A Critical Look At Tokenwise Reward-Guided Text Generation

Ahmad Rashid
University of Waterloo
Vector Institute

Ruotian Wu
University of Waterloo
Vector Institute

Julia Grosse
University of Tübingen
Tübingen AI Center

Agustinus Kristiadi*
Western University
Vector Institute

Pascal Poupart*
University of Waterloo
Vector Institute

Abstract

Large language models (LLMs) can be improved by aligning with human preferences through fine-tuning—the so-called reinforcement learning from human feedback (RLHF). However, the cost of fine-tuning an LLM is prohibitive for many users. Due to their ability to bypass LLM fine-tuning, prediction-time tokenwise reward-guided text generation (RGTG) methods have recently been proposed. They use a reward model trained on full sequences to score partial sequences during decoding in a bid 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, we propose to train a Bradley-Terry reward model on partial sequences explicitly, and autoregressively sample from the implied tokenwise policy during decoding. We study the properties of this reward model and the resulting policy: we show that this policy is proportional to the ratio of two distinct RLHF policies. Our simple approach outperforms previous RGTG methods and performs similarly to strong offline baselines without large-scale LLM fine-tuning. Code for our work is available at https://github.com/ahmadrash/PARGS

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 performed via 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). Although 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 & Raffel (2023) explored tokenwise reward-guided text generation (RGTG) techniques that avoid any fine-tuning of the LLM. More precisely, the LLM remains frozen (i.e., not fine-tuned), and the reward model is used at the decoding time to adjust the softmax scores of the tokens directly. Unlike DPO, this line of work retains the reward modeling step, but training

*Equal Supervision

reward models is typically a much cheaper endeavor than fine-tuning text-generation LLMs since smaller models can be utilized for reward modeling. 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 based on human preference data.

While RGTG is an interesting alternative to the standard offline RLHF, it is often based on heuristics and still poorly analyzed. For instance, ARGS (Khanov et al., 2024) proposed to simply use a reward model trained on full sequences to score each partial sequence during autoregressive decoding. Meanwhile, Deng & Raffel (2023) used a custom tokenwise loss to distill a reward model trained on full sequences. Thus, it is unclear whether these approaches can give rise to a sound tokenwise text generation policy. Controlled decoding (CD; Mudgal et al., 2024), on the other hand, uses rollouts from the base model along with a reward model trained on full sequences to distill the partial reward.

In this work, we analyze this common RGTG approach. First, we show that the usage of full-sequence reward models to score partial sequences in a tokenwise policy is pathological. To alleviate this, we propose to explicitly train a Bradley-Terry (BT) reward model on partial sequences. We prove that this text generation policy is a ratio of two different RLHF policies trained on sequences of different lengths. Ideally, the policy would be derived from a single RLHF policy, but as we shall also show in Section 3, the exact computation of such a policy is intractable. By deriving the policy from a ratio of distinct RLHF policies, we obtain a tractable sampling procedure. This is akin to the argument of Zhao et al. (2024) where they use a similar ratio to derive a sequential Monte Carlo method.

We empirically validate our analysis on three different text generation datasets on two recent LLMs. Evidence shows that our RGTG approach achieves better performance compared to ARGS and CD, matching the performance of the more expensive, offline PPO and DPO baselines. In summary:

(i) We analyze the recent practice of using full-sequence reward models for guiding the LLM decoding process. In particular, we show a deficiency in this approach.
(ii) We thus propose to explicitly train a BT reward model on partial sequences and sample from the induced per-token policy induced by it during the decoding time.
(iii) We show that this reward model induces a ratio of two distinct RLHF policies over sequences with different lengths. This is a trade-off that one must make to make tokenwise RGTG free of the aforementioned deficiency and yet still tractable.
(iv) Extensive experiments with multiple LLMs on text generation validate our insights.

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 $|\mathbf{x}|$. The same notation applies to $\mathbf{y}$.

2.1 Reinforcement Learning from Human Feedback

LLMs generally consist of probabilistic models that can generate a response $\mathbf{y}$ given a prompt $\mathbf{x}$. More specifically, the generation of $\mathbf{y}$ is done 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) and 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 BT model (Bradley & Terry, 1952) which minimizes the loss:

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

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 $\mathbf{x}$ (also referred to as policy). We refer to the LLM and its policy interchangeably. One can then copy the LLM and fine-tune it to maximize

$$\max_\theta \mathop{\mathbb{E}}_{\substack{\mathbf{x}\sim\mathcal{D},\\ \mathbf{y}\sim\pi_\theta(\mathbf{y}|\mathbf{x})}} [r_\phi(\mathbf{y}|\mathbf{x})] - \frac{1}{\beta} D_{\text{KL}}[\pi_\theta(\mathbf{y}|\mathbf{x}) \,\|\, \pi_{\text{ref}}(\mathbf{y}|\mathbf{x})], \tag{2}$$

where the KL term forms a regularizer that ensures that the fine-tuned model will not differ too much from the reference model. The optimization problem above can be solved by many RL techniques, including the popular PPO algorithm (Schulman et al., 2017). This RL optimization is quite costly in practice due to the size of the LLM.

The optimization (2) has a closed form solution of the form (Peters & Schaal, 2007)

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

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. 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 fine-tunes 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 solving an equivalent optimization problem. However, note that both PPO and DPO based RLHF are still very costly in practice since they require fine-tuning (a copy of) the target LLM π_{ref}.

