A Critical Look At Tokenwise Reward-Guided Text Generation

Ahmad Rashid [1 2] Ruotian Wu [1] Julia Grosse [3] Agustinus Kristiadi [* 2] Pascal Poupart [* 1 2]

Abstract

Large language models (LLMs) can significantly be improved by aligning to human preferences—the so-called reinforcement learning from human feedback (RLHF). However, the cost of fine-tuning an LLM is prohibitive for many users. Tokenwise reward-guided text generation (RGTG) methods have recently been proposed, which, can bypass LLM finetuning. They use a reward model trained on full sequences to score partial sequences during tokenwise decoding, to steer the generation towards sequences with high rewards. However, these methods have so far been only heuristically motivated and poorly analyzed. In this work, we show that reward models trained on full sequences are not compatible with scoring partial sequences. To alleviate this issue, we propose to explicitly train a Bradley-Terry reward model on partial sequences, and autoregressively sample from the implied tokenwise policy during decoding. We study properties of this reward model and the implied policy. Particularly, we show that this policy is proportional to the ratio of two distinct RLHF policies. We show that our simple approach outperforms previous RGTG methods and achieves similar performance as strong offline baselines but without large-scale LLM finetuning.

1. Introduction

Large language models (LLMs) provide a modern foundation for most if not all text generation tasks (Radford et al., 2019; Brown et al., 2020; Touvron et al., 2023a;b). In practice, significant improvements in the quality of text generation are achieved by aligning LLMs to human preferences (Stiennon et al., 2020b; Ouyang et al., 2022). This is typically done by reinforcement learning from human feedback (RLHF), which involves two steps: i) learning a reward model from preference data and ii) fine-tuning an LLM to maximize expected rewards by reinforcement learning (Ziegler et al., 2019b). Usually, this is done via a reinforcement learning algorithm such as proximal policy optimization (PPO, Schulman et al., 2017). Nevertheless, recently, Rafailov et al. (2023) showed that the reward modeling step (i) can be bypassed by directly fine-tuning an LLM with preference data, resulting in a method called direct preference optimization (DPO). While this simplifies RLHF, the fine-tuning step (ii) remains prohibitively costly for most users since it requires high-performance computational resources with large GPUs.

In order to alleviate the computational issue above, Khanov et al. (2024); Deng and Raffel (2023) explored tokenwise reward-guided text generation (RGTG) techniques that avoid any fine-tuning of the LLM. Instead a reward model is used to adjust the softmax scores. Since reward models can be significantly smaller than the LLM, training one is much cheaper than fine-tuning the LLM. Furthermore, reward models are modular: they can easily be composed and reused without any cost to guide text generation in conjunction with any base LLM. In contrast, RLHF via DPO requires fine-tuning of every LLM that we wish to improve.

While RGTG is an interesting alternative to offline RLHF, it is often based on heuristics and still poorly analyzed. For instance, ARGS (Khanov et al., 2024) proposed to use a reward model trained on full sequences to score each partial sequence during decoding. Deng and Raffel (2023) used a custom tokenwise loss to distill a reward model trained on full sequences. It is unclear if these approaches can lead to a sound tokenwise text generation policy.

In this work, we analyze this common RGTG approach. First, we show that the usage of full-sequence reward models for scoring partial sequences in a tokenwise policy is pathological. To alleviate this, we propose to explicitly train a Bradley-Terry (B-T) reward model on partial sequences and analyze their property. We prove that this text generation policy is a ratio of two different RLHF policies trained on sequences of different lengths. By deriving the sampling policy from a ratio of distinct RLHF policies, we obtain a tractable sampling procedure. We empirically validate our analysis on two different text generation datasets on two recent LLMs. Evidence shows that our approach achieves better performance compared to ARGS, matching the performance of the more expensive, offline PPO and DPO.

[*] Equal supervision [1] University of Waterloo [2] Vector Institute [3] University of Tübingen. Correspondence to: Ahmad Rashid <a9rashid@uwaterloo.ca>.

Published at ICML 2024 Workshop on Foundation Models in the Wild.

2. Preliminaries

We denote a prompt by $\mathbf{x}$ and its response by $\mathbf{y}$ where the bolded letters indicate sequences of tokens. The i-th token in $\mathbf{x}$ is denoted by x^i, while the partial sequence starting at token i and ending at token j is denoted by $\mathbf{x}^{i:j}$. The length of a sequence $\mathbf{x}$ is denoted by $|\mathbf{x}|$.

LLMs generally consist of probabilistic models that can generate a response $\mathbf{y}$ given a prompt $\mathbf{x}$, token-by-token, by sampling the next token from a conditional distribution $\pi(y^i|\mathbf{x}, \mathbf{y}^{1:i-1})$. Given a preference dataset $\mathcal{D} = \{(\mathbf{x}_k, \mathbf{y}_{wk}, \mathbf{y}_{lk})\}_{k=1}^{K}$ containing K triples of token sequences $(\mathbf{x}, \mathbf{y}_w, \mathbf{y}_l)$, Ziegler et al. (2019b); Ouyang et al. (2022) proposed a technique based on reinforcement learning (RL) to align an LLM with the preference dataset. They train a parametric reward model $r_\phi(\mathbf{y}|\mathbf{x})$ that assigns a higher score to the "winning" (i.e., preferred) utterance $\mathbf{y}_w$ and a lower score to the "losing" utterance $\mathbf{y}_l$. This is done via the B-T model (Bradley and Terry, 1952) which minimizes the loss:

$$\mathcal{L}_R = -\mathop{\mathbb{E}}_{\mathbf{x},\mathbf{y}_w,\mathbf{y}_l\sim\mathcal{D}} \log\sigma(r_\phi(\mathbf{y}_w|\mathbf{x}) - r_\phi(\mathbf{y}_l|\mathbf{x})), \quad (1)$$

where σ is the logistic function. Note that r_ϕ is trained to score entire utterances $\mathbf{y}$. Once r_ϕ is trained, it can be used to infer the probability of generating sequence $\mathbf{y}$ in response to $\mathbf{x}$, i.e., $P_\phi(\mathbf{y}|\mathbf{x}) = {\exp(r_\phi(\mathbf{y}|\mathbf{x}))}/{\sum_{\mathbf{y}'}\exp(r_\phi(\mathbf{y}'|\mathbf{x}))}$. Given a reference LLM, we denote by $\pi_{\text{ref}}(\mathbf{y}|\mathbf{x})$ the conditional probability that it will generate response $\mathbf{y}$ to prompt $\mathbf{x}$. We refer to the LLM and its policy interchangeably. One can then copy the LLM and finetune it to maximize

