DON'T LET BANDIT FEEDBACK PULL CONTINUAL LLM-RECOMMENDER UPDATES OFF TARGET

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative LLM-based recommenders (LLM-Rec) require continual post-deployment updates, yet deployment logs provide only policy-shaped contextual bandit feedback: outcomes are observed solely for items exposed by a prior serving policy, inducing exposure bias and yielding partial, asymmetric signals consisting of relatively reliable positive responses and ambiguous no-responses. We propose an **Anchored Bandit Policy Optimization (ABPO)** framework for continual LLM-Rec updates that combines group-relative policy optimization (GRPO) with explicit treatment of exposure bias and feedback ambiguity. Specifically, we insert the exposed recommendation as a logged anchor into each GRPO rollout group, so that group-relative normalization is calibrated against the action actually exposed by the prior policy rather than against newly sampled rollouts alone. Because both positive- and no-responses are observed only through prior-policy exposure, we apply self-normalized inverse propensity scoring to the fixed anchor for both feedback types to correct for policy mismatch. At the same time, we treat the two feedback types asymmetrically in reliability: positive responses provide relatively direct endorsement signals, whereas no-responses remain ambiguous because they may reflect either true disinterest or unobserved external factors. To avoid overly aggressive updates from ambiguous no-responses, we temper their penalties with self-certainty, using the model's output-token confidence as a verifier-free reliability signal. Across five domains from Amazon Reviews and MovieLens, our method yields consistent post-update gains in recommendation accuracy while mitigating prior-policy-induced exposure bias more effectively than prior baselines. A real-world online A/B test further shows that our method achieves strong CTR and gradual improvement over successive production updates, demonstrating its practical effectiveness in production [1].

1 INTRODUCTION

Recent LLM-based generative recommender models (LLM-Rec) are rapidly gaining adoption in large-scale recommendation systems, as they can flexibly consume user histories expressed in natural language and generate not only item recommendations but also human-readable rationales (Chen et al., 2024; Park et al., 2026b; 2025). However, in real-world deployment, an LLM-Rec operates within a closed feedback loop—*recommendation* $\rightarrow$ *exposure* $\rightarrow$ *user response* $\rightarrow$ *next recommendation*—so its effective data distribution is no longer fixed and instead co-evolves with user behavior over time. Practical factors such as user interest drift, seasonality, and changes in UI or exposure mechanisms can therefore quickly invalidate a policy that was optimal at deployment. As a result, the LLM-Rec must be continually updated from post-deployment feedback, making continual updating a central challenge for sustaining recommendation quality in production (Yin et al., 2025; Min et al., 2025).

A fundamental challenge in continual updating is that post-deployment feedback lacks ground-truth labels. Instead, it arrives as **policy-shaped contextual bandit feedback**, where observations are induced by the prior serving policy. Because the logs record outcomes only for exposed items, the resulting data are inherently affected by exposure bias (Han et al., 2025; Min et al., 2025). Consequently, naïvely updating the model on such logs can create a self-reinforcing feedback loop, in which the model becomes increasingly biased toward items that were historically favored by the

[1] We released our data and code to support reproducibility.

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.

prior policy. Moreover, each logged sample provides only a partial and unpaired signal—either an implicit positive response or a silent no-response to the exposed recommendation (Zhang et al., 2025). Existing approaches typically address only one side of this difficulty: some update the model using only positive-response samples, which tends to reinforce previously exposed successes (Han et al., 2025), while others treat no-response as an explicit negative signal, which oversimplifies the inherently ambiguous nature of user disengagement (Min et al., 2025; Zhang et al., 2025). As a result, repeated updates on such biased and ambiguous feedback can progressively deteriorate both recommendation accuracy and diversity relative to the no-update model (Fig. 1).

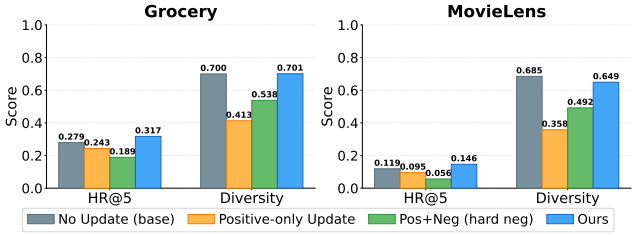

Figure 1: RLVR-based policy updates on contextual-bandit logs degrade both recommendation accuracy and diversity.

To address these challenges, we propose an **Anchored Bandit Policy Optimization (ABPO)** framework for continually updating generative LLM-based recommenders. Our method builds on the Reinforcement Learning with Verified Rewards (RLVR) paradigm and adopts Group Relative Policy Optimization (GRPO), a representative update rule for RLVR-based LLM-Rec training. For each logged context, we insert the exposed recommendation as a fixed anchor into the rollout group, so that group-relative normalization is calibrated against the actual action taken by the prior serving policy. Rather than treating the exposed recommendation as a ground-truth label to imitate, we use it as a *calibration reference* that grounds each update in the policy-shaped feedback actually observed at deployment. The anchor automatically shifts the group reward baseline according to its observed feedback: positive-response anchors receive high item-matching rewards and raise the baseline, whereas no-response anchors receive low rewards and lower it. This suppresses blind self-reinforcement on positive exposures and buffers overly aggressive penalties on no-response exposures, stabilizing group-relative updates under biased and partial bandit feedback.

Within this anchored framework, we correct the logged-anchor contribution for exposure bias using self-normalized inverse propensity scoring (SNIPS). Since both positive- and no-responses are observed only for items exposed by the prior policy, both feedback types inherit off-policy exposure bias; accordingly, we compute SNIPS corrections separately for positive- and no-response logged anchors within each mini-batch. At the same time, the two feedback types differ in reliability. A positive response provides a relatively direct endorsement signal, whereas a no-response is inherently ambiguous and may reflect weak interest, exposure effects, or simple inattention rather than explicit rejection. Thus, treating every no-response as a hard negative can over-penalize otherwise plausible recommendations. We therefore add a separate *self-certainty* reward for no-response cases, using the model's output confidence as an intrinsic reliability signal when external negative evidence is ambiguous. This makes deployment feedback bias-corrected and reliability-aware rather than supervised ground truth, enabling stable continual adaptation without overfitting to historically exposed actions or noisy no-response signals.

We evaluate our method on the Amazon Reviews and MovieLens datasets by constructing an initial offline training set together with a simulated post-deployment contextual bandit feedback stream. Starting from state-of-the-art LLM-Rec backbones, we update the deployed model using the same feedback logs under (1) our method and (2) widely adopted feedback-driven updating baselines from prior work, and then compare post-update performance on subsequent test data against the pre-update model. Across domains, our method achieves strong post-update gains in recommendation accuracy while also improving recommendation diversity (i.e., reducing popularity bias), outperforming existing approaches in most settings. Beyond offline evaluation, ABPO has been deployed in our real-world service, where it generally achieves higher click-through-rate than competing update strategies while showing gradual improvement over successive production updates. It remains actively used in production, demonstrating its practical effectiveness under partial observability and off-policy logging.

2 Background and Motivation

2.1 Contextual Bandit Feedback

In real-world deployment, feedback logs collected over time are inherently off-policy because they are not passive records of user preference, but interaction data generated under the intervention of the recommender policy that was live at the time. Under this mismatch, naïvely treating logged bandit

feedback as the ground-truth labels causes the model to learn not the true user preference but the exposure distribution induced by the prior serving policy. Because feedback is observable only for items the historical policy chose to expose, reinforcing positive-response logs encourages the model to repeatedly regenerate items that were frequently shown and happened to succeed, yielding strong *self-reinforcement* and amplifying popularity bias while suppressing long-tail items. Conversely, interpreting a no-response as an explicit rejection is even more problematic: no-responses can arise from limited attention, UI placement and presentation effects, time constraints, or simple absence of reaction, so treating them as hard negatives introduces abundant false negatives and drives overly aggressive, biased updates that collapse diversity.

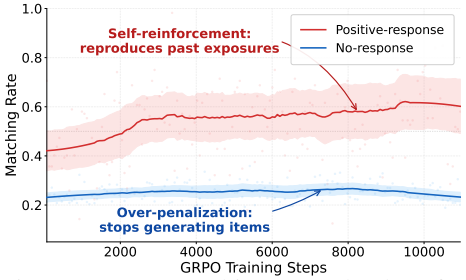

Figure 2: Item-matching reward values for exposed items during GRPO update training.

Figure 2 substantiates these failure modes. When we repeatedly apply Group Relative Policy Optimization (GRPO) across successive post-deployment update rounds, where the model is periodically updated with newly accumulated contextual-bandit feedback, under a reward design that assigns +1 when the model matches a positive-response exposure and -1 when it matches a no-response exposure, the matching rate for positive-response exposures steadily increases, approaching $\sim$60% in later updates. This indicates that the policy progressively reproduces items that the prior policy exposed and that happened to receive positive feedback. In contrast, the matching rate for no-response exposures quickly decays toward a small value, as repeated penalization makes the policy stop generating those items altogether. Together, these trends indicate that contextual-bandit updating can over-amplify past exposed successes while over-penalizing ambiguous disengagement, ultimately degrading generalization and exacerbating popularity bias. These observations motivate the need to correct the bias of contextual-bandit logs by explicitly accounting for the asymmetric reliability of positive- versus no-response feedback.

2.2 LLM-Rec Update Setting

To formalize (1) how we first train the deployed recommender via supervised learning and then (2) continually update it using post-deployment feedback, we begin by defining the recommendation task and the deployment-feedback setting.

Definition 1. User Context and Supervised Next-Item Prediction. For a user, let $\mathbf{x}_t = (x_1, x_2, \ldots, x_t)$ denote the interaction history up to time t, where each x_i represents a previously interacted item together with its associated metadata and chronological context. In the supervised setting, the ground-truth next item is denoted by $x_{t+1} \in \mathcal{V}$, where $\mathcal{V}$ is the item set. The supervised training dataset is defined as $\mathcal{D}_{\text{sup}} = \{(\mathbf{x}_t, x_{t+1})\}$. We obtain an initial deployed policy $\pi_{\theta_{\text{init}}}$ by training a generative recommender on $\mathcal{D}_{\text{sup}}$ using a standard LLM training recipe, consistent with prior LLM-Rec training paradigms used in works such as Rec-R1 (Lin et al., 2025) and TWiCE-Rec (Park et al., 2026a), resulting in the objective $\mathcal{L}_{\text{sup}} = \mathbb{E}_{(\mathbf{x}_t, x_{t+1}) \in \mathcal{D}_{\text{sup}}} \left[\log \pi_{\theta_{\text{init}}}(x_{t+1} \mid \mathbf{x}_t) \right]$.