2.2 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.

However, 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 BT 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,

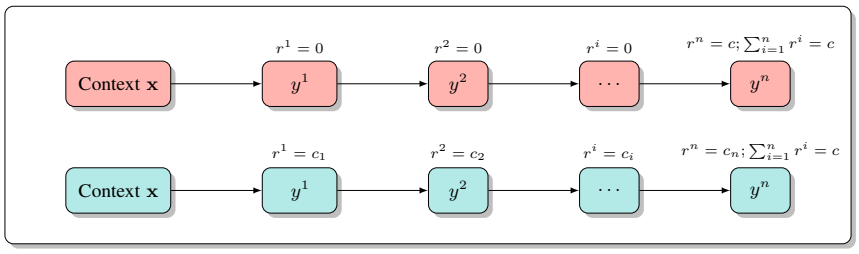

Figure 1: A pathology of using a reward model trained on full-sequence to predict partial sequences in decoding-time RGTG. We denote $r^i = r(y^i|\mathbf{x}, \mathbf{y}^{1:i-1})$. While the total reward over the full sequence $\mathbf{y} = (y^1, \ldots, y^n)$ might be nonzero c, it could be in the extreme 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 means we can have an *unguided* decoding in a *reward-guided* decoding. By explicitly training r on partial sequences, we could avoid this issue **(bottom)**: While $\mathbf{y}$ might achieve the same final reward c, nonvanishing reward signals over partial sequences could be avoided.

i.e. $r_\phi(\mathbf{y}^{1:i}|\mathbf{x})$. Hence, it is unclear whether the inferred scores for partial sequences are reasonable. 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. Meanwhile, Deng & Raffel (2023) learn the reward model by minimizing a cumulative squared loss to distill a full sequence reward model instead of using the BT loss (1), making the connection to RLHF policy looser. Mudgal et al. (2024) also distill a partial reward model from a full sequence reward model, but the tokenwise policy is not the marginal of the full-sequence policy. Nevertheless, tokens are sampled from a different tokenwise RL formulation that follows a similar derivation as RLHF.

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 fine-tune the LLM and thus, while defining a partial-sequence reward model, is not a RGTG method.

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 A.

Theorem 1. *A reward model r trained to minimize the BT 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 for an example. If we learn a reward model based on preferences over full sequences only as proposed by Khanov et al. (2024)

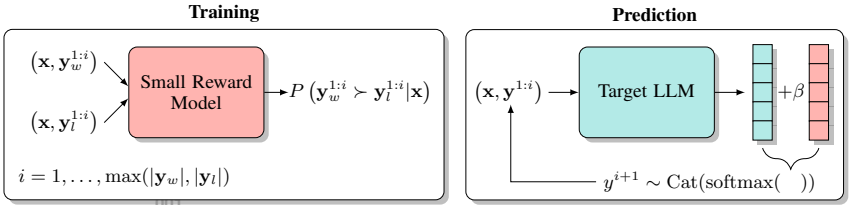

Figure 2: The proposed approach to alleviating the problem in Theorem 1. First, we (i) train the reward model r on partial sequences explicitly (when $|\mathbf{y}_w| \neq |\mathbf{y}_l|$, we pad to the longest sequence) and (ii) sample from the weighted sum of the logits and the rewards of the next token during decoding. This is in contrast to some previous RGTG methods where the reward model is trained on full sequences, but decoding relies on partial sequence scoring.

and Deng & Raffel (2023), then we may not obtain adequate rewards for partial sequences. As a concrete example, suppose that 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 the identity in (8) and therefore, could be the solution when minimizing the BT 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. Hence the ARGS method (Khanov et al., 2024) may utilize a reward model that does not score partial sequences properly, and negatively impact token-by-token generation.

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

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

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 $\mathbf{y}_l^{1:i}$. Strictly speaking, it is hard for human annotators to compare partial sequences due to their incomplete nature, and most preference datasets do not include preferences over partial sequences. Nevertheless, we can interpret $\mathbf{y}_w^{1:i}$ as the prefix of a winning sequence that is preferred over a losing sequence with prefix $\mathbf{y}_l^{1:i}$. The following lemma shows that the resulting reward model ensures that the probability that a first partial sequence is preferred to a second partial sequence corresponds to the probability that the first sequence is extended to a winning full sequence while the second sequence is extended to a losing full sequence according to the preference data distribution P_{data}. Proof in Appendix A.

Lemma 2. *In the limit of infinite preference data, optimizing a sufficiently expressive reward model according to* (4) *under the assumption that partial sequences inherit the winning/losing label of full sequences yields a reward model r_ϕ with the following property:*

$$\sigma(r_\phi(\mathbf{y}_1^{1:i}|\mathbf{x}) - r_\phi(\mathbf{y}_2^{1:j}|\mathbf{x})) = P_{\text{data}}(\mathbf{y}_1 \succcurlyeq \mathbf{y}_2 | \mathbf{x}, \mathbf{y}_1^{1:i}, \mathbf{y}_2^{1:j}), \tag{5}$$

where P_{data} is the distribution the preference dataset was sampled from and $\mathbf{y}_1 \succcurlyeq \mathbf{y}_2$ indicates that $\mathbf{y}_1$ is preferred to $\mathbf{y}_2$.

