ITEM REPRESENTATIONS MATTER: BRIDGING HARD SEMANTIC IDS AND SOFT PROTOTYPE MIXTURES FOR GENERATIVE RETRIEVAL

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern recommender systems rely on generative retrieval techniques to predict the next item either as discrete codes or as a continuous vector. On the one hand, Semantic IDs (SIDs) give discrete models shared parameters and pretrained semantic structure, but discretizing items into hard codes discards some information from the original content embeddings. On the other hand, continuous models avoid quantization, yet typically use flat item lookup tables with no explicit parameter tying across items. We ask whether a continuous recommender can introduce shared structure without making it a discrete bottleneck. We propose **CAP-Mix**, a flow recommender that represents each item as *a soft mixture of shared prototypes* plus *an item-specific vector*. The assignments, prototypes, and item vectors are trained jointly, and the soft representation is retained at inference. Pretrained information is optional: SID responsibilities can serve as a prior over assignments, while a full content embedding can initialize the item-specific vector without changing the initial flat table. In matched comparisons on three datasets, CAP-Mix improves over a flat table on the two sparse Amazon datasets and yields a small gain on Steam when trained without pretrained information. Given only an offline SID, it outperforms a hard table built from the same tokenizer by 4.45%–14.90% in HR@5 $\uparrow$. Given the full content embedding, it matches or improves a strong flat table while hard assignments regress on the sparse Amazon datasets. Overall, averaged across 6 benchmarks, CAP-Mix improves HR@5 $\uparrow$ and NDCG@5 $\uparrow$ by $+6.9\%$ and $+13.6\%$ relative to the best evaluated baseline for each dataset and metric, while delivering 1.4–$161\times$ the inference throughput $\uparrow$ of the discrete baselines in our experiments.

1 INTRODUCTION

Generative retrieval techniques enable modern recommender systems to perform large-scale next-item retrieval by learning the next-item distribution and generating recommendations from it, using either discrete tokens (Rajput et al., 2023) or continuous item embeddings (Li et al., 2026b). Either way, a recommender needs an item representation table (or lookup table) as input, i.e., the map from a catalog to the representations it generates or scores against, and how that map is built is a lever in its own right, open to external information. Traditionally, it is *flat*: one independently parameterized vector per item, with no explicit parameter tying between item rows. Recently, discrete methods have replaced that default, whereas continuous methods have retained it.

Discrete models tokenize each item embedding into codes and emit those codes autoregressively (Deng et al., 2025; Rajput et al., 2023) or by masked diffusion (Shah et al., 2025). They owe much of their strength to those codes being Semantic IDs (SIDs) (Singh et al., 2024). A residual quantizer (van den Oord et al., 2017; Lee et al., 2022; Tong et al., 2026) turns each item's pretrained content embedding into a tuple of codes, so the item vocabulary becomes a clustering (Ju et al., 2025). Two things follow: items sharing a code share the parameters attached to it, so their gradients are aggregated through the shared codeword, introducing explicit cross-item coupling and thereby pooling training signal across items; and because the codes are quantized from a content embedding, that grouping imports semantic structure. However, tokenization has two costs: committing each item to a single codeword at each level generally introduces reconstruction error, and

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.

distinct items may also collide on the same code tuple. A recent line makes the tokenizer itself adaptive by regularizing it (Wang et al., 2024), coupling it to the recommender (Liu et al., 2025a), or relaxing the assignment so gradients pass through it (Li et al., 2026a; Fu et al., 2026), but recommendation is still organized around emitting discrete codes and both costs stand. They belong to tokenizing, not to when the tokenizer is fit.

Continuous models instead represent each item as a vector in $\mathbb{R}^d$ and retrieve those nearest to the one they predict (Wang et al., 2023; Liu et al., 2025b; Li et al., 2026b). They do not tokenize, and are free to seed the content embedding straight into their item table, so semantic structure reaches them without any codes. What they do not obtain automatically is explicit cross-item parameter sharing: every item row remains independently parameterized. This raises our central question: can a continuous generative recommender obtain adaptive cross-item parameter sharing without converting items into discrete tokens?

