

RewardPartial: A Completion Potential Evaluation Framework for Truncated Best-of-N Paradigm

Xin Wang^{1,4§}, Yidong Wang¹, Qiufeng Wang², Bo Jia³,
Yilin Niu⁴, Jie Tang⁵, Cunxiang Wang^{4†}

¹Peking University, ²Southeast University,

³Beijing University of Posts and Telecommunications,

⁴Zhipu AI, ⁵Tsinghua University

[§]Work done during an internship at Zhipu AI. [†]Corresponding author

Correspondence: wangcunxiang303@gmail.com

Abstract

Traditional decoding-time alignment methods, such as Best-of-N sampling, enhance output quality by selecting high-reward responses from multiple candidates. However, these approaches incur substantial computational overhead, which limits their practicality. Recent advances aim to mitigate this inefficiency by scoring and filtering partial generations early in the decoding process. However, these methods face a critical challenge: conventional reward models (RMs), optimized to assess complete responses, exhibit poor performance when evaluating truncated text. To address this, we introduce **RewardPartial**¹, a novel evaluation paradigm that shifts the focus from superficial partial-text quality to estimating a response’s completion potential. We provide theoretical analysis for this paradigm through the perspective of Markov Decision Process and construct TruncEval, a specialized dataset for training RewardPartial models to accurately predict this potential. Our evaluation on the TruncEval test set demonstrates that RewardPartial models deliver significant performance gains over conventional reward models. When deployed in early-filtering pipelines, our method achieves superior alignment performance compared to Best-of-N sampling, offering a more efficient solution for high-quality text generation.

1 Introduction

Decoding-time alignment (Huang et al., 2024; Shi et al., 2024b) has emerged as a promising training-free paradigm (Shi et al., 2024a; Zanella et al., 2024; Wang et al., 2024) that leverages reward models (Mahan et al., 2024; Chen et al., 2025; Ye et al., 2024) during inference to preserve the general capabilities of LLMs while enabling flexible preference adaptation. However, conventional decoding-time methods like Best-of-N sampling (Verdun et al.,

2025; Gui et al., 2024; Kang et al., 2025) incur prohibitive computational waste by generating multiple full responses prior to evaluation. To address this, a more efficient paradigm (Sun et al., 2024; Li et al., 2024) has emerged: partially generating responses with controlled tokens, scoring them with Reward Model (RM), and selectively completing only high-reward candidates, thus balancing alignment quality with computational efficiency. For simplicity, we refer to it as Truncated Best-of-N.

Although Truncated Best-of-N demonstrates considerable potential, its reliance on conventional reward models (RMs) to assess partial responses raises concerns regarding its alignment reliability. Deng and Raffel (2023); Yang and Klein (2021); Khanov et al. (2024) demonstrated that RMs often lead to suboptimal alignment quality when applied directly to incomplete text. Similarly, Li et al. (2024) found a non-positive correlation between an RM’s scores for the partial and complete versions of the same response. This leads to a fundamental inconsistency in evaluation across the generation timeline. As a result, as depicted in Figure 1, the Truncated Best-of-N alignment paradigm can suffer from suboptimal performance of RMs on partial responses, ultimately undermining the effectiveness of the alignment process.

To address this critical inconsistency, we propose the development of a novel class of RM, **RewardPartial**, specifically designed to perform predictive evaluation of partial responses. This RM would be trained not to score the present state of a text fragment, but to estimate the ultimate alignment quality and overall reward of the completed response that the partial text is likely to lead to. By learning to project the future trajectory and final quality from an incomplete prefix, RewardPartial would provide temporally consistent and forward-looking assessments. This capability would seamlessly integrate into the Truncated Best-of-N paradigm, enabling accurate early selection of high-potential candi-

¹<https://github.com/RewardPartial/RewardPartial>

Evolution of Best-of-N Sampling for Efficient Decoding-Time Alignment

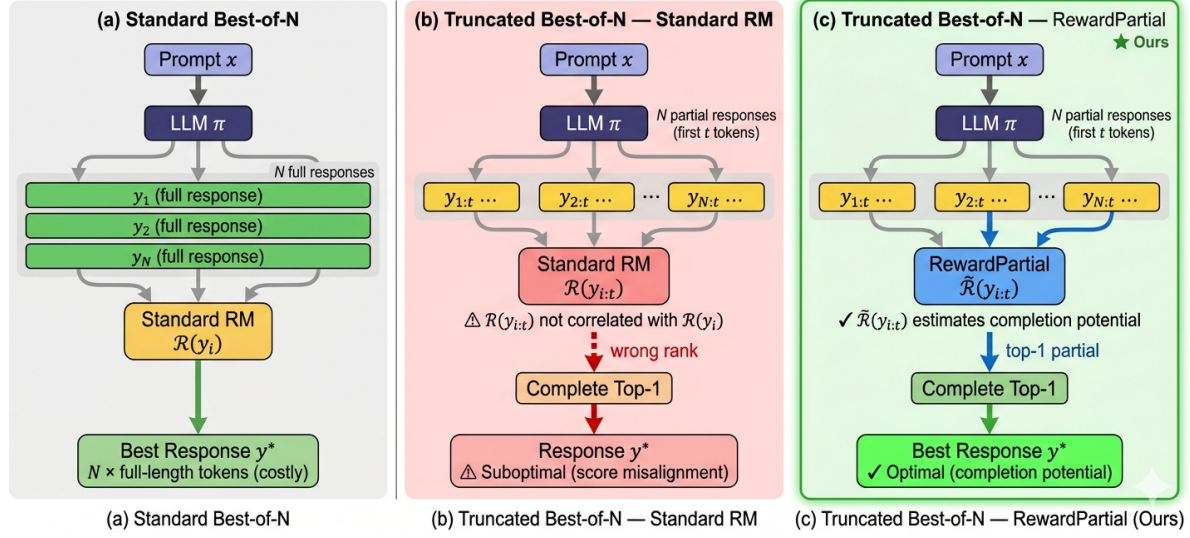


Figure 1: **Evolution of Best-of-N sampling for efficient decoding-time alignment.** From the costly standard approach (a) to the efficient but unreliable truncation with standard RMs (b), which exhibits a scoring misalignment between partial and complete responses. Our method (c) addresses this misalignment by integrating a novel RewardPartial model, specifically trained to predict the completion potential of partial responses, thereby ensuring both efficiency and alignment fidelity.

dates without the need for wasteful full generation.

The design of RewardPartial is grounded in a key empirical observation: a partial response substantially predetermines its eventual perceived quality. As illustrated in Figure 2, when the same truncated response is independently extended by multiple LLMs with diverse architectures, scales, and capabilities, the resulting complete responses consistently receive remarkably similar RM scores (Detailed in Appendix A.1). This indicates that a partial response possesses a quantifiable completion potential—its capacity to evolve into high-reward terminal states. Building on this insight, we adopt a Monte Carlo estimation (Dagum et al., 2000; Li et al., 2025) approach to approximate the completion potential of partial responses that we generate multiple possible completions using a diverse set of LLMs and aggregate their reward scores. This procedure allows us to construct TruncEval, a novel dataset in which partial responses are annotated with empirically derived completion potentials rather than superficial quality scores. Trained on TruncEval train set via Supervised Fine-tuning, RewardPartial learns to predict completion potential, thereby providing a reliable and efficient early-stage scoring mechanism for the Truncated Best-of-N framework.

Our experimental results comprehensively val-

idate both the accuracy of RewardPartial and its practical effectiveness in real-world applications. On TruncEval test set, RewardPartial achieves significant improvements across all metrics. For instance, Qwen2.5-7B-Instruct-RewardPartial (Yang et al., 2025) attains an exact match score of 0.7278 and a Pearson correlation (Sedgwick, 2012; Harary, 2024) of 0.8351, substantially outperforming the base Qwen2.5-7B-Instruct model (0.3789 and 0.5272, respectively) as well as the strongest proprietary API alternatives. More critically, the extensive Best-of-N experiments on AlpacaEval 2.0 (Dubois et al., 2024; Zheng et al., 2025) reveal that our method consistently surpasses conventional sampling approaches across different generator models (Qwen1.5/3/7/14B), N (3, 10, 30) and RMs (Llama3.1-8B-Instruct and Qwen2.5-7B-Instruct), showing stable performance gains when applied within the Truncated Best-of-N paradigm. Specifically, compared with Standard Best-of-3, Truncated Best-of-N achieves 50.2% win rate and 20.8% lose rate under similar tokens usage. Similar gains can also be seen on math and code tasks, and our method can also improve the efficiency and performance of model training.

