SCAR: Shapley Credit Assignment for More Efficient RLHF

Anonymous ACL submission

Abstract

Reinforcement Learning from Human Feedback (RLHF) often suffers from sparse reward signals, making credit assignment to tokens challenging. We propose Shapley Credit Assignment Rewards (SCAR), a method that uses Shapley values from cooperative game theory to distribute sequence-level rewards across tokens or text spans according to their marginal contributions. SCAR provides dense rewards without requiring auxiliary critique models or fine-grained human annotations, while offering a principled game-theoretic foundation for credit assignment. We show theoretically that SCAR preserves the original optimal policy, and empirically demonstrate across sentiment control, summarization, and instruction tuning tasks that it converges faster and achieves higher final rewards than standard RLHF and attention-based dense reward baselines. To improve scalability on long sequences, we further introduce an Owen value–inspired adaptive text segmentation strategy. Our findings suggest that SCAR provides a more effective and theoretically sound method for credit assignment in RLHF, leading to more efficient alignment of LLMs.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities in diverse natural language tasks, yet aligning their outputs with complex human instructions and preferences remains a critical challenge (Brown et al., 2020; Bai et al., 2022). Reinforcement Learning from Human Feedback (RLHF) has emerged as a powerful paradigm for this alignment (Christiano et al., 2017; Stiennon et al., 2020; Ouyang et al., 2022), enabling models to generate more helpful, harmless, and honest responses. The standard RLHF pipeline typically involves training a reward model (RM) on human preference data and then using this RM to fine-tune the LLM via reinforcement learning algorithms like PPO (Schulman et al., 2017).

A fundamental challenge in this pipeline arises from the nature of reward delivery. LLMs generate text auto-regressively, selecting one token at a time. Each token selection constitutes an action within an RL framework. However, the RM typically provides only a single scalar reward signal upon the completion of the entire sequence (Stiennon et al., 2020; Bai et al., 2022). This results in a sparse reward setting (Hare, 2019), which is notoriously difficult for RL algorithms to optimize efficiently (Yu, 2018). The core difficulty lies in credit assignment (Sutton, 1984; Lansdell et al., 2019): determining which specific token choices (actions) early in the generation process were responsible for the final high or low reward. This sparsity can lead to slow convergence, high variance gradients, training instability (Henderson et al., 2018; Razin et al., 2024), and potentially suboptimal final policies.

Recognizing this limitation, recent work has focused on densifying the reward signal specifically for RLHF. One approach involves training dense reward models using fine-grained human annotations at intermediate generation steps (Lightman et al., 2023; Wu et al., 2023a). While effective, this approach incurs substantial annotation costs. To mitigate these costs, auxiliary LLMs have been proposed as critics to generate intermediate feedback (Klissarov et al., 2024; Cao et al., 2024; Yoon et al., 2024), though this relies on the critic's inherent capabilities. An alternative strategy, most relevant to our work, aims to derive dense rewards by redistributing the existing sequence-level reward from the RM without new annotations or auxiliary models. A notable example is Attention-Based Credit (ABC) (Chan et al., 2024), which repurposes the RM's final layer's attention weights to proportionally distribute the final reward across tokens.

While ABC offers a practical heuristic, relying on attention weights for credit assignment presents several limitations. For example, attention mecha-

nisms are not explicitly designed for, nor do they offer a theoretically grounded guarantee of, fair credit attribution; their interpretability for attribution is debated (Jain and Wallace, 2019; Wiegreffe and Pinter, 2019).

In our pursuit of a more principled approach to dense credit assignment in RLHF, we turn to cooperative game theory and propose Shapley Credit Assignment Rewards (SCAR). The core idea is to model sequence generation as a cooperative game where individual text segments (e.g., tokens, or larger spans like phrases or sentences) are the players. The value of any coalition of these players (i.e., a subset of segments) is determined by querying the reward model with the partial sequence formed by concatenating the segments in that coalition. We then employ the Shapley value (Shapley, 1953) to fairly distribute the total sequence-level reward among these segments. This distribution is based on each segment's average marginal contribution across all possible orderings of player arrivals, offering a theoretically grounded method that uniquely satisfies desirable axioms such as *efficiency, symmetry*, and *linearity*, thereby ensuring a fair and principled allocation of credit (Shapley, 1953).

Recognizing that the exact computation of Shapley values is exponentially complex in the number of players and thus prohibitive for long sequences, we incorporate adaptive text segmentation to maintain a tractable number of players. We leverage constituency parsing to segment the generated text into a hierarchical structure of syntactically coherent units (e.g., phrases or clauses), which then serve as the players. This significantly reduces the computational burden by grouping semantically related tokens. For longer responses, segmentation at the sentence level provides a further practical alternative. We empirically validate SCAR across three NLP tasks with different response lengths, demonstrating that it consistently leads to faster convergence and achieves higher final reward scores compared to standard sparse RLHF and attention-based dense reward baselines.

2 Related Work

2.1 RL for Text Generation

Reinforcement Learning (RL) has been increasingly leveraged for fine-tuning LLMs in text generation tasks (Ryang and Abekawa, 2012; Norouzi et al., 2016; Li et al., 2016; Buck et al., 2018; Bahdanau et al., 2017). Unlike standard supervised fine-tuning which relies on maximizing the likelihood of ground-truth sequences, RL enables optimization directly towards sequence-level objectives or metrics that are not differentiable, such as ROUGE scores in summarization or human preferences for dialogue quality (Stiennon et al., 2020; Ouyang et al., 2022). A particularly successful application of this is Reinforcement Learning from Human Feedback (RLHF) (Ouyang et al., 2022; Christiano et al., 2017; Stiennon et al., 2020; Bai et al., 2022), which has become a widely adopted technique for aligning LLMs with complex human values and instructions. The typical RLHF process involves learning a reward model (RM) from a dataset of human comparisons between different model outputs, followed by fine-tuning the LLM policy to maximize the scalar reward assigned by the RM to the generated text sequences. However, a widely acknowledged challenge in this standard RLHF framework is the inherent sparsity of the reward signal: the RM typically provides feedback only after the entire sequence has been generated (Stiennon et al., 2020; Bai et al., 2022). This terminal reward makes the temporal credit assignment problem—identifying which specific token choices (actions) contributed positively or negatively to the final outcome—particularly difficult (Sutton, 1984).

2.2 Dense Reward Strategies

Ng et al. (1999) laid the groundwork for potential-based reward shaping in RL, demonstrating that such shaping can effectively reduce training time without changing the optimal policy. This concept has inspired subsequent research in augmenting learning processes through auxiliary rewards. More recently, Attention Based Credit (ABC) (Chan et al., 2024) proposed using the reward model's internal attention weights to redistribute the terminal reward across tokens. ABC provides density "for free" and preserves optimality via potential-based reward shaping (PBRS) (Ng et al., 1999).

2.3 Shapley Values in LLMs

Several previous works have already proposed approaches to interpreting the behaviour of LLMs using Shapley values. Goldshmidt and Horovicz (2024) enhanced LLM output interpretability by calculating the Shapley values of sub-sequences. Mohammadi (2024) demonstrated how the Shapley value can uncover that the LLM decisions are

disproportionately influenced by tokens providing minimal informative content. Liu et al. (2023) used Shapley values to quantify the value of prompts equitably in multi-prompt learning methods. In addition to enhancing the interpretability of LLM outputs, Shapley value methods are also used for model pruning/compression (Sun et al., 2025) and dataset refinement (He et al., 2024). However, to the best of our knowledge, there's limited work on solving the sparse reward problem in RLHF using Shapley values. We note contemporaneous work by Koo et al. (2025) which also explores Shapley values for reward distribution. Their approach estimates per-token rewards using interpretability methods and then employs Bayesian Optimization (BO) in a bilevel framework to learn parameters for a shaping function that combines these estimates. Our work, SCAR, differs in two ways. First, to address the inefficiency of token-level Shapley calculations for longer responses, SCAR allows for adaptive text segmentation, including grouping tokens into larger syntactic spans (e.g., via constituency parsing) for more efficient credit assignment. Secondly, unlike Koo et al. (2025) requiring an outer optimization loop (like BO) to learn shaping parameters, SCAR directly uses the (approximated) Shapley values from these text units to construct the dense reward signal (Equation 6).