A simple way to use an SID in a continuous recommender is to replace its flat lookup table with a hard SID table whose assignments are frozen. If the tokenizer assigns item i the code $c_i^{(s)}$ at level s, then its embedding is $\mathbf{x}_i = \sum_s C^{(s)}[c_i^{(s)}] + \mathbf{r}_i$, where $C^{(s)}$ contains the codeword vectors and the item-specific vector $\mathbf{r}_i$ can be initialized from the original content embedding. This preserves continuous retrieval while transferring the SID's parameter sharing: items with the same code update a shared vector. However, our diagnostic comparison (Appendix F, Table 6) shows that, without content embeddings, semantic assignments outperform matched random assignments, indicating that the imported structure is useful. When the full content embedding initializes $\mathbf{r}_i$, however, frozen hard semantic assignments hurt, whereas random assignments remain near the flat baseline. These observations suggest that sharing can help, but that a frozen semantic partition can become restrictive when its underlying continuous representation is already available. This motivates shared structure that adapts jointly with the recommender while retaining item-specific degrees of freedom.

Following these observations, we propose CAP-Mix, a flow recommender whose item table is a co-adaptive mixture of shared prototypes plus an item-specific vector. Rather than selecting one codeword per level, an item *softly* mixes over all of them, so the grouping is differentiable and the ranking objective moves it continuously instead of by discrete reassignment, while the item-specific vector keeps it from being reduced to the group it lands in. We call it *prototype-mixture* item table:

$$\mathbf{x}_i \;=\; \frac{\gamma}{S}\sum_{s=1}^{S} A_i^{(s)} C^{(s)} + \mathbf{r}_i, \qquad \sum_{k=1}^{K} A_i^{(s)}[k] \;=\; 1, \tag{1}$$

with $C^{(s)}$ the K shared prototypes of level s, $A_i^{(s)}$ the weights item i places on them, and $\mathbf{r}_i$ the item's own vector, and $\gamma > 0$ a scale factor, all learned with the flow in a single stage.

Empirically, we evaluate CAP-Mix across 6 public benchmarks (Table 2), where CAP-Mix improves average HR@5 $\uparrow$ and NDCG@5 $\uparrow$ by $+6.9\%$ and $+13.6\%$, respectively, relative to the state-of-the-art hard and soft baselines for each dataset and metric, while delivering 1.4–$161\times$ the inference throughput $\uparrow$ of the evaluated discrete baselines (Table 3).

2 Preliminaries

2.1 Flow-based generative retrieval for recommender systems

Flow matching (Lipman et al., 2023) learns to transform samples from a simple source distribution p_{source} into samples from a target distribution p_{data}. It learns a velocity field $v_\theta(\mathbf{x}, t)$ that tells each sample how to move at time t. Let $\mathcal{I}$ be the item catalog, $H = (i_1, \ldots, i_m)$ a user's interaction history, and i_{m+1} the next item. A flow-based generative recommender can be decomposed into three components. First, the item encoder $E_\theta : \mathcal{I} \to \mathbb{R}^D$ maps each item i to an embedding $\mathbf{x}_i$. Applied elementwise to H, it yields $(\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_m})$. Second, the flow starts from a source $\mathbf{x}_0$ and follows the learned velocity field $v_\theta(\mathbf{x}, t)$ toward a predicted next-item embedding by integrating from $t = 0$ to $t = 1$. Writing v for the exact velocity field, explicit Euler approximates this transport using N evaluations of the learned field:

$$\hat{\mathbf{x}}_1 = \mathbf{x}_0 + \int_0^1 v(\mathbf{x}_t, t)\,\mathrm{d}t \approx \mathbf{x}_0 + h\sum_{n=0}^{N-1} v_\theta(\hat{\mathbf{x}}_{t_n}, t_n), \qquad t_n = nh, \quad h = 1/N. \tag{2}$$