In conclusion, we make three key contributions.

- We propose a novel reward model paradigm **RewardPartial** to resolve the critical incon-

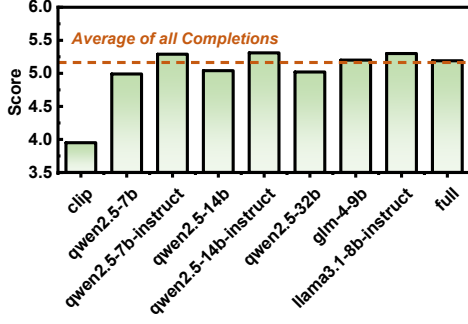


Figure 2: Comparison of reward model scores across three response types: (1) original full-length answers, (2) clipped segments by first 100 tokens and (3) full responses completed by diverse LLMs. All model completions achieved highly similar scores (ranging from 4.99 to 5.31 with the average of 5.16), much higher than the truncated segments’ scores (3.95), indicating that response quality is largely driven by the initial partial context. Experiment details and extended analyses are provided in Appendix A.1.

sistency in the efficient Truncated Best-of-N framework, where conventional RMs cannot reliably assess partial responses.

- We provide both theoretical grounding and empirical evidence to demonstrate that partial responses can predetermine final quality, and we introduce the first benchmark, **TruncEval**, designed specifically to evaluate the completion potential of truncated responses.
- We extensively validate RewardPartial’s superior scoring accuracy on our curated TruncEval test set and demonstrate its practical effectiveness, enabling the Truncated Best-of-N paradigm to achieve higher alignment quality with similar computational cost.

2 Methodology

In this section, we first present our theoretical analysis and empirical verification of RewardPartial and then introduce the construction and application methods of RewardPartial.

2.1 A Theory Analysis and Empirical Verification

In the previous experiment Figure 2, we have confirmed that RMs consistently undervalue partial responses compared to their complete versions:

$$E[R(y_{1:t})] \ll E[R(y_{1:t} \oplus y_{t+1:T})] \quad (1)$$

where $y_{1:t}$ denotes a truncated response at t token and $y_{t+1:T}$ denotes its completion to a full response.

Evaluate a submitted unfinished response to a user’s query. First provide a rating score, then complete it before writing your critiques.

[BEGIN DATA]

[Query]: {Query}

[Unfinished_Response]: {Unfinished_Response}

[END DATA]

First provide your final rating for the response on a scale of 1 to 10 by strictly following this format: "[[rating]]", for example: "Rating: [[5]]".

Then

1. First write "[Response Completion]" and provide what you judge to be a natural completion.

2. Write "[Judgment]" and your critiques of both the original and completed portions.

Figure 3: Prompt Template of RewardPartial for Completion Potential Evaluation

Additionally, $\oplus$ denotes concatenation, R denotes reward from RM and E denotes expectation.

To address this RM bias in the Truncated Best-of-N paradigm, we define the language generation process as a Markov Decision Process (as detailed in Appendix A.2) and propose evaluating partial responses through their **completion potential** with:

$$\tilde{R}(y_{1:t}) = E_{y_{t+1:T} \sim p(\cdot | x, y_{1:t})} [R(y_{1:T})] \quad (2)$$

By Monte Carlo Estimation theory, the expectation in Eq. 2 can be practically estimated by:

$$\tilde{R}(y_{1:t}) \approx \frac{1}{N} \sum_{i=1}^N R(y_{1:t} \oplus \hat{y}_{t+1:T}^{(i)}), \quad \hat{y}_{t+1:T}^{(i)} \sim p(\cdot | x, y_{1:t}) \quad (3)$$

Through N completion trials using diverse LLMs, we need to verify the estimation stability:

$$CV(\{R(y_{1:t} \oplus M_i(y_{t+1:T}))\}_{i=1}^N) < \epsilon \quad (4)$$

A low CV (coefficient of variation) value indicates that the estimation variance is controlled, confirming the reliability of Monte Carlo estimation with limited samples. In our practice, the threshold $\epsilon = 0.15$ is set as discussed in Appendix A.2.

2.2 TruncEval Dataset Construction For Training and Testing RewardPartial

Leveraging Monte Carlo Estimation to predict the completion contents and scores of partial responses, we integrate response completions with their corresponding scores to construct the TruncEval dataset through a carefully designed prompt (Figure 3). This paradigm of RM generates completions for truncated responses before evaluating them, thereby developing the capability to assess

Algorithm 1 Different Baselines of Best-of-N with similar Token Consumption

Input: Prompt set X with K prompts, Base LLM π , Reward models R_p for full responses/ R_f for partial responses, Truncation tokens length L , Full generation times N

Output: Optimized responses $\{y_k^*\}$ of two methods

```
1: Method 1: Best-of-N
2: for each prompt  $x_k \in X$  do
3:   Generate  $N$  full responses with  $\pi$ , score with  $R_f$ 
4:   Select highest-scoring response  $y_k^*$ 
5:   Record its token consumption  $t_k$ 
6: end for
7: Calculate total tokens usage  $T$  and average  $t_{avg}$  across all prompts
8: Method 2: Truncated Best-of-N'(ours)
9: Compute  $N'$  based on  $T$ ,  $t_{avg}$ ,  $L$  and  $K$  and maintain similar token budget with Best-of-N
10:  $N' \leftarrow \lfloor (T - K \times (t_{avg} - L)) / L \rfloor$ 
11: for each prompt  $x_k \in X$  do
12:   Generate  $N'$  responses truncated at  $L$  tokens with  $\pi$ 
13:   Score partial responses with  $R_p$ , select best partial
14:   Complete the selected partial to final response  $y_k^*$ 
15: end for
16: return All optimized responses  $\{y_k^*\}$  of two methods
```

the completion potential of partial outputs. To optimize practical efficiency in application, our prompt design emphasizes the Rating section in the beginning, allowing for immediate truncation after score retrieval, which significantly reduces token consumption during RM scoring while maintaining evaluation accuracy.

Using TruncEval train set, we train RewardPartial from general language model to predict the expected reward of partial responses using Supervised Fine-tuning:

$$\mathcal{L}_{\text{SFT}}(\theta) = E_{(x,y) \sim \mathcal{D}} \left[- \sum_{t=1}^{|T|} \log P_{\theta}(y_t \mid x, y_{<t}) \right]$$

After getting RewardPartial models, we evaluate them on TruncEval test set and demonstrate the performance on truncated responses through four metrics: Exact Match, Kendall's τ , Spearman's ρ and Pearson's r .

2.3 Integration into Decoding-Time Alignment (Best-of-N Sampling)

The practical efficacy of RewardPartial must be demonstrated in real application with Truncated Best-of-N. To enable a fair comparison with Best-of-N sampling paradigm under similar token consumption, we designed an equitable evaluation scheme (as illustrated in the algorithm 1) to assess quality differences and validate the effectiveness of our RewardPartial in real-world Decoding-Time

Alignment scenarios. Another point to claim is that despite the higher frequency of RM usage in Truncated Best-of-N than Best-of-N, the prompt design formats scores as "Rating: [[5]]" in the beginning and can be truncated afterward, limiting RM token consumption to five tokens (only 1% of generation tokens). To verify that switching from RewardFull to RewardPartial does not come at the cost of extra overhead elsewhere in the pipeline, we additionally measure wall-clock latency, FLOPs, and peak GPU memory of RewardPartial under Truncated Best-of-N' against RewardFull under standard Best-of-N with matched compute budget (Appendix A.3, Table 7), which shows that switching to RewardPartial introduces no additional computational overhead over the standard pipeline on the RM side.