3 Shapley Credit Assignment Rewards

This section details our proposed method, Shapley Credit Assignment Rewards (SCAR). We first review the standard RLHF setup and the inherent reward sparsity problem. We then introduce a game-theoretic perspective on text generation for credit assignment, define SCAR based on Shapley values, and finally discuss efficient approximation techniques.

3.1 RLHF and Reward Sparsity

We formulate the autoregressive text generation process using the standard Markov Decision Process (MDP) formalism, denoted by $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$. The process begins in an initial state $s_0 \in \mathcal{S}$, which corresponds to the input prompt x. At each discrete time step $t \in \{0, 1, \ldots, T-1\}$, the agent (the language model) is in state s_t, representing the concatenation of the prompt and the sequence generated thus far: $s_t = x \oplus y_{1:t}$ (where $y_{1:0}$ denotes an empty sequence, so $s_0 = x$, and $\oplus$ denotes token concatenation). The agent selects an action $a_t \in \mathcal{A}$, which corresponds to choosing the next token y_{t+1} from the vocabulary $\mathcal{V}$ according to its policy $\pi_\theta(a_t|s_t)$, parameterized by θ. We identify the action space with the vocabulary, i.e., $\mathcal{A} = \mathcal{V}$. The state transition function $P(s_{t+1}|s_t, a_t)$ is deterministic in this context. Upon taking action $a_t = y_{t+1}$ in state s_t, the system transitions to next state $s_{t+1} = s_t \oplus a_t = x \oplus y_{1:t+1}$. This generation process continues until a maximum sequence length T is reached or a designated end-of-sequence (EOS) token is generated. For notational simplicity, we often assume a fixed horizon T. The discount factor γ is typically set to 1 in finite-horizon text generation tasks.

In standard RLHF pipeline (Christiano et al., 2017; Stiennon et al., 2020; Ouyang et al., 2022), a reward model R_ϕ, parameterized by ϕ, is trained beforehand on a dataset of human preferences $\mathcal{D}_{\text{pref}} = \{(y^w, y^l)_i\}$, where y^w is preferred over y^l by human annotators. The reward model assigns a scalar score $r_\phi(x, y)$ reflecting the quality or preference level of a *completed* sequence y given the prompt x.

To stabilize training and prevent the policy π_θ from drifting too far from a reference distribution (often the initial pre-trained LLM, denoted π_{ref}), the reward signal used for optimization is typically augmented with a penalty term at each step t. A common choice is the Kullback-Leibler (KL) divergence between the current policy π_θ and the reference policy π_{ref}. The standard objective is often formulated as:

$$\mathcal{J}(\theta) = \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_\theta}\left[\sum_{t=1}^{T} R_t^{\text{orig}}\right] \tag{1}$$

where $\mathcal{D}$ is the dataset and the reward at each timestep t in the standard sparse setup is given by

$$R_t^{\text{orig}} = R_t^{\text{KL}} + \mathbb{I}(t = T) \cdot r_\phi(x, y) \tag{2}$$

Here, $R_t^{\text{KL}} = -\beta \log(\pi_\theta(y_t|x, y_{<t})/\pi_{\text{ref}}(y_t|x, y_{<t}))$ is the KL penalty associated with timestep t, β is the KL coefficient, and $\mathbb{I}(t = T)$ is an indicator function ensuring the reward model score $r_\phi(x, y)$ is assigned only at the final timestep T. This terminal assignment makes the reward signal inherently *sparse*, posing a significant challenge for credit assignment during RL training. Such sparsity directly undermines the efficiency of the learning process and frequently leads to the convergence

to suboptimal policies (Ng et al., 1999; Bellemare et al., 2016; Wu et al., 2023b). To overcome this, a principled approach to redistribute the terminal reward more densely across the generation steps is desirable.

3.2 A Game-Theoretic Framework for Credit Assignment

We frame the generation of a sequence y for a given prompt x as a cooperative game. Let the generated text y be segmented into N contiguous units, $y = (u_1, u_2, \ldots, u_N)$. These units could be tokens, spans, sentences, or paragraphs depending on the task. The "players" in this game are these N text units. Let $\mathcal{P} = \{u_1, \ldots, u_N\}$ be the set of players.

Characteristic Function. The value of cooperation among a subset (coalition) $S \subseteq \mathcal{P}$ of players is defined by a characteristic function $v : 2^{\mathcal{P}} \to \mathbb{R}$. This function should quantify the collective contribution of the units in S towards the final reward objective. We define $v(S)$ based on the reward model's evaluation of the partial text sequence formed by concatenating the units $\{u_i \mid u_i \in S\}$ in their original order. Let y_S denote this concatenated partial sequence. Then, the value function is defined as:

$$v(S) = r_\phi(x, y_S) \tag{3}$$

For the empty set, $v(\emptyset) = 0$. The value of the grand coalition $v(\mathcal{P})$ corresponds to the original sparse reward for the complete sequence, $v(\mathcal{P}) = r_\phi(x, y)$. Note that evaluating r_ϕ on partial sequences y_S requires careful consideration, as y_S represents an incomplete sequence. Ideally, $v(S)$ could represent the expected reward obtained by keeping the units in S fixed and sampling the remaining units from the current policy π_θ. In our implementation, we resort to a practical approximation. To evaluate $v(S)$, we construct a sequence in which the tokens belonging to units $u_i \in S$ are placed in their original order. The positions corresponding to units $u_j \notin S$ are filled using empty spaces.

Shapley Value Calculation. The Shapley Value $\mathrm{SV}_{u_i}(v)$ for a player $u_i \in \mathcal{P}$ (text unit u_i) quantifies its fair contribution to the grand coalition value $v(\mathcal{P})$, calculated as the average marginal contribution of player u_i over all possible permutations of player arrivals:

$$\mathrm{SV}_{u_i}(v) = \sum_{S \subseteq \mathcal{P} \setminus \{u_i\}} \frac{|S|!\,(N - |S| - 1)!}{N!} [v(S \cup \{u_i\}) - v(S)]. \tag{4}$$

The Shapley value uniquely satisfies axioms such as *efficiency* ($\sum_{u_i \in \mathcal{P}} \mathrm{SV}_{u_i}(v) = v(\mathcal{P})$), *symmetry* (equal reward for equal contribution), *linearity*, and the *null player* property (no contribution means no credit), making it a principled choice for fair credit allocation (Shapley, 1953).

3.3 Shapley Values as Dense Rewards

We use the calculated Shapley values to define a dense reward signal for the RL agent. Let unit u_i consist of tokens generated from timestep $t_{i-1} + 1$ up to and including timestep t_i (with $t_0 = 0$, so t_i marks the completion timestep of unit u_i). We assign the Shapley value $\mathrm{SV}_{u_i}(v)$ associated with unit u_i as an additional reward component specifically at the timestep t_i marking the completion of that unit. Let R_t^{shap} denote this Shapley-based reward at timestep t. Then,

$$R_t^{\text{shap}} = \begin{cases} \mathrm{SV}_{u_i}(v) & \text{if } t = t_i \text{ for some unit } u_i \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

This component distributes the total reward $r_\phi(x, y)$ across the episode based on the Shapley contributions, since $\sum_{t=1}^{T} R_t^{\text{shap}} = \sum_{i=1}^{N} \mathrm{SV}_{u_i}(v) = r_\phi(x, y)$ (due to the efficiency property, where N is the total number of units).