Here, $\hat{\mathbf{x}}_{t_n}$ denotes the numerical state at time t_n, with $\hat{\mathbf{x}}_{t_0} = \mathbf{x}_0$. Depending on the architecture, the encoded history may determine the source, condition the velocity field, or both. Third, the decoder $D_\theta(\hat{\mathbf{x}}_1, H) \in \Delta^{|\mathcal{I}|-1}$ converts this endpoint into a distribution over the catalog by scoring it against the item table $\mathbf{X} = [\mathbf{x}_i^\top]_{i\in\mathcal{I}} \in \mathbb{R}^{|\mathcal{I}|\times D}$. Figure 1(a) illustrates this complete encoder–flow–decoder path. During training, the target endpoint is the next-item embedding $\mathbf{x}_1 = \mathbf{x}_{i_{m+1}}$. For a source $\mathbf{x}_0$ and target $\mathbf{x}_1$, conditional flow matching commonly uses the straight path and regresses the conditional velocity

$$\mathbf{x}_t = (1-t)\mathbf{x}_0 + t\mathbf{x}_1, \qquad v^\star = \mathbf{x}_1 - \mathbf{x}_0, \qquad \mathcal{L}_{\mathrm{CFM}} = \mathbb{E}_{t\sim\mathcal{U}[0,1]} \left\| v_\theta(\mathbf{x}_t, t) - v^\star \right\|_2^2. \tag{3}$$

The same item table therefore embeds the history, defines the flow's training targets, and supports retrieval. Existing flow-based recommenders (Liu et al., 2025b; Li et al., 2026b) focus primarily on the transport, whereas we study how the item-table parameterization can enable sharing across items.

2.2 Semantic IDs

A Semantic ID (SID) represents an item by an ordered tuple of codes, typically obtained by applying residual quantization to a pretrained content embedding (Rajput et al., 2023; Ju et al., 2025). Given L codebooks $Q^{(\ell)} \in \mathbb{R}^{K\times D}$, residual quantization decomposes the content embedding $\mathbf{e}_i$ into selected codewords and a final residual $\boldsymbol{\varepsilon}_i$:

$$\mathrm{SID}(i) = \big(c_i^{(1)}, \ldots, c_i^{(L)}\big), \qquad \mathbf{e}_i = \sum_{\ell=1}^{L} Q^{(\ell)}[c_i^{(\ell)}] + \boldsymbol{\varepsilon}_i, \qquad c_i^{(\ell)} \in \{1, \ldots, K\}. \tag{4}$$

The SID retains the selected indices but not $\boldsymbol{\varepsilon}_i$. Methods differ in how they obtain these assignments: RQ-KMeans selects the nearest centroid (Ju et al., 2025), RQ-VAE learns its quantization space through reconstruction (Lee et al., 2022), and RQ-GMM assigns items using posterior mixture probabilities (Tong et al., 2026). Regardless of the fitting procedure, the SID used at inference is *hard*: each level selects one codeword, whose index becomes that level's code. Items that share a code reuse its parameters, thereby pooling recommendation gradients across their interactions. Deriving the codes from content embeddings also transfers pretrained semantic structure.

Recent methods remove the need for a separately pretrained, frozen tokenizer by adapting it jointly with the recommender (Li et al., 2026a; Fu et al., 2026; Wang et al., 2026), but their deployed IDs remain hard and discard the residual $\boldsymbol{\varepsilon}_i$. SIDs therefore share parameters through discrete code assignments at the cost of quantizing the item representation. Flat-table continuous recommenders avoid quantization but learn each item embedding independently. Our goal is to combine adaptive sharing with trainable continuous representations.

3 Method: CAP-Mix

Following Section 2.1, CAP-Mix uses a continuous flow to predict a next-item embedding and a catalog decoder to model $p(i \mid H)$. Its main contribution is the **item encoder** E_θ, which replaces a flat item table with a *prototype-mixture* parameterization. This representation learns soft item groupings for cross-item sharing while retaining an item-specific vector, and can incorporate pretrained SIDs or content embeddings when available. Figure 1 provides an overview.