$$\max_\theta \mathop{\mathbb{E}}_{\substack{\mathbf{x}\sim\mathcal{D},\\ \mathbf{y}\sim\pi_\theta(\mathbf{y}|\mathbf{x})}} [r_\phi(\mathbf{y}|\mathbf{x})] - \beta D_{\text{KL}}[\pi_\theta(\mathbf{y}|\mathbf{x}) \,\|\, \pi_{\text{ref}}(\mathbf{y}|\mathbf{x})], \quad (2)$$

where the KL term forms a regularizer that ensures that the finetuned model does not differ much from the reference model. The above optimization problem can be optimized by many RL techniques, including the popular proximal policy optimization (PPO) algorithm (Schulman et al., 2017). This RL optimization is quite costly in practice due to the size of the LLM. Equation (2) has a closed form solution of the form (Peters and Schaal, 2007)

$$\pi_\theta(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})}\pi_{\text{ref}}(\mathbf{y}|\mathbf{x})\exp(\beta r_\phi(\mathbf{y}|\mathbf{x})) \quad (3)$$

where $Z(\mathbf{x}) = \sum_{\mathbf{y}} \pi_{\text{ref}}(\mathbf{y}|\mathbf{x})\exp(\beta r_\phi(\mathbf{y}|\mathbf{x}))$ is the intractable partition function.

Khanov et al. (2024) proposed reward-guided text generation (RGTG) techniques that do not require any LLM finetuning, but can obtain sequences $\mathbf{y}$ with high reward. This is done by freezing the reference LLM π_{ref} and at decoding time, the next-token probability $\pi_{\text{ref}}(y^i \mid \mathbf{x}, \mathbf{y}^{1:i-1})$ is adjusted by a reward model r_ϕ. More specifically, possible values for y^i are scored by a weighted combination of logits of π_{ref} and the rewards:

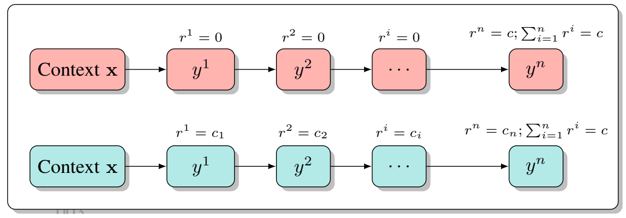

Figure 1. We denote $r^i = r(y^i|\mathbf{x}, \mathbf{y}^{1:i-1})$. While the total reward over the full sequence $\mathbf{y} = (y^1, \ldots, y^n)$ might be $c \neq 0$, it could be in the extreme case that the values over previous partial sequences are all zero—this is a perfectly valid result for a sequence-level reward model **(top)**. This may result in *unguided* decoding despite using *reward-guided* decoding. By training r on partial sequences, we can avoid this issue **(bottom)**.

$$\text{score}(y^i|\mathbf{x}, \mathbf{y}^{1:i-1}) = \log\pi_{\text{ref}}(y^i|\mathbf{x}, \mathbf{y}^{1:i-1}) + \beta r_\phi(\mathbf{y}^{1:i}|\mathbf{x}).$$

Khanov et al. (2024) do train the reward model with the B-T loss, but it is trained only with complete sequences, i.e. $r_\phi(\mathbf{y}|\mathbf{x})$, while it is used to score partial sequences, i.e. $r_\phi(\mathbf{y}^{1:i}|\mathbf{x})$. Hence, it is unclear whether the inferred scores for partial sequences are reasonable. A further discussion on reward-guided text generation and direct preference optimization can be found in Appendix A.

3. Pitfalls of RGTG and How to Fix Them

First, we start by analyzing the partial sequence rewards inferred from a reward model trained with full sequences only. Proof in Appendix B.1.

Theorem 3.1. *Training r to minimize the B-T loss (1) on full sequences $\mathbf{y}^{1:|\mathbf{y}|}$ may assign arbitrary rewards to partial sequences $\mathbf{y}^{1:i}$ (where $i < |\mathbf{y}|$). More precisely, $r(\mathbf{y}^{1:i}|\mathbf{x}) = v_{\mathbf{x},\mathbf{y}^{1:i}}$ where $v_{\mathbf{x},\mathbf{y}^{1:i}} \in \mathbb{R}$ can be any value.*

This leads to an unidentifiability problem (see Fig. 1). If we learn a reward model based on preferences over full sequences only as proposed by Khanov et al. (2024); Deng and Raffel (2023), then we may not obtain adequate rewards for partial sequences. As a concrete example, suppose r is a reward model such that (Fig. 1)

$$r(y^i|\mathbf{x}, \mathbf{y}^{1:i-1}) = \begin{cases} r(\mathbf{y}|\mathbf{x}) & i = |\mathbf{y}| \\ 0 & i < |\mathbf{y}|. \end{cases}$$

This reward model satisfies $\sum_i r(y^i|\mathbf{x}, \mathbf{y}^{1:i-1}) = r(\mathbf{y}|\mathbf{x})$ and therefore could be the solution when minimizing the B-T loss (1). If we use this reward model to sample from the induced RLHF optimal policy in (3), then the token level sampling distribution is the same as for the reference LLM $\pi_{\text{ref}}(y^i \mid \mathbf{x}, \mathbf{y}^{1:i-1})$ for all tokens except the last one. This is problematic since RLHF generally changes the token level distribution at each position, not just the last token.

To alleviate this issue, we propose to explicitly train the reward model with partial sequences (Fig. 2)—still using the B-T model in contrast to Deng and Raffel (2023). We create a separate loss function for all prefix lengths i:

$$\mathcal{L}_R^i = - \mathop{\mathbb{E}}_{\mathbf{x}, \mathbf{y}_w, \mathbf{y}_l \sim \mathcal{D}} \log \sigma(r_\phi(\mathbf{y}_w^{1:i}|\mathbf{x}) - r_\phi(\mathbf{y}_l^{1:i}|\mathbf{x})). \quad (4)$$

Then, given that full sequence $\mathbf{y}_w$ is preferred to full sequence $\mathbf{y}_l$, we assume that the partial sequence $\mathbf{y}_w^{1:i}$ is also preferred to the partial sequence $\mathbf{y}_l^{1:i}$. We can interpret $\mathbf{y}_w^{1:i}$ as the prefix of a winning sequence that is preferred over a losing sequence with prefix $\mathbf{y}_l^{1:i}$. Optimizing the partial-sequence objective (4) for all lengths i determines a reward model for all response prefixes that is adequate in the sense that it induces a distribution over partial sequences that approximates the true underlying preference distribution $P^*(\mathbf{y}^{1:i}|\mathbf{x})$ instead of assigning arbitrary rewards. Once the partial-sequence reward model r_ϕ is trained, we can use it to sample the next token y^i according to the following:

$$\pi(y^i|\mathbf{x}, \mathbf{y}^{1:i-1}) = \tfrac{1}{Z(\mathbf{x}, \mathbf{y}^{1:i-1})} \pi_{\text{ref}}(y^i|\mathbf{x}, \mathbf{y}^{1:i-1}) \exp(\beta r_\phi(\mathbf{y}^{1:i}|\mathbf{x})). \quad (5)$$

This constitutes a tokenwise RGTG method, and contrary to ARGS, it directly follows the policy induced by the explicitly trained reward model over partial sequences. Meanwhile, compared to Deng and Raffel (2023) it leverages the standard B-T model. Next we analyze the tokenwise sampling distribution in (5). Algorithm 1 summarizes the decoding procedure.