Hence, 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 (due to finite data) instead of assigning arbitrary rewards in the sense of Theorem 1.

Once the partial-sequence reward model r_ϕ is trained, we can use it to sample the next token y^i conditioned on the previous tokens $\mathbf{x}, \mathbf{y}^{1:i}$ according to the following conditional distribution:

$$\pi(y^i|\mathbf{x}, \mathbf{y}^{1:i-1}) = \frac{1}{Z(\mathbf{x}, \mathbf{y}^{1:i-1})} \pi_{\text{ref}}(y^i|\mathbf{x}, \mathbf{y}^{1:i-1}) \exp(\beta r_\phi(\mathbf{y}^{1:i}|\mathbf{x})). \tag{6}$$

Algorithm 1 summarizes the decoding procedure. Contrary to the previous approach of Khanov et al. (2024), it directly follows the policy induced by the explicitly trained reward model on partial sequences. Meanwhile, compared to Deng & Raffel (2023), it uses the standard BT model instead of a custom squared loss function that distills a full-sequence reward model.

Let us now analyze the tokenwise sampling distribution in (6). By the definition of conditional distributions, we can rewrite it as a ratio of two partial sequence distributions: $\pi(y^i|\mathbf{x}, \mathbf{y}^{1:i-1}) = \pi(\mathbf{y}^{1:i}|\mathbf{x})/\pi(\mathbf{y}^{1:i-1}|\mathbf{x})$. However, it is still unclear how this distribution relates to RLHF policies —- the main point of the tokenwise RGTG methods. The following theorem shows how the decoding process by following this distribution relates to RLHF-induced policies. Proof in Appendix A.

Theorem 3. *Given a reward model trained according to the partial-sequence BT objective in (4), the induced token generation distribution π (6) is proportional to the ratio:*

$$\pi(y^i|\mathbf{x}, \mathbf{y}^{1:i-1}) \propto \frac{\pi_{\text{RLHF},i}(\mathbf{y}^{1:i}|\mathbf{x})}{\pi_{\text{RLHF},i-1}(\mathbf{y}^{1:i-1}|\mathbf{x})} \tag{7}$$

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, 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 obtained via PPO or DPO.

Algorithm 1 Decoding with our approach.

Input: 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
Output: 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 $\mathbf{y}^{1:l}$

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 on 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) and Zhao et al. (2024), a conditional distribution over full sequences does not give us an immediate procedure for token-wise sampling. Mathematically, we can derive a token-level policy from a full-sequence policy as follows:

$$\pi_{\text{RLHF}}(y^i|\mathbf{x}, \mathbf{y}^{1:i-1}) = \frac{\pi_{\text{RLHF}}(\mathbf{y}^{1:i}|\mathbf{x})}{\pi_{\text{RLHF}}(\mathbf{y}^{1:i-1}|\mathbf{x})} = \frac{\sum_{\mathbf{y}^{i+1:|\mathbf{y}|}} \pi_{\text{RLHF}}(\mathbf{y}|\mathbf{x})}{\sum_{\mathbf{y}^{i:|\mathbf{y}|}} \pi_{\text{RLHF}}(\mathbf{y}|\mathbf{x})}.$$

However, the summations in the above equation are exponentially large in the length $|\mathbf{y}|$ of the sequences. This exponential complexity was also noted by Zhao et al. (2024) who proposed a twisted sequential Monte Carlo technique to approximate the computation and mitigate the exponential complexity. In contrast, 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. The ratio policy (6) described here can thus be seen as a necessary tradeoff if one wants to perform tokenwise RGTG without the pathology in Theorem 1.

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., 2023) have provided a direct method to align LLMs to human preferences. The PPO algorithm has been especially 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 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 (Rafailov et al., 2023; 2024) models the LLM itself as a Bradley-Terry model and optimizes the RLHF objective without any need for RL. TDPO (Zeng et al., 2024) incorporates token-level KL divergence into the DPO objective to improve content diversity. These methods, however, fine-tune the base LLM, which can be expensive as we scale. Some works have attempted to improve alignment by gathering more fine-grained rewards by using either LLMs (Cao et al., 2024) or human annotators (Wu et al., 2023).

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 value functions (Dathathri et al., 2019; Krause et al., 2021; Yang & Klein, 2021; Chaffin et al., 2022; Liu et al., 2023b). PPLM (Dathathri et al., 2019) uses the gradients from an attribute classifier to guide LLM generation. Gedi (Krause et al., 2021) uses attribute conditioned language models as discriminators to update LLM generation probabilities using Bayes rule. These algorithms are different from our work as they do not align LLMs using human preference data. Deng & Raffel (2023) use a reward model trained on preference data in the decoding process, however, they use a cumulative squared loss function that is different from the RLHF framework. Mudgal et al. (2024) uses a similar loss function with the key difference that instead of training with samples from a preference dataset, they take as input a full sequence reward model and train a partial sequence value function based on roll-outs (i.e., sampled token sequences) from the base LLM. Therefore, for each new base LLM, the value function needs to be retrained with new roll-outs, limiting portability to new or updated language models. The closest work to our method is Khanov et al. (2024), which is also based on the Bradley-Terry model, but they use a reward model trained on full sequences, which we have argued can lead to pitfalls. Different from our work, Zhao et al. (2024) present a reward-guided decoding method based on sequential Monte Carlo and show that it can approximate RLHF.