3.1 Prototype-mixture item representations

CAP-Mix maintains S shared codebooks $C^{(s)} \in \mathbb{R}^{K\times D}$, whose K rows are D-dimensional prototype vectors. For item i, $A_i^{(s)} \in \Delta^{K-1}$ is a soft assignment over the prototypes in codebook s, and $\mathbf{r}_i \in \mathbb{R}^D$ is an item-specific vector. The item embedding is then

$$\mathbf{x}_i = \underbrace{\mathbf{r}_i}_{\text{identity}} + \underbrace{\frac{\gamma}{S}\sum_{s=1}^{S} A_i^{(s)} C^{(s)}}_{\text{cross-item sharing}}, \qquad A_i^{(s)}[k] \geq 0, \quad \sum_{k=1}^{K} A_i^{(s)}[k] = 1, \tag{5}$$

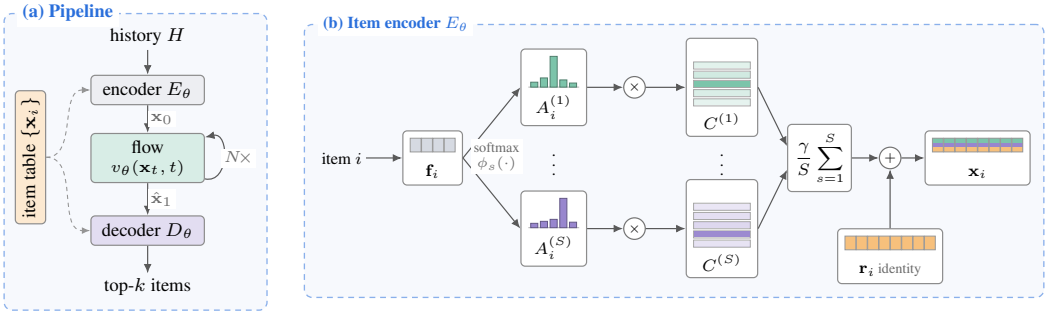


Figure 1: **(a)** A general flow-based retrieval pipeline has three components, and the item table is shared across them: the encoder builds it, the flow is trained to land on one of its rows, and the decoder retrieves against it. **(b)** CAP-Mix mainly changes the encoder of the pipeline: An attention block over the S level tokens maps each item's short feature $\mathbf{f}_i \in \mathbb{R}^{D_f}$ to a *soft* assignment $A_i^{(s)}$ over the K prototypes of level s, which then blends the rows of that level's codebook $C^{(s)}$. This, combined with a trainable identity vector $\mathbf{r}_i$, provides the item embedding $\mathbf{x}_i \in \mathbb{R}^D$. Nothing is quantized or frozen, so the clustering is shaped by the training objective and the assignments are kept soft during training and inference.

where $A_i^{(s)} C^{(s)} \in \mathbb{R}^D$ is a convex mixture of the prototypes, $\gamma > 0$ controls the contribution of the shared component, and $1/S$ keeps its scale stable as S changes.

Cross-item sharing and item-specific memorization. Consider a differentiable objective $\mathcal{L}(\mathbf{X})$ whose dependence on the codebooks and item-specific vectors is only through the composed item table $\mathbf{X}$, and let $\mathbf{g}_i = \nabla_{\mathbf{x}_i}\mathcal{L}$. Differentiating Equation 5 gives

$$\nabla_{C^{(s)}[k]}\mathcal{L} = \frac{\gamma}{S}\sum_{i\in\mathcal{I}} A_i^{(s)}[k]\mathbf{g}_i, \qquad \nabla_{\mathbf{r}_i}\mathcal{L} = \mathbf{g}_i. \tag{6}$$

The prototype channel aggregates recommendation gradients across items by their assignment weights, providing parameter sharing that can improve generalization. Meanwhile, $\mathbf{r}_i$ receives the unrestricted gradient $\mathbf{g}_i$, preserving item-specific memorization and uniqueness even when assignments coincide. Appendix E derives the resulting coupling under fixed assignments.

