SHAPE-GRPO: SHAPLEY-ENHANCED REWARD ALLOCATION FOR MULTI-CANDIDATE LLM TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

In user-agent interaction scenarios such as recommendation, brainstorming, and code suggestion, Large Language Models (LLMs) often generate sets of candidate recommendations where the objective is to maximize the collective utility of the entire set rather than individual candidates independently. However, existing reinforcement learning post-training paradigms, such as Group Relative Policy Optimization (GRPO), typically assign the same set-level scalar reward to every candidate in the set. This leads to noisy training signals where poor candidates free-ride on the high reward produced by a single strong peer, resulting in suboptimal exploration. To address this, we propose Shapley-Enhanced GRPO (ShapE-GRPO). By leveraging the permutation-invariant nature of set-level utility, we derive a Shapley-enhanced formulation from cooperative game theory to decompose set-level rewards into granular, candidate-specific signals. We show that our formulation preserves the fundamental axioms of the Shapley value while remaining computationally efficient with polynomial-time complexity. Empirically, ShapE-GRPO consistently outperforms standard GRPO across diverse datasets with accelerated convergence during training.

1 INTRODUCTION

In many user–agent interaction scenarios, LLM-based agents generate multiple candidate recommendations from which users select one to execute. Examples include shopping (Li et al., 2023), summarization (Zhang et al., 2025a), brainstorming ideas (Lim & Perrault, 2024), and code suggestions (Chen et al., 2021). In such settings, the objective of generation is no longer to maximize the reward of each individual candidate independently, but rather to maximize the usefulness of the entire set of candidates presented to the user. For instance, consider a user asking for places to visit near Seattle for a weekend trip. If recommendations are generated solely based on the popularity of individual destinations, an LLM might suggest Mount Rainier, Olympic National Park, and the North Cascades. While each destination is individually popular, this set of recommendations is suboptimal because these locations are highly correlated: a user who is not interested in outdoor activities is unlikely to visit any of them. In contrast, replacing one of these national parks with a destination such as the Boeing Factory Tour, although individually less popular, can significantly improve the coverage of user preferences. As a result, the probability that at least one candidate matches the user's interests increases. In applications such as shopping recommendations, this directly translates into higher click-through rates and improved downstream value.

The core research question of this paper is how to effectively allocate a set-level reward signal to individual candidates within that set to guide more efficient exploration during LLM post-training. Traditional algorithms, such as proximal policy optimization (PPO) (Schulman et al., 2017) and GRPO (Shao et al., 2024), typically assign rewards at the level of a single complete response. In scenarios where rewards are determined by a *set* of candidates, these methods assign the *same* scalar reward to every candidate in the set. This approach leads to a significant *reward allocation problem*. For example, in a brainstorming task where a model generates K ideas, a single brilliant suggestion might result in a high set-level reward. Under traditional frameworks, the remaining $K-1$ mediocre or poor ideas would "inherit" that high reward, creating noisy training signals and reinforcing suboptimal behavior. Furthermore, the scale of the state-action space compounds this inefficiency. If a model is capable of generating N distinct ideas and is prompted to produce a set of K candidates, the number of possible combinations grows combinatorially at $\mathcal{O}(N^K)$. Without

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

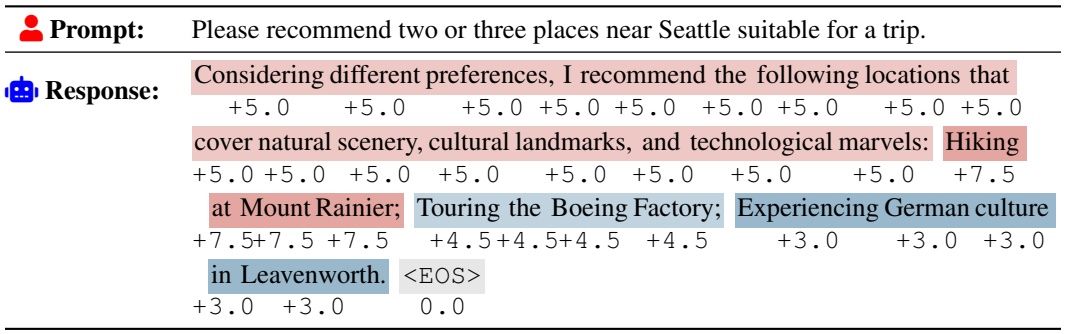


Figure 1: When the user considers the ratings for the three suggestions to be 5.0, 4.0, and 3.0, respectively, our ShapE-GRPO performs candidate-level (with broadcasting to token-level) reward allocation as above. However, GRPO only uses a single +5.0 sequence-level reward shared across tokens.

a mechanism to provide granular feedback within each set, exploring and learning from such a vast combinatorial space becomes highly sample-inefficient.

In settings where LLMs act as recommenders generating multiple candidates, the semantic utility and the resulting reward are often *permutation-invariant*, e.g., industrial-scale systems at YouTube (Ie et al., 2019), meaning that the total value is independent of the candidates' ordering within the set. This characteristic naturally motivates the use of the Shapley value from cooperative game theory (Shapley et al., 1953) as a principled mechanism for fair reward allocation. Leveraging this framework, we propose an appropriately granular *Shapley-enhanced* reward allocation mechanism. We introduce a candidate-level reward formulation inspired by the Shapley value that decomposes the set-level reward and broadcasts it to the token level. Our approach preserves the fundamental axiomatic properties of the Shapley value: *efficiency*, *symmetry*, *additivity*, and the *null player* property (see Appendix B for details). Furthermore, we demonstrate that under our specific formulation, these exact values can be computed in *polynomial* time, ensuring the method is computationally viable for large-scale LLM post-training. Finally, empirical evaluations across multiple datasets show that our approach significantly enhances training efficiency and model performance compared to the standard GRPO baseline.

1.1 Our Contributions

Shapley-Enhanced reward allocation for multi-candidate generation. We propose ShapE-GRPO, a Shapley-enhanced reward allocation framework with an appropriate granularity that provides effective candidate-level reward signals for RL post-training in multi-candidate generation settings. It fills the gap in how rewards at the set level can be assigned during LLM post-training, and the underlying candidate-level Shapley values satisfy the standard axioms from cooperative game theory.

Unbiased, informative and efficiently computable candidate-level rewards. We theoretically show that under mild assumptions, the proposed candidate-level Shapley rewards are unbiased estimators of the set-level reward and provide meaningful gradients that guide training toward higher-performing candidates. We further prove that in the LLM-as-a-recommender setting, the Shapley values can be computed in polynomial time, ensuring that faster convergence in terms of the number of steps does not come at the cost of increased per-step training time.

Improved performance and a new recommendation benchmark. We evaluate ShapE-GRPO on multiple LLM generation tasks, including summarization and code generation, and show that it consistently improves set-level performance compared with the standard GRPO baseline. Furthermore, we release a curated Netflix movie recommendation dataset tailored to the LLM-as-a-recommender scenario, enabling future research on multi-candidate recommendation and set-level optimization for language models.

1.2 Related Works

We summarize below three lines of prior work that are most relevant to our study.

RLHF and LLM Alignment Many works use reinforcement learning from human feedback (RLHF) to align large language models with human preferences (Christiano et al., 2017; Stiennon et al., 2020; Ouyang et al., 2022; Bai et al., 2022; Rafailov et al., 2023; Yuan et al., 2023; Gao et al., 2023; Casper et al., 2023; Shao et al., 2024; Ethayarajh et al., 2024; Liu et al.; Yu et al., 2025; Zheng et al., 2025), but our work differs in studying multi-candidate generation where utility depends on a set of candidates rather than a single response, and proposing a principled candidate-level reward allocation framework.

Fine-Grained Reward Modeling for LLM Training Our work is related to fine-grained reward modeling for LLM training (Chan et al., 2024; Zeng et al., 2024; Yoon et al., 2024; Xia et al., 2024; Zhong et al., 2024; Li et al., 2025; Zhou et al., 2025; Zhang et al., 2025b; Chen et al., 2025; Zhu et al., 2025), but these methods rely on heuristic token-level decomposition and are inherently order-sensitive.

Shapley Value in Machine Learning Moreover, our work connects to Shapley value in machine learning (Ghorbani & Zou, 2019; Sim et al., 2020; Schoch et al., 2022; Liu et al., 2023; Schoch et al., 2023; Wang et al., 2024a;b; Horovicz & Goldshmidt, 2024; Cao et al., 2025; Xiao et al., 2025; Tamine et al., 2025), and differs by formulating reward allocation over unordered candidate sets and preserving computational efficiency.

Due to space limitations, we defer further discussion to Appendix A.1.

2 Preliminaries

In this section, we briefly review the GRPO objective and the definition of Shapley value from cooperative game theory.

GRPO Algorithm. GRPO is a reinforcement learning objective used in LLM post-training that optimizes a policy using group-relative advantages computed from multiple sampled responses. For each prompt q, GRPO samples a group of outputs $\{o_1, o_2, \ldots, o_G\}$ from the old policy $\pi_{\theta_{\text{old}}}$ and updates the current policy π_θ by maximizing the following objective with an oracle reward function $R(\cdot)$:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{q\sim P(Q),\,\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(O|q)}\Bigg[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\Big(\min\Big(\frac{\pi_\theta(o_{i,t}\mid q,o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}\mid q,o_{i,<t})}A_{i,t},\\ \text{clip}\Big(\frac{\pi_\theta(o_{i,t}\mid q,o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}\mid q,o_{i,<t})},1-\epsilon,1+\epsilon\Big)A_{i,t}\Big)-\beta D_{\text{KL}}[\pi_\theta\|\pi_{\text{ref}}]\Big)\Bigg].$$

Here $A_{i,t} = \frac{R(o_i)-\text{mean}(R)}{\text{std}(R)}$ denotes the advantage shared by every token in the response, ϵ is the PPO-style clipping parameter, and β controls the KL penalty with respect to the reference policy π_{ref}. Following prevalent GRPO implementations (Sheng et al., 2024; nem, 2025), we employ an augmented standard deviation, clamping $\text{std}(R)$ to 1 when all rewards are identical, to prevent numerical instability.

