Zhu Q, Men R, Lin R, Li T, Xia T, Ren X, Ren X, Fan Y, Su Y, Zhang Y, Wan Y, Liu Y, Cui Z, Zhang Z, Qiu Z. Qwen2.5 technical report, 2025

[38] Jiang A Q, Sablayrolles A, Mensch A, Bamford C, Chaplot D S, Casas d. l D, Bressand F, Lengyel G, Lample G, Saulnier L, Lavaud L R, Lachaux M A, Stock P, Scao T L, Lavril T, Wang T, Lacroix T, Sayed W E. Mistral 7B, 2023

[39] Lambert N, Pyatkin V, Morrison J, Miranda L, Lin B Y, Chandu K, Dziri N, Kumar S, Zick T, Choi Y, others . RewardBench: Evaluating reward models for language modeling. arXiv preprint arXiv:2403.13787, 2024

[40] Liu Y, Yao Z, Min R, Cao Y, Hou L, Li J. Rm-bench: Benchmarking reward models of language models with subtlety and style. arXiv preprint arXiv:2410.16184, 2024

[41] Tan S, Zhuang S, Montgomery K, Tang W Y, Cuadron A, Wang C, Popa R A, Stoica I. Judgebench: A benchmark for evaluating llm-based judges. arXiv preprint arXiv:2410.12784, 2024

[42] Malik S, Pyatkin V, Land S, Morrison J, Smith N A, Hajishirzi H, Lambert N. Rewardbench 2: Advancing reward model evaluation. arXiv preprint arXiv:2506.01937, 2025

[43] Dubois Y, Galambosi B, Liang P, Hashimoto T B. Length-controlled AlpacaEval: A simple way to debias automatic evaluators. arXiv preprint arXiv:2404.04475, 2024

[44] Li T, Chiang W L, Frick E, Dunlap L, Wu T, Zhu B, Gonzalez J E, Stoica I. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. arXiv preprint arXiv:2406.11939, 2024

[45] Liu A, Feng B, Xue B, Wang B, Wu B, Lu C, Zhao C, Deng C, Zhang C, Ruan C, others . DeepSeek-V3 technical report. arXiv preprint arXiv:2412.19437, 2024

[46] Stiennon N, Ouyang L, Wu J, Ziegler D, Lowe R, Voss C, Radford A, Amodei D, Christiano P F. Learning to summarize with human feedback. In: Advances in Neural Information Processing Systems. 2020, 3008–3021

[47] Ren J, Rajbhandari S, Aminabadi R Y, Ruwase O, Yang S, Zhang M, Li D, He Y. ZeRO-offload: Democratizing billion-scale model training. In: USENIX Annual Technical Conference. 2021, 551–564

[48] Dao T. FlashAttention-2: Faster attention with better parallelism and work partitioning. In: International Conference on Learning Representations. 2024

[49] Wang Z, Zeng J, Delalleau O, Egert D, Evans E, Shin H, Soares F, Dong Y, Kuchaiev O. Helpsteer3: Human-annotated feedback and edit data to empower inference-time scaling in open-ended general-domain tasks. In: Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2025, 25640–25662