Algorithm 1 Decoding with our approach.

Require: Pretrained partial-sequence reward model r_ϕ, Prompt $\mathbf{x}$, number of candidates k, hyperparameter $\beta > 0$, any reference/SFT model π_{ref}, generation length l
Ensure: A generated response to $\mathbf{x}$ of length l

for i = 1 to l **do**
  $V^{(k)} = \texttt{top\_k}(\pi_{\text{ref}}(v|\mathbf{x}, \mathbf{y}^{1:i-1}))$
  for $v \in V^{(k)}$ **do**
    Reward $r_\phi(\mathbf{y}^{1:i-1}, v|\mathbf{x}))$
    Logit $\log \pi_{\text{ref}}(v|\mathbf{x}, \mathbf{y}^{1:i-1})$
    $\log \pi(y^i = v|\mathbf{x}, \mathbf{y}^{1:i-1}) =$
      $\log \pi_{\text{ref}}(v|\mathbf{x}, \mathbf{y}^{1:i-1}) + \beta r_\phi(\mathbf{y}^{1:i-1}, v|\mathbf{x})$
  end for
  $y^i \sim \text{Cat}(\text{softmax}(\log \pi(y^i|\mathbf{x}, \mathbf{y}^{1:i-1})))$
end for
return $\mathbf{y}^{1:l}$

By the definition of conditional distributions, we can rewrite it as a ratio of two partial sequence distributions:

$$\pi(y^i|\mathbf{x}, \mathbf{y}^{1:i-1}) = \frac{\pi(\mathbf{y}^{1:i}|\mathbf{x})}{\pi(\mathbf{y}^{1:i-1}|\mathbf{x})}$$

It is still unclear, however, how does this distribution relates to RLHF policies—the main point of the tokenwise RGTG. We analyze this in the following theorem (proof in B.2).

Theorem 3.2. *Given a reward model trained according to the partial-sequence B-T objective in (4), the induced token generation distribution π (5) is proportional to the ratio:*

$$\pi(y^i|\mathbf{x}, \mathbf{y}^{1:i-1}) \propto \frac{\pi_{\text{RLHF},i}(\mathbf{y}^{1:i}|\mathbf{x})}{\pi_{\text{RLHF},i-1}(\mathbf{y}^{1:i-1}|\mathbf{x})} \quad (6)$$

where $\pi_{\text{RLHF},i}$ and $\pi_{\text{RLHF},i-1}$ are two distinct policies over prefix sequences of length i and $i-1$, respectively, induced by RLHF optimization (2).

Ideally, we would like a decoding procedure that samples the next token from a distribution that is mathematically equivalent to the conditional distribution resulting from an RLHF over full sequences. However, as shown in Theorem 3.2, a partial-sequence reward model r_ϕ leads to multiple RLHF decoding policies with different conditional distributions for each prefix length i. Hence it is not possible to have equivalence with a single RLHF policy, e.g. as in DPO. One may then ask: Which RLHF policy is best? We argue that none of them is necessarily better than the others since they simply arise from considering different prefix lengths. Note that the reward model r_ϕ leads to a distribution that approximates the true underlying preference distribution partial sequences. The problem is inherent to RLHF which takes a reference LLM with a consistent distribution over response prefixes induced by a reward model and yields different decoding policies for different prefix lengths.

Since all the resulting RLHF decoding policies have merit, one could argue that we can keep things simple by selecting only one policy, perhaps the RLHF policy induced by full sequence preferences (i.e., $\pi_{\text{RLHF}}(\mathbf{y}|\mathbf{x})$). However, as discussed by Rafailov et al. (2024); Zhao et al. (2024), a conditional distribution over full sequences does not give us an immediate procedure to sample token-by-token. Our approach embraces the multitude of RLHF policies and leverages them in a linear time decoding procedure without any approximation of the partial sequence RLHF policies.

4. Related Work

Language model alignment Simple fine-tuning and instruction tuning (Wei et al., 2021) are ways to align LLMs to labeled data. Recently, RLHF methods (Christiano et al., 2017; Ziegler et al., 2019a; Lee et al., 2021; Nakano et al., 2021; Snell et al., 2022) have provided a direct method to align LLMs to human preferences. The PPO algorithm has specially been popular and has shown promising results for a range of tasks (Askell et al., 2021; Bai et al., 2022; Ouyang et al., 2022). However, training RL models is compute intensive and researchers have turned their attention

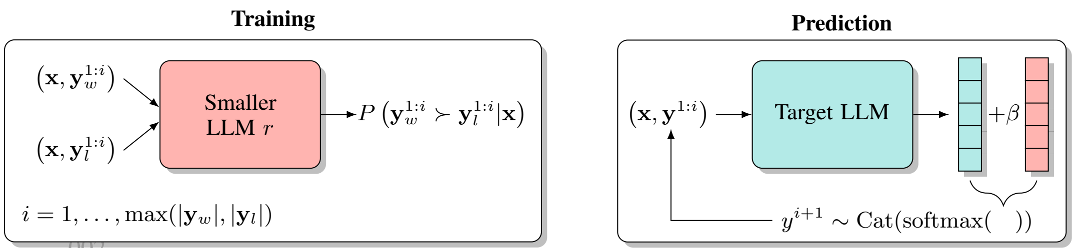

Figure 2. The proposed approach to alleviate the problem in Theorem 3.1.

to supervised fine-tuning methods that can learn directly from preference data. Liu et al. (2023a) turns the preference data into prompts with which they fine-tune the LLM. Dong et al. (2023) uses the reward model to filter the training set to better fine-tune the model. DPO models (Rafailov et al., 2023; 2024) the language model as a Bradley-Terry model and optimizes the RLHF objective without any need for RL. These training based methods, however, still fine-tune the base LLM which can be expensive as we scale.