Definition 2. Post-Deployment Contextual Bandit Feedback. After deployment, $\pi_{\theta_{\text{init}}}$ no longer receives full next-item supervision for every candidate action. For each later context $\mathbf{x}_{t'} = (x_1, \ldots, x_{t'})$ with $t' > t$, a candidate set $\mathcal{C}_{t'} \subseteq \mathcal{V}$ is specified by the deployment or evaluation protocol, and the logging policy samples $a^{\log} \in \mathcal{C}_{t'}$ from $e_0(\cdot \mid \mathbf{x}_{t'})$. Here, $e_0(a \mid \mathbf{x}_{t'})$ abbreviates the candidate-conditional probability $e_0(a \mid \mathbf{x}_{t'}, \mathcal{C}_{t'})$; we omit $\mathcal{C}_{t'}$ for readability. In our offline experiments, $\mathcal{C}_{t'}$ is instantiated as described in Appendix A. The environment then returns partial feedback $y \in \{1, 0\}$ only for the exposed item, where $y = 1$ denotes an observed positive response, such as a click, and $y = 0$ denotes a no-response. Importantly, $y = 0$ is not treated as a deterministic negative label or as a mismatch with a fixed ground-truth next item; it may also result from non-examination, timing, position bias, or other unobserved factors. The resulting post-deployment bandit dataset is defined as $\mathcal{D}_{\text{bandit}} = \{(\mathbf{x}_{t'}, a^{\log}, y, e_0(a^{\log} \mid \mathbf{x}_{t'}))\}$, where $e_0(a^{\log} \mid \mathbf{x}_{t'})$ is the item-level logging propensity, i.e., the probability that the logging policy exposes item $a^{\log}$ under context $\mathbf{x}_{t'}$. In our offline bandit construction, this propensity is known because the logged recommendation is sampled from a controlled stochastic item-level recommendation distribution. Concretely, for each candidate item $a \in \mathcal{C}_{t'}$, we compute $s_0(a, \mathbf{x}_{t'})$, the length-normalized average token log-probability assigned by the LLM to the textual representation of item a, and define $e_0(a \mid \mathbf{x}_{t'}) = \frac{\exp(s_0(a, \mathbf{x}_{t'})/\tau)}{\sum_{b \in \mathcal{C}_{t'}} \exp(s_0(b, \mathbf{x}_{t'})/\tau)}$. The logged recommendation is then sampled as $a^{\log} \sim \text{Categorical}(e_0(\cdot \mid \mathbf{x}_{t'}))$. Thus, the stored

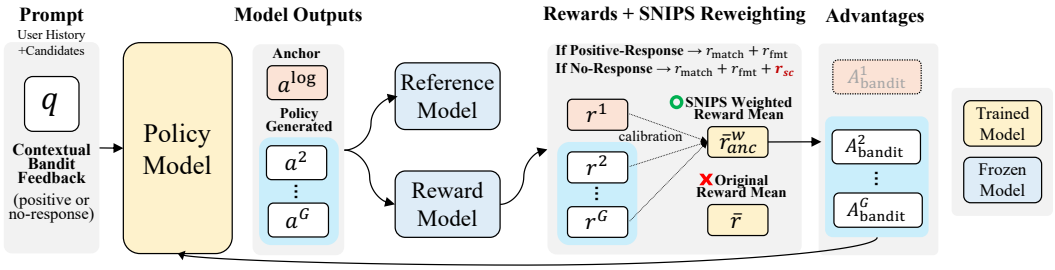

Figure 3: Overview of Anchored Contextual Bandit Reinforcement Learning

propensity is the normalized item-level exposure probability induced by the softmax distribution over candidate items, not the raw token-level likelihood of the generated recommendation text. Unlike $\mathcal{D}_{\text{sup}}$, $\mathcal{D}_{\text{bandit}}$ is policy-shaped: feedback is observed only for the recommendation exposed by the logging policy, while counterfactual outcomes for unexposed items remain unobserved. Therefore, the dataset is inherently partial and biased by the exposure pattern induced by e_0.

Definition 3. Continual Policy Update Problem Setting. Given an initial policy $\pi_{\theta_{\text{init}}}$ trained on $\mathcal{D}_{\text{sup}}$, we collect a post-deployment contextual-bandit dataset $\mathcal{D}_{\text{bandit}}$. Our goal is to train an updated policy π_θ that improves future post-deployment recommendations. Ideally, this corresponds to maximizing $J(\pi_\theta) = \mathbb{E}_{\mathbf{x}\sim p_{\text{deploy}}}\mathbb{E}_{a\sim\pi_\theta(\cdot|\mathbf{x})}[r(\mathbf{x}, a)]$, where p_{deploy} is the context distribution underlying $\mathcal{D}_{\text{bandit}}$, and $r(\mathbf{x}, a)$ is the latent user reward. However, $\mathcal{D}_{\text{bandit}}$ provides feedback only for the logged action $a^{\log}$ exposed by $\pi_{\theta_{\text{init}}}$, making the update problem one of learning from partial, off-policy contextual-bandit evidence rather than supervised ground truth. However, this optimization must rely solely on $\mathcal{D}_{\text{bandit}}$, which provides only partial supervision—feedback is observed only for recommendations previously exposed by $\pi_{\theta_{\text{init}}}$. Thus, our problem is to continually update a deployed policy from policy-shaped contextual bandit feedback, rather than to solve a standard supervised next-item prediction problem.

3 Anchored Bandit Policy Optimization (ABPO)

3.1 From Vanilla GRPO to Contextual Bandit Feedback

RLVR methods such as Group Relative Policy Optimization (GRPO) (Shao et al., 2024; Guo et al., 2025) serve as an effective framework for continual model updates, allowing the generative policy to be iteratively refined by contrasting diverse recommendation candidates against task-specific reward signals (Yin et al., 2025). Formally, in the LLM-Rec update task, we construct a prompt q consisting of the user context $\mathbf{x}_t$ and a candidate recommendation item set, and ask the model to generate the most suitable item for the target user. Then, given a prompt q, the policy $\pi_{\theta_{\text{old}}}$ samples G outputs $\{a^j\}_{j=1}^G$. Here, $\pi_{\theta_{\text{old}}}$ is the policy from the previous GRPO step, whereas $\pi_{\theta_{\text{init}}}$ is the initially deployed model—these two are distinct. Assuming K reward functions, the aggregated reward for the j-th rollout is computed as

$$r^j_{\text{sum}} = r^j_1 + r^j_2 + \cdots + r^j_K, \tag{1}$$

where r^j_k denotes the k-th reward evaluated on rollout a^j. In recommendation, a commonly used reward is the item-matching reward, which assigns $+1$ if a rollout a^j matches a positive-response exposure, -1 if it matches a no-response exposure, and 0 otherwise. The group-relative advantage for the j-th rollout is then computed by normalizing the aggregated rewards within the rollout group:

$$A^j_{\text{sum}} = \frac{r^j_{\text{sum}} - \text{mean}\{r^1_{\text{sum}}, \ldots, r^G_{\text{sum}}\}}{\text{std}\{r^1_{\text{sum}}, \ldots, r^G_{\text{sum}}\}}. \tag{2}$$

The corresponding multi-reward GRPO optimization objective is

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{\{a^j\}_{j=1}^G\sim\pi_{\theta_{\text{old}}}(\cdot|q)}\left[\frac{1}{G}\sum_{j=1}^G \min\left(s^j(\theta)\,A^j_{\text{sum}},\ \text{clip}(s^j(\theta), 1-\epsilon, 1+\epsilon)\,A^j_{\text{sum}}\right)\right], \tag{3}$$

where $s^j(\theta) = \frac{\pi_\theta(a^j|q)}{\pi_{\theta_{\text{old}}}(a^j|q)}$ and ϵ denotes the clipping threshold. For simplicity, we leave out the KL-divergence regularizer from this formulation. Since $a^{\log}$ is simply the item exposed by the prior deployed policy under partial observability, treating it as a pseudo ground-truth target can create a self-reinforcing feedback loop. Therefore, an effective update method should explicitly account for contextual-bandit bias and calibrate how strongly the exposed recommendation influences policy optimization. Fig. 3 illustrates the architecture of our framework.

3.2 Anchor-Augmented Rollout Group

To account for the fact that all logged feedback is filtered through the exposure mechanism of the prior serving policy, we modify GRPO's rollout-group construction by injecting the exposed recommendation into each rollout group as a fixed anchor. For each logged prompt q, we generate a rollout group of size G under the previous-step policy $\pi_{\theta_{\text{old}}}$, denoted by $\{a^1, a^2, \ldots, a^G\}$, and explicitly replace the first rollout with the exposed recommendation, i.e., $a^1 = a^{\log}$. The remaining $G-1$ rollouts are sampled from the current rollout policy: $a^j \sim \pi_{\theta_{\text{old}}}(\cdot \mid q), \quad j = 2, \ldots, G$. This anchor-augmented construction is the key departure from vanilla GRPO. Rather than evaluating newly sampled rollouts in isolation, this design places them in direct comparison with the historically exposed recommendation under the same logged context. As a result, the exposed recommendation becomes part of the reward statistics used for group-relative normalization, allowing the mean reward of the group to reflect the behavioral baseline induced by the previous policy.

This design is particularly important under policy-shaped contextual bandit feedback. Without anchoring, a sampled rollout rarely matches the logged exposure in a large item space, making the item-matching reward extremely sparse. Anchoring ensures that each group contains at least one rollout tied to the actually observed feedback, so the sparse bandit signal reliably enters group-relative normalization. When the exposed recommendation receives a positive response, anchoring $a^{\log}$ makes it a high-reward reference, raising the group mean and reducing the relative advantage of rollouts that simply copy the logged action. This discourages blind self-reinforcement of the incumbent policy's exposure tendency. Conversely, when the exposed recommendation receives no response, anchoring $a^{\log}$ makes it a low-reward reference, lowering the group mean and preventing penalties on alternative rollouts from being overly amplified. Thus, the anchor serves as a context-dependent calibration point for reward normalization under sparse contextual-bandit feedback.

3.3 Self-Normalized Inverse Propensity Scoring (SNIPS) Reweighting

For the i-th example in a mini-batch, the logged anchor $a_i^{\log}$ is an exposed recommendation produced by the initially deployed logging policy $\pi_{\theta_{\text{init}}}$, whereas the non-anchor rollouts $a_i^{2:G}$ are sampled from the previous-step GRPO policy $\pi_{\theta_{\text{old}}}$. Therefore, the logged anchor reflects the historical exposure distribution and is used only to calibrate the group-relative baseline, not as an additional on-policy rollout in the GRPO surrogate. To account for this exposure mismatch, we use a Self-Normalized Inverse Propensity Scoring (SNIPS)-style weight only when the logged anchor contributes to the group baseline. For each logged anchor, we define the item-level propensity ratio

$$w_i^{\log} = \mathbf{sg}\left[\frac{e_{\text{old}}(a_i^{\log} \mid q_i)}{e_0(a_i^{\log} \mid q_i)}\right], \tag{4}$$

where $e_0(a_i^{\log} \mid q_i)$ denotes the item-level logging propensity under the initial logging policy, and $e_{\text{old}}(a_i^{\log} \mid q_i)$ denotes the item-level exposure probability assigned to the same logged item by the rollout policy used to generate the current group. Both propensities are computed over the same candidate item set $\mathcal{C}_i$. Specifically, for $p \in \{0, \text{old}\}$, we compute a length-normalized LLM token score $s_p(a, q_i)$ for each $a \in \mathcal{C}_i$, and define $e_p(a \mid q_i) = \frac{\exp(s_p(a,q_i)/\tau)}{\sum_{b \in \mathcal{C}_i} \exp(s_p(b,q_i)/\tau)}$. Thus, the ratio in Eq. equation 4 is an item-level propensity ratio, not a ratio of raw token-level likelihoods. The stop-gradient operator $\mathbf{sg}[\cdot]$ prevents the propensity ratio from directly contributing gradients to the policy update. Since positive- and no-responses differ in reliability and scale, we self-normalize the weights separately for the two feedback types. Let $\mathcal{B}_1$ and $\mathcal{B}_0$ denote the sets of positive-response and no-response anchors in the mini-batch, respectively. For each feedback type $y \in \{1, 0\}$, define

$$\bar{w}_y^{\log} = \frac{1}{|\mathcal{B}_y|} \sum_{i \in \mathcal{B}_y} w_i^{\log}, \qquad \hat{w}_i^{\log} = \frac{w_i^{\log}}{\bar{w}_y^{\log} + \delta}. \tag{5}$$