We then define the total reward R_t provided to the RL agent at timestep t as a convex combination of the dense Shapley-based credit allocation and the original sparse terminal reward, while retaining the per-step KL penalty. Using a hyperparameter $\alpha \in [0, 1]$, the total reward is:

$$R_t(\alpha) = R_t^{\text{KL}} + \alpha \cdot R_t^{\text{shap}} + (1-\alpha) \cdot \mathbb{I}(t = T) \cdot r_\phi(x, y) \tag{6}$$

Here, α controls the interpolation:

- If $\alpha = 0$, $R_t(0) = R_t^{\text{orig}}$, recovering the standard sparse reward signal.
- If $\alpha = 1$, $R_t(1) = R_t^{\text{KL}} + R_t^{\text{shap}}$. The terminal reward $r_\phi(x, y)$ is entirely replaced by an equivalent value distributed densely according to Shapley contributions throughout the episode.
- If $0 < \alpha < 1$, the agent receives both the intermediate Shapley-based rewards and a residual portion of the original terminal reward.

This formulation allows flexible control over the density of the reward signal, balancing immediate feedback with the final outcome signal.

Theorem 3.1 (Policy Invariance under SCAR Reward Shaping)**.** *Consider a parameterized language model π_θ with a learned reward model R_ϕ. Let $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R_t^{orig}, \gamma)$ be the original MDP with its reward from the reward model and $\widehat{\mathcal{M}} = (\mathcal{S}, \mathcal{A}, P, R_t(\alpha), \gamma)$ be the MDP with dense Shapley reward. If π_θ is optimal for $\widehat{\mathcal{M}}$, then π_θ is also optimal for $\mathcal{M}$, and vice versa.*

Proof. See Appendix A. □

3.4 Efficient Approximation of Shapley Values

The direct calculation of Shapley values using Equation (4) necessitates evaluating the characteristic function $v(S)$ (defined in Eq. (3)) for all 2^N possible coalitions S of the N text units. This exponential complexity renders exact computation practically infeasible for typical sequence lengths encountered in text generation (Shapley, 1953). To make SCAR practical, we employ two key strategies: adaptive segmentation of text into units and efficient approximation of their Shapley values.

Adaptive Text Segmentation as Players. The definition of "players" (text units u_i) in our cooperative game is crucial for both interpretability and computational tractability. We adapt the granularity of these units based on the task and the typical length of the generated responses, aiming to keep the number of players N manageable. We experiment with three levels of segmentation:

- **Token-level:** For tasks producing very short responses, each token y_t can be treated as an individual player u_i. This offers the finest granularity but results in a larger N.
- **Span-level:** For medium-length responses, we leverage constituency parsing (Marcus et al., 1993) to establish a hierarchical grammatical structure over the generated sequence y. This process yields a constituency tree where tokens (leaf nodes) are organized into hierarchically nested constituents. Players are then defined as these syntactic constituents (e.g., noun phrases, verb phrases), formed by grouping tokens that share a common parent or ancestor node within this tree. This approach reduces N while preserving semantic coherence within each player unit, as constituents are inherently meaningful linguistic units.[1]
- **Sentence-level:** For tasks generating longer, multi-sentence responses, each sentence in the output y constitutes a player. Segmentation is achieved using standard sentence boundary detection. This approach markedly reduces N, especially for verbose outputs.

The choice of segmentation strategy is a hyperparameter, allowing a trade-off between the granularity of credit assignment and the computational cost of Shapley value estimation.

Approximation Using Owen Values. To ensure the practical applicability of SCAR, we employ an approximation scheme based on Owen value (Owen, 1977), which is a coalitional extension of Shapley values designed for games where players are grouped into a predefined coalition structure (Aumann and Dreze, 1974). For our task, a hierarchical structure $\mathcal{B}$ is imposed on the sequence of N text units, achieved by applying a heuristic parsing algorithm to the units. This partition $\mathcal{B}$ defines nested groupings of the units. The Owen value is then computed with respect to this partition $\mathcal{B}$. Marginal contributions are evaluated by forming coalitions structurally: combinations involving subsets within a unit's own group are explored, while units belonging to other groups in the partition are treated as indivisible blocks, as they are either entirely included or entirely excluded from a coalition, rather than exploring all their individual subsets. By limiting the evaluation to coalitions dictated by the partition structure $\mathcal{B}$, the number of required characteristic function evaluations (reward model queries) is substantially reduced compared to the exact Shapley computation. Consequently, the computational complexity is reduced from exponential, $O(2^N)$, to quadratic in N, rendering the approach tractable. We use the SHAP package (Lundberg and Lee, 2017) for Shapley values and Owen values computation.

4 Experiments

In this section, we empirically evaluate the effectiveness of SCAR across three distinct tasks characterized by varying response lengths. Our primary objective is to demonstrate that SCAR enables more efficient and effective training compared to standard sparse RLHF and alternative dense reward baselines.

[1] https://www.nltk.org/howto/parse.html

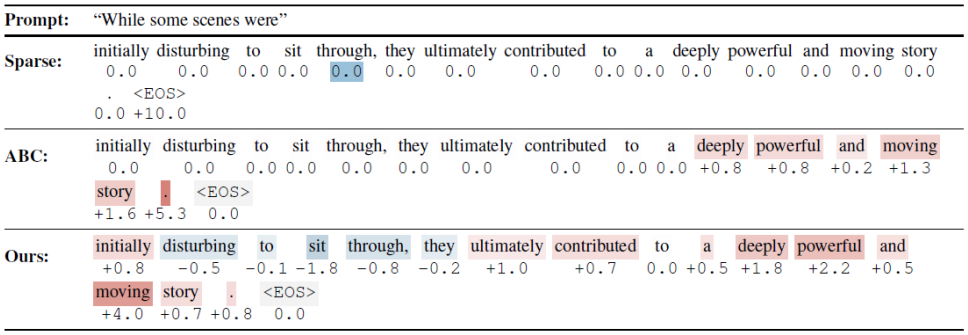

Figure 1: Comparison of reward distribution strategies for an example generated sequence. Sparse RLHF assigns the total reward at the end. SCAR and ABC distribute this reward across tokens/spans based on their respective methodologies, shown with background highlights (color hue for sign, intensity for magnitude; more intense/saturated means higher absolute contribution) and numerical scores.

4.1 Experimental Setup

Evaluation Tasks and Models. We evaluate our proposed method across three diverse tasks prevalent in RLHF research: sentiment control, text summarization, and instruction tuning (Chan et al., 2024; Cao et al., 2024; Yoon et al., 2024). For sentiment control and instruction tuning, we utilize the implementation from Chan et al. (2024). However, for summarization, due to difficulties in reproducing the results, we switched to the implementation from Huang et al. (2024). The policy models we tested are Qwen3 8B (Yang et al., 2025), Qwen2.5-7B (Yang et al., 2024), and the Llama3.1-8B model (Dubey et al., 2024). The datasets and reward models used for each task are described in more detail below.

Sentiment Control: The objective is to generate positive reviews of movies. We use the IMDB dataset (Maas et al., 2011). The policy model is initialized by fine-tuning for one epoch on the IMDB training set. During RLHF training, prompts are generated by randomly selecting the first 4 to 8 tokens from reviews in the training data. The reward signals are provided using a pre-trained sentiment classifier, same as (Chan et al., 2024).

Text Summarization: We evaluate our method on the automatic text summarization task, following prior work (Stiennon et al., 2020; Chan et al., 2024; Lee et al., 2023). For this evaluation, we use the Reddit TL;DR dataset (V"olske et al., 2017), specifically the filtered version provided by Stiennon et al. (2020), which includes approximately 116K training examples, 6K validation examples, and 6K test examples. We initialize policy models via supervised fine-tuning on the training set for 2,000 steps with a batch size of 64. Additionally, we train a 1B-parameter reward model (initialized using the SFT model) on 92K human preference pairs, achieving approximately 74% accuracy on the validation set.

