NEIGHBORHOOD-AWARE UNLEARNING FOR GENERATIVE RECOMMENDATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative recommendation predicts items by autoregressively generating semantic identifiers (SIDs), whose tokens are shared across semantically related items. With this paradigm gaining adoption, the question of how to support unlearning in it has become pressing. If training interactions must later be removed, such as injected spam or interactions with items a user no longer wants, this sharing lets an unlearning update unintentionally affect the predictions for the interactions' neighbors. We show that this effect is locally structured and hidden by global metrics: unlearning can leave recommendation utility on the whole test set nearly unchanged while substantially degrading it on neighborhoods of the forgotten interactions' SID. We propose Neighborhood-Aware Unlearning (NAU), which augments interaction-level unlearning with explicit control over the probability mass assigned to the forgotten items' neighborhoods. Across three datasets and four SID tokenizers, NAU restores spam-induced exposure toward the retrained reference while retaining more than 95% of retrained utility, and recovers the neighborhood utility that forget-and-repair unlearning loses at no global cost. We identify local probability redistribution as an important failure mode of unlearning in SID-based generative recommenders and show that explicitly controlling this redistribution can mitigate its effects.

1 INTRODUCTION

Generative recommendation (GenRec) predicts the next item from a user's interaction history by autoregressively generating its semantic identifier (SID) (Rajput et al., 2023). SIDs represent items as sequences of discrete tokens derived from item representations, such that semantically related items can share parts of their IDs. As the paradigm is rapidly being adopted across diverse real-world recommendation settings (Collins et al., 2026; D'Amico et al., 2026; Li et al., 2026), it becomes urgent to consider how unlearning can be achieved within it (Chen et al., 2022; Schelter et al., 2023). Malicious interactions such as spam, fake reviews, or shilling attacks can manipulate recommendations to promote target items (Li et al., 2022; Wu et al., 2021). Users may request the removal of interactions involving unwanted items from their recommendation history and future predictions (Schelter et al., 2023). Retraining without the deleted interactions is the natural reference, but it is often too expensive for frequent deletion requests (Chen et al., 2024; Lubitzsch et al., 2026), which motivates efficient approximate unlearning.

Challenges of unlearning for GenRec. In a conventional dense recommender, each item has its own output embedding, so an update that lowers one item's score directly affects that item's embedding and shared model parameters, but not the output embeddings of other items (Kang & McAuley, 2018; Li et al., 2023). In GenRec, candidate items are ranked by the probability of generating their SIDs. The item SID probability is the product of the conditional probabilities of the SID tokens. Items that share an SID prefix therefore share the corresponding prefix-level factors, so updates to these factors can affect multiple items under the same history, with the directly coupled items identifiable from the SID structure. Straightforward adaptations of existing deletion and machine-unlearning strategies do not fully address this shared-token coupling in GenRec: (i) inference-time filtering can remove a target from the recommendation list, while the model may still prefer to generate its SID; (ii) directly unlearning the forgotten interactions can also affect other items that share SID tokens with the target; and (iii) retain-set repair (Jia et al., 2023) can recover overall recommendation quality but does not explicitly control such spillover. *The central challenge is therefore*

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.

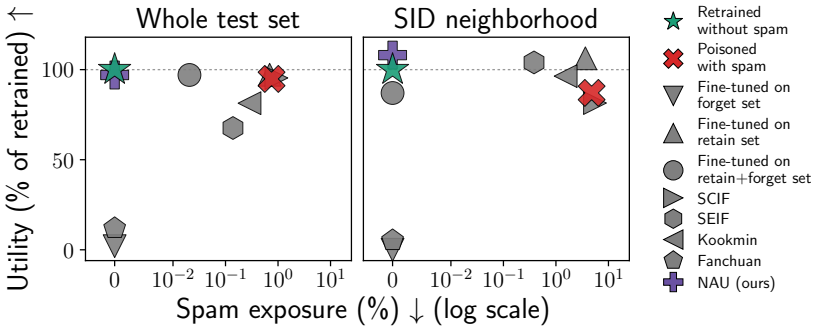


Figure 1: Utility ↑ (NDCG@10 as a percentage of the retrained model's) versus spam exposure after spam removal, on the whole test set (left) and on the test cases whose next item shares the forgotten target's first SID token (right). Only NAU removes spam exposure while preserving model utility on both the whole test set and the neighborhood of the forgotten interactions.

to remove the influence of a target item without unnecessarily propagating the unlearning effect to other items that share SID tokens with the target.

Diagnosing SID-induced unlearning spillover. We call the items that share SID tokens with a target its *SID neighborhood*. Our key observation is that unlearning effects concentrate there: global utility can stay close to the retrained reference while utility on the neighborhood changes considerably (Figure 1), and methods that ignore the neighborhood cannot both remove influence of interactions to forget and preserve local and global utility. This motivates using neighborhoods not only to diagnose unlearning effects, but also to control them.

Our approach. We propose Neighborhood-Aware Unlearning (NAU) for GenRec to remove the influence of forgotten interactions while explicitly controlling how unlearning affects items that share SID tokens with the corresponding target items. Because GenRec is trained through token-level SID generation whereas deletion requests concern interactions and items, NAU defines an SID-aware item-level objective over the generated identifier. We further design neighborhood mass control to regulate the total generation probability assigned to the target item's neighborhood. This allows NAU to preserve legitimate neighboring items in spam removal, or suppress neighborhood exposure when nearby items are also relevant to the removal request. NAU keeps the model architecture and SIDs unchanged and requires no second-order information. On three Amazon recommendation datasets (Beauty, Sports, and Toys) and four SID or tokenization settings, NAU achieves removal behavior close to retraining while retaining over 95% of its recommendation utility, at three orders of magnitude lower runtime than retraining. For unwanted item removal, the same neighborhood-control principle further reduces unwanted exposure while preserving recommendation utility.

Contributions. In summary, our main contributions are:

- We formulate unlearning for GenRec under two practical deletion scenarios: spam removal and unwanted item removal (Section 2).
- We identify and analyze SID-induced unlearning spillover, showing that unlearning effects are concentrated around items neighboring the target and motivating explicit neighborhood-level control (Section 5).
- We propose neighborhood-aware unlearning (NAU), which combines an SID-aware unlearning objective with neighborhood mass control (Section 3).
- We provide a systematic evaluation of representative general-purpose and recommendation-specific unlearning methods on three Amazon recommendation datasets and four SID or tokenization settings, covering both spam removal and unwanted item removal. Our results reveal distinct forgetting–utility failure modes of existing methods and show that NAU achieves effective forgetting while preserving over 95% of retrained utility (Sections 4 and 5).

We release our implementation, experimental logs, and results at https://anonymous.4open.science/r/NAU-4FF1.

2 PRELIMINARIES AND PROBLEM DEFINITION

Generative recommendation with semantic identifiers. A SID-based GenRec system consists of two main components: an item tokenizer and a generative recommender. Let $\mathcal{I}$ be the item set. The tokenizer maps each item $i \in \mathcal{I}$ to a semantic identifier $\mathrm{SID}(i) = (c_i^1, \ldots, c_i^L)$, where c_i^l is the semantic token at level l and L is the identifier length. A common construction obtains these tokens as discrete quantization codes from item representations (Luo et al., 2025; Rajput et al., 2023). Semantically similar items may share one or more SID tokens. Given these SIDs, the recommender predicts the next item from a user's interaction history by autoregressively generating its identifier. For a training example $(\boldsymbol{x}, i)$, the generation probability of item i is $\mathbb{P}_{\boldsymbol{\theta}}(i \mid \boldsymbol{x}) = \prod_{l=1}^{L} \mathbb{P}_{\boldsymbol{\theta}}\big(c_i^l \mid c_i^{<l}, \boldsymbol{x}\big)$. At inference time, valid generated SIDs are mapped back to items and ranked by their generation probabilities. Because different items may share SID tokens, updates involving shared tokens can affect the generation probabilities of multiple items.

Item neighborhoods. We use an item neighborhood to capture items close to a target under a chosen proximity criterion. For an item i, let $\mathcal{N}(i) \subseteq \mathcal{I} \setminus \{i\}$ denote its neighborhood. It may contain items sharing an SID prefix with i, or the top-k items under an embedding-based similarity function, $\mathcal{N}_k(i) := \text{top-}\, k_{j \in \mathcal{I} \setminus \{i\}} \mathrm{sim}(i, j)$, We treat neighborhood construction as modular and only assume access to a similarity function.

Unlearning interactions in GenRec. We study interaction-level unlearning in GenRec, where deletion requests are specified over individual training interactions. A training example is represented as $(\boldsymbol{x}, i)$, where $\boldsymbol{x}$ is the interaction history and i is the next item. Let D be the original training set, $D_f \subset D$ the forget set, and $D_r = D \setminus D_f$ the retain set. Let $\boldsymbol{\theta}_o$ denote the original model trained on D, $\boldsymbol{\theta}_r$ the reference model retrained from scratch on D_r, and $\boldsymbol{\theta}_u$ the model produced by an unlearning algorithm from $\boldsymbol{\theta}_o$. The goal is to remove the influence of D_f while preserving recommendation behavior learned from D_r, at substantially lower cost than retraining.

Unlearning scenarios. We consider two practical deletion scenarios motivated by prior work (Chen et al., 2024; Li et al., 2022; Lubitzsch et al., 2026; Schelter et al., 2023; Wu et al., 2021). They differ in the desired effect on the target item and its neighborhood.

(1) *Spam removal.* Fake or poisoned interactions promote a set of target items $\mathcal{I}_{\mathrm{p}} \subset \mathcal{I}$, and D_f contains these injected interactions. The goal is to remove the resulting promotion effect without globally suppressing target items that may otherwise remain legitimate.

(2) *Unwanted item removal.* A subset of users $\mathcal{U}_{\mathrm{u}} \subset \mathcal{U}$ request removal of interactions involving unwanted items $\mathcal{I}_{\mathrm{u}} \subset \mathcal{I}$, and D_f contains the corresponding interactions. Unlike spam removal, the goal here is behavioral: requesting users should stop being recommended $\mathcal{I}_{\mathrm{u}}$; we therefore judge removal directly (using UHF@10, see Section 4) and use the retrained model only as the reference for other users (using UHR@10, see Section 4). When nearby items share the semantics motivating the request, the desired reduction in exposure may also extend to their neighborhood.

Evaluation criteria. We evaluate unlearning along three dimensions: (i) effectiveness, measuring whether the influence associated with D_f is removed; (ii) utility, measuring whether recommendation quality on retained data is preserved; and (iii) efficiency, the computational cost relative to retraining. For spam removal, effectiveness is measured by whether promoted-item exposure returns toward the retrained reference. For unwanted item removal, it is measured by whether unwanted-item exposure decreases for requesting users. Because unlearning may affect neighboring items differently from the overall item space, we evaluate both global recommendation quality and local neighborhood behavior.

3 METHODOLOGY

Overview. We propose Neighborhood-Aware Unlearning (NAU) for GenRec. NAU consists of three main components: (i) an item-level loss defined over generated SIDs; (ii) forget-and-retain optimization that removes the influence of forgotten interactions while preserving retained behavior; and (iii) neighborhood mass control that explicitly regulates how unlearning affects neighboring

items. Unlike standard forget-and-retain optimization (Kurmanji et al., 2023; Triantafillou et al., 2024), NAU does not treat the target of each forgotten interaction in isolation, but explicitly controls its surrounding item neighborhood.

Item-level loss over SIDs. GenRec is trained by predicting SID tokens, whereas unlearning requests concern interactions and their target items. We therefore define an item-level loss using the likelihood of the complete target SID. For a training pair $(\boldsymbol{x}, i)$,

$$\ell(i \mid \boldsymbol{x}; \boldsymbol{\theta}) = -\sum_{l=1}^{L} \log \mathbb{P}_{\boldsymbol{\theta}} \left(c_i^l \mid c_i^{<l}, \boldsymbol{x} \right), \tag{1}$$

i.e., the negative log-likelihood of the complete target SID.

Forget-and-retain optimization. We optimize over both forgotten and retained interactions to remove the requested influence while maintaining useful recommendation behavior.

Retain repair. We preserve recommendation behavior on retained interactions with $\mathcal{L}_{\text{ret}} = \mathbb{E}_{(\boldsymbol{x}_r, i_r) \sim D_r} \left[\ell(i_r \mid \boldsymbol{x}_r; \boldsymbol{\theta}) \right]$.

Forget removal. We remove the influence of forgotten interactions by maximizing their item-level loss, $\mathcal{L}_{\text{for}} = -\mathbb{E}_{(\boldsymbol{x}_f, i_f) \sim D_f} \left[\ell(i_f \mid \boldsymbol{x}_f; \boldsymbol{\theta}) \right]$. Minimizing $\mathcal{L}_{\text{for}}$ therefore performs gradient ascent on the forgotten examples and reduces the generation probability of their targets under the corresponding histories.

Target separation. Forget removal suppresses a target under its forgotten histories, but does not directly constrain its ranking under other histories. We therefore introduce a contrastive separation term over retained examples. Let $\phi(i \mid \boldsymbol{x}; \boldsymbol{\theta}) = \log \mathbb{P}_{\boldsymbol{\theta}}(i \mid \boldsymbol{x})$ denote the item score induced by its generated SID probability, and let $\mathcal{I}_f = \{i : (\boldsymbol{x}, i) \in D_f\}$ denote the set of target items associated with forgotten interactions. We define

$$\mathcal{L}_{\text{sep}} = \mathbb{E}_{(\boldsymbol{x}_r, i_r) \sim D_r} \left[-\log \frac{\exp\left(\phi(i_r \mid \boldsymbol{x}_r; \boldsymbol{\theta})/\tau\right)}{\exp\left(\phi(i_r \mid \boldsymbol{x}_r; \boldsymbol{\theta})/\tau\right) + \sum_{i_f \in \mathcal{I}_f \setminus \{i_r\}} \exp\left(\phi(i_f \mid \boldsymbol{x}_r; \boldsymbol{\theta})/\tau\right)} \right], \tag{2}$$

where τ is a temperature. This term discourages forget-associated target items from remaining highly ranked under retained histories.

Neighborhood mass control. The preceding terms operate on exact target items but do not explicitly control how unlearning affects their neighbors. For a forgotten interaction $(\boldsymbol{x}_f, i_f)$, let $\mathcal{N}(i_f)$ denote the neighborhood of i_f defined in Section 2. We control the total generation probability assigned to this neighborhood through

$$\mathcal{L}_{\text{nbr}} = \mathbb{E}_{(\boldsymbol{x}_f, i_f) \sim D_f} \left[-\log \sum_{j \in \mathcal{N}(i_f)} \mathbb{P}_{\boldsymbol{\theta}} \left(j \mid \boldsymbol{x}_f \right) \right]. \tag{3}$$

Full objective. Combining the three components, our final unlearning objective NAU is

$$\mathcal{L} = \mathcal{L}_{\text{ret}} + \lambda_f \mathcal{L}_{\text{for}} + \lambda_s \mathcal{L}_{\text{sep}} + \lambda_n \mathcal{L}_{\text{nbr}}, \tag{4}$$

where λ_f, λ_s, and λ_n control the strengths of the forget, separation, and neighborhood terms, respectively. The sign of λ_n determines how the neighborhood is treated. When $\lambda_n > 0$, minimizing Eq. 4 increases the probability assigned to neighboring items, which is useful in spam removal when these items remain legitimate alternatives. When $\lambda_n < 0$, it suppresses neighborhood probability, which is useful in unwanted item removal when nearby items may share the semantics motivating the deletion request.

4 Experimental Setup

Research questions. Our experiments are organized around the following research questions:

- **(RQ1)** How well does NAU remove scenario-specific target exposure while preserving recommendation utility compared to existing unlearning baselines?
- **(RQ2)** How do the retain, forget, separation, and neighborhood mass terms contribute to the final unlearning behavior?

(RQ3) Can the same neighborhood-aware framework handle both the spam removal and unwanted-item removal scenario?

(RQ4) How does unlearning affect the target item's SID neighborhood compared with the overall item space?

Datasets and backbone. We evaluate interaction-level unlearning for GenRec on three sequential recommendation datasets from Amazon: Beauty, Sports, and Toys (He & McAuley, 2016; McAuley et al., 2015), using the five-core preprocessing released by Geng et al. (2022). The Amazon subsets are standard evaluation datasets for GenRec methods (Acharya et al., 2025; Cheng et al., 2026; Fu et al., 2026; Ju et al., 2025; Kim et al., 2026; Liu et al., 2026; Rajput et al., 2023; Wang et al., 2024a;b). Dataset statistics are in Appendix A. Our backbone is TIGER (Rajput et al., 2023), which quantizes item representations into SIDs and autoregressively generates the next item's identifier. Our main experiments construct the SIDs with RQ-KMeans (Luo et al., 2025); we additionally evaluate RQ-VAE (Rajput et al., 2023; Zeghidour et al., 2021) and LETTER's collaboratively regularized tokenizer (Wang et al., 2024a) in Appendix D.

Unlearning scenarios. For spam removal, we inject a 1% bandwagon attack promoting one target item, drawn from the unpopular (bottom 20% by interaction count), mid-popularity (30-70%), or popular (top 5%) items, and unlearn the injected fake users; for unwanted-item removal, 12 to 14 users per seed request deletion of their interactions with a sensitive category (Appendix C).

Baselines. We compare against fine-tuning, forget-only gradient ascent, forget+repair, SCIF (Li et al., 2023), and the SEIF, Kookmin, and Fanchuan baselines of the NeurIPS 2023 unlearning competition (Triantafillou et al., 2024); TRACER (Chen et al., 2026), a concept-unlearning method for GenRec that reassigns SIDs associated with a target concept, is included where its tokenizer setup is applicable. Descriptions are in Appendix B.

Neighborhood construction. NAU's training neighborhood $\mathcal{N}(i)$ is the top-k ($k = 8$) cosine neighbors of i in the pre-quantization embedding space. For evaluation we use the tokenizer's own structure instead: the *SID neighborhood* of a target is the set of test cases whose next item shares the target's first SID token, and the target's own test cases are excluded from the neighborhood scope.

Evaluation metrics. We evaluate effectiveness, utility, and efficiency, using the retrained model $\boldsymbol{\theta}_r$ as the reference where appropriate.