Shapley Value. In cooperative game theory, the Shapley value provides a principled method for allocating the value generated by a coalition of players among individual participants (candidates in our setting). Suppose we are given an oracle set-level reward function $R(\cdot)$ mapping any subset of candidates to a non-negative reward. Let the candidate set be $\mathcal{C} = \{c^1, c^2, \ldots, c^K\}$. The Shapley value assigns a credit ϕ^i to each candidate c^i based on its expected marginal contribution to all

possible coalitions. Formally, the Shapley value of candidate c^i is defined as

$$\phi^i = \sum_{S \subseteq \mathcal{C} \setminus \{c^i\}} \frac{|S|!(K - |S| - 1)!}{K!} \Big(R(S \cup \{c^i\}) - R(S) \Big)^1. \tag{1}$$

Intuitively, the Shapley value measures the expected marginal contribution of candidate c^i when candidates join the coalition in a random order.

3 METHODOLOGY

In this section, we formulate the LLM-as-a-recommender problem as a cooperative game and demonstrate how ShapE-GRPO allocates rewards to individual candidates to provide a correct training signal.

3.1 PROBLEM FORMULATION

For a single interaction between an LLM and a user, we decompose the LLM output o as $o = w + \{c^1, \ldots, c^K\}$[2], where c^i denotes the i-th candidate recommendation extracted from the response and K is the total number of candidates. For example, in the travel planning scenario shown in Figure 1, the candidates correspond to recommended travel destinations. The component w represents the general reasoning or explanation process accompanying the recommendations.

In LLM-as-a-recommender settings, the user reward is typically determined by the best candidate among the generated options (Zong et al., 2016; Chen et al., 2016; Ie et al., 2019; Wang et al., 2024c). Therefore, under the GRPO framework, the reward of the response is given by

$$R(o) = R(\{c^1, \ldots, c^K\}) = \max_i R(c^i),$$

where $R(c^i)$ denotes the utility of the i-th candidate, such as whether the user clicks the recommendation or adopts the proposed solution. While our framework supports *arbitrary reward functions*, we choose $\max$ for its notational simplicity and its alignment with the LLM-as-a-recommender paradigm.

However, this shared reward assignment leads to sparse and misaligned training signals. In particular, when at least one candidate is correct, all candidates within the response may be implicitly treated as successful. As a result, incorrect candidates can *free-ride* on the success of correct ones, receiving positive reinforcement despite not contributing to the final outcome. This lack of proper incentives can significantly reduce training efficiency.

3.2 SHAPE-GRPO ALGORITHM

We view all candidates in a response as players in a cooperative game. Given the candidate set decomposition $\{c^1, \ldots, c^K\}$ and the reward function $R(\cdot)$, we compute the scaled Shapley value of each candidate according to Equation (1), which measures its marginal contribution to the final reward. For the reasoning component w, we retain the original reward assignment and allocate the sequence-level reward $R(o)$ to all tokens in w. In contrast, the candidates receive individualized rewards derived from their Shapley values.

This design provides several important advantages. First, the Shapley value distinguishes the marginal contribution of different candidates to the overall reward, providing a correct and fine-grained post-training signal. Second, compared with the token-level Shapley allocation, which depends on the semantic order of tokens, user rewards in recommender scenarios do not depend on the ordering of recommendations. Therefore, candidate-level Shapley values preserve all desirable economic properties of the Shapley value. Finally, although a response may contain thousands of tokens, it usually includes only a small number of candidates. Computing Shapley values at the

[1] Following the standard convention, we define $R(\emptyset) = 0$.

[2] For simplicity, we assume each input generates exactly K candidates. In cases where K varies across inputs, it can be replaced by the respective total number of candidates. Throughout this paper, we use subscripts to denote output indices and superscripts to denote candidate indices.

candidate level therefore introduces only negligible additional computational overhead, which we analyze formally in Section 4.

Therefore, we introduce our *ShapE-GRPO* algorithm. With the Shapley values computed for each candidate, we rescale them by the number of candidates K to maintain a comparable reward magnitude. Note that the original Shapley value decomposes the sequence-level reward $R(o)$ among candidates. In contrast, ShapE-GRPO redistributes the advantage signal used in policy optimization. Specifically, let ϕ_i^j denote the Shapley value of candidate c_i^j in the i-th output. We define the ShapE-GRPO advantage for the t-th token as

$$A_{i,t}^{\text{ShapE}} = \frac{R^{\text{ShapE}}(o_{i,t}) - \text{mean}(R)}{\text{std}(R)},$$

where we adopt original $\text{mean}(R)$ and $\text{std}(R)$ implementations in GRPO to avoid extra compute overhead. We define the token-level Shapley-enhanced reward, distinguishing whether a token belongs to a candidate or to the reasoning part, as

$$R^{\text{ShapE}}(o_{i,t}) = \begin{cases} R(o_i), & \text{if } o_{i,t} \in w_i, \\ K\phi_i^j, & \text{if } o_{i,t} \in c_i^j. \end{cases}$$

Using these advantages, the ShapE-GRPO objective becomes

$$\mathcal{J}_{\text{ShapE-GRPO}}(\theta) = \mathbb{E}_{q\sim P(Q),\, \{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(O|q)} \Bigg[\frac{1}{G}\sum_{i=1}^{G} \frac{1}{|o_i|}\sum_{t=1}^{|o_i|} \Bigg(\min\Bigg(\frac{\pi_\theta(o_{i,t} \mid q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t} \mid q, o_{i,<t})} A_{i,t}^{\text{ShapE}}, \\ \text{clip}\Big(\frac{\pi_\theta(o_{i,t} \mid q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t} \mid q, o_{i,<t})}, 1-\epsilon, 1+\epsilon \Big) A_{i,t}^{\text{ShapE}} \Bigg) - \beta D_{\text{KL}}[\pi_\theta \| \pi_{\text{ref}}] \Bigg) \Bigg].$$

By allocating rewards according to each candidate's scaled marginal contribution, ShapE-GRPO eliminates the free-rider problem present in shared reward schemes and provides correct training signals for the LLM-as-a-recommender setting.

4 Theoretical Analysis

In this section, we provide theoretical insights into the proposed ShapE-GRPO method. First, we show that when the advantage serves as the directional signal for policy updates, ShapE-GRPO can be interpreted as a reweighting of the GRPO training signal. This reweighting preserves the overall optimization structure while eliminating the free-rider problem, ensuring that incorrect candidates cannot receive positive reward incentives. Second, we analyze the computational complexity of the proposed Shapley-enhanced advantage. We show that computing Shapley values at the candidate level introduces only negligible additional overhead. This result establishes the efficiency of the candidate-level Shapley reward allocation scheme.

We now consider an output o_i. Unlike different outputs, which may have large variations in length, the number of tokens associated with different candidates is typically similar in practice. For example, in travel planning tasks, candidates correspond to location names, while in movie recommender systems, candidates may be represented by fixed-length movie identifiers. Therefore, the token counts of different candidates are approximately balanced. We present in Figure 2 the distribution of the coefficient of variation (CV) of candidate lengths using Qwen3-8B (Yang et al., 2025), namely token numbers, for the textual summarization task used in our experiments. The variation is significantly smaller than the response length variation reported in prior work (Perez-Ramirez et al., 2025). Based on this observation, we present the following proposition showing that ShapE-GRPO does not introduce additional bias compared with the original GRPO objective.

Proposition 4.1. *Assuming that the candidates $\{c_i^1, \ldots, c_i^K\}$ in the output o_i have equal lengths, the ShapE-GRPO advantage is a reweighting of the GRPO advantage. Specifically,*

$$\sum_{t=1}^{|o_i|} A_{i,t} = \sum_{t=1}^{|o_i|} A_{i,t}^{\textit{ShapE}}.$$

Moreover, ShapE-GRPO ensures that each candidate receives an appropriate reward allocation according to its marginal contribution, thereby eliminating the free-rider phenomenon. In particular, incorrect candidates cannot obtain positive reward signals due to the success of other candidates. This prevents false-positive reinforcement in the RL training process and can lead to faster convergence.

Proposition 4.2. *Consider the candidates $\{c_i^1, \ldots, c_i^K\}$ in the output o_i and a general non-negative reward function $R(\cdot)$. Then, for any incorrect candidate c_i^j with $R(c_i^j) = 0$, it holds that the ShapE-GRPO advantage satisfies*

$$A_{i,t}^{\textit{ShapE}} \leq 0 \textit{ for any token } o_{i,t} \in c_i^j.$$

We note that our prompt engineering explicitly instructs the LLM to generate K distinct candidates in each response. This design provides denser reward signals for training. At the same time, it partially mitigates an issue observed in standard GRPO training, where repeated sampling may produce identical responses, leading to degenerate reward variance and vanishing advantage signals (Zhang et al., 2025c). By encouraging diversity among candidates within a single response, our framework improves the efficiency of reward feedback and stabilizes policy updates. Extending this idea to the general GRPO setting, even when only a single candidate is required to be generated per response, may constitute a promising direction for future research.