Instruction Tuning: We evaluate models on the task of following user instructions. To do this, we fine-tune language models using the helpfulness subset of the Anthropic Helpful and Harmless (HH) dataset (Bai et al., 2022), which contains 43K human-written prompts paired with model responses that have been ranked by human annotators for helpfulness. Preference is based on which response is more informative and helpful for the task. For the reward model, we use the 3B reward model provided by Dong et al. (2023). This reward model was trained using the same HH dataset, where it learns to assign a scores to candidate completions based on their predicted usefulness.

Baselines. We compare our proposed method against three key baselines. **RLHF** represents the standard approach using the sparse terminal reward (Eq. 2) with KL regularization. **ABC (Attention Based Credit)** (Chan et al., 2024) uses reward model attention scores for a dense reward distribution. **Uniform** is a baseline that distributes the terminal reward evenly across all tokens. For fair comparison, all methods are optimized using the PPO objective (Schulman et al., 2017) with consistent hyperparameters, detailed in Appendix C. All methods initialize their policy models using the same SFT checkpoint to ensure a common starting point. All experiments were conducted on A100 GPUs (80GB VRAM), and results are averaged over 5 random seeds.

Evaluation Metrics. We track the average reward $r_\phi(x, y)$ per episode during training to eval-

uate learning speed and the level of convergence. The final performance is reported as the mean reward on the test set after convergence. For the summarization task, we additionally employ **LLM-as-a-judge** (Zheng et al., 2023) evaluation to compare the quality (e.g., accuracy, coverage, conciseness, clarity, and coherence) of summaries generated by models trained with different methods. We randomly sampled 1K summaries from the TL;DR test set for LLM evaluation. To mitigate potential positional bias in these pairwise comparisons, we randomize the presentation order of summaries. For the instruction-tuning task, we use AlpacaEval (Li et al., 2023) to compare the quality of 1K model's response. AlpacaEval is designed to better handle potential issues such as length bias, thereby providing a more reliable assessment of response quality. The prompt used for evaluation can be found in the Appendix D.

4.2 Results

Figure 1 provides a qualitative illustration of how SCAR distributes rewards compared to sparse RLHF and ABC for an example generated sequence. Sparse RLHF, by definition, assigns the entire reward only at the end of the sequence. ABC, which uses attention scores from the reward model's final layer to distribute rewards, tends to concentrate rewards on tokens near the end of the sequence. As seen in the example, significant credit is assigned to the final punctuation mark (“.”), while earlier, crucial tokens receive almost zero attention scores. Furthermore, standard attention scores are non-negative, making it difficult for ABC to assign explicit negative credit to tokens or spans that detract from the output quality. For instance, a phrase like “*disturbing to sit through*” that negatively impacts the perceived sentiment, would not receive negative rewards from ABC. In contrast, SCAR can assign both positive and negative rewards to tokens based on their game-theoretic marginal contribution.

Table 1 presents the average reward scores achieved by each method on recent models. As shown in the table, SCAR-tuned policy consistently achieves the highest performance compared to the baselines. This consistent pattern across diverse tasks and models suggests that the principled, Shapley value-based credit assignment offered by our method effectively improves learning efficiency and enhances the final policy performance.

Across all three tasks—sentiment control, SCAR consistently demonstrated significant advantages over the baseline methods, as depicted in Figure 2. Compared to standard sparse RLHF and the Uniform baseline, SCAR exhibited markedly faster convergence, reaching higher reward levels earlier in the training process. When compared to the attention-based dense reward method (ABC), SCAR generally showed a steeper initial learning curve and achieved comparable or superior final reward performance. This consistent pattern across diverse tasks suggests that the principled, Shapley value-based credit assignment offered by SCAR effectively accelerates learning and enhances the final policy quality in RLHF by providing a more informative, dense reward signal.

Table 1: Average rewards for the trained policy on the test sets for sentiment control (IMDB), text summarization (TL;DR), and instruction tuning (Anthropic HH) with recent models. Higher scores indicate better performance. Results are averaged values over 5 random seeds.

Task	Model	RLHF	ABC	SCAR
IMDB	Qwen3-8B	8.32± 0.24	8.97± 0.37	**9.75± 0.13**
	Qwen2.5-7B	7.92± 0.36	8.60± 0.17	**9.61± 0.26**
	Llama3.1-8B	7.18± 0.21	8.44± 0.27	**9.37± 0.14**
TL;DR	Qwen3-8B	4.15± 0.10	4.22± 0.14	**4.79± 0.05**
	Qwen2.5-7B	3.72± 0.07	4.14± 0.09	**4.55± 0.02**
	Llama3.1-8B	3.58± 0.09	3.95± 0.13	**4.37± 0.07**
HH-RLHF	Qwen3-8B	7.42± 0.24	7.34± 0.49	**8.11± 0.23**
	Qwen2.5-7B	7.25± 0.31	7.28± 0.25	**7.74± 0.22**
	Llama3.1-8B	7.08± 0.27	6.82± 0.30	**7.58± 0.26**

Table 2: LLM-as-Judge pairwise win rates for SCAR against baselines.

Baselines	Win (%)
Text Summarization (Reddit TL;DR)	
vs. RLHF	61.2%
vs. ABC	60.3%
Instruction Tuning (Anthropic HH)	
vs. RLHF	56.3%
vs. ABC	54.9%

For summarization, we use `gemini-2.5-pro` to compare anonymized model outputs (SCAR vs. baselines) on coherence, relevance, conciseness, and overall quality (evaluation prompt in Appendix D). As shown in Table 2, summaries generated by the SCAR-tuned model were preferred over those from the ABC-tuned model in 60.3% and over the sparse RLHFC-tuned model in 61.2%. For the instruction tuning task, we leveraged AlpacaEval (Li et al., 2023) and used `gpt-4-turbo` as an LLM judge, to conduct robust

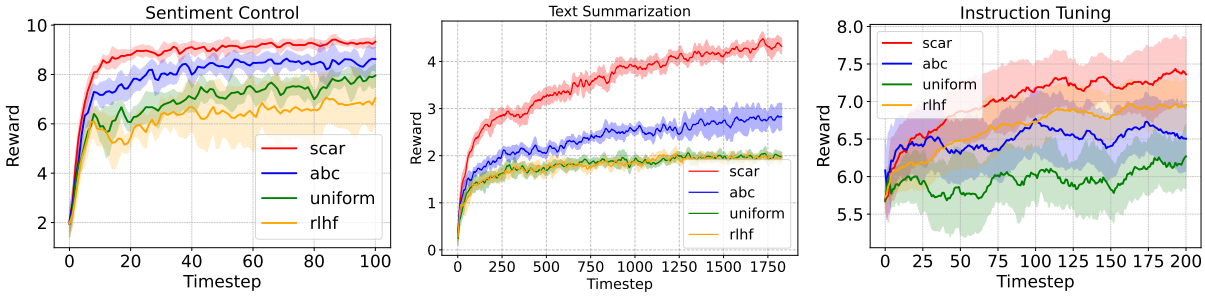

Figure 2: Average reward per timestep during RLHF training for sentiment control (left), text summarization (center), and instruction tuning (right). Curves show the mean reward across five random seeds, with shaded regions representing the standard deviation. SCAR consistently demonstrates faster convergence and achieves higher or comparable final reward levels compared to sparse RLHF, Uniform reward distribution, and Attention-Based Credit (ABC) baselines. To demonstrate SCAR's effectiveness across a broader range of models, we use Pythia (Biderman et al., 2023) and OpenLLaMA model for the text summarization and instruction tuning task.