Partial Rewards Outside of preference data alignment and RLHF, prior work in reinforcement learning for language modeling has looked at partial reward models for improving text generation. Hao et al. (2022) show that a sequence to sequence model trained with supervised learning is a valid partial reward model for text generation under a Markov decision process. Lee et al. (2023) do not train an explicit reward model but instead introduce a ranking function which can rank the next token for partial sequences. Both these methods modify language model training.

TL;DR Summarization			
Method	**LLM**	**Single y?**	$r \pm$ SE
Top-k	frozen	yes	-0.11±0.28
CD	frozen	yes	0.32±0.33
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 y?**	$r \pm$ SE
Top-k	frozen	yes	-1.42±0.21
CD	frozen	yes	-1.08±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: Average reward (over 100 samples) ± standard error for the TL;DR summarization and HH dialogue tasks. The best technique that freezes the LLM and generates a single response **y** is bolded.

TL;DR Summarization			
Method A	**vs**	**Method B**	**Win-Tie (%)**
PARGS		CD	75 - 0
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		CD	52 - 8
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.

5 Experiments

We evaluate our proposed approach, which we call **P**artial **A**lignment as **R**eward-**G**uided **S**ampling (PARGS)—in contrast to ARGS which considers full sequences and greedy decoding instead of sampling—on two generation tasks: summarization and dialogue generation.

5.1 Setup

Summarization task We use the Reddit TL;DR dataset (Völske et al., 2017), where, the context $\boldsymbol{x}$ is a post on the Reddit forum and $\boldsymbol{y}$ 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 GPT2-large, fine-tuned on the TL;DR training set. We use a pre-trained reward model based on the DeBerta-v3-large architecture 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 ($N = 10$ for all our experiments) and returning the best one according to the reward model, RLHF models based on PPO and DPO, the reward-base decoding method ARGS (Khanov et al., 2024) and controlled decoding (CD; Mudgal et al., 2024). We use CD-Fudge as the baseline in all our CD experiments, noting that its performance is similar to CD-Q (see Table 4 in Mudgal 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 **x** and two responses **y** with a label indicating the preferred response. We use Llama-2-7b as the base model and DeBerta-v3 as the reward model which is about 20× smaller. Details in B.

Fine-grained text generation task We also evaluate our model on the UltraFeedback dataset (Ganqu Cui et al., 2024). We use Zephyr-7B as the base LLM and Phi-1.5 (1.3 billion parameters) as the reward model.

Ultra Feedback				Ultra Feedback			
Method	**LLM**	**Single y?**	$r \pm$ SE	**Method A**	**vs**	**Method B**	**Win-Tie (%)**
Top-k	frozen	yes	-0.18±0.12	PARGS		CD	53 - 13
CD	frozen	yes	-0.04±0.01	PARGS		ARGS	42 - 23
ARGS	frozen	yes	0.01±0.12	PARGS		Best-of-N	29 - 19
PARGS	frozen	yes	**0.21±0.09**	PARGS		Top-k	52 - 15
PARGS-G	frozen	yes	**0.21±0.12**	PARGS		DPO	65 - 7
DPO	trained	yes	-0.57±0.09				
Best-of-N	frozen	no	1.15 ±0.08				

Table 3: Average reward (100 samples) ± std. error and GPT-4 evaluation for Ultra Feedback.

Machine translation task We perform additional experiments on machine-translation on the IWSLT-2017 dataset (Cettolo et al., 2017). We used the post-edit dataset from Kreutzer et al. (2020) on the IWSLT-2017 English-German dataset to provide token-wise reward signals. We use Gemma-2b as both the base model and the reward model. The evaluation is based on the standard BLUE score.

Evaluation Following Khanov et al. (2024), we compare all methods based on *average reward*, on the test samples, as measured by the reward model. 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. Since evaluating language generation, especially unconditionally, is nuanced and human evaluation is very expensive, we use GPT-4-based evaluation, which has been shown to align with human assessment (Zheng et al., 2023; Rafailov et al., 2023). 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 score (Lin, 2004) and the BLEU score to evaluate the dialogue and translation tasks, respectively.

5.2 Results

Table 1 (left) shows the average reward for the summaries 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 $\mathbf{y}$. We also note that PARGS outperforms DPO and is competitive with PPO based RLHF that incurs a large cost to fine-tune the LLM, and Best-of-N that incurs significant overhead to generate multiple responses. Upon siginficance testing we observed PARGS to be significantly better than all algorithms expect PPO. Details in Appendix C. Note that we also evaluate our algorithm with greedy decoding (PARGS-G) for a direct comparison with ARGS.

Similarly, Table 1 (right) 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× smaller than the base LLM. Again, PARGS achieved the highest reward among the techniques that freeze the LLM and generate a single response. We observe that Best-of-N achieved the highest average reward followed by DPO, but incurred overhead to generate multiple responses and fine-tune the LLM respectively. Finally, Table 3 (left) presents average rewards on the UltraFeedback dataset. We observe that PARGS outperforms all methods except Best-of-N. Significance testing (see Appendix C) reveals that PARGS is significantly better.