Finally, we show that our algorithm incurs only negligible additional overhead compared with GRPO. Specifically, due to the special structure of the LLM-as-a-recommender setting, the exact Shapley values can be computed with only $\mathcal{O}(K^2)$ additional floating-point operations, rather than the exponential $\mathcal{O}(2^K)$ complexity in the general case.

Theorem 4.1. *Consider the candidates $\{c_i^1, \ldots, c_i^K\}$ in the output o_i, sorted in descending order of their rewards, i.e.,*

$$R(c_i^{(1)}) \geq \cdots \geq R(c_i^{(K)}).$$

Then for any $o_{i,t} \in c_i^{(j)}$, its Shapley-enhanced reward is given by

$$R^{\textit{ShapE}}(o_{i,t}) = \sum_{k=j}^{K} \frac{K(R(c_i^{(k)}) - R(c_i^{(k+1)}))}{k}, \tag{2}$$

where we define $R(c_i^{(K+1)}) = 0$.

In particular, in the binary reward case where $R(\cdot) \in \{0, 1\}$, suppose there are $m \geq 1$ correct candidates. Then, the tokens within these candidates receive a Shapley-enhanced reward $\frac{K}{m}$, while all incorrect candidates receive zero reward.

This immediately leads to the following corollary, which shows that ShapE-GRPO introduces only polynomial additional computational overhead.

Corollary 4.1. *For a general reward function $R(\cdot)$, computing the ShapE reward $R^{\textit{ShapE}}(\cdot)$ for each output requires only an additional $\mathcal{O}(K^2)$ floating-point operations. In the binary reward case, the complexity further reduces to $\mathcal{O}(K)$.*

In summary, our setting allows the Shapley values to be computed efficiently and exactly. Unlike many machine learning applications, where Shapley values must be approximated, our algorithm admits an exact computation in $\mathcal{O}(K^2)$ time. Moreover, in contrast to token-wise Shapley allocation, our method only computes Shapley values over the K candidates. In typical LLM scenarios, the number of candidates is significantly smaller than the number of tokens in a response. By fixing the reward of the reasoning component to $R(o)$, we avoid introducing reward bias while also eliminating the dependence on token order. This yields an appropriately grained reward allocation scheme for the LLM-as-a-recommender setting. Finally, we note that the Shapley-enhanced reward only requires floating-point computations. Since gradients of the advantage are not needed during training, the method avoids additional gradient computations entirely. These three properties ensure that ShapE-GRPO introduces negligible computational overhead. At the same time, the denser reward allocation improves the training signal, which empirically accelerates convergence toward the targeted policy and often leads to faster training.

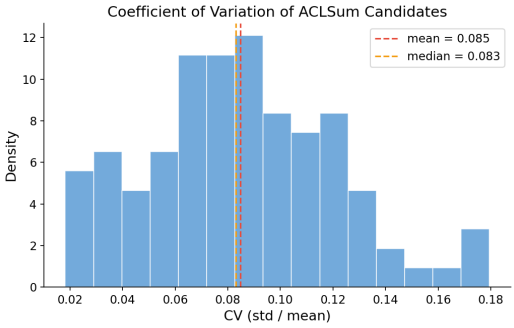


Figure 2: Distribution of candidate length CV for summarization.

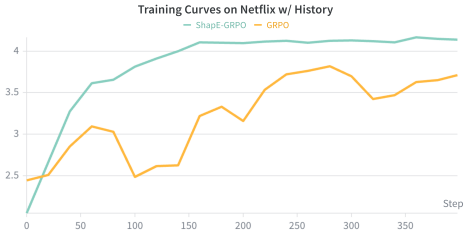


Figure 3: Training curves on the Netflix dataset with user history. ShapE-GRPO converges faster and exhibits significantly more stable training dynamics compared with GRPO.

5 Experiments

In this section, we evaluate ShapE-GRPO on multiple tasks, including summarization, code suggestion, and movie recommendation. Across all evaluated datasets, ShapE-GRPO consistently outperforms the standard GRPO baseline, demonstrating the effectiveness of the proposed reward allocation scheme.

5.1 Experimental Setup

We evaluate ShapE-GRPO on three representative tasks. We train the Qwen3-8B model (Yang et al., 2025) for all experiments, using two GH200 GPUs. Additional implementation details, e.g., prompts and hyperparameter settings, as well as results with Llama3.1-8B-Instruct (Grattafiori et al., 2024), are provided in Appendix C.

Summarization. We use the ACLSum dataset (Takeshita et al., 2024), a high-quality benchmark for scientific paper summarization. ACLSum contains scientific publications from major NLP venues and provides expert-annotated summaries that capture multiple aspects of each paper, including the challenge, approach, and outcome of the work. The dataset is manually curated and validated by domain experts to ensure high-quality reference summaries. For each input document, the model generates multiple candidate summaries. We adopt an LLM-as-a-judge evaluation protocol (Zheng et al., 2023) and use GPT-5.2 (Singh et al., 2025) to score the generated summaries based on their quality relative to the reference summaries. The obtained scores are then normalized to the continuous range $[0, 1]$ and used as the reward signal during training.

Code Suggestion. We use the DS-1000 dataset, a widely used benchmark for data science code generation (Lai et al., 2023). DS-1000 contains programming problems covering practical data science tasks across several popular Python libraries, including `NumPy`, `Pandas`, `SciPy`, `Scikit-learn`, `Matplotlib`, `TensorFlow`, and `PyTorch`. The problems are constructed from real-world StackOverflow questions and reformulated to reflect realistic programming scenarios. To simulate a typical user interaction where a programmer asks an LLM for coding suggestions, the model generates multiple candidate code solutions for each prompt. The user only needs one working solution among the suggestions to solve the problem. Therefore, we adopt a binary reward scheme: the reward is 1 if at least one candidate solution passes the DS-1000 execution tests, and 0 otherwise. This setup naturally matches the LLM-as-a-recommender formulation studied in this paper.

Movie Recommendation. Existing benchmarks for LLM-as-a-recommender are limited. To evaluate our method in a realistic recommendation setting, we construct a dataset based on the Netflix Prize data[3]. We tailor the dataset to simulate a video platform recommendation scenario. For each

[3]Original data can be found at `https://www.kaggle.com/datasets/netflix-inc/netflix-prize-data`

user, we use their ratings for movies released before 2005 as the user history and ask the model to recommend movies released in 2005. This temporal split prevents information leakage between training signals and evaluation targets. To study the impact of user context, we provide two versions of the task: one with user history and one without user history. The model generates multiple candidate movie recommendations for each user, mimicking the behavior of real-world video platforms where multiple suggestions are displayed after a user finishes watching a video. We use the user's ground-truth rating for the recommended movie as the reward signal, following the original five-point rating scale in the Netflix dataset. The objective is therefore to maximize the expected user rating of the recommended movies, which can be interpreted as a proxy for user engagement or click-through rate in real-world recommendation systems.

5.2 Main Results

ShapE-GRPO consistently improves the performance of the pretrained model during post-training across multiple datasets, including summarization, code suggestion, and movie recommendation tasks. The key difference between ShapE-GRPO and GRPO lies in the training signal: ShapE-GRPO replaces the standard advantage $A_{i,t}$ in GRPO with the Shapley-enhanced advantage $A_{i,t}^{\text{ShapE}}$, which allocates rewards at the candidate level. We first show that $A_{i,t}^{\text{ShapE}}$ contains statistically additional information compared with $A_{i,t}$. We perform a Fasano-Franceschini (FF) test (Fasano & Franceschini, 1987) to compare the corresponding signal distributions. The resulting D_{FF} statistics are 0.760, 0.280, 0.585, and 0.701 for ACLSum, DS-1000, Netflix with user history, and Netflix without user history, respectively. All corresponding p-values are below 10^{-5}. We include more details in Appendix C.4.

To study the impact of candidate-level reward allocation, we additionally compare against Winner-Takes-All (WTA) and Direct-Candidate-Reward (DCR) baselines.

- WTA adopts a candidate-level reward allocation that assigns the reward only to the highest-scoring candidate within each response, with all remaining candidates receiving zero, while keeping the reasoning-part reward and the outer GRPO optimization unchanged. When multiple candidates achieve the same highest score, we split the candidate-level reward evenly among them. This comparison tests whether the best-only reward is already sufficient, or whether coalition-aware reward allocation is needed.
- For DCR, the reward assigned to each candidate is simply $R(c^i)$, rather than the Shapley-enhanced reweighted reward. This allows us to test whether ShapE-GRPO's candidate-level coalition-aware reward allocation drives further performance improvement in the fine-grained setting.

We summarize the main results in Table 1. Across all datasets, ShapE-GRPO consistently improves over the standard GRPO baseline, the Winner-Takes-All baseline and the Direct-Candidate-Reward baseline. This shows that the gains cannot be explained by heuristics that only reward the single best candidate within each response or naively allocate rewards at the candidate level. Instead, the results support the value of coalition-aware candidate reward allocation under the set-level objective.

On ACLSum, ShapE-GRPO improves over GRPO, Winner-Takes-All and Direct-Candidate-Reward, indicating that the best-only reward is too myopic even for continuous summary-quality rewards, while directly assigning rewards at the candidate level fails to provide a sufficiently informative learning signal. On DS-1000, ShapE-GRPO again achieves the strongest performance. The gains are even larger in the recommendation setting: on Netflix with user history, ShapE-GRPO outperforms all baselines by a wide margin, and on Netflix without user history it still yields the best average rating. Notably, the Qwen3-8B model struggles to effectively utilize user history. After ShapE-GRPO training, however, the model learns to better incorporate historical preferences, resulting in a much larger performance gain when user history is available.