3 Construction and Evaluation

We construct the TruncEval dataset through multi-model completion strategies and use it to train and test RewardPartial. Our comprehensive evaluation demonstrate significant improvement over existing reward modeling paradigms. We further conduct targeted ablation studies to validate key design decisions, including data combination strategies and optimal truncation locations.

3.1 TruncEval Dataset Construction

Our TruncEval dataset is derived from the UltraFeedback-Binarized (Cui et al., 2023) corpus, which comprises 61k prompts and their corresponding chosen responses. We first filter responses shorter than 200 tokens (tokenized by Qwen2.5) and truncate the remaining responses to their first 100 tokens. From this pool, we randomly sample 6,000 instances and split them into 5,000 train samples and 1,000 test samples. The original chosen responses have an average length of 458.28 tokens.

To evaluate completion potential, we generate diverse continuations for each truncated response using seven LLMs with varying architectures, scales, and training regimes: Qwen2.5-7B (Yang et al., 2025), Qwen2.5-14B, Qwen2.5-32B, Qwen2.5-7B-Instruct, Qwen2.5-14B-Instruct, Llama3.1-8B-Instruct (Grattafiori et al., 2024), and GLM-4-9B-0414 (Team GLM et al., 2024). Then we score all completions on a 10-point scale using the AutoJ-13B (Li et al., 2023) reward model.

Our analysis revealed two key findings: First, completed responses consistently receive signifi-

Model	Exact Match	$K\tau$	$S\rho$	Pr
GPT-5.5	0.2734	0.3287	0.3716	0.3782
Claude-4.7-Opus	0.4263	0.3614	0.4027	0.3019
Gemini-3.0-Pro	0.3215	0.2913	0.3198	0.4143
Llama-3.1-8B-Instruct	0.5345	0.3418	0.3663	0.4374
Qwen2.5-7B-Instruct	0.3789	0.3759	0.4300	0.5272
RewardPartial (Llama8B)	0.7354	0.7757	0.8218	0.8452
RewardPartial (Qwen7B)	<u>0.7278</u>	<u>0.7572</u>	<u>0.8017</u>	<u>0.8351</u>

Table 1: Evaluation Results on TruncEval test set. RewardPartial (Llama8B)/(Qwen7B) stands for RewardPartial trained from Llama3.1-8B-Instruct and Qwen2.5-7B-Instruct respectively. The metric $K\tau$, $S\rho$ and Pr mean Kendall’s τ , Spearman’s ρ and Pearson’s r .

cantly higher scores than their truncated counterparts (average scores of 4.99-5.31 versus 3.95). Second, different models’ completions of the same truncated response achieved remarkably similar scores, as evidenced by a low coefficient of variation (0.14). Critically, as detailed in Appendix A.1, this completion-score stability holds consistently across three independent reward models (AutoJ-13B, GPT-4.1, and ArmoRM-Llama3-8B), confirming that completion potential is a robust, scorer-agnostic signal rather than an artifact of any single evaluator’s biases.

Based on our analysis in Eq.3 and Eq.4, we employ Monte Carlo Estimation to determine the completion potential of these partial responses. We select the completion closest to the mean score to pair with each truncated response, forming the Supervised Fine-tuning data for RewardPartial. Additionally, we use this data to train RewardFull for pair comparison in subsequent experiments, as detailed in Appendix A.4.

Data Construction Cost Analysis. The TruncEval dataset construction involves 210,000 total completions ($7 \text{ models} \times 5 \text{ completions} \times 6,000 \text{ samples}$) followed by reward scoring with AutoJ-13B. We estimate the total computational cost on 80GB GPUs using vLLM for batched inference. The entire pipeline costs approximately 20 GPU hours as a one-time expense. This is more consuming compared to typical RLHF data annotation pipelines but is substantially amortized by the efficiency gains during deployment, where RewardPartial eliminates the need for full-length generation of most candidates in Truncated Best-of-N sampling.

3.2 RewardPartial Training and Evaluation

We fine-tune the base Instruct model on the TruncEval training set to obtain RewardPartial, training it to assess the potential of partial completions. Fine-tuning is performed via full-parameter optimization

over three epochs at a learning rate of 1.0×10^{-5} using four 80GB GPUs. To evaluate RewardPartial’s ability to score partial responses, we test it on the TruncEval test set and measure its reward outputs and the ground-truth scores. A full comparison with baseline models is provided in Table 1, evaluated across four metrics: Exact Match, Kendall’s τ , Spearman’s ρ , and Pearson’s r . All models are evaluated under the same prompting setup for partial responses, shown in Figure 3.

The baselines in Table 1 represent a broad spectrum of alternative RM paradigms. The base Instruct models (Llama-3.1-8B-Instruct, Qwen2.5-7B-Instruct) represent zero-shot conventional RM evaluation. The proprietary API models (GPT-5.5, Claude-4.7-Opus, Gemini-3.0-Pro) represent the strongest zero-shot models, benefiting from vastly larger scale and RLHF pretraining. That even these large-scale models fail to accurately predict completion potential confirms that the gap is conceptual rather than a capacity limitation, which means standard RMs evaluate current text quality but not future completion trajectory.

The experimental results confirm the superior performance of RewardPartial models across all evaluation metrics. The Llama-3.1-8B-Instruct-RewardPartial model achieves 0.7354 in Exact Match, 0.7757 in Kendall’s τ , 0.8218 in Spearman’s ρ , and 0.8452 in Pearson’s r , outperforming the base Llama-3.1-8B-Instruct model by margins of 0.2009, 0.4339, 0.4555, and 0.4078 respectively. Similarly, the Qwen2.5-7B-Instruct-RewardPartial model shows substantial improvements over its base version. The consistent superiority across all correlation metrics and model families (Appendix A.5) demonstrates that the improvements are driven by the completion-potential training objective, not by scale or architecture alone. To assess the robustness of our approach, we also train RewardPartial variants across different model structures and sizes as shown in Appendix A.5.

3.3 Ablation Studies

We conduct two systematic ablation studies to validate key design choices in our approach.

Data Composition Ablation: To evaluate the impact of our multi-times completion strategy, we compare against a baseline using only the original UltraFeedback chosen response as partial responses’ completions estimation instead of multi-times completions and aggregation (RewardPartial-Origin in Table 3). While this simplified ver-

Model	Reward Model	Best-of-3		Our		Best-of-10		Our		Best-of-30		Our	
		lc win	win	lc win	win	lc win	win	lc win	win	lc win	win	lc win	win
Qwen2.5-1.5B	Llama8B	0.43	0.62	0.09	0.12	0.34	0.50	1.10	1.49	0.86	1.30	2.22	2.86
	Qwen7B	0.24	0.37	0.31	0.50	0.46	0.62	1.52	1.99	1.22	1.49	2.14	2.86
Qwen2.5-3B	Llama8B	7.45	6.83	8.24	7.83	9.09	8.51	9.76	9.44	10.22	10.25	11.00	11.06
	Qwen7B	7.72	7.20	7.23	7.45	9.75	8.82	9.73	10.06	11.86	11.30	11.77	12.05
Qwen2.5-7B	Llama8B	5.66	6.34	8.26	8.82	8.04	9.01	10.99	11.55	7.02	8.26	13.04	14.04
	Qwen7B	7.09	7.33	8.21	8.70	10.33	10.31	11.92	12.67	13.23	12.92	13.56	14.53
Qwen2.5-14B	Llama8B	11.66	8.76	16.96	13.54	17.18	13.78	19.30	16.52	18.80	15.22	21.83	18.76
	Qwen7B	16.65	11.80	16.34	13.17	18.90	13.79	19.14	16.52	24.26	18.07	24.09	20.99

Table 2: Complete performance comparison between different sampling methods across models. The table shows win rates (length control win rate and win rate, simplified for lc win and win in the table) for Best-of-N sampling versus our Truncated Best-of-N method under similar tokens usage as Algorithm 1, evaluated using two different reward models (Llama8B and Qwen7B stand for Llama3.1-8b-Instruct and Qwen2.5-7B-Instruct). Specifically, Llama3.1-8b-Instruct-RewardFull is used in Best-of-N and Llama3.1-8b-Instruct-RewardPartial is used in our method, same as Qwen2.5-7B-Instruct. All N values (3, 10, 30) are included.

