UniGenRec: Towards Unbiased Generative Recommendation

Anonymous authors
Paper under double-blind review

Abstract

Generative recommendation (GenRec) retrieves items by generating identifiers from user histories; semantic ID (SID)-based systems represent each item as a sequence of tokens. Imbalanced interactions repeatedly reinforce the token paths of popular items, potentially weakening the influence of user history. Balanced SID assignments need not balance interaction frequencies, while global token counts can hide imbalances among the valid continuations of a particular prefix. We propose UniGenRec, which explicitly factors historical frequency at each SID prefix into the training distribution alongside the history-conditioned generator. A gate conditioned on user history and SID prefix controls this factor during supervised fine-tuning; we extend the same factorization to policy optimization. At inference, the frequency branch and gate are removed; only the generator is used. Across three datasets, three SID-based GenRec configurations (TIGER, MiniOneRec, and LC-Rec), and three generative backbones, UniGenRec improves recommendation performance over the corresponding base models and leads the compared methods on MIND, KuaiRec, and Beauty. Recommendation performance is $3.38\times$ the base model on MIND, $2.37\times$ on KuaiRec, and $1.51\times$ on Beauty. It also broadens catalog coverage and reduces head-item amplification relative to training clicks.

1 Introduction

Generative recommendation (GenRec) casts item retrieval as autoregressive generation of item identifiers from a user's interaction history (Rajput et al., 2023; Kong et al., 2025). Applied to video and product recommendation (Deng et al., 2025; Rajput et al., 2023), it can benefit from sequence-model adaptation and reward-based training (Zheng et al., 2024; Deng et al., 2025).

A central design choice is how to represent items as sequences. Semantic ID (SID)-based methods encode each item as a short sequence of discrete tokens derived from semantic representations; related items can thus share tokens (Rajput et al., 2023; Wang et al., 2024). To predict the next item, the generator chooses each SID token conditioned on the user's history and the prefix generated so far, among that prefix's valid continuations.

Token sharing also transmits interaction imbalance: popular items supply more training targets, repeatedly reinforcing the choices along their shared SID prefixes (Fan et al., 2026). A generator that relies too heavily on these frequencies can favor popular continuations across users and weaken the influence of individual histories (Yin et al., 2026; Zheng et al., 2026). We call this overreliance *token-frequency bias*. Figure 1 compares exposure on MIND-small at $K = 50$. MiniOneRec and SASRec (Kang & McAuley, 2018) concentrate recommendations on a few head items, whereas UniGenRec spreads exposure more evenly and stays closer to the dev-click distribution (a–c). UniGenRec also does not further amplify head bias, and its cumulative curve stays closer to the dev-click reference (green) than either baseline (d).

Existing approaches address popularity bias at both the item and token levels. MACR adjusts item scores through counterfactual reasoning, while DICE separates interest from conformity (Wei et al., 2021; Zheng et al., 2021). In SID generation, however, correcting only an item's final score cannot recover a path pruned at an earlier token choice. Within GenRec, CRAB rebalances popular codes and regularizes token relationships, while Ghost combines popularity-aware tokenization with asymmetric unlikelihood training (Fan et al., 2026; Yin et al., 2026). TWMT reweights token losses using

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.

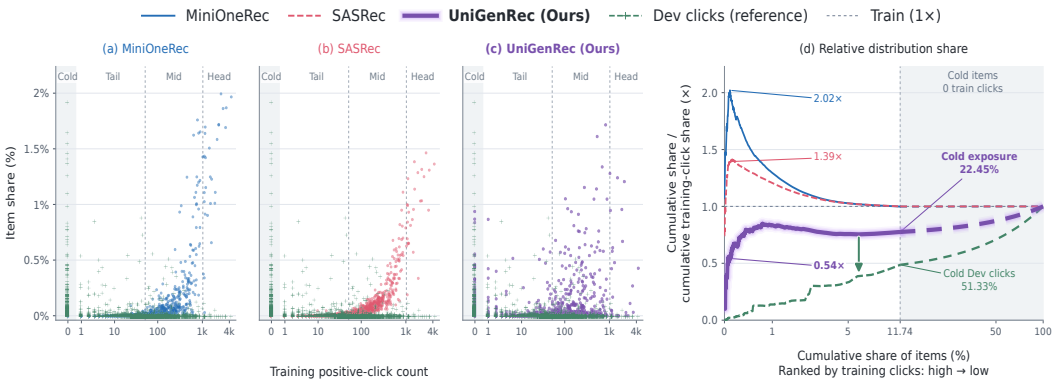

Figure 1: (a–c) UniGenRec spreads exposure more evenly than MiniOneRec and SASRec: it is less concentrated on a few head items and closer to the dev-click distribution. (d) UniGenRec does not amplify head bias relative to training clicks, and its cumulative curve stays closer to the dev-click reference (green) than the other two methods.

semantic information and token frequency, and FSGR combines balanced quantization with layer-specific frequency calibration (Chiu et al., 2026; Zheng et al., 2026).

Balanced code assignments do not ensure balanced training frequencies: popular items contribute more targets, repeating the tokens along their SID paths. Nor do the global or layer-level counts used by TWMT and FSGR necessarily describe the competition after a particular full prefix. The same token can be frequent after one prefix but rare after another; a marginal count can thus hide the imbalance among that prefix's valid continuations.

Keeping the item-to-SID mapping fixed, *UniGenRec* instead treats historical frequency among each prefix's valid continuations as an explicit factor of the training distribution. A separate branch estimates this factor from interactions preceding each target event; a gate conditioned on user history and SID prefix controls its contribution alongside the generator's score. The branch can thus account for some observed imbalance without requiring the generator to reproduce all of it. **UniGenRec-SFT** uses this factorization in supervised training, and **UniGenRec-PO** extends it to the policy update. At inference, the frequency branch and gate are removed: only the generator is used, without frequency-based logit calibration.

We evaluate UniGenRec on MIND-small, KuaiRec-small, and Amazon Reviews 2023 Beauty using three SID-based GenRec configurations: TIGER, MiniOneRec, and LC-Rec. UniGenRec improves full-catalog Recall and nDCG over each base configuration across all three datasets, and leads the compared methods on all four retrieval metrics under each framework on MIND-small, KuaiRec-small, and Beauty. Under MiniOneRec, for example, Recall@50 rises from 0.8215% to 2.7786% on MIND-small and from 1.6176% to 3.8412% on KuaiRec-small (Table 1). Comparing MiniOneRec recommendations with and without observed histories also shows a larger history-dependent gain (Table 2). On MIND-small, two-stage UniGenRec broadens catalog coverage and reduces head-item amplification relative to training clicks (Table 3); these measures do not, by themselves, establish fairness.

Our main contributions are as follows:

- We formulate token-frequency bias among the valid continuations of each SID prefix and explicitly factor their historical frequency into the training distribution, rather than relying on global token counts.
- UniGenRec trains the history-conditioned generator with a context-dependent frequency gate and extends this factorization from supervised training to policy optimization. At inference, the frequency branch and gate are removed, leaving only the generator.
- Across three SID-based GenRec configurations, UniGenRec improves full-catalog retrieval over the base models and leads the compared methods, and also shows larger gains from observed histories and less head-item amplification.

2 Related Work

2.1 Bias in Generative Recommendation

SID-based generative recommendation retrieves items through successive token predictions (Rajput et al., 2023). Shared tokens allow interaction imbalance to accumulate across items: CRAB shows that popular items can reinforce common codes and amplify recommendation concentration (Fan et al., 2026). Ghost further analyzes how sparse positive supervision for tail-item tokens and repeated competition with head-item branches can suppress tail retrieval (Yin et al., 2026). These findings connect item-level popularity bias (disproportionate exposure of popular items) to token-frequency bias (overreliance on frequent choices along the paths to those items). We focus on competition among the valid continuations of each SID prefix, where frequency may outweigh evidence from user history.

2.2 Debiasing in Generative Recommendation

Existing methods intervene in SID construction, generator training, or both. CRAB aggregates item interaction counts into code popularity, splits over-popular codes, and regularizes representations using their hierarchical relationships (Fan et al., 2026). Ghost organizes tail-item SIDs around related head items and uses asymmetric unlikelihood training to penalize inappropriate head-item continuations (Yin et al., 2026). Rebalanced codes can still receive unequal interaction counts within a prefix, and penalties on selected head-item alternatives do not characterize the frequency distribution over all valid continuations.

TWMT combines semantic weights derived from prefix refinement with frequency weights based on the inverse effective number of samples, mixing these objectives through curriculum learning (Chiu et al., 2026). Its semantic weighting uses prefix structure, whereas its frequency weighting uses token counts aggregated over the dataset. FSGR balances semantic quantization and then calibrates prediction logits using empirical token frequencies within each SID layer, with calibration strength varying by depth (Zheng et al., 2026). Such marginal or layer-level statistics can mask differences in a token's relative frequency after different full prefixes. UniGenRec addresses this distinction by treating historical frequency within each prefix's valid continuation set as a factor of the training distribution, carried by a separate branch. The same factor is used in supervised fine-tuning and policy optimization under a fixed SID mapping, and it is not applied as a logit calibration at inference.

2.3 Debiasing in Traditional Recommendation

Popularity debiasing in conventional recommendation aims to distinguish user preference from the effects of imbalanced observations (Chen et al., 2023; Klimashevskaia et al., 2024). Logged feedback is subject to selection bias, so observed clicks do not fully reveal user preference (Schnabel et al., 2016; Saito et al., 2020). Earlier work examines the popularity–accuracy trade-off (Steck, 2011), improves long-tail coverage through ranking regularization (Abdollahpouri et al., 2017), and documents different effects of popularity bias across user groups (Abdollahpouri et al., 2019).

MACR models multiple causes of interaction and removes the estimated direct popularity effect through counterfactual scoring (Wei et al., 2021). DICE learns separate interest and conformity embeddings from cause-specific training pairs (Zheng et al., 2021). PDA removes popularity confounding during training and adjusts scores with desired popularity information at inference, while TIDE separates stable item quality from time-varying conformity (Zhang et al., 2021; Zhao et al., 2023). PP-Rec makes the same split for news recommendation, scoring each click from personalized user interest together with time-aware news popularity (Qi et al., 2021). These methods motivate modeling historical frequency explicitly without discarding its useful information.