- For spam removal, $\mathrm{SH@}K$ is the hit rate of promoted target items in the top-K list over the test histories $\mathcal{Q}$: $\mathrm{SH@}K = \frac{1}{|\mathcal{Q}|}\sum_{\boldsymbol{x}\in\mathcal{Q}} \mathbb{I}\left[\mathrm{TopK}_{\boldsymbol{\theta}}(\boldsymbol{x}) \cap \mathcal{I}_{\mathrm{p}} \neq \emptyset\right]$. Because the retrained model still recommends the promoted items to some users, $\mathrm{SH@}10 = 0$ is not the target. We therefore report $\Delta^{\mathrm{retr}}_{\mathrm{SH}} = |\mathrm{SH}_U\mathrm{@}10 - \mathrm{SH}_R\mathrm{@}10|$, averaged over (seed, target) cells, where lower is better and over-suppression counts as much as residual exposure. Unlike inference-time filtering, all reported unlearning methods modify the model itself; no items are removed from the generated candidate set at evaluation time.
- Utility is NDCG@10 on the retain test set, where higher is better; $\tau_P = \mathrm{NDCG}_U\mathrm{@}10/\mathrm{NDCG}_R\mathrm{@}10$ measures utility relative to retraining and defines the 95% utility threshold used for model selection.
- For unwanted-item removal, the unwanted-item hit rates $\mathrm{UHF@}K$ and $\mathrm{UHR@}K$ are computed like $\mathrm{SH@}K$ with $\mathcal{I}_{\mathrm{u}}$ in place of $\mathcal{I}_{\mathrm{p}}$, over the requesting and the remaining users; lower $\mathrm{UHF@}10$ is better, and $\Delta^{\mathrm{retr}}_{\mathrm{UHR}} = |\mathrm{UHR}_U\mathrm{@}10 - \mathrm{UHR}_R\mathrm{@}10|$ measures how far the update moves the remaining users away from retraining.
- Efficiency $\mathcal{E}$ is the median wall-clock time per request on one H200 GPU.

Neighborhood comparisons use paired bootstrap intervals over test cases, since all methods are scored on the same cases (Appendix G). $\mathrm{SH@}K$, NDCG@K, and τ_P are percentages, reported as mean $\pm$ s.d. across five data seeds.

Implementation details. For every baseline we report the configuration with the lowest validation $\Delta^{\mathrm{retr}}_{\mathrm{SH}}$ (for unwanted-item removal, the lowest $\mathrm{UHF@}10$) among those keeping validation $\tau_P \geq 95\%$, selected separately per dataset (Tables 7 and 8). NAU's weights ($\lambda_f = 0.1$, $\lambda_s = 0.01$, $\lambda_n = 0.01$) were selected once on the Beauty validation split with RQ-KMeans SIDs: among

Table 1: Spam removal on Beauty, Sports, and Toys with TIGER + RQ-KMeans SIDs under a 1% bandwagon attack; mean ± s.d. over five data seeds × three targets per dataset. $\Delta_{\mathrm{SH}}^{\mathrm{retr}} = |\mathrm{SH}_U@10 - \mathrm{SH}_R@10|$ is the distance of the promoted item's exposure from the retrained model's, so leftover exposure and over-suppression both count. In the Retrain row, the italic value in parentheses is the retrained model's own SH@10, the reference for that column. For each baseline, we report the configuration with the smallest validation $\Delta_{\mathrm{SH}}^{\mathrm{retr}}$ among those keeping validation $\tau_P \geq 95\%$ (ties broken by higher τ_P), or, when no configuration reaches 95%, the smallest $\Delta_{\mathrm{SH}}^{\mathrm{retr}}$ overall. NAU is reported at one operating point for all datasets ($\lambda_f = 0.1$, $\lambda_s = 0.01$, $\lambda_n = 0.01$); the indented row below it uses a stronger separation weight ($\lambda_s = 0.1$) and is not ranked. All values are percentages. Bold and underline mark the best and second-best value per column among methods keeping $\tau_P \geq 95\%$. TRACER has no Beauty run under RQ-KMeans; cells marked † are its Beauty result under its own tokenizer, where τ_P is measured against that tokenizer's retrained reference.

	Beauty			Sports			Toys		
Method	$\Delta_{\mathrm{SH}}^{\mathrm{retr}}\downarrow$	NDCG@10↑	$\tau_P\uparrow$	$\Delta_{\mathrm{SH}}^{\mathrm{retr}}\downarrow$	NDCG@10 ↑	$\tau_P\uparrow$	$\Delta_{\mathrm{SH}}^{\mathrm{retr}}\downarrow$	NDCG@10 ↑	$\tau_P\uparrow$
Retrained	*(0.072)*	3.766 ±0.13	100.0	*(0.139)*	2.033 ±0.07	100.0	*(0.073)*	3.465 ±0.11	100.0
Poisoned	0.902 ±0.59	3.589 ±0.03	95.4 ±3.32	0.168 ±0.17	2.031 ±0.06	100.0 ±2.96	0.221 ±0.14	3.387 ±0.06	97.8 ±3.29
Baselines									
Finetune	0.767 ±0.67	**3.626** ±0.04	**96.4** ±2.80	0.281 ±0.23	**2.082** ±0.04	**102.5** ±3.52	0.823 ±0.32	3.350 ±0.06	96.8 ±4.03
Forget only	0.072 ±0.04	0.086 ±0.14	2.2 ±3.57	0.139 ±0.21	0.014 ±0.02	0.7 ±1.20	0.073 ±0.06	0.012 ±0.01	0.3 ±0.24
Forget + Repair	0.106 ±0.12	3.607 ±0.07	95.9 ±3.29	**0.137** ±0.20	2.068 ±0.04	101.8 ±3.83	0.195 ±0.14	**3.355** ±0.05	**96.9** ±3.65
SCIF	1.309 ±0.66	3.563 ±0.05	94.7 ±2.95	0.467 ±0.37	2.027 ±0.05	99.8 ±3.88	1.232 ±0.81	3.318 ±0.06	95.8 ±3.59
SEIF	0.243 ±0.28	2.506 ±0.08	66.6 ±3.57	0.139 ±0.19	1.218 ±0.05	59.9 ±2.09	0.206 ±0.15	2.185 ±0.09	63.1 ±4.29
Kookmin	0.476 ±0.32	3.046 ±0.09	81.0 ±3.41	0.148 ±0.09	1.626 ±0.07	80.0 ±3.94	0.558 ±0.23	2.664 ±0.03	77.0 ±2.47
Fanchuan	0.072 ±0.04	1.225 ±0.09	32.6 ±3.01	0.139 ±0.21	0.643 ±0.03	31.6 ±0.96	0.073 ±0.06	0.805 ±0.10	23.3 ±3.13
TRACER	1.063 ±0.67†	3.566 ±0.07†	99.1 ±1.98†	0.410 ±0.34	2.028 ±0.05	99.8 ±4.03	1.079 ±0.78	3.314 ±0.07	95.7 ±4.08
Ours									
NAU *(ours)*	**0.081** ±0.04	3.621 ±0.04	96.2 ±3.01	0.138 ±0.21	2.067 ±0.04	101.8 ±3.84	**0.074** ±0.06	3.349 ±0.06	96.8 ±4.22
NAU, λ_s=0.1	0.072 ±0.04	3.617 ±0.04	96.1 ±3.14	0.139 ±0.21	2.067 ±0.04	101.8 ±3.86	0.072 ±0.06	3.350 ±0.06	96.8 ±4.13

settings of the full objective whose exposure is not resolvably different from the lowest, the one with the highest mean of whole-set and neighborhood τ_P (Table 6); they are used unchanged for all datasets and tokenizers, and unwanted-item removal uses $\lambda_s = 0$, $\lambda_n = -1$, with λ_n selected the same way on the Beauty validation split (Table 8).

5 Experimental Results

Overall performance for unlearning spam. To answer **RQ1**, we compare NAU with existing unlearning baselines in the spam-removal scenario. Table 1 reports results on Beauty, Sports, and Toys with TIGER and RQ-KMeans SIDs.

Among methods retaining at least 95% of retrained NDCG@10, NAU achieves the smallest $\Delta_{\mathrm{SH}}^{\mathrm{retr}}$ on Beauty and Toys (0.081% and 0.074%), and is within 0.001 points of the best result on Sports (0.138% versus 0.137%), while retaining 96.2 to 101.8% of retrained utility. The baselines exhibit three main failure modes: (i) Methods such as Finetune, SCIF, and TRACER preserve utility but remain far from the retrained exposure, on Beauty and Toys often farther than the poisoned model itself; (ii) Forget only, Fanchuan, SEIF, and Kookmin can achieve smaller exposure distances only with substantial utility degradation; and (iii) Forget+Repair provides the strongest baseline trade-off, but still remains farther from retraining than NAU on Beauty and Toys at comparable utility. For example, on Toys, Finetune, SCIF, and TRACER have $\Delta_{\mathrm{SH}}^{\mathrm{retr}}$ values of 0.823%, 1.232%, and 1.079%, respectively, compared with 0.074% for NAU. Forget only and Fanchuan keep at most 2.2% and 32.6% of retrained utility, while SEIF and Kookmin retain only 59.9 to 81.0%. Compared with Forget+Repair, NAU reduces $\Delta_{\mathrm{SH}}^{\mathrm{retr}}$ from 0.106% to 0.081% on Beauty and from 0.195% to 0.074% on Toys; on Sports, the two methods are nearly identical (0.137% versus 0.138%). A stronger separation weight ($\lambda_s = 0.1$, indented row) is slightly closer to retrained exposure on Beauty and Toys (0.072% on both) but preserves less neighborhood utility (Section 5). As analyzed next, similar whole-set utility can still mask substantially different behavior around the target's SID neighborhood. Results with RQ-VAE, LETTER, and TRACER identifiers are consistent (Appendix D).

Ablation of the unlearning objective. We ablate the NAU objective on Beauty to answer **RQ2**. Table 2 reports performance on both the whole test set and the target-item SID neighborhood.

Table 2: Ablation of NAU on Beauty with TIGER + RQ-KMeans over five seeds. Indented rows are alternative additions to $\mathcal{L}_{\text{ret}} + \mathcal{L}_{\text{for}}$ (siblings, not a ladder). Since retrained exposure is 0% in both query sets, SH@10 also equals the absolute distance to retrained exposure. τ_P denotes NDCG@10 as a percentage of the retrained model's utility within each query set.

Objective	Whole test set			SID neighborhood		
	$\Delta_{\text{SH}}^{\text{retr}}$ (%)↓	NDCG@10↑	τ_P (%)↑	$\Delta_{\text{SH}}^{\text{retr}}$ (%)↓	NDCG@10↑	τ_P (%)↑
Retrained	*0.000* ±*0.00*	*3.766* ±*0.13*	*100.0*	*0.000* ±*0.00*	*2.779* ±*1.63*	*100.0*
Poisoned	*0.753* ±*0.79*	*3.571* ±*0.09*	*94.8*	*4.780* ±*11.03*	*2.420* ±*1.23*	*87.1*
$\mathcal{L}_{\text{ret}}$ only	0.692 ±0.77	3.659 ±0.06	97.2	3.633 ±4.84	2.954 ±1.76	106.3
$+ \mathcal{L}_{\text{for}}$	0.021 ±0.04	3.654 ±0.07	97.0	0.000 ±0.00	2.419 ±1.44	87.0
$+ \mathcal{L}_{\text{sep}}$ ($\lambda_s = .01$)	0.001 ±0.00	3.648 ±0.05	96.9	0.000 ±0.00	2.428 ±1.42	87.4
$+ \mathcal{L}_{\text{sep}}$ ($\lambda_s = .1$)	0.000 ±0.00	3.644 ±0.04	96.8	0.000 ±0.00	2.021 ±1.50	72.7
$+ \mathcal{L}_{\text{nbr}}$ ($\lambda_n = .01$)	0.004 ±0.01	3.651 ±0.07	97.0	0.000 ±0.00	3.036 ±1.74	109.3
$+ \mathcal{L}_{\text{nbr}}$ ($\lambda_n = .1$)	0.000 ±0.00	3.548 ±0.08	94.2	0.000 ±0.00	4.641 ±3.64	167.0
NAU ($\lambda_s = .01, \lambda_n = .01$)	0.000 ±0.00	3.653 ±0.05	97.0	0.000 ±0.00	3.003 ±1.70	108.1
NAU ($\lambda_s = .1, \lambda_n = .01$)	0.000 ±0.00	3.652 ±0.05	97.0	0.000 ±0.00	2.659 ±1.54	95.7
NAU ($\lambda_s = .1, \lambda_n = .1$)	0.000 ±0.00	3.557 ±0.05	94.5	0.000 ±0.00	3.094 ±2.00	111.3

Role of each objective term. The four terms play complementary roles: (i) $\mathcal{L}_{\text{ret}}$ preserves recommendation utility but removes little spam exposure on its own; (ii) adding $\mathcal{L}_{\text{for}}$ provides the main removal effect, reducing whole-set $\Delta_{\text{SH}}^{\text{retr}}$ from 0.692% to 0.021% while retaining 97.0% of retrained utility; (iii) $\mathcal{L}_{\text{sep}}$ further suppresses target exposure to near zero; and (iv) $\mathcal{L}_{\text{nbr}}$ primarily regulates how the update affects neighboring items.

The neighborhood results reveal changes that are largely hidden by whole-set utility. Adding $\mathcal{L}_{\text{for}}$ keeps whole-set utility at 97.0% of retraining, but neighborhood utility drops to 87.0%. The separation term at $\lambda_s = 0.1$, which removes the promoted item completely, reduces neighborhood utility further to 72.7% at an unchanged whole-set utility of 96.8%, whereas the weaker $\lambda_s = 0.01$ leaves both nearly unchanged (96.9% and 87.4%). In contrast, the variant with $\mathcal{L}_{\text{nbr}}$ at $\lambda_n = 0.01$ yields 109.3% neighborhood utility while maintaining 97.0% whole-set utility. The full objective ($\lambda_s = 0.01, \lambda_n = 0.01$) combines both: it matches the retrained exposure and reaches 108.1% of retrained neighborhood utility at 97.0% whole-set utility; with the stronger separation ($\lambda_s = 0.1$), the neighborhood term still recovers neighborhood utility from 72.7% to 95.7%.

Overall, the retain term preserves utility, the forget and separation terms control target exposure, and neighborhood mass control regulates the local effect of unlearning. Additional ablations on Sports and Toys show the same qualitative pattern and are reported in Appendix E.

Unwanted item removal. To answer **RQ3**, we evaluate whether the same neighborhood-aware framework can also handle unwanted item removal. Unlike spam removal, unwanted item removal may require suppressing nearby items that share similar unwanted semantics. We therefore reverse the direction of neighborhood control by using a negative neighborhood weight.

Comparison with baselines. Table 3 compares NAU with existing unlearning methods on Beauty. Among methods retaining at least 95% of retrained utility, (i) NAU and Forget only reach the lowest UHF@10 (9.419%, against 14.850% before deletion); (ii) methods achieving substantially lower exposure, such as SEIF and Fanchuan, incur severe utility degradation; and (iii) at the same UHF@10, NAU moves the remaining users' exposure less than Forget only ($\Delta_{\text{UHR}}^{\text{retr}}$ of 0.666 versus 1.111), keeping it about as close to retraining as before deletion (0.702), at 95.5% of retrained utility.

Effect of neighborhood control. In Table 4, $\lambda_n = -1$ lowers UHF@10 on Sports on every seed with both tokenizers ($27.6\% \rightarrow 11.1\%$ and $22.0\% \rightarrow 12.1\%$) at unchanged utility, while also lowering the remaining users' exposure (UHR@10 $1.4\% \rightarrow 0.5\%$), since it acts on the category rather than on specific users. On Beauty and Toys, where only 12 to 14 users request deletion, it mainly raises τ_P (on every seed in three of four cells). The same mechanism thus suppresses rather than preserves probability mass around unwanted items.

Across datasets. In the full comparisons (Appendix F), NAU has the second-lowest UHF@10 on Toys among methods keeping 95% utility (9.7%, behind Finetune's 6.5%), and the lowest of any

Table 3: Unwanted item removal on Beauty (Hair Loss Products) with TIGER + RQ-KMeans, five data seeds. UHF@10 and UHR@10 are the unwanted item hit rates of the requesting and the remaining users; $\Delta^{\text{retr}}_{\text{UHR}}$ is the mean over seeds of $|\text{UHR}_U@10 - \text{UHR}_R@10|$, the distance to the retrained model of the same seed. Each method uses the configuration selected on the validation split (Table 8; †: $\tau_P < 95\%$, never ranked); NAU uses $\lambda_n = -1$, $\lambda_s = 0$. Retraining does not lower UHF@10 below the original model.

Method	UHF@10 (%)↓	UHR@10 (%)	$\Delta^{\text{retr}}_{\text{UHR}}$ (%)↓	NDCG@10↑	τ_P (%)↑	$\mathcal{E}$ (s)↓
Finetune	14.424 ±13.461	2.948 ±0.550	0.612 ±0.299	3.510 ±0.162	96.3 ±5.0	0.3
Forget only	**9.419 ±6.109**	1.990 ±1.176	1.111 ±0.684	3.516 ±0.183	96.4 ±5.8	0.3
Forget+Repair	12.606 ±14.899	2.543 ±0.840	0.734 ±0.337	3.522 ±0.168	96.6 ±5.2	0.5
SCIF	15.981 ±8.902	3.303 ±0.424	**0.481 ±0.379**	**3.682 ±0.139**	**101.0 ±4.6**	33.9
SEIF†	0.000 ±0.000	0.000 ±0.000	2.823 ±0.474	0.030 ±0.032	0.8 ±0.9	0.3
Kookmin†	1.818 ±4.066	1.230 ±0.939	1.699 ±1.180	2.788 ±0.179	76.5 ±5.6	0.3
Fanchuan†	0.000 ±0.000	0.000 ±0.000	2.823 ±0.474	0.164 ±0.052	4.5 ±1.4	1.2
NAU *(Ours)*	**9.419 ±6.109**	2.277 ±0.589	0.666 ±0.530	3.484 ±0.191	95.5 ±5.6	3.7
Original (before deletion)	14.850 ±7.051	3.449 ±0.843	0.702 ±0.460	3.765 ±0.134	103.3 ±4.4	–
Retrain (reference)	16.237 ±9.987	2.823 ±0.474	0.000	3.648 ±0.038	100.0	*17,399*

Table 4: Effect of the neighborhood term in unwanted item removal across datasets and tokenizers: NAU with $\lambda_n = -1$ against the same objective without the term ($\lambda_n = 0$; $\lambda_f = 0.1$, $\lambda_s = 0$), paired over five data seeds. Each cell gives $\lambda_n = 0 \rightarrow -1$ (mean over seeds) and, in parentheses, on how many of the five seeds the value decreases (UHF, UHR) or increases (τ_P). Retrained UHF@10 and UHR@10 are given for reference.