We further compare the training dynamics of ShapE-GRPO and GRPO on the Netflix dataset with user history. This is the most challenging dataset in our experiments: the average prompt length is around fifteen thousand tokens, and it most closely reflects the intended LLM-as-a-recommender scenario.

Dataset	QWEN3-8B	GRPO	WTA	DCR	ShapE-GRPO
ACLSum (↑)	0.654	0.658	0.642	0.662	**0.672**
DS-1000 (↑)	0.398[4]	0.465	0.418	0.505	**0.510**
Netflix w/ history (↑)	2.872[5]	3.710	3.550	3.721	**4.141**
Netflix w/o history (↑)	3.395	3.627	2.497	3.668	**3.821**

Table 1: Performance comparison across four datasets using QWEN3-8B. Winner-Takes-All assigns candidate-level reward only to the highest-scoring candidate, Direct-Candidate-Reward uses the original reward of each candidate, whereas ShapE-GRPO allocates reward by scaled Shapley values. ShapE-GRPO consistently improves over the standard GRPO baseline, the Winner-Takes-All baseline and the Direct-Candidate-Reward baseline.

Figure 3 shows the training score curves of the two methods. ShapE-GRPO reaches saturation after approximately 160 training steps, after which the performance remains nearly flat. In contrast, GRPO requires around 240 steps to reach its final performance level. Moreover, the checkpoints produced during GRPO training exhibit large fluctuations in evaluation performance. Specifically, we evaluate the model every 20 training steps and observe large fluctuations in evaluation-set performance during GRPO training. This behavior reflects the inherently high variance caused by sparse reward signals in GRPO. In comparison, ShapE-GRPO produces a much more stable training trajectory due to the denser and more informative reward allocation.

These consistent improvements across diverse tasks and faster convergence rates suggest that more accurate reward allocation at the candidate level leads to more effective and efficient policy optimization. By allocating rewards according to each candidate's rescaled marginal contribution, ShapE-GRPO provides a more informative training signal than the shared reward scheme used in GRPO, the heuristic best-only reward used in Winner-Takes-All and the original candidate-level reward in Direct-Candidate-Reward.

6 CONCLUSION AND DISCUSSION

In this paper, we propose ShapE-GRPO, a Shapley-enhanced policy optimization framework for the LLM-as-a-recommender setting. We show both theoretically and empirically that ShapE-GRPO consistently improves the performance of the standard GRPO algorithm across different model architectures and tasks. Our key insight is that reward signals in LLM-as-a-recommender problems are naturally generated at the set level. By allocating rewards according to each candidate's marginal contribution, ShapE-GRPO provides a permutation-invariant, effective and efficient reward allocation scheme. The proposed Shapley-enhanced advantage preserves the original motivation of Shapley value while remaining computationally tractable in the candidate-level setting. Together, these results demonstrate that candidate-level reward allocation provides a principled and practical solution for reinforcement learning in LLM-as-a-recommender scenarios.

Questions naturally arise for future exploration. While Shapley value provides a principled way to allocate rewards among candidates, it is only one possible credit allocation scheme. Recent proposals of alternative value concepts, such as instrumental value (Ai et al., 2024), raise the question of how different allocation mechanisms may affect learning dynamics and policy performance. In addition, generating multiple candidates within a single response appears to increase output diversity, which may further improve the efficiency and diversity of GRPO sampling. Another interesting direction concerns the scoring of candidates: when an oracle reward model is unavailable, it remains unclear how unsupervised approaches, e.g., gradient-based (Jung et al., 2025), would perform for candidate evaluation. We leave these interesting questions as promising directions for future work.

[4]We limit the maximum output length to 4096 tokens during training. Since the pretrained model often exceeds this limit, we allow up to 16384 tokens when evaluating the pretrained model. When restricted to 4096 tokens, the pretrained model achieves a score of 0.273.

[5]When restricted to 4096 tokens, the pretrained model achieves a score of 2.434.

AI USE STATEMENT

We used LLMs only for minor writing assistance, such as identifying grammatical errors and improving readability. LLMs were not used for developing research ideas, theoretical results, methodologies, literature review, or interpreting results. All AI-assisted edits were reviewed by the authors, who take full responsibility for the final content of the paper.

REFERENCES

Nemo rl: A scalable and efficient post-training library. https://github.com/NVIDIA-NeMo/RL, 2025. GitHub repository.

Rui Ai, Boxiang Lyu, Zhaoran Wang, Zhuoran Yang, and Haifeng Xu. An instrumental value for data production and its application to data pricing. *arXiv preprint arXiv:2412.18140*, 2024.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

Meng Cao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. Scar: Shapley credit assignment for more efficient rlhf. *arXiv preprint arXiv:2505.20417*, 2025.

Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, Jérémy Scheurer, Javier Rando, Rachel Freedman, Tomasz Korbak, David Lindner, Pedro Freire, et al. Open problems and fundamental limitations of reinforcement learning from human feedback. *arXiv preprint arXiv:2307.15217*, 2023.

Alex J Chan, Hao Sun, Samuel Holt, and Mihaela Van Der Schaar. Dense reward for free in reinforcement learning from human feedback. *arXiv preprint arXiv:2402.00782*, 2024.

Hongzhan Chen, Tao Yang, Shiping Gao, Ruijun Chen, Xiaojun Quan, Hongtao Tian, and Ting Yao. Discriminative policy optimization for token-level reward models. *arXiv preprint arXiv:2505.23363*, 2025.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Wei Chen, Wei Hu, Fu Li, Jian Li, Yu Liu, and Pinyan Lu. Combinatorial multi-armed bandit with general reward functions. *Advances in Neural Information Processing Systems*, 29, 2016.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*, 2024.

Giovanni Fasano and Alberto Franceschini. A multidimensional version of the kolmogorov–smirnov test. *Monthly Notices of the Royal Astronomical Society*, 225(1):155–170, 1987.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, pp. 10835–10866. PMLR, 2023.

Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. In-context autoencoder for context compression in a large language model. *arXiv preprint arXiv:2307.06945*, 2023.

Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *International conference on machine learning*, pp. 2242–2251. PMLR, 2019.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Miriam Horovicz and Roni Goldshmidt. TokenSHAP: Interpreting large language models with Monte Carlo shapley value estimation. In Lotem Peled-Cohen, Nitay Calderon, Shir Lissak, and Roi Reichart (eds.), *Proceedings of the 1st Workshop on NLP for Science (NLP4Science)*, pp. 1–8, Miami, FL, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.nlp4science-1.1. URL https://aclanthology.org/2024.nlp4science-1.1/.

Eugene Ie, Vihan Jain, Jing Wang, Sanmit Narvekar, Ritesh Agarwal, Rui Wu, Heng-Tze Cheng, Morgane Lustman, Vince Gatto, Paul Covington, et al. Reinforcement learning for slate-based recommender systems: A tractable decomposition and practical methodology. *arXiv preprint arXiv:1905.12767*, 2019.

Jaehun Jung, Seungju Han, Ximing Lu, Skyler Hallinan, David Acuna, Shrimai Prabhumoye, Mostafa Patwary, Mohammad Shoeybi, Bryan Catanzaro, and Yejin Choi. Prismatic synthesis: Gradient-based data diversification boosts generalization in llm reasoning. *arXiv preprint arXiv:2505.20161*, 2025.

Yuhang Lai, Chengxi Li, Yiming Wang, Tianyi Zhang, Ruiqi Zhong, Luke Zettlemoyer, Wen-tau Yih, Daniel Fried, Sida Wang, and Tao Yu. Ds-1000: A natural and reliable benchmark for data science code generation. In *International Conference on Machine Learning*, pp. 18319–18345. PMLR, 2023.

Jiahui Li, Lin Li, Tai-Wei Chang, Kun Kuang, Long Chen, Jun Zhou, and Cheng Yang. Red: Unleashing token-level rewards from holistic feedback via reward redistribution. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 4993–5022, 2025.

Jinming Li, Wentao Zhang, Tian Wang, Guanglei Xiong, Alan Lu, and Gerard Medioni. Gpt4rec: A generative framework for personalized recommendation and user interests interpretation. *arXiv preprint arXiv:2304.03879*, 2023.

Gionnieve Lim and Simon T Perrault. Rapid aideation: Generating ideas with the self and in collaboration with large language models. *arXiv preprint arXiv:2403.12928*, 2024.

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

Tianqi Liu, Zhen Qin, Junru Wu, Jiaming Shen, Misha Khalman, Rishabh Joshi, Yao Zhao, Mohammad Saleh, Simon Baumgartner, Jialu Liu, et al. Lipo: Listwise preference optimization through learning-to-rank, 2024a. *URL https://arxiv. org/abs/2402.01878*.

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. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

Daniel F Perez-Ramirez, Dejan Kostic, and Magnus Boman. Castillo: Characterizing response length distributions of large language models. *arXiv preprint arXiv:2505.16881*, 2025.

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. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Stephanie Schoch, Haifeng Xu, and Yangfeng Ji. Cs-shapley: class-wise shapley values for data valuation in classification. *Advances in Neural Information Processing Systems*, 35:34574–34585, 2022.

Stephanie Schoch, Ritwick Mishra, and Yangfeng Ji. Data selection for fine-tuning large language models using transferred shapley values. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, pp. 266–275, 2023.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Lloyd S Shapley et al. A value for n-person games. 1953.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

Rachael Hwee Ling Sim, Yehong Zhang, Mun Choon Chan, and Bryan Kian Hsiang Low. Collaborative machine learning with incentive-aware model rewards. In *International conference on machine learning*, pp. 8927–8936. PMLR, 2020.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

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

Sotaro Takeshita, Tommaso Green, Ines Reinig, Kai Eckert, and Simone Paolo Ponzetto. Aclsum: A new dataset for aspect-based summarization of scientific publications. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 6660–6675, 2024.