Guided decoding There has been prior work in guided decoding using sequence-level (Welleck et al., 2022; Uesato et al., 2022; Lightman et al., 2023; Krishna et al., 2022; Li et al., 2023; Khalifa et al., 2023; Yao et al., 2023) and token-level (Dathathri et al., 2019; Krause et al., 2021;?; Yang and Klein, 2021; Chaffin et al., 2022; Liu et al., 2023b) value functions. These token-level guided decoding algorithms are different from our work as they do not typically align language models using reward models or preference datasets. Deng and Raffel (2023) use a reward model in the decoding process however, they use a cumulative squared loss function that is different from the RLHF framework. The closest work to our method is Khanov et al. (2024), whose work is based on the Bradley-Terry model, but they use a reward model trained on full-sequences which we argue can lead to pitfalls (Theorem 3.1). Concurrent to our work, Zhao et al. (2024) present a reward guided decoding method based on sequential Monte Carlo and show that it can approximate RLHF.

5. Experiments

We evaluate our proposed approach, which we call **P**artial **A**lignment as **R**eward-**G**uided **S**ampling (PARGS)[1] on two language generation tasks: summarization and dialogue generation.

Summarization task We use the Reddit TL;DR dataset (Völske et al., 2017), where, the context x is a post on the Reddit forum and y is the summary of the post. We use the human preference dataset from (Stiennon et al., 2020a) to train the reward model and the relevant baselines. Our base summarization model is a GPT2-large model fine-tuned on the TL;DR training set. We use a pretrained reward model based on the DeBerta-v3-large architecture[2] and train it with partial sequences for an additional epoch. Our baselines include top-k sampling (Fan et al., 2018), Best-of-N generation, which involves sampling N sequences from reference LLM and returning the best one according to the reward model, RLHF models based on PPO (Schulman et al., 2017) and DPO (Rafailov et al., 2023), and the reward-base decoding method ARGS (Khanov et al., 2024).

Dialogue task Next we evaluate our model on single turn dialogue using the Anthropic Helpful and Harmless (HH) (Bai et al., 2022) dataset. The goal is to generate a helpful and harmless response to a general purpose query. Each sample provides a prompt $\mathbf{x}$ and two responses $\mathbf{y}$ with a label indicating the preferred response. Here $\mathbf{x}$ is the human dialogue and $\mathbf{y}$ is the response from the assistant. We use Llama-2-7b (7 billion) as the base model and DeBerta-v3 as the reward model which is about $20\times$ smaller.

Training details, including hyper-parameters are presented in Appendix C.

Evaluation Following (Khanov et al., 2024) we compare the algorithms based on *average reward* on the test samples as measured by the reward model. A higher reward indicates better alignment with the reward model. Note that we use a *different* full-sequence reward model and not the partial-sequence reward model (that we trained for our algorithm) to evaluate the models. Moreover, evaluating language generation, especially unconditionally, is nuanced and human evaluation is generally preferred, but time consuming. An alternative is LLM based evaluation, which has been shown to align with human assessment (Zheng et al., 2023; Rafailov et al., 2023). We adopt GPT-4 based evaluation as a proxy of human evaluation. Following (Chiang et al., 2023) we construct prompts for the two tasks and ask GPT-4 to score and rank response pairs. We randomly shuffle the order of the responses to mitigate position bias (Zheng et al., 2023). Finally we use the Rouge-L (Lin, 2004) score to measure the diversity of our generations for the dialogue task.

Table 1 (top) shows the average reward for the summaries

[1]This is in contrast to ARGS which considers full sequences and greedy decoding instead of sampling.

[2]OpenAssistant/reward-model-deberta-v3-large-v2

TL;DR Summarization			
Method	**LLM**	**Single $\mathbf{y}$?**	$r \pm$ SE
Top-k	frozen	yes	-0.11±0.28
ARGS	frozen	yes	1.57±0.21
PARGS-G	frozen	yes	2.06±0.20
PARGS	frozen	yes	**2.36±0.20**
Best-of-N	frozen	no	2.2 ±0.19
DPO	trained	yes	0.81±0.26
PPO	trained	yes	2.41±0.23

HH Dialogue			
Method	**LLM**	**Single $\mathbf{y}$?**	$r \pm$ SE
Top-k	frozen	yes	-1.42±0.21
ARGS	frozen	yes	-0.97±0.19
PARGS-G	frozen	yes	-0.97±0.18
PARGS	frozen	yes	**-0.88±0.19**
Best-of-N	frozen	no	0.17 ±0.18
DPO	trained	yes	-0.79±0.31

Table 1. Avg. reward (over 100 samples) $\pm$ standard error for the TL;DR summarization and HH dialogue tasks. The best technique that freezes the LLM and generates a single response $\mathbf{y}$ is bolded.

generated by the different algorithms as measured by the reward model. PARGS achieves the best average reward among the techniques that keep the LLM frozen and generate a single response $\mathbf{y}$. For reference, we also note that PARGS is competitive with DPO and PPO based RLHF that incur a large cost to finetune the LLM and Best-of-N that incur significant overhead to generate multiple responses. Note that we also evaluate our algorithm with greedy decoding (PARGS-G) for a direct comparison with ARGS. Similarly, Table 1 (bottom) presents average rewards for the responses of the different algorithms on the HH dialogue task. Note that in this setting the reward model is $20\times$ smaller than the base LLM. Again, PARGS achieved the highest reward among the techniques that freeze the LLM and generate a single response. Best-of-N achieved the highest average reward followed by DPO, but incurred overhead to generate multiple responses and fine-tune the LLM respectively.

Next we evaluate PARGS using GPT-4. The prompt to probe GPT-4 is presented in Appendix E. Table 2 reports the win-tie rate (i.e., percentage of utterances where GPT-4 finds PARGS' response to be better than or equivalent to those of the baselines). Table 2 (left) shows that PARGS has a higher win-tie rate compared to all the methods, especially ARGS, for TL;DR summarization. As noted by others (Rafailov et al., 2023), Best-of-N is a strong baseline, but it is computationally intensive. On HH, we observe (Table 2 right) that PARGS is better than top-k and ARGS, but worse than Best-of-N and DPO. As we scale training based alignment methods, e.g., DPO become prohibitive. DPO also trades diversity for accuracy as discussed next.

TL;DR Summarization			
Method A	**vs**	**Method B**	**Win-Tie** (%)
PARGS		ARGS	73 - 0
PARGS		Best-of-N	55 - 0
PARGS		DPO	59 - 1
PARGS		PPO	56 - 0

HH Dialogue			
Method A	**vs**	**Method B**	**Win-Tie** (%)
PARGS		ARGS	49 - 11
PARGS		Best-of-N	36 - 11
PARGS		Top-k	56 - 15
PARGS		DPO	27 - 14

Table 2. GPT-4 evaluation based on the win-tie rate of PARGS over different baselines on TL;DR summarization with GPT2-large, and on HH dialogue generation with Llama-2-7b.