Dataset	SIDs	UHF@10 (%)↓	UHR@10 (%)	τ_P (%)↑	Retrained UHF / UHR
Beauty	RQ-KMeans	12.2 → 9.4 (1/5)	2.3 → 2.3 (2/5)	93.3 → 95.5 (5/5)	16.2 / 2.82
Beauty	LETTER	15.4 → 10.1 (3/5)	4.1 → 4.4 (3/5)	89.4 → 89.7 (4/5)	15.0 / 6.66
Sports	RQ-KMeans	27.6 → 11.1 (5/5)	1.4 → 0.5 (5/5)	91.1 → 92.4 (3/5)	29.3 / 1.42
Sports	LETTER	22.0 → 12.1 (5/5)	1.4 → 0.6 (5/5)	94.2 → 95.0 (3/5)	37.1 / 1.91
Toys	RQ-KMeans	5.0 → 9.7 (0/5)	2.3 → 2.4 (3/5)	92.1 → 95.1 (5/5)	18.1 / 3.51
Toys	LETTER	14.8 → 9.7 (3/5)	3.4 → 2.9 (5/5)	90.5 → 93.5 (5/5)	6.6 / 3.44

utility-preserving method on Sports (11.1%, from 49.8%), at 92.4% utility; there, even Finetune, without a forget term, stays below the threshold (93.0%).

Neighborhood spillover. To answer **RQ4**, Figure 2 compares, for the terms of the NAU objective, utility on the whole test set with utility on the forgotten target's SID neighborhood; Figure 1 makes the same comparison for the baselines. On the whole test set, all objectives shown lie within 0.2 points of each other (96.8 to 97.0% of retrained NDCG@10); on the neighborhood, they spread over 35 points. Fine-tuning on the retain and forget sets (Forget+Repair) drops to 87.0% on the neighborhood, and adding the separation term at $\lambda_s = 0.1$, which removes the promoted item completely, costs a further 14.3 points (95% interval $[-28.7, -2.8]$) with no whole-set change. NAU both removes the promoted item and gains 21.0 points on the neighborhood over Forget+Repair ($[+8.8, +35.6]$) at identical whole-set utility. With the stronger separation weight ($\lambda_s = 0.1$), the neighborhood term still recovers most of the separation loss, 23.0 points over the separation term alone ($[+7.6, +44.0]$), but ends at 95.7% of retrained neighborhood utility; λ_s thus trades strict removal of the promoted item against neighborhood preservation. Across datasets and target popularities, NAU's neighborhood gain over Forget+Repair is positive in eight of nine settings (Figure 3).

6 Related Work

GenRec with SIDs. GenRec formulates recommendation as generating an item identifier rather than scoring all items (Geng et al., 2022; Rajput et al., 2023), commonly an SID obtained by quantizing item representations (Hou et al., 2023; Rajput et al., 2023), as in TIGER (Rajput et al., 2023) and LETTER (Wang et al., 2024a). Recent work studies SID construction (Fu et al., 2026), generative retrieval objectives (Penha et al., 2025), scalability (Liu et al., 2026), long SIDs (Xia et al., 2026), and deployment (D'Amico et al., 2026), all under a fixed training set.

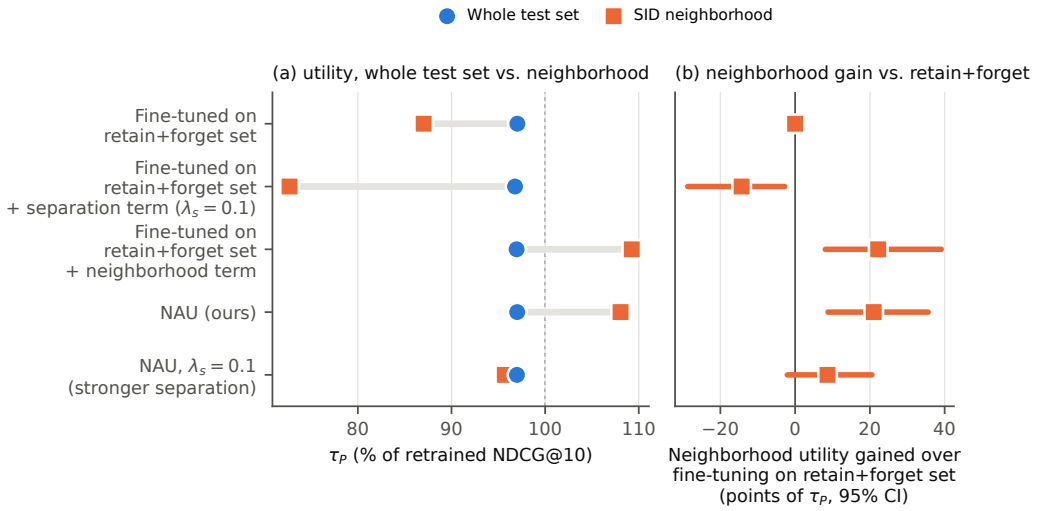


Figure 2: Effect of each objective term on the neighborhood (spam removal on Beauty, TIGER + RQ-KMeans, mid-popularity target, five seeds). (a) Utility as a percentage of the retrained model's, on the whole test set and on the target's SID neighborhood (523 test cases whose next item shares the target's first SID token). (b) Neighborhood utility gained over fine-tuning on the retain and forget sets, in percentage points, with 95% paired bootstrap intervals (Appendix G); intervals excluding 0 mark reliable differences. The last row uses a stronger separation weight ($\lambda_s = 0.1$) and is not selected (Table 6). Baselines are compared in Figure 1 and Table 1.

Machine unlearning for recommendation. Machine unlearning removes the influence of selected training data without retraining from scratch (Cao & Yang, 2015). Recommender unlearning has been studied for score-based, matrix-factorization, graph-based, and neural recommenders (Dong et al., 2024; Li et al., 2023; Wu et al., 2023a;b), including exact unlearning for specific architectures (Schelter et al., 2023), typically via influence estimation, gradient updates, retain-set repair, or model-specific approximations. These methods are not designed for SID-based GenRec, where each interaction supervises an autoregressive token sequence, and do not model how forgetting one item moves probability mass over items that share its tokens. TRACER (Chen et al., 2026) unlearns concepts in GenRec by reassigning the SIDs of the concept's items; we instead delete interactions while keeping users, items, and SIDs fixed, and control how probability is redistributed over the existing neighborhood.

7 Conclusion

We studied interaction-level unlearning for GenRec, where items are generated through SIDs and semantically related items may share identifier tokens. This makes it challenging to remove the influence of forgotten interactions without unnecessarily affecting neighboring items. We proposed NAU, a neighborhood-aware unlearning framework that combines SID-aware item-level unlearning with neighborhood mass control to regulate how unlearning affects nearby items. Across spam removal and unwanted item removal scenarios, NAU achieves effective forgetting while preserving recommendation utility, with the model architecture and SIDs kept fixed.

Limitations and future directions. Our study focuses on practical approximate unlearning for SID-based GenRec. One limitation is that we evaluate a representative set of deletion scenarios, while real systems may face more diverse and repeated deletion requests. Another limitation is that neighborhood construction is treated as a configurable design choice; future work could study adaptive neighborhoods that depend on the tokenizer, item representation space, and deletion goal. These directions can further strengthen unlearning methods that treat forgetting not only as exact-target removal, but as controlled local change in the item space.

AI DISCLOSURE STATEMENT

In this work, we used generative AI tools for support in implementing methods, assisting with translation, and support in qualitative and thematic data analysis. We have not used generative AI tools for generating synthetic datasets, helping develop theoretical models or conceptual frameworks, formulating mathematical claims, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, cleaning and reforming the dataset, interpreting results, and providing critical ingredients for proving mathematical claims, assisting in writing proofs are not applicable to this work. Additionally, we used generative AI tools for brainstorming, sourcing/searching for information, and editing a research paper to improve readability. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Krishna Acharya, Aleksandr V. Petrov, and Juba Ziani. GLoSS: Generative language models with semantic search for sequential recommendation, 2025. URL https://arxiv.org/abs/2506.01910.

Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *Proceedings of the 2015 IEEE Symposium on Security and Privacy*, SP '15, pp. 463–480, USA, 2015. IEEE Computer Society. ISBN 9781467369497. doi: 10.1109/SP.2015.35. URL https://doi.org/10.1109/SP.2015.35.

Chaochao Chen, Jiaming Zhang, Yizhao Zhang, Li Zhang, Lingjuan Lyu, Yuyuan Li, Biao Gong, and Chenggang Yan. CURE4Rec: A benchmark for recommendation unlearning with deeper influence. *Advances in Neural Information Processing Systems*, 37:99128–99144, 2024.

Chong Chen, Fei Sun, Min Zhang, and Bolin Ding. Recommendation unlearning. In *Proceedings of the ACM Web Conference 2022*, pp. 2768–2777. Association for Computing Machinery, 2022.

Ziheng Chen, Jiali Cheng, Zezhong Fan, Hadi Amiri, Diyuan Wu, Gabriele Tolomei, and Yang Zhang. TRACER: Token reassignment for concept erasure in generative recommendation, 2026. URL https://arxiv.org/abs/2606.07688.

Wenzhuo Cheng, Menghang Gong, Qixin Guo, Hang Zheng, Zhaobin Yang, Jianguo Lou, and Zhengwei Zheng. CapsID: Soft-routed variable-length semantic ids for generative recommendation, 2026. URL https://arxiv.org/abs/2605.05096.

Liam Collins, Jiwen Ren, Donald Loveland, Bhuvesh Kumar, Clark Mingxuan Ju, Xuan Guo, Mo Li, Alvin Hou, Yi Cui, Peng Yang, Jian Wang, Saud Afzal Shafi, Nga Than, Ruiming Lu, Wenfeng Zhuo, Dongheng Li, Lili Zhang, Mingtao Zhang, Jinchao Ye, Vincent Xue, Chunhui Zhu, and Neil Shah. Llm-based generative retrieval for snapchat content recommendation, 2026. URL https://arxiv.org/abs/2607.28895.

Edoardo D'Amico, Marco De Nadai, Praveen Chandar, Divita Vohra, Shawn Lin, Max Lefarov, Paul Gigioli, Gustavo Penha, Ilya Kopysitsky, Ivo Joel Senese, Darren Mei, Francesco Fabbri, Oguz Semerci, Yu Zhao, Vincent Tang, Brian St. Thomas, Alexandra Ranieri, Matthew N. K. Smith, Aaron Bernkopf, Bryan Leung, Ghazal Fazelnia, Mark VanMiddlesworth, Timothy Christopher Heath, Petter Pehrson Skiden, Alice Y. Wang, Doug J. Cole, Andreas Damianou, Maya Hristakeva, Reid Wilbur, Tarun Chillara, Vladan Radosavljevic, Pooja Chitkara, Sainath Adapa, Juan Elenter, Bernd Huber, Jacqueline Wood, Saaketh Vedantam, Jan Stypka, Sandeep Ghael, Martin D. Gould, David Murgatroyd, Yves Raimond, Mounia Lalmas, and Paul N. Bennett. Deploying semantic ID-based generative retrieval for large-scale podcast discovery at Spotify, 2026. URL https://arxiv.org/abs/2603.17540.

Yushun Dong, Binchi Zhang, Zhenyu Lei, Na Zou, and Jundong Li. IDEA: A flexible framework of certified unlearning for graph neural networks. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 621–630. Association for Computing Machinery, 2024. URL https://doi.org/10.1145/3637528.3671744.

Junchen Fu, Xuri Ge, Alexandros Karatzoglou, Ioannis Arapakis, Suzan Verberne, Joemon M. Jose, and Zhaochun Ren. Differentiable semantic ID for generative recommendation. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 369–379. Association for Computing Machinery, 2026. URL https://doi.org/10.1145/3805712.3809641.

Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. Recommendation as language processing (RLP): A unified pretrain, personalized prompt & predict paradigm (P5). In *Proceedings of the 16th ACM Conference on Recommender Systems*, pp. 299–315, 2022. doi: 10.1145/3523227.3546767.

Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web*, pp. 507–517, 2016. doi: 10.1145/2872427.2883037.

Yupeng Hou, Zhankui He, Julian McAuley, and Wayne Xin Zhao. Learning vector-quantized item representation for transferable sequential recommenders. In *Proceedings of the ACM Web Conference 2023*, WWW '23, pp. 1162–1171. Association for Computing Machinery, 2023. URL https://doi.org/10.1145/3543507.3583434.

Jinghan Jia, Jiancheng Liu, Parikshit Ram, Yuguang Yao, Gaowen Liu, Yang Liu, Pranay Sharma, and Sijia Liu. Model sparsity can simplify machine unlearning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 51584–51605. Curran Associates, Inc., 2023. doi: 10.52202/075280-2246.

Clark Mingxuan Ju, Liam Collins, Leonardo Neves, Bhuvesh Kumar, Louis Yufeng Wang, Tong Zhao, and Neil Shah. Generative recommendation with semantic IDs: A practitioner's handbook. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, CIKM '25, pp. 6420–6425. Association for Computing Machinery, 2025. URL https://doi.org/10.1145/3746252.3761612.

Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. *arXiv preprint arXiv:1808.09781*, 2018. URL https://arxiv.org/abs/1808.09781.

Tongyoung Kim, SooJin Yoon, SeongKu Kang, Jinyoung Yeo, and Dongha Lee. MVIGER: Multi-view variational integration of complementary knowledge for generative recommender. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 834–844. Association for Computing Machinery, 2026. URL https://doi.org/10.1145/3805712.3809591.

Meghdad Kurmanji, Peter Triantafillou, Jamie Hayes, and Eleni Triantafillou. Towards unbounded machine unlearning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Haoyang Li, Shimin Di, and Lei Chen. Revisiting injective attacks on recommender systems. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 29989–30002. Curran Associates, Inc., 2022. doi: 10.52202/068431-2174.

Ying Li, Shradha Sehgal, Arjun Rao, Rein Houthooft, Yaochen Zhu, and Ashish Rastogi. GenRec: An LLM-backed recommendation ranker at Netflix, 2026. URL https://arxiv.org/abs/2608.10257.

Yuyuan Li, Chaochao Chen, Xiaolin Zheng, Yizhao Zhang, Biao Gong, Jun Wang, and Linxun Chen. Selective and collaborative influence function for efficient recommendation unlearning. *Expert Syst. Appl.*, 234(C), December 2023. URL https://doi.org/10.1016/j.eswa.2023.121025.

Jingzhe Liu, Liam Collins, Jiliang Tang, Tong Zhao, Neil Shah, and Mingxuan Ju. Understanding generative recommendation with semantic ids from a model-scaling view. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, pp. 3234–3245. Association for Computing Machinery, 2026. URL https://doi.org/10.1145/3770855.3817976.

Pierre Sicco Lubitzsch, Maarten de Rijke, and Sebastian Schelter. ERASE – a real-world aligned benchmark for unlearning in recommender systems. In *SIGIR 2026: 49th international ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 3310–3318. ACM, July 2026. URL https://doi.org/10.1145/3805712.3808596.

Xinchen Luo, Jiangxia Cao, Tianyu Sun, Jinkai Yu, Rui Huang, Wei Yuan, Hezheng Lin, Yichen Zheng, Shiyao Wang, Qigen Hu, Changqing Qiu, Jiaqi Zhang, Xu Zhang, Zhiheng Yan, Jingming Zhang, Simin Zhang, Mingxing Wen, Zhaojie Liu, and Guorui Zhou. Qarm: Quantitative alignment multi-modal recommendation at kuaishou. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pp. 5915–5922, New York, NY, USA, 2025. Association for Computing Machinery. URL https://doi.org/10.1145/3746252.3761502.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 43–52, 2015. doi: 10.1145/2766462.2767755.

Gustavo Penha, Edoardo D'Amico, Marco De Nadai, Enrico Palumbo, Alexandre Tamborrino, Ali Vardasbi, Max Lefarov, Shawn Lin, Timothy Heath, Francesco Fabbri, and Hugues Bouchard. Semantic IDs for joint generative search and recommendation. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems*, pp. 1296–1301. Association for Computing Machinery, 2025. URL https://doi.org/10.1145/3705328.3759300.

Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Tran, Jonah Samost, Maciej Kula, Ed Chi, and Maheswaran Sathiamoorthy. Recommender systems with generative retrieval. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 10299–10315. Curran Associates, Inc., 2023. doi: 10.52202/075280-0452.

Sebastian Schelter, Mozhdeh Ariannezhad, and Maarten de Rijke. Forget me now: Fast and exact unlearning in neighborhood-based recommendation. In *Proceedings of the 46th international ACM SIGIR conference on research and development in information retrieval*, pp. 2011–2015, 2023. URL https://doi.org/10.1145/3539618.3591989.

Eleni Triantafillou, Peter Kairouz, Fabian Pedregosa, Jamie Hayes, Meghdad Kurmanji, Kairan Zhao, Vincent Dumoulin, Julio Jacques Junior, Ioannis Mitliagkas, Jun Wan, Lisheng Sun Hosoya, Sergio Escalera, Gintare Karolina Dziugaite, Peter Triantafillou, and Isabelle Guyon. Are we making progress in unlearning? Findings from the first NeurIPS unlearning competition, 2024. URL https://arxiv.org/abs/2406.09073.

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. Association for Computing Machinery, 2024a. URL https://doi.org/10.1145/3627673.3679569.

Ye Wang, Jiahao Xun, Minjie Hong, Jieming Zhu, Tao Jin, Wang Lin, Haoyuan Li, Linjun Li, Yan Xia, Zhou Zhao, and Zhenhua Dong. EAGER: Two-stream generative recommender with behavior-semantic collaboration. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 3245–3254. Association for Computing Machinery, 2024b. URL https://doi.org/10.1145/3637528.3671775.

Chenwang Wu, Defu Lian, Yong Ge, Zhihao Zhu, and Enhong Chen. Triple adversarial learning for influence based poisoning attack in recommender systems. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 1830–1840. Association for Computing Machinery, 2021. URL https://doi.org/10.1145/3447548.3467335.