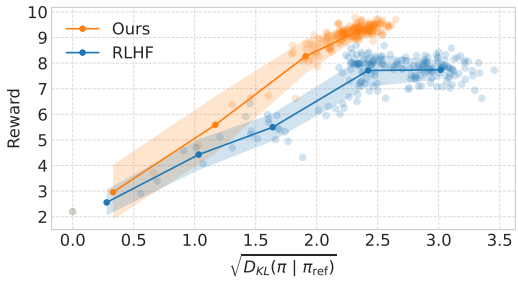

Figure 3: Reward-KL tradeoff on the sentiment control (IMDB) task. The y-axis represents the average per-batch reward during training and the x-axis shows the square root of the KL divergence between the learned policy (π) and the reference policy (π_{ref}).

pairwise evaluations. These evaluations assessed helpfulness, harmlessness, and adherence to instructions. SCAR-generated responses achieved a win rate of 54.9% when compared against ABC, and a win rate of 56.3% against standard sparse RLHF. These results provide further evidence that the improvements from SCAR translate to genuinely higher-quality outputs according to human-like preferences.

4.3 Analysis

Reward-KL Tradeoff. A crucial aspect of RLHF is managing the trade-off between maximizing the reward signal and maintaining proximity to a reference policy (π_{ref}). This is often controlled by a KL-divergence penalty term in the reward function (Eq. 2), preventing the policy from drifting too far and generating undesirable or out-of-distribution text, or "reward hacking". Figure 3 illustrates this trade-off by plotting the achieved reward against the KL-divergence from the reference policy $D_{\text{KL}}(\pi \mid\mid \pi_{\text{ref}})$. The plot demonstrates that our method achieves a more favorable reward-KL frontier compared to the standard sparse RLHF baseline. This improved frontier suggests that the principled credit assignment from SCAR enables the improvements are genuine and do not come at the cost of generating out-of-distribution or gibberish text.

Computational Efficiency. Details can be seen in Appendix B.4.

Evaluation of partial sequences We conducted a series of diagnostic evaluations to assess the ability of the reward models (RMs) to handle partial or incomplete sequences. Representative examples are provided in Appendix B.5.

5 Conclusion

We introduce Shapley Credit Assignment Rewards (SCAR), a novel method to address reward sparsity in RLHF by generating dense rewards using Shapley values. Unlike heuristic approaches, SCAR provides a principled, game-theoretic allocation of the reward model's score to text segments. It preserves the optimal policy via potential-based reward shaping and, empirically, demonstrated significantly faster convergence and superior final performance across multiple tasks compared to standard RLHF and other dense reward baselines. Moreover, by efficiently approximating Shapley values with Owen values, SCAR significantly reduces computational overhead while retaining theoretical policy invariance and delivering strong empirical performance.

Limitations

Despite its strengths, SCAR has limitations: the computational overhead of Shapley values, even with optimizations like Owen values and adaptive segmentation, is still non-negligible, especially for extremely long sequences. The assumption that the reward model can meaningfully score partial sequences, which may not suit certain types like rule-based models that only evaluate final answers (e.g., in mathematical reasoning). Future work will target more efficient approximation techniques, robust and adaptive segmentation methods, and rigorous evaluation on larger-scale language models and broader tasks.

References

Robert J Aumann and Jacques H Dreze. 1974. Cooperative games with coalition structures. *International Journal of game theory*, 3:217–237.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. An actor-critic algorithm for sequence prediction. In *International Conference on Learning Representations*.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, and 1 others. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. 2016. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, and 1 others. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Christian Buck, Jannis Bulian, Massimiliano Ciaramita, Wojciech Gajewski, Andrea Gesmundo, Neil Houlsby, and Wei Wang. 2018. Ask the right questions: Active question reformulation with reinforcement learning. In *International Conference on Learning Representations*.

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

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

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. 2023. Raft: Reward ranked finetuning for generative foundation model alignment. *arXiv preprint arXiv:2304.06767*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

Roni Goldshmidt and Miriam Horovicz. 2024. Tokenshap: Interpreting large language models with monte carlo shapley value estimation. *arXiv preprint arXiv:2407.10114*.

Joshua Hare. 2019. Dealing with sparse rewards in reinforcement learning. *arXiv preprint arXiv:1910.09281*.

Yexiao He, Ziyao Wang, Zheyu Shen, Guoheng Sun, Yucong Dai, Yongkai Wu, Hongyi Wang, and Ang Li. 2024. Shed: Shapley-based automated dataset refinement for instruction fine-tuning. *arXiv preprint arXiv:2405.00705*.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. 2018. Deep reinforcement learning that matters. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press.

Shengyi Huang, Michael Noukhovitch, Arian Hosseini, Kashif Rasul, Weixun Wang, and Lewis Tunstall. 2024. The n+ implementation details of RLHF with

PPO: A case study on TL;DR summarization. In *First Conference on Language Modeling*.

Sarthak Jain and Byron C. Wallace. 2019. Attention is not Explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3543–3556, Minneapolis, Minnesota. Association for Computational Linguistics.

Martin Klissarov, Pierluca D'Oro, Shagun Sodhani, Roberta Raileanu, Pierre-Luc Bacon, Pascal Vincent, Amy Zhang, and Mikael Henaff. 2024. Motif: Intrinsic motivation from artificial intelligence feedback. In *The Twelfth International Conference on Learning Representations*.

Ryan Koo, Ian Yang, Vipul Raheja, Mingyi Hong, Kwang-Sung Jun, and Dongyeop Kang. 2025. Learning explainable dense reward shapes via bayesian optimization. *arXiv preprint arXiv:2504.16272*.

Benjamin James Lansdell, Prashanth Ravi Prakash, and Konrad Paul Kording. 2019. Learning to solve the credit assignment problem. *arXiv preprint arXiv:1906.00889*.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Ren Lu, Thomas Mesnard, Johan Ferret, Colton Bishop, Ethan Hall, Victor Carbune, and Abhinav Rastogi. 2023. Rlaif: Scaling reinforcement learning from human feedback with ai feedback.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202, Austin, Texas. Association for Computational Linguistics.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Alpacaeval: An automatic evaluator of instruction-following models. `https://github.com/tatsu-lab/alpaca_eval`.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Hanxi Liu, Xiaokai Mao, Haocheng Xia, Jian Lou, and Jinfei Liu. 2023. Prompt valuation based on shapley values. *arXiv preprint arXiv:2312.15395*.

Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Behnam Mohammadi. 2024. Explaining large language models decisions using shapley values. *arXiv preprint arXiv:2404.01332*.

Andrew Y Ng, Daishi Harada, and Stuart Russell. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pages 278–287. Citeseer.

Mohammad Norouzi, Samy Bengio, zhifeng Chen, Navdeep Jaitly, Mike Schuster, Yonghui Wu, and Dale Schuurmans. 2016. Reward augmented maximum likelihood for neural structured prediction. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc.

Guilliermo Owen. 1977. Values of games with a priori unions. In *Mathematical economics and game theory: Essays in honor of Oskar Morgenstern*, pages 76–88. Springer.

Noam Razin, Hattie Zhou, Omid Saremi, Vimal Thilak, Arwen Bradley, Preetum Nakkiran, Joshua M. Susskind, and Etai Littwin. 2024. Vanishing gradients in reinforcement finetuning of language models. In *The Twelfth International Conference on Learning Representations*.

Seonggi Ryang and Takeshi Abekawa. 2012. Framework of automatic text summarization using reinforcement learning. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 256–265, Jeju Island, Korea. Association for Computational Linguistics.

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

Lloyd S Shapley. 1953. Stochastic games. *Proceedings of the national academy of sciences*, 39(10):1095–1100.

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

Chuan Sun, Han Yu, and Lizhen Cui. 2025. Efficient shapley value-based non-uniform pruning of large language models. *arXiv preprint arXiv:2505.01731*.