Relation to flat and hard item tables. Equation 5 contains flat and hard item tables as special cases: $\gamma = 0$ yields the flat $\mathbf{x}_i = \mathbf{r}_i$, while fixed one-hot $A_i^{(s)}$ selects one prototype per codebook as in a hard SID lookup. Also setting $\mathbf{r}_i = \mathbf{0}$ gives the fully quantized case and discards item information not captured by the selected codewords. CAP-Mix instead keeps assignments soft and retains $\mathbf{r}_i$ for item-specific flexibility.

Learning soft assignments. Each item has a learned low-dimensional feature $\mathbf{f}_i \in \mathbb{R}^{D_f}$. We derive one token per codebook from $\mathbf{f}_i$ and process the S tokens jointly with a lightweight attention block:

$$\begin{aligned} \mathbf{z}_i^{(s)} &= W_{\text{in}}^{(s)}\mathbf{f}_i + \mathbf{p}^{(s)}, \quad (\mathbf{h}_i^{(1)}, \dots, \mathbf{h}_i^{(S)}) = \text{Attn}(\mathbf{z}_i^{(1)}, \dots, \mathbf{z}_i^{(S)}), \\ \phi_s(\mathbf{f}_i) &= W_{\text{out}}^{(s)}\mathbf{h}_i^{(s)}, \qquad\qquad A_i^{(s)} = \text{softmax}(\phi_s(\mathbf{f}_i))\,. \end{aligned} \tag{7}$$

Here, $W_{\text{in}}^{(s)}\mathbf{f}_i + \mathbf{p}^{(s)}$ is a codebook-specific affine projection, $\mathbf{h}_i^{(s)}$ is its contextualized output, and $W_{\text{out}}^{(s)}$ maps this output to K assignment logits. This parameterization has three properties. First, because $D_f < D$, $\mathbf{f}_i$ serves as an assignment bottleneck: it stores compact information for selecting shared prototypes, while detailed item-specific information remains in $\mathbf{r}_i$. Second, attention lets the assignments interact: it is bidirectional across codebooks by default and, when an offline residual SID provides a level ordering, causally masked to respect that ordering. Third, no $\arg\max$ or straight-through estimator is used: the assignments remain soft and differentiable during training and inference, and all representation parameters are learned jointly with the recommender.

3.2 Using optional pretrained information

Pretrained information is optional: CAP-Mix can learn entirely from recommendation interactions or incorporate an offline SID or a full content embedding when either is available.

From scratch. CAP-Mix requires neither a tokenizer nor content embeddings. All representation parameters are randomly initialized and learned solely from recommendation interactions. The resulting item groupings are therefore induced by the recommendation objective rather than imported from pretrained content information.

Offline Semantic ID. An offline residual tokenizer supplies both a distribution $\rho_i^{(t)} \in \Delta^{K-1}$ over codes at each depth t and the level ordering induced by its residual construction. We follow this ordering with S parallel causal chains of depth T, sharing the tokenizer prior at each depth across chains:

$$A_i^{(s,t)} = \mathrm{softmax}\Big(\phi_{s,t}(\mathbf{f}_i) + \alpha_t \log\Big(\rho_i^{(t)} + \epsilon_\rho\Big)\Big), \qquad \mathbf{x}_i = \mathbf{r}_i + \frac{\gamma}{S}\sum_{s,t} A_i^{(s,t)} C^{(s,t)}. \tag{8}$$

Here, the sum ranges over $s \in \{1,\dots,S\}$ and $t \in \{1,\dots,T\}$, and $\alpha_t \geq 0$ is a learned trust parameter. Because $\epsilon_\rho > 0$ gives the prior full support, for any finite α_t the learned logits can realize any distribution in the interior of the simplex. The assignments therefore remain soft and differentiable. Equation 5 is recovered when $T = 1$. Appendix D gives the pooling derivation and causal attention structure.