Next we evaluate PARGS using GPT-4. The prompt used to probe GPT-4 is presented in Appendix J. 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 CD and ARGS, but worse than Best-of-N and DPO. As we scale training based alignment methods, e.g., DPO become prohibitive. On UltraFeedback (Table 3 right) we observe that PARGS outperforms all methods except Best-of-N. We perform human

evaluation for PARGS against ARGS, CD and DPO on UltraFeedback. PARGS wins against all three baselines. Detailed results in Appendix F.

We perform additional experiments on machine-translation on the IWSLT-2017 dataset (Cettolo et al., 2017). We used the post-edit dataset from Kreutzer et al. (2020) on the IWSLT-2017 English-German dataset to provide token-wise reward signals. We use Gemma-2b as both the base model and the reward model. The evaluation is based on the standard BLUE score.

The translation direction is English to German and the edited/corrected sequence is considered the winning sequence. Table 4 compares ARGS, PARGS-G and greedy decoding. We observe that applying ARGS reduces the BLUE score of the greedy baseline where PARGS-G increases it by 1.5 on average.

Table 4: BLEU Score on IWSLT-17 English-German

Method	**BLEU** ↑
Greedy	31.7 ± 3.6
ARGS	29.4 ± 3.4
PARGS-G	33.2 ± 3.5

We evaluate the diversity of generation on 50 samples from the UltraFeedback dataset. We compare different methods by generating 10 responses for each prompt, evaluating the Rouge-L score between each generated pair and averaging it over all the samples. A lower Rouge-L score indicates a greater diversity. Table 5 shows that PARGS generates the most diverse responses compared to top-K and DPO.

Table 5: Diversity Results

Method	**ROUGE-L** ↓
Top-k	0.230 ± 0.011
DPO	0.206 ± 0.006
PARGS	0.203 ± 0.008

Note that Best-of-N generates $N\times$ the number of samples using top-K generation.

Additional Results A discussion on decoding costs is presented in Appendix D. A study on effect of the hyperparameter β and K in top-K is presented in Appendix G. We present an empirical analysis validating our assumption that the partial sequence of a winning sequence wins over the partial sequence of a losing sequence in Appendix H.

6 Conclusion

We have 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 reward during the autoregressive decoding process and may lead to subpar performance. To alleviate this, we 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 reinforcement learning from human feedback (RLHF) policies. This means that this policy is not equivalent to the standard offline RLHF methods. However, we have also shown that it is intractable to obtain a tokenwise policy that is equivalent to a *single* RLHF policy. Our experiment results validated our approach: it performs better than recent RGTG methods such as ARGS, that leverages full-sequence reward models, and CD. We discuss limitations of our work in Appendix E.

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 CGS-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.

Meng Cao, Lei Shu, Lei Yu, Yun Zhu, Nevan Wichers, Yinxiao Liu, and Lei Meng. Beyond sparse rewards: Enhancing reinforcement learning with language model critique in text generation. *arXiv preprint arXiv:2401.07382*, 2024.

Mauro Cettolo, Marcello Federico, Luisa Bentivogli, Jan Niehues, Sebastian Stüker, Katsuhito Sudoh, Koichiro Yoshino, and Christian Federmann. Overview of the IWSLT 2017 evaluation campaign. In *Proceedings of the 14th International Conference on Spoken Language Translation*, pp. 2–14, Tokyo, Japan, December 14-15 2017. International Workshop on Spoken Language Translation. URL https://aclanthology.org/2017.iwslt-1.1.

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

Lifan Yuan Ganqu Cui, Ning Ding, Guanming Yao, Bingxiang He, Wei Zhu, Yuan Ni, Guotong Xie, Ruobing Xie, Yankai Lin, Zhiyuan Liu, and Maosong Sun. Ultrafeedback: Boosting language models with scaled ai feedback. In *ICML*, 2024.

Yongchang Hao, Yuxin Liu, and Lili Mou. Teacher forcing recovers reward functions for text generation. *NeurIPS*, 2022.

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.

Julia Kreutzer, Nathaniel Berger, and Stefan Riezler. Correct me if you can: Learning from error corrections and markings. In *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation*, 2020.

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.

Youngwon Lee, Jinu Lee, and Seung-won Hwang. Learning to rank generation with pairwise partial rewards. In *EMNLP*, 2023.

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*, pp. 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.

Sidharth Mudgal, Jong Lee, Harish Ganapathy, YaGuang Li, Tao Wang, Yanping Huang, Zhifeng Chen, Heng-Tze Cheng, Michael Collins, Trevor Strohman, et al. Controlled decoding from language models. In *ICML*, 2024.

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. In *COLM*, 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. In *ICLR*, 2023.

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*, 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*, pp. 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.

Zeqiu Wu, Yushi Hu, Weijia Shi, Nouha Dziri, Alane Suhr, Prithviraj Ammanabrolu, Noah A Smith, Mari Ostendorf, and Hannaneh Hajishirzi. Fine-grained human feedback gives better rewards for language model training. *NeurIPS*, 2023.

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.

Yongcheng Zeng, Guoqing Liu, Weiyu Ma, Ning Yang, Haifeng Zhang, and Jun Wang. Token-level direct preference optimization. In *ICML*, 2024.