Mélissa Tamine, Otmane Sakhi, and Benjamin Heymann. Data valuation for llm fine-tuning: Efficient shapley value approximation via language model arithmetic. *arXiv preprint arXiv:2512.15765*, 2025.

Jiachen T Wang, Zhun Deng, Hiroaki Chiba-Okabe, Boaz Barak, and Weijie J Su. An economic solution to copyright challenges of generative ai. *arXiv preprint arXiv:2404.13964*, 2024a.

Jiachen T Wang, Prateek Mittal, Dawn Song, and Ruoxi Jia. Data shapley in one training run. *arXiv preprint arXiv:2406.11011*, 2024b.

Yiliu Wang, Wei Chen, and Milan Vojnovic. Combinatorial bandits for maximum value reward function under value-index feedback. In *The Twelfth International Conference on Learning Representations*, 2024c.

Han Xia, Songyang Gao, Qiming Ge, Zhiheng Xi, Qi Zhang, and Xuan-Jing Huang. Inverse-q*: Token level reinforcement learning for aligning large language models without preference data. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 8178–8188, 2024.

Yingtai Xiao, Yuqing Zhu, Sirat Samyoun, Wanrong Zhang, Jiachen T. Wang, and Jian Du. TokenShapley: Token level context attribution with shapley value. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 3882–3894, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.200. URL https://aclanthology.org/2025.findings-acl.200/.

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

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

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Hongyi Yuan, Zheng Yuan, Chuanqi Tan, Wei Wang, Songfang Huang, and Fei Huang. Rrhf: Rank responses to align language models with human feedback. *Advances in Neural Information Processing Systems*, 36:10935–10950, 2023.

Yongcheng Zeng, Guoqing Liu, Weiyu Ma, Ning Yang, Haifeng Zhang, and Jun Wang. Token-level direct preference optimization. *arXiv preprint arXiv:2404.11999*, 2024.

Guoxi Zhang and Jiuding Duan. Vickreyfeedback: Cost-efficient data construction for reinforcement learning from human feedback. In *International Conference on Principles and Practice of Multi-Agent Systems*, pp. 351–366. Springer, 2024.

Haopeng Zhang, Philip S Yu, and Jiawei Zhang. A systematic survey of text summarization: From statistical methods to large language models. *ACM Computing Surveys*, 57(11):1–41, 2025a.

Songming Zhang, Xue Zhang, Tong Zhang, Bojie Hu, Yufeng Chen, and Jinan Xu. Alignditil: Token-level language model alignment as adaptive policy distillation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 19791–19807, 2025b.

Xingjian Zhang, Siwei Wen, Wenjun Wu, and Lei Huang. Edge-grpo: Entropy-driven grpo with guided error correction for advantage diversity. *arXiv preprint arXiv:2507.21848*, 2025c.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, et al. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.

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. *Advances in neural information processing systems*, 36:46595–46623, 2023.

Han Zhong, Zikang Shan, Guhao Feng, Wei Xiong, Xinle Cheng, Li Zhao, Di He, Jiang Bian, and Liwei Wang. Dpo meets ppo: Reinforced token optimization for rlhf. *arXiv preprint arXiv:2404.18922*, 2024.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

Wenxuan Zhou, Shujian Zhang, Lingxiao Zhao, and Tao Meng. T-reg: Preference optimization with token-level reward regularization. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27876–27889, 2025.

Mingkang Zhu, Xi Chen, Zhongdao Wang, Bei Yu, Hengshuang Zhao, and Jiaya Jia. Tgdpo: Harnessing token-level reward guidance for enhancing direct preference optimization. *arXiv preprint arXiv:2506.14574*, 2025.

Shi Zong, Hao Ni, Kenny Sung, Nan Rosemary Ke, Zheng Wen, and Branislav Kveton. Cascading bandits for large-scale recommendation problems. *arXiv preprint arXiv:1603.05359*, 2016.

A Omitted details in Section 1

A.1 Detailed Related Works

We summarize below three lines of existing literature pertinent to our work.

RLHF and LLM Alignment Reinforcement learning from human feedback has become the dominant paradigm for aligning large language models with human preferences. Early work demonstrated that reward models trained from human comparisons can guide reinforcement learning policies in complex tasks (Christiano et al., 2017), a framework that was later adapted to large language models in systems such as InstructGPT (Ouyang et al., 2022). Since then, a growing body of work has explored alternative algorithms and formulations for preference optimization and alignment,

including preference-based reinforcement learning methods, direct optimization objectives, and improved training pipelines (Stiennon et al., 2020; Bai et al., 2022; Rafailov et al., 2023; Yuan et al., 2023; Shao et al., 2024; Zhang & Duan, 2024; Liu et al.; Yu et al., 2025; Zheng et al., 2025). Several studies have further investigated the theoretical foundations, stability properties, and limitations of RLHF-based training (Gao et al., 2023; Casper et al., 2023), while others propose scalable alternatives that bypass explicit RL but still optimize preference signals (Ethayarajh et al., 2024). Despite differences in algorithms and optimization strategies, these approaches share a common structural assumption: a single reward is assigned to each generated response. In typical RLHF pipelines, a model produces one or several candidate recommendations for a prompt, and a reward model evaluates the whole response, after which policy optimization algorithms such as PPO or related variants update the model to maximize this single sparse reward. While this formulation has proven effective for aligning single responses, it does not naturally capture scenarios where models generate sets of candidate recommendations and the utility depends on the coverage or diversity of the entire set. Such settings arise naturally in many applications, where users select one response from multiple candidates. In contrast to prior work that focuses on sequence-level rewards, our work explicitly models candidate-level reward signals and studies how to allocate reward among candidates in multi-candidate generation using a Shapley-enhanced framework.

Fine-Grained Reward Modeling for LLM Training Recent work has moved beyond sequence-level supervision and explored token-level reward assignment for aligning large language models. Instead of assigning a single scalar reward to an entire response, these methods attempt to provide dense token-level reward signals so that different tokens within the same completion can receive different optimization pressures. Representative examples include ABC (Chan et al., 2024), which redistributes sequence-level reward to tokens using model attention, TDPO (Zeng et al., 2024), TLCR (Yoon et al., 2024), Inverse-Q* (Xia et al., 2024), RTO (Zhong et al., 2024), RED (Li et al., 2025), T-REG (Zhou et al., 2025), AlignDistil (Zhang et al., 2025b), Q-RM (Chen et al., 2025) and TGDPO (Zhu et al., 2025). Although these methods provide denser supervision than standard sequence-level RLHF, they also introduce several limitations. In particular, token-level reward assignment often requires additional reward models, auxiliary discriminators, reward redistribution modules, or Monte Carlo estimation, which can significantly increase training complexity. Moreover, many approaches rely on heuristic decompositions of a holistic response reward, such as attention redistribution, self-generated token labels, or induced token-level surrogates, rather than a uniquely justified allocation rule.

Shapley Value in Machine Learning Shapley value has been widely adopted in machine learning as a principled tool for contribution attribution, incentive design, and data valuation. A major line of work studies data valuation, beginning with Data Shapley (Ghorbani & Zou, 2019) and later scalable variants such as Wang et al. (2024b), which reduces the need for repeated retraining by estimating attribution during a single run. Closely related ideas have been used for collaborative and economic allocation, including Sim et al. (2020); Schoch et al. (2022); Wang et al. (2024a). More recently, methods based on Shapley value have also appeared in LLM-related settings, including SCAR (Cao et al., 2025), which redistributes a sequence-level reward to tokens or spans for RLHF, as well as Shapley value attribution for prompts, context tokens, and prompt interpretation, such as Liu et al. (2023); Horovicz & Goldshmidt (2024); Xiao et al. (2025). Shapley ideas have also been applied to LLM fine-tuning data selection and valuation, for example, in Schoch et al. (2023) and recent work on efficient Shapley approximation for LLM fine-tuning (Tamine et al., 2025). However, applying Shapley within a single response introduces an inherent mismatch with the cooperative-game formulation: token- or sentence-level attribution is fundamentally order-sensitive, since swapping tokens or sentences typically changes the semantics of the response. As a result, the "players" in these formulations are not permutation-invariant, which weakens the theoretical justification of Shapley allocation. In contrast, our work studies candidate-level Shapley reward allocation over unordered candidate sets, where the reward is naturally invariant to candidate ordering. This allows our formulation to better align with the cooperative-game motivation of Shapley values while remaining computationally efficient.

B Omitted Proofs in Section 4

We begin by reviewing several fundamental properties of the Shapley value, including efficiency, symmetry, additivity, and the null player property. They are the theoretical drivers behind ShapE-GRPO's empirical advantages over our heuristic baselines.

- *Efficiency.* The sum of the Shapley values of all candidates equals the value of the total output, so that all the gain is distributed among the candidates:
$$\sum_{j=1}^{K} \phi^j = R(o).$$
- *Symmetry.* If c^i and c^j are two candidates which are equivalent in the sense that $R(S \cup \{c^i\}) = R(S \cup \{c^j\})$ for every subset S of which contains neither c^i nor c^j, then $\phi^i = \phi^j$.
- *Additivity.* If two cooperative games described by reward functions R_1 and R_2 are combined, then the distributed rewards should correspond to the rewards derived from R_1 and the rewards derived from R_2:
$$\phi^i(R_1 + R_2) = \phi^i(R_1) + \phi^i(R_2).$$
- *Null player.* The Shapley value ϕ^i of a null player (candidate) c^i in a cooperative game is zero. A candidate c^i is null if $R(S \cup \{c^i\}) = R(S)$ for all sets S that do not contain c^i.