Jiancan Wu, Yi Yang, Yuchun Qian, Yongduo Sui, Xiang Wang, and Xiangnan He. GIF: A general graph unlearning strategy via influence function. In *Proceedings of the ACM Web Conference 2023*, WWW '23, pp. 651–661. ACM, April 2023a. URL http://dx.doi.org/10.1145/3543507.3583521.

Kun Wu, Jie Shen, Yue Ning, Ting Wang, and Wendy Hui Wang. Certified edge unlearning for graph neural networks. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2606–2617. Association for Computing Machinery, 2023b. URL https://doi.org/10.1145/3580305.3599271.

Ming Xia, Guoxin Ma, Zhiqin Zhou, and Dongmin Huang. Unleash the potential of long semantic IDs for generative recommendation. *arXiv preprint arXiv:2602.13573*, 2026. URL https://arxiv.org/abs/2602.13573.

Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. SoundStream: An end-to-end neural audio codec. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 30: 495–507, November 2021. URL https://doi.org/10.1109/TASLP.2021.3129994.

APPENDIX OVERVIEW

- Appendix A: dataset statistics and training set construction.
- Appendix B: baseline descriptions.
- Appendix C: construction of the spam and unwanted-item deletion scenarios.
- Appendix D: efficiency, NAU variants, and spam removal with RQ-VAE, LETTER, and TRACER tokenizers.
- Appendix E: per-term ablations on Sports and Toys.
- Appendix F: full unwanted-item results and the neighborhood-weight ablation.
- Appendix G: how the intervals for neighborhood comparisons are computed.
- Appendix H: SID prefix-sharing statistics and level-restricted updates.

A DATASET STATISTICS

Table 5 summarizes the dataset statistics after preprocessing. Following TIGER, a user's history is capped at 20 items during training. The statistics are computed before this cap. For each dataset, we follow the next-item prediction setting, where each training example is represented as $(\boldsymbol{x}, i)$, with $\boldsymbol{x}$ denoting the user history and i the next item. Interactions are split into training, validation, and test sets in chronological order. The backbone GenRec model is trained to autoregressively generate the SID of the next item. All unlearning methods start from the same original model $\boldsymbol{\theta}_o$. During evaluation, valid generated SIDs are mapped back to items and ranked by generation probability.

Table 5: Dataset statistics after preprocessing.

Dataset	#Users	#Items	#Interactions	Avg. length
Beauty	22,363	12,101	198,502	8.88
Sports	35,598	18,357	296,337	8.32
Toys	19,412	11,924	167,597	8.63

B BASELINES

We compare with representative removal and unlearning baselines: (i) *Original*, the model $\boldsymbol{\theta}_o$ before unlearning; (ii) *Retrain*, the reference $\boldsymbol{\theta}_r$ trained from scratch on D_r; (iii) *Finetune* (retain only), which fine-tunes $\boldsymbol{\theta}_o$ on D_r; (iv) *Forget only*, gradient ascent on D_f; (v) *Forget+Repair*, gradient ascent on D_f combined with fine-tuning on D_r; (vi) *SCIF* (Li et al., 2023), which applies an influence-function update to a subset of the parameters to remove the forget set while keeping retain performance; (vii) *SEIF*, which adds noise to $\boldsymbol{\theta}_o$ and then fine-tunes on D_r; (viii) *Kookmin*, which reinitializes a subset of the parameters and fine-tunes on D_r; (ix) *Fanchuan*, which first pushes the predictions on forgotten examples toward a uniform distribution and then alternates a contrastive forget objective with fine-tuning on D_r (SEIF, Kookmin, and Fanchuan are from the NeurIPS 2023 unlearning competition (Triantafillou et al., 2024)); and (x) *TRACER* (Chen et al., 2026), which reassigns the SIDs of the items to be forgotten and regularizes the recommender toward the original identifiers.

Selection of NAU's weights. Table 6 reports the validation grid behind NAU's operating point. Whole-set exposure barely separates the settings: among settings of the full objective ($\lambda_s, \lambda_n > 0$) that keep $\tau_P \geq 95\%$, $\lambda_s = 0.1$ has the lowest $\Delta_{\mathrm{SH}}^{\mathrm{retr}}$ (0.079%), but $\lambda_s = 0.01$ (0.086%) is identical to it on 12 of 15 cells and their paired difference is not resolved. Among such tied settings we select the highest combined utility, the mean of whole-set and neighborhood τ_P: $\lambda_s = 0.01$ reaches 100.2% against 95.8% (103.6% against 95.0% on the neighborhood) and is ahead on both parts, so the choice does not depend on their weighting. Settings without the separation term ($\lambda_s = 0$) are ablations of NAU rather than candidates; with $\lambda_n = 0.01$ such a setting reaches a similar combined utility (101.7%).

The forget weight was checked in the same way: without the separation and neighborhood terms, $\lambda_f = 0.01$ barely removes the promoted item (validation $\Delta_{\mathrm{SH}}^{\mathrm{retr}}$ of 0.832%, close to the poisoned

Table 6: Selection of NAU's weights on the validation split (Beauty, TIGER + RQ-KMeans, $\lambda_f = 0.1$, three targets $\times$ five data seeds). Whole set: $\Delta_{\mathrm{SH}}^{\mathrm{retr}}$ and τ_P on all validation cases; neighborhood: the same on the validation cases whose next item shares the target's first SID token (the target's own cases excluded), with τ_P as a ratio of pooled NDCG@10. Selection is among settings of the full objective ($\lambda_s, \lambda_n > 0$) that keep whole-set $\tau_P \geq 95\%$: settings whose whole-set $\Delta_{\mathrm{SH}}^{\mathrm{retr}}$ is not resolvably different from the lowest one (paired 95% interval over the five seeds includes 0) are treated as tied, and among them the highest combined utility, the mean of whole-set and neighborhood τ_P, is selected (✓); the choice is the same for any weighting of the two. Settings below 95% whole-set utility are shown in gray; – marks settings not scored on the neighborhood.

		Whole set		SID neighborhood		Combined	
λ_s	λ_n	$\Delta_{\mathrm{SH}}^{\mathrm{retr}}$ (%) ↓	τ_P (%) ↑	$\Delta_{\mathrm{SH}}^{\mathrm{retr}}$ (%) ↓	τ_P (%) ↑	τ_P (%) ↑	Sel.
0.0	0.0	0.112	96.9	0.678	86.5	91.7	
0.0	0.01	0.089	96.7	0.863	106.6	101.7	
0.0	0.1	0.094	93.7	0.874	157.9	125.8	
0.01	0.0	0.086	97.0	0.879	87.6	92.3	
0.01	0.01	0.086	96.9	0.874	103.6	100.2	✓
0.01	0.1	0.072	93.9	–	–	–	
0.1	0.0	0.079	96.8	0.874	80.2	88.5	
0.1	0.01	0.079	96.6	0.874	95.0	95.8	
0.1	0.1	0.079	94.2	0.874	147.4	120.8	

model) and $\lambda_f = 1.0$ removes it but keeps only 76.3% of retrained utility, while $\lambda_f = 0.1$ reaches 0.112% at 96.9%.

Selection of baseline configurations. Table 7 lists every baseline configuration run on all five seeds and three targets, with validation and test metrics; the selected configurations are those in Table 1, and validation and test agree closely.

For unwanted-item removal, Table 8 applies the same procedure with UHF@10 in place of $\Delta_{\mathrm{SH}}^{\mathrm{retr}}$. NAU's λ_n is chosen on Beauty: $\lambda_n = -1$ has the lowest validation UHF@10 (9.9%, against 11.4% at $\lambda_n = -10$ and 13.0% at $\lambda_n = 0$). No setting of NAU or of SEIF, Kookmin and Fanchuan keeps validation $\tau_P \geq 95\%$ (NAU reaches 93.3%), so the lowest UHF@10 decides.

C Deletion Scenario Construction

Spam removal. We inject a bandwagon attack before training. Items are ranked by interaction count, and one target is drawn per attack from the bottom 20% (unpopular), the 30th to 70th percentile (mid), or the top 5% (popular) of this ranking. The attack adds fake users equal to 1% of the real users (226 on Beauty, 360 on Sports, 197 on Toys). Each fake session has a length drawn from a lognormal-Poisson fit to the real session lengths, scaled by 0.8; it contains the target once, at a random position between 20% and 90% of the session, and every other position is a filler item drawn uniformly from the 20% most popular items. The forget set D_f is all interactions of the fake users. We build one attack per (dataset, data seed, popularity stratum), with data seeds $\{2, 3, 5, 7, 11\}$, and train the poisoned model $\boldsymbol{\theta}_o$ and the retrained reference $\boldsymbol{\theta}_r$ (on the unattacked data) for each.

Unwanted-item removal. Following ERASE (Lubitzsch et al., 2026), the unwanted set $\mathcal{I}_{\mathrm{u}}$ is a sensitive category: Hair Loss Products on Beauty (taxonomy level 2, 106 items, 0.88% of the catalog), Guns & Rifles on Sports (level 3, 202 items, 1.10%), and toy weapons on Toys (68 items, 0.57%, matched by the keywords gun, rifle, pistol, weapon, sword, knife, and dagger and their plurals). Users who interacted with the category are shuffled with the data seed and added one at a time until their interactions with it reach 0.01% of all interactions, which gives 12 to 14 requesting users per seed. The forget set is the requesting users' interactions with $\mathcal{I}_{\mathrm{u}}$; their other interactions stay in the retain set. The retrained reference is trained on the training data with exactly these interactions removed.

D Additional Spam-Removal Results

All tables follow the selection rule of the main results (the smallest $\Delta_{\mathrm{SH}}^{\mathrm{retr}}$ among configurations with $\tau_P \geq 95\%$); validation runs were made for TIGER + RQ-KMeans, so for the other tokenizers the rule is applied on the test split.

Relative efficiency. Table 9 reports τ_E, the fraction of retraining cost saved, on Beauty; every selected configuration saves more than 99.7%.

Additional NAU variants. Table 10 compares NAU variants on Beauty that differ in the separation score and in how the neighborhood and separation weights are chosen.

TIGER with RQ-VAE SIDs. Tables 11 to 13 report spam-removal results when SIDs are constructed using RQ-VAE. For unpopular and mid-popularity targets, NAU is closest to retrained exposure in most blocks while keeping $\tau_P \geq 95\%$. For popular targets, which the retrained model still recommends, NAU suppresses the target almost completely and is farther from retraining than the best baselines; the separation term is the likely cause (Table 19), and a target-dependent λ_s is a natural extension.

LETTER-style identifiers. Tables 14 to 16 report results with LETTER-style identifiers. These results further test whether the observed behavior depends on a specific SID construction. NAU removes the promoted target in every stratum; Forget+Repair often matches its utility and its distance to retrained exposure, so the advantage on these tokenizers lies in removal rather than utility.

TRACER-compatible tokenization. Table 17 reports results under the TRACER-compatible tokenization setting. This comparison provides an additional robustness check under an identifier construction aligned with the TRACER baseline.

E Additional Ablation Results

This section provides additional per-term ablations of the NAU objective on Sports and Toys. Table 18 is the Toys counterpart of Table 2. The per-term tables for Sports and Toys (Tables 20 and 21) use the earlier protocol (all strata, removal and utility on the whole test set only) with the stronger separation weight ($\lambda_s = 0.1$); Table 18 includes the reported setting. Lower $\mathrm{SH@10}$ indicates stronger removal of promoted targets, while higher $\mathrm{NDCG@10}$ and τ_P indicate better utility preservation.

On the whole test set, the forget term carries most of the removal, separation drives it to zero or near zero, and $\lambda_n = 0.01$ leaves utility unchanged while $\lambda_n = 0.1$ costs 2 to 7 points of τ_P.

Probability-mass ablation. Table 19 repeats the Beauty ablation over all three popularity strata and five seeds and reports the local effect in log probability mass instead of neighborhood utility; beyond prefix-disjoint controls, the neighborhood's mass relative to retraining grows with λ_n ($+0.04$ without the term, $+0.18$ for NAU, $+1.55$ at $\lambda_n = 0.1$), ($+0.04$ without the term, $+0.33$ for NAU, $+1.55$ at $\lambda_n = 0.1$), and the separation term pushes the target below its retrained mass (about 4 nats for NAU, about 18.5 with the stronger $\lambda_s = 0.1$), which is also why NAU can remove popular targets that the retrained model still recommends (Appendix D).

Neighborhood effect across datasets and targets. Figure 3 repeats the comparison of Figure 2 for all three datasets and target popularities. NAU's neighborhood gain over Forget+Repair is positive in eight of nine settings and resolved for the unpopular and mid-popularity Beauty targets; with the stronger separation weight ($\lambda_s = 0.1$), the neighborhood term's gain over the separation term alone is positive in all nine settings and resolved in five.

F Additional Unwanted-Item Results

This section reports full unwanted-item deletion results against existing baselines. They complement Section 5, where Table 3 compares methods on Beauty and Table 4 isolates the neighborhood term on all three datasets. Each method is at the configuration selected on the validation split (Table 8).

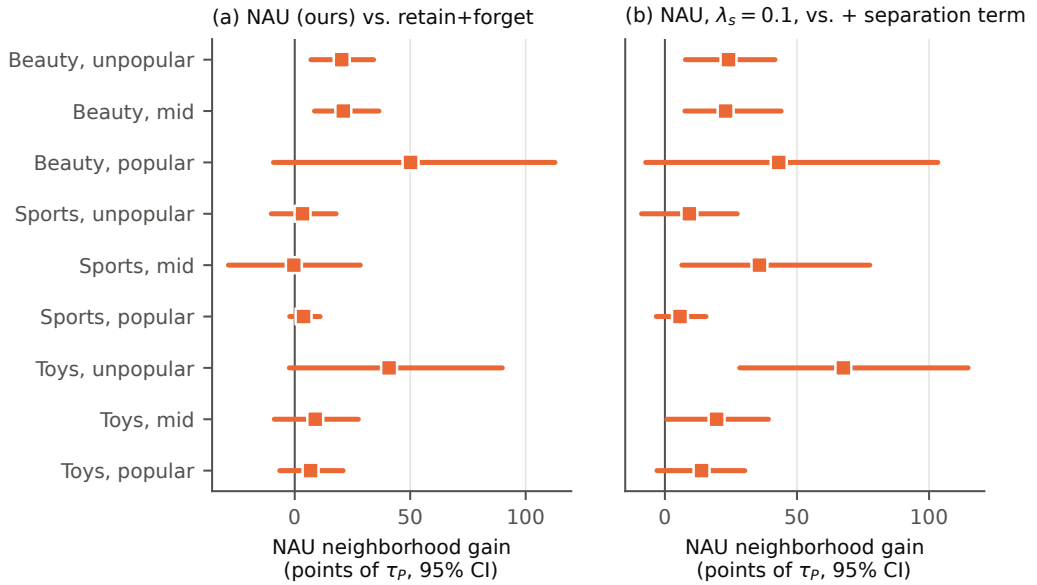


Figure 3: Neighborhood utility gained on each dataset and target popularity (TIGER + RQ-KMeans, five seeds per setting), in points of the retrained model's neighborhood utility: (a) NAU over fine-tuning on the retain and forget sets; (b) NAU with the stronger separation weight ($\lambda_s = 0.1$) over fine-tuning on the retain and forget sets with the separation term ($\lambda_s = 0.1$). Bars are 95% paired bootstrap intervals over the shared test cases (Appendix G); the neighborhood excludes the target's own test cases.

UHF@10 measures unwanted-item exposure for users who requested deletion (lower is better), while UHR@10 measures it for the remaining users, and $\Delta^{\text{retr}}_{\text{UHR}}$ how far the update moves them from retraining. On Beauty and Toys the retrained model does not lower UHF@10 below the original model, since only 12 to 14 users request deletion, so we report UHF@10 itself rather than its distance to retraining.

G Intervals for Neighborhood Comparisons

Neighborhood utility is estimated from few test cases: on Beauty, 523 test cases have a next item in the forgotten target's SID neighborhood, and only about 30 of them have that item in the retrained model's top 10. A handful of test users can therefore move a neighborhood result by several points, and a difference between two methods is meaningful only if it survives that variation.

We measure this variation with a paired bootstrap. For every neighborhood test case q and a method A compared against a reference method B, we compute the difference $d_q = \text{NDCG@10}_A(q) - \text{NDCG@10}_B(q)$. Both methods are scored on the same test cases, which is what makes the comparison paired: differences in how easy a test case is cancel out. The reported gain is the mean of d_q over all cases, expressed in percentage points of the retrained model's neighborhood NDCG@10. To see how much this mean would change with a different set of test users, we draw as many cases as we have, at random and with replacement, recompute the mean on this resampled set, and repeat 3000 times. The 95% interval spans the middle 95% of these 3000 means, i.e., from the 2.5th to the 97.5th percentile.

An interval that lies entirely above 0 means that method A gains neighborhood utility over B no matter which of the plausible test sets we happen to have; an interval entirely below 0 means a reliable loss; an interval that contains 0 means that the data cannot tell the two methods apart. The interval reflects the sampling of test cases for the trained and unlearned models at hand, pooled over the five data seeds; it does not resample the seeds themselves. As a complementary check, the sign of the NAU gain over fine-tuning on the retain and forget sets holds on every seed that has

more than three neighborhood test cases (4 of 5). Comparisons against the retrained model itself are not resolved at this sample size (intervals of about ± 40 points), because the retrained model is a separately trained network whose rankings differ from the unlearned models' on most neighborhood test cases; we therefore only compare unlearning methods with each other.

H SID AND NEIGHBORHOOD DIAGNOSTICS

This section provides additional diagnostics for the SID structure and its role in unlearning. Table 24 analyzes how many items share SID prefixes at different levels, while Table 25 studies how restricting updates to different SID levels affects forgetting and utility.

SID prefix-sharing statistics. Table 24 reports neighborhood statistics across SID levels and tokenizers. The results show that early SID levels induce broad neighborhoods, while later levels often become highly sparse or singleton-dominated. This supports our choice to treat neighborhood construction as a configurable component rather than assuming that every SID level provides a meaningful neighborhood.

Level-restricted unlearning updates. Table 25 reports a diagnostic experiment that restricts the unlearning update to different SID levels. This analysis complements the main objective ablation by showing how different parts of the SID representation contribute to forgetting and utility preservation. The results indicate that effective unlearning is not only determined by the target item itself, but also by which identifier levels and model parameters are allowed to change.