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 Proofs

Theorem 1. *A reward model r trained to minimize the BT 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.*

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:

$$r(\mathbf{y}^{1:i}|\mathbf{x}) = \sum_{j=1}^{i} r(y^j|\mathbf{x}, \mathbf{y}^{1:j-1}) \qquad \text{for all } \mathbf{x}, \mathbf{y}, i \tag{8}$$

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 (8). □

Lemma 2. *In the limit of infinite preference data, optimizing a sufficiently expressive reward model according to (4) under the assumption that partial sequences inherit the winning/losing label of full sequences yields a reward model r_ϕ with the following property:*

$$\sigma(r_\phi(\mathbf{y}_1^{1:i}|\mathbf{x}) - r_\phi(\mathbf{y}_2^{1:j}|\mathbf{x})) = P_{\text{data}}(\mathbf{y}_1 \succcurlyeq \mathbf{y}_2|\mathbf{x}, \mathbf{y}_1^{1:i}, \mathbf{y}_2^{1:j}), \tag{5}$$

where P_{data} is the distribution the preference dataset was sampled from and $\mathbf{y}_1 \succcurlyeq \mathbf{y}_2$ indicates that $\mathbf{y}_1$ is preferred to $\mathbf{y}_2$.

Proof. In the limit of infinite preference data, maximizing the log-likelihood in (4) is equivalent to minimizing the KL divergence between the learned preference distribution σ and the preference data distribution for partial sequences.

$$\operatorname*{argmax}_\phi \mathop{\mathbb{E}}_{\mathbf{x},\mathbf{y}_1,\mathbf{y}_2\sim P_{\text{data}}} \log \sigma(r_\phi(\mathbf{y}_1^{1:i}|\mathbf{x}) - r_\phi(\mathbf{y}_2^{1:j}|\mathbf{x})) \tag{9}$$

$$= \operatorname*{argmin}_\phi - \mathop{\mathbb{E}}_{\mathbf{x},\mathbf{y}_1,\mathbf{y}_2\sim P_{\text{data}}} \log \sigma(r_\phi(\mathbf{y}_1^{1:i}|\mathbf{x}) - r_\phi(\mathbf{y}_2^{1:j}|\mathbf{x})) \tag{10}$$

$$= \operatorname*{argmin}_\phi \mathop{\mathbb{E}}_{\mathbf{x},\mathbf{y}_1,\mathbf{y}_2\sim P_{\text{data}}} \log \frac{P_{\text{data}}(\mathbf{y}_1 \succcurlyeq \mathbf{y}_2|\mathbf{x}, \mathbf{y}_1^{1:i}, \mathbf{y}_2^{1:j})}{\sigma(r_\phi(\mathbf{y}_1^{1:i}|\mathbf{x}) - r_\phi(\mathbf{y}_2^{1:j}|\mathbf{x}))} \tag{11}$$

$$= \operatorname*{argmin}_\phi KL(P_{\text{data}}(\mathbf{y}_1 \succcurlyeq \mathbf{y}_2|\mathbf{x}, \mathbf{y}_1^{1:i}, \mathbf{y}_2^{1:j})||\sigma(r_\phi(\mathbf{y}_1^{1:i}|\mathbf{x}) - r_\phi(\mathbf{y}_2^{1:j}|\mathbf{x}))) \tag{12}$$

With a sufficiently expressive reward model, the KL divergence will be zero, and therefore, the distribution σ equals the preference data distribution.

$$\sigma(r_\phi(\mathbf{y}_1^{1:i}|\mathbf{x}) - r_\phi(\mathbf{y}_2^{1:j}|\mathbf{x})) = P_{\text{data}}(\mathbf{y}_1 \succcurlyeq \mathbf{y}_2|\mathbf{x}, \mathbf{y}_1^{1:i}, \mathbf{y}_2^{1:j}) \tag{13}$$

□

Theorem 3. *Given a reward model trained according to the partial-sequence BT objective in (4), the induced token generation distribution π (6) is proportional to the ratio:*

$$\pi(y^i|\mathbf{x}, \mathbf{y}^{1:i-1}) \propto \frac{\pi_{\text{RLHF},i}(\mathbf{y}^{1:i}|\mathbf{x})}{\pi_{\text{RLHF},i-1}(\mathbf{y}^{1:i-1}|\mathbf{x})} \tag{7}$$

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).

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:

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

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:

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

This completes the proof of the theorem. □

B 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.

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. We report the training parameters on Table 7 and 8.

Training DPO Models We train two DPO models on the original preference dataset, one is trained based on GPT2-Large [1] on the TL;DR dataset, and the other is trained based on

[1] vistagi/gpt2-large-tldr-sum

Llama-2-7b [2] on the HH-RLHF dataset. We also adopt the TRL library to train the DPO models. The training parameters are reported on Table 9.

Partial Sequence Data Generation We randomly sample a subset of the set of all partial sequences to maintain a reasonable computational budget. We present an ablation on the tldr summarization dataset where we present the average reward achieved by PARGS when training on different subsets, as well as the wall clock time.

Dataset Size	Average Reward	Wall clock time (approx)
1x	1.64 ± 0.22	1 hour
1.5x	2.32 ± 0.19	1.5 hour
2x	2.36 ± 0.20	2. hour
3x	2.23 ± 0.20	3 hour

Table 6: Average Reward with different partial sequence dataset sizes. x is the size of the full sequence dataset.

We can observe from the results on Table 6 that we get diminishing returns when the dataset is more than 1.5x. On the TLDR dataset we sample 2x and on the other datasets 1.5x of the total dataset size. Note that the wall clock time is for training on 4 RTX6000 GPUs.