The resulting weight is used to form a SNIPS weighted reward mean:

$$\bar{r}_{i,\text{anc}}^{w} = \frac{\hat{w}_i^{\log} r_i^{\log} + \sum_{j=2}^{G} r_i^j}{\hat{w}_i^{\log} + G - 1}. \tag{6}$$

The advantages used in the GRPO surrogate are then computed only for the policy-sampled rollouts: $A_{i,\text{bandit}}^j = (r_i^j - \bar{r}_{i,\text{anc}}^{w})/\sigma_i, \quad j = 2, \ldots, G$, where σ_i is the weighted standard deviation of the

anchor-augmented reward group, computed with the same anchor weight used in $\bar{r}^{w}_{i,\mathrm{anc}}$. Specifically, $\sigma_i = \sqrt{\frac{\hat{w}_i^{\log}\left(r_i^{\log}-\bar{r}^{w}_{i,\mathrm{anc}}\right)^2+\sum_{j=2}^{G}\left(r_i^{j}-\bar{r}^{w}_{i,\mathrm{anc}}\right)^2}{\hat{w}_i^{\log}+G-1}} + \epsilon_{\mathrm{std}}$. Thus, SNIPS is applied to both positive- and no-response anchors because both are historical exposures generated by the logging policy. However, the logged anchor affects learning only through the weighted group baseline; it does not receive a separate PPO-ratio term or direct off-policy surrogate update. The sampled non-anchor rollouts are optimized with the standard GRPO likelihood ratio $s_i^j(\theta) = \frac{\pi_\theta(a_i^j|q_i)}{\pi_{\theta_{\mathrm{old}}}(a_i^j|q_i)}, \;\; j = 2, \ldots, G.$

3.4 Asymmetric Bandit-Aware Reward Construction

In recommendation, the two feedback cases in contextual bandit logs pose fundamentally different challenges: positive-response events primarily suffer from off-policy bias, whereas no-response events cannot be safely interpreted as clean negatives. Motivated by this asymmetry, we design an asymmetric update mechanism, where positive- and no-response events affect policy optimization through distinct GRPO objective instantiations.

3.4.1 Reward Function

Positive-Response Case. For positive-response events, the exposed recommendation $a^{\log}$ corresponds to an item to which the user responded favorably after being exposed by the historical policy. First, we define the item-matching reward as $r_{\text{match}} = \mathbb{I}\left[a^j = a^{\log}\right]$, which provides a binary signal indicating whether the rollout coincides with the logged action. This captures a core objective in recommendation, namely, improving recommendation accuracy. Second, we define the format compliance reward r_{fmt} as a binary signal for whether the output follows the prescribed schema: the item ID must appear within `<item_id> ... </item_id>` and the item title within `<item> ... </item>`. In industrial generative recommenders, such format compliance is not an auxiliary preference but a minimum requirement for reliable deployment. We aggregate these reward components as in Eq. equation 1, and then normalize the aggregated rewards (i.e., $r_{\text{match}} + r_{\text{fmt}}$) over all anchor-augmented rollouts according to Eq. equation 2 to compute the advantage A^j_{sum}.

No-Response Case. In this case, a rollout should be penalized when it matches $a^{\log}$, as it reproduces an exposure that did not receive user engagement. Accordingly, we define the item-matching reward as $r^j_{\text{match}} = -\mathbb{I}\left[a^j = a^{\log}\right]$, which assigns a penalty when the rollout coincides with the exposed recommendation. This reward discourages the policy from repeatedly reproducing recommendations associated with no-response outcomes. However, a no-response is not a clean rejection. It may reflect weak interest, limited exposure, timing effects, or simple inattention, rather than explicit negative preference. Thus, relying only on the item-matching penalty can over-penalize plausible recommendations under ambiguous feedback. To mitigate this issue, we incorporate self-certainty (Zhao et al., 2025) as an auxiliary intrinsic reward for no-response events. Self-certainty provides a verifier-free confidence signal from the model itself, allowing the update to consider whether a generated recommendation is internally coherent and confidently supported by the policy when external negative evidence is ambiguous. Specifically, for a rollout a^j, its self-certainty is defined as $r^j_{\text{sc}} = \frac{1}{T}\sum_{t=1}^{T} \mathrm{KL}\Big(U \,\|\, \pi_\theta(\cdot \mid q, a^j_{<t})\Big)$, where U denotes the uniform distribution over the token vocabulary, and $a^j_{<t} = (a^j_1, \ldots, a^j_{t-1})$ denotes the token prefix before the t-th token. A larger value indicates a more concentrated output distribution and hence higher model confidence. Therefore, self-certainty does not replace the no-response penalty; rather, it tempers updates by complementing the sparse and ambiguous item-matching signal with an intrinsic reliability signal. Compared with entropy- or perplexity-based confidence measures, self-certainty is less sensitive to length bias and avoids directly encouraging degenerate low-entropy outputs, making it a more suitable sequence-level confidence signal for this setting (Zhao et al., 2025). As in the positive-response case, the format compliance reward is also applied to no-response events. We aggregate the resulting reward components as $r^j = r^j_{\text{match}} + r^j_{\text{fmt}} + \lambda_{\text{sc}} r^j_{\text{sc}}$, and normalize them across all anchor-augmented rollouts to compute the advantage A^j_{sum}.

3.5 Anchor-Constrained Optimization Objective

With the SNIPS-weighted anchored baseline defined above, the final training objective keeps the same GRPO surrogate form, but applies it only to the policy-sampled non-anchor rollouts. The final objective is

$$\mathcal{J}_{\mathrm{ABPO}}(\theta) = \mathbb{E}_{i,\; a_i^{2:G}\sim\pi_{\theta_{\mathrm{old}}}(\cdot|q_i)}\left[\frac{1}{G-1}\sum_{j=2}^{G}\min\Big(s_i^j(\theta)\,A^j_{i,\mathrm{bandit}},\; \mathrm{clip}(s_i^j(\theta), 1-\epsilon, 1+\epsilon)\,A^j_{i,\mathrm{bandit}}\Big)\right], \qquad (7)$$

where $s_i^j(\theta) = \frac{\pi_\theta(a_i^j|q_i)}{\pi_{\theta_{\text{old}}}(a_i^j|q_i)}$. Therefore, our method preserves the optimization backbone of GRPO while modifying two core components to fit logged recommendation feedback: (1) the group-relative baseline is anchor-calibrated using the historically exposed action, without treating it as an additional on-policy rollout in the surrogate; and (2) the scalar utility used for group-relative comparison is redesigned to treat positive- and no-response logs differently. This yields a simple and practical extension of GRPO for continual updating from bandit feedback in generative recommendation. Note that mini-batches are stratified to include both positive- and no-response examples, which stabilizes SNIPS normalization and optimization under severe feedback imbalance.

Remark: Baseline Calibration by Anchored Rollouts. Fix a logged context q and its exposed action $a^{\log}$. In our anchored GRPO update, the logged action is not treated as an additional on-policy rollout in the surrogate. Instead, it is used only to calibrate the group-relative reward baseline. Specifically, let $a^{2:G} \overset{\text{iid}}{\sim} \pi_{\theta_{\text{old}}}(\cdot \mid q)$ denote the policy-sampled rollouts used in the current GRPO step, and let $r^{\log} = r(a^{\log})$ and $r^j = r(a^j)$. Without propensity weighting, the anchored baseline is $\bar{r}_{\text{anc}} = \frac{1}{G} r^{\log} + \frac{G-1}{G} \bar{r}_{\text{roll}}, \quad \bar{r}_{\text{roll}} = \frac{1}{G-1} \sum_{j=2}^{G} r^j$. Taking expectation over the policy-sampled rollouts gives $\mathbb{E}_{a^{2:G} \sim \pi_{\theta_{\text{old}}}}[\bar{r}_{\text{anc}}] = \frac{1}{G} r^{\log} + \frac{G-1}{G} V^{\pi_{\theta_{\text{old}}}}(q)$. Thus, compared with the standard GRPO baseline, the group baseline is pulled toward the realized logged reward by a factor $1/G$.

This calibration acts asymmetrically across feedback types. A positive-response anchor, with high $r^{\log}$, raises the baseline and thus reduces the advantage of rollouts that merely replicate the exposed item. A no-response anchor, with lower $r^{\log}$, lowers the baseline and avoids over-penalizing alternative rollouts under ambiguous feedback. In both cases, the anchor influences learning only through baseline calibration, not through a direct off-policy surrogate term.

4 EXPERIMENTS

In this section, we address the following research questions: **RQ1**. Does our update method improve recommendation accuracy under continual updates from policy-shaped contextual bandit feedback? **RQ2**. Does our framework mitigate popularity bias and improve distributional generalization under continual updates? **RQ3**. Does logged-action anchoring improve the stability and effectiveness of vanilla GRPO updates under contextual bandit feedback? **RQ4**. Is asymmetric treatment of positive-and no-response feedback necessary? **RQ5**. Does our method increase CTR over baselines in online A/B tests?

4.1 DATASETS

To validate the effectiveness of our proposed framework, we conduct experiments across five distinct subsets of the **Amazon Reviews 2023** dataset (Hou et al., 2024) and the **MovieLens** benchmark (Harper & Konstan, 2015). These datasets encompass a wide array of recommendation contexts, ranging from e-commerce retail interactions to cinematic consumption patterns; comprehensive data distributions and statistics are available in Appendix A.

4.2 BASELINES AND EVALUATION SETTINGS

We evaluate our approach using multiple SOTA LLM-Rec backbones: **Rec-SAVER**, **Rec-R1**, **GDPO**, and **G^{2}RPO**. Initially, every backbone is trained on a supervised next-item prediction task using its original methodology to ensure specialized learning from user interaction histories. We then simulate post-deployment updates by refining these trained models with subsequently collected contextual-bandit feedback: the deployed logging policy first exposes recommendations, and the resulting positive- and no-response signals are used as partial feedback (see Appendix A for details). Our study compares several update strategies, including **DEALRec**, **RL-Rec**, **UL-Rec**, and our proposed method, across all backbones. The final recommendation performance is measured on a chronologically distinct test set for every backbone-update pair. Full implementation details are provided in Appendix B.

4.3 PERFORMANCE EVALUATION (RQ1–2)

(1) Recommendation Accuracy (RQ1). We first evaluate whether our update method improves recommendation accuracy under continual updates from policy-shaped contextual bandit feedback. To this end, we report standard top-K recommendation metrics, including HR@1, HR@5, and NDCG@5. Overall, our update method achieves the best or highly competitive accuracy on the vast majority of domain–backbone pairs. In particular, it consistently outperforms *no update* as well as

Table 1: Accuracy comparison on ***Amazon*** and ***MovieLens***; the best and second-best results are marked in **bold** and underline, respectively.