Efficiency ensures that ShapE-GRPO forms a valid reward allocation rule matching the overall reward. Symmetry guarantees that the reward allocation is independent of the ordering of candidates. Additivity reflects additive rewards. For instance, multiple user clicks on recommended items yield proportional gains in reward. Finally, the null player property implies that incorrect candidates should receive no reward, preventing free-riding behavior.

B.1 Proof of Proposition 4.1

Since we adopt the same mean(R) and std(R) as in original GRPO, we only need to prove
$$\sum_{t=1}^{|o_i|} R^{\text{ShapE}}(o_{i,t}) = |o_i| R(o_i).$$
We use $|w_i|$ and $|c_i^j|$ to denote the length of the reasoning part and the j-th candidate in the i-th response. Since all candidates have equal lengths, we know that
$$\sum_{t=1}^{|o_i|} R^{\text{ShapE}}(o_{i,t}) = |w_i| R(o_i) + \sum_{j=1}^{K} |c_i^j| K \phi_i^j = |w_i| R(o_i) + |c_i^1| K \sum_{j=1}^{K} \phi_i^j.$$
Due to the efficiency property of the Shapley value, we know that
$$\sum_{j=1}^{K} \phi_i^j = R(o_i).$$
Therefore, it holds that
$$\sum_{t=1}^{|o_i|} R^{\text{ShapE}}(o_{i,t}) = (|w| + K|c_i^1|) R(o_i) = (|w| + \sum_{j=1}^{K} |c_i^j|) R(o_i) = |o_i| R(o_i),$$
which finishes our proof.

This highlights that ShapE-GRPO is particularly well suited to LLM-as-a-recommender tasks, especially when SKUs, catalog IDs, or other fixed-format identifiers index items. Importantly, however, our implementation does not rely on candidates having equal lengths. The algorithm itself, computing scaled Shapley values for each candidate and broadcasting them to the corresponding candidate tokens, operates regardless of length variation. Empirically, ShapE-GRPO consistently outperforms all baselines across multiple datasets and model families, suggesting that its gains are intrinsic to candidate-level reward allocation and remain robust to the moderate length variation encountered in practice.

B.2 Proof of Proposition 4.2

When $|G| > 1$, since $R(\cdot)$ is a non-negative function, we know that

$$\text{mean}(R) \geq 0.$$

Recall that

$$A_{i,t}^{\text{ShapE}} = \frac{R^{\text{ShapE}}(o_{i,t}) - \text{mean}(R)}{\text{std}(R)}.$$

For any token corresponding to an incorrect candidate, i.e., $o_{i,t} \in c_i^j$ with $R(c_i^j) = 0$, we know that

$$R^{\text{ShapE}}(o_{i,t}) = K\phi_i^j.$$

Since for any set S, we have

$$R(S \cup \{c_i^j\}) = \max_{c_i^k \in S \cup \{c_i^j\}} R(c_i^k) = \max_{c_i^k \in S} R(c_i^k) = R(S).$$

Then, the definition of Shapley value immediately yields that

$$R^{\text{ShapE}}(o_{i,t}) = K \sum_S \frac{|S|!(K - |S| - 1)!}{K!} \Big(R(S \cup \{c_i^j\}) - R(S) \Big) = K\phi_i^j = K * 0 = 0.$$

When $|G| = 1$, we adopt the widely-used resolution by setting $\text{mean}(R) = 0$ (Sheng et al., 2024). Therefore, together with the above case, it holds that

$$A_{i,t}^{\text{ShapE}} \leq \frac{R^{\text{ShapE}}(o_{i,t})}{\text{std}(R)} = 0,$$

when $o_{i,t} \in c_i^j$ and $R(c_i^j) = 0$.

B.3 Proof of Theorem 4.1

We sort the rewards of the candidates as

$$R(c_i^{(1)}) \geq R(c_i^{(2)}) \geq \cdots \geq R(c_i^{(K-1)}) \geq R(c_i^{(K)}).$$

Recall that the reward of the whole output is $R(o_i) = \max_j R(c_i^{(j)})$. We use $[j]$ to denote $\{1, 2, \ldots, j\}$ for simplicity and assume $R(c_i^{(K+1)}) = 0$. It then holds that

$$R(c_i^{(j)}) = \sum_{k=j}^{K} R(c_i^{(k)}) - R(c_i^{(k+1)}).$$

We define

$$k_i^\star(S) = \arg \min_{j:\, c_i^{(j)} \in S} j,$$

and

$$U_i^j(S) = \begin{cases} 1, & \text{if there exists } k \in [j] \text{ such that } c_i^{(k)} \in S, \\ 0, & \text{otherwise.} \end{cases}$$

Therefore, it holds that

$$R(S) = R(c_i^{(k_i^\star(S))}) = \sum_{k=1}^{K} (R(c_i^{(k)}) - R(c_i^{(k+1)})) U_i^k(S).$$

Due to the additivity of the Shapley value, we know that

$$\phi_i^{(j)}(R) = \sum_{k=1}^{K} (R(c_i^{(k)}) - R(c_i^{(k+1)})) \phi_i^{(j)}(U_i^k).$$

Here $\phi_i^{(j)}$ is the Shapley value of the candidate with the j-th largest reward.

For $\phi_i^{(j)}(U_i^k)$, it holds that

$$\phi_i^{(j)}(U_i^k) = \begin{cases} 0, & \text{if } j > k, \\ \frac{1}{k}, & \text{otherwise.} \end{cases}$$

The second case holds because

$$\sum_j \phi_i^{(j)}(U_i^k) = 1,$$

and the symmetry property of the Shapley value, that is, any $j \leq k$ plays the same role for U_i^k.

We hence have

$$\phi_i^{(j)} = \sum_{k=j}^{K} \frac{R(c_i^{(k)}) - R(c_i^{(k+1)})}{k}.$$

Using the definition of $R^{\text{ShapE}}(\cdot)$, we know that for $o_{i,t} \in c_i^{(j)}$, it holds that

$$R^{\text{ShapE}}(o_{i,t}) = K\phi_i^{(j)} = \sum_{k=j}^{K} \frac{K(R(c_i^{(k)}) - R(c_i^{(k+1)}))}{k},$$

which ends the proof.

For the binary case, since our reward function is permutation-invariant, all candidates with the same reward will have the same Shapley value. When $m = 0$, it is trivial that all candidates will have Shapley value zero as $R(o_i) = 0$. When $m > 0$, note that $\sum_j \phi_i^j = R(o_i) = 1$. We know that $\phi_i^j = \frac{1}{m}$ when $R(c_i^j) = 1$ due to the symmetry property and zero otherwise due to the null player property. By rescaling by K, it yields that

$$R^{\text{ShapE}}(o_{i,t}) = \begin{cases} \frac{K}{|\{j:\, R(c_i^j)=1\}|}, & \text{if } o_{i,t} \in c_i^j \text{ and } R(c_i^j) = 1, \\ 0, & \text{if } o_{i,t} \in c_i^j \text{ and } R(c_i^j) = 0. \end{cases}$$

We remark that $R(\cdot)$ can be any general reward function, even negative, in our proof for Theorem 4.1, demonstrating that our ShapE-GRPO framework can accommodate a wide range of endogenous rewards.

B.4 Proof of Corollary 4.1

From the proof of Theorem 4.1, we know that we can utilize Equation (2) to calculate $R^{\text{ShapE}}(\cdot)$ for every token. For the sorting part, the extra time complexity is $\mathcal{O}(K \log K)$. For every candidate, the compute overhead using Equation (2) is $\mathcal{O}(K)$. Since we have K candidates, the extra overhead is bounded by $\mathcal{O}(K^2)$. All remaining computations are already part of the standard GRPO procedure.

For the binary case, we only need to derive m in $\mathcal{O}(K)$ extra overhead. Then, allocating $\frac{K}{m}$ to correct candidates needs another $\mathcal{O}(K)$ overhead. Therefore, the total extra floating-point operations can be reduced to $\mathcal{O}(K)$.

Therefore, in the LLM-as-a-recommender setting, incorporating candidate-level dense reward signals incurs only polynomial additional computation, which is negligible in modern reinforcement learning frameworks.

C Omitted Experimental Details

C.1 Dataset Construction

For all datasets, we require the LLM to generate $K = 4$ candidates. As shown in Figure 4, the average response reward exhibits diminishing returns as the number of candidates K increases. Here, we use the first k candidates among all K candidates to simulate the setting where only $k < K$ candidates are generated. To balance performance and computational cost, we finally choose $K = 4$.