Full content embedding. Let $\bar{\mathbf{e}}_i \in \mathbb{R}^D$ be the PCA-projected and coordinatewise-standardized content embedding. We initialize the item-specific vector from this embedding and set every prototype to zero:

$$\mathbf{r}_i \leftarrow \bar{\mathbf{e}}_i, \qquad C^{(s)} \leftarrow \mathbf{0}. \tag{9}$$

Both $\mathbf{r}_i$ and $C^{(s)}$ remain trainable after initialization. By Equation 5, $\mathbf{x}_i = \bar{\mathbf{e}}_i$ initially, so CAP-Mix and the flat content-initialized baseline begin from the same unquantized item embeddings. The prototype channel then learns cross-item sharing from this common starting point.

3.3 Flow backbone and decoder

We build on the standard flow backbone (Li et al., 2026b), retaining its behavior encoder, velocity field, and base flow objective while replacing its item encoder and decoder. Appendix B provides the full specification. A behavior encoder B_θ maps the m most recent item embeddings to a history-dependent source, while the next-item embedding supplies the target:

$$\mathbf{x}_0 = B_\theta(E_\theta(i_1),\dots,E_\theta(i_m)), \qquad \mathbf{x}_1 = E_\theta(i_{m+1}). \tag{10}$$

We train v_θ on the straight interpolant using the conditional flow-matching loss in Equation 3. Following FlowRec, we also form the one-step endpoint estimate

$$\widetilde{\mathbf{x}}_1 = \mathbf{x}_t + (1-t)v_\theta(\mathbf{x}_t, t). \tag{11}$$

Decoder. For a query $\mathbf{u} \in \mathbb{R}^D$, the decoder combines dot-product scores against the shared item table with a history-dependent co-visitation prior through log-linear pooling:

$$D_\theta(\mathbf{u}, H)[i] = \mathrm{softmax}_{i\in\mathcal{I}}\big(\mathbf{u}^\top \mathbf{x}_i + w \log \widetilde{g}(i \mid H)\big). \tag{12}$$

The prior $\widetilde{g}(i \mid H)$ favors items that frequently co-occur with the items in the user's recent history. We compute it once from the training interactions before model training and keep it fixed. The trust $w \geq 0$ controls its influence. Setting $w = 0$ defines the no-prior ablation with a pure dot-product decoder. Appendix D gives the complete construction.

3.4 Training objectives and inference

Base flow objective. Using the decoder in Equation 12, we apply next-item classification losses to the behavior source and the one-step endpoint:

$$\mathcal{L}_{\mathrm{src}} = -\log D_\theta(\mathbf{x}_0, H)[i_{m+1}], \qquad \mathcal{L}_{\mathrm{align}} = -\log D_\theta(\widetilde{\mathbf{x}}_1, H)[i_{m+1}]. \tag{13}$$

Together with the conditional flow-matching loss, the inherited FlowRec objective is

$$\mathcal{L}_{\mathrm{flow}} = \mathcal{L}_{\mathrm{src}} + \lambda_{\mathrm{cfm}}\mathcal{L}_{\mathrm{CFM}} + \lambda_{\mathrm{align}}\mathcal{L}_{\mathrm{align}}. \tag{14}$$

Here, $\mathcal{L}_{\mathrm{src}}$ makes the history-dependent source predictive of the next item, while $\mathcal{L}_{\mathrm{align}}$ applies the same supervision to the one-step endpoint estimate. λ_{cfm} and λ_{align} are fixed throughout our experiments; their values are given in Section 4.1.