Table 7: Hyperparameter selection on the validation split (TIGER + RQ-KMeans; TRACER on Beauty with its own tokenizer). For every candidate configuration, pooled over five data seeds and three targets: distance to the retrained model's exposure $\Delta_{\mathrm{SH}}^{\mathrm{retr}}$ and τ_P, on the validation split (used for selection) and on the test split (reported in Table 1). The selected configuration (✓) has the smallest validation $\Delta_{\mathrm{SH}}^{\mathrm{retr}}$ among those with validation $\tau_P \geq 95\%$, ties broken by higher τ_P; if none reaches 95%, the smallest validation $\Delta_{\mathrm{SH}}^{\mathrm{retr}}$ overall. Finetune and SCIF have a single configuration; NAU's selection is shown in Table 6.

			Validation		Test		
Dataset	Method	Configuration	$\Delta_{\mathrm{SH}}^{\mathrm{retr}} \downarrow$	$\tau_P \uparrow$	$\Delta_{\mathrm{SH}}^{\mathrm{retr}} \downarrow$	$\tau_P \uparrow$	Sel.
Beauty	Fanchuan	fanchuan_t0.007	0.079	11.9	0.072	12.6	
Beauty	Fanchuan	fanchuan_t0.07	0.079	32.0	0.072	32.4	✓
Beauty	Forget only	unified_abl_lf0.1_ls0.0_ln0.0_lr0.0	0.079	2.1	0.072	1.9	
Beauty	Forget only	unified_abl_lf1.0_ls0.0_ln0.0_lr0.0	0.079	2.7	0.072	2.5	✓
Beauty	Forget+Repair	unified_abl_lf1.0_ls0.0_ln0.0	0.079	75.8	0.072	72.8	
Beauty	Forget+Repair	unified_abl_lf0.1_ls0.0_ln0.0	0.108	96.4	0.106	95.8	✓
Beauty	Kookmin	kookmin_ir0.0001	0.643	82.0	0.575	81.0	
Beauty	Kookmin	kookmin_ir0.001	0.550	81.9	0.493	80.9	✓
Beauty	Kookmin	kookmin_ir0.01	0.846	74.8	0.775	75.0	
Beauty	SEIF	seif_std0.06	0.258	66.1	0.258	66.4	✓
Beauty	SEIF	seif_std0.006	0.278	68.0	0.256	68.6	
Beauty	TRACER	tracer_lf1.0_lc0.1_plr0.001_ep4	1.185	98.2	1.063	99.1	✓
Beauty	TRACER	tracer_lf1.0_lc1.0_plr0.001_ep4	4.262	98.5	3.546	99.3	
Beauty	TRACER	tracer_lf0.1_lc1.0_plr0.001_ep4	5.821	98.6	4.848	99.5	
Beauty	TRACER	tracer_lf0.1_lc0.1_plr0.001_ep4	1.520	98.5	1.323	99.5	
Sports	Fanchuan	fanchuan_t0.07	0.146	30.5	0.139	31.7	✓
Sports	Fanchuan	fanchuan_t0.007	0.146	11.7	0.139	12.8	
Sports	Forget only	unified_abl_lf0.1_ls0.0_ln0.0_lr0.0	0.146	0.7	0.139	0.5	
Sports	Forget only	unified_abl_lf1.0_ls0.0_ln0.0_lr0.0	0.146	0.8	0.139	0.7	✓
Sports	Forget+Repair	unified_abl_lf0.1_ls0.0_ln0.0	0.142	99.8	0.139	102.1	✓
Sports	Forget+Repair	unified_abl_lf1.0_ls0.0_ln0.0	0.146	68.3	0.139	69.5	
Sports	Kookmin	kookmin_ir0.001	0.177	79.3	0.165	80.3	✓
Sports	Kookmin	kookmin_ir0.01	0.252	73.3	0.224	72.5	
Sports	Kookmin	kookmin_ir0.0001	0.242	79.8	0.203	80.8	
Sports	SEIF	seif_std0.006	0.157	60.5	0.144	61.4	
Sports	SEIF	seif_std0.06	0.122	58.7	0.112	59.5	✓
Sports	TRACER	tracer_lf1.0_lc0.1_plr0.001_ep4	0.462	97.0	0.410	99.8	✓
Sports	TRACER	tracer_lf0.1_lc0.1_plr0.001_ep4	0.694	97.0	0.602	99.8	
Sports	TRACER	tracer_lf0.1_lc1.0_plr0.001_ep4	3.877	96.9	3.488	99.3	
Sports	TRACER	tracer_lf1.0_lc1.0_plr0.001_ep4	3.312	96.6	2.949	99.2	
Toys	Fanchuan	fanchuan_t0.07	0.073	21.4	0.074	23.4	✓
Toys	Fanchuan	fanchuan_t0.007	0.073	3.2	0.074	3.5	
Toys	Forget only	unified_abl_lf0.1_ls0.0_ln0.0_lr0.0	0.073	0.3	0.074	0.4	
Toys	Forget only	unified_abl_lf1.0_ls0.0_ln0.0_lr0.0	0.073	0.5	0.074	0.4	✓
Toys	Forget+Repair	unified_abl_lf0.1_ls0.0_ln0.0	0.164	97.0	0.185	96.7	✓
Toys	Forget+Repair	unified_abl_lf1.0_ls0.0_ln0.0	0.073	69.6	0.074	69.8	
Toys	Kookmin	kookmin_ir0.001	0.403	75.5	0.397	74.7	
Toys	Kookmin	kookmin_ir0.01	0.592	68.2	0.573	68.9	
Toys	Kookmin	kookmin_ir0.0001	0.384	76.8	0.385	76.1	✓
Toys	SEIF	seif_std0.06	0.206	60.4	0.216	61.3	
Toys	SEIF	seif_std0.006	0.183	62.6	0.206	63.1	✓
Toys	TRACER	tracer_lf1.0_lc1.0_plr0.001_ep4	2.555	96.7	2.293	96.2	
Toys	TRACER	tracer_lf1.0_lc0.1_plr0.001_ep4	1.150	96.6	1.079	95.7	✓
Toys	TRACER	tracer_lf0.1_lc1.0_plr0.001_ep4	3.327	97.1	2.999	96.6	
Toys	TRACER	tracer_lf0.1_lc0.1_plr0.001_ep4	1.471	97.1	1.377	96.2	

Table 8: Hyperparameter selection for unwanted-item removal on the validation split (TIGER + RQ-KMeans). For every candidate configuration, averaged over five data seeds: UHF@10 and τ_P on the validation split (used for selection) and on the test split (reported in Appendix F). The selected configuration (✓) has the lowest validation UHF@10 among those with validation $\tau_P \geq 95\%$; if none reaches 95%, the lowest validation UHF@10 overall. NAU ($\lambda_f = 0.1$, $\lambda_s = 0$) selects λ_n once on the Beauty validation split with the same rule and uses it on all datasets. Finetune, Forget only and Forget+Repair have a single configuration.

			Validation		Test		
Dataset	Method	Configuration	UHF@10 ↓	τ_P ↑	UHF@10 ↓	τ_P ↑	Sel.
Beauty	SCIF	scif_d0.001	24.99	99.6	15.98	101.0	
Beauty	SCIF	scif_d0.01	24.99	99.6	15.98	101.0	✓
Beauty	SCIF	scif_d0.1	26.53	99.6	15.98	100.9	
Beauty	SEIF	seif_std0.006	15.50	60.4	10.06	63.9	
Beauty	SEIF	seif_std0.06	17.04	59.2	8.25	62.0	
Beauty	SEIF	seif_std0.6	0.00	0.8	0.00	0.9	✓
Beauty	Kookmin	kookmin_ir0.0001	19.81	81.9	7.99	85.5	
Beauty	Kookmin	kookmin_ir0.001	9.62	80.5	4.90	84.6	
Beauty	Kookmin	kookmin_ir0.01	8.23	73.1	6.71	77.4	✓
Beauty	Fanchuan	fanchuan_t0.007	0.00	4.1	0.00	5.0	✓
Beauty	Fanchuan	fanchuan_t0.07	0.00	10.4	0.00	10.8	
Beauty	Fanchuan	fanchuan_t1.15	0.00	21.6	0.00	23.0	
Beauty	NAU	$\lambda_n = -0.1$	13.10	91.0	11.07	93.9	
Beauty	NAU	$\lambda_n = -1$	9.90	93.3	9.42	95.8	✓
Beauty	NAU	$\lambda_n = -10$	11.43	93.6	7.75	95.8	
Beauty	NAU	$\lambda_n = 0$	12.97	90.1	7.71	93.2	
Sports	SCIF	scif_d0.001	44.91	100.0	43.71	98.9	
Sports	SCIF	scif_d0.01	38.37	101.3	32.87	100.4	
Sports	SCIF	scif_d0.1	38.29	101.3	35.73	100.5	✓
Sports	SEIF	seif_std0.006	22.19	44.8	15.32	46.2	
Sports	SEIF	seif_std0.06	17.89	44.6	19.07	45.7	
Sports	SEIF	seif_std0.6	0.00	0.0	0.00	0.1	✓
Sports	Kookmin	kookmin_ir0.0001	31.85	75.2	29.90	74.7	
Sports	Kookmin	kookmin_ir0.001	31.61	74.2	29.90	74.4	
Sports	Kookmin	kookmin_ir0.01	28.74	68.8	27.05	68.7	✓
Sports	Fanchuan	fanchuan_t0.007	0.00	7.6	0.00	7.4	✓
Sports	Fanchuan	fanchuan_t0.07	0.00	16.3	0.00	17.3	
Sports	Fanchuan	fanchuan_t1.15	0.00	21.1	0.00	22.0	
Sports	TRACER	tracer_lf0.1_lc0.1_ep4	34.32	101.4	32.87	100.6	
Sports	TRACER	tracer_lf0.1_lc1.0_ep4	37.19	101.5	39.51	100.6	
Sports	TRACER	tracer_lf1.0_lc0.1_ep4	34.32	101.2	32.87	100.4	✓
Sports	TRACER	tracer_lf1.0_lc1.0_ep4	37.19	101.4	36.65	100.6	
Sports	NAU	$\lambda_n = -0.1$	25.02	91.5	19.68	90.9	
Sports	NAU	$\lambda_n = -1$	15.72	93.6	12.68	92.5	✓
Sports	NAU	$\lambda_n = -10$	12.72	94.3	9.97	93.9	
Sports	NAU	$\lambda_n = 0$	31.13	92.1	27.64	91.0	
Toys	SCIF	scif_d0.001	23.12	102.2	11.15	103.3	
Toys	SCIF	scif_d0.01	23.12	102.1	11.15	103.3	✓
Toys	SCIF	scif_d0.1	23.12	102.1	11.15	103.3	
Toys	SEIF	seif_std0.006	8.36	54.1	6.54	55.0	
Toys	SEIF	seif_std0.06	10.02	53.5	6.67	54.1	
Toys	SEIF	seif_std0.6	0.00	0.5	0.00	1.2	✓
Toys	Kookmin	kookmin_ir0.0001	19.79	77.9	9.74	79.0	
Toys	Kookmin	kookmin_ir0.001	11.69	77.1	5.00	77.8	✓
Toys	Kookmin	kookmin_ir0.01	15.02	70.0	1.67	71.2	
Toys	Fanchuan	fanchuan_t0.007	0.00	4.0	0.00	5.2	✓
Toys	Fanchuan	fanchuan_t0.07	0.00	7.4	0.00	8.9	
Toys	Fanchuan	fanchuan_t1.15	0.00	17.3	1.54	18.3	
Toys	TRACER	tracer_ep4	18.12	103.2	11.15	104.2	
Toys	TRACER	tracer_lf0.1_lc0.1_ep4	21.46	102.4	11.15	103.4	
Toys	TRACER	tracer_lf0.1_lc1.0_ep4	18.12	103.3	11.15	104.4	
Toys	TRACER	tracer_lf1.0_lc0.1_ep4	21.46	102.0	11.15	103.3	
Toys	TRACER	tracer_lf1.0_lc1.0_ep4	18.12	103.2	11.15	104.2	✓
Toys	NAU	$\lambda_n = -0.1$	9.87	91.6	8.21	92.7	
Toys	NAU	$\lambda_n = -1$	11.54	94.1	9.74	95.0	✓
Toys	NAU	$\lambda_n = -10$	8.33	94.5	8.21	95.6	
Toys	NAU	$\lambda_n = 0$	11.69	91.5	5.00	92.5	

Table 9: Efficiency of unlearning methods on Beauty (TIGER + RQ-KMeans), each at its configuration in Table 1. $\mathcal{E}$ is the median wall-clock time per unlearning request on one H200 GPU over three targets $\times$ five data seeds, and $\tau_E = 1 - \mathcal{E}/\mathcal{E}_{\text{retrain}}$ the fraction of retraining cost saved. † TRACER under its own tokenizer, as in Table 1.

Method	$\mathcal{E}$ (s) $\downarrow$	$\tau_E \uparrow$
Finetune	0.5	0.99997
Forget only	0.3	0.99998
Forget + Repair	0.7	0.99995
SCIF	35.3	0.99774
SEIF	0.5	0.99997
Kookmin	0.4	0.99998
Fanchuan	1.6	0.99990
TRACER†	2.7	0.99983
NAU (ours)	3.9	0.99975
NAU, $\lambda_s = 0.1$	3.5	0.99978
Retrain (reference)	*15,586*	*0*

Table 10: Additional NAU variants on Beauty with TIGER and RQ-KMeans SIDs under the bandwagon attack. The variants differ in the separation score and the neighborhood/separation weight selection strategy. SH@10, NDCG@10, and τ_P are reported as percentages.

Target	Variant	SH@10 (%)	NDCG@10 (%)	τ_P (%)	$\mathcal{E}$ (s)
Unpopular	NAU (gen. sep.)	0.000±0.000	3.600±0.119	95.8±6.2	3.5
	NAU (cos. sep.)	0.000±0.000	3.504±0.092	93.2±5.4	2.9
	NAU	0.000±0.000	3.596±0.115	95.6±6.1	3.4
	NAU (grid)	0.000±0.000	3.612±0.106	96.1±5.8	3.7
Mid	NAU (gen. sep.)	0.000±0.000	3.654±0.041	97.1±2.8	3.5
	NAU (cos. sep.)	0.000±0.000	3.545±0.053	94.2±2.3	2.9
	NAU	0.000±0.000	3.656±0.044	97.2±3.1	1.7
	NAU (grid)	0.004±0.008	3.599±0.041	95.7±3.2	1.4
Popular	NAU (gen. sep.)	0.000±0.000	3.596±0.148	95.5±2.6	3.4
	NAU (cos. sep.)	0.107±0.240	3.515±0.161	93.4±3.1	2.9
	NAU	0.000±0.000	3.603±0.145	95.7±2.6	1.6
	NAU (grid)	0.000±0.000	3.610±0.163	95.9±2.6	3.0

Table 11: Spam-removal results on Beauty with TIGER + RQ-VAE SIDs under the bandwagon attack. Each baseline uses the configuration selected for TIGER + RQ-KMeans on the validation split of the same dataset (Table 7). NAU is reported at its operating point ($\lambda_f = 0.1$, $\lambda_s = 0.01$, $\lambda_n = 0.01$, generative separation score); the indented row below it uses a stronger separation weight ($\lambda_s = 0.1$) and is not ranked. SH@10, $\Delta_{\mathrm{SH}}^{\mathrm{retr}}$, NDCG@10, and τ_P are reported as percentages. $\Delta_{\mathrm{SH}}^{\mathrm{retr}}$ is the mean over (seed, target) cells of $|\mathrm{SH@10} - \mathrm{SH@10}_{\mathrm{retr}}|$, so it measures closeness to retraining rather than raw suppression; it is the same quantity Table 1 reports. SH@10 measures target suppression; closeness to retraining also depends on retrained exposure. Higher NDCG@10 and τ_P indicate better utility preservation. $\mathcal{E}$ reports unlearning time per request in seconds.

Target	Method	SH@10 (%)	$\Delta_{\mathrm{SH}}^{\mathrm{retr}}$ (%) ↓	NDCG@10 (%)	τ_P (%)	$\mathcal{E}$ (s)
Unpopular	*Baselines*					
	Finetune	0.365±0.362	0.361±0.362	**3.206±0.106**	**98.6±2.4**	0.9
	Forget only	0.000±0.000	0.004±0.008	0.166±0.288	5.2±9.1	0.4
	Forget + Repair	0.016±0.036	0.020±0.035	3.196±0.104	98.3±2.6	**0.8**
	SCIF	0.729±1.081	0.725±1.082	3.203±0.097	98.5±1.2	35.2
	SEIF	0.119±0.190	0.115±0.182	2.267±0.192	69.8±6.6	0.5
	Kookmin	0.193±0.119	0.190±0.114	2.811±0.156	86.5±5.2	0.4
	Fanchuan	0.000±0.000	0.004±0.008	1.113±0.291	34.4±9.7	1.6
	TRACER	0.658±1.220	0.655±1.221	3.173±0.101	97.6±1.6	2.3
	Ours					
	NAU	**0.000±0.000**	**0.004±0.008**	3.206±0.084	98.6±2.0	4.0
	NAU, $\lambda_s = 0.1$	0.000±0.000	0.004±0.008	3.205±0.093	98.6±2.1	3.5
Mid	*Baselines*					
	Finetune	1.116±1.561	1.116±1.561	**3.224±0.107**	**99.2±3.4**	**0.5**
	Forget only	0.000±0.000	0.000±0.000	0.045±0.055	1.4±1.7	0.4
	Forget + Repair	0.058±0.080	0.058±0.080	3.213±0.089	98.8±3.0	0.7
	SCIF	2.482±4.410	2.482±4.410	3.209±0.157	98.7±5.3	35.2
	SEIF	0.061±0.050	0.061±0.050	2.358±0.112	72.6±3.6	0.5
	Kookmin	0.391±0.553	0.391±0.553	2.716±0.077	83.5±2.5	0.4
	Fanchuan	0.000±0.000	0.000±0.000	1.078±0.206	33.2±6.8	1.6
	TRACER	0.794±1.121	0.794±1.121	3.077±0.118	94.7±3.3	2.4
	Ours					
	NAU	**0.020±0.029**	**0.020±0.029**	3.211±0.090	98.8±3.0	4.3
	NAU, $\lambda_s = 0.1$	0.000±0.000	0.000±0.000	3.189±0.103	98.1±3.3	3.4
Popular	*Baselines*					
	Finetune	0.441±0.402	0.227±0.226	**3.286±0.071**	**101.1±2.9**	**0.9**
	Forget only	0.000±0.000	0.436±0.479	0.030±0.051	0.9±1.5	0.3
	Forget + Repair	0.002±0.004	0.434±0.478	3.284±0.080	101.0±2.8	1.3
	SCIF	0.494±0.369	**0.208±0.118**	3.241±0.070	99.7±2.3	35.1
	SEIF	0.056±0.045	0.395±0.424	2.277±0.126	70.0±4.0	0.5
	Kookmin	0.849±1.083	0.485±0.555	2.760±0.193	84.9±6.7	0.4
	Fanchuan	0.000±0.000	0.436±0.479	0.938±0.086	28.9±2.8	1.6
	TRACER	0.454±0.249	0.265±0.251	3.184±0.046	97.9±1.4	2.5
	Ours					
	NAU	**0.000±0.000**	0.436±0.479	3.281±0.080	100.9±2.9	4.2
	NAU, $\lambda_s = 0.1$	0.000±0.000	0.436±0.479	3.275±0.075	100.7±3.0	3.5

Table 12: Spam-removal results on Sports with TIGER + RQ-VAE SIDs under the bandwagon attack. Each baseline uses the configuration selected for TIGER + RQ-KMeans on the validation split of the same dataset (Table 7). NAU is reported at its operating point ($\lambda_f = 0.1$, $\lambda_s = 0.01$, $\lambda_n = 0.01$, generative separation score); the indented row below it uses a stronger separation weight ($\lambda_s = 0.1$) and is not ranked. SH@10, $\Delta_{\mathrm{SH}}^{\mathrm{retr}}$, NDCG@10, and τ_P are reported as percentages. $\Delta_{\mathrm{SH}}^{\mathrm{retr}}$ is the mean over (seed, target) cells of $|\mathrm{SH@10} - \mathrm{SH@10}_{\mathrm{retr}}|$, so it measures closeness to retraining rather than raw suppression; it is the same quantity Table 1 reports. SH@10 measures target suppression; closeness to retraining also depends on retrained exposure. Higher NDCG@10 and τ_P indicate better utility preservation. $\mathcal{E}$ reports unlearning time per request in seconds.