C Significance Testing

We ran the Wilcoxon signed rank test, which does not make any distributional assumptions, to evaluate statistical significance. We report the p-values below (a p-value less than 0.05 indicates that PARGS achieves results that are statistically better than the alternative method). Table 10 and Table 11 show the p-values of the rewards of PARGS vs various baselines. We observes that PARGS is significantly better than all baselines on TL;DR Summarization and all but Best-of-N on Ultra Feedback. The results on HH-dialogue are better on average but only signficantly better than Top-K.

D Decoding costs

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 & Raffel, 2023). At decode time we analyse k-tokens using the reward model. In our experiments $k = 10$, so the total inference cost is $C_{\text{forward}} + 10 C_{\text{RM}}$

	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
	β	1.5

	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
	β	2

Table 7: Training Hyperparameters for Deberta-large-v3 partial reward models

[2] argsearch/llama-7b-sft-float32

	Parameters	Value
phi1_5	Number of epoches	1
	Learning rate	2e-6
	Batch size	2
	Floating point format	fp16
	gradient accumulation steps	8
	DeepSpeed Zero stage	3
	Max. sequence length	512
	β	1

Table 8: Training Hyperparameters for Ultra Feedback reward model

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

Table 9: Training Hyperparameters for DPO models

TL;DR Summarization			
Method A	**vs**	**Method B**	**p-value**
PARGS		Top-K	6.67×10^{-14}
PARGS		CD	7.41×10^{-13}
PARGS		ARGS	4.82×10^{-6}
PARGS		Best-of-N	7.75×10^{-3}
PARGS		DPO	4.02×10^{-10}

Ultra Feedback			
Method A	**vs**	**Method B**	**p-value**
PARGS		Top-K	1.15×10^{-5}
PARGS		CD	4.07×10^{-3}
PARGS		ARGS	3.98×10^{-2}
PARGS		Best-of-N	1.0
PARGS		DPO	2.46×10^{-11}

Table 10: P-values of the reward of different methods compared to PARGS

HH-Dialogue			
Method A	**vs**	**Method B**	**p-value**
PARGS		Top-K	1.0×10^{-2}
PARGS		CD	2.01×10^{-1}
PARGS		ARGS	4.60×10^{-1}
PARGS		Best-of-N	8.9×10^{-1}
PARGS		DPO	9.9×10^{-1}

Table 11: P-values of the reward of different methods compared to PARGS

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× the base model. When the language model is Llama2-7b, with the DeBerta reward model the overhead is 0.47×. Note that the Best-of-N decoding cost overhead would always be 9×.

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.

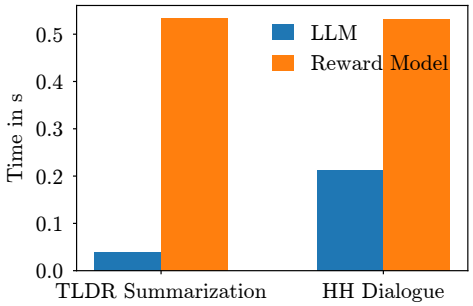

Figure 3: Runtime overhead.

E Limitations

One 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. Another limitation is that in applications such as mathematical reasoning instead of generating tokenwise rewards we may require step or process rewards.

F Human Evaluation

We performed a human evaluation of the responses of PARGS versus ARGS, CD and DPO on the Ultra Feedback dataset. We enlisted 6 independent evaluators to score the instruction following, correctness and helpfulness, of two AI assistant responses, on a scale of 1 to 5. We used the score to mark a win, tie or loss for PARGS. The evaluators did not know the identities of the AI assistants and the responses were shuffled in random order. We can observe from the results on Table 12 that PARGS has a high winning rate. We also observed a large percentage of ties.

Ultra Feedback			
Method A	**vs**	**Method B**	**Win-Tie (%)**
PARGS		ARGS	45 - 50
PARGS		CD	50 - 20
PARGS		DPO	60 - 25

Table 12: Human Evaluation based on 20 evaluations

Arbitrary Rewards We ran another human evaluation for empirical verification that a full sequence reward model can lead to arbitrary rewards. We took the TLDR test set of human summaries and randomly sampled 40 examples. Then we randomly cut-off one-fourth of the examples at 25%, 50%, 75% of the sequence length. We kept the last one-fourth at full sequence length. For each prompt the dataset had two responses. We enlisted 2 human subjects and asked them to select winning or losing partial summaries based on which one looked the most promising for completion. If they could not choose between the two they could mark a tie. Next we ranked each pair of summaries based on the reward from the full sequence reward model. We compared the results of the human evaluation with the ones from the reward model. We removed the ties from the evaluation scores.

We report the results on Table 13. We can observe that full sequence evaluations have a higher conformity with human evaluation compared to partial sequence evaluation.