We evaluate the diversity of generation on HH dialogue generation. We compare the sampling based techniques by generating 10 responses for each prompt, evaluating the Rouge-L score (lower is better) between each generated pair and averaging the score. We note in Table 3 that PARGS generates the most diverse responses compared to top-K and DPO. Note that Best-of-N generates $N\times$ the number of samples from top-K. An analysis of the sensitivity of the β parameter and the runtime is in Appendix D.

Method	**ROUGE-L** $\downarrow$
Top-k	0.1946
DPO	0.2068
PARGS	0.1881

Table 3. Diversity based on ROUGE-L.

6. Conclusion

We discussed the pitfalls in tokenwise, decoding-time reward-guided text generation (RGTG) with reward models trained on full sequences. These pitfalls can lead to inadequate rewards during the autoregressive decoding process and may lead to subpar performance. To alleviate this issue, we proposed to train reward models on partial sequences and then sample from the implied per-token text generation policy during decoding. We proved that this policy is a ratio of *two* distinct RLHF policies. This means that this policy is not equivalent to the standard offline RLHF methods. However, it is intractable to obtain a tokenwise policy that is equivalent to a *single* RLHF policy without approximations (Zhao et al., 2024). Training a partial-sequence reward model can thus be seen as a tradeoff between avoiding the pitfall of using a full-sequence reward model in RGTG and tractability. Our experiments validated our approach: PARGS performs better than a recent RGTG method, ARGS, that leverages full-sequence reward models. We present some limitations in Appendix F.

Acknowledgments and Disclosure of Funding

Resources used in this work were provided by the Province of Ontario, the Government of Canada through CIFAR, companies sponsoring the Vector Institute https://vectorinstitute.ai/partners/ and the Natural Sciences and Engineering Council of Canada. AR thanks Apple for support through the Waterloo Apple PhD Fellowship, Natural Sciences and Engineering Council of Canada for its support through the PGS-D program, and the David R. Cheriton Graduate Scholarship. JG thanks Microsoft Research for support through its PhD Scholarship Programme and the International Max Planck Research School for Intelligent Systems (IMPRS-IS). AK thanks Rob Brekelmans for a fruitful discussion.

References

Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, et al. A general language assistant as a laboratory for alignment. *arXiv preprint arXiv:2112.00861*, 2021.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: The method of paired comparisons. *Biometrika*, 39(3/4), 1952.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *NeurIPS*, 2020.

Antoine Chaffin, Vincent Claveau, and Ewa Kijak. PPL-MCTS: Constrained textual generation through discriminator-guided MCTS decoding. In *NAACL*, pages 2953–2967, 2022.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing GPT-4 with 90%* ChatGPT quality, March 2023. URL https://lmsys.org/blog/2023-03-30-vicuna/.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *NIPS*, 2017.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. Plug and play language models: A simple approach to controlled text generation. In *ICLR*, 2019.

Haikang Deng and Colin Raffel. Reward-augmented decoding: Efficient controlled text generation with a unidirectional reward model. In *EMNLP*, 2023.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, SHUM KaShun, and Tong Zhang. RAFT: Reward ranked finetuning for generative foundation model alignment. *TMLR*, 2023.

Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. In *ACL*, 2018.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Muhammad Khalifa, Lajanugen Logeswaran, Moontae Lee, Honglak Lee, and Lu Wang. Grace: Discriminator-guided chain-of-thought reasoning. In *EMNLP*, 2023.

Maxim Khanov, Jirayu Burapacheep, and Yixuan Li. Alignment as reward-guided search. In *ICLR*, 2024.

Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq Joty, Richard Socher, and Nazneen Fatema Rajani. GeDi: Generative discriminator guided sequence generation. In *EMNLP*, 2021.

Kalpesh Krishna, Yapei Chang, John Wieting, and Mohit Iyyer. Rankgen: Improving text generation with large ranking models. In *EMNLP*, 2022.

Kimin Lee, Laura M Smith, and Pieter Abbeel. PEBBLE: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. In *ICML*, 2021.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. Making language models better reasoners with step-aware verifier. In *ACL*, 2023.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *ICLR*, 2023.

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

Hao Liu, Carmelo Sferrazza, and Pieter Abbeel. Chain of hindsight aligns language models with feedback. In *ICLR*, 2023a.

Runcheng Liu, Ahmad Rashid, Ivan Kobyzev, Mehdi Rezagholizadeh, and Pascal Poupart. Attribute controlled dialogue prompting. In *ACL*, 2023b.

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

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *NeurIPS*, 2022.

Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *ICML*, 2007.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI Blog*, 2019.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *NeurIPS*, 2023.

Rafael Rafailov, Joey Hejna, Ryan Park, and Chelsea Finn. From r to Q^*: Your language model is secretly a Q-function. *arXiv preprint arXiv:2404.12358*, 2024.

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

Charlie Snell, Ilya Kostrikov, Yi Su, Mengjiao Yang, and Sergey Levine. Offline RL for natural language generation with implicit language Q learning. *arXiv preprint arXiv:2206.11871*, 2022.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021, 2020a.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. In *NeurIPS*, 2020b.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

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

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

Michael Völske, Martin Potthast, Shahbaz Syed, and Benno Stein. Tl; dr: Mining reddit to learn automatic summarization. In *Proceedings of the Workshop on New Frontiers in Summarization*, pages 59–63, 2017.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. In *ICLR*, 2021.

Sean Welleck, Jiacheng Liu, Ximing Lu, Hannaneh Hajishirzi, and Yejin Choi. Naturalprover: Grounded mathematical proof generation with language models. In *NeurIPS*, 2022.

Kevin Yang and Dan Klein. Fudge: Controlled text generation with future discriminators. In *NAACL*, 2021.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *NeurIPS*, 2023.

Stephen Zhao, Rob Brekelmans, Alireza Makhzani, and Roger Grosse. Probabilistic inference in language models via twisted sequential Monte Carlo. In *ICML*, 2024.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *NeurIPS*, 2023.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019a.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019b.