LLM-Rec Backbone	Update Method	No-Update			DEALRec Update			RL-Rec Update			UL-Rec Update			Ours		
	Metric	HR@1	HR@5	N@5	HR@1	HR@5	N@5	HR@1	HR@5	N@5	HR@1	HR@5	N@5	HR@1	HR@5	N@5
	Dataset Domain															
Rec-SAVER	Fashion	12.72	39.60	26.92	17.04	34.37	26.62	19.95	44.69	33.95	14.09	29.52	23.00	**20.48**	**51.45**	**36.90**
	Grocery	10.03	27.92	19.48	10.04	23.34	17.49	9.77	24.30	18.04	10.13	18.88	15.09	**10.50**	**31.72**	**21.84**
	Health	4.51	18.29	11.47	5.10	15.26	10.64	**8.44**	17.53	13.60	3.42	11.36	7.71	6.20	**21.81**	**14.30**
	Clothing	8.09	26.82	17.70	7.58	23.40	16.32	**9.07**	26.30	18.59	8.98	20.81	15.55	8.55	**32.76**	**21.24**
	CDs	10.67	35.22	23.18	12.11	32.22	23.11	19.71	37.20	29.43	15.56	30.26	23.82	**20.59**	**42.46**	**32.12**
	MovieLens	2.23	11.88	7.09	3.65	9.93	7.09	**3.73**	9.54	6.92	2.24	5.63	4.15	2.98	**14.63**	**8.88**
Rec-R1	Fashion	10.16	34.88	22.69	13.78	34.93	25.84	16.42	37.57	28.41	19.25	30.24	25.57	**21.03**	**51.71**	**37.97**
	Grocery	10.23	25.93	18.61	**10.32**	21.45	16.61	8.88	21.18	15.85	8.79	17.89	14.02	9.47	**27.86**	**19.29**
	Clothing	10.46	29.65	19.90	6.57	21.67	14.92	9.57	22.52	16.81	9.41	19.69	15.26	**13.33**	**31.00**	**22.62**
	Health	2.92	9.69	6.45	4.02	15.72	10.44	5.80	16.22	11.38	**5.80**	11.89	9.28	4.94	**21.93**	**13.64**
	CDs	8.71	32.59	21.03	16.23	29.88	23.96	18.66	31.10	25.83	18.12	29.25	24.62	**20.02**	**38.03**	**29.89**
	MovieLens	1.84	9.73	5.75	2.73	8.01	5.66	2.72	8.02	5.68	1.77	4.28	3.20	**3.63**	**11.17**	**7.50**
GDPO	Fashion	15.09	39.71	28.33	16.84	41.29	29.96	23.61	38.37	31.98	19.33	32.84	27.16	**32.80**	**43.73**	**39.26**
	Grocery	12.01	21.92	17.46	10.65	19.52	15.71	12.34	19.96	16.74	10.32	17.38	14.45	**12.51**	**25.39**	**19.54**
	Clothing	11.40	22.63	17.42	8.80	19.79	14.99	10.86	20.49	16.33	11.80	19.65	16.39	**12.54**	**26.28**	**20.15**
	Health	7.57	17.27	12.97	7.52	15.19	11.87	6.27	15.89	11.57	6.87	13.01	10.41	**8.30**	**18.22**	**13.67**
	CDs	15.68	31.54	24.27	18.41	29.13	24.50	21.00	30.64	26.63	20.70	29.14	25.60	**21.07**	**37.22**	**29.84**
	MovieLens	1.32	7.45	4.56	2.42	6.54	4.73	2.74	7.95	5.67	1.64	3.75	2.83	**2.75**	**9.95**	**6.60**
G^2RPO	Fashion	22.10	45.97	34.70	21.23	37.01	30.11	**28.96**	42.50	36.88	15.50	30.76	24.24	25.46	**49.87**	**38.54**
	Grocery	11.17	28.60	20.43	9.84	22.79	17.18	11.22	21.18	16.98	10.23	19.01	15.30	**13.13**	**28.70**	**21.57**
	Clothing	10.46	29.65	19.90	7.95	21.86	15.76	**15.32**	24.49	20.64	9.85	20.50	16.00	11.23	**33.50**	**22.59**
	Health	7.18	19.74	13.89	5.85	16.33	11.79	9.90	16.18	13.55	6.99	13.28	10.57	**10.43**	**20.39**	**15.84**
	CDs	17.58	36.54	27.79	16.74	31.73	25.26	23.94	36.21	30.99	17.96	30.92	25.41	**24.85**	**39.58**	**32.89**
	MovieLens	3.29	11.72	7.64	3.32	7.97	5.87	2.67	7.42	5.29	2.85	7.95	5.65	**4.30**	**11.97**	**8.36**

naive feedback-driven baselines such as DEALRec and UL-Rec updates, indicating that our method can exploit contextual bandit feedback more effectively for improving future recommendation quality. At the same time, *no update* still outperforms several alternative update strategies in many settings, underscoring that update rules which fail to properly account for the structure of contextual-bandit logs can degrade recommendation quality despite using additional feedback.

(2) Popularity Bias (RQ2). We next evaluate whether our update method mitigates prior-policy reinforcement bias while maintaining strong recommendation quality. In recommender systems, optimizing only for accuracy on policy-shaped feedback can make the model repeatedly imitate items favored by the prior policy, resulting in repetitive recommendations and reduced exposure to alternative or long-tail items. To quantify this effect, we measure popularity-based diversity.

Let $p(x)$ denote the normalized popularity of item x estimated from the training data, and let $R_u@K$ be the top-K recommendation list for user u. We define Diversity@K as $\text{Diversity@}K(u) = \frac{1}{K}\sum_{x \in R_u@K} -\log p(x)$. For comparability, we min–max normalize this score to $[0, 1]$ using dataset-level item popularity, multiply it by 100 to obtain a 0–100 scale, and report the average over all users. A value close to 100 indicates stronger long-tail emphasis, whereas a value close to 0 indicates higher popularity bias. Our results show that our update method yields higher diversity than competing LLM-based update strategies in most cases, indicating that its accuracy gains are not obtained by simply over-reinforcing a narrow set of already popular items. Overall, our method achieves a better balance between recommendation accuracy and popularity-aware diversity under continual updates from policy-shaped deployment feedback.

Table 2: Comparison of recommendation diversity across baselines. Each cell reports Div@1 / Div@5.

Domain	Backbone	DEALRec	RLRec	UL-Rec	Ours
Fashion	GRPO	23.73 / 27.12	25.53 / 28.51	27.88 / 26.95	**31.07 / 32.44**
	GDPO	26.61 / 27.43	30.72 / 30.25	28.50 / 28.71	**37.07 / 35.22**
Grocery	GRPO	27.27 / 27.57	26.71 / 27.71	26.31 / 26.83	**28.94 / 28.52**
	GDPO	27.01 / 27.57	29.38 / 27.97	28.78 / 27.07	**30.60 / 28.70**
Clothing	GRPO	28.06 / 29.56	29.77 / 29.89	29.60 / 29.37	**31.53 / 30.79**
	GDPO	29.63 / 29.89	30.53 / 30.27	30.02 / 30.03	**31.31 / 31.33**
Health	GRPO	10.03 / 10.55	**10.70** / 10.73	9.82 / 9.98	9.55 / **11.17**
	GDPO	11.34 / 11.36	11.06 / 10.97	10.56 / 10.81	**12.19 / 12.08**
CDs	GRPO	20.23 / 22.34	21.95 / 23.38	21.56 / 23.24	**24.42 / 25.24**
	GDPO	**24.85** / 23.11	23.60 / 24.17	24.07 / 24.37	22.32 / **26.00**
MovieLens	GRPO	3.66 / 3.64	**3.89** / 3.67	3.35 / 3.40	3.45 / **3.75**
	GDPO	3.61 / 3.56	3.43 / **3.65**	3.91 / 3.38	**3.94** / 3.60

4.4 Ablation Study (RQ3–4)

As shown in Fig. 4, we conduct a component-wise ablation study, using Rec-R1 as the backbone. We compare six variants: (A) **No Update**; (B) **Vanilla GRPO** (uniformly treating positive- and no-response feedback with fixed rewards); (C) **+Anchoring Only** (inserting the logged action as an

anchor into each rollout group); (D) **+Anchoring+SNIPS** (adding importance correction for positive-response samples); (E) **+Anchoring+SC Reward** (adding self-certainty reward for no-response samples); (F) **Ours** (our full method: anchoring + SNIPS + SC reward).

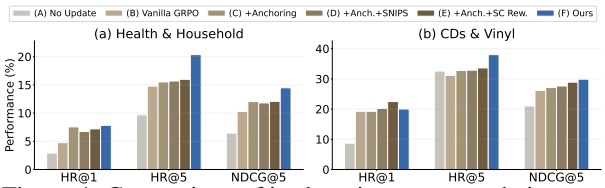

Figure 4: Comparison of in-domain recommendation performance across model variants.

(1) Naively applying GRPO to contextual bandit feedback is not sufficient. Compared with **No Update** (A), **Vanilla GRPO** (B) often improves performance, but the gains are inconsistent and several domains show little benefit or even degradation. This indicates that simply applying GRPO to policy-shaped contextual bandit feedback does not reliably lead to better generalization.

(2) Logged-action anchoring is the key stabilizing design. Comparing **Vanilla GRPO** (B) and **+Anchoring Only** (C), we observe that anchoring improves recommendation performance on most domains, showing that the logged action serves as a useful reference point rather than a label to imitate directly. This supports our answer to **RQ3**: inserting the logged action into each rollout group makes GRPO updates more effective under contextual bandit feedback.

(3) Response feedback benefits from off-policy correction. Comparing **+Anchoring Only** (C) and **+Anchoring+SNIPS** (D), SNIPS yields additional gains on most domains. This suggests that response feedback is informative but still off-policy, making importance-based correction beneficial.

(4) No-response should not be treated as a hard negative. Comparing **+Anchoring Only** (C) and **+Anchoring+SC Reward** (E), we find that self-certainty reward improves performance in many cases. This is because no-response is inherently ambiguous: it does not necessarily mean that the exposed item was truly bad. Rather than giving a fixed negative penalty, self-certainty introduces a softer confidence-aware signal, which reduces overly aggressive suppression and improves generalization.

(5) The full design is complementary and most effective. Finally, **Ours** (F) consistently outperforms the partial variants (B–E), showing that anchoring, SNIPS, and self-certainty reward play complementary roles. Together, these results confirm that effective continual updating under contextual bandit feedback requires both a stable anchor-based relative learning signal and asymmetric handling of positive- and no-response feedback.

4.5 Online A/B Test (RQ5)

Table 3: Monthly Online A/B Test Results

Model	Feb.			Mar.			Apr.		
	Imp.	Clk.	CTR	Imp.	Clk.	CTR	Imp.	Clk.	CTR
Popolarity	106.58K	3.26K	3.06%	100.33K	3.48K	3.47%	95.05K	3.13K	3.30%
ML model	136.08K	9.70K	7.13%	128.92K	9.12K	7.07%	122.81K	8.71K	7.09%
GRPO-Update	184.56K	6.69K	3.63%	180.17K	8.16K	4.53%	170.74K	7.18K	4.20%
Ours	323.09K	23.36K	**7.23%**	306.67K	22.94K	**7.48%**	293.69K	22.37K	**7.62%**

We validated our method via an online A/B test on a production recommender system (Feb 2026–Apr 2026), comparing it against a rolling-window popularity model, a periodically retrained ML recommender, and a GRPO-based continual update baseline. Both the GRPO-Update baseline and our method use Rec-R1 as the backbone. Although traffic shares differed, this imbalance reflects a production-oriented allocation policy: verified LLM-Rec variants with stronger offline and preliminary online performance received larger traffic shares to improve overall service quality. User assignment and metric aggregation were conducted at the user level, reducing the risk of inflated significance from repeated impressions. As shown in Table 3, our method improves across all three months, increasing from 7.23% CTR in February to 7.48% in March and 7.62% in April. It outperforms ML-Retrain in every month: 7.23% vs. 7.13% in February, 7.48% vs. 7.07% in March, and 7.62% vs. 7.09% in April. By contrast, ML-Retrain remains nearly flat, GRPO-Update improves in March but drops in April, and the popularity-based model fluctuates over time. Further details are provided in Appendix D.