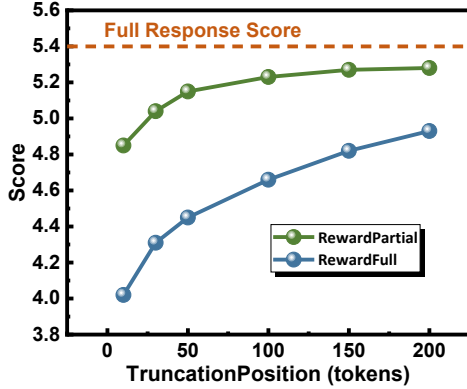


Figure 4: The scores of RewardPartial and RewardFull on partial responses truncated at different tokens length.

sion still outperforms the base model (0.6814 vs 0.5345 Exact Match), our full multi-times approach achieved superior results (0.7354), demonstrating that diverse completions provide more robust potential estimation. Furthermore, we introduce RewardPartial-Sample, which randomly samples truncation points rather than using our theoretically grounded selection strategy. Although RewardPartial-Sample outperforms RewardPartial-Origin (0.7020 vs 0.6814 Exact Match), it still falls short of RewardPartial across all metrics (e.g., 0.7020 vs 0.7354 Exact Match, 0.8100 vs 0.8452 Pearson’s r), confirming that our information-theoretic truncation point selection yields higher-quality training data than naive random sampling.

Truncation Position Analysis: As depicted in Figure 4, we evaluate how truncation position affects evaluation performance by testing segments at 10, 30, 50, 100, 150 and 200 tokens. RewardPar-

Model	Exact Match	τ	ρ	r
Llama-3.1-8B-Instruct	0.5345	0.3418	0.3663	0.4374
RewardPartial-Origin	0.6814	0.7372	0.7857	0.8227
RewardPartial-Sample	0.7020	0.7150	0.8010	0.8100
RewardPartial	0.7354	0.7757	0.8218	0.8452

Table 3: Evaluation Results on TruncEval test set. RewardPartial-Origin stands for RewardPartial trained with the data of original full response instead of multi completions and aggregation. RewardPartial-Sample denotes the variant where partial responses are randomly sampled during dataset construction without applying our theoretical framework for selecting informative truncation points. The metric τ , ρ and r mean Kendall’s τ , Spearman’s ρ and Pearson’s r .

tial consistently outperforms RewardFull across all truncation positions and demonstrate slower score decline when responses are truncated at an earlier stage. The analysis indicates that 100 tokens provide the ideal compromise between computational efficiency and scoring accuracy. At this length, RewardPartial can reliably assess completion potential without requiring excessive context.

4 Using RewardPartial in Real Alignment

To verify the effectiveness of RewardPartial’s paradigm for evaluating partial responses based on their completion potential, we conduct extensive experiments in decoding-time alignment scenarios. We mainly use AlpacaEval 2.0 to test the alignment performance due to its high consistency with human preference. In AlpacaEval 2.0, we use GPT-4o as a judge to make pairwise comparison of testing responses with GPT-4 Preview’ baseline

responses. The metrics used for evaluating the response quality are Length Control Win Rate and Win Rate. More evaluation details of AlpacaEval 2.0 can be seen in Appendix A.6.

4.1 Main Results

To validate the effectiveness of RewardPartial in practical decoding-time alignment scenarios, we conduct experiments under different setups to compare our Truncated Best-of-N approach with Best-of-N sampling under equivalent token budgets as in Algorithm 1. We test different Qwen2.5 series models under different scales of generated times N and rewarded by different RewardPartial/RewardFull.

As shown in Table 2, our method demonstrates the superiority of Truncated Best-of-N Approach with RewardPartial compared to Best-of-N. The experimental results demonstrate consistent advantages of our Truncated Best-of-N method over traditional Best-of-N sampling across multiple model scales and sampling budgets. As shown in Table 1, our approach achieves superior performance in 22 out of 24 comparative scenarios and metrics when evaluated by the Llama8B reward model, and in 19 out of 24 cases with the Qwen7B reward model. For example, Qwen2.5-14B achieves up to 4.8% absolute improvement in win rate (13.54% vs 8.76% win rate with Llama8B reward at Best-of-3), Qwen2.5-7B achieves up to 2.4% absolute improvement in lc win rate (10.31% vs 12.67% win rate with Qwen7B reward at Best-of-10).

Among the few evaluation results where Best-of-N outperforms our solution, most are under LC win rate. Our approach achieves superior performance in 23 out of 24 comparative scenarios on win rate than Best-of-N, which emphasizes that our Truncated Best-of-N with RewardPartial solution has better improvement in sentence quality instead of length control. Although the effect is slightly worse in length control, it is still better than Best-of-N with the same token consumption overall.

Direct Pairwise Quality Assessment: To further demonstrate the quality comparison of full sentence in Best-of-N and our method, we make a direct pairwise quality assessment and analysis. We use Qwen2.5-7B generations scored by Llama3.1-8B-Instruct-RewardPartial to accomplish this as Figure 5. Our method achieves a 50.2% win rate, 28.9% tie rate and 20.8% lose rate against best-of-3 sampling while maintaining comparable token consumption. The results under different value of N are same, showing the stable performance im-

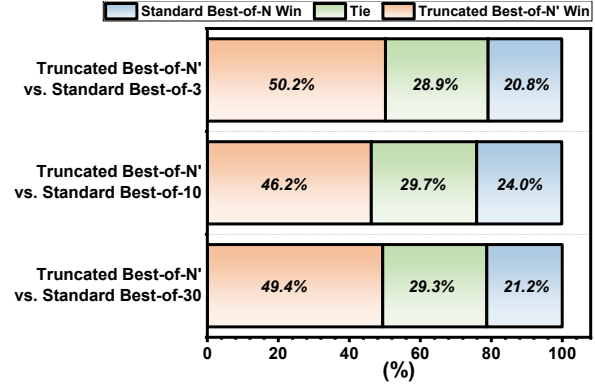


Figure 5: Pairwise comparison between Best-of-N sampling and our Truncated Best-of-N method using Qwen2.5-7B as generator and Llama3.1-8B as RM. The results demonstrate our method’s superior alignment performance compared with Best-of-N. For example, our method achieves 50.2% win rate versus 20.8% lose rate under similar token consumption when $N=3$.

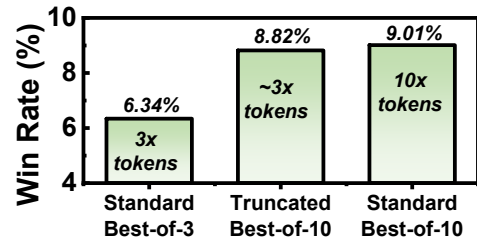


Figure 6: Token usage and performance comparison between different baselines.

provement of our method.

Human Validation: To verify that the gains in Figure 5 reflect genuine human preference rather than an artifact of the GPT-4o judge, we recruited three CS graduate students to perform blind pairwise evaluation on the same prompts and outputs used in the GPT-4o comparison, following an identical evaluation protocol. The annotators were compensated at a rate consistent with prevailing market standards. As shown in Table 4, majority-vote human judgments closely track the GPT-4o judgments across all three N settings (win/lose gaps less than 2 points), with 80-83% human-GPT-4o agreement and 84% inter-annotator agreement. This indicates that the improvement of RewardPartial-guided Truncated Best-of-N over Standard Best-of-N is supported by human preference, not judge-specific bias.