Code matching. The item-level losses can be satisfied through the item-specific vectors $\mathbf{r}_i$ alone, leaving the shared assignments uninformative for retrieval. We therefore train the predicted endpoint to recover the target item's soft assignments, encouraging items retrieved in similar contexts to use compatible prototypes. Let $A^{(s,t)} \in \mathbb{R}^{|\mathcal{I}|\times K}$ collect all item assignments for chain s at depth t, and let $\widetilde{\mathbf{g}}_H = [\widetilde{g}(i \mid H)]_{i\in\mathcal{I}}$. Here and below, $\sum_{s,t}$ ranges over all S chains and T depths, with $T = 1$ outside the residual-SID setting. We define

$$\boldsymbol{\pi}_H^{(s,t)} = \mathrm{sg}\Big(A^{(s,t)\top}\widetilde{\mathbf{g}}_H\Big), \qquad q_H^{(s,t)} = \mathrm{softmax}\Big(\widetilde{\mathbf{x}}_1 C^{(s,t)\top} + w \log \boldsymbol{\pi}_H^{(s,t)}\Big). \tag{15}$$

Here, $\boldsymbol{\pi}_H^{(s,t)}$ marginalizes the fixed item-space prior into code space. The stop-gradient operator sg prevents this computation from updating the assignments. The code matching loss is

$$\mathcal{L}_{\mathrm{match}} = -\frac{1}{ST}\sum_{s,t}\Big\langle A_{i_{m+1}}^{(s,t)}, \log q_H^{(s,t)}\Big\rangle. \tag{16}$$

Gradients still pass through the target assignment and, through $q_H^{(s,t)}$, to the prototypes and flow model, allowing these components to co-adapt.

Maintaining prototype utilization. Because the item-specific channel can support retrieval even when assignments concentrate on only a few prototypes, we penalize catalog-wide utilization below a target K_{eff}:

$$\bar{A}^{(s,t)} = \frac{1}{|\mathcal{I}|}\sum_{i\in\mathcal{I}} A_i^{(s,t)}, \qquad \mathcal{L}_{\mathrm{bal}} = \frac{1}{ST}\sum_{s,t}\Big[\log K_{\mathrm{eff}} - \mathcal{H}\Big(\bar{A}^{(s,t)}\Big)\Big]_+, \tag{17}$$

where $\mathcal{H}$ is entropy and $[z]_+ = \max(z, 0)$. Since $\exp(\mathcal{H}(\bar{A}^{(s,t)}))$ is the effective number of utilized prototypes, the loss vanishes once it reaches K_{eff}. The loss depends only on the catalog-wide marginal $\bar{A}^{(s,t)}$, leaving individual assignments free to be soft or sharp without forcing discretization.

Full objective. All trainable components are optimized jointly. Given weights $\lambda_{\mathrm{match}}, \lambda_{\mathrm{bal}} \geq 0$,

$$\mathcal{L} = \mathcal{L}_{\mathrm{flow}} + \lambda_{\mathrm{match}}\mathcal{L}_{\mathrm{match}} + \lambda_{\mathrm{bal}}\mathcal{L}_{\mathrm{bal}}. \tag{18}$$

Inference. After training, we materialize the item table $\mathbf{X}$ from Equation 5, retaining the learned soft mixtures without discretization. Given a history H, we look up its item embeddings and compute the behavior source $\mathbf{x}_0$. Starting from $\mathbf{x}_0$, we numerically integrate the learned velocity field as in Equation 2 to obtain $\hat{\mathbf{x}}_1$. The decoder in Equation 12 then scores the catalog, and we return the top-k items. Because the table is materialized once, the prototype-mixture encoder adds no per-query computation relative to a flat item table.

4 Experiments

Our experiments address three questions: (**RQ1**) when does a prototype-mixture item table improve over flat and hard SID alternatives under different levels of pretrained information; (**RQ2**) how does the complete CAP-Mix system compare with continuous and discrete generative recommenders in predictive performance and efficiency; and (**RQ3**) how do semantic initialization, the co-visitation prior, and target-item frequency affect the observed gains?