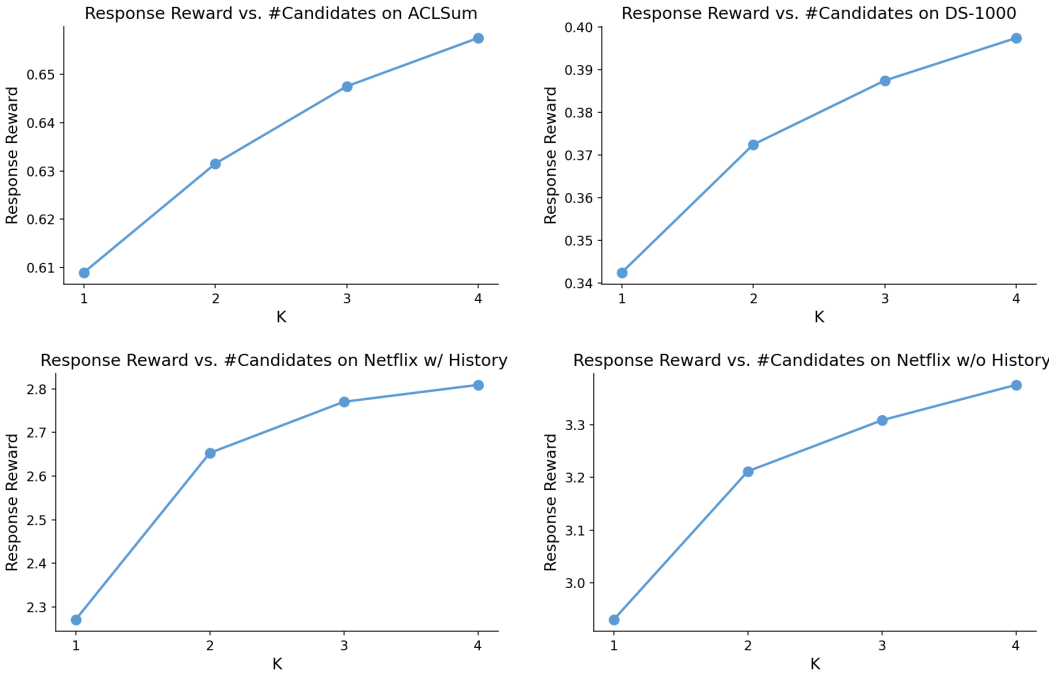


Figure 4: Response reward as a function of the number of candidates across different datasets using QWEN3-8B.

Summarization. For the summarization task, we use the ACLSum dataset. We combine the original training and validation splits to form the training set, resulting in 150 samples. The original test split is used as the evaluation set, which contains 100 samples. For each paper, we ask the LLM to generate four candidate summaries.

For each document, the model is asked to generate a summary. We adopt an LLM-as-a-judge evaluation protocol and use GPT-5.2 to score the generated summaries based on the annotated aspects provided in the dataset, namely challenge, approach, and outcome. For each aspect, the judge assigns a score from 0 to 3 according to whether the generated summary covers the corresponding content and the quality of the description. An additional score of up to 1 is given to reflect the overall quality of the summary. The final reward is normalized to the range $[0, 1]$.

Code Suggestion. For the code suggestion task, we use the DS-1000 benchmark. For each programming problem, we ask the LLM to generate four candidate solutions, mimicking realistic coding assistance scenarios where a problem may admit multiple valid solution strategies (e.g., Bubble Sort or Quick Sort). As long as one candidate solution is correct, the user can pass the test cases and solve the problem. Therefore, we adopt a binary reward setting in this task. We randomly sample 600 problems to form the training set and use the remaining 400 problems as the evaluation set.

For evaluation, we execute the generated code in a fixed Python environment. We report the core package versions needed for our DS-1000 setup: `python=3.10`, `numpy=1.26.4`, `pandas=1.5.3`, `scikit-learn=1.4.0`, `scipy=1.12.0`, `matplotlib=3.8.4`, `seaborn=0.13.2`, `torch=2.2.0`, and `tensorflow=2.16.1`. These packages cover the main dependencies used in our evaluation pipeline while avoiding unnecessary environment complexity.

Movie Recommendation. For the Netflix recommendation task, we construct a dataset based on the Netflix Prize data. We set the number of candidates to $K = 4$. We first randomly sample 1000 user data points to form the training pool. For each user, we use their ratings for movies released before 2005 as the user history, and treat movies released in 2005 as the candidate set. In total,

there are 512 candidate movies released in 2005. If a user has not watched any movie released in 2005, we remove that user from the dataset. To control the prompt length, we retain at most 800 historical movie ratings for each user. For the dataset without user history, we simply remove the history information. After filtering, the final training set contains 796 data points. We construct the evaluation set using the same procedure, resulting in 775 evaluation instances.

For the reward signal, if the movie recommended by the LLM has been watched by the user and a rating is available, we use the user's rating as the reward. If the user has not watched the recommended movie, we interpret this as the user not accepting the recommendation and assign a reward of 0.

C.2 Prompt Templates

In this section, we list the prompt templates used for the four datasets and for the LLM-as-a-judge evaluation protocol. For a fair comparison, all algorithms use the same prompt templates. Since LLMs have strong instruction-following abilities (Zhou et al., 2023), we can simply extract candidates from structured outputs.

ACLSum Summarization Prompt.

```
You are an expert scientific paper summarizer.

You are given the full text of a scientific paper. Your goal is to
produce 4 DISTINCT concise summaries of the paper, each covering
the challenge, approach, and outcome.

Paper:
{document}

Your job:
Provide EXACTLY 4 DISTINCT summaries that accurately capture the
key information in the paper (challenge, approach, and outcome).

CRITICAL RULES (MUST FOLLOW):
- Each summary MUST be wrapped in <summary> </summary> tags.
- Each summary should be concise (2-4 sentences), covering the
challenge addressed, the approach proposed, and the
outcome/results.
- Do NOT output any extra text besides the required output format.
- We will use the best summary among the 4 as your final score.
- OUTPUT FORMAT (STRICT):
    "Reasoning: " followed by a short paragraph (one or two
    sentences). \\n
    "Summaries:" \\n
    1. <summary> Your first summary </summary>\\n
    2. <summary> Your second summary </summary>\\n
    3. <summary> Your third summary </summary>\\n
    4. <summary> Your fourth summary </summary>\\n

EXAMPLE OUTPUT:
Reasoning: I provide four distinct summaries highlighting
different aspects of the paper about dependency-based answer
ranking for QA.
Summaries:
1. <summary> This paper addresses the challenge of ranking answer
candidates for QA when NER produces large candidate sets. The
authors propose a statistical method using Dynamic Time Warping on
dependency relation paths. Their approach outperforms
state-of-the-art methods by up to 20%. </summary>
```

```
2. <summary> The authors tackle limitations of surface-level
ranking in QA systems by introducing a novel dependency path
correlation method. Experiments show significant improvements,
especially on harder questions. </summary>
3. <summary> To improve answer ranking in QA, this work uses
Dynamic Time Warping to compute correlations between dependency
relation paths. The method achieves up to 20% improvement over
existing syntactic approaches. </summary>
4. <summary> This paper proposes a statistical ranking approach
for QA that leverages dependency structures via DTW. Results
demonstrate strong performance gains, particularly when NER-based
candidate sets are large. </summary>
```

DS-1000 Code Suggestion Prompt.

```
You are a Python data processing expert.

You are given a programming task. The problem statement includes
any starter code and imports.
Your goal is to produce 4 DISTINCT correct solutions.

Problem:
{problem_description}

Your job:
Provide EXACTLY 4 DISTINCT valid Python programs that correctly
solve the problem.

CRITICAL RULES (MUST FOLLOW):
- The 4 solutions MUST be meaningfully different in implementation
strategy (not just renaming variables).
- Do NOT output any extra text besides the required output format.
- OUTPUT FORMAT (STRICT):
    "Reasoning: " followed by a short paragraph (one or two
    sentences). \n
    "Solutions:" \n
    1. <code> python_program </code>\n
    2. <code> python_program </code>\n
    3. <code> python_program </code>\n
    4. <code> python_program </code>\n
- Each python_program must be syntactically valid Python code and
executable via exec() without raising a SyntaxError.

EXAMPLE OUTPUT:
Reasoning: I provide four distinct correct approaches while
ensuring each program defines result and follows the required
formatting.
Solutions:
1. <code> import numpy as np\ndf['#1'] = np.roll(df['#1'],
shift=1) </code>
2. <code> Another solution  </code>
3. <code> Third Solution </code>
4. <code> Fourth Solution </code>
```

Netflix w/ history Movie Recommendation Prompt.

```
You are a movie recommendation system.

You are given:
(A) Historical ratings for movies released BEFORE 2005.
```

```
(B) Candidate movies released IN OR AFTER 2005.

User {uid} historical ratings:
{hist_lines}

Candidate movies:
{cand_lines}

Your job:
Pick EXACTLY 4 DISTINCT movies from the candidate list that the
user will like most.

CRITICAL RULES (MUST FOLLOW):
- ONLY choose from the candidate list.
- DO NOT copy or repeat the candidate list.
- Output EXACTLY 6 lines and NOTHING ELSE:
  Line 1: "Reasoning: " followed by a short paragraph (one or two
  sentences).
  Line 2: "Recommendations:"
  Lines 3-6: numbered list in this exact format:
    i. <movie_id> | <movie_title> ({year})
- Use each movie_id at most once.

EXAMPLE OUTPUT:
Reasoning: The user tends to rate dramas and thrillers highly, so
I prioritize critically acclaimed films in those genres.
Recommendations:
1. 123 | Example Movie (2006)
2. 456 | Another Movie (2007)
3. 789 | Third Movie (2005)
4. 1011 | Fourth Movie (2008)
```