Richard Stuart Sutton. 1984. *Temporal credit assignment in reinforcement learning*. University of Massachusetts Amherst.

Michael V"olske, Martin Potthast, Shahbaz Syed, and Benno Stein. 2017. TL;DR: Mining Reddit to learn automatic summarization. In *Proceedings of the Workshop on New Frontiers in Summarization*, pages 59–63, Copenhagen, Denmark. Association for Computational Linguistics.

Sarah Wiegreffe and Yuval Pinter. 2019. Attention is not not explanation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 11–20, Hong Kong, China. Association for Computational Linguistics.

Zeqiu Wu, Yushi Hu, Weijia Shi, Nouha Dziri, Alane Suhr, Prithviraj Ammanabrolu, Noah A Smith, Mari Ostendorf, and Hannaneh Hajishirzi. 2023a. Fine-grained human feedback gives better rewards for language model training. *arXiv preprint arXiv:2306.01693*.

Zeqiu Wu, Yushi Hu, Weijia Shi, Nouha Dziri, Alane Suhr, Prithviraj Ammanabrolu, Noah A Smith, Mari Ostendorf, and Hannaneh Hajishirzi. 2023b. Fine-grained human feedback gives better rewards for language model training. In *Advances in Neural Information Processing Systems*, volume 36, pages 59008–59033. Curran Associates, Inc.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024. Qwen2. 5 technical report. *arXiv e-prints*, pages arXiv–2412.

Eunseop Yoon, Hee Suk Yoon, SooHwan Eom, Gunsoo Han, Daniel Nam, Daejin Jo, Kyoung-Woon On, Mark Hasegawa-Johnson, Sungwoong Kim, and Chang Yoo. 2024. TLCR: Token-level continuous reward for fine-grained reinforcement learning from human feedback. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 14969–14981, Bangkok, Thailand. Association for Computational Linguistics.

Yang Yu. 2018. Towards sample efficient reinforcement learning. In *IJCAI*, pages 5739–5743.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

A Optimality Preservation

We demonstrate that optimizing with the dense reward signal $R_t(\alpha)$ (defined in Eq. 6) leads to the same optimal policy as optimizing with the original sparse reward R_t^{orig} (defined in Eq. 2).

Proof of Theorem 3.1 In our context, the state s_t corresponds to the history $(x, y_{<t})$. The transition $s_t \to s_{t+1}$ involves selecting action y_t. The discount factor $\gamma = 1$. We need to show that the difference between our Shapley values-based reward and the original reward constitutes such a potential function difference.

Let $F_t = R_t(\alpha) - R_t^{\text{orig}}$ be the shaping term added at timestep t. Substituting the definitions from Eq. 6 and Eq. 2:

$$\begin{aligned}
F_t &= \Big(R_t^{\text{KL}} + \alpha \cdot R_t^{\text{shap}} + (1-\alpha)\cdot \mathbb{I}(t=T)\cdot r_\phi(x,y)\Big) \\
&\quad - \big(R_t^{\text{KL}} + \mathbb{I}(t=T)\cdot r_\phi(x,y)\big) \\
&= \alpha R_t^{\text{shap}} + (1-\alpha)\mathbb{I}(t=T) r_\phi(x,y) - \mathbb{I}(t=T) r_\phi(x,y) \\
&= \alpha R_t^{\text{shap}} - \alpha \mathbb{I}(t=T) r_\phi(x,y)
\end{aligned}$$

Let's consider the total undiscounted return for an episode $y = (y_1, \ldots, y_T)$. The total return under the original reward is $G^{\text{orig}} = \sum_{t=1}^{T} R_t^{\text{orig}}$. The total return under the SCAR reward is $G^{\text{shap}}(\alpha) = \sum_{t=1}^{T} R_t(\alpha)$. We have:

$$\begin{aligned}
G^{\text{shap}}(\alpha) &= \sum_{t=1}^{T} R_t(\alpha) = \sum_{t=1}^{T} (R_t^{\text{orig}} + F_t) \\
&= \sum_{t=1}^{T} R_t^{\text{orig}} + \sum_{t=1}^{T} F_t \\
&= G^{\text{orig}} + \sum_{t=1}^{T} \Big(\alpha R_t^{\text{shap}} - \alpha \mathbb{I}(t=T) r_\phi(x,y)\Big) \\
&= G^{\text{orig}} + \alpha \left(\sum_{t=1}^{T} R_t^{\text{shap}}\right) - \alpha r_\phi(x,y)
\end{aligned}$$

By the efficiency property of Shapley Values, $\sum_{i=1}^{M} \text{SV}_i(v) = v(N) = r_\phi(x,y)$.

Since $\sum_{t=1}^{T} R_t^{\text{shap}} = \sum_{i=1}^{M} \text{SV}_i(v)$, we have $\sum_{t=1}^{T} R_t^{\text{shap}} = r_\phi(x, y)$. Substituting this back:

$$\begin{aligned} G^{\text{shap}}(\alpha) &= G^{\text{orig}} + \alpha r_\phi(x, y) - \alpha r_\phi(x, y) \\ &= G^{\text{orig}} \end{aligned}$$

The total undiscounted reward accumulated over any complete episode is identical for both the original reward R_t^{orig} and the SCAR reward $R_t(\alpha)$.

Since the objective in RL is to maximize the expected total return, $J(\pi) = \mathbb{E}_{\tau\sim\pi}[G]$, and we have shown that $G^{\text{shap}}(\alpha) = G^{\text{orig}}$ for any episode (trajectory) τ, it follows that the expected total returns are identical for any policy π:

$$J^{\text{shap}}(\pi) = \mathbb{E}_{\tau\sim\pi}[G^{\text{shap}}(\alpha)] = \mathbb{E}_{\tau\sim\pi}[G^{\text{orig}}] = J^{\text{orig}}(\pi) \tag{7}$$

Because the objectives $J^{\text{shap}}(\pi)$ and $J^{\text{orig}}(\pi)$ are identical for all policies π, any policy π^* that maximizes J^{shap} must also maximize J^{orig}, and vice versa.

Therefore, optimizing the policy π_θ using the dense reward $R_t(\alpha)$ is equivalent to optimizing using the original sparse reward R_t^{orig} in terms of the resulting optimal policy. This ensures that the benefits of denser rewards provided by Shapley values reshaping (e.g., faster convergence, improved stability) do not come at the cost of altering the fundamental goal of the optimization.

B Additional Experimental Results

In this section, we present additional experiments that could not be included in the main text due to space limitations.

B.1 Ablations on span length settings

For a more adaptive span configuration, we introduce a hyperparameter p that denotes the length ratio between a partial sequence and the full sequence. For example, $p = \frac{1}{10}$ indicates that each partial sequence contains one-tenth of the tokens in the full sequence. This segmentation scheme is heuristic-free and can be applied consistently across diverse text domains and tasks.

Table 3 presents the results under different values of p. We observe that performance generally improves as the sequence is divided into finer-grained segments. However, the gains begin to saturate when the average segment length reaches approximately one-tenth of the full sequence length. Further reducing the segment size yields only marginal improvements while substantially increasing computational cost.

B.2 Baselines for Span-Level Dense Reward Settings

To verify that the observed performance improvements arise from the Shapley credit assignment mechanism itself—rather than simply from introducing span-level dense supervision—we compare SCAR against two additional span-level dense reward baselines. The first baseline directly uses the reward model's evaluations of individual spans as dense rewards ("Direct span reward"). The second baseline redistributes the final sequence-level reward across spans, where the allocation weights are determined by the reward model's span evaluations ("Weighted span reward").

The results are shown in Table 4. Across both tasks and model scales, SCAR consistently outperforms the two alternative dense reward formulations. This suggests that the gains are not solely attributable to the presence of span-level supervision, but rather to the principled credit assignment provided by the Shapley-based formulation.