These methods score items rather than the intermediate SID decisions. Adjusting a final item score does not specify how frequency should affect a prefix's competing continuations, and a correction applied only after decoding cannot recover paths already pruned. This difference motivates combining factor separation with the token-level perspective of generative debiasing. UniGenRec realizes this combination with a prefix-conditioned frequency component alongside the history-conditioned generator.

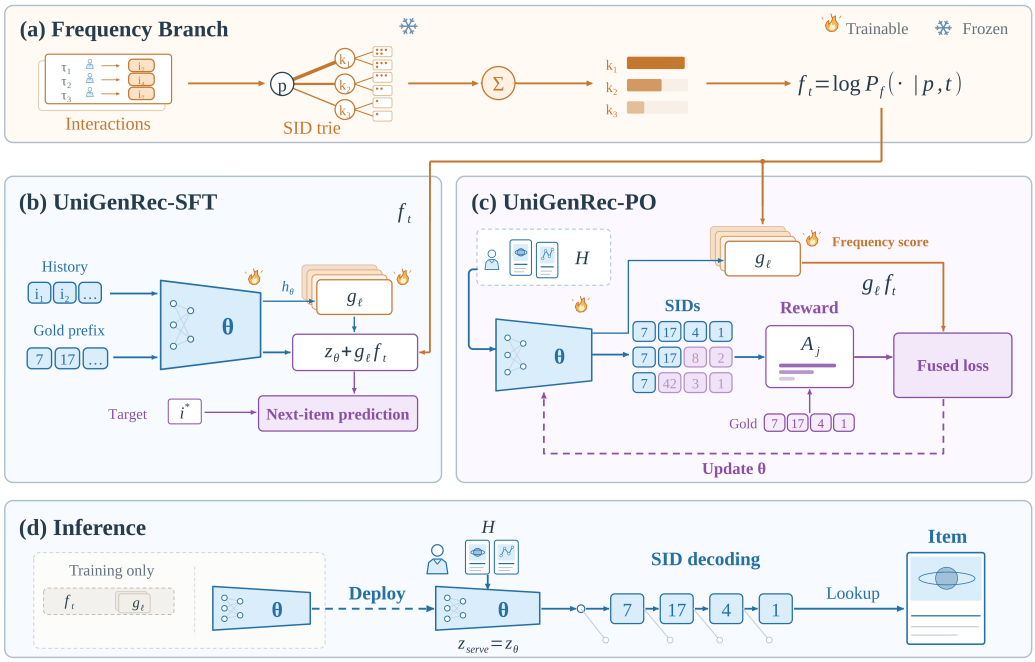

Figure 2: **Overview of UniGenRec.** (a) Frequency branch. (b) UniGenRec-SFT jointly trains the SID-aligned backbone and gates under the factored training distribution. (c) UniGenRec-PO samples SIDs from the backbone alone and updates the backbone and gates through the factored policy. (d) Inference uses the generative model alone.

3 Method

We study next-item recommendation from training events $\mathcal{D}_{\text{tr}} = \{(H_n, y_n, t_n)\}_{n=1}^{N}$, which pair the history H_n before time t_n with the next item y_n (Hidasi et al., 2015; Yuan et al., 2019). Each item i is encoded as an SID $c(i) = (c_{i,1}, \ldots, c_{i,L_i})$ (Rajput et al., 2023). The generator predicts this sequence autoregressively from history H and prefix p. A catalog trie restricts each step to legal continuations $\mathcal{C}(p)$, and the logits z_θ define $\pi_\theta(\cdot \mid H, p) = \text{softmax}_{\mathcal{C}(p)}[z_\theta]$.

UniGenRec makes historical frequency explicit at these prefix-level decisions. The training distribution combines a prefix-level frequency factor with the history-conditioned generator, with a gate setting the factor's weight. The same factor is used in policy optimization, while inference uses the generative model alone (Figure 2).

3.1 Frequency Decomposition

Let $P_f(k \mid p, t)$ be the relative frequency of token k after SID prefix p, estimated from training interactions before time t, and write $f_t(k \mid p) = \log P_f(k \mid p, t)$. This is the frequency factor. In the training distribution it is combined with the generator score; the sum is how the factor enters the likelihood, not a calibration applied at inference. A scalar α_s, shared by all legal next tokens, sets the factor's weight:

$$\widetilde{z}(k \mid s) = z_\theta(k \mid H, p) + \alpha_s f_t(k \mid p), \tag{1}$$

where $s = (H, p, t)$ collects the history, prefix, and event time. The training policy $\pi^R(\cdot \mid s)$ is the softmax of $\widetilde{z}$ over $\mathcal{C}(p)$. The generative model remains conditioned on (H, p); time enters only through f_t. Training computes the loss from the factored policy π^R and updates θ jointly with the gates; inference uses z_θ alone, without the frequency factor. Under a local fit, $\pi_\theta \propto P_{\text{data}}/P_f^{\alpha_s}$ for the observed next-token distribution P_{data}, so the generator need not reproduce the entire frequency imbalance itself (Appendix D). This relation motivates removing the explicit frequency term at inference; frequency information can still be represented in the model's learned parameters.

Frequency branch. Frequency is measured among tokens competing after the same full prefix. For example, token 42 may be frequent after prefix (7) but rare after (12), a distinction that global token counts miss. We therefore aggregate interactions over each legal child's item subtree and normalize within its sibling set. For a target event at time t, we count earlier training interactions with item i, giving each interaction unit weight:

$$\rho_i(t) = \sum_{n=1}^{N} \mathbf{1}[y_n = i,\, t_n < t]. \tag{2}$$

We then normalize the smoothed branch counts over siblings:

$$P_f(k \mid p, t) = \frac{m_t(p,k)}{\sum_{j \in \mathcal{C}(p)} m_t(p,j)}, \qquad m_t(p,k) = \epsilon + \sum_{i:\, c(i) \succeq p \oplus k} \rho_i(t). \tag{3}$$

Here $p \oplus k$ appends k to p, and $c(i) \succeq p \oplus k$ selects items below that edge. We use $\epsilon = 1$, so every legal branch remains defined and the distribution is uniform when no earlier interactions are present. Time determines which training interactions enter the counts; all included interactions contribute equally.

3.2 Supervised Fine-Tuning (UniGenRec-SFT)

The information provided by user history and the frequency imbalance among continuations vary across prefixes, so UniGenRec-SFT uses a context-dependent gate to control the frequency contribution. At SID depth ℓ, a gate $g_{\phi_\ell}(H,p) \in (0,1)$ reads the hidden state $\mathbf{h}_\theta(H,p)$ immediately before the SID token and produces a coefficient shared by all legal continuations. Setting $\alpha_s = g_{\phi_\ell}(H,p)$ in Equation (1) yields the training distribution $\pi^R_{\theta,\phi}$. Appendix D gives the gate parameterization.

Training objective. Fine-tuning also supervises alignment, formatting, and EOS tokens. Frequency decomposition changes only the relative probabilities inside a legal SID continuation set. Let P_θ be the full-vocabulary distribution and $\mathcal{R}$ the SID-routing positions among N_{tok} supervised tokens. At a routing position we keep the mass on the legal set and replace only the within-set distribution by $\pi^R_{\theta,\phi}$; other positions keep P_θ:

$$\mathcal{L}_{\text{UniGenRec-SFT}} = -\frac{1}{N_{\text{tok}}} \sum_{q=1}^{N_{\text{tok}}} \log P_{\text{train}}(y_q \mid s_q), \tag{4}$$

where $P_{\text{train}}(y_q \mid s_q) = P_\theta\big(\mathcal{C}(p_q) \mid s_q\big)\, \pi^R_{\theta,\phi}(y_q \mid s_q)$ if $q \in \mathcal{R}$, and $P_\theta(y_q \mid s_q)$ otherwise.

Optimization. SID alignment first adapts the pretrained generative model to the SID vocabulary, without the frequency branch. UniGenRec-SFT then trains the generative model and the depth-specific gates jointly for the same number of epochs as the other methods. The target likelihood is computed from the full score $\widetilde{z}$. Gradients update θ and pass through g_{ϕ_ℓ}; they are stopped at f_t, which is a count rather than a parameter. The resulting checkpoint can be served directly and initializes UniGenRec-PO.

3.3 Policy Optimization (UniGenRec-PO)

UniGenRec-PO continues from the UniGenRec-SFT checkpoint to improve recommendation accuracy through reward-based learning. It retains GRPO's group-relative advantages and clipped policy updates (Shao et al., 2024), while carrying the frequency decomposition into the policy update. Candidate rewards combine SID-prefix matches with item-level feedback and are normalized within each group to obtain path advantages A_j.

Updating under the factored policy. Standard GRPO forms its policy ratio from the generator's probabilities. UniGenRec-PO instead uses the factored probabilities from Equation (1) for both the current and old policies. The old policy is a frozen snapshot $(\theta_{\text{old}}, \phi_{\text{old}})$ taken before the update, so the denominator does not use the current gate. For token $k_{j,\ell}$ at context $s_{j,\ell}$ on candidate path j, the

ratio becomes

$$r_{j,\ell}(\theta,\phi)=\frac{\pi^R_{\theta,\phi}(k_{j,\ell}\mid s_{j,\ell})}{\pi^R_{\theta_{\text{old}},\phi_{\text{old}}}(k_{j,\ell}\mid s_{j,\ell})}. \tag{5}$$

The reward-weighted update therefore acts on the factored training distribution, which includes the historical frequency factor. The current step still updates the gates together with θ; only the snapshot in the denominator is held fixed. Gradients enter g_{ϕ_ℓ} as well as θ, and remain stopped at f_t and at $(\theta_{\text{old}},\phi_{\text{old}})$. This update allows the gate to represent historical frequency alongside the generator, but does not enforce a unique division of information between them.