5 Conclusion

We introduced **Anchored Bandit Policy Optimization (ABPO)** for continual LLM-Rec updates from policy-shaped deployment feedback. ABPO anchors updates on logged exposures, corrects off-policy bias with SNIPS, and tempers ambiguous no-responses with self-certainty, leading to a more consistent upward CTR trend in production than other update strategies.

AI USE STATEMENT

We used generative AI tools solely for grammar correction and language polishing. All AI-assisted text was reviewed by the authors, who take full responsibility for the final content of this work.

ETHICS STATEMENT

Our online experiments were conducted in a production recommender system under applicable internal policies and privacy requirements. The experiments used standard recommendation interactions without exposing users to interventions beyond the normal service experience. All reported results are aggregated, and no personally identifiable information or user-level data is disclosed.

REPRODUCIBILITY STATEMENT

To ensure full reproducibility, we publicly release all code and datasets used in this work. Comprehensive details regarding model configurations, training and evaluation protocols, and implementation specifics are provided in the Appendix.

REFERENCES

Yuxin Chen, Junfei Tan, An Zhang, Zhengyi Yang, Leheng Sheng, Enzhi Zhang, Xiang Wang, and Tat-Seng Chua. On softmax direct preference optimization for recommendation. *Advances in Neural Information Processing Systems*, 37:27463–27489, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Eric Han, Jun Chen, Karthik Abinav Sankararaman, Xiaoliang Peng, Tengyu Xu, Eryk Helenowski, Kaiyan Peng, Mrinal Kumar, Sinong Wang, Han Fang, et al. Reinforcement learning from user feedback. *arXiv preprint arXiv:2505.14946*, 2025.

F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.

Yupeng Hou, Jiacheng Li, Zhankui He, An Yan, Xiusi Chen, and Julian McAuley. Bridging language and items for retrieval and recommendation. *arXiv preprint arXiv:2403.03952*, 2024.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Jiacheng Lin, Tian Wang, and Kun Qian. Rec-r1: Bridging generative large language models and user-centric recommendation systems via reinforcement learning. *arXiv preprint arXiv:2503.24289*, 2025.

Xinyu Lin, Wenjie Wang, Yongqi Li, Shuo Yang, Fuli Feng, Yinwei Wei, and Tat-Seng Chua. Data-efficient fine-tuning for llm-based recommendation. In *Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval*, pp. 365–374, 2024.

Shih-Yang Liu, Xin Dong, Ximing Lu, Shizhe Diao, Peter Belcak, Mingjie Liu, Min-Hung Chen, Hongxu Yin, Yu-Chiang Frank Wang, Kwang-Ting Cheng, et al. Gdpo: Group reward-decoupled normalization policy optimization for multi-reward rl optimization. *arXiv preprint arXiv:2601.05242*, 2026.

Erxue Min, Hsiu-Yuan Huang, Xihong Yang, Min Yang, Xin Jia, Yunfang Wu, Hengyi Cai, Junfeng Wang, Shuaiqiang Wang, and Dawei Yin. Ctr-guided generative query suggestion in conversational search. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 2624–2634, 2025.

Chung Park, Taesan Kim, Hyeongjun Yun, Dongjoon Hong, Junui Hong, Kijung Park, MinCheol Cho, Mira Myong, Jihoon Oh, et al. Towards trustworthy llm-based recommendation via rationale integration. *arXiv preprint arXiv:2601.02364*, 2025.

Chung Park, Taesan Kim, Hyeongjun Yun, Dongjoon Hong, Junui Hong, Kijung Park, MinCheol Cho, Min Sung Choi, Jihwan Seok, and Jaegul Choo. Think wise, collaborate effectively: A rationale-aware llm-based recommender with reinforcement learning from collaborative signals. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 15609–15616, 2026a.

Chung Park, Hyeongjun Yun, Taesan Kim, Junui Hong, Dongjoon Hong, Mira Myong, Jihoon Oh, MinCheol Cho, Kijung Park, Min sung Choi, Jihwan Seok, and Jaegul Choo. More than what was chosen: Llm-based explainable recommendation beyond noisy user preferences. *arXiv preprint arXiv*, 2026b.

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

Alicia Y Tsai, Adam Kraft, Long Jin, Chenwei Cai, Anahita Hosseini, Taibai Xu, Zemin Zhang, Lichan Hong, Ed H Chi, and Xinyang Yi. Leveraging llm reasoning enhances personalized recommender systems. *arXiv preprint arXiv:2408.00802*, 2024.

Junhao Yin, Haolin Wang, Peng Bao, Ju Xu, and Yongliang Wang. From clicks to preference: A multi-stage alignment framework for generative query suggestion in conversational system. *arXiv preprint arXiv:2508.15811*, 2025.

Jizhi Zhang, Chongming Gao, Wentao Shi, Xin Chen, Jingang Wang, Xunliang Cai, and Fuli Feng. Leveraging unpaired feedback for long-term llm-based recommendation tuning. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 24507–24521, 2025.

Xuandong Zhao, Zhewei Kang, Aosong Feng, Sergey Levine, and Dawn Song. Learning to reason without external rewards. *arXiv preprint arXiv:2505.19590*, 2025.

A DATASET

We conduct experiments on five **Amazon Review 2023**[1] (Hou et al., 2024) domains and the **MovieLens** dataset[2] (Harper & Konstan, 2015), which together cover heterogeneous recommendation scenarios ranging from e-commerce purchases to movie consumption, as summarized in Table 4. Specifically, the Amazon Review dataset includes five practically relevant domains: (1) *Amazon* ***Fashion***, (2) ***Clothing*** *Shoes and Jewelry*, (3) ***Grocery*** *and Gourmet Food*, (4) ***Health*** *and Household*, and (5) ***CDs*** *and Vinyl*. In addition, MovieLens 32M (***ml-32m***) serves as a standard large-scale benchmark for movie recommendation.

Table 4: Dataset Statistics

Dataset	Domain	#Users	#Items	Avg Length	Avg Item Purchase	Sparsity
Amazon Review	Fashion	11,028	59,004	6.54	1.23	0.9994
	Clothing	13,766	107,353	11.83	1.65	0.9994
	Grocery	11,334	6,149	10.76	21.02	0.9926
	Health	21,152	74,784	8.49	2.50	0.9994
	CDs	17,035	83,774	13.71	1.97	0.9993
MovieLens	ml-32m	10,623	12,259	48.96	42.43	0.9960

For each dataset, we assume that every user is associated with a chronological interaction sequence over timesteps $1 : T + N$. Based on this sequence, we construct three types of data: (1) data for training the initial policy model, (2) offline contextual bandit logs generated by the learned logging policy, and (3) a held-out test set. The initial policy is first trained in a standard next-item prediction manner, using the interaction history up to timestep T to predict the item at timestep $T+1$. Specifically, multiple backbone models, including Rec-SAVER and Rec-R1, are separately trained on this data and subsequently used as the initially deployed logging policies $\pi_{\theta_{\text{init}}}$ for continual updating.

Offline construction of synthetic bandit feedback. Definition 2 describes post-deployment contextual bandit feedback in an online system. Since standard sequential recommendation datasets do not contain real randomized exposure-response logs, we construct synthetic one-step bandit feedback from next-item sequences.

For each user sequence, the prefix $\mathbf{x}_{1:T}$ is treated as the initial history, and x_{T+N} is reserved as the held-out evaluation target. We construct update logs only for

$$t' \in \{T, \ldots, T + N - 2\},$$

so that the next item used to construct synthetic feedback, $x_{t'+1}$, always belongs to $\{x_{T+1}, \ldots, x_{T+N-1}\}$ and never equals the held-out evaluation target x_{T+N}.

For each update context $\mathbf{x}_{1:t'}$, we construct a controlled M-item candidate set $\mathcal{C}_{t'}$. Specifically, we include the next observed item $x_{t'+1}$ as the unique latent positive item and uniformly sample M-1 distractor items from the user's non-interacted items:

$$\mathcal{C}_{t'} = \{x_{t'+1}\} \cup \mathcal{N}_{t'}, \qquad |\mathcal{N}_{t'}| = M - 1, \qquad |\mathcal{C}_{t'}| = M.$$

The distractor set $\mathcal{N}_{t'}$ is sampled from items that do not appear in the user's observed interaction sequence and excludes the held-out evaluation target x_{T+N}. This controlled candidate construction ensures that a positive feedback event is possible for every update context while the held-out evaluation target is never included in any training-time candidate set. Unless otherwise specified, we use a candidate set size of M=200 items.

Given $\mathcal{C}_{t'}$, the logging policy scores every candidate item $a \in \mathcal{C}_{t'}$ using the length-normalized LLM likelihood

$$s_0(a, \mathbf{x}_{1:t'}) = \frac{1}{|r(a)|} \sum_j \log \pi_{\theta_{\text{init}}} \left(r_j(a) \mid \text{Prompt}(\mathbf{x}_{1:t'}, \mathcal{C}_{t'}), r_{<j}(a) \right),$$

where $r(a)$ denotes the textual representation of item a. These scores define a candidate-level logging distribution

$$e_0(a \mid \mathbf{x}_{1:t'}) = \frac{\exp(s_0(a, \mathbf{x}_{1:t'})/\tau)}{\sum_{b \in \mathcal{C}_{t'}} \exp(s_0(b, \mathbf{x}_{1:t'})/\tau)}.$$

The logged recommendation is then sampled as

$$a_{t'}^{\log} \sim \text{Categorical}\left(e_0(\cdot \mid \mathbf{x}_{1:t'})\right),$$

[1] https://amazon-reviews-2023.github.io

[2] https://grouplens.org/datasets/movielens

and we store its logging propensity $e_0(a_{t'}^{\log} \mid \mathbf{x}_{1:t'})$. This propensity is exact within our offline construction because all 200 items in $\mathcal{C}_{t'}$ are scored and the logged item is sampled from the resulting controlled candidate-level softmax distribution. It is therefore a candidate-conditional exposure probability, not a raw token likelihood and not a probability normalized over the full item catalog.

Because real user responses to the sampled exposure are unavailable, we assign a synthetic bandit feedback label by checking whether the sampled exposed item matches the next observed item:

$$y_{t'} = \mathbf{1}\left[a_{t'}^{\log} = x_{t'+1}\right].$$

Thus, $y_{t'} = 1$ means that the logging policy exposed the latent positive item in the candidate set, while $y_{t'} = 0$ means that it exposed one of the distractor items. We use $y_{t'} = 0$ as a no-positive-response proxy rather than an explicit negative preference, since the offline sequence does not provide direct evidence that the user would dislike the exposed distractor.