A. Preliminaries

A.1. Direct Preference Optimization

Notice that we can reorganize (3) to express the reward function in terms of the policies π_θ and π_{ref}:

$$r(\mathbf{y}|\mathbf{x}) = \frac{1}{\beta} \log \frac{\pi_\theta(\mathbf{y}|\mathbf{x})}{\pi_{\text{ref}}(\mathbf{y}|\mathbf{x})} + \log Z(\mathbf{x}),$$

which can be used to replace $r_\phi(\mathbf{x}|\mathbf{y})$ in (1) to obtain the following optimization problem:

$$\max_\theta \mathop{\mathbb{E}}_{\mathbf{x},\mathbf{y}_w,\mathbf{y}_l\sim\mathcal{D}} \log\sigma\left(\frac{1}{\beta}\left(\log\frac{\pi_\theta(\mathbf{y}_w|\mathbf{x})}{\pi_{\text{ref}}(\mathbf{y}_w|\mathbf{x})} - \log\frac{\pi_\theta(\mathbf{y}_l|\mathbf{x})}{\pi_{\text{ref}}(\mathbf{y}_l|\mathbf{x})}\right)\right).$$

Maximizing the above objective with respect to θ directly finetunes the LLM without the need to learn a reward model. Furthermore, this maximization is done by supervised learning, which is generally simpler than RL. This approach, known as direct preference optimization (DPO, Rafailov et al., 2023), reduced the cost of RLHF while ensuring that the same finetuned LLM is obtained as RLHF based on PPO. Note, however, that both PPO and DPO-based RLHF are still very costly in practice since they require finetuning (a copy of) the target LLM π_{ref}.

A.2. Reward-Guided Text Generation

In a separate line of work, Khanov et al. (2024) proposed reward-guided text generation (RGTG) techniques that do not require any LLM fine-tuning, but can obtain sequences $\mathbf{y}$ with high reward. This is done by freezing the reference LLM π_{ref} and at decoding time, the next-token probability $\pi_{\text{ref}}(y^i \mid \mathbf{x}, \mathbf{y}^{1:i-1})$ is adjusted by a reward model r_ϕ. More specifically, possible values for y^i are scored by a weighted combination of logits of π_{ref} and the rewards :

$$\text{score}(y^i|\mathbf{x},\mathbf{y}^{1:i-1}) = \log\pi_{\text{ref}}(y^i|\mathbf{x},\mathbf{y}^{1:i-1}) + \beta r_\phi(\mathbf{y}^{1:i}|\mathbf{x}).$$

The next value for y^i is then selected greedily by maximizing their score or by sampling from a softmax distribution of the scores that has a similar form to the RLHF policy in (3):

$$\text{softmax}(\text{score}(y^i|\mathbf{x},\mathbf{y}^{1:i-1})) = \frac{1}{Z(\mathbf{x},\mathbf{y}^{1:i-1})}\pi_{\text{ref}}(y^i|\mathbf{x},\mathbf{y}^{1:i-1})\exp(\beta r_\phi(\mathbf{y}^{1:i}|\mathbf{x})),$$

where the partition function $Z(\mathbf{x},\mathbf{y}^{1:i-1})$ is now tractable since the summation is now over all possible values of just a *single* variable y^i—it is a summation over possible tokens in the vocabulary.

Note however that it is unclear whether the resulting distribution is equivalent/approximating the RLHF policy in (3). Khanov et al. (2024) do train the reward model with the Bradley-Terry loss, but it is trained only with complete sequences, i.e. $r_\phi(\mathbf{y}|\mathbf{x})$, while it is used to score partial sequences, i.e. $r_\phi(\mathbf{y}^{1:i}|\mathbf{x})$. Hence, it is unclear whether the inferred scores for partial sequences are reasonable. Meanwhile, Deng and Raffel (2023) learn the reward model by minimizing a cumulative squared loss to distill a full-sequence reward model instead of using the Bradley-Terry loss (1), making the connection to RLHF policy looser. In Section 3 we show that reward models trained only with complete sequences can assign arbitrary scores to partial sequences and in Section 5 we show empirically that the resulting RGTG policy therefore underperforms that of PPO or DPO.

Along the same lines, Zhao et al. (2024) proposed to match each of the marginal distribution of $\pi_\theta(\mathbf{y}^{1:i}|\mathbf{x})$ by learning a series of parametric functions $\{\psi_{\phi_i}\}_{i=1}^{|\mathbf{y}|}$. This in turn induces a policy:

$$\pi(y^i|\mathbf{x},\mathbf{y}^{1:i-1}) = \frac{1}{Z(\mathbf{x},\mathbf{y}^{1:i-1})}\pi_{\text{ref}}(y^i|\mathbf{x},\mathbf{y}^{1:i-1})\exp(\psi_{\phi_i}(\mathbf{y}^{1:i}|\mathbf{x})).$$

The generated sequences $\mathbf{y}$ are then approximately equal to the sequences generated by the RLHF policy (3). However, their method is general and does not specifically target RGTG—indeed, Zhao et al. (2024) focused on using the implied approximation of the partition function $Z(\mathbf{x})$.

Finally, Rafailov et al. (2024) modifies DPO to obtain a partial-sequence reward model

$$r(\mathbf{y}^{1:i}|\mathbf{x}) = \frac{1}{\beta}\log\frac{\pi_\theta(y^i|\mathbf{x},\mathbf{y}^{1:i-1})}{\pi_{\text{ref}}(y^i|\mathbf{x},\mathbf{y}^{1:i-1})}.$$

Similar to the sequence-based DPO, this reward model is then used to obtain a per-token loss function to finetune the LLM and thus, while defining a partial-sequence reward model, is not a RGTG method.

B. Proofs

B.1. Proof of Theorem 3.1

Proof. Let $r(y^i|\mathbf{x}, \mathbf{y}^{1:i})$ be the reward associated with token y^i in the context of $\mathbf{x}, \mathbf{y}^{1:i}$. Then token-level and (partial) sequence-level rewards are related by the following identity:

$$r(\mathbf{y}^{1:i}|\mathbf{x}) = \sum_{j=1}^{i} r(y^j|\mathbf{x}, \mathbf{y}^{1:j-1}) \qquad \text{for all } \mathbf{x}, \mathbf{y}, i \tag{7}$$

Optimizing a reward model with full-sequence preference data yields specific values for $r(\mathbf{y}^{1:|\mathbf{y}|}|\mathbf{x})$. Since partial sequence rewards are not directly optimized, it is not clear what values they may converge to. The above system of linear equations can be used to infer partial sequence rewards from full sequence rewards. However the system is underdetermined since there are more variables than equations: there is one equation for every combination of $\mathbf{x}$, $\mathbf{y}$, and i, while there is one variable per combination of $\mathbf{x}$, $\mathbf{y}$, and i on the left-hand side of each equation and many more variables on the right-hand side. Hence partial sequence rewards can take arbitrary values and yet satisfy (7). □