Sampling and regularizing the serving model. Candidates are still drawn from the generator alone by trie-constrained sampled-beam decoding, so rewards evaluate the model used at inference. Sampling from the factored policy would steer those rollouts toward high-frequency continuations and let popular items dominate the group; drawing from the generator alone keeps that factor out of the candidate set. Because sampling and updating use different policies, we introduce a weight $w_{j,\ell}$: the ratio of the snapshot factored probability $\pi^R_{\theta_{\text{old}},\phi_{\text{old}}}$ to the sampling-policy probability $\pi_{\theta_{\text{old}}}$, capped and treated as stop-gradient. This provides a local approximation rather than an exact correction for the coupled sampled-beam distribution. The resulting objective is

$$\mathcal{L}_{\text{UniGenRec-PO}}=-\mathbb{E}_{j,\ell}[w_{j,\ell}\,\ell_{\text{clip}}(r_{j,\ell},A_j)]+\beta_{\text{KL}}\mathcal{L}_{\text{KL}}, \tag{6}$$

where ℓ_{clip} is the standard GRPO clipped surrogate and the expectation averages tokens within each path, then paths within the group. The KL term anchors the generator alone to the frozen UniGenRec-SFT reference, without the frequency branch.

Reward. A snapshot of the generator draws G complete legal SIDs without the frequency branch, and c^* is the gold SID. Each candidate is credited for every gold prefix it matches, then receives one item-level term: a hit bonus if it is the gold SID, or a rank penalty on the other candidates when the gold SID is in the group.

$$R_j=\sum_{\ell=1}^{L^*}\mathbf{1}\left[c^{(j)}_{1:\ell}=c^*_{1:\ell}\right]+R_{\text{item},j}. \tag{7}$$

The two item-level outcomes never apply to the same candidate. Rewards are standardized within the group to path advantages A_j, which enter Equation (6). Appendix E gives the item-term cases, sampling weights, clipped surrogate, KL term, and optimization settings.

3.4 Inference

After UniGenRec-SFT or UniGenRec-PO, we drop the frequency branch and the gates. Trie-constrained decoding uses z_θ, excludes items already in the history, and ranks legal SIDs by mean token log-probability:

$$s_\theta(i\mid H)=\frac{1}{L_i}\sum_{\ell=1}^{L_i}\log\pi_\theta(c_{i,\ell}\mid H,c_{i,<\ell}), \tag{8}$$

The resulting SIDs are mapped to items using the original codebook.

4 Experiments

We report full-catalog retrieval, history-sensitivity and exposure diagnostics under MiniOneRec, and ablations of the training stages and frequency gate.

4.1 Experimental Setup

We evaluate on MIND-small (MIND) (Wu et al., 2020), Amazon Reviews 2023 Beauty (Beauty) (Hou et al., 2026), and KuaiRec-small (KuaiRec) (Gao et al., 2022), retrieving from the retained full catalog. Throughout the paper, MIND and KuaiRec refer to these small versions. Appendix H (Table 10) reports Amazon Reviews 2023 Industrial & Scientific (Hou et al., 2026) under its official name and protocol. The main comparison uses Qwen2.5-1.5B-Instruct (Qwen Team,

Table 1: Full-catalog retrieval under the Qwen2.5-1.5B-Instruct protocol. The UniGenRec rows report UniGenRec-SFT for TIGER and LC-Rec, and UniGenRec-SFT followed by UniGenRec-PO for MiniOneRec. R and N are Recall and nDCG (%); bold/underline mark best and second-best.

	MIND				KuaiRec				Beauty			
Setting	R@10	R@50	N@10	N@50	R@10	R@50	N@10	N@50	R@10	R@50	N@10	N@50
TIGER	0.1019	0.5835	0.0446	0.1712	0.2911	1.3408	0.1063	0.3168	1.4255	1.9957	1.2067	1.3330
+ CRAB	0.3621	0.9216	0.1231	0.2418	0.6871	1.4506	0.3768	0.5286	1.5681	2.2096	1.3486	1.4959
+ Ghost	0.2788	0.7845	0.0974	0.1986	0.5631	1.4983	0.2920	0.5049	1.4968	2.1383	1.2872	1.4319
+ TWMT	0.3140	0.8428	0.1128	0.2164	0.7587	1.3122	0.4097	0.5361	1.6393	2.2808	1.4098	1.5532
+ FSGR	0.3520	1.0290	0.1247	0.2651	0.6728	1.5603	0.3684	0.5412	1.7106	2.3521	1.4711	1.6263
+ UniGenRec	**0.9410**	**2.3868**	**0.5128**	**0.8516**	**0.9830**	**3.2161**	**0.7284**	**1.1528**	**2.4234**	**3.8489**	**1.8028**	**2.1375**
MiniOneRec	0.1241	0.8215	0.0595	0.2038	0.3722	1.6176	0.1476	0.4281	2.0670	2.9223	1.7711	1.9649
+ CRAB	0.1797	1.0475	0.0880	0.2682	0.8637	1.8371	0.4730	0.7010	2.2808	3.3500	1.8273	2.0849
+ Ghost	0.1436	0.8799	0.0811	0.2304	0.1097	0.2911	0.0602	0.0971	2.0670	2.9936	1.7691	1.9809
+ TWMT	0.1362	0.8799	0.0637	0.2162	0.4151	1.6510	0.1676	0.4455	2.1383	3.1361	1.7894	2.0083
+ FSGR	0.1778	1.1689	0.0911	0.2984	0.8494	1.9230	0.4688	0.7184	2.2808	3.4925	1.8273	2.1230
+ UniGenRec	**1.1096**	**2.7786**	**0.5705**	**0.9242**	**1.1738**	**3.8412**	**0.8047**	**1.3452**	**2.8510**	**4.4191**	**1.9982**	**2.3631**
LC-Rec	0.2529	1.1494	0.1302	0.3174	0.4342	1.7464	0.1638	0.4524	2.1383	4.7755	1.5189	2.0948
+ TWMT	0.2714	1.1967	0.1326	0.3246	0.4581	1.8132	0.1846	0.4718	2.2096	4.8468	1.5361	2.1255
+ UniGenRec	**1.1170**	**4.7014**	**0.4788**	**1.2245**	**1.2931**	**2.6387**	**0.6592**	**0.9395**	**2.8510**	**7.4840**	**1.7223**	**2.7863**

Note. CRAB, Ghost, and FSGR are omitted from the LC-Rec block; they conflict with USM indexing.

2024). MiniOneRec replacements with Qwen2.5-0.5B and LLaMA-3-8B (Grattafiori et al., 2024) are reported in Appendix I.

We compare UniGenRec with the base configuration and applicable CRAB, Ghost, TWMT, and FSGR variants under TIGER, MiniOneRec, and LC-Rec (Table 1). Baselines and training stages are described in Appendix B, and implementation details in Appendices C–E.

4.2 Overall Recommendation Performance

Recall@K and nDCG@K are percentages for $K \in \{10, 50\}$, with EmpGap@K and PerLift@K under MiniOneRec in Section 4.3. Main tables average three random seeds; data construction and the MostPop protocol are in Appendix F. UniGenRec improves all four retrieval metrics over the corresponding base configuration in each SID framework (Table 1). On MIND, KuaiRec, and Beauty it also leads the other methods on all four metrics. For MiniOneRec, Recall@50 rises from 0.8215% to 2.7786% on MIND, from 1.6176% to 3.8412% on KuaiRec, and from 2.9223% to 4.4191% on Beauty; with LC-Rec, MIND Recall@50 rises from 1.1494% to 4.7014%.

4.3 Improved Personalization

Table 2 reports EmpGap@K (observed minus empty history) and PerLift@K (Recall minus MostPop) under MiniOneRec.

Table 2: EmpGap and PerLift under MiniOneRec.

	EmpGap: Recall(history) − Recall(empty)						*PerLift:* Recall − MostPop					
	MIND		KuaiRec		Beauty		MIND		KuaiRec		Beauty	
Setting	@10	@50	@10	@50	@10	@50	@10	@50	@10	@50	@10	@50
MiniOneRec	0.0306	0.4724	0.3626	1.5699	0.3326	0.7365	0.0676	0.3390	0.2863	−0.5153	1.3542	1.2830
+ CRAB	0.1519	0.5289	0.8589	1.6653	0.6890	1.2830	0.1232	0.5650	0.7778	−0.2958	1.5681	1.7106
+ Ghost	0.0519	0.3696	0.1097	0.2481	0.3564	0.7128	0.0871	0.3973	0.0239	−1.8419	1.3542	1.3542
+ TWMT	−0.1232	0.5363	0.4056	1.5842	0.4514	0.9266	0.0797	0.3973	0.3292	−0.4819	1.4255	1.4968
+ FSGR	0.1186	1.1022	0.1622	0.2100	0.6415	1.4968	0.1213	0.6863	0.7635	−0.2100	1.5681	1.8532
+ UniGenRec	**0.8966**	**2.4868**	**1.1643**	**1.9802**	**1.7819**	**2.9223**	**1.0531**	**2.2961**	**1.0879**	**1.7083**	**2.1383**	**2.7798**
Rel.	29.30×	5.26×	3.21×	1.26×	5.36×	3.97×	15.58×	6.77×	3.80×	—	1.58×	2.17×

Positive EmpGap means the history raises recall; positive PerLift means the list beats global popularity. The Rel. row is UniGenRec divided by MiniOneRec when the denominator is positive. Differences and ratios are computed before rounding.

4.4 Bias Mitigation Effectiveness

Table 3: Coverage and exposure on MIND at $K = 50$, averaged over three seeds as in Table 1. UniGenRec is the two-stage MiniOneRec checkpoint, UniGenRec-SFT followed by UniGenRec-PO. The other rows are the corresponding MiniOneRec runs after standard GRPO.

Method	Coverage (%) ↑	HitCov (%) ↑	InterDiv ↑	MGU_{tr} ↓	Amp_{tr}
MiniOneRec	3.3385	5.8958	0.6849	0.2206	10.82×
CRAB	1.5589	4.1133	0.6819	0.2287	14.94×
Ghost	4.1954	4.2048	0.6539	0.2295	15.51×
TWMT	1.9528	3.1079	0.5978	0.1813	4.49×
UniGenRec	**8.5686**	**7.7697**	**0.8446**	**0.1241**	0.56×

On MIND at $K = 50$, two-stage UniGenRec is the only compared method whose head amplification falls below $1\times$, and it has the highest catalog coverage and hit-item coverage (Table 3). CRAB and Ghost amplify the head more than MiniOneRec; TWMT amplifies less but stays above $1\times$. Metric definitions are in Appendix G.