Netflix w/o history Movie Recommendation Prompt.

```
You are a movie recommendation system.

You are given:
(A) Candidate movies released IN OR AFTER 2005.

Candidate movies:
{cand_lines}

Your job:
Pick EXACTLY 4 DISTINCT movies from the candidate list that the
user will like most.

CRITICAL RULES (MUST FOLLOW):
- ONLY choose from the candidate list.
- DO NOT copy or repeat the candidate list.
- Output EXACTLY 6 lines and NOTHING ELSE:
Line 1: "Reasoning: " followed by a short paragraph (one or two
sentences).
Line 2: "Recommendations:"
Lines 3-6: numbered list in this exact format:
    i. <movie_id> | <movie_title> ({year})
- Use each movie_id at most once.

EXAMPLE OUTPUT:
Reasoning: The user tends to rate dramas and thrillers highly, so
I prioritize critically acclaimed films in those genres.
```

```
Recommendations:
1. 123 | Example Movie (2006)
2. 456 | Another Movie (2007)
3. 789 | Third Movie (2005)
4. 1011 | Fourth Movie (2008)
```

LLM-as-a-Judge Prompt.

```
You are an EXTREMELY STRICT scientific paper summary evaluator.
You must judge whether a generated summary faithfully and
precisely covers the three key aspects (challenge, approach,
outcome) described in the reference ground truth.

IMPORTANT | Be harsh and precise:
- Only give credit when the summary EXPLICITLY and ACCURATELY
reflects the specific content in the ground truth.
- Vague, generic, or paraphrased statements that do not convey the
SAME specific meaning as the ground truth should receive LOW
scores.
- Do NOT give partial credit for tangentially related statements.
- If the summary fabricates details not present in the ground
truth, penalize accordingly.

You will be given:
- A **ground truth** with three components: challenge, approach,
and outcome.
- A **generated summary** to evaluate.

Scoring rubric (be strict | most summaries should NOT get full
marks):

1. **Challenge coverage** (0-3): Does the summary precisely
capture the challenge / problem from the ground truth?
   - 3: Precisely and completely captures the challenge with
   correct specific details (e.g., exact problem, dataset, or
   limitation mentioned in GT). Reserve this score for
   near-perfect matches.
   - 2: Captures the core idea of the challenge but misses
   important specifics or nuances from the GT.
   - 1: Only vaguely or partially related to the challenge; uses
   generic language that could apply to many papers.
   - 0: Does not mention the challenge, or the stated challenge is
   incorrect / irrelevant.

2. **Approach coverage** (0-3): Does the summary precisely capture
the approach / method from the ground truth?
   - 3: Precisely and completely describes the approach with
   correct specific details (e.g., exact method name, technique,
   or key innovation from GT). Reserve this score for near-perfect
   matches.
   - 2: Captures the general approach but misses key technical
   specifics or innovations mentioned in GT.
   - 1: Only vaguely or partially related; describes a generic
   method without the GT's specific details.
   - 0: Does not mention the approach, or the stated approach is
   incorrect / irrelevant.

3. **Outcome coverage** (0-3): Does the summary precisely capture
the outcome / results from the ground truth?
```

```
   - 3: Precisely and completely captures the outcome with correct
   specific details (e.g., exact metrics, improvements, or
   findings from GT). Reserve this score for near-perfect matches.
   - 2: Captures the general outcome but misses specific numbers,
   comparisons, or key findings from GT.
   - 1: Only vaguely mentions results; uses generic language like
   "achieves good performance" without GT specifics.
   - 0: Does not mention the outcome, or the stated outcome is
   incorrect / irrelevant.

4. **Overall quality** (0-1): Is the summary well-written,
coherent, and concise?
   - 1: Well-organized, fluent, and appropriately concise. No
   hallucinated content.
   - 0.5: Acceptable but has minor issues (verbose, awkward
   phrasing, or minor inaccuracies).
   - 0: Poorly written, incoherent, excessively verbose/short, or
   contains hallucinated content.

Total score = challenge_score + approach_score + outcome_score +
overall_score  (range: 0-10)

CALIBRATION GUIDELINE:
- A perfect summary that matches all GT details precisely: 9-10
- A good summary that captures main ideas but misses some
specifics: 5-7
- A mediocre summary with only vague coverage: 2-4
- A poor or irrelevant summary: 0-2

Return ONLY a JSON object in this exact format (no other text):
{"challenge_score": <float 0-3>, "approach_score": <float 0-3>,
"outcome_score": <float 0-3>, "overall_score": <float 0-1>,
"score": <float 0-10>, "reason": "<brief explanation>"}
```

C.3 Training Hyperparameters

We implement both ShapE-GRPO and GRPO using the `verl` training framework (Sheng et al., 2024). For each prompt, we perform one rollout that generates four candidates. When training the Qwen3-8B model, we observe that the model frequently generates long reasoning traces within the `<think>` and `</think>` tags, which can cause the response length to exceed the predefined limit. To control the length of the reasoning process, we introduce a length penalty on the thinking segment. We set the target thinking length to 2,048 tokens. If the actual thinking length is $l > 2048$, we apply a penalty proportional to the overflow $(l - 2048)/2048$. For ShapE-GRPO, this penalty is applied at the token level, assigning a penalty of $(l - 2048)/2048$ to each token that exceeds the limit. That is

$$\text{length penalty} = \frac{\max(l - 2048, 0)}{2048}.$$

For GRPO, since the reward signal is sparse and assigned only at the sequence level, we subtract the same penalty $(l - 2048)/2048$ directly from the overall reward.

We then report the main training hyperparameters used in our experiments. Table 2 summarizes the most important settings. Extending this to larger-scale experiments is an interesting direction.

To demonstrate the effectiveness of ShapE-GRPO, we intentionally avoid extensive hyperparameter tuning for specific models or datasets, and instead use a unified set of training parameters across all experiments. For each dataset, we set the number of training steps in proportion to the size of the training set. Additional fine-grained tuning could potentially further improve the performance of ShapE-GRPO, which we leave for future work.

Hyperparameter	Value
Training steps	ACLSum:75 DS-1000:300 Netflix:398
Learning rate	1e-6
Max response length	4096
Sampling temperature	0.7
Top-p	0.9
KL coefficient (β)	0.01
Clip range (ϵ)	0.2

Table 2: Main training hyperparameters used in our experiments.

C.4 Additional Results

To characterize the candidate-level signals from ShapE-GRPO relative to the flat signal from vanilla GRPO, we conduct a Fasano–Franceschini test (Fasano & Franceschini, 1987) comparing the two signal distributions on all datasets using pretrained QWEN3-8B. The results are summarized below:

Dataset	D_{FF} statistic	p-value
ACLSum	0.760	$< 10^{-5}$
DS-1000	0.280	$< 10^{-5}$
Netflix w/ history	0.585	$< 10^{-5}$
Netflix w/o history	0.701	$< 10^{-5}$

Table 3: Fasano-Franceschini test results on four datasets with QWEN3-8B. p-values are estimated using 10^5 Monte Carlo permutations. In all cases, no permuted statistic exceeded the observed statistic, so the empirical p-value is below the Monte Carlo resolution of 10^{-5}.

Across all datasets, the ShapE-GRPO signal distribution is statistically significantly distinct from the flat GRPO signal, confirming that the candidate-level Shapley-enhanced allocation produces a genuinely non-flat distribution rather than a trivial reweighting. This distinct distribution corresponds precisely to the additional learning signal that GRPO lacks, and is consistent with the observed empirical gains.

We also find that ShapE-GRPO maintains and even strengthens its advantage over the baselines as the number of rollouts increases, highlighting the benefit of its principled reward allocation. Using ACLSum as an example, we report results with $4\times$ more rollouts per prompt below. GRPO improves from 0.658 to 0.683, WTA from 0.642 to 0.660, DCR from 0.662 to 0.667, while ShapE-GRPO improves more substantially from 0.672 to 0.713. Notably, the performance gap between ShapE-GRPO and the baselines becomes larger with more rollouts. These results further suggest that the observed improvements are robust to a larger rollout budget and are not artifacts of noise under the smaller-budget setting.

To further evaluate the generality of ShapE-GRPO across different large language model architectures, we conduct additional experiments using LLAMA 3.1-8B-INSTRUCT. We compare the performance of ShapE-GRPO and the mentioned baselines on the same tasks across different model architectures. We adopt the same training configuration used in the experiments with QWEN3-8B, including identical hyperparameters, training procedures, and evaluation protocols.

An interesting observation from Table 4 is that the performance on Netflix with user history remains slightly lower than that of the setting without history, even after training with ShapE-GRPO. This suggests that LLAMA 3.1-8B-INSTRUCT may not fully utilize the long contextual information contained in user interaction histories. In recommendation tasks, informative signals are often sparse and distributed across long prompts, which can make it challenging for the model to effectively extract the most relevant information during both training and inference. As a result, the potential

Dataset	LLAMA 3.1-8B-INSTRUCT	GRPO	WTA	DCR	ShapE-GRPO
ACLSum (↑)	0.664	0.707	0.749	0.669	**0.780**
DS-1000 (↑)	0.180	0.205	0.178	0.201	**0.278**
Netflix w/ history (↑)	1.885	3.783	3.325	3.325	**4.027**
Netflix w/o history (↑)	2.130	3.783	3.311	3.199	**4.071**

Table 4: Performance comparison across four datasets using LLAMA 3.1-8B-INSTRUCT. ShapE-GRPO consistently improves over the pretrained model, the standard GRPO baseline and two heuristic baselines.

benefit of additional historical context may not be fully realized. One possible direction to address this limitation is to integrate long-context optimization techniques (Ge et al., 2023) to improve the effectiveness and efficiency of attention over long sequences. Exploring the interaction between such long-context techniques and ShapE-GRPO constitutes an interesting direction for future work.

Overall, Table 4 shows that ShapE-GRPO consistently improves over both the pretrained model and the standard GRPO baseline across all datasets when applied to LLAMA 3.1-8B-INSTRUCT. These results further support that the benefits of candidate-level reward allocation are not tied to a specific model architecture, but instead generalize across different LLM families.