Target	Method	SH@10 (%)	$\Delta_{\mathrm{SH}}^{\mathrm{retr}}$ (%) ↓	NDCG@10 (%)	τ_P (%)	$\mathcal{E}$ (s)
Unpopular	*Baselines*					
	Finetune	0.494±0.690	0.494±0.690	**1.719±0.095**	**106.0±8.9**	1.3
	Forget only	0.000±0.000	0.000±0.000	0.085±0.178	6.0±12.7	0.4
	Forget + Repair	0.178±0.395	0.178±0.395	1.710±0.095	105.4±9.0	**1.0**
	SCIF	0.725±1.004	0.725±1.004	1.658±0.091	102.3±10.5	34.9
	SEIF	0.011±0.015	0.011±0.015	1.001±0.073	61.8±6.8	0.7
	Kookmin	0.147±0.211	0.147±0.211	1.336±0.092	82.4±8.4	0.5
	Fanchuan	0.000±0.000	0.000±0.000	0.535±0.088	33.1±6.8	2.1
	TRACER	0.725±1.035	0.725±1.035	1.657±0.091	102.4±11.0	4.5
	Ours					
	NAU	**0.042±0.090**	**0.042±0.090**	1.700±0.098	104.8±8.2	8.1
	NAU, $\lambda_s = 0.1$	0.000±0.000	0.000±0.000	1.711±0.097	105.4±8.5	4.2
Mid	*Baselines*					
	Finetune	1.237±1.330	1.189±1.268	1.690±0.064	104.2±8.6	**1.3**
	Forget only	0.000±0.000	0.048±0.107	0.000±0.001	0.0±0.1	0.5
	Forget + Repair	0.107±0.154	0.059±0.075	**1.696±0.065**	**104.7±9.2**	1.7
	SCIF	1.705±1.942	1.657±1.903	1.664±0.063	102.7±9.6	35.2
	SEIF	0.052±0.056	0.054±0.058	0.983±0.082	60.4±2.8	0.7
	Kookmin	0.411±0.380	0.364±0.371	1.322±0.077	81.4±5.3	0.5
	Fanchuan	0.000±0.000	0.048±0.107	0.545±0.073	33.4±2.3	2.1
	TRACER	1.549±1.736	1.501±1.673	1.660±0.064	102.5±9.9	4.7
	Ours					
	NAU	**0.077±0.143**	**0.029±0.041**	1.680±0.062	103.7±9.4	5.1
	NAU, $\lambda_s = 0.1$	0.000±0.000	0.048±0.107	1.680±0.064	103.7±9.1	4.2
Popular	*Baselines*					
	Finetune	0.406±0.417	0.393±0.413	1.679±0.058	103.7±9.9	**1.4**
	Forget only	0.000±0.000	0.035±0.055	0.001±0.001	0.0±0.1	0.4
	Forget + Repair	**0.000±0.000**	**0.035±0.055**	**1.686±0.041**	**104.2±10.5**	1.7
	SCIF	0.747±0.715	0.712±0.738	1.618±0.050	100.0±10.5	35.4
	SEIF	0.011±0.009	0.030±0.049	1.006±0.092	62.2±8.5	0.7
	Kookmin	0.289±0.482	0.303±0.470	1.403±0.102	86.6±9.3	0.5
	Fanchuan	0.000±0.000	0.035±0.055	0.555±0.049	34.4±4.8	2.1
	TRACER	0.475±0.420	0.469±0.405	1.621±0.048	100.2±10.6	4.5
	Ours					
	NAU	0.000±0.000	0.035±0.055	1.663±0.057	102.8±11.2	5.2
	NAU, $\lambda_s = 0.1$	0.000±0.000	0.035±0.055	1.672±0.044	103.3±10.7	4.2

Table 13: Spam-removal results on Toys with TIGER + RQ-VAE SIDs under the bandwagon attack. Each baseline uses the configuration selected for TIGER + RQ-KMeans on the validation split of the same dataset (Table 7). NAU is reported at its operating point ($\lambda_f = 0.1$, $\lambda_s = 0.01$, $\lambda_n = 0.01$, generative separation score); the indented row below it uses a stronger separation weight ($\lambda_s = 0.1$) and is not ranked. SH@10, $\Delta_{\mathrm{SH}}^{\mathrm{retr}}$, NDCG@10, and τ_P are reported as percentages. $\Delta_{\mathrm{SH}}^{\mathrm{retr}}$ is the mean over (seed, target) cells of $|\mathrm{SH@10} - \mathrm{SH@10}_{\mathrm{retr}}|$, so it measures closeness to retraining rather than raw suppression; it is the same quantity Table 1 reports. SH@10 measures target suppression; closeness to retraining also depends on retrained exposure. Higher NDCG@10 and τ_P indicate better utility preservation. $\mathcal{E}$ reports unlearning time per request in seconds.

Target	Method	SH@10 (%)	$\Delta_{\mathrm{SH}}^{\mathrm{retr}}$ (%) ↓	NDCG@10 (%)	τ_P (%)	$\mathcal{E}$ (s)
Unpopular	*Baselines*					
	Finetune	0.457±0.295	0.453±0.294	3.009±0.048	98.1±3.6	**0.9**
	Forget only	0.000±0.000	0.004±0.009	0.048±0.062	1.5±1.9	0.4
	Forget + Repair	0.031±0.043	0.033±0.042	3.005±0.063	97.9±3.7	1.3
	SCIF	0.651±0.577	0.647±0.579	**3.034±0.058**	**98.8±2.9**	34.2
	SEIF	0.104±0.091	0.100±0.088	1.984±0.145	64.7±5.9	0.5
	Kookmin	0.282±0.252	0.278±0.246	2.364±0.057	77.0±2.6	0.4
	Fanchuan	0.000±0.000	0.004±0.009	0.710±0.136	23.0±3.8	1.6
	TRACER	0.500±0.423	0.496±0.422	2.975±0.088	96.9±2.8	4.0
	Ours					
	NAU	**0.011±0.025**	**0.015±0.025**	3.001±0.071	97.8±3.7	4.1
	NAU, $\lambda_s = 0.1$	0.001±0.002	0.005±0.009	2.999±0.066	97.7±3.5	3.4
Mid	*Baselines*					
	Finetune	0.511±0.522	0.475±0.545	2.951±0.156	96.0±2.4	**0.9**
	Forget only	0.000±0.000	0.036±0.066	0.020±0.033	0.6±1.1	0.4
	Forget + Repair	0.052±0.085	0.077±0.089	2.957±0.134	96.2±1.7	1.3
	SCIF	1.289±1.985	1.253±2.004	2.948±0.102	96.0±1.7	34.4
	SEIF	0.040±0.014	0.043±0.031	1.961±0.127	63.9±5.3	0.5
	Kookmin	0.175±0.161	0.139±0.174	2.348±0.112	76.4±2.1	0.4
	Fanchuan	0.000±0.000	0.036±0.066	0.791±0.221	25.8±7.4	1.6
	TRACER	0.656±0.821	0.620±0.843	2.933±0.093	95.5±1.8	3.9
	Ours					
	NAU	**0.001±0.002**	**0.035±0.067**	**2.959±0.135**	**96.3±2.0**	4.1
	NAU, $\lambda_s = 0.1$	0.000±0.000	0.036±0.066	2.955±0.147	96.2±2.5	3.4
Popular	*Baselines*					
	Finetune	1.005±0.589	0.787±0.636	**3.005±0.081**	**97.9±4.9**	**0.9**
	Forget only	0.000±0.000	0.217±0.120	0.004±0.006	0.1±0.2	0.4
	Forget + Repair	0.097±0.046	**0.129±0.110**	2.992±0.078	97.5±5.1	1.3
	SCIF	9.370±16.800	9.153±16.800	2.981±0.072	97.1±4.0	34.3
	SEIF	0.108±0.104	0.140±0.115	2.014±0.154	65.5±3.7	0.5
	Kookmin	0.561±0.387	0.353±0.403	2.329±0.072	75.8±2.4	0.4
	Fanchuan	0.000±0.000	0.217±0.120	0.685±0.120	22.3±3.8	1.6
	TRACER	1.327±0.744	1.110±0.699	2.921±0.077	95.2±5.2	4.4
	Ours					
	NAU	**0.019±0.025**	0.199±0.116	2.991±0.087	97.5±5.3	3.9
	NAU, $\lambda_s = 0.1$	0.000±0.000	0.217±0.120	2.979±0.074	97.1±5.0	3.4

Table 14: Spam-removal results on Beauty with TIGER + LETTER SIDs under the bandwagon attack. Each baseline uses the configuration selected for TIGER + RQ-KMeans on the validation split of the same dataset (Table 7). NAU is reported at its operating point ($\lambda_f = 0.1$, $\lambda_s = 0.01$, $\lambda_n = 0.01$, generative separation score); the indented row below it uses a stronger separation weight ($\lambda_s = 0.1$) and is not ranked. SH@10, $\Delta_{\mathrm{SH}}^{\mathrm{retr}}$, NDCG@10, and τ_P are reported as percentages. $\Delta_{\mathrm{SH}}^{\mathrm{retr}}$ is the mean over (seed, target) cells of $|\mathrm{SH@10} - \mathrm{SH@10}_{\mathrm{retr}}|$, so it measures closeness to retraining rather than raw suppression; it is the same quantity Table 1 reports. SH@10 measures target suppression; closeness to retraining also depends on retrained exposure. Higher NDCG@10 and τ_P indicate better utility preservation. $\mathcal{E}$ reports unlearning time per request in seconds.

Target	Method	SH@10 (%)	$\Delta_{\mathrm{SH}}^{\mathrm{retr}}$ (%) ↓	NDCG@10 (%)	τ_P (%)	$\mathcal{E}$ (s)
Unpopular	*Baselines*					
	Finetune	0.194±0.142	0.193±0.142	**3.823±0.062**	**101.0±5.3**	**0.9**
	Forget only	0.000±0.000	0.001±0.002	0.011±0.015	0.3±0.4	0.4
	Forget + Repair	**0.000±0.000**	**0.001±0.002**	3.778±0.082	99.8±5.8	1.3
	SCIF	0.245±0.183	0.244±0.182	3.764±0.062	99.4±3.8	40.0
	SEIF	0.123±0.157	0.123±0.156	2.684±0.240	70.7±3.3	0.6
	Kookmin	0.225±0.124	0.224±0.124	3.128±0.104	82.6±3.7	0.6
	Fanchuan	0.000±0.000	0.001±0.002	1.244±0.189	32.7±3.7	1.6
	TRACER	0.140±0.166	0.140±0.166	3.767±0.063	99.4±3.7	4.1
	Ours					
	NAU	0.000±0.000	0.001±0.002	3.797±0.090	100.3±5.8	3.5
	NAU, $\lambda_s = 0.1$	0.000±0.000	0.001±0.002	3.791±0.089	100.2±5.8	3.5
Mid	*Baselines*					
	Finetune	0.276±0.411	0.276±0.411	**3.799±0.052**	**100.3±4.6**	**0.5**
	Forget only	0.000±0.000	0.000±0.000	0.015±0.023	0.4±0.6	0.4
	Forget + Repair	0.001±0.002	0.001±0.002	3.785±0.083	100.0±4.6	1.3
	SCIF	0.523±0.731	0.523±0.731	3.781±0.083	99.9±5.0	56.7
	SEIF	0.013±0.012	0.013±0.012	2.746±0.160	72.4±3.5	0.6
	Kookmin	0.127±0.144	0.127±0.144	3.232±0.155	85.4±6.1	0.6
	Fanchuan	0.000±0.000	0.000±0.000	1.359±0.227	36.0±6.6	1.8
	TRACER	0.275±0.410	0.275±0.410	3.778±0.090	99.8±4.9	3.9
	Ours					
	NAU	**0.000±0.000**	**0.000±0.000**	3.766±0.067	99.5±4.6	3.4
	NAU, $\lambda_s = 0.1$	0.000±0.000	0.000±0.000	3.785±0.050	100.0±4.3	3.5
Popular	*Baselines*					
	Finetune	0.268±0.205	0.198±0.226	**3.789±0.099**	**100.1±5.0**	**0.6**
	Forget only	0.000±0.000	0.247±0.238	0.014±0.031	0.4±0.8	0.3
	Forget + Repair	0.001±0.002	0.246±0.238	3.744±0.063	98.9±4.1	0.7
	SCIF	0.296±0.239	0.248±0.249	3.721±0.131	98.2±4.8	69.2
	SEIF	0.080±0.102	0.210±0.234	2.627±0.066	69.4±4.3	1.0
	Kookmin	0.354±0.424	0.326±0.438	3.136±0.148	82.9±7.2	0.7
	Fanchuan	0.000±0.000	0.247±0.238	1.135±0.071	29.9±1.4	1.9
	TRACER	0.141±0.122	**0.195±0.215**	3.722±0.127	98.3±4.8	2.7
	Ours					
	NAU	**0.000±0.000**	0.247±0.238	3.747±0.062	98.9±3.9	3.9
	NAU, $\lambda_s = 0.1$	0.000±0.000	0.247±0.238	3.751±0.063	99.1±4.0	3.5

Table 15: Spam-removal results on Sports with TIGER + LETTER SIDs under the bandwagon attack. Each baseline uses the configuration selected for TIGER + RQ-KMeans on the validation split of the same dataset (Table 7). NAU is reported at its operating point ($\lambda_f = 0.1$, $\lambda_s = 0.01$, $\lambda_n = 0.01$, generative separation score); the indented row below it uses a stronger separation weight ($\lambda_s = 0.1$) and is not ranked. SH@10, $\Delta_{\mathrm{SH}}^{\mathrm{retr}}$, NDCG@10, and τ_P are reported as percentages. $\Delta_{\mathrm{SH}}^{\mathrm{retr}}$ is the mean over (seed, target) cells of $|\mathrm{SH@10} - \mathrm{SH@10}_{\mathrm{retr}}|$, so it measures closeness to retraining rather than raw suppression; it is the same quantity Table 1 reports. SH@10 measures target suppression; closeness to retraining also depends on retrained exposure. Higher NDCG@10 and τ_P indicate better utility preservation. $\mathcal{E}$ reports unlearning time per request in seconds.