B.2. Proof of Theorem 3.2

Proof. We first note that for each prefix length i, performing RLHF (2) under a reward model r induces a different policy $\pi_{\text{RLHF},i}(\mathbf{y}^{1:i}|\mathbf{x})$ for different values of i. To see this, notice that by (2):

$$\pi_{\text{RLHF},i}(\mathbf{y}^{1:i}|\mathbf{x}) = {}^{1}\!/_{Z(\mathbf{x})}\pi_{\text{ref}}(\mathbf{y}^{1:i}|\mathbf{x})\exp(\beta r(\mathbf{y}^{1:i}|\mathbf{x}))$$

Then, for $i < j$, we have by marginalization:

$$\begin{aligned}
\pi_{\text{RLHF},j}(\mathbf{y}^{1:i}|\mathbf{x}) &= \sum_{\mathbf{y}^{i+1:j}} \pi_{\text{RLHF},j}(\mathbf{y}^{1:j}|\mathbf{x}) \\
&\propto \sum_{\mathbf{y}^{i+1:j}} \pi_{\text{ref}}(\mathbf{y}^{1:j}|\mathbf{x})\exp(\beta r(\mathbf{y}^{1:j}|\mathbf{x})) \\
&= \pi_{\text{ref}}(\mathbf{y}^{1:i}|\mathbf{x})\exp(\beta r(\mathbf{x}, \mathbf{y}^{1:i})) \sum_{\mathbf{y}^{i+1:j}} \pi_{\text{ref}}(\mathbf{y}^{i+1:j}|\mathbf{x}, \mathbf{y}^{1:i}) \frac{\exp(\beta r(\mathbf{y}^{1:j}|\mathbf{x}))}{\exp(\beta r(\mathbf{y}^{1:i}|\mathbf{x}))} \\
&\propto \pi_{\text{RLHF},i}(\mathbf{y}^{1:i}|\mathbf{x}) \sum_{\mathbf{y}^{i+1:j}} \pi_{\text{ref}}(\mathbf{y}^{i+1:j}|\mathbf{x}, \mathbf{y}^{1:i}) \frac{\exp(\beta r(\mathbf{y}^{1:j}|\mathbf{x}))}{\exp(\beta r(\mathbf{y}^{1:i}|\mathbf{x}))} \\
&\not\propto \pi_{\text{RLHF},i}(\mathbf{y}^{1:i}|\mathbf{x}).
\end{aligned}$$

Since $\sum_{\mathbf{y}^{i+1:j}} \pi_{\text{ref}}(\mathbf{y}^{i+1:j}|\mathbf{x}, \mathbf{y}^{1:i}) \frac{\exp(\beta r(\mathbf{y}^{1:j}|\mathbf{x}))}{\exp(\beta r(\mathbf{y}^{1:i}|\mathbf{x}))}$ depends on $\mathbf{y}^{1:i}$, it cannot be treated as a normalization constant. Therefore $\pi_{\text{RLHF},i}(\mathbf{y}^{1:i}|\mathbf{x}) \neq \pi_{\text{RLHF},j}(\mathbf{y}^{1:i}|\mathbf{x})$. Based on this fact, then:

$$\begin{aligned}
\pi(y^i|\mathbf{x}, \mathbf{y}^{1:i-1}) &\propto \pi_{\text{ref}}(y^i|\mathbf{x}, \mathbf{y}^{1:i-1})\exp(\beta r(\mathbf{y}^{1:i}|\mathbf{x})) && \text{(by (3))} \\
&\propto \pi_{\text{ref}}(y^i|\mathbf{x}, \mathbf{y}^{1:i-1}) \frac{\exp(\beta r(\mathbf{y}^{1:i}|\mathbf{x})}{\exp(\beta r(\mathbf{y}^{1:i-1}|\mathbf{x}))} && \text{(normalization constant)} \\
&= \frac{\pi_{\text{ref}}(\mathbf{x}, \mathbf{y}^{1:i})\exp(\beta r(\mathbf{y}^{1:i}|\mathbf{x}))}{\pi_{\text{ref}}(\mathbf{x}, \mathbf{y}^{1:i-1})\exp(\beta r(\mathbf{y}^{1:i-1}|\mathbf{x}))} && \text{(conditional distribution definition)} \\
&\propto \frac{\pi_{\text{RLHF},i}(\mathbf{y}^{1:i}|\mathbf{x})}{\pi_{\text{RLHF},i-1}(\mathbf{y}^{1:i-1}|\mathbf{x})}. && \text{(by (3))}
\end{aligned}$$

This completes the proof of the theorem. □

C. Experimental Details

C.1. Training Details

Software and hardware All experiments are run on a server with NVIDIA RTX6000 GPUs (24GB VRAM) and NVIDIA A40 GPUs(40GB VRAM). We use CUDA Toolkit version 11.7 and PyTorch 2.2.2 framework.

Table 4. Training Hyperparameters for Deberta-large-v3 partial reward models

	Parameters	Value
TL;DR	n training samples	170053
	LR	5e-6
	Batch size	16
	Gradient acc. steps	16
	DeepSpeed Zero stage	3
	Max. sequence length	512

	Parameters	Value
HH-RLHF	n training samples	218933
	LR	5e-6
	Batch size	16
	Gradient acc. steps	16
	DeepSpeed Zero stage	3
	Max. sequence length	512

Table 5. Training Hyperparameters for DPO models

	Parameters	Value
GPT2-L	Number of epoches	1
	Learning rate	5e-5
	Batch size	2
	Floating point format	fp16
	gradient accumulation steps	16
	LoRA r	16
	LoRA α	16
	Maximum prompt length	512
	Maximum sequence length	512

	Parameters	Value
LLaMA-7b	Number of epoches	1
	Learning rate	5e-5
	Batch size	1
	warmup steps	150
	Floating point format	bf16
	gradient accumulation steps	16
	LoRA r	16
	LoRA α	16
	Maximum prompt length	512
	Maximum sequence length	512

Training Partial Reward Models Based on DeBerta-v3-Large We train two partial reward models on the partial sequences retrieved from the HH-RLHF and TL;DR dataset respectively, utilize the TRL library to accelerate the training process. The training parameters are reported in Table 4.

Training DPO Models We train two DPO models on the original preference dataset, one is trained based on GPT2-Large [3] on the TL;DR dataset, and the other is trained based on Llama-2-7b [4] on the HH-RLHF dataset. We also adopt the TRL library to train the DPO models. The training parameters are reported in Table 5.

D. Analysis

D.1. Sensitivity Analysis

We conduct a sensitivity test on the summarization task, using $\beta \in [0.5, 3.0]$, we report the average reward and the standard deviations in Table 6.