B.3 Token-level vs Span-level SCAR

For the text summarization task, we implemented and compared token-level and span-level SCAR. As illustrated by their reward curves in Figure 4, span-level SCAR achieved performance comparable to that of token-level SCAR. However, span-level SCAR demonstrated significantly greater computational efficiency: on our training platform using the Pythia-1b model, 1K training steps required 48 GPU hours for token-level SCAR versus only 7 GPU hours for span-level SCAR. This makes span-level SCAR approximately seven times more efficient in this specific task (a more detailed time consumption analysis is provided in Figure 5). Due to the high computational cost of token-level SCAR, a statistical study involving multiple runs was infeasible for this method. Consequently, Figure 4 presents a single representative run for each approach to illustrate their comparative performance.

B.4 Cost analysis

Calculating Shapley values introduces computational overhead compared to sparse RLHF or ABC. Figure 5 plots reward against GPU hours on the TL;DR task. It shows that token-level SCAR, due to its per-step cost, lag behind ABC in reward per GPU hour. However, span-level SCAR demonstrates a much better efficiency profile. Despite

Table 3: Performance under different fixed-length segmentation granularities.

Task + Model	$\mathbf{p = \frac{1}{20}}$	$\mathbf{p = \frac{1}{10}}$	$\mathbf{p = \frac{1}{5}}$	$\mathbf{p = \frac{1}{2}}$	$\mathbf{p = 1}$ (Sparse)
TL;DR, Qwen3-8B	4.72 ± 0.13	4.74 ± 0.11	4.63 ± 0.17	4.12 ± 0.08	4.15 ± 0.10
TL;DR, Qwen2.5-7B	4.50 ± 0.08	4.49 ± 0.07	4.41 ± 0.15	3.74 ± 0.10	3.72 ± 0.07
HH-RLHF, Qwen3-8B	7.96 ± 0.30	8.01 ± 0.28	7.68 ± 0.24	7.47 ± 0.22	7.42 ± 0.24
HH-RLHF, Qwen2.5-7B	7.78 ± 0.26	7.70 ± 0.25	7.37 ± 0.23	7.24 ± 0.19	7.25 ± 0.31

Table 4: Comparison of rewards under different types of span-level dense reward settings.

Task + Model	**Direct span reward**	**Weighted span reward**	**SCAR**
TL;DR, Qwen3-8B	3.94 ± 0.10	4.15 ± 0.13	4.79 ± 0.05
TL;DR, Qwen2.5-7B	3.61 ± 0.07	4.07 ± 0.07	4.55 ± 0.02
HH-RLHF, Qwen3-8B	7.15 ± 0.29	7.40 ± 0.31	8.08 ± 0.19
HH-RLHF, Qwen2.5-7B	7.06 ± 0.36	7.22 ± 0.29	7.69 ± 0.22

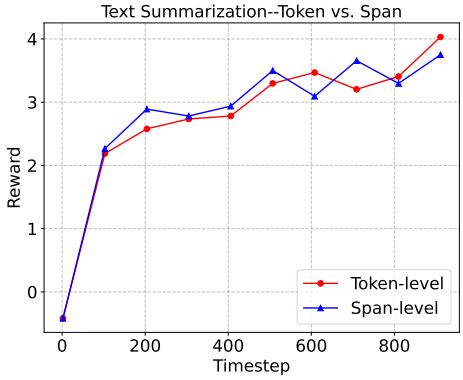

Figure 4: Comparison between token-level and span-level SCAR on the text summarization (TL;DR) task. The y-axis represents the reward during training, and the x-axis shows the training timestep.

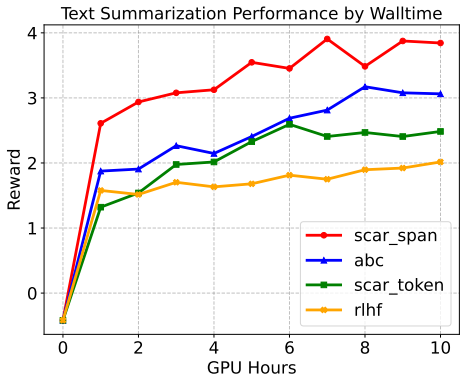

Figure 5: Rewards/GPU hours curves on the TL;DR dataset. We sampled one run from each method. The y-axis represents the reward and the x-axis shows the GPU hours used for training.

the inherent cost of Shapley calculations, its superior sample efficiency allows span-level SCAR to achieve higher reward levels than baselines within the same GPU time. This highlights that by managing the per-step cost through effective segmentation, SCAR's principled dense rewards can lead to greater overall training efficiency.

In addition to using dense reward for high sample efficiency, we should also focus on the extra time introduced by calculating the dense reward. As we analyzed in section 3.4, the time complexity of calculating Shapley values is $O(2^N)$. When we use the token-level Shapley value, n would be the length of the generated text. As shown in figure 5, token-level SCAR fails to achieve ABC's performance within a limited time. Using span-level SCAR (based on Owen values) significantly reduces N, achieving higher reward within the same time scale.

Additionally, to better characterize the computational overhead of different partition strategies, we report the approximate number of reward model (RM) forward passes per sequence, the average number of subsequences, and the proportion of total runtime spent on RM forward evaluation and Shapley value computation.

Table 5 shows that finer-grained partitioning substantially increases computational cost, particularly for long-form dialogue tasks such as HH-RLHF. In contrast, moderate partition granularities provide a more favorable balance between efficiency and performance. Specifically, partitioning each sequence into approximately 10 subsequences achieves the best performance–efficiency tradeoff, with only around a $1.5\times$ increase in overall runtime compared to sparse reward training.

B.5 Evaluations of partial sequences

To provide further evidence of the reward models' accuracy on partial or incomplete sequences,

Table 5: Reward model forward pass counts under different partition strategies.

Task + Partition Method	Avg. RM Forward Passes	Avg. # Subsequences	% Total Time
TL;DR, span	75.3	12.4	49.2
TL;DR, sentence	18.5	6.2	13.5
TL;DR, $p = \frac{1}{5}$	12.3	5	7.2
TL;DR, $p = \frac{1}{10}$	57.2	10	38.6
HH-RLHF, span	881.4	53.7	95.6
HH-RLHF, sentence	10.9	5.6	6.1
HH-RLHF, $p = \frac{1}{5}$	9.8	5	5.7
HH-RLHF, $p = \frac{1}{10}$	52.1	10	36.9
HH-RLHF, $p = \frac{1}{20}$	235.0	20	86.3

Table 6: Sentiment control partial/incomplete sequence analysis. Overall response score: 9.9683. Response: "While some scenes were initially disturbing to sit through, they ultimately contributed to a deeply powerful and moving story."

Text	Score
disturbing to sit through	-6.2960
deeply powerful	8.5214
moving story	6.6705
story	1.5888
scenes	1.6383
they ultimately contributed	1.6652
moving	1.3947
powerful	1.6845
disturbing	-6.2683
While some scenes were initially to sit through, they ultimately contributed to a deeply powerful and moving story.	9.9356
While some scenes were initially disturbing to sit through, they ultimately contributed to a deeply powerful and story.	9.3892
While some scenes were initially disturbing to sit through, they ultimately contributed to a and moving story.	9.1769

Table 7: Instruction-following partial sequence analysis. The query is: "Do you have any good video card suggestions for a good gaming PC?" Overall response score: 7.3948.

Segment	Score
Let me see what my sources say. Hmm, interesting!	1.7638
It seems that the NVIDIA 1080 and the AMD RX 5700 have very positive reviews.	6.9662
However, the GTX 1080 is the top pick for the best GPU (Graphics Processing Unit), which is a kind of supercomputer that takes in your input and sends out a high-res image to your monitor.	4.5639
I can do that very quickly.	0.1931
Let me know your budget or screen resolution, and I can suggest the best option for you.	4.7314

we evaluate the models by feeding different segments of a given example into the reward model and record the resulting scores. The results are presented in Table 6 and Table 7.