Target	Method	SH@10 (%)	$\Delta_{\mathrm{SH}}^{\mathrm{retr}}$ (%) ↓	NDCG@10 (%)	τ_P (%)	$\mathcal{E}$ (s)
Unpopular	*Baselines*					
	Finetune	0.240±0.231	0.167±0.204	**1.992±0.053**	**105.8±7.0**	1.3
	Forget only	0.000±0.000	0.078±0.105	0.011±0.021	0.6±1.1	0.3
	Forget + Repair	0.030±0.066	0.097±0.100	1.976±0.055	104.9±7.2	**1.0**
	SCIF	0.334±0.352	0.256±0.345	1.946±0.060	103.3±6.4	34.8
	SEIF	0.157±0.210	0.079±0.106	1.206±0.087	64.0±6.3	0.8
	Kookmin	0.276±0.325	0.198±0.231	1.555±0.148	82.7±10.7	0.5
	Fanchuan	0.000±0.000	0.078±0.105	0.653±0.038	34.7±3.1	2.1
	TRACER	0.221±0.240	0.166±0.227	1.954±0.061	103.7±6.6	4.9
	Ours					
	NAU	**0.013±0.030**	**0.081±0.103**	1.979±0.049	105.1±7.0	4.2
	NAU, $\lambda_s = 0.1$	0.000±0.000	0.078±0.105	1.982±0.038	105.2±6.6	4.2
Mid	*Baselines*					
	Finetune	0.625±1.247	0.625±1.248	**1.972±0.026**	**104.6±4.0**	1.4
	Forget only	0.000±0.000	0.001±0.001	0.000±0.000	0.0±0.0	0.3
	Forget + Repair	0.007±0.016	0.008±0.016	1.966±0.026	104.3±4.2	**1.0**
	SCIF	1.233±2.583	1.233±2.583	1.934±0.044	102.6±3.8	39.5
	SEIF	0.020±0.027	0.020±0.027	1.141±0.074	60.5±3.4	0.8
	Kookmin	0.240±0.269	0.239±0.270	1.496±0.069	79.4±5.2	0.5
	Fanchuan	0.000±0.000	0.001±0.001	0.617±0.021	32.8±2.3	2.3
	TRACER	0.951±2.067	0.951±2.068	1.932±0.046	102.4±3.7	4.7
	Ours					
	NAU	**0.000±0.000**	**0.001±0.001**	1.943±0.025	103.0±3.8	4.2
	NAU, $\lambda_s = 0.1$	0.000±0.000	0.001±0.001	1.943±0.031	103.0±3.8	4.3
Popular	*Baselines*					
	Finetune	0.545±0.903	0.524±0.915	**1.915±0.047**	**101.6±2.8**	**0.8**
	Forget only	0.000±0.000	0.021±0.032	0.003±0.004	0.2±0.2	0.5
	Forget + Repair	0.002±0.005	0.023±0.031	1.908±0.056	101.1±2.1	0.9
	SCIF	2.331±4.948	2.310±4.960	1.886±0.061	100.0±2.7	68.6
	SEIF	0.013±0.011	0.022±0.027	1.234±0.124	65.3±3.8	0.8
	Kookmin	0.133±0.094	0.137±0.067	1.571±0.145	83.1±4.9	0.5
	Fanchuan	0.000±0.000	0.021±0.032	0.692±0.097	36.6±4.6	2.1
	TRACER	1.615±3.515	1.595±3.527	1.884±0.054	99.9±2.9	4.7
	Ours					
	NAU	**0.000±0.000**	**0.021±0.032**	1.910±0.058	101.2±2.2	4.2
	NAU, $\lambda_s = 0.1$	0.000±0.000	0.021±0.032	1.904±0.066	100.9±2.2	4.2

Table 16: Spam-removal results on Toys with TIGER + LETTER SIDs under the bandwagon attack. Each baseline uses the configuration selected for TIGER + RQ-KMeans on the validation split of the same dataset (Table 7). NAU is reported at its operating point ($\lambda_f = 0.1$, $\lambda_s = 0.01$, $\lambda_n = 0.01$, generative separation score); the indented row below it uses a stronger separation weight ($\lambda_s = 0.1$) and is not ranked. SH@10, $\Delta_{\mathrm{SH}}^{\mathrm{retr}}$, NDCG@10, and τ_P are reported as percentages. $\Delta_{\mathrm{SH}}^{\mathrm{retr}}$ is the mean over (seed, target) cells of $|\mathrm{SH@10} - \mathrm{SH@10}_{\mathrm{retr}}|$, so it measures closeness to retraining rather than raw suppression; it is the same quantity Table 1 reports. SH@10 measures target suppression; closeness to retraining also depends on retrained exposure. Higher NDCG@10 and τ_P indicate better utility preservation. $\mathcal{E}$ reports unlearning time per request in seconds.

Target	Method	SH@10 (%)	$\Delta_{\mathrm{SH}}^{\mathrm{retr}}$ (%) ↓	NDCG@10 (%)	τ_P (%)	$\mathcal{E}$ (s)
Unpopular	*Baselines*					
	Finetune	0.518±0.398	0.501±0.415	3.374±0.059	94.6±3.3	0.5
	Forget only	0.000±0.000	0.018±0.034	0.000±0.000	0.0±0.0	0.3
	Forget + Repair	**0.012±0.013**	**0.026±0.031**	3.391±0.048	95.0±2.6	**1.2**
	SCIF	0.874±0.741	0.856±0.759	**3.411±0.054**	**95.6±2.8**	39.1
	SEIF	0.098±0.087	0.109±0.072	2.124±0.117	59.6±4.4	0.6
	Kookmin	0.440±0.294	0.422±0.280	2.469±0.172	69.1±4.2	0.4
	Fanchuan	0.000±0.000	0.018±0.034	0.923±0.157	25.9±4.6	1.8
	TRACER	0.554±0.483	0.537±0.501	3.403±0.059	95.3±2.9	2.4
	Ours					
	NAU	0.002±0.003	0.018±0.034	3.384±0.037	94.8±2.5	3.4
	NAU, $\lambda_s = 0.1$	0.000±0.000	0.018±0.034	3.383±0.038	94.8±2.5	3.4
Mid	*Baselines*					
	Finetune	0.378±0.201	0.264±0.206	3.525±0.129	98.8±4.6	**0.6**
	Forget only	0.000±0.000	0.121±0.200	0.000±0.000	0.0±0.0	0.3
	Forget + Repair	0.013±0.019	0.130±0.191	3.459±0.193	96.9±5.7	1.3
	SCIF	0.778±0.657	0.657±0.754	3.526±0.098	98.8±3.8	66.8
	SEIF	0.062±0.110	0.077±0.089	2.334±0.374	65.5±11.4	0.5
	Kookmin	0.226±0.232	0.150±0.085	2.681±0.247	75.2±7.9	0.4
	Fanchuan	0.000±0.000	0.121±0.200	1.082±0.341	30.4±9.9	1.8
	TRACER	0.510±0.408	0.424±0.469	**3.536±0.087**	**99.1±3.6**	4.1
	Ours					
	NAU	**0.004±0.009**	**0.123±0.199**	3.510±0.160	98.4±5.1	3.4
	NAU, $\lambda_s = 0.1$	0.000±0.000	0.121±0.200	3.504±0.159	98.2±5.1	3.4
Popular	*Baselines*					
	Finetune	0.522±0.393	0.385±0.335	3.585±0.066	100.4±3.0	**0.5**
	Forget only	0.000±0.000	0.176±0.062	0.000±0.000	0.0±0.0	0.4
	Forget + Repair	0.071±0.105	**0.132±0.073**	**3.587±0.065**	**100.5±3.3**	0.8
	SCIF	0.919±0.698	0.768±0.660	3.563±0.053	99.8±3.3	66.8
	SEIF	0.123±0.167	0.103±0.064	2.455±0.279	68.8±7.7	0.7
	Kookmin	0.352±0.339	0.252±0.273	2.862±0.168	80.2±5.6	0.4
	Fanchuan	0.000±0.000	0.176±0.062	1.088±0.114	30.5±3.1	1.6
	TRACER	0.652±0.566	0.530±0.500	3.557±0.056	99.7±3.3	4.2
	Ours					
	NAU	**0.026±0.036**	0.150±0.071	3.570±0.054	100.0±2.7	3.5
	NAU, $\lambda_s = 0.1$	0.004±0.009	0.172±0.062	3.562±0.061	99.8±3.0	3.5

Table 17: Spam-removal results on Beauty with TIGER + TRACER SIDs under the bandwagon attack. Each baseline uses the configuration selected for TIGER + RQ-KMeans on the validation split of the same dataset (Table 7). NAU is reported at its operating point ($\lambda_f = 0.1$, $\lambda_s = 0.01$, $\lambda_n = 0.01$, generative separation score); the indented row below it uses a stronger separation weight ($\lambda_s = 0.1$) and is not ranked. SH@10, $\Delta_{\mathrm{SH}}^{\mathrm{retr}}$, NDCG@10, and τ_P are reported as percentages. $\Delta_{\mathrm{SH}}^{\mathrm{retr}}$ is the mean over (seed, target) cells of $|\mathrm{SH@10} - \mathrm{SH@10}_{\mathrm{retr}}|$, so it measures closeness to retraining rather than raw suppression; it is the same quantity Table 1 reports. SH@10 measures target suppression; closeness to retraining also depends on retrained exposure. Higher NDCG@10 and τ_P indicate better utility preservation. $\mathcal{E}$ reports unlearning time per request in seconds.

Target	Method	SH@10 (%)	$\Delta_{\mathrm{SH}}^{\mathrm{retr}}$ (%) ↓	NDCG@10 (%)	τ_P (%)	$\mathcal{E}$ (s)
Unpopular	*Baselines*					
	Finetune	1.250±0.936	1.250±0.936	3.700±0.072	102.8±3.3	**0.5**
	Forget only	0.000±0.000	0.000±0.000	0.020±0.034	0.5±0.9	0.4
	Forget + Repair	0.097±0.129	0.097±0.129	**3.701±0.077**	102.8±3.2	0.7
	SCIF	1.729±1.660	1.729±1.660	3.649±0.056	101.4±3.0	39.9
	SEIF	0.535±0.714	0.535±0.714	2.607±0.193	72.5±5.6	0.5
	Kookmin	0.917±0.406	0.917±0.406	3.115±0.181	86.6±6.0	0.7
	Fanchuan	0.000±0.000	0.000±0.000	1.243±0.255	34.6±7.7	1.8
	TRACER	1.403±1.303	1.403±1.303	3.650±0.046	101.5±3.0	2.7
	Ours					
	NAU	**0.004±0.006**	**0.004±0.006**	3.701±0.070	**102.9±3.4**	4.1
	NAU, $\lambda_s = 0.1$	0.000±0.000	0.000±0.000	3.687±0.063	102.5±3.6	3.4
Mid	*Baselines*					
	Finetune	0.636±0.597	0.635±0.596	**3.572±0.107**	**99.3±2.7**	**0.6**
	Forget only	0.000±0.000	0.001±0.002	0.020±0.041	0.6±1.2	0.4
	Forget + Repair	0.046±0.090	0.045±0.088	3.563±0.116	99.0±2.8	0.7
	SCIF	0.960±0.877	0.959±0.876	3.520±0.157	97.8±3.3	40.0
	SEIF	0.111±0.126	0.110±0.125	2.516±0.156	70.0±5.4	0.9
	Kookmin	0.706±1.076	0.705±1.076	3.104±0.147	86.2±4.0	0.6
	Fanchuan	0.000±0.000	0.001±0.002	1.249±0.237	34.8±7.5	1.8
	TRACER	0.683±0.657	0.682±0.655	3.505±0.158	97.4±3.3	4.4
	Ours					
	NAU	**0.021±0.048**	**0.021±0.046**	3.558±0.112	98.9±2.8	4.1
	NAU, $\lambda_s = 0.1$	0.000±0.000	0.001±0.002	3.554±0.115	98.8±3.0	3.5
Popular	*Baselines*					
	Finetune	0.855±0.705	1.039±1.399	**3.633±0.087**	**101.0±4.2**	**0.5**
	Forget only	0.000±0.000	1.269±2.020	0.021±0.035	0.6±1.0	0.4
	Forget + Repair	0.074±0.094	1.226±2.044	3.616±0.081	100.5±4.1	0.8
	SCIF	0.887±0.726	**1.020±1.311**	3.550±0.072	98.6±1.8	69.0
	SEIF	0.470±0.603	1.333±1.890	2.547±0.142	70.8±4.0	0.6
	Kookmin	1.261±1.460	0.905±0.934	3.047±0.085	84.7±3.5	0.6
	Fanchuan	0.000±0.000	1.269±2.020	1.123±0.115	31.2±3.0	1.6
	TRACER	0.807±0.653	1.103±1.461	3.544±0.074	98.5±2.1	2.7
	Ours					
	NAU	**0.000±0.000**	1.269±2.020	3.622±0.086	100.7±4.4	4.0
	NAU, $\lambda_s = 0.1$	0.000±0.000	1.269±2.020	3.628±0.088	100.9±4.3	3.4

Table 18: Ablation of NAU on Toys with TIGER + RQ-KMeans (mid-popularity target, five seeds). The SID neighborhood contains 563 test cases whose next item shares the target's first SID token; the target's own test cases are excluded. Indented rows are alternative additions to $\mathcal{L}_{\text{ret}} + \mathcal{L}_{\text{for}}$ (siblings, not a ladder). $\Delta_{\text{SH}}^{\text{retr}}$ is the absolute distance to the retrained model's exposure (0.067% on the whole test set and 2.664% in the neighborhood). τ_P denotes NDCG@10 as a percentage of the retrained model's utility within each query set.

Whole test set

Objective	SH@10 (%) ↓	$\Delta_{\text{SH}}^{\text{retr}}$ (%) ↓	NDCG@10 ↑	τ_P (%) ↑
Retrained	*0.067* ±*0.12*	*0.000*	*3.461* ±*0.11*	*100.0*
Poisoned	*0.297* ±*0.23*	*0.230*	*3.377* ±*0.06*	*97.6*
$\mathcal{L}_{\text{ret}}$ only	0.656 ±0.57	0.589	3.348 ±0.03	96.7
+ $\mathcal{L}_{\text{for}}$	0.159 ±0.23	0.092	3.345 ±0.03	96.6
+ $\mathcal{L}_{\text{sep}}$ ($\lambda_s = .01$)	0.113 ±0.18	0.046	3.352 ±0.03	96.8
+ $\mathcal{L}_{\text{sep}}$ ($\lambda_s = .1$)	0.018 ±0.04	0.049	3.340 ±0.03	96.5
+ $\mathcal{L}_{\text{nbr}}$ ($\lambda_n = .01$)	0.007 ±0.01	0.060	3.345 ±0.04	96.6
+ $\mathcal{L}_{\text{nbr}}$ ($\lambda_n = .1$)	0.000 ±0.00	0.067	3.231 ±0.05	93.3
NAU ($\lambda_s = .01, \lambda_n = .01$, ours)	0.001 ±0.00	0.066	3.361 ±0.04	97.1
NAU ($\lambda_s = .1, \lambda_n = .01$)	0.000 ±0.00	0.067	3.358 ±0.05	97.0
NAU ($\lambda_s = .1, \lambda_n = .1$)	0.000 ±0.00	0.067	3.236 ±0.04	93.5

SID neighborhood

Objective	SH@10 (%) ↓	$\Delta_{\text{SH}}^{\text{retr}}$ (%) ↓	NDCG@10 ↑	τ_P (%) ↑
Retrained	*2.664* ±*3.71*	*0.000*	*5.206* ±*4.75*	*100.0*
Poisoned	*3.730* ±*4.47*	*1.066*	*4.224* ±*4.34*	*81.1*
$\mathcal{L}_{\text{ret}}$ only	8.526 ±15.72	5.861	4.907 ±4.30	94.3
+ $\mathcal{L}_{\text{for}}$	4.440 ±9.53	1.776	4.416 ±4.48	84.8
+ $\mathcal{L}_{\text{sep}}$ ($\lambda_s = .01$)	2.664 ±2.74	0.000	4.270 ±4.32	82.0
+ $\mathcal{L}_{\text{sep}}$ ($\lambda_s = .1$)	0.710 ±0.81	1.954	3.893 ±4.17	74.8
+ $\mathcal{L}_{\text{nbr}}$ ($\lambda_n = .01$)	0.000 ±0.00	2.664	4.788 ±4.14	92.0
+ $\mathcal{L}_{\text{nbr}}$ ($\lambda_n = .1$)	0.000 ±0.00	2.664	6.182 ±3.33	118.8
NAU ($\lambda_s = .01, \lambda_n = .01$, ours)	0.000 ±0.00	2.664	4.879 ±4.32	93.7
NAU ($\lambda_s = .1, \lambda_n = .01$)	0.000 ±0.00	2.664	4.912 ±4.19	94.4
NAU ($\lambda_s = .1, \lambda_n = .1$)	0.000 ±0.00	2.664	5.923 ±3.19	113.8

Table 19: Ablation of NAU on Beauty (TIGER + RQ-KMeans, five seeds × three targets). Indented rows each add one term to $\mathcal{L}_{\text{ret}} + \mathcal{L}_{\text{for}}$ (siblings, not a ladder). Local effect: signed change in log probability mass relative to the retrained model of the same seed, on held-out test histories, for the forgotten target (Δ_{tgt}) and its SID-prefix neighbors beyond prefix-disjoint controls ($\Delta_{\text{nbr}} - \Delta_{\text{ctrl}}$); mean ± s.d. across seeds. † Local effect measured on a matched configuration rather than the ablation's own runs. $\mathcal{L}_{\text{for}}$ does most of the removal, $\mathcal{L}_{\text{nbr}}$ moves mass into the neighborhood in proportion to λ_n, and a strong separation term ($\lambda_s = 0.1$) pushes the target far below its retrained mass.

Objective	**Removal and utility**				**Local effect** (nats)	
	SH@10 (%)	$\Delta_{\text{SH}}^{\text{retr}}$ (%) ↓	NDCG@10 ↑	τ_P (%) ↑	Δ_{tgt}	$\Delta_{\text{nbr}} - \Delta_{\text{ctrl}}$
$\mathcal{L}_{\text{ret}}$ only	0.986 ±0.947	0.914	3.630 ±0.109	96.5 ±3.7	+5.07 ±1.69	−0.14 ±0.21
+ $\mathcal{L}_{\text{for}}$	0.110 ±0.211	0.108	3.622 ±0.118	96.3 ±4.0	+1.67 ±4.17	+0.04 ±0.31
+ $\mathcal{L}_{\text{sep}}$ ($\lambda_s = 0.1$)	0.000 ±0.000	0.072	3.615 ±0.105	96.1 ±3.9	−19.04 ±5.57	−0.11 ±0.17
+ $\mathcal{L}_{\text{nbr}}$ ($\lambda_n = 0.01$)	0.038 ±0.118	0.103	3.615 ±0.111	96.1 ±3.8	−1.74 ±3.23†	+0.36 ±0.37†
NAU ($\lambda_s = 0.01, \lambda_n = 0.01$)	0.010 ±0.034	0.081	3.621 ±0.107	96.2 ±3.8	−4.32 ±3.39	+0.33 ±0.34
NAU at $\lambda_s = 0.1$	0.000 ±0.000	0.072	3.617 ±0.108	96.1 ±3.9	−18.53 ±4.08	+0.18 ±0.30
NAU at $\lambda_s = 0.1$, $\lambda_n = 0.1$ (over-strong dose)	0.000 ±0.000	0.072	3.519 ±0.107	93.5 ±3.5	−18.44 ±9.07	+1.55 ±0.63

Table 20: Per-term ablation of the NAU objective on Sports with TIGER + RQ-KMeans (bandwagon, 1% poison, $n_{\text{target}} = 1$), at $\lambda_f = 0.1$, $\lambda_s = 0.1$, $\lambda_n = 0.01$ and $\boldsymbol{w}_r = \boldsymbol{w}_f = \mathbf{1}$, with the generative separation score ϕ_{gen} and embedding/mass neighbours in every row that activates the corresponding term, so only the WEIGHTS vary. The one exception is the over-strong dose row, which is only available with the cosine score and is marked accordingly. Each data seed is scored against its own clean and poisoned reference, with unlearning replicates averaged within a seed and the spread taken across seeds; ± is therefore absent wherever the column rests on a single data seed. SH@10, NDCG@10, τ_P and $\Delta\tau_P$ are reported as percentages, matching the main results tables. $\Delta\tau_P$ is paired against the $+\mathcal{L}_{\text{forget}}$ row on (stratum, seed), which removes the seed intercept.