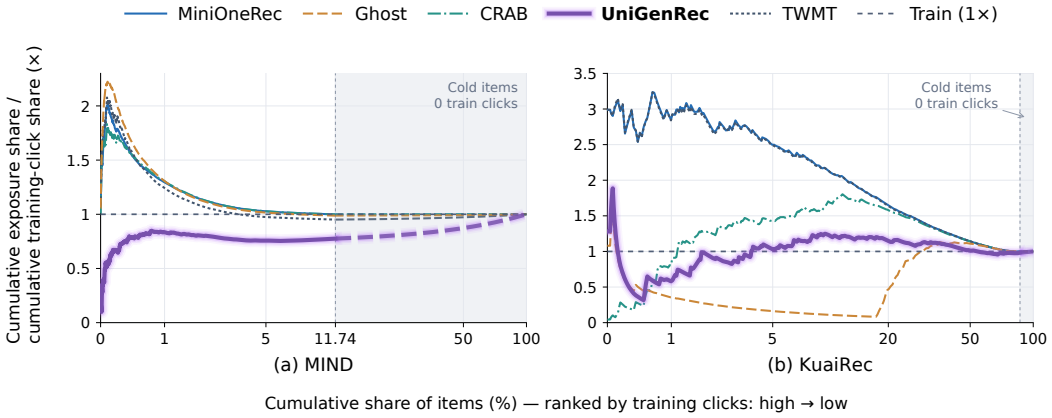

Figure 3: Training-relative exposure under MiniOneRec. Items are ordered from most to least clicked in training. The dashed line is the training distribution, and the panels use separate vertical scales.

Figure 3 compares each method with the training distribution: a curve above the dashed $1\times$ line at the head indicates additional head exposure. On MIND, UniGenRec stays below $1\times$ until the cold tail, while the baselines open near $2\times$. On KuaiRec the split is larger: MiniOneRec and TWMT stay near $3\times$, whereas UniGenRec tracks the training reference except at the extreme head. The frequency factor enters the training distribution, while serving uses z_θ alone. Under a local fit, $\pi_\theta \propto P_{\text{data}}/P_f^{\alpha_s}$, so the explicit factor can account for part of the training imbalance. The observed exposure and PerLift differences (Table 2) are consistent with this mechanism but do not determine how much frequency information remains in the served generator.

4.5 Ablation Study

Stage ablation. MiniOneRec-SFT and UniGenRec-SFT start from the same SID-aligned model (Table 4). UniGenRec-SFT improves all four metrics over MiniOneRec-SFT on both datasets. The SFT pair isolates the frequency factorization before policy optimization; on KuaiRec, the PO stage further raises Recall@50 from 2.5624% to 3.8412%. From that checkpoint, UniGenRec-PO beats standard GRPO on all four KuaiRec metrics and on every MIND metric except Recall@50 (2.7786% versus 2.8079%).

Table 4: Training-stage ablation on MiniOneRec. R and N follow Table 1. Bold marks the best value per column.

	MIND				KuaiRec			
Setting	R@10	R@50	N@10	N@50	R@10	R@50	N@10	N@50
MiniOneRec-SFT	0.1074	0.8864	0.0516	0.2249	0.3483	1.7242	0.1312	0.4578
UniGenRec-SFT	0.9762	2.7860	0.4937	0.8769	1.1468	2.5624	0.6366	0.9158
MiniOneRec-SFT + GRPO	0.1241	0.8215	0.0595	0.2038	0.3722	1.6176	0.1476	0.4281
UniGenRec-SFT + GRPO	1.0290	**2.8079**	0.5317	0.9186	1.1643	2.3047	0.8027	1.0355
UniGenRec-SFT + UniGenRec-PO	**1.1096**	2.7786	**0.5705**	**0.9242**	**1.1738**	**3.8412**	**0.8047**	**1.3452**

Coefficient ablation. Table 5 asks whether the frequency step has to depend on the user history and on the prefix that history is extending. The MiniOneRec-SFT row is the backbone-only baseline from Table 4. Global fixed uses one coefficient for every user and every prefix. Gate w/o history keeps the prefix but drops H; Gate w/o prefix keeps the history but drops p. UniGenRec-SFT is the default gate $\alpha_s = g_{\phi_\ell}(H, p)$.

Table 5: Frequency-coefficient ablation under MiniOneRec. R and N follow Table 1. Bold marks UniGenRec-SFT.

	MIND				KuaiRec			
Setting	R@10	R@50	N@10	N@50	R@10	R@50	N@10	N@50
MiniOneRec-SFT	0.1074	0.8864	0.0516	0.2249	0.3483	1.7242	0.1312	0.4578
Global fixed ($g = 0.5$)	0.3640	1.1781	0.1735	0.3482	0.6776	2.0136	0.2528	0.5459
Gate w/o history	0.8558	2.6749	0.4212	0.8029	0.9830	2.2618	0.5858	0.8611
Gate w/o prefix	0.8632	2.6665	0.4227	0.8023	0.9877	2.3572	0.5718	0.8779
UniGenRec-SFT	**0.9762**	**2.7860**	**0.4937**	**0.8769**	**1.1468**	**2.5624**	**0.6366**	**0.9158**

A fixed coefficient improves on the backbone, but cannot adapt to the user or the current prefix. Adding either context improves Recall@10 further; the full gate $g_{\phi_\ell}(H, p)$ performs best on all four reported metrics for both datasets.

5 Conclusion

This paper presented UniGenRec, which treats token-frequency bias as an explicit factor of the training distribution over the continuations that compete at each SID prefix. A frequency branch estimates that factor from preceding interactions, and a gate controls its contribution alongside the generator. UniGenRec-SFT trains under this factored distribution, and UniGenRec-PO extends it to the policy update. At inference, the frequency branch and gates are removed; the generator serves alone.

Across TIGER, MiniOneRec, and LC-Rec, this training improves full-catalog Recall and nDCG on MIND-small, KuaiRec-small, and Beauty (Hou et al., 2026). On MIND-small, KuaiRec-small, and Beauty it also leads the compared methods on all four retrieval metrics. Under MiniOneRec, it shows larger gains from observed histories, while two-stage UniGenRec on MIND-small broadens catalog coverage and reduces head-item amplification relative to training clicks. Adapting the frequency branch to items that never appear in training remains future work.

AI use statement

Generative AI tools assisted literature retrieval, organization, language editing, and LaTeX; the authors verified the sources, mathematics, results, and final text.

Ethics statement

This work studies aggregate frequency effects. Lower exposure concentration does not by itself establish fairness, so quality and distributional effects should be assessed together.

Reproducibility statement

The appendices specify the model, SID construction, training, frequency estimation, and evaluation. We will release the configurations and diagnostic scripts.

References

Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. Controlling popularity bias in learning-to-rank recommendation. In *Proceedings of the eleventh ACM conference on recommender systems*, pp. 42–46, 2017.

Himan Abdollahpouri, Masoud Mansoury, Robin Burke, and Bamshad Mobasher. The unfairness of popularity bias in recommendation. *arXiv preprint arXiv:1907.13286*, 2019.

Allison JB Chaney, Brandon M Stewart, and Barbara E Engelhardt. How algorithmic confounding in recommendation systems increases homogeneity and decreases utility. In *Proceedings of the 12th ACM conference on recommender systems*, pp. 224–232, 2018.

Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He. Bias and debias in recommender system: A survey and future directions. *ACM Transactions on Information Systems*, 41(3):1–39, 2023.

Wei-Ning Chiu, Chuan-Ju Wang, and Pu-Jen Cheng. Token-weighted multi-target learning for generative recommenders with curriculum learning. *arXiv preprint arXiv:2601.17787v1*, 2026.

Jiaxin Deng, Shiyao Wang, Kuo Cai, Lejian Ren, Qigen Hu, Weifeng Ding, Qiang Luo, and Guorui Zhou. Onerec: Unifying retrieve and rank with generative recommender and iterative preference alignment. *arXiv preprint arXiv:2502.18965*, 2025.

Zezhong Fan, Ziheng Chen, Luyi Ma, Jin Huang, Lalitesh Morishetti, Kaushiki Nag, Sushant Kumar, and Kannan Achan. Crab: Codebook rebalancing for bias mitigation in generative recommendation. *arXiv preprint arXiv:2604.05113*, 2026.

Chongming Gao, Shijun Li, Wenqiang Lei, Jiawei Chen, Biao Li, Peng Jiang, Xiangnan He, Jiaxin Mao, and Tat-Seng Chua. Kuairec: A fully-observed dataset and insights for evaluating recommender systems. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pp. 540–550, 2022.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

Yupeng Hou, Jiacheng Li, Xiangjun Fu, Zhankui He, An Yan, Xiusi Chen, and Julian McAuley. Bridging language and items for retrieval and recommendation: Benchmarking llms as semantic encoders. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3251–3265, 2026.

Meng Jiang, Keqin Bao, Jizhi Zhang, Wenjie Wang, Zhengyi Yang, Fuli Feng, and Xiangnan He. Item-side fairness of large language model-based recommendation system. In *Proceedings of the ACM web conference 2024*, pp. 4717–4726, 2024.

Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*, pp. 197–206. IEEE, 2018.

Anastasiia Klimashevskaia, Dietmar Jannach, Mehdi Elahi, and Christoph Trattner. A survey on popularity bias in recommender systems. *User Modeling and User-Adapted Interaction*, 34(5): 1777–1834, 2024.

Xiaoyu Kong, Leheng Sheng, Junfei Tan, Yuxin Chen, Jiancan Wu, An Zhang, Xiang Wang, and Xiangnan He. Minionerec: An open-source framework for scaling generative recommendation. *arXiv preprint arXiv:2510.24431*, 2025.

Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11523–11532, 2022.