Tokens Usage and Performance Comparison: We use a practical case from Table 2 to analyze the difference of tokens usage and alignment performance. When we use Qwen2.5-7B model as generator to infer $N=3$ times, the average tokens usage of each prompt is 466 tokens. Following Algorithm 1

Setting	Judge	Win%	Tie%	Lose%
vs. BoN=3	GPT-4o	50.2	28.9	20.8
	Human	48.7	30.0	21.3
vs. BoN=10	GPT-4o	46.2	29.7	24.0
	Human	45.3	30.7	24.0
vs. BoN=30	GPT-4o	49.4	29.3	21.2
	Human	48.0	30.7	21.3

Table 4: Human vs. GPT-4o pairwise judgments of RewardPartial-guided Truncated Best-of-N against Standard Best-of-N. Human = majority vote of 3 annotators. Human-GPT-4o agreement: 82.7% / 80.0% / 81.3% for $N = 3/10/30$; inter-annotator agreement: 84%.

for pair comparison, the equivalent N' in Truncated Best-of- N' is 10 ($\lceil [466 \times 3 - (466 - 100)] / 100 \rceil$, truncated at 100 tokens). As shown in Figure 6, while Standard Best-of-3 achieves a 6.34% win rate, Truncated Best-of-10 reaches 8.82% with comparable token usage. In contrast, standard Best-of-10 requires 3.3 times tokens while delivering only marginal performance gains.

4.2 RewardPartial vs. RewardFull in Truncated Best-of-N

To isolate the contribution of RewardPartial from the Truncated Best-of-N framework itself, we conduct ablation studies comparing our approach with traditional RMs (RewardFull) in partial response evaluation. Our RewardPartial significantly outperforms RewardFull in Truncated Best-of-N in all model sizes and N, demonstrating its superior capability to evaluate partial responses, with details can be seen in Appendix A.4.

4.3 Generalization Testing on out-of-distribution Tasks

While TruncEval primarily focuses on open-domain questions, we extend our evaluation to mathematical reasoning tasks using GSM8K (Cobbe et al., 2021) and code tasks using HumanEval (Chen et al., 2021) to assess Truncated Best-of-N’s performance on out-of-distribution Tasks. As illustrated on Table 5, our experiments reveal that Truncated Best-of-N method with RewardPartial achieves competitive results, demonstrating its advantage over Best-of-N and RewardFull on Truncated Best-of-N. For example, the Qwen2.5-14B model reaches 40.3% accuracy with RewardPartial, outperforming both the Best-of-N baseline (33.6%) and the RewardFull variant (35.3%). These results demonstrate the generalization capability of our approach across

Model	Best-of-N=10	Truncated Best-of-N'	
	RewardFull	RewardPartial	RewardFull
<i>GSM8K (Accuracy %)</i>			
Qwen2.5-1.5B	3.2	3.9	3.6
Qwen2.5-7B	25.0	25.3	22.9
Qwen2.5-14B	33.6	40.3	35.3
<i>HumanEval (Pass@1 %)</i>			
Qwen2.5-1.5B	14.6	17.1	11.0
Qwen2.5-7B	33.5	39.6	33.5
Qwen2.5-14B	59.2	65.2	57.6

Table 5: Performance comparison on GSM8K and HumanEval of different baselines with similar tokens.

Method	lc win rate	win rate
Qwen2.5-7B (Base)	8.83	7.65
DPO-with-Standard-Best-of-N	14.86	12.80
DPO-with-Truncated-Best-of-N'	17.66	15.84

Table 6: Evaluation of DPO models on AlpacaEval 2.0 after training under different data constructed from Best-of-N and Truncated Best-of-N'.

different problem domains like code and math.

4.4 Application in Training Scenario

Despite Best-of-N is a training-free paradigm in Decoding-Time Alignment, it can also be used in specific training scenario like Direct Preference Optimization (DPO) where policy model generate multiple candidates and select the best and worst ones to construct Chosen-Rejected pairs. We also test the usage of Truncated Best-of-N with RewardPartial in this scenario and demonstrate the superiority of our method as in Table 6. Experiment details can be seen in Appendix A.7.

5 Related Works

5.1 Decoding-Time Alignment

Reinforcement Learning from Human Feedback (RLHF) has advanced language model alignment (Christiano et al., 2017; Lee et al., 2021; Ouyang et al., 2022), but suffers from limitations like degrading base capabilities and generalization (Zheng et al., 2024). This has spurred interest in decoding-time alignment methods. Best-of-N sampling, a foundational approach, selects the highest-reward response from multiple candidates but incurs high computational costs. Recent work explores alternatives like reward-guided search (Deng and Raffel, 2023), controlled decoding (Mudgal et al., 2024), and hybrid methods using Monte Carlo Tree Search (Browne et al., 2012). However, these often over-engineer, lack generalization, or struggle with consistent alignment. In contrast, we propose RewardPartial in Truncated

Best-of-N, a simpler yet more general and effective framework for decoding-time alignment.

5.2 Reward Model on Incomplete Text

Accurate reward evaluation for partially generated text is crucial for decoding-time alignment methods. However, conventional item-level reward models are designed to assess complete responses, leading to inaccurate assessments when applied to truncated text (Deng and Raffel, 2023; Khanov et al., 2024). While several approaches can be used to mitigate this inaccuracy in evaluating partial responses, they have significant drawbacks. Token-level RMs (Chen et al., 2024) offer intermediate output evaluation but suffer from training instability. Process reward models (PRMs) (Uesato et al., 2022), while capable of intermediate process assessment, are not designed with alignment tasks in mind. In contrast, we propose RewardPartial, which predicts the completion potential of truncated responses rather than their surface-level quality, naturally fitting the Truncated Best-of-N paradigm for decoding-time alignment.

ment in Truncated Best-of-N sampling. Future work could explore more efficient completion sampling strategies or distillation techniques to reduce this initialization cost without compromising evaluation quality.

6 Conclusion

We identify a critical limitation in decoding-time alignment that conventional reward models fail to accurately assess partial responses, fundamentally constraining the performance of efficient Truncated Best-of-N approaches. To address this, we propose a paradigm shift from surface-level assessment to predicting completion potential. Supported by theory demonstration, evaluation on our new benchmark TruncEval and experiments on real Truncated Best-of-N application, we show that RewardPartial enables more effective truncated sampling while remaining efficient. We hope this work inspires further exploration in efficient alignment.

7 Limitations

While our RewardPartial approach demonstrates strong performance across diverse evaluation scenarios, one current limitation lies in its computational overhead during the data construction phase. The Monte Carlo estimation process requires generating multiple completions for each partial response, which incurs greater initial computational costs compared to conventional reward modeling approaches. However, this overhead is a one-time cost during dataset creation and is substantially offset by the efficiency gains during actual deploy-