Target	Loss terms	SH@10 (%)	NDCG@10 (%)	τ_P (%)	$\Delta\tau_P$ (%)
Unpopular	$\mathcal{L}_{\text{retain}}$ only	0.184±0.231	2.082±0.024	102.5±3.8	+0.85±0.50
	+ $\mathcal{L}_{\text{forget}}$	0.000±0.000	2.065±0.029	101.7±4.0	–
	+ $\mathcal{L}_{\text{sep}}$ ($\lambda_s = 0.1$)	0.000±0.000	2.059±0.026	101.4±3.5	-0.29±0.61
	+ $\mathcal{L}_{\text{neighborhood}}$ ($\lambda_n = 0.01$)	0.000±0.000	2.069±0.041	101.9±4.0	+0.18±1.14
	+ $\mathcal{L}_{\text{sep}}$ + $\mathcal{L}_{\text{neighborhood}}$ = NAU ($\lambda_s = 0.1$, $\lambda_n = 0.01$)	0.000±0.000	2.063±0.031	101.6±3.3	-0.11±0.83
	NAU at $\lambda_n = 0.1$ (over-strong dose, cosine sep)	0.000±0.000	2.007±0.049	98.8±3.7	-2.88±1.75
Mid	$\mathcal{L}_{\text{retain}}$ only	0.353±0.444	2.094±0.079	103.0±2.2	+0.33±1.43
	+ $\mathcal{L}_{\text{forget}}$	0.004±0.007	2.087±0.070	102.7±3.0	–
	+ $\mathcal{L}_{\text{sep}}$ ($\lambda_s = 0.1$)	0.000±0.000	2.086±0.071	102.7±2.7	-0.05±0.52
	+ $\mathcal{L}_{\text{neighborhood}}$ ($\lambda_n = 0.01$)	0.001±0.001	2.081±0.085	102.4±3.4	-0.32±0.86
	+ $\mathcal{L}_{\text{sep}}$ + $\mathcal{L}_{\text{neighborhood}}$ = NAU ($\lambda_s = 0.1$, $\lambda_n = 0.01$)	0.000±0.000	2.084±0.073	102.5±3.4	-0.16±0.61
	NAU at $\lambda_n = 0.1$ (over-strong dose, cosine sep)	0.000±0.000	2.001±0.103	98.5±5.6	-4.22±3.02
Popular	$\mathcal{L}_{\text{retain}}$ only	0.642±0.698	2.070±0.060	102.0±5.6	+0.84±0.75
	+ $\mathcal{L}_{\text{forget}}$	0.017±0.014	2.053±0.049	101.1±5.2	–
	+ $\mathcal{L}_{\text{sep}}$ ($\lambda_s = 0.1$)	0.000±0.000	2.048±0.052	100.9±5.4	-0.24±0.49
	+ $\mathcal{L}_{\text{neighborhood}}$ ($\lambda_n = 0.01$)	0.068±0.118	2.061±0.040	101.5±4.6	+0.37±0.66
	+ $\mathcal{L}_{\text{sep}}$ + $\mathcal{L}_{\text{neighborhood}}$ = NAU ($\lambda_s = 0.1$, $\lambda_n = 0.01$)	0.000±0.000	2.054±0.057	101.2±5.5	+0.06±0.77
	NAU at $\lambda_n = 0.1$ (over-strong dose, cosine sep)	0.000±0.000	1.921±0.115	94.6±7.3	-6.50±3.76

Table 21: Per-term ablation of the NAU objective on Toys with TIGER + RQ-KMeans (bandwagon, 1% poison, $n_{\text{target}} = 1$), at $\lambda_f = 0.1$, $\lambda_s = 0.1$, $\lambda_n = 0.01$ and $\boldsymbol{w}_r = \boldsymbol{w}_f = \mathbf{1}$, with the generative separation score ϕ_{gen} and embedding/mass neighbours in every row that activates the corresponding term, so only the WEIGHTS vary. The one exception is the over-strong dose row, which is only available with the cosine score and is marked accordingly. Each data seed is scored against its own clean and poisoned reference, with unlearning replicates averaged within a seed and the spread taken across seeds; ± is therefore absent wherever the column rests on a single data seed. SH@10, NDCG@10, τ_P and $\Delta\tau_P$ are reported as percentages, matching the main results tables. $\Delta\tau_P$ is paired against the $+\mathcal{L}_{\text{forget}}$ row on (stratum, seed), which removes the seed intercept.

Target	Loss terms	SH@10 (%)	NDCG@10 (%)	τ_P (%)	$\Delta\tau_P$ (%)
Unpopular	$\mathcal{L}_{\text{retain}}$ only	1.092±1.290	3.342±0.058	96.5±3.8	-0.35±0.41
	+ $\mathcal{L}_{\text{forget}}$	0.240±0.255	3.354±0.070	96.9±4.0	–
	+ $\mathcal{L}_{\text{sep}}$ ($\lambda_s = 0.1$)	0.000±0.000	3.337±0.056	96.4±3.2	-0.50±0.99
	+ $\mathcal{L}_{\text{neighborhood}}$ ($\lambda_n = 0.01$)	0.013±0.030	3.349±0.065	96.7±3.5	-0.16±0.45
	+ $\mathcal{L}_{\text{sep}}$ + $\mathcal{L}_{\text{neighborhood}}$ = NAU ($\lambda_s = 0.1$, $\lambda_n = 0.01$)	0.000±0.000	3.347±0.080	96.7±4.6	-0.18±1.02
	NAU at $\lambda_n = 0.1$ (over-strong dose, cosine sep)	0.000±0.000	3.282±0.053	94.8±3.5	-2.10±1.29
Mid	$\mathcal{L}_{\text{retain}}$ only	0.660±0.572	3.344±0.038	96.6±3.6	+0.04±1.28
	+ $\mathcal{L}_{\text{forget}}$	0.172±0.236	3.343±0.019	96.5±2.8	–
	+ $\mathcal{L}_{\text{sep}}$ ($\lambda_s = 0.1$)	0.016±0.037	3.362±0.041	97.1±4.0	+0.58±1.45
	+ $\mathcal{L}_{\text{neighborhood}}$ ($\lambda_n = 0.01$)	0.006±0.011	3.330±0.028	96.2±3.1	-0.36±0.54
	+ $\mathcal{L}_{\text{sep}}$ + $\mathcal{L}_{\text{neighborhood}}$ = NAU ($\lambda_s = 0.1$, $\lambda_n = 0.01$)	0.000±0.000	3.355±0.035	96.9±3.5	+0.36±1.05
	NAU at $\lambda_n = 0.1$ (over-strong dose, cosine sep)	0.000±0.000	3.250±0.036	93.9±3.5	-2.68±1.30
Popular	$\mathcal{L}_{\text{retain}}$ only	0.848±0.956	3.365±0.113	97.2±5.1	-0.12±0.81
	+ $\mathcal{L}_{\text{forget}}$	0.243±0.392	3.369±0.086	97.3±4.6	–
	+ $\mathcal{L}_{\text{sep}}$ ($\lambda_s = 0.1$)	0.007±0.010	3.326±0.078	96.1±4.5	-1.23±0.89
	+ $\mathcal{L}_{\text{neighborhood}}$ ($\lambda_n = 0.01$)	0.159±0.341	3.333±0.102	96.3±5.0	-1.03±0.60
	+ $\mathcal{L}_{\text{sep}}$ + $\mathcal{L}_{\text{neighborhood}}$ = NAU ($\lambda_s = 0.1$, $\lambda_n = 0.01$)	0.005±0.012	3.348±0.098	96.7±4.7	-0.59±0.83
	NAU at $\lambda_n = 0.1$ (over-strong dose, cosine sep)	0.006±0.014	3.256±0.087	94.1±4.5	-3.24±1.01

Table 22: Unwanted item removal on Sports (Guns & Rifles) with TIGER + RQ-KMeans, five data seeds. UHF@10 and UHR@10 are the unwanted item hit rates of the requesting and the remaining users; $\Delta^{\text{retr}}_{\text{UHR}}$ is the mean over seeds of $|\text{UHR}_U@10 - \text{UHR}_R@10|$, the distance to the retrained model of the same seed. Each method uses the configuration selected on the validation split (Table 8; †: $\tau_P < 95\%$, never ranked); NAU uses $\lambda_n = -1$, $\lambda_s = 0$.

Method	UHF@10 (%) ↓	UHR@10 (%)	$\Delta^{\text{retr}}_{\text{UHR}}$ (%) ↓	NDCG@10 ↑	τ_P (%)	$\mathcal{E}$ (s) ↓
Finetune†	29.651 ±14.035	1.602 ±0.252	0.180 ±0.126	1.894 ±0.066	93.0 ±6.4	0.3
Forget only†	16.606 ±12.736	0.694 ±0.327	0.727 ±0.358	1.915 ±0.094	94.1 ±8.6	0.3
Forget+Repair†	30.718 ±10.996	1.475 ±0.338	0.221 ±0.121	1.899 ±0.071	93.2 ±6.7	0.5
SCIF	**32.870 ±16.770**	1.707 ±0.385	0.334 ±0.231	**2.045 ±0.086**	**100.5 ±9.4**	34.5
SEIF†	0.000 ±0.000	0.000 ±0.000	1.422 ±0.197	0.003 ±0.005	0.1 ±0.2	0.3
Kookmin†	31.332 ±14.574	1.408 ±0.295	0.126 ±0.089	1.374 ±0.094	67.4 ±5.4	0.3
Fanchuan†	0.000 ±0.000	0.001 ±0.003	1.420 ±0.198	0.165 ±0.085	8.1 ±4.4	1.1
TRACER	**32.870 ±16.770**	1.662 ±0.396	**0.310 ±0.226**	2.043 ±0.093	100.5 ±9.7	4.2
NAU *(Ours)*†	11.144 ±3.881	0.543 ±0.132	0.878 ±0.241	1.880 ±0.086	92.4 ±8.0	2.6
Original (before deletion)	49.838 ±17.318	2.234 ±0.378	0.812 ±0.362	2.033 ±0.071	99.8 ±6.9	–
Retrain (reference)	29.289 ±9.368	1.422 ±0.197	0.000	2.043 ±0.111	100.0	*20,808*

Table 23: Unwanted item removal on Toys (toy weapons) with TIGER + RQ-KMeans, five data seeds. UHF@10 and UHR@10 are the unwanted item hit rates of the requesting and the remaining users; $\Delta^{\text{retr}}_{\text{UHR}}$ is the mean over seeds of $|\text{UHR}_U@10 - \text{UHR}_R@10|$, the distance to the retrained model of the same seed. Each method uses the configuration selected on the validation split (Table 8; †: $\tau_P < 95\%$, never ranked); NAU uses $\lambda_n = -1$, $\lambda_s = 0$. Retraining does not lower UHF@10 below the original model.

Method	UHF@10 (%) ↓	UHR@10 (%)	$\Delta^{\text{retr}}_{\text{UHR}}$ (%) ↓	NDCG@10 ↑	τ_P (%)	$\mathcal{E}$ (s) ↓
Finetune	**6.538 ±6.940**	2.854 ±0.387	0.767 ±0.463	3.118 ±0.077	95.1 ±2.9	0.3
Forget only†	4.872 ±4.455	1.879 ±0.513	1.634 ±0.404	2.945 ±0.113	89.9 ±5.3	0.3
Forget+Repair†	6.667 ±9.129	2.363 ±0.289	1.149 ±0.374	3.067 ±0.074	93.6 ±3.4	0.5
SCIF	11.154 ±7.045	2.911 ±0.195	**0.602 ±0.340**	3.392 ±0.070	103.5 ±3.7	32.9
SEIF†	0.000 ±0.000	0.021 ±0.046	3.492 ±0.266	0.031 ±0.029	1.0 ±0.9	0.3
Kookmin†	8.205 ±8.338	2.809 ±0.612	0.774 ±0.742	2.500 ±0.161	76.3 ±6.2	0.3
Fanchuan†	0.000 ±0.000	0.000 ±0.000	3.513 ±0.305	0.182 ±0.034	5.5 ±1.0	1.1
TRACER	11.154 ±8.901	2.937 ±0.312	0.663 ±0.281	**3.416 ±0.086**	**104.2 ±3.9**	4.0
NAU *(Ours)*	9.744 ±7.057	2.380 ±0.278	1.133 ±0.253	3.119 ±0.133	95.1 ±5.3	2.4
Original (before deletion)	16.154 ±12.999	2.947 ±0.328	0.613 ±0.488	3.465 ±0.110	105.7 ±4.5	–
Retrain (reference)	18.124 ±9.249	3.513 ±0.305	0.000	3.280 ±0.095	100.0	*20,536*

Table 24: Neighborhood statistics across SID levels. Early levels define broader prefix-sharing neighborhoods, while later levels become increasingly sparse or singleton-dominated. A high singleton share indicates that the corresponding level is less suitable for prefix-based neighborhood construction.

Dataset	Tokenizer	Level	Distinct prefixes	Unique (%)	Mean nbrs.	Median	P90	Singletons (%)
Beauty	RKMeans	0	256	2.1	82.90	74	141	0.2
		1	6,150	50.8	3.16	1	8	30.6
		2	10,483	86.6	0.50	0	2	78.3
		3 (dedup)	12,101	100.0	0.00	0	0	100.0
	RQVAE	0	103	0.9	171.90	151	297	0.0
		1	6,370	52.6	3.94	1	10	32.7
		2	11,498	95.0	0.15	0	0	91.3
		3 (dedup)	12,101	100.0	0.00	0	0	100.0
Toys	RKMeans	0	256	2.1	83.28	83	142	0.2
		1	5,531	46.4	4.57	2	13	27.8
		2	10,025	84.1	0.65	0	2	74.5
		3 (dedup)	11,924	100.0	0.00	0	0	100.0
	RQVAE	0	116	1.0	166.65	141	318	0.0
		1	5,538	46.4	4.90	2	14	28.1
		2	10,918	91.6	0.29	0	1	85.6
		3 (dedup)	11,924	100.0	0.00	0	0	100.0
Sports	RKMeans	0	256	1.4	111.86	102	191	0.0
		1	7,772	42.3	4.94	2	13	22.5
		2	14,964	81.5	0.76	0	2	70.4
		3 (dedup)	18,357	100.0	0.00	0	0	100.0
	RQVAE	0	114	0.6	243.06	213	459	0.0
		1	8,367	45.6	3.99	2	10	25.1
		2	17,109	93.2	0.19	0	1	87.9
		3 (dedup)	18,357	100.0	0.00	0	0	100.0
Food k5	RKMeans	0	256	0.2	1034.07	1007	1684	0.0
		1	19,600	11.7	31.69	18	74	2.8
		2	90,371	54.0	5.06	1	13	37.6
		3 (dedup)	167,428	100.0	0.00	0	0	100.0
	RQVAE	0	148	0.1	1745.87	1537	3253	0.0
		1	21,972	13.1	29.58	15	70	3.2
		2	105,966	63.3	2.96	1	7	47.6
		3 (dedup)	167,428	100.0	0.00	0	0	100.0

Table 25: Restricting the unlearning update to individual semantic ID levels, on beauty, TIGER + RQ-KMeans, at the full objective's operating point. Levels are numbered 1 to $L = 4$, level 4 being the deduplication digit. "codes only" freezes the backbone, so the update reaches the identifier embeddings and nothing else; those are the discriminating arms, because with the backbone free every restriction removes the target and the levels cannot be told apart. Each data seed is scored against its own reference, unlearning replicates averaged within a seed and the spread taken across seeds.

target	update restricted to	SH@10	NDCG@10	$\tau_{\mathcal{P}}$
unpopular: attack landed on all seeds				
	all parameters (unrestricted)	0.00000±0.00000	0.03586±0.00112	0.954±0.059
	levels 3,4 frozen	0.00000	0.03666	0.974
	levels 1,2 + backbone	0.00155±0.00326	0.03605±0.00127	0.959±0.065
	levels 1,2,3 + backbone	0.00140±0.00292	0.03604±0.00130	0.959±0.065
	levels 1,2, codes only	0.00934±0.01058	0.03604±0.00103	0.958±0.055
	levels 3,4, codes only	0.01207	0.03684	0.979
	level 1 only, codes	0.01283±0.01285	0.03591±0.00093	0.955±0.052
	level 2 only, codes	0.01110±0.01211	0.03596±0.00094	0.956±0.053
	level 3 only, codes	0.01426±0.01449	0.03595±0.00102	0.956±0.054
	level 4 only, codes	0.01397	0.03667	0.975
mid: attack landed on all seeds				
	all parameters (unrestricted)	0.00000±0.00000	0.03656±0.00044	0.972±0.031
	levels 3,4 frozen	0.00000	0.03643	0.968
	levels 1,2 + backbone	0.00025±0.00041	0.03645±0.00059	0.969±0.027
	levels 1,2,3 + backbone	0.00026±0.00043	0.03652±0.00066	0.970±0.027
	levels 1,2, codes only	0.00504±0.00641	0.03613±0.00077	0.960±0.017
	levels 3,4, codes only	0.01285	0.03631	0.965
	level 1 only, codes	0.00747±0.00842	0.03606±0.00079	0.958±0.016
	level 2 only, codes	0.00637±0.00799	0.03603±0.00077	0.957±0.017
	level 3 only, codes	0.00936±0.01055	0.03609±0.00084	0.959±0.016
	level 4 only, codes	0.01280	0.03630	0.965
popular: attack landed on 4/5 seeds				
	all parameters (unrestricted)	0.00000±0.00000	0.03603±0.00145	0.957±0.026
	levels 3,4 frozen	0.00000	0.03560	0.946
	levels 1,2 + backbone	0.00166±0.00181	0.03611±0.00137	0.959±0.019
	levels 1,2,3 + backbone	0.00167±0.00190	0.03606±0.00138	0.958±0.022
	levels 1,2, codes only	0.01020±0.01180	0.03549±0.00133	0.943±0.021
	levels 3,4, codes only	0.00136	0.03541	0.941
	level 1 only, codes	0.01319±0.01558	0.03547±0.00136	0.942±0.023
	level 2 only, codes	0.01299±0.01353	0.03540±0.00105	0.940±0.020
	level 3 only, codes	0.01597±0.01711	0.03542±0.00124	0.941±0.021
	level 4 only, codes	0.00192	0.03537	0.940