Sequence Length %	Agreement %
25 %	50 %
50 %	43 %
75 %	50 %
100 %	80%

Table 13: Conformity of Full reward model with human judgement for different sequence lengths

G Sensitivity Analysis

We conduct a sensitivity test on the summarization task, using $\beta \in \{0.5, 1.0, 1.5, 2.0, 2.5\}$ and $k \in \{5, 10, 15\}$, we report the average reward and the standard deviations in Table 14, and the diversity score measure in Rouge-L in Table 15.

k/β	$\beta = 0.5$	$\beta = 1.0$	$\beta = 1.5$	$\beta = 2.0$	$\beta = 2.5$
$k = 5$	0.12 ± 0.33	3.31 ± 0.22	3.65 ± 0.22	3.72 ± 0.20	2.20 ± 0.30
$k = 10$	-0.02 ± 0.38	3.35 ± 0.20	3.88 ± 0.17	3.88 ± 0.16	2.65 ± 0.25
$k = 15$	0.61 ± 0.38	1.04 ± 0.39	2.07 ± 0.29	2.21 ± 0.23	2.88 ± 0.27

Table 14: Average Reward of summarization task with different value of β and k

k/β	$\beta = 0.5$	$\beta = 1.0$	$\beta = 1.5$	$\beta = 2.0$
$k = 5$	0.29 ± 0.03	0.31 ± 0.04	0.30 ± 0.03	0.30 ± 0.03
$k = 10$	0.27 ± 0.03	0.26 ± 0.03	0.27 ± 0.03	0.28 ± 0.03
$k = 15$	0.25 ± 0.03	0.25 ± 0.02	0.24 ± 0.03	0.28 ± 0.03

Table 15: Diversity based on ROUGE-L with different value of β and k. Lower score is better

For the reward scores, we observe that $\beta = 2.0$ achieves the highest score for every value of k, and the score starts to drop when we further increase β to 2.5. Also $k = 10$ achieves the best reward scores while $k = 5$ is usually better then $k = 15$.

Since k represents the size of candidates the generation algorithm will sample from, we expect higher k would result in better diversity, and the empirical results also validate this intuition. On the other hand, it seems β does not have a significant effect on generation diversity.

H Analysis

We empirically evaluate the assumption that the partial sequence of a winning sequence wins over the partial sequence of a losing sequence. We cut the losing sequence at 25% and 50% of the total length, and sample multiple completions from π_{ref}. Then we use a reward model to evaluate whether the winning sequence achieves a higher reward compared to all completions of the losing sequence. Table 16 shows the win-rate of a single winning sequence over multiple completions from the losing sequence at different lengths. This result demonstrates that our assumption is reasonable.

Sequence Length %	Win-Rate %
25 %	62 %
50 %	60 %

Table 16: Win-rate of winning sequence over 3 completions of the losing sequence on the TL;DR test set

I Generated Samples

The following are two example generations on Ultra-Feedback and TL;DR of all the baselines including our method.

Ultra Feedback Generation Samples

Prompt
In the Monkey King Story, how did Monkey king fight with vampire? Assistant:

Generations
[PARGS] As per the legend, Monkey King fought with a vampire named Yin Yang Shou, who was the leader of the demons. Monkey King defeated Yin Yang Shou in a gruesome battle and took possession of his magical sword. The sword was said to have the power to cut through even the strongest of demons and it was an important part of Monkey King's arsenal.

[DPO] Monkey King fought with vampire in the story of Journey to the West.

[ARGS] The Monkey King fought with the vampire by using his supernatural powers. He used his strength to fight the vampire and eventually defeated him.

[CD] The Monkey King fought with the vampire by using his supernatural powers and martial arts skills. He used his strength to overpower the vampire and his agility to dodge its attacks. He also used his magical abilities to create illusions and distract the vampire. In the end, the Monkey King was able to defeat the vampire and save the day.

[Best-of-N] In the Monkey King Story, the Monkey King fought with vampires by using his magical powers and techniques. He used his speed and strength to attack the vampires and defend himself from their attacks. Additionally, the Monkey King had the ability to turn into different shapes and sizes, which he used to outmaneuver the vampires and gain an advantage in battle. Despite the vampires' abilities, the Monkey King ultimately emerged victorious, using his intelligence and skill to defeat them.

TL;DR Generation Samples

Prompt
SUBREDDIT: r/relationships
TITLE: Me [19 M] with my GF [19 F] broke up after almost 3 years, hard time staying friends
POST: We broke up for college last year and I know I had a rough time, going from constantly seeing a best friend to not even communicating. It was an understandably mutual decision as we were both going forward in life, but it still hurt a lot.

Anyways after a few months of college, I thought I was done with her. Then she began drunk texting me quite often. I was civil at first then ignored them as I realized it was just hurting me and going nowhere. So I got over her again.Now that we're back for summer, she's been wanting to hang out with mutual friends as we were all close. I thought I was at peace and had gotten over her but just seeing her again ruined my day. I felt (and still feel) like I'm going through the breakup all over again.

Avoiding her is out of the question since my best friends are close friends of hers as well. We were all one big happy family and now there's this awkward rift between my ex and I that is stemming from my confused feelings and was damn near palpable to my friends when we all got together today.

Also, I would never want to leave my friends, especially because of her.

TL;DR:

Generations
[PARGS] Ex and I broke up a few months ago, still hurting and I'm having trouble avoiding her, especially since best friends are close friends of hers.

[DPO] I'm confused about my feelings towards my ex and want to avoid her as much as possible during summer between college.

[ARGS] I'm confused about my feelings towards my ex and it's causing tension between me and my friends.

[CD] I am confused as hell and having trouble coping. I am a wreck and would love some advice.

[Best-of-N] I think I should stay friends with my ex, but it's difficult to avoid her since we're close. Any advice on what I should do?

J 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]