References

- Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfschagen, Stephen Tavenor, Diego Perez, Spyridon Samothrakis, and Simon Colton. 2012. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. [Evaluating large language models trained on code](#).
- Xiushi Chen, Gaotang Li, Ziqi Wang, Bowen Jin, Cheng Qian, Yu Wang, Hongru Wang, Yu Zhang, Denghui Zhang, Tong Zhang, Hanghang Tong, and Heng Ji. 2025. [Rm-r1: Reward modeling as reasoning](#). *Preprint*, arXiv:2505.02387.
- Zhipeng Chen, Kun Zhou, Wayne Xin Zhao, Junchen Wan, Fuzheng Zhang, Di Zhang, and Ji-Rong Wen. 2024. Improving large language models via fine-grained reinforcement learning with minimum editing constraint. *CoRR*.
- Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Bingxiang He, Wei Zhu, Yuan Ni, Guotong Xie, Ruobing Xie, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2023. [UltraFeedback: Boosting Language Models with Scaled AI Feedback](#). *arXiv e-prints*, arXiv:2310.01377.
- Paul Dagum, Richard Karp, Michael Luby, and Sheldon Ross. 2000. An optimal algorithm for monte carlo estimation. *SIAM Journal on computing*, 29(5):1484–1496.
- Haikang Deng and Colin Raffel. 2023. Reward-augmented decoding: Efficient controlled text generation with a unidirectional reward model. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.
- Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto. 2024. Length-controlled alpaca-eval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 181 others. 2024. [The Llama 3 Herd of Models](#). *arXiv e-prints*, arXiv:2407.21783.
- Lin Gui, Cristina Gârbacea, and Victor Veitch. 2024. [Bonbon alignment for large language models and the sweetness of best-of-n sampling](#). *Preprint*, arXiv:2406.00832.
- Marc Harary. 2024. [Efficient algorithms for the sensitivities of the pearson correlation coefficient and its statistical significance to online data](#). *Preprint*, arXiv:2405.14686.
- 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*.
- Zhewei Kang, Xuandong Zhao, and Dawn Song. 2025. [Scalable best-of-n selection for large language models via self-certainty](#). *Preprint*, arXiv:2502.18581.
- Maxim Khanov, Jirayu Burapachee, and Yixuan Li. 2024. Args: Alignment as reward-guided search. In *Proceedings of the International Conference on Learning Representations*.
- Kimin Lee, Laura Smith, and Pieter Abbeel. 2021. Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. In *International Conference on Machine Learning*.
- Bolian Li, Yifan Wang, Anamika Lochab, Ananth Grama, and Ruqi Zhang. 2024. [Cascade Reward Sampling for Efficient Decoding-Time Alignment](#). *arXiv e-prints*, arXiv:2406.16306.
- Fengpei Li, Haoxian Chen, Jiahe Lin, Arkin Gupta, Xiaowei Tan, Honglei Zhao, Gang Xu, Yuriy Nevmyvaka, Agostino Capponi, and Henry Lam. 2025. [Prediction-enhanced monte carlo: A machine learning view on control variate](#). *Preprint*, arXiv:2412.11257.
- Junlong Li, Shichao Sun, Weizhe Yuan, Run-Ze Fan, Hai Zhao, and Pengfei Liu. 2023. Generative judge for evaluating alignment. *arXiv preprint arXiv:2310.05470*.
- Dakota Mahan, Duy Van Phung, Rafael Rafailov, Chase Blagden, Nathan Lile, Louis Castricato, Jan-Philipp Fränken, Chelsea Finn, and Alon Albalak. 2024. Generative reward models. *arXiv preprint arXiv:2410.12832*.
- Sidharth Mudgal, Jong Lee, Harish Ganapathy, YaGuang Li, Tao Wang, Yanping Huang, Zhifeng Chen, Heng-Tze Cheng, Michael Collins, Trevor

- Strohman, and 1 others. 2024. Controlled decoding from language models. *International Conference on Machine Learning*.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.
- Philip Sedgwick. 2012. Pearson’s correlation coefficient. *Bmj*, 345.
- Luohe Shi, Yao Yao, Zuchao Li, Lefei Zhang, and Hai Zhao. 2024a. Reference trustable decoding: A training-free augmentation paradigm for large language models. *Advances in Neural Information Processing Systems*, 37:80034–80055.
- Ruizhe Shi, Yifang Chen, Yushi Hu, Alisa Liu, Hannaneh Hajishirzi, Noah A. Smith, and Simon S. Du. 2024b. [Decoding-time language model alignment with multiple objectives](#). *Preprint*, arXiv:2406.18853.
- Hanshi Sun, Momin Haider, Ruiqi Zhang, Huitao Yang, Jiahao Qiu, Ming Yin, Mengdi Wang, Peter Bartlett, and Andrea Zanette. 2024. Fast best-of-n decoding via speculative rejection. *arXiv preprint arXiv:2410.20290*.
- Team GLM, :, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, Hao Yu, Hongning Wang, Jiadai Sun, Jiajie Zhang, Jiale Cheng, Jiayi Gui, Jie Tang, and 40 others. 2024. [ChatGLM: A Family of Large Language Models from GLM-130B to GLM-4 All Tools](#). *arXiv e-prints*, arXiv:2406.12793.
- Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, L. Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. [Solving math word problems with process- and outcome-based feedback](#). *ArXiv*, abs/2211.14275.
- Claudio Mayrink Verdun, Alex Oesterling, Himabindu Lakkaraju, and Flavio P Calmon. 2025. Soft best-of-n sampling for model alignment. *arXiv preprint arXiv:2505.03156*.
- Xuan Wang, Zeshan Pang, Yuliang Lu, and Xuehu Yan. 2024. [Mge: A training-free and efficient model generation and enhancement scheme](#). *Preprint*, arXiv:2402.17486.
- An Yang, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoyan Huang, Jiandong Jiang, Jianhong Tu, Jianwei Zhang, Jingren Zhou, and 1 others. 2025. Qwen2. 5-1m technical report. *arXiv preprint arXiv:2501.15383*.
- Kevin Yang and Dan Klein. 2021. Fudge: Controlled text generation with future discriminators. *arXiv preprint arXiv:2104.05218*.
- Zihuiwen Ye, Fraser Greenlee-Scott, Max Bartolo, Phil Blunsom, Jon Ander Campos, and Matthias Gall  . 2024. [Improving reward models with synthetic critiques](#). *Preprint*, arXiv:2405.20850.
- Luca Zanella, Willi Menapace, Massimiliano Mancini, Yiming Wang, and Elisa Ricci. 2024. [Harnessing large language models for training-free video anomaly detection](#). *Preprint*, arXiv:2404.01014.
- Chen Zheng, Ke Sun, Hang Wu, Chenguang Xi, and Xun Zhou. 2024. [Balancing Enhancement, Harmlessness, and General Capabilities: Enhancing Conversational LLMs with Direct RLHF](#). *arXiv e-prints*, arXiv:2403.02513.
- Xiaosen Zheng, Tianyu Pang, Chao Du, Qian Liu, Jing Jiang, and Min Lin. 2025. [Cheating automatic llm benchmarks: Null models achieve high win rates](#). *Preprint*, arXiv:2410.07137.

A Appendices.

A.1 Empirical Analysis of RM on full and truncated responses

We conduct an empirical experiment to analyze how RM performs on full and truncated responses. From UltraFeedback-Binarized which contains 60,000 prompts and corresponding chosen responses, we first filter chosen responses with lengths exceeding 200 tokens using the Qwen2.5-7B tokenizer and randomly retain 1000 items, then truncate each selected response to its first 20/100/200 tokens. We then use 7 different models (Qwen/Llama/GLM series) to complete the truncated responses five times and use different RM (GPT4.1, AutoJ-13B and ArmoRM-Llama3-8B) to score them. Here we exclude models like Qwen1.5-7B that lack basic dialogue completion capabilities, as they cannot generate coherent partial responses prior to truncation, making them unsuitable for this experiment.

Our results reveal two key observations. First, as illustrated in Figure 7, when different LLMs are used to complete 100-token truncated responses, the average scores assigned by different RMs exhibit remarkable consistency and remain closely aligned with those of the original full-length responses. Notably, truncated responses evaluated without any completion yield significantly lower scores compared to their completed counterparts. Second, as illustrated in Figure 8, we observe that the degree of alignment between completed and original responses varies depending on the truncation length (20, 100, or 200 tokens). When responses are truncated at 200 tokens, the completions exhibit minimal deviation in scores, closely

matching the full responses. In contrast, 20-token truncations lead to noticeably higher divergence in the completed outputs. Notably, 100 tokens emerges as an optimal trade-off, effectively balancing token efficiency with low completion bias across models.

A.2 Theoretical Analysis of RewardPartial

The core theoretical foundation of RewardPartial lies in modeling language generation as a Markov Decision Process (MDP), which provides rigorous mathematical justification for our completion potential metric.