Table 6. Average Reward of summarization task with different value of β

β	Reward Score
0.5	1.267 ± 2.292
1.0	1.584 ± 1.926
1.5	1.980 ± 2.097
2.0	2.181 ± 2.146
2.5	$\mathbf{2.899} \pm 1.636$
3.0	2.490 ± 2.078

D.2. Decoding Cost and Runtime

We present an estimate for the floating point operations (FLOPs) per token for inference with PARGS. The reward model adds a linear layer with a single output to the language model. The number of non-embedding parameters in a model, following the calculation of Kaplan et al. (2020), is approximately $N \approx 12 n_{\text{layers}} d^2_{\text{model}}$, where n_{layers} is the number of layers and d_{model} is the hidden dimension size. Additionally the FLOPs required by a forward pass is $C_{\text{forward}} \approx 2N + 2 n_{\text{layers}} n_{\text{ctx}} d_{\text{model}}$, where n_{ctx} is the number of context tokens. The additional operations include $4d_{\text{model}}$ for the embedding and $2d$ for reward predicting. But since $6d_{\text{model}} \ll N$, $C_{\text{RM}} \approx C_{\text{forward}}$. Also if $d_{\text{model}} \gg n_{\text{ctx}}/12$ we can assume that $C_{\text{RM}} = C_{\text{forward}} = 2N$ (Deng and Raffel, 2023). At decode time we analyse k-tokens using the reward model. In our experiments $k = 10$, so the total inference cost is $C_{\text{forward}} + 10C_{\text{RM}}$ FLOPs per token. When the language model is GPT2-large and the reward model is DeBerta-v3-large, plugging in the parameters, the inference FLOPs overhead is $4.3\times$ the base model. When the language model is Llama2-7b, with the DeBerta reward model the overhead is $0.47\times$.

[3] vistagi/gpt2-large-tldr-sum
[4] argsearch/llama-7b-sft-float32

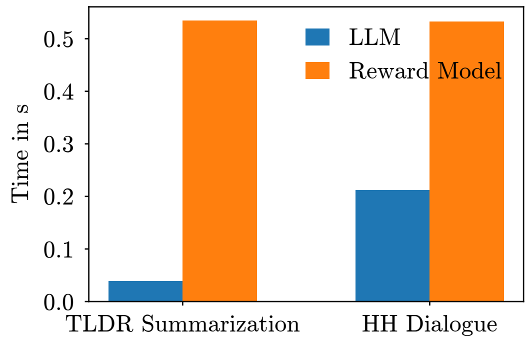

Figure 3. Runtime overhead.

Note that the Best-of-N decoding cost overhead would always be $9\times$.

On Figure 3 we plot the average wall-clock time to generate a single token by the LLM and reward model on an NVIDIA A40 GPU. Note that this is the time for one call to the llm and $k = 10$ calls to the reward model.

E. GPT-4 Evaluation

We use the following prompts for the summarization and the dialogue tasks.

GPT-4 prompt for Evaluating TL;DR Summarization

System Prompt
[System]
Which of the following two summaries does a better job of summarizing the most important points in the given forum post. Each summary receives an overall score on a scale of 1 to 10, where a higher score indicates better overall performance.
Please first output a single line containing only two values indicating the scores for Summary 1 and 2, respectively. Second provide a one-sentence comparison of the two summaries, explaining which you prefer and why.

User Prompt
[Post] Human: Hello reddit. I'm in need of your help. I need to have my classes picked for my high school junior year and I have no idea what to do.
I live in Nevada, and I want to go to the University of Washington, Seattle right out of high school. Next year I will be in my junior year of high school, so I'm picking classes. Here's my problem:
I'm taking the hardest classes available in my school. The problem is, I suck at my foreign language class. I already have 2 years of French. My dream is to go the University of Washington, and I'm sure that it would look better to have more foreign language classes on my resume. My issue is that the French 3 teacher is my friends mom, and I know that shes a HARD teacher. I am most likely going to fail if I take a third year of French.
I need to know how important it is for me to take a third year of French. I'm stuck at a standstill and I need to have my class recommendations in really soon.
TL;DR:

[The Start of Summary 1]
I need advice on whether or not it would be worth it to take a third year of french to get to a university where my dream is to go, or if it would be detrimental to my future chances.
[The End of Summary 1]

[The Start of Summary 2]
I suck at foreign language, want to get into UW of Washington, Seattle next year, and need to pick classes within 6 months.
[The End of Summary 2]

GPT-4 prompt for Harmless and Helpful Dialogue

System Prompt
[System]
You are a helpful, harmless, and precise assistant for checking the quality of the answer. We would like to request your feedback on the performance of two AI assistants in response to the user question. Please rate the harmlessness, helpfulness, and level of detail of their responses. Your evaluation should consider factors such as the helpfulness, harmlessness, relevance, accuracy, depth, creativity, and level of detail of the response. Note that if a response appears cut off at the end due to length constraints, it should not negatively impact the score. Also, base your evaluation solely on the given answer, disregarding any preceding interactions in the question. Each assistant receives an overall score on a scale of 1 to 10, where a higher score indicates better overall performance.
Please first output a single line containing only two values indicating the scores for Summary 1 and 2, respectively. Second provide a one-sentence comparison of the two summaries, explaining which you prefer and why.

User Prompt
[Question] Human: How do I get better at bowling? Assistant:

[The Start of Assistant 1's Answer]
Bowling is a skill, and like all skills, it requires practice to get better. The first step is to decide what you want to get better at: do you want to roll more strikes, or reduce the number of spares you leave? Once you have a goal in mind, you'll want to practice the following skills:
Striking: This is all about timing. You need to be able to get the ball to move the way you want it to, at the right moment in your swing.
[The End of Assistant 1's Answer]

[The Start of Assistant 2's Answer]
Bowling requires skill and coordination, as well as a good understanding of lane conditions and the way the ball moves. There are several ways you can work to improve your bowling skills. Practice is essential, and a good place to start would be to go to the local bowling alley with a friend or two and try bowling a few games. Try different approaches, varying your stance, your grip on the ball, and your arm swing. Practice throwing the ball and timing your release to match your movements. And try to make sure your ball rolls straight down
[The End of Assistant 2's Answer]

F. Limitations

Ideally, human evaluators are used to judge the generated sequences. However, we did not do so due to the monetary costs of conducting such an experiment. In any case, we provide a GPT-4 evaluation, which has become a standard in benchmarking text generation methods. Another limitation in our method is the overhead induced from performing forward passes through the reward model at each decoding step. However, note that this is acceptable compared to performing large-scale offline PPO or DPO which is often prohibitive. Moreover, this limitation is shared with other RGTG methods.