Masoud Mansoury, Himan Abdollahpouri, Mykola Pechenizkiy, Bamshad Mobasher, and Robin Burke. Fairmatch: A graph-based approach for improving aggregate diversity in recommender systems. In *Proceedings of the 28th ACM conference on user modeling, adaptation and personalization*, pp. 154–162, 2020.

Tao Qi, Fangzhao Wu, Chuhan Wu, and Yongfeng Huang. Pp-rec: News recommendation with personalized user interest and time-aware news popularity. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 5457–5467, 2021.

Qwen Team. Qwen2.5 technical report, 2024.

Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Tran, Jonah Samost, et al. Recommender systems with generative retrieval. *Advances in Neural Information Processing Systems*, 36:10299–10315, 2023.

Yuta Saito, Suguru Yaginuma, Yuta Nishino, Hayato Sakata, and Kazuhide Nakata. Unbiased recommender learning from missing-not-at-random implicit feedback. In *Proceedings of the 13th international conference on web search and data mining*, pp. 501–509, 2020.

Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. Recommendations as treatments: Debiasing learning and evaluation. In *international conference on machine learning*, pp. 1670–1679. PMLR, 2016.

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

Harald Steck. Item popularity and recommendation accuracy. In *Proceedings of the fifth ACM conference on Recommender systems*, pp. 125–132, 2011.

Wenjie Wang, Honghui Bao, Xinyu Lin, Jizhi Zhang, Yongqi Li, Fuli Feng, See-Kiong Ng, and Tat-Seng Chua. Learnable item tokenization for generative recommendation. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pp. 2400–2409, 2024.

Tianxin Wei, Fuli Feng, Jiawei Chen, Ziwei Wu, Jinfeng Yi, and Xiangnan He. Model-agnostic counterfactual reasoning for eliminating popularity bias in recommender system. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pp. 1791–1800, 2021.

Fangzhao Wu, Ying Qiao, Jiun-Hung Chen, Chuhan Wu, Tao Qi, Jianxun Lian, Danyang Liu, Xing Xie, Jianfeng Gao, Winnie Wu, et al. Mind: A large-scale dataset for news recommendation. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pp. 3597–3606, 2020.

Jun Yin, Bangguo Zhu, Peng Huo, Ruochen Liu, Hao Chen, Senzhang Wang, Shirui Pan, and Chengqi Zhang. Echoes in filter bubble: Diagnosing and curing popularity bias in generative recommenders. *arXiv preprint arXiv:2605.16825*, 2026.

Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M Jose, and Xiangnan He. A simple convolutional generative network for next item recommendation. In *Proceedings of the twelfth ACM international conference on web search and data mining*, pp. 582–590, 2019.

Yang Zhang, Fuli Feng, Xiangnan He, Tianxin Wei, Chonggang Song, Guohui Ling, and Yongdong Zhang. Causal intervention for leveraging popularity bias in recommendation. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, pp. 11–20, 2021.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025.

Zihao Zhao, Jiawei Chen, Sheng Zhou, Xiangnan He, Xuezhi Cao, Fuzheng Zhang, and Wei Wu. Popularity bias is not always evil: Disentangling benign and harmful bias for recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 35(10):9920–9931, 2023.

Bowen Zheng, Yupeng Hou, Hongyu Lu, Yu Chen, Wayne Xin Zhao, Ming Chen, and Ji-Rong Wen. Adapting large language models by integrating collaborative semantics for recommendation. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pp. 1435–1448. IEEE, 2024.

Yu Zheng, Chen Gao, Xiang Li, Xiangnan He, Yong Li, and Depeng Jin. Disentangling user interest and conformity for recommendation with causal embedding. In *Proceedings of the web conference 2021*, pp. 2980–2991, 2021.

Yuchen Zheng, Sihan Xu, Jingwen Yang, Xiangrui Cai, Haiwei Zhang, and Xiaojie Yuan. Fsgr: Mitigating token frequency bias for fair sid-based generative recommendation. *arXiv preprint arXiv:2608.12845*, 2026.

A Notation

Table 6: Principal notation.

Symbol	Meaning
$\mathcal{D}_{\text{tr}}, (H, y, t)$	Training records; history, target, event time
$c(i), L_i, p, \mathcal{C}(p)$	SID, SID length, prefix, legal children
$z_\theta, P_\theta, \pi_\theta$	Logits, full-vocabulary policy, routing policy
$\rho_i(t)$	Number of training interactions with item i before time t
m_t, P_f, f_t	Smoothed branch count, historical frequency distribution, log-frequency
$s, s_q, s_{j,\ell}$	History–prefix–time state; supervised-token context; state at depth ℓ of rollout j
$k_{j,\ell}, L_j$	Generated SID token and path length (excluding EOS and padding)
α_s, g_{ϕ_ℓ}	Frequency coefficient and depth-specific gate
$\mathcal{R}, N_{\text{tok}}$	SID-routing positions and number of supervised tokens
$\pi^R, \pi_{\text{ref}}, G$	Frequency-adjusted training policy, frozen SFT reference policy, group size
$(\theta_{\text{old}}, \phi_{\text{old}})$	Frozen GRPO snapshot of the generator and gates
$R_j, A_j, w_{j,\ell}, r_{j,\ell}$	Path reward, advantage, generation-to-decomposed weight, policy ratio
ℓ_{clip}	Standard GRPO clipped surrogate
$\delta, w_{\max}, \beta_{\text{KL}}$	PPO clipping threshold, sampling-weight cap, KL coefficient
PerLift@K	Recall gain over a most-popular recommender at cutoff K (percentage points)

B Comparison Details

This appendix describes the compared methods, their shared implementation, and the scope of each comparison.

Base configurations. Each block in Table 1 instantiates a recommendation recipe with the common Qwen2.5-1.5B-Instruct generative model under the same data and full-catalog evaluation protocol. Some implementation settings differ from those in the original papers. Base denotes the corresponding configuration without an added frequency-aware intervention. Using a shared backbone limits architectural differences, helping isolate the representation and training components under study. TIGER constructs SIDs through residual quantization and autoregressively predicts the target item's SID (Rajput et al., 2023; Lee et al., 2022). MiniOneRec combines SID-based supervised learning, language–SID alignment, and recommendation-oriented GRPO (Kong et al., 2025; Shao et al., 2024). LC-Rec combines quantized item indexing with auxiliary alignment tasks to integrate language and collaborative semantics (Zheng et al., 2024).

Frequency-aware baselines. We compare UniGenRec with four frequency-aware methods that modify SID construction, training, or both. CRAB splits frequently used codes and introduces hierarchical semantic regularization (Fan et al., 2026). Ghost combines popularity-aware item tokenization with asymmetric unlikelihood training to address competition between head and tail items (Yin et al., 2026). TWMT combines semantic-information and token-frequency weighting through curriculum learning (Chiu et al., 2026). FSGR couples balanced semantic quantization with depth-dependent log-frequency calibration (Zheng et al., 2026).

Training stages. Every setting in Table 1 uses one epoch of SID alignment and four epochs of SFT. UniGenRec-SFT uses those same four epochs to train the generative model and the gates jointly, with no extra epoch that fits the gate alone. Policy optimization follows the original recipe of each framework and is not added to TIGER or LC-Rec. The MiniOneRec block uses two epochs of policy optimization with 32 sampled-beam candidates: standard GRPO for Base, CRAB, Ghost, TWMT, and FSGR, and UniGenRec-PO for UniGenRec. Both policy-optimization variants use learning rate 5×10^{-7}. The TIGER and LC-Rec blocks stop after SFT; their UniGenRec rows are UniGenRec-SFT checkpoints, without UniGenRec-PO. The MiniOneRec UniGenRec row reports UniGenRec-SFT followed by UniGenRec-PO; UniGenRec-SFT alone is also reported in the training-stage comparison. Standard GRPO updates the serving generative model without the frequency branch; UniGenRec-PO uses the factored policy update described in Section 3.3. The TWMT baseline uses the original v1 formulation during SFT (Chiu et al., 2026).

SID mappings and comparison scope. Within a SID recipe, Base, TWMT, and UniGenRec share one item-to-SID map, vocabulary, and initialization. Appendix C describes only this shared TIGER/MiniOneRec RQ-VAE map. LC-Rec, CRAB, Ghost, and FSGR each use their own original item-to-SID procedure and are compared as complete method packages rather than as drop-in training add-ons on the shared map. CRAB, Ghost, and FSGR are omitted from the LC-Rec block because those index rewrites are incompatible with LC-Rec indexing. All retrieval scores use the evaluation protocol in Appendix F.

SASRec reference. SASRec (Kang & McAuley, 2018) is a self-attentive sequential scorer. We retain it as a conventional sequential reference in Figure 1 and Appendix G; it is not a SID generator under the shared 1.5B protocol.

C SID Construction and Frequency Branch

This section specifies the SID map shared by Base, TWMT, and UniGenRec under TIGER and MiniOneRec. Item titles and descriptions are encoded with Qwen3-Embedding-4B (Zhang et al., 2025) and quantized by RQ-VAE (Lee et al., 2022). RQ-VAE is trained on catalog embeddings rather than user sequences. We use three residual levels with codebook size 256 and latent width 32. Training uses Adam, learning rate 10^{-3}, and batch size 20,480 on a single GPU. The resulting SID vocabulary is added to Qwen2.5-Instruct; a deterministic fourth token resolves collisions, giving SIDs of length $L = 4$. For the fixed-catalog utility setting, content from the catalog may be used to construct the codebook, but held-out evaluation interaction labels may not. LC-Rec, CRAB, Ghost, and FSGR each use their own original item-to-SID procedure and do not reuse this RQ-VAE map.

SID alignment adapts the pretrained generative model to a fixed item-to-SID map. Its supervision comes from catalog metadata paired with that map; it does not use user interaction sequences. Each catalog item contributes two examples: title-to-SID and the reverse SID-to-title pair. The text is the official news title for MIND; the video caption, with cover-text and category fallbacks, for KuaiRec; and the official product title for Beauty and Industrial (Hou et al., 2026).

The frequency branch has no trainable parameters. Only user–item interaction records in the training set update $\rho_i(t)$ and $m_t(p, k)$; evaluation interactions never update these statistics. Each earlier training interaction contributes one count on all datasets. Same-timestamp events are scored before their counts are added, and $\epsilon = 1$ is applied to each legal child.