MDP Formulation of Language Generation

We define the language generation process as an MDP with the following components:

- **State** (s_t): The partially generated response $\{w_1, w_2, \dots, w_t\}$ at step t
- **Action** (a_t): Selection of next token $w_{t+1} \sim \pi(\cdot|s_t)$
- **Transition**: Deterministic state update $s_{t+1} = s_t \oplus a_t$
- **Reward**: Terminal reward $R(s_T)$ from RM, with zero intermediate rewards
- **Discount factor** ($\gamma = 1$): No temporal discounting in generation

The *completion potential* $V(s_t)$ is formally defined as the state value function:

$$V(s_t) = E_\pi[R(s_T)|s_t] \quad (5)$$

Dynamic Programming Property The completion potential satisfies the Bellman equation for MDPs:

Theorem 1 (Bellman Equation for Completion Potential). *The value function $V(s_t)$ satisfies:*

$$V(s_t) = E_{a_t \sim \pi(\cdot|s_t)}[V(s_{t+1})] \quad (6)$$

Expanding the right-hand side:

$$\begin{aligned} E_{a_t}[V(s_{t+1})] &= \sum_{a_t} \pi(a_t|s_t) \cdot E_\pi[R(s_T)|s_{t+1} = s_t \oplus a_t] \\ &= E_\pi[R(s_T)|s_t] \\ &= V(s_t) \end{aligned} \quad (7)$$

Monte Carlo Estimation RewardPartial estimates $V(s_t)$ through Monte Carlo sampling:

$$\hat{V}(s_t) = \frac{1}{N} \sum_{i=1}^N R(s_T^{(i)}) \quad (8)$$

Theorem 2 (Unbiased Estimation). *The Monte Carlo estimator $\hat{V}(s_t)$ is unbiased:*

$$E[\hat{V}(s_t)] = V(s_t) \quad (9)$$

By linearity of expectation and the definition in (5):

$$\begin{aligned} E[\hat{V}(s_t)] &= E\left[\frac{1}{N} \sum_{i=1}^N R(s_T^{(i)})\right] \\ &= \frac{1}{N} \sum_{i=1}^N E[R(s_T^{(i)})|s_t] \\ &= V(s_t) \end{aligned} \quad (10)$$

Practical Convergence Analysis While the Monte Carlo estimator $\hat{V}_N(s_t)$ is theoretically guaranteed to converge to the true value $V(s_t)$ almost surely as $N \rightarrow \infty$, practical implementations must account for finite-sample estimation errors. A commonly used empirical criterion for assessing estimation stability is when the coefficient of variation (CV) satisfies:

$$CV = \frac{\sigma_N(s_t)}{|\mu_N(s_t)|} < \epsilon \quad (11)$$

where $\mu_N(s_t) = E[\hat{V}_N(s_t)]$ denotes the sample mean and $\sigma_N(s_t) = \sqrt{\text{Var}(\hat{V}_N(s_t))}$ represents the sample standard deviation. Here, ϵ is a predefined threshold that balances accuracy and computational efficiency.

The choice of ϵ varies across different domains depending on the required precision and resource constraints. In some applications, stricter thresholds (e.g., $\epsilon = 0.05$) are adopted for higher accuracy, while looser thresholds (e.g., $\epsilon = 0.30$) may suffice in scenarios prioritizing computational speed. In our analysis, we set $\epsilon = 0.15$, a commonly used intermediate value that ensures statistically stable estimation while maintaining reasonable efficiency.

A.3 Computational Efficiency of RewardPartial vs. RewardFull

While RewardPartial requires N' scoring calls to select the same number of final candidates that RewardFull selects with N calls under Standard Best-of- N ($N' > N$, following Algorithm 1), each RewardPartial call scores a truncated prefix rather than a full response, so its input is considerably shorter than a RewardFull call's input. The output-side saving from the "Rating: [[5]]" prompt format applies equally to both methods and by itself

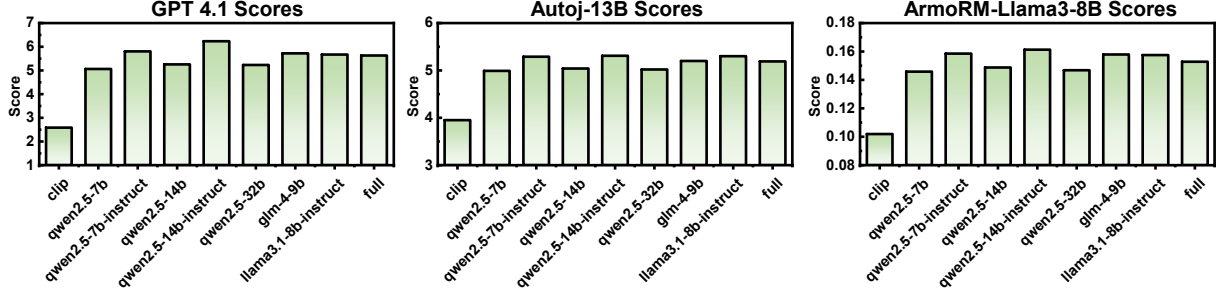


Figure 7: Comparison of RM scores for original full/clip (truncated at 100 tokens) responses vs. completed responses from clip across different LLMs. Truncated responses without completion receive significantly lower scores, while completed versions closely match the scores of full responses across 3 different types of RMs.

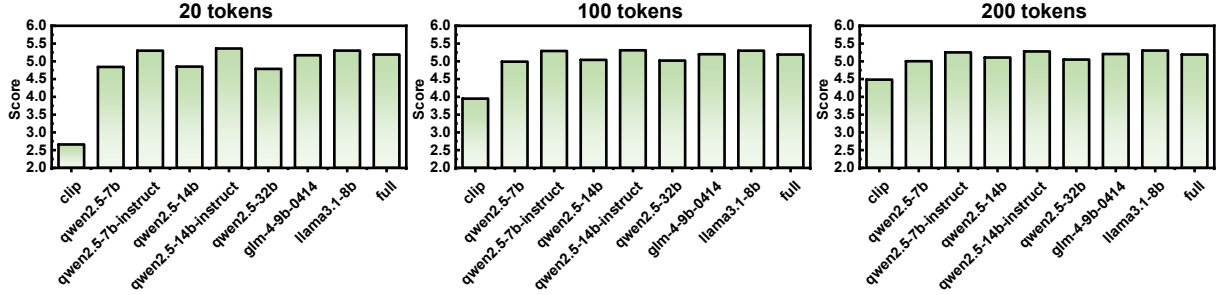


Figure 8: Comparison of RM scores for original full/clip (truncated at 20/100/200 tokens) responses vs. completed responses from clip across different LLMs. Longer truncations (200 tokens) yield minimal score deviation, while shorter ones (20 tokens) show higher divergence. 100 tokens provides an optimal balance between efficiency and low completion bias.

does not explain whether the increased number of calls raises overall cost. We therefore directly measure end-to-end computational cost rather than relying on token counts alone.

Setup. We use Qwen2.5-7B as the generator and evaluate on the 805 AlpacaEval 2.0 prompts. Following Algorithm 1, $N=10$ under Standard Best-of- N corresponds to $N'=40$ under Truncated Best-of- N' for the same final selection budget. We run RM scoring for RewardFull (805×10 full-response calls) and RewardPartial (805×40 truncated-prefix calls) on the same single 80GB GPU with identical batched-inference settings and automatic prefix caching enabled, repeating for two RM backbones (Llama-3.1-8B and Qwen2.5-7B). We report the number of scoring calls, total input/output tokens processed, cache-effective FLOPs (i.e., accounting for compute actually skipped on cache hits), wall-clock latency, peak GPU memory, and the prefix-cache hit rate observed during scoring.

Result. RewardPartial issues $4.00\times$ more scoring calls, and because each call’s shorter truncated-prefix input does not fully offset this increase, total input tokens processed rise by $1.39\text{--}1.43\times$ and cache-effective FLOPs rise modestly by $1.07\times$.

Despite this, wall-clock latency *decreases* ($0.84\text{--}0.88\times$) and peak GPU memory stays essentially unchanged ($0.99\text{--}1.00\times$). The reason is that RewardPartial’s scoring prompt is dominated by the fixed instruction template rather than the (short, truncated) response itself, so a much larger fraction of each call’s input is byte-identical across calls; this drives prefix-cache hit rates up by $3.1\text{--}3.2\times$ ($11.4\%\rightarrow 35.5\%$ for Llama-3.1-8B, $10.0\%\rightarrow 32.2\%$ for Qwen2.5-7B) relative to RewardFull, where the scored full responses are longer and more mutually distinct. The resulting cache reuse removes enough redundant computation that it more than compensates for the raw increase in calls and tokens, so switching to RewardPartial does not slow down, and if anything speeds up, the RM side of the Truncated Best-of- N' pipeline.