A few insights can be given based on the examples:

- RMs provide reliable scores for partial sequences of moderate length.
- Reward models can effectively identify semantically rich or task-relevant segments (e.g., "deeply powerful" and "disturbing to sit through" for sentiment control, or specific GPU models in response to a video card recommendation query), while assigning lower scores to irrelevant or low-quality segments (e.g., "Let me see what my sources say. Hmm, interesting!" and "I can do that very quickly.").
- Scores for very short sequences are less reliable (e.g., the single positive words "moving" and "powerful" receive scores of only 1.3947 and 1.6845, respectively). However, this limitation is mitigated by the Shapley value mechanism, which averages marginal contributions across all possible sequence combinations, including longer and more informative subse-

quences that yield more accurate scores. This effect is supported by a comparison between individual word scores and their corresponding Shapley values, as shown in Figure 1.

We emphasize that this does not undermine the utility of the resulting Shapley values. The key requirement is not that the reward model produce perfectly calibrated scores on partial sequences, but rather that its evaluations preserve a monotonic preference structure. In particular, as long as partial sequences containing task-relevant content are consistently assigned higher scores than those without such content, the resulting Shapley values can still effectively identify the core lexical or semantic units that contribute to the final reward.

To better illustrate this phenomenon, we conduct an additional analysis using Qwen3-8B to identify task-relevant units under different segmentation granularities. We then examine whether partial sequences containing at least one relevant unit are systematically preferred by the reward model.

In Table 8, "Probability" denotes the frequency with which the reward model assigns a higher score to order-preserving, randomly constructed partial sequences containing at least one relevant unit than to partial sequences containing none. "Average Score Advantage" measures the mean score difference between these two groups. "Final Score" reports the downstream performance achieved by SCAR under the corresponding granularity setting.

The results show that, despite potential OOD effects, the reward model consistently exhibits a strong monotonic preference for sequences containing task-relevant units. Moreover, settings with stronger monotonicity generally correlate with better downstream performance, supporting our claim that relative preference consistency is sufficient for effective Shapley-based credit assignment.

Therefore, the RM is reliable when evaluating long enough partial sequences. While RM evaluation may be noisy for extremely short fragments, this noise does not significantly affect the calculated Shapley values and SCAR's overall performance.

C Table of Hyperparameters

To ensure experimental reproducibility, this section details the hyperparameters employed. Our Proximal Policy Optimization (PPO) implementation for sentiment control and instruction tuning experiments is based on the Hugging Face TRL library[2] and code from (Chan et al., 2024). For the text summarization task, our implementation is built upon the codebase available at https://github.com/vwxyzjn/summarize_from_feedback_details.

[2] https://huggingface.co/docs/trl/main/en/ppo_trainer

Table 8: Monotonic preference analysis of reward model evaluations on partial sequences under different segmentation granularities.

Task + Granularity	Probability	Avg. Score Advantage	Final Score (Qwen3-4B)
IMDB, token-level	85.1%	9.826	9.34
IMDB, span-level	81.4%	9.661	9.47
TL;DR, span-level	84.9%	3.354	4.12
TL;DR, sentence-level	88.7%	3.472	4.33
HH-RLHF, span-level	74.4%	6.204	7.48
HH-RLHF, sentence-level	77.6%	6.891	7.71

Table 9: Hyperparameters used in the sentiment control task.

Hyperparameter	Value
PPO Hyperparameters	
Learning rate	1.41e-5
Batch size	128
Mini-batch size	128
Gradient accumulation steps	1
KL coefficient (β)	0.2
Shapley value coefficient (α)	0.8
Discount factor (γ)	1
Clip range	0.2
LLM Hyperparameters	
Min generation length	16
Max generation length	24
Temperature	0.7
Top-p	1.0

Table 10: Hyperparameters used in the text summarization task.

Hyperparameter	Value
PPO Hyperparameters	
Learning rate	3e-6
Batch size	64
Mini-batch size	16
Gradient accumulation step	4
KL coefficient (β)	0.05
Shapley value coefficient (α)	1.0
Over-length sequence reward penalty	-1
Discount factor (γ)	1
Clip range	0.2
LLM Hyperparameters	
Generation length	53
Temperature	0.7
Top-p	1.0

Table 11: Hyperparameters used in the instruction tuning task.

Hyperparameter	Value
PPO Hyperparameters	
Learning rate	1.41e-5
Batch size	16
Mini-batch size	2
Gradient accumulation step	8
KL coefficient (β)	0.2
Shapley value coefficient (α)	0.8
Discount factor (γ)	1
Clip range	0.2
LoRA rank	32
LoRA α	32
LoRA dropout	0.0
LLM Hyperparameters	
Min generation length	8
Max generation length	256
Temperature	1.0
Top-p	1.0

D Evaluation Prompt for Text Summarization

This section presents the prompt used to query the LLM for summarization quality evaluation. To mitigate potential position bias, the baseline summary and the summary generated by our method were randomly assigned to the `summary_A` and `summary_B` placeholders within the prompt.

Human Evaluation Prompt for Text Summarization

You are an expert human evaluator specializing in text summarization. Your task is to meticulously compare two summaries, "Summary A" and "Summary B," generated from the same "Original Document." Your goal is to determine which summary is of higher quality overall.
Please consider the following criteria in your evaluation:

1. **Accuracy & Faithfulness:**
   - Does the summary accurately represent the main points of the original document?
   - Does it avoid introducing new information or misinterpreting facts from the document (hallucinations)?
2. **Coverage & Comprehensiveness:**
   - Does the summary cover the most important information and key takeaways from the original document?
   - Are there any critical omissions of essential information?
3. **Conciseness & Succinctness:**
   - Is the summary brief and to the point, avoiding unnecessary jargon, redundancy, or overly verbose phrasing, while still capturing essential information?
   - Is it significantly shorter than the original document, as a good summary should be?
4. **Clarity & Readability:**
   - Is the summary well-written, grammatically correct, easy to understand, and fluent?
   - Is the language clear and precise?
5. **Coherence:**
   - Do the sentences in the summary flow logically? Does it make sense as a standalone piece of text?
   - Is there a logical structure to the summary?

Input:
Original Document:

```
{original_document}
```

Summary A:

```
{summary_A}
```

Summary B:

```
{summary_B}
```

Instructions for your response:

1. **Reasoning:**
   - First, briefly state your understanding of the main purpose or key points of the **Original Document**.
   - Then, provide a step-by-step comparative analysis of Summary A and Summary B based on the criteria listed above (Accuracy, Coverage, Conciseness, Clarity, Coherence).
   - For each criterion, explicitly compare A and B. For instance, "Regarding Accuracy, Summary A does X well, while Summary B struggles with Y..."
   - Point out specific strengths and weaknesses of each summary. You can reference parts of the summaries or the original document if helpful (e.g., "Summary A correctly captures the detail about X from paragraph 2 of the document, whereas Summary B omits this.").

2. **Overall Decision:**
   - After your detailed reasoning, clearly state which summary you believe is better overall and why, making a holistic judgment. If they are of very comparable quality, or if one excels in some areas while the other excels in others making a clear choice difficult, you can indicate that.

Output Format:
First, provide your detailed **Reasoning** as described above. Then, on a new line, write "**Overall Decision:**" followed by your overall assessment. Finally, on a separate, new line, output *only* one letter:

- 'A' if Summary A is better.
- 'B' if Summary B is better.
- 'C' if both summaries are of very similar quality (a tie), or if one is not definitively superior to the other across the most important criteria.

Begin your evaluation now.