D UniGenRec-SFT Training Details

Gate architecture. At SID depth ℓ, the gate maps the hidden state $\mathbf{h}_\theta(H, p)$ to a scalar:

$$g_{\phi_\ell}(H, p) = \sigma\Big(\mathbf{v}_\ell^\top \tanh\big[\mathbf{W}_\ell \,\mathrm{LN}\big(\mathbf{h}_\theta(H, p)\big) + \mathbf{e}_\ell\big] + a_\ell\Big). \tag{9}$$

The parameters $\mathbf{W}_\ell$, $\mathbf{v}_\ell$, and a_ℓ are independent across depths; $\mathbf{e}_\ell$ is a depth embedding. LayerNorm has no trainable affine parameters, and the hidden width is 128.

Equivalent implementation of the prediction loss. For a routing position r, write $\mathcal{C}_r = \mathcal{C}(p_r)$, and let z_r and $\widetilde{z}_r$ denote the generative model's logits before and after frequency adjustment, respectively. The target lies in $\mathcal{C}_r$, so Equation (4) gives

$$\begin{aligned} \ell_r^{\text{UniGenRec}} &= -\log P_\theta(\mathcal{C}_r \mid s_r) - \log \pi^R_{\theta,\phi}(y_r \mid s_r) \\ &= -\log P_\theta(y_r \mid s_r) + \mathrm{CE}_{\mathcal{C}_r}(\widetilde{z}_r, y_r) - \mathrm{CE}_{\mathcal{C}_r}(z_r, y_r). \end{aligned} \tag{10}$$

Here $\mathrm{CE}_{\mathcal{C}_r}$ is cross-entropy normalized only over the legal children. This correction replaces the original within-set conditional loss while leaving the legal-set marginal unchanged. Let $\mathcal{L}_{\text{LM}}^{\text{sum}}$ be the ordinary next-token loss summed over all supervised tokens. The implementation of Equation (4) is therefore

$$\mathcal{L}_{\text{UniGenRec-SFT}} = \frac{1}{N_{\text{tok}}}\left[\mathcal{L}_{\text{LM}}^{\text{sum}} + \sum_{r \in \mathcal{R}} \Big(\mathrm{CE}_{\mathcal{C}_r}(\widetilde{z}_r, y_r) - \mathrm{CE}_{\mathcal{C}_r}(z_r, y_r)\Big)\right]. \tag{11}$$

This form preserves the objective without explicitly materializing the legal-set mass.

Interpretation under a local fit. For UniGenRec-SFT, Equation (1) uses $\alpha_s = g_{\phi_\ell}(H, p)$. If the adjusted policy fits the observed conditional distribution P_{data} at a given context, then

$$\pi_\theta(k \mid H, p) \propto \frac{P_{\text{data}}(k \mid H, p, t)}{P_f(k \mid p, t)^{g_{\phi_\ell}(H,p)}}. \tag{12}$$

Optimization schedule. Under the shared-backbone protocol, every compared method starts from Qwen2.5-1.5B-Instruct. SID alignment is not full-parameter fine-tuning: one epoch on the catalog corpus updates only the newly added SID embeddings and the matching output-head rows; the rest of the backbone stays frozen. UniGenRec-SFT then follows Section 3.2: four epochs of joint training, the same SFT length as the other methods. This second stage is full-parameter fine-tuning of the 1.5B backbone together with the four depth-specific gates; we do not use LoRA or other adapters. Each final linear map is initialized with small random weights so that the initial gate value is near $1/2$. The loss uses $\pi^R_{\theta,\phi}$. Gradients update θ and $\{\phi_\ell\}_{\ell=1}^4$, and pass through g_{ϕ_ℓ}. They do not enter f_t.

All SFT runs, including UniGenRec-SFT and the compared methods, use eight H100 GPUs in BF16. The optimizer is AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.95$) with weight decay 0.01. The initial learning rate is 3×10^{-4}, with a linear warmup over the first 3% of SFT steps and cosine decay to zero. The global batch is 128 sequences: per-device batch 4, gradient accumulation 4, and data-parallel size 8. Inputs are truncated to 512 tokens; user histories keep at most the 50 most recent items, as in Appendix F. The Qwen2.5-0.5B replacement uses the same optimizer, schedule, and global batch. The LLaMA-3-8B replacement keeps the same global batch by using per-device batch 1 and accumulation 16.

E UniGenRec-PO Training Details

Candidate generation and rewards. A snapshot π_{old} of the generator draws the group $\mathcal{B} = \{c^{(j)}\}_{j=1}^G$ by trie-constrained sampled-beam decoding, without the frequency branch. Prefix credit in Equation (7) requires a full prefix match. The item term rewards an exact SID match. When the gold SID appears in the group, the other candidates receive rank penalties, larger for higher-ranked negatives. Let b_j be the zero-based beam rank and $d_j = 1/\log_2(b_j + 2)$:

$$R_{\text{item},j} = \begin{cases} \lambda_{\text{hit}}, & c^{(j)} = c^*, \\ -\lambda_{\text{rank}}\, d_j \big/ \sum_{m=1}^G d_m, & c^* \in \mathcal{B} \text{ and } c^{(j)} \neq c^*, \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

We standardize rewards within the group to path advantages $A_j = (R_j - \bar{R})/(s_R + 10^{-4})$, where $\bar{R}$ and s_R are the group reward mean and standard deviation. The same A_j is shared by every SID token on path j.

Frequency-adjusted policy ratio. Updates use the current factored policy $\pi^R_{\theta,\phi}$ and a frozen snapshot $\pi^R_{\theta_{\text{old}},\phi_{\text{old}}}$ from Equation (1), with the same f_t. The current policy uses $\alpha_s = g_{\phi_\ell}(\mathbf{h}_\theta(H, p))$; the old policy uses $\alpha_s = g_{\phi_{\text{old},\ell}}(\mathbf{h}_{\theta_{\text{old}}}(H, p))$. The pair $(\theta_{\text{old}}, \phi_{\text{old}})$ is copied before the update and treated as stop-gradient. Thus, the denominator uses a fixed GRPO snapshot rather than re-evaluating the current gate on old hidden states. The live gates remain trainable: gradients of the numerator pass through g_{ϕ_ℓ}, while f_t and the snapshot stay stop-gradient. In Equation (5), $k_{j,\ell} = c^{(j)}_\ell$ is the generated token and $s_{j,\ell} = (H, c^{(j)}_{<\ell}, t)$ is its state.

Sampling-to-update weights. Because candidates come from the generator alone, we use an additional capped, stop-gradient weight to reweight each sampled token toward the old frequency-adjusted policy:

$$w_{j,\ell} = \text{sg}\left[\min\left(\frac{\pi^R_{\theta_{\text{old}},\phi_{\text{old}}}(k_{j,\ell} \mid s_{j,\ell})}{\pi_{\theta_{\text{old}}}(k_{j,\ell} \mid s_{j,\ell})},\, w_{\max}\right)\right]. \tag{14}$$

These token-level weights provide a local approximation for reweighting sampled updates toward the frequency-adjusted policy. They do not constitute an exact correction for the coupled sampled-beam distribution.

Policy surrogate and KL regularization. Expanding ℓ_{clip} and averaging first over tokens, then over paths in Equation (6) gives the weighted clipped objective:

$$\mathcal{L}_{\text{PG}} = -\frac{1}{G}\sum_{j=1}^{G}\frac{1}{L_j}\sum_{\ell=1}^{L_j} w_{j,\ell}\min\{r_{j,\ell}A_j,\ \text{clip}(r_{j,\ell}, 1-\delta, 1+\delta)A_j\}. \tag{15}$$

The KL term regularizes the serving generative model against the frozen UniGenRec-SFT reference:

$$\mathcal{L}_{\text{KL}} = \frac{1}{G}\sum_{j=1}^{G}\frac{1}{L_j}\sum_{\ell=1}^{L_j} D_{\text{KL}}(\pi_\theta(\cdot \mid s_{j,\ell}) \,\|\, \pi_{\text{ref}}(\cdot \mid s_{j,\ell}))\,. \tag{16}$$

Both distributions are normalized over legal children and omit the frequency branch. The complete objective is

$$\mathcal{L}_{\text{UniGenRec-PO}} = \mathcal{L}_{\text{PG}} + \beta_{\text{KL}}\mathcal{L}_{\text{KL}}. \tag{17}$$

The surrogate and KL each average over the SID tokens of a path and then over paths. The length L_j excludes EOS and padding.

Optimization settings. UniGenRec-PO continues full-parameter updates of the backbone and gates for two epochs from the UniGenRec-SFT checkpoint, retaining a frozen copy as the reference. It uses the same AdamW settings, weight decay, sequence length, and eight-H100 BF16 setup as SFT. The learning rate is 5×10^{-7}, with no warmup and a constant schedule; the same value is used for every standard GRPO run in the MiniOneRec block. The global batch is 32 sequences (per-device batch 1, accumulation 4) because each example expands to $G = 32$ sampled completions. A trie-constrained sampled beam of width 32 gives $G = 32$ distinct completions per input. The item-level weights in Equation (13) are $\lambda_{\text{hit}} = 4$ and $\lambda_{\text{rank}} = 1$. The generation temperature is 1, the PPO clipping threshold is $\delta = 0.2$, the sampling-weight cap is $w_{\max} = 10$, and $\beta_{\text{KL}} = 10^{-2}$. Advantage standardization adds 10^{-4} to the denominator. Frequency scores receive no gradients. Stop-gradient applies to f_t and to the snapshot $(\theta_{\text{old}}, \phi_{\text{old}})$; it does not block gradients through the current gate g_{ϕ_ℓ}. Generation uses the generative model alone; the local surrogate in Equation (15) optimizes the frequency-adjusted policy.

F Data Construction and Evaluation Protocol

F.1 Dataset construction

The following counts refer to individual next-item prediction events, not the number of examples after adding auxiliary training tasks.

Table 7: Dataset event counts and catalog sizes. Amazon Reviews 2023 Beauty and Industrial & Scientific follow Hou et al. (2026).