A.4 Details about RewardFull

In the Best-of- N paradigm, a reward model (RM) is required to score and select the best candidate among all responses. To ensure fair pairwise comparisons between Truncated Best-of- N (evaluating truncated responses) and Best-of- N (evaluating full responses), we construct training data using the TruncEval training set, replacing truncated re-

RM Backbone	Metric	RewardFull ($N=10$)	RewardPartial ($N'=40$)	Ratio (P/F)
Llama-3.1-8B	Scoring Calls	8,050	32,200	$4.00\times$
	Input Tokens	4,175,129	5,955,753	$1.43\times$
	Output Tokens	40,250	161,000	$4.00\times$
	FLOPs (cache-effective)	6.0×10^{16}	6.4×10^{16}	$1.07\times$
	Wall-clock Latency	411 s	344 s	$0.84\times$
	Peak GPU Memory	68.0 GB	68.0 GB	$1.00\times$
	Prefix-cache Hit Rate	11.4%	35.5%	$3.1\times$
Qwen2.5-7B	Scoring Calls	8,050	32,200	$4.00\times$
	Input Tokens	4,168,908	5,799,163	$1.39\times$
	Output Tokens	40,240	160,998	$4.00\times$
	FLOPs (cache-effective)	5.8×10^{16}	6.2×10^{16}	$1.07\times$
	Wall-clock Latency	374 s	330 s	$0.88\times$
	Peak GPU Memory	68.4 GB	67.8 GB	$0.99\times$
	Prefix-cache Hit Rate	10.0%	32.2%	$3.2\times$

Table 7: End-to-end computational cost of RewardFull under Standard Best-of- N ($N=10$) vs. RewardPartial under the matched-budget Truncated Best-of- N' ($N'=40$; Qwen2.5-7B generator, AlpacaEval 2.0, single 80GB GPU, batched inference with automatic prefix caching). Ratio (P/F) = RewardPartial value divided by RewardFull value.

Model	N	RewardFull		RewardPartial (our)	
		lc win rate	win rate	lc win rate	win rate
Qwen2.5-1.5B	3	0.24	0.37	0.09	0.12
	10	0.91	1.24	1.10	1.49
	30	1.80	2.24	2.22	2.86
Qwen2.5-3B	3	7.39	6.96	8.24	7.83
	10	8.07	7.20	9.76	9.44
	30	7.29	6.83	11.00	11.06
Qwen2.5-7B	3	7.64	7.95	8.26	8.82
	10	7.81	8.45	10.99	11.55
	30	9.69	9.94	13.04	14.04
Qwen2.5-14B	3	16.61	12.17	16.96	13.54
	10	18.42	13.29	19.30	16.52
	30	19.58	14.53	21.83	18.76

Table 8: Performance comparison between RewardFull and RewardPartial (our) methods across different models and N values. Both RewardFull and RewardPartial are finetuned from Llama3.1-8B-Instruct here.

sponses with their corresponding full-length counterparts, as illustrated in the Figure 9. RewardFull models are also fine-tuned via Supervised Fine-tuning (SFT). For fair comparison in practical application, if RewardPartial used in Best-of-N is trained using Llama3.1-8B-Instruct, we will use the RewardFull derived from the Llama3.1-8B-Instruct as well.

To demonstrate the contribution of RewardPartial from the Truncated Best-of-N framework itself, we compare the performance of Truncated Best-of-N framework under different Reward Models of RewardPartial and RewardFull. As the table 8 shows, only with RewardPartial can Truncated Best-of-N achieve really good performance.

A.5 Model Generalization on RewardPartial

To test the generalization of RewardPartial, we train more RewardPartial models from different

	Exact Match	Kendall's τ	Spearman's ρ	Pearson's r
Llama-3.1-8B-Instruct-RewardPartial	0.7354	0.7757	0.8218	0.8452
Qwen2.5-3B-Instruct-RewardPartial	0.7150	0.7516	0.8019	0.8147
Qwen2.5-7B-Instruct-RewardPartial	0.7278	0.7572	0.8017	0.8351
Qwen2.5-14B-Instruct-RewardPartial	0.7573	0.7922	0.8354	0.8581
Llama-3.1-8B-Instruct	0.5345	0.3418	0.3663	0.4374
Qwen2.5-3B-Instruct	0.3800	0.2226	0.2428	0.2514
Qwen2.5-7B-Instruct	0.3789	0.3759	0.4300	0.5272
Qwen2.5-14B-Instruct	0.2171	0.3427	0.3950	0.3782

Table 9: Evaluation Results on TruncEval test set. The metric τ , ρ and r mean Kendall's τ , Spearman's ρ and Pearson's r .

Instruct models of different models structures and scales. We further test them on TruncEval test set and demonstrate their effectiveness on evaluating truncated responses as shown in Table 9.

A.6 Evaluation Details of AlpacaEval 2.0

AlpacaEval 2.0 is an automatic evaluator for instruction-following language models and has been verified with high correlation with human preference. The metrics of AlpacaEval 2.0 contains win-rates and length-control win-rates, both of which used by default to test LLMs ability. AlpacaEval 2.0 contains 805 prompts with GPT-4-Preview's answers as default baseline. It evaluates model by measuring the fraction of times a powerful LLM (here we use GPT-4o) prefers the outputs from that model over baseline.

In practical evaluation, we also make a direct preference comparison of outputs from Best-of-N and Truncated Best-of-N, the pairwise prompt we use is shown in Figure 10.

```

Evaluate a response to a user's query. First
provide a rating score, then write your critiques.

[BEGIN DATA]
***
[Query]: {Query}
***
[Response]: {Response}
***
[END DATA]

First provide your final rating for the response
on a scale of 1 to 10 by strictly following this
format: "[rating]", for example: "Rating: [[5]]".

Then
Write "[Judgment]" and your critiques of both
the original and completed portions.

```

Figure 9: The prompt of RewardFull used in training data and practical application.

```

SYSTEM_PROMPT = "Please act as an impartial judge and
evaluate the quality of the responses provided by two AI
assistants to the user question displayed below. You should
choose the assistant that follows the user's instructions and
answers the user's question better. Your evaluation should
consider factors such as the helpfulness, relevance, accuracy,
depth, creativity, and level of detail of their responses. Begin
your evaluation by comparing the two responses and provide a
short explanation. Avoid any position biases and ensure that
the order in which the responses were presented does not
influence your decision. Do not allow the length of the
responses to influence your evaluation. Do not favor certain
names of the assistants. Be as objective as possible. After
providing your explanation, output your final verdict by strictly
following this format: \"[[A]]\" if assistant A is better, \"[[B]]\" if
assistant B is better, and \"[[C]]\" for a tie."

USER_PROMPT = "
[User Question]
{question}

[The Start of Assistant A's Answer]
{Answer_A}
[The End of Assistant A's Answer]

[The Start of Assistant B's Answer]
{Answer_B}
[The End of Assistant B's Answer]"

```

Figure 10: The prompt of pairwise comparison with position exchange.

A.7 Application Details on DPO

Although Best-of-N is typically a training-free paradigm for decoding-time alignment, it can also be adapted for specific training scenarios, such as Direct Preference Optimization (DPO). In this setup, the policy model generates multiple candidate responses, from which the best and worst are selected to construct chosen-rejected pairs.

Our experimental setup is as follows: We randomly sample 2,000 prompts from UltraFeedback-Binarized, ensuring none overlap with those in TruncEval. Using Qwen-2.5-7B as the policy model, we generate 4 ($N = 4$) candidate responses per prompt and rank them using Reward-Full (Llama3.1-8B-Instruct-RewardFull) to identify the best and worst candidates. For comparison, we also generate N' truncated responses (stopping at 100 tokens), score them using RewardPartial

(Llama3.1-8B-Instruct-RewardPartial), and then complete the highest-scoring and lowest-scoring truncated candidates to form additional DPO training data. To ensure pair comparisons, we calculate the value of N' through total tokens consumption in Best-of-N to strictly control the total token consumption across both processes. For DPO training, the models are trained for 1 epoch with a learning rate of 5.0×10^{-7} , $\beta = 0.1$ and a global batch size of 32 (4 per device * 8 GPUs). The training use cosine learning rate scheduling with 10% warmup ratio, maintaining the maximum sequence length of 2048 tokens and with a maximum prompt length of 512 tokens.