The resulting synthetic contextual bandit dataset is

$$\mathcal{D}_{\text{bandit}} = \left\{\left(\mathbf{x}_{1:t'}, a_{t'}^{\log}, y_{t'}, e_0(a_{t'}^{\log} \mid \mathbf{x}_{1:t'})\right)\right\}_{t'=T}^{T+N-2}.$$

Finally, each updated model is evaluated on the held-out next-item prediction task: given the evaluation context

$$\mathbf{x}_{1:T+N-1},$$

the model predicts the held-out item x_{T+N}. Since update feedback labels are constructed only from next items up to x_{T+N-1}, and x_{T+N} is excluded from all training-time candidate sets, the held-out target is not used for logging, feedback construction, or policy updating.

B Reproducibility and Implementation Details

All experiments were conducted on a cluster of eight NVIDIA H200 GPUs. To ensure reproducibility, we make our implementation code publicly available through anonymous links.

■ **Code Link**: https://anonymous.4open.science/r/ABPO-5CD8

Hyperparameters. Table 5 summarizes the hyperparameter configurations used in our experiments, including the learning rate, batch size, optimizer, warm-up ratio, gradient clipping threshold, weight decay, and training epochs. We use the same LLM backbone (i.e., GEMMA-3-4B-IT) for all methods to ensure a fair comparison.

Training proceeds in two stages. First, we obtain the initial deployed policy $\pi_{\theta_{\text{init}}}$ through supervised fine-tuning (SFT) on the supervised recommendation dataset $\mathcal{D}_{\text{sup}}$. This SFT checkpoint is shared by all RL-based update methods, including GRPO-based baselines and our proposed ABPO. Second, each method performs continual policy updates using the post-deployment contextual-bandit dataset $\mathcal{D}_{\text{bandit}}$.

Table 5: Training hyperparameters

Hyperparameter	Backbone		Update
	SFT	RL	
Batch size	8	4	4
Epochs	2	2	1
Max seq. length	10k	10k	12k
Optimizer	Paged AdamW 8-bit		
Learning rate	5.0×10^{-5}		
Warm-up ratio	0.05		
Grad. accum. steps	8		
Weight decay	0.01		
Grad. clipping	1.0		
LoRA r / α / dropout	8 / 16 / 0		
LoRA modules	v, q, o, gate, up, down proj.		
Precision	bf16		
Grad. checkpointing	True		
Parallelism	DeepSpeed ZeRO-2		
Hardware	8 × NVIDIA H200 GPUs		

For all RL-based update methods, we use the same rollout group size (16), decoding configuration, optimizer, and parameter-efficient fine-tuning setup unless otherwise specified. Our ABPO uses the same GRPO-style clipped policy objective as the baselines, but differs in how the rollout group is constructed and normalized: the logged exposure is inserted as an anchor, SNIPS is applied to correct the anchor contribution under policy mismatch, and reward components are normalized over the anchor-augmented group. Parameter-efficient fine-tuning is performed via LoRA (Hu et al., 2022), and all implementation details required for reproducibility are provided in Table 5.

Evaluation Protocol. We adopt a standard leave-one-out evaluation strategy, where the final interaction in each user sequence is reserved for testing, the penultimate interaction for validation, and all earlier interactions for training. During evaluation, each inference prompt explicitly includes a candidate set of 200 items, consisting of the ground-truth target item and unseen negative candidates. To better reflect realistic candidate competition, we complement randomly sampled negatives with popularity-based unseen hard negatives. For all baselines and our model, the user context and candidate set in each prompt are kept identical. DPO-based baselines require explicit construction of negative samples and were therefore trained with additional negatives.

C BASELINES

LLM-Rec Update Model. We survey and organize prior LLM-Rec update methods relevant to our study, and describe how they are implemented in our experiments.

■ **DEALRec** (Lin et al., 2024) reduces the cost of updating LLM-based recommenders by pruning recommendation data before few-shot fine-tuning. It selects influential user sequences using an influence score estimated from a lightweight surrogate recommender and an effort score based on LLM gradients, while applying coverage-enhanced sampling to preserve data diversity. We implemented this baseline from scratch.

■ **RLRec** (Yin et al., 2025) aligns an LLM-based generative recommender with post-deployment user feedback by converting noisy click logs into preference signals. It builds an initial policy via supervised fine-tuning on clicked suggestions, models preference uncertainty with a Gaussian Reward Model (GaRM), and further updates the policy using GRPO with a composite reward to improve user engagement while controlling reward hacking. We implemented this baseline from scratch.

■ **ULRec** (Zhang et al., 2025) updates LLM-based recommenders using unpaired post-interaction feedback to improve long-term user satisfaction. It decomposes successful trajectories into step-level positive supervision for SFT, and further applies KTO on negative feedback signals such as user disengagement to discourage recommendations that reinforce echo chambers. We implemented this baseline from scratch.

LLM-Rec Backbone. We summarize the LLM-based baselines used in our experiments and describe their implementations below.

■ **Rec-SAVER** (Tsai et al., 2024) enhances LLM-based recommendation by fine-tuning models with target-item labels and corresponding rationales. The approach shows that rationale-supervised instruction tuning is effective for both zero-shot transfer and task-specific fine-tuning, including non-overlapping cross-domain settings. We evaluate this baseline with our own in-house implementation.

■ **Rec-R1** (Lin et al., 2025) applies GRPO to recommendation, where the reward is based on whether the generated item matches the ground-truth next item. We built this baseline using a public GRPO implementation[3].

■ **GDPO** (Liu et al., 2026) is a reward-decoupled extension of GRPO for multi-objective RL training. Rather than aggregating multiple rewards before normalization, GDPO normalizes each reward component independently and then combines the resulting signals for policy optimization. This helps preserve reward-specific variations that may otherwise be obscured by scalar reward aggregation. Prior evaluations on tool-use, mathematical reasoning, and coding benchmarks show that GDPO can provide more stable training and stronger overall performance than GRPO. Although it was not developed for recommendation, its ability to handle heterogeneous rewards makes it a suitable baseline for LLM-based recommender training. We implement GDPO using a public GRPO codebase and adapt it to our recommendation setting.

■ **G^{2}RPO** [4] extends GDPO with accuracy-aware dynamic reward aggregation for LLM-based recommendation. Instead of uniformly summing normalized reward signals, it computes aggregation weights by solving a simplex-constrained linear program that aligns the final update direction with the accuracy gradient while preventing auxiliary rewards from inducing harmful updates. This design

[3] https://github.com/huggingface/open-r1

[4] https://anonymous.4open.science/r/G2RPO-FDB9

Table 6: Monthly online A/B test results from February to April. Δ denotes the CTR difference between our method and each baseline, i.e., Ours − Baseline, in percentage points. Statistical significance is computed using a two-sided two-proportion z-test.

Month	Model	Impressions	Clicks	CTR (%)	Δ	z	p
February	Popularity	106,583	3,260	3.06	+4.17	48.98	< 0.001
	ML-Retrain	136,084	9,702	7.13	+0.10	1.20	0.232
	GRPO-Update	184,557	6,691	3.63	+3.60	52.34	< 0.001
	Ours	323,085	23,357	7.23	–	–	–
March	Popularity	100,333	3,481	3.47	+4.01	44.75	< 0.001
	ML-Retrain	128,922	9,121	7.07	+0.40	4.67	< 0.001
	GRPO-Update	180,169	8,160	4.53	+2.95	40.65	< 0.001
	Ours	306,672	22,937	7.48	–	–	–
April	Popularity	95,047	3,133	3.30	+4.32	46.75	< 0.001
	ML-Retrain	122,813	8,708	7.09	+0.53	5.88	< 0.001
	GRPO-Update	170,743	7,179	4.20	+3.41	45.92	< 0.001
	Ours	293,686	22,366	7.62	–	–	–

Table 7: Month-to-month CTR changes in the online A/B test. Statistical significance is computed using a two-sided two-proportion z-test between the CTRs of the two corresponding months.

Model	**Feb.→Mar.**			**Mar.→Apr.**			**Feb.→Apr.**		
	Δ CTR	z	p	Δ CTR	z	p	Δ CTR	z	p
Popularity	+0.41%p	5.26	< 0.001	-0.17%p	-2.12	0.034	+0.24%p	3.04	0.002
ML-Retrain	-0.05%p	-0.55	0.584	+0.02%p	0.15	0.879	-0.04%p	-0.39	0.700
GRPO-Update	+0.90%p	13.81	< 0.001	-0.32%p	-4.70	< 0.001	+0.58%p	8.90	< 0.001
Ours	+0.25%p	3.80	< 0.001	+0.14%p	2.00	0.046	+0.39%p	5.78	< 0.001

enables multi-reward optimization over accuracy, diversity, and groundedness without treating all objectives as equally tradeable.

D Detailed Online A/B Test Analysis

We validated our method via an online A/B test on a production recommender system (February 2026–April 2026), comparing it against a rolling-window popularity model, a periodically retrained conventional ML recommender, and a GRPO-based continual update model, with users assigned through stable hash-based randomization. Although traffic shares were not uniform, larger shares were allocated to LLM-Rec variants after their effectiveness had been verified, to improve overall production performance, while the randomized user-level assignment kept user populations comparable. Both the GRPO-based baseline and our method use Rec-R1 as the backbone. To examine whether each model benefits from successive production updates, we further provide a month-level breakdown from February to April 2026, where each update was customized to reflect the business logic of the production service. Table 6 reports impressions, clicks, CTRs, and pairwise statistical comparisons between our method and each baseline. Statistical significance is computed using a two-sided two-proportion z-test at the impression level.

To further analyze whether the observed improvements reflect a consistent trend rather than a single-month fluctuation, Table 7 reports month-to-month CTR changes for each model. In particular, we compare February to March, March to April, and February to April using the same two-sided two-proportion z-test.

Across the three months, our method is the only model that shows a monotonic improvement in CTR: 7.23% in February, 7.48% in March, and 7.62% in April. As shown in Table 7, the improvement is statistically significant in both consecutive intervals: +0.25%p from February to March ($z = 3.80$, $p < 0.001$) and +0.14%p from March to April ($z = 2.00$, $p = 0.046$). Overall, this yields a statistically significant +0.39%p gain from February to April ($z = 5.78$, $p < 0.001$). These results suggest that the improvement is not driven by a single-month fluctuation, but follows a consistent upward trend across successive production update rounds.

The comparison with ML-Retrain provides a conservative reference point because ML-Retrain is the strongest non-LLM continual-update baseline in terms of absolute CTR. In February, our method is numerically higher than ML-Retrain, but the difference is not statistically significant (+0.10%p, $z = 1.20$, $p = 0.232$). However, after additional update rounds, the gap becomes statistically significant: +0.40%p in March ($z = 4.67$, $p < 0.001$) and +0.53%p in April ($z = 5.88$, $p < 0.001$). By contrast, ML-Retrain remains nearly flat over time, with no statistically significant change from February to March ($z = -0.55$, $p = 0.584$), from March to April ($z = 0.15$, $p = 0.879$), or from February to April ($z = -0.39$, $p = 0.700$). This suggests that periodic supervised retraining alone does not yield sustained post-deployment improvement under the observed feedback stream.

The popularity-based baseline shows a different pattern. Its CTR increases from 3.06% in February to 3.47% in March (+0.41%p, $z = 5.26$, $p < 0.001$), but decreases again to 3.30% in April (-0.17%p, $z = -2.12$, $p = 0.034$). Although its February-to-April change is statistically significant (+0.24%p, $z = 3.04$, $p = 0.002$), the month-to-month pattern is not monotonic, and its absolute CTR remains substantially below our method in every month.