Dataset	Training events	Evaluation events	Catalog size
MIND	231,530	107,968 (official dev)	65,238
Amazon Reviews 2023 Beauty	38,579	1,403 (test)	29,180
Amazon Reviews 2023 Industrial & Scientific	297,922	70,897 (test)	25,848
KuaiRec	166,241	20,957 (test)	3,327

All reported runs use only the training split for optimization. We train for four SFT epochs before any applicable policy-optimization stage, then evaluate on the evaluation split in Table 7. The validation split, where it exists, is never used for training, early stopping, or checkpoint selection.

MIND. We use the official training split for training and the official development split for evaluation, with no extra validation partition. Each clicked item (label=1) in an impression becomes a separate prediction target. Multiple targets from the same impression share its original pre-impression history and do not observe one another. We retain the most recent 50 catalog-supported history items and omit impressions with empty retained histories. The fixed catalog is the union of news items in the official training and development metadata. Thus, the reported 107,968 events are the complete evaluation population after preprocessing, not the raw development impressions.

Amazon Reviews 2023 Beauty. We use the Amazon Reviews 2023 Beauty category (Hou et al., 2026) without additional k-core filtering. Each review is treated as an implicit-feedback next-item target. We deduplicate user–item reviews, order all retained interactions globally by timestamp, and split them by 80%/10%/10% event position into training, validation, and test. Table 7 reports the training and test event counts; the validation partition is never used for training, early stopping, or checkpoint selection. Histories contain at most 50 strictly earlier interactions; events without prior history are omitted as prediction targets but retained for constructing subsequent histories. The catalog contains the retained target and history items; missing product titles receive a neutral placeholder.

Amazon Reviews 2023 Industrial & Scientific. We use the Amazon Reviews 2023 Industrial & Scientific category (Hou et al., 2026) and adopt the publisher's 5core/timestamp_w_his benchmark. The publisher deduplicates user–item reviews and applies 5-core filtering; we do not repeat this filtering. Every released review interaction is treated as an implicit-feedback target, irrespective of its rating. These events are reviews, not observed clicks. The publisher's global timestamp split provides the training and test counts in Table 7; we train on the training split only. We preserve the official history ordering, including its source-order convention for timestamp ties, and truncate model inputs to the most recent 50 items. Empty-history events are retained. The catalog contains the retained target and history items; items are not removed because of missing titles, which receive a neutral placeholder.

KuaiRec. We use the KuaiRec 2.0 small matrix, containing 4,676,570 viewing records from 1,411 users over 3,327 videos. A positive interaction is defined by a finite watch_ratio strictly greater than 2; it is not an explicit click label. Among 217,175 positive records, we discard 7,610 with missing or non-finite timestamps, without imputation. We chronologically order the remaining 209,565 positives and split them by global 80%/10%/10% event position into training, validation, and test. Histories contain at most 50 strictly earlier positive interactions; all events sharing a timestamp are processed before updating histories with that timestamp's interactions. Events without prior positive history are omitted as prediction targets but retained for constructing subsequent histories. This removes 1,411 initial training events and yields the counts in Table 7. No additional k-core filtering is applied, and all 3,327 videos remain in the catalog. This small-matrix temporal adaptation is distinct from the official Big-to-Small protocol.

F.2 History and catalog availability

For Beauty and Industrial (Hou et al., 2026), as well as KuaiRec, evaluation follows a rolling-history protocol: earlier observed interactions may appear in later queries' histories, while only training interactions are used as optimization targets or incorporated into training frequency statistics. Catalogs remain fixed during evaluation, without query-time filtering by item release or availability. Content-based SID construction and item alignment may access the complete catalog, including evaluation-only items, but not their held-out interaction labels. Consequently, this is a content-transductive temporal recommendation setting, not strictly inductive cold-start evaluation. Industrial additionally inherits the publisher's full-period 5-core filtering and retrospective metadata availability.

F.3 Full-corpus retrieval and aggregation

Retrieval considers the complete catalog, excluding items in the supplied, truncated history—not the user's entire lifetime history. No sampled negative candidate set is used. Generative models use legal-SID-constrained beam search with beam width 50. The shared TIGER/MiniOneRec map and LC-Rec use a single SID length; only CRAB, Ghost, and FSGR may emit variable-length SIDs. For those three methods, search is performed separately for each SID length, and recovered candidates are merged using the length-normalized score in Equation (8). EOS is neither generated nor scored. This is approximate search over the full catalog, not exhaustive scoring of every item. Our backbone-only variants discard the auxiliary frequency branch at inference and apply no prior subtraction. Recall@10/50 and nDCG@10/50 are computed from the same final ordered Top-50 list. Each event has one target, so Recall@K equals its hit indicator; a hit at one-based rank $r \leq K$ contributes $1/\log_2(r+1)$ to nDCG@K. Metrics are averaged over events, rather than equally over users or evaluation shards. Any target excluded because it occurs in the input history remains in the denominator as a miss.

F.4 Cold and warm targets

An event is Warm if its target occurs among the materialized next-item training targets, and Cold otherwise. This classification remains fixed across checkpoints and cutoffs. Presence in item metadata, historical context, or SID alignment does not make an item Warm; Cold therefore means interaction-cold, not necessarily newly released or content-unseen. Table 8 reports the resulting event and catalog counts on the MIND and KuaiRec evaluation sets. Table 9 uses these fixed denominators under the MiniOneRec configuration. Recall is computed separately over the cold-target and warm-target event subsets. UniGenRec improves warm-target Recall and cold-target Recall@50 on both datasets. On KuaiRec, cold Recall@50 rises from 0.0344% to 0.5615%.

Table 8: Cold-target support on the MIND and KuaiRec evaluation sets. A catalog item is Cold if it never appears as a materialized next-item training target.

Dataset	Warm events	Cold events	Cold-event share	Catalog size	Cold items	Cold-item share
MIND	52,544	55,424	51.33%	65,238	57,580	88.26%
KuaiRec	3,503	17,454	83.28%	3,327	449	13.50%

Table 9: Recall (%) for warm and cold target items under the MiniOneRec configuration. Cold items have no next-item target occurrences in training. Bold marks the best reported value per column.

	MIND				KuaiRec			
Method	Warm@10	Warm@50	Cold@10	Cold@50	Warm@10	Warm@50	Cold@10	Cold@50
MiniOneRec	0.2550	1.6500	0.0000	0.0361	2.2267	9.5061	0.0000	0.0344
TWMT	0.2798	1.7814	0.0000	0.0253	2.4836	9.7060	0.0000	0.0344
Ghost	0.2950	1.7833	0.0000	0.0235	0.6566	1.7414	0.0000	0.0000
CRAB	0.3692	2.1468	0.0000	0.0054	5.1670	10.9906	0.0000	0.0000
UniGenRec	**2.2800**	**5.5166**	0.0000	**0.1828**	**6.9940**	**20.1827**	**0.0057**	**0.5615**

F.5 Checkpoint selection

We train for four SFT epochs on the training split and evaluate that checkpoint on the evaluation split. Validation events are not used to select a model. When policy optimization is used, it starts from this four-epoch SFT checkpoint. In Table 1, the MiniOneRec UniGenRec row reports UniGenRec-SFT followed by UniGenRec-PO; the TIGER and LC-Rec UniGenRec rows report UniGenRec-SFT only.

F.6 Personalization diagnostics

Table 2 reports two differences, both in percentage points because Recall is reported as a percentage. Empty-history gap compares the same method with and without the observed history:

$$\text{EmpGap@}K = \text{Recall@}K(H) - \text{Recall@}K(\emptyset), \qquad K \in \{10, 50\}. \tag{18}$$

The empty-history input contains no prior interactions. Both recalls use the same evaluation targets, catalog, cutoff, and history-exclusion rule on items that would otherwise be visible. A positive EmpGap@K means the observed history raises recall; a value near zero indicates little change in aggregate recall, even if the retrieved lists differ. Personalization lift compares a method with a most-popular recommender on the observed-history evaluation:

$$\text{PerLift@}K = \text{Recall@}K(\text{method}) - \text{Recall@}K(\text{Pop}), \qquad K \in \{10, 50\}. \tag{19}$$

The most-popular recommender ranks items by training interaction counts and returns the top K items after applying the same history exclusion mask used by the generative methods. Both Recall values are averaged over the same evaluation events, catalog, and cutoff, without the frequency branch. On MIND, this baseline attains Recall@$10 = 0.0565\%$ and Recall@$50 = 0.4826\%$; its nDCG@10 and nDCG@50 are 0.0170% and 0.1012%. On Beauty, it attains Recall@$10 = 0.7128\%$ and Recall@$50 = 1.6393\%$; its nDCG@10 and nDCG@50 are 0.3426% and 0.5650%. On KuaiRec, it attains Recall@$10 = 0.0859\%$ and Recall@$50 = 2.1329\%$; its nDCG@10 and nDCG@50 are 0.0324% and 0.4597%. The nDCG values are not used in PerLift. Popularity counts use training interactions only, without temporal decay; recommendations exclude the most recent 50 items in the supplied history. Table 2 reports this comparison under MiniOneRec on all three datasets. PerLift@K is computed within each seed before averaging across seeds. Because Recall is reported

as a percentage, PerLift@K is an absolute difference in percentage points, not a relative improvement rate. A positive value means that the method retrieves the target more often than recommending the globally most popular items. The Rel. row of each panel in Table 2 reports UniGenRec relative to MiniOneRec. For PerLift,

$$\text{Rel@}K = \frac{\text{PerLift@}K\text{(UniGenRec)}}{\text{PerLift@}K\text{(MiniOneRec)}}, \qquad K \in \{10, 50\}. \tag{20}$$

A value greater than 1 means UniGenRec's lift over the most-popular recommender is larger than that of MiniOneRec. EmpGap uses the same ratio across datasets.