4.1 Setup

Datasets and preprocessing. We evaluate on 6 public benchmarks: Amazon *Beauty* (2014) (McAuley et al., 2015; He & McAuley, 2016), Amazon *Industrial & Scientific* and Amazon *Office* (2023) (Hou et al., 2026), *MovieLens-1M* (Harper & Konstan, 2015), *Yelp-2022* (Yelp Inc., 2022), and *Steam-2018* (Kang & McAuley, 2018). All datasets are processed identically following prior work (Geng et al., 2022; Rajput et al., 2023; Shah et al., 2025): iterative 5-core filtering, consecutive-repeat removal, chronological ordering, and leave-one-out splitting, with each user's penultimate and final interactions used for validation and testing. Appendix A reports dataset statistics, sources, metadata, and complete preprocessing details.

Table 1: **Item tables across three information regimes.** Each cell reports test HR@5 ↑ (top) and change from the dataset's scratch FLAT anchor (bottom; *ref.*). The final Δ compares CAP-Mix with FLAT (scratch), HARD (offline SID), or FLAT (PCA-text). HARD uses the same tokenizer; all variants have matched backbones and non-table hyperparameters.

	FROM SCRATCH			OFFLINE SID			PCA-TEXT			
dataset	FLAT	CAP-Mix	*vs.* FLAT	HARD	CAP-Mix	*vs.* HARD	FLAT	HARD	CAP-Mix	*vs.* FLAT
Beauty	5.77 *ref.*	5.89 +2.08%	+2.08%	5.80 +0.52%	6.32 +9.53%	+8.97%	6.69 +15.94%	6.43 +11.44%	6.79 +17.68%	+1.49%
Industrial	2.75 *ref.*	2.88 +4.73%	+4.73%	3.02 +9.82%	3.47 +26.18%	+14.90%	3.81 +38.55%	3.52 +28.00%	3.79 +37.82%	−0.52%
Steam	8.29 *ref.*	8.35 +0.72%	+0.72%	8.32 +0.36%	8.69 +4.83%	+4.45%	8.13 −1.93%	8.19 −1.21%	8.30 +0.12%	+2.09%

Evaluation protocol. All models are trained and evaluated on identical item sequences using full-catalog ranking without candidate sampling. We report HR@5 ↑ and NDCG@5 ↑ as three-seed means scaled by 100; Appendix G additionally reports $k \in \{10, 20\}$. We use early stopping, select the checkpoint with the best validation performance, and evaluate only that checkpoint on the test split.

Information regimes. To separate content information from parameter-sharing benefits, we evaluate three regimes that differ in the pretrained information available. FROM SCRATCH provides none: item representations are initialized randomly and learned from interactions. OFFLINE SID provides the stored codes and posterior responsibilities of a balanced RQ-GMM (see Appendix C), but not the content embeddings from which they were computed. PCA-TEXT provides the full content embeddings, projected through PCA to D dimensions and kept trainable during training.

Implementation and model selection. All controlled item-table variants use the same D=128 flow backbone, three-item histories, three Euler steps, and matched non-table hyperparameters. We encode item metadata once with frozen Qwen3-Embedding-4B (Zhang et al., 2025). In PCA-TEXT, we initialize the trainable item-specific vectors with standardized D-dimensional PCA projections of the content embeddings. CAP-Mix zero-initializes its prototypes, making its initial table identical to the flat baseline. We select configurations by the three-seed validation geometric mean of HR@5 ↑ and NDCG@5 ↑. Appendices A–C give the SID construction and remaining setup details.

4.2 RQ1: WHEN DO PROTOTYPE MIXTURES HELP?

Our central hypothesis is that continuous item embeddings benefit from shared prototypes when that structure complements, rather than replaces, an item-specific continuous representation. We test this by holding the flow backbone and training protocol fixed while varying the item-table parameterization. Because the same table embeds histories, defines the flow targets, and supports retrieval, ranking differences measure the downstream utility of the learned item embeddings.

Item-table variants. The FLAT table uses $\mathbf{x}_i = \mathbf{r}_i$. The HARD SID table uses $\mathbf{x}_i = \mathbf{r}_i + \sum_s C^{(s)}[c_i^{(s)}]$, with trainable codewords and item-specific vectors but frozen one-hot assignments. CAP-Mix instead uses the prototype