GRPO-Update also improves from February to March, increasing from 3.63% to 4.53% (+0.90%p, $z = 13.81$, $p < 0.001$). However, this improvement is not sustained: its CTR significantly drops from 4.53% in March to 4.20% in April (-0.32%p, $z = -4.70$, $p < 0.001$). Although the February-to-April change remains positive overall (+0.58%p, $z = 8.90$, $p < 0.001$), the decline after March suggests that naive GRPO-style continual updating may be less stable under policy-shaped contextual-bandit feedback.

Our method significantly outperforms GRPO-Update in every month, with gaps of +3.60%p in February ($z = 52.34$, $p < 0.001$), +2.95%p in March ($z = 40.65$, $p < 0.001$), and +3.41%p in April ($z = 45.92$, $p < 0.001$). This comparison is particularly informative because both methods use the same LLM-based recommender backbone. Therefore, the performance gap mainly reflects the effect of our proposed update design: anchoring the logged exposure, correcting off-policy exposure bias, and tempering ambiguous no-response feedback.

Aggregating all three months, our method obtains 68,660 clicks from 923,443 impressions, corresponding to a CTR of 7.44%. This outperforms ML-Retrain by +0.34%p ($z = 6.74$, $p < 0.001$), GRPO-Update by +3.32%p ($z = 80.08$, $p < 0.001$), and the popularity-based model by +4.17%p ($z = 81.13$, $p < 0.001$). Together, the aggregate and month-level results suggest that our method not only improves overall CTR, but also enables more stable continual improvement across successive production update rounds.

E Theoretical Analysis

We formalize the baseline-calibration effect of the logged anchor under contextual bandit feedback. Throughout this section, expectations are conditioned on a fixed user context q and a logged tuple $(a^{\log}, y)$, and we suppress this conditioning for brevity. The reward function $r(a) \equiv r(a, q, a^{\log}, y)$ may depend on the logged tuple through the feedback-dependent reward design.

A key distinction in ABPO is that the logged anchor is not optimized as an on-policy rollout. The logged action $a^{\log}$ was sampled from the initial item-level logging distribution $e_0(\cdot \mid q)$, whereas the non-anchor rollouts used in the current GRPO update are sampled from the item-level rollout distribution induced by the previous-step policy:

$$\rho(\cdot \mid q) := e_{\text{old}}(\cdot \mid q).$$

The optimized policy π_θ appears only through the GRPO likelihood ratio in the final surrogate objective. The analysis below is therefore with respect to the rollout-generating distribution ρ, not the optimizing policy π_θ.

For notational simplicity, we write $e_p(a \mid q)$ for item-level exposure propensities. In our offline construction, the candidate set $\mathcal{C}(q)$ is deterministically constructed from the context and included in the LLM prompt. For $p \in \{0, \text{old}\}$, e_p is obtained by first computing a length-normalized average token log-probability $s_p(a, q)$ for each candidate item $a \in \mathcal{C}(q)$, and then applying a temperature-scaled softmax:

$$e_p(a \mid q) = \frac{\exp(s_p(a, q)/\tau)}{\sum_{b \in \mathcal{C}(q)} \exp(s_p(b, q)/\tau)}.$$

Thus, e_0 and e_{old} are item-level exposure probabilities over the same candidate item set, not raw token-level likelihoods of the generated item text.

For clarity, we analyze the centered reward $r(a_j) - \bar{r}^{\omega}_{\text{anc}}$. In implementation, ABPO further divides this quantity by a positive weighted standard deviation, as in the main text. This normalization changes the scale of the advantage but not the direction of the baseline-induced shift analyzed below.

Notation. Let

$$V^{\rho}(q) = \mathbb{E}_{a\sim\rho(\cdot|q)}[r(a)]$$

denote the expected reward under the rollout distribution. For the $G-1$ policy-sampled non-anchor rollouts

$$a_2, \ldots, a_G \overset{\text{iid}}{\sim} \rho(\cdot \mid q),$$

define the rollout-only baseline

$$\bar{r}_{\text{roll}} = \frac{1}{G-1}\sum_{j=2}^{G} r(a_j). \tag{8}$$

The logged anchor contributes only to the group-relative baseline. Given a nonnegative stopped-gradient anchor weight $\omega \geq 0$, define the weighted anchored baseline as

$$\bar{r}^{\omega}_{\text{anc}} = \frac{\omega r(a^{\log}) + \sum_{j=2}^{G} r(a_j)}{\omega + G - 1}. \tag{9}$$

The unweighted anchor baseline is recovered by setting $\omega = 1$. In ABPO, ω corresponds to the self-normalized logged-anchor weight $\hat{w}^{\log}$, computed from an item-level propensity ratio and treated as stop-gradient.

The centered advantages induced by this baseline are

$$A^{\omega}_j = r(a_j) - \bar{r}^{\omega}_{\text{anc}}, \qquad j = 2, \ldots, G. \tag{10}$$

There is no corresponding surrogate advantage for the logged anchor $a^{\log}$.

In the implementation, the final advantage additionally uses the weighted standard deviation

$$\sigma_{\omega} = \sqrt{\frac{\omega\left(r(a^{\log}) - \bar{r}^{\omega}_{\text{anc}}\right)^2 + \sum_{j=2}^{G}\left(r(a_j) - \bar{r}^{\omega}_{\text{anc}}\right)^2}{\omega + G - 1} + \epsilon_{\text{std}}}, \tag{11}$$

and applies

$$\tilde{A}^{\omega}_j = \frac{r(a_j) - \bar{r}^{\omega}_{\text{anc}}}{\sigma_{\omega}}, \qquad j = 2, \ldots, G.$$

Since $\sigma_{\omega} > 0$, the sign of the centered advantage shift is preserved.

E.1 Baseline Calibration

Lemma 1 (Rollout-Only Baseline is Centered at the Rollout Distribution)**.** *The expected rollout-only baseline satisfies*

$$\mathbb{E}[\bar{r}_{\text{roll}}] = V^{\rho}(q) = \mathbb{E}_{a\sim\rho(\cdot|q)}[r(a)]. \tag{12}$$

Proof. Since $a_2, \ldots, a_G$ are sampled i.i.d. from $\rho(\cdot \mid q)$, each rollout satisfies

$$\mathbb{E}[r(a_j)] = V^{\rho}(q), \qquad j = 2, \ldots, G.$$

Therefore, by linearity of expectation,

$$\mathbb{E}[\bar{r}_{\text{roll}}] = \mathbb{E}\left[\frac{1}{G-1}\sum_{j=2}^{G} r(a_j)\right] = \frac{1}{G-1}\sum_{j=2}^{G} V^{\rho}(q) = V^{\rho}(q).$$

□

Lemma 1 shows that without anchoring, the group-relative baseline is centered at the average reward of the current rollout distribution. Thus, although the reward function may depend on the logged tuple $(a^{\log}, y)$, the realized logged reward $r(a^{\log})$ does not explicitly enter the rollout-only baseline.

Theorem 1 (Weighted Anchoring Shifts the Baseline Toward the Logged Reward)**.** *Condition on a fixed nonnegative stopped-gradient anchor weight ω. Then the expected anchored baseline satisfies*

$$\mathbb{E}[\bar{r}^{\omega}_{\text{anc}} \mid \omega] = \frac{\omega}{\omega + G - 1} r(a^{\log}) + \frac{G-1}{\omega + G - 1} V^{\rho}(q). \tag{13}$$

Equivalently, if

$$\alpha_{\omega} = \frac{\omega}{\omega + G - 1},$$

then

$$\mathbb{E}[\bar{r}^{\omega}_{\text{anc}} \mid \omega] = \alpha_{\omega} r(a^{\log}) + (1 - \alpha_{\omega}) V^{\rho}(q). \tag{14}$$

Thus, the anchored baseline is a convex combination of the realized logged reward and the rollout distribution's expected reward.

Moreover, define the deviations from the logged reward as

$$\delta_{\text{roll}} = \mathbb{E}[\bar{r}_{\text{roll}}] - r(a^{\log}), \qquad \delta^{\omega}_{\text{anc}} = \mathbb{E}[\bar{r}^{\omega}_{\text{anc}} \mid \omega] - r(a^{\log}). \tag{15}$$

Then

$$\delta^{\omega}_{\text{anc}} = \frac{G-1}{\omega + G - 1} \delta_{\text{roll}}. \tag{16}$$

In particular, for the unweighted case $\omega = 1$,

$$\delta^{1}_{\text{anc}} = \frac{G-1}{G} \delta_{\text{roll}}. \tag{17}$$

Proof. Using Eq. equation 9,

$$\begin{aligned}
\mathbb{E}[\bar{r}^{\omega}_{\text{anc}} \mid \omega] &= \mathbb{E}\left[\frac{\omega r(a^{\log}) + \sum_{j=2}^{G} r(a_j)}{\omega + G - 1} \,\middle|\, \omega \right] \\
&= \frac{\omega r(a^{\log}) + \sum_{j=2}^{G} \mathbb{E}[r(a_j)]}{\omega + G - 1} \\
&= \frac{\omega r(a^{\log}) + (G-1) V^{\rho}(q)}{\omega + G - 1}.
\end{aligned} \tag{18}$$

This proves Eq. equation 13. Rewriting with $\alpha_{\omega} = \omega/(\omega + G - 1)$ gives Eq. equation 14.

By Lemma 1,

$$\delta_{\text{roll}} = V^{\rho}(q) - r(a^{\log}).$$

Similarly,

$$\begin{aligned}
\delta^{\omega}_{\text{anc}} &= \frac{\omega r(a^{\log}) + (G-1) V^{\rho}(q)}{\omega + G - 1} - r(a^{\log}) \\
&= \frac{G-1}{\omega + G - 1} \left(V^{\rho}(q) - r(a^{\log}) \right) \\
&= \frac{G-1}{\omega + G - 1} \delta_{\text{roll}}.
\end{aligned} \tag{19}$$

Setting $\omega = 1$ yields Eq. equation 17. □

Theorem 1 formalizes the calibration role of the logged anchor. The anchor does not create a direct off-policy surrogate term. Instead, its reward shifts the baseline used to evaluate the current policy-sampled rollouts. The strength of this shift is controlled by the stopped-gradient weight ω.

E.2 Effect on Non-Anchor Rollout Advantages

Since the logged anchor is not optimized directly, the relevant quantity is not the anchor's own advantage, but how the anchored baseline changes the advantages of the policy-sampled rollouts.