G Coverage and Training-Relative Exposure

This appendix defines the five list-level diagnostics in Table 3. Results are means over the same three random seeds as Table 1, on the MIND evaluation set at cutoff $K = 50$. In the MiniOneRec block of Table 1, UniGenRec is UniGenRec-SFT followed by UniGenRec-PO. Figure 1 compares item exposure for MiniOneRec, SASRec, and UniGenRec. The comparison rows use the same MiniOneRec configuration after their own standard GRPO stage. The evaluation contains 107,968 events, 65,238 catalog items, and 2,188 distinct target items. Coverage and HitCov are percentages; the remaining columns are raw scores or ratios. An upward arrow marks greater coverage or list difference; a downward arrow marks a smaller deviation from the specified training reference. These arrows do not denote unconditional recommendation quality or fairness. The Cold/Tail/Mid/Head bins in Figure 1 are a different grouping protocol from the top-1% versus rest partition used here.

Let $\mathcal{I}$ be the catalog, $\mathcal{E}$ the evaluation events, y_e the observed target of event e, and R_e^K the set of K distinct retrieved items.

Coverage. Catalog coverage is the fraction of items that appear in at least one list,

$$\text{Coverage@}K = \frac{\left|\bigcup_{e\in\mathcal{E}} R_e^K\right|}{|\mathcal{I}|}. \tag{21}$$

We use this standard aggregate-diversity definition (Mansoury et al., 2020). It measures how widely the catalog is recommended, not whether the exposed items are relevant.

HitCov. Hit-item coverage is

$$\text{HitCov@}K = \frac{\left|\{y_e : e \in \mathcal{E},\, y_e \in R_e^K\}\right|}{\left|\{y_e : e \in \mathcal{E}\}\right|}. \tag{22}$$

The denominator is the number of distinct observed targets, not the catalog size or the number of events. HitCov complements catalog coverage by measuring how many distinct observed targets are retrieved at least once.

InterDiv. Inter-user list difference is the mean Jaccard distance between Top-K sets of different users,

$$\text{InterDiv@}K = \frac{1}{M}\sum_{m=1}^{M}\left(1 - \frac{|R_{e_m}^K \cap R_{e'_m}^K|}{|R_{e_m}^K \cup R_{e'_m}^K|}\right), \qquad u(e_m) \neq u(e'_m). \tag{23}$$

A value of 0 means identical sets and 1 means empty intersection. Following the use of set overlap to measure homogenization (Chaney et al., 2018), we compare recommendation sets rather than consumed-item sets, so this is not a reproduction of that protocol. We sample $M = 100{,}000$ user pairs uniformly with replacement from 48,593 users, then sample one evaluation request uniformly from each user. All methods share this sample with seed 42. A larger value means more different lists, not that the differences match user interests.

Training-adapted MGU. Let c_i be the number of training positive-click targets for item i, and define the training share and equal-weight recommendation share

$$r_{\text{tr}}(i) = \frac{c_i}{\sum_j c_j}, \qquad q_K(i) = \frac{\sum_{e\in\mathcal{E}} \mathbf{1}[i \in R_e^K]}{K|\mathcal{E}|}. \tag{24}$$

The quantity q_K is an offline position share, not an online display probability. Let A be the items in the top 1% of training clicks and B the remainder, with tied items receiving equal fractional membership. Write t_G and q_G for the training-click and recommendation shares of group G. Adapting the Mean Group Unfairness metric of Jiang et al. (2024), we compare training positive-click shares with recommendation shares for this two-group partition:

$$\mathrm{MGU}_{\mathrm{tr}}@K = \frac{1}{2} \sum_{G \in \{A,B\}} |q_G - t_G| = |q_A - t_A|. \tag{25}$$

The original definition compares recommendation shares with historical interaction shares over possibly different groups. A lower value means closer agreement with the specified training partition, not generally better fairness.

Training-relative amplification. We define

$$\mathrm{Amp}_{\mathrm{tr}}@K = \frac{q_A/q_B}{t_A/t_B}. \tag{26}$$

This is the odds ratio of the hot group versus the rest, relative to the corresponding training odds. It is not the scalar q_A/t_A (head exposure share divided by head training share). Those two quantities coincide only if $q_B = t_B$; in general, $\mathrm{Amp}_{\mathrm{tr}}$ and $\mathrm{MGU}_{\mathrm{tr}}$ need not move together. A value of $1\times$ preserves the training odds, $> 1\times$ further amplifies the head, and $< 1\times$ attenuates it. Table 3 reports both as means over those three seeds; MiniOneRec gives $\mathrm{Amp}_{\mathrm{tr}} = 10.82\times$ and $\mathrm{MGU}_{\mathrm{tr}} = 0.2206$. Amplification has no monotone quality direction, so Table 3 does not mark an arrow for this column.

Relative to MiniOneRec, two-stage UniGenRec increases catalog and hit-item coverage, raises inter-user list difference, and changes head amplification from $10.82\times$ to $0.56\times$. CRAB and Ghost increase that amplification further; TWMT reduces it but remains above $1\times$. These results show broader catalog coverage that includes observed targets, alongside less head exposure relative to the specified training partition. They do not imply that UniGenRec is best on every diversity or fairness axis.

H Amazon Reviews 2023 Industrial & Scientific Results

Table 1 includes Beauty (Hou et al., 2026); Table 10 separately reports the Amazon Reviews 2023 Industrial & Scientific (Hou et al., 2026) results under the official dataset name. Appendix F describes its evaluation protocol. R and N denote Recall and nDCG (%). Bold and underlined values indicate the best and second-best results within each framework block.

Table 10: Full-catalog retrieval on Amazon Reviews 2023 Industrial & Scientific (Hou et al., 2026) under the shared Qwen2.5-1.5B-Instruct protocol. The UniGenRec rows follow the same checkpoint convention as Table 1. Bold and underlined values are the best and second-best within each framework.

	TIGER				MiniOneRec				LC-Rec			
Setting	R@10	R@50	N@10	N@50	R@10	R@50	N@10	N@50	R@10	R@50	N@10	N@50
Base	0.8096	2.2789	0.4474	0.8165	1.1087	<u>2.9987</u>	0.5523	0.9606	1.2036	<u>2.8863</u>	0.5842	<u>0.9146</u>
+ CRAB	0.8411	2.0316	0.4618	0.7824	1.1453	2.7886	0.5806	0.9407	—	—	—	—
+ Ghost	**0.9526**	1.9126	**0.5486**	0.8124	**1.2469**	2.6799	**0.6478**	<u>0.9647</u>	—	—	—	—
+ TWMT	0.8811	**2.3048**	0.4912	<u>0.8196</u>	1.1566	2.7787	0.5922	0.9482	<u>1.2586</u>	2.7147	<u>0.6186</u>	0.9084
+ FSGR	0.8468	2.1284	0.4586	0.7984	1.1528	2.8642	0.5764	0.9528	—	—	—	—
+ UniGenRec	<u>0.9145</u>	<u>2.2864</u>	<u>0.4968</u>	**0.8542**	<u>1.2342</u>	**3.0189**	<u>0.6150</u>	**1.0397**	**1.4862**	**3.1426**	**0.6742**	**1.0524**

Note. CRAB, Ghost, and FSGR are omitted under LC-Rec because they are incompatible with its uniform semantic mapping (USM) indexing.

I Effect of the Generative Backbone

The main comparison uses Qwen2.5-1.5B-Instruct. Tables 11 and 12 repeat the MiniOneRec comparison in Table 1 with Qwen2.5-0.5B (Qwen Team, 2024) and LLaMA-3-8B (Grattafiori et al., 2024). Each table covers the same three datasets, six method settings, and four retrieval metrics.

Table 11: Generative backbone replacement with Qwen2.5-0.5B under the MiniOneRec configuration. R and N denote Recall and nDCG (%).

	MIND				KuaiRec				Beauty			
Setting	R@10	R@50	N@10	N@50	R@10	R@50	N@10	N@50	R@10	R@50	N@10	N@50
MiniOneRec	0.1871	1.2041	0.0913	0.3019	0.4295	2.0136	0.1757	0.5311	1.9244	2.7085	1.6517	1.8314
+ CRAB	0.2649	1.5708	0.1303	0.3975	0.5058	2.3763	0.3157	0.6866	2.0670	2.9936	1.7018	1.9412
+ Ghost	0.2214	1.3087	0.1220	0.3455	0.1002	0.2672	0.0527	0.0907	1.9244	2.7798	1.6524	1.8416
+ TWMT	0.1973	1.3421	0.0929	0.3341	0.4390	2.1520	0.1816	0.5561	2.2096	3.2074	1.6612	1.8718
+ FSGR	0.2603	1.7765	0.1350	0.4459	0.5010	2.5862	0.3146	0.7249	2.3521	3.4212	1.7412	1.9527
+ UniGenRec	**0.8465**	**3.1454**	**0.3962**	**0.8816**	**1.0259**	**4.2182**	**0.7204**	**1.3117**	**2.7085**	**4.2053**	**1.8716**	**2.2114**

Table 12: Generative backbone replacement with LLaMA-3-8B under the MiniOneRec configuration. R and N denote Recall and nDCG (%).

	MIND				KuaiRec				Beauty			
Setting	R@10	R@50	N@10	N@50	R@10	R@50	N@10	N@50	R@10	R@50	N@10	N@50
MiniOneRec	0.1417	0.8688	0.0665	0.2198	0.3913	1.6796	0.1566	0.4461	2.0670	2.9936	1.7813	1.9875
+ CRAB	0.1945	1.1059	0.0940	0.2892	0.8923	1.8800	0.4840	0.7170	2.1383	3.1361	1.7965	2.0024
+ Ghost	0.1556	0.9160	0.0861	0.2434	0.1193	0.3102	0.0642	0.1031	2.2808	3.5638	1.9552	2.2301
+ TWMT	0.1473	0.9216	0.0677	0.2312	0.4342	1.7035	0.1756	0.4585	2.3521	3.6351	1.8568	2.0845
+ FSGR	0.1936	1.2235	0.0981	0.3174	0.8732	1.9707	0.4788	0.7354	2.5659	4.0627	1.9415	2.3012
+ UniGenRec	**1.1466**	**2.8647**	**0.5885**	**0.9532**	**1.2168**	**3.9366**	**0.8207**	**1.3722**	**2.9223**	**4.4904**	**2.0290**	**2.3857**