Proposition 1 (Non-Anchor Advantage Shift)**.** *Let*

$$A_j^{\mathrm{roll}} = r(a_j) - \bar{r}_{\mathrm{roll}}, \qquad j = 2, \dots, G,$$

be the rollout-only group-relative advantage. Under the weighted anchored baseline, the non-anchor advantage satisfies

$$A_j^{\omega} = A_j^{\mathrm{roll}} + \frac{\omega}{\omega + G - 1} \left(\bar{r}_{\mathrm{roll}} - r(a^{\log}) \right), \qquad j = 2, \dots, G. \tag{20}$$

Taking expectation over the policy-sampled rollouts gives

$$\mathbb{E}[A_j^{\omega} \mid \omega] = \frac{\omega}{\omega + G - 1} \left(V^{\rho}(q) - r(a^{\log}) \right), \qquad j = 2, \dots, G. \tag{21}$$

Proof. By definition,

$$A_j^{\omega} = r(a_j) - \bar{r}_{\mathrm{anc}}^{\omega}, \qquad A_j^{\mathrm{roll}} = r(a_j) - \bar{r}_{\mathrm{roll}}.$$

Therefore,

$$\begin{aligned} A_j^{\omega} - A_j^{\mathrm{roll}} &= \bar{r}_{\mathrm{roll}} - \bar{r}_{\mathrm{anc}}^{\omega} \\ &= \bar{r}_{\mathrm{roll}} - \frac{\omega r(a^{\log}) + \sum_{k=2}^{G} r(a_k)}{\omega + G - 1}. \end{aligned} \tag{22}$$

Since

$$\sum_{k=2}^{G} r(a_k) = (G-1)\bar{r}_{\mathrm{roll}},$$

we obtain

$$\begin{aligned} A_j^{\omega} - A_j^{\mathrm{roll}} &= \bar{r}_{\mathrm{roll}} - \frac{\omega r(a^{\log}) + (G-1)\bar{r}_{\mathrm{roll}}}{\omega + G - 1} \\ &= \frac{\omega}{\omega + G - 1} \left(\bar{r}_{\mathrm{roll}} - r(a^{\log}) \right). \end{aligned} \tag{23}$$

This proves Eq. equation 20. Taking expectation and using Lemma 1 gives Eq. equation 21. □

Proposition 1 shows that the logged action itself is not assigned a surrogate advantage. Instead, the logged reward shifts the advantages of the non-anchor rollouts through the baseline.

E.3 Implications for Positive- and No-Response Feedback

The sign of the baseline shift is determined by the relative ordering between the logged reward $r(a^{\log})$ and the rollout distribution's expected reward $V^{\rho}(q)$.

Corollary 1 (Positive-Response Calibration)**.** *Suppose a positive-response logged exposure satisfies*

$$r(a^{\log}) > V^{\rho}(q). \tag{24}$$

Then the weighted anchored baseline is higher in expectation than the rollout-only baseline:

$$\mathbb{E}[\bar{r}_{\mathrm{anc}}^{\omega} \mid \omega] - \mathbb{E}[\bar{r}_{\mathrm{roll}}] = \frac{\omega}{\omega + G - 1} \left(r(a^{\log}) - V^{\rho}(q) \right) > 0. \tag{25}$$

Consequently, the expected centered advantages of the policy-sampled rollouts are shifted downward by

$$\mathbb{E}[A_j^{\omega} \mid \omega] = -\frac{\omega}{\omega + G - 1} \left(r(a^{\log}) - V^{\rho}(q) \right) < 0, \qquad j = 2, \dots, G. \tag{26}$$

Proof. The baseline shift follows directly from Eq. equation 13 and Lemma 1:

$$\mathbb{E}[\bar{r}^{\omega}_{\mathrm{anc}} \mid \omega] - \mathbb{E}[\bar{r}_{\mathrm{roll}}] = \frac{\omega}{\omega + G - 1} \left(r(a^{\log}) - V^{\rho}(q) \right).$$

Under condition equation 24, this quantity is positive. Eq. equation 26 follows from Proposition 1. □

Thus, when a positive-response logged item receives a higher reward than the rollout distribution's average reward, anchoring raises the comparison baseline. This reduces the relative advantage of policy rollouts that do not improve upon the positive logged reference, without directly applying a policy-gradient update to the logged action itself.

Corollary 2 (No-Response Buffering)**.** *Suppose a no-response logged exposure satisfies*

$$r(a^{\log}) < V^{\rho}(q). \tag{27}$$

Then the weighted anchored baseline is lower in expectation than the rollout-only baseline:

$$\mathbb{E}[\bar{r}^{\omega}_{\mathrm{anc}} \mid \omega] - \mathbb{E}[\bar{r}_{\mathrm{roll}}] = \frac{\omega}{\omega + G - 1} \left(r(a^{\log}) - V^{\rho}(q) \right) < 0. \tag{28}$$

Consequently, the expected centered advantages of the policy-sampled rollouts are shifted upward by

$$\mathbb{E}[A^{\omega}_{j} \mid \omega] = \frac{\omega}{\omega + G - 1} \left(V^{\rho}(q) - r(a^{\log}) \right) > 0, \qquad j = 2, \ldots, G. \tag{29}$$

Proof. The proof is identical to that of Corollary 1, with the inequality direction reversed under condition equation 27. □

Thus, when a no-response logged item receives a lower reward than the rollout distribution's average reward, anchoring lowers the baseline. This can buffer alternative rollouts from being overly penalized under ambiguous no-response feedback. Again, the effect is mediated only through the baseline, not through a direct off-policy surrogate term for the logged anchor.

E.4 Anchor-Level Interpretation of SNIPS

We do not use SNIPS to characterize the full ABPO objective. SNIPS is used only to determine the stopped-gradient coefficient with which the logged reward contributes to the anchored baseline in Eq. equation 9. The logged anchor is not assigned a GRPO likelihood ratio or a direct off-policy surrogate advantage.

Item-level propensity ratio. Let $\rho = e_{\mathrm{old}}$ denote the item-level rollout distribution induced by the previous-step policy, and let e_0 denote the item-level logging distribution used to sample the logged exposure. For any fixed feedback type $y \in \{0, 1\}$, let $f_y(a, q)$ denote a bounded scalar function of the logged action, such as the feedback-specific logged reward. Define the item-level importance ratio

$$w(a, q) = \frac{e_{\mathrm{old}}(a \mid q)}{e_0(a \mid q)}. \tag{30}$$

Both e_0 and e_{old} are defined over the same deterministic candidate set $\mathcal{C}(q)$ included in the prompt. They are obtained by softmax-normalizing length-normalized LLM token scores over candidate items:

$$e_p(a \mid q) = \frac{\exp(s_p(a, q)/\tau)}{\sum_{b \in \mathcal{C}(q)} \exp(s_p(b, q)/\tau)}, \qquad p \in \{0, \mathrm{old}\}.$$

Thus, Eq. equation 30 is an item-level propensity ratio, not a ratio of raw token-level likelihoods or sequence probabilities.

Under the absolute-continuity condition

$$e_{\mathrm{old}}(a \mid q) > 0 \quad \Rightarrow \quad e_0(a \mid q) > 0, \tag{31}$$

we have the anchor-level IPS identity

$$\mathbb{E}_{a \sim e_0(\cdot \mid q)} \left[w(a, q) f_y(a, q) \right] = \mathbb{E}_{a \sim e_{\mathrm{old}}(\cdot \mid q)} \left[f_y(a, q) \right]. \tag{32}$$

This identity only concerns the scalar logged-anchor contribution under item-level propensities. It does not imply that the complete ABPO objective is an unbiased off-policy estimator.

Practical self-normalized weighting. In practice, for each logged example i, we use the stopped-gradient item-level propensity ratio

$$w_i^{\log} = \mathrm{sg}\left[\frac{e_{\mathrm{old}}(a_i^{\log} \mid q_i)}{e_0(a_i^{\log} \mid q_i)}\right]. \tag{33}$$

To reduce scale imbalance across feedback types, we self-normalize the weights separately for positive-response and no-response anchors. Let $\mathcal{B}_1$ and $\mathcal{B}_0$ denote the corresponding subsets in a mini-batch. For $i \in \mathcal{B}_y$, define

$$\bar{w}_y^{\log} = \frac{1}{|\mathcal{B}_y|}\sum_{k\in\mathcal{B}_y} w_k^{\log}, \qquad \hat{w}_i^{\log} = \frac{w_i^{\log}}{\bar{w}_y^{\log} + \delta}. \tag{34}$$

The feedback-stratified normalization is used as a practical scale-control device across positive- and no-response anchors, rather than as a claim of unbiased counterfactual estimation within each feedback-conditioned stratum.

The resulting weight is used only in the anchored baseline:

$$\bar{r}_{i,\mathrm{anc}}^{w} = \frac{\hat{w}_i^{\log} r_i^{\log} + \sum_{j=2}^{G} r_i^j}{\hat{w}_i^{\log} + G - 1}. \tag{35}$$

The corresponding weighted standard deviation is

$$\sigma_i = \sqrt{\frac{\hat{w}_i^{\log}\left(r_i^{\log} - \bar{r}_{i,\mathrm{anc}}^{w}\right)^2 + \sum_{j=2}^{G}\left(r_i^j - \bar{r}_{i,\mathrm{anc}}^{w}\right)^2}{\hat{w}_i^{\log} + G - 1}} + \epsilon_{\mathrm{std}}. \tag{36}$$

The advantages entering the GRPO surrogate are computed only for policy-sampled rollouts:

$$A_{i,\mathrm{bandit}}^{j} = \frac{r_i^j - \bar{r}_{i,\mathrm{anc}}^{w}}{\sigma_i}, \qquad j = 2, \ldots, G. \tag{37}$$

There is no term $A_{i,\mathrm{bandit}}^{1}$ in the final surrogate, since the logged anchor is not optimized as a policy-sampled rollout.

Self-normalized anchor-level target. For a fixed feedback stratum y, consider logged examples $i \in \mathcal{B}_y$ and a scalar anchor-level quantity $f_i = f_y(a_i^{\log}, q_i)$. The self-normalized estimator is

$$\hat{\mu}_{\mathrm{SNIPS}}^{(y)} = \frac{\sum_{i\in\mathcal{B}_y} w_i^{\log} f_i}{\sum_{i\in\mathcal{B}_y} w_i^{\log} + |\mathcal{B}_y|\delta}. \tag{38}$$

Under standard integrability assumptions and i.i.d. sampling within the feedback stratum, the weak law of large numbers gives

$$\hat{\mu}_{\mathrm{SNIPS}}^{(y)} \xrightarrow{p} \frac{\mathbb{E}\left[w^{\log} f_y(a^{\log}, q) \mid y\right]}{\mathbb{E}\left[w^{\log} \mid y\right] + \delta}. \tag{39}$$

When $\delta = 0$ and exact item-level propensities are available, this reduces to the usual self-normalized IPS target at the anchor level.

Scope limitation. The SNIPS result above should be interpreted only as an anchor-level exposure-calibration statement based on item-level propensities. Because the full objective also contains policy-sampled non-anchor rollouts, GRPO likelihood ratios, feedback-stratified normalization, and reward normalization within anchor-augmented groups, we do not claim that SNIPS makes the complete ABPO objective an unbiased estimator of a counterfactual bandit objective. Instead, SNIPS is used as a practical, stopped-gradient exposure-correction coefficient for the logged reward's contribution to the group-relative baseline.

F LIMITATIONS

While ABPO consistently improves continual updates for LLM-based recommenders in both offline benchmarks and an online deployment, several limitations remain. First, our offline contextual-bandit feedback is synthetically constructed from public sequential recommendation datasets. Although this protocol allows controlled and reproducible evaluation of exposure-biased feedback, it cannot fully capture all sources of real-world no-response ambiguity, such as user inattention, interface effects, delayed intent, or changing preferences. Second, our offline evaluation follows a candidate-ranking setting rather than full-catalog retrieval. This is a common evaluation protocol for LLM-based recommenders, but future work should further validate ABPO under larger candidate pools, harder negative sampling, and full retrieval-generation pipelines, although our online A/B test uses the full candidate set. Third, our online A/B test is conducted in one production environment. While it provides evidence of practical utility, the exact magnitude of the gains may depend on traffic composition, logging policy, product constraints, and serving infrastructure. Future work will collect real-world feedback over longer periods and carefully examine the effectiveness of ABPO updates based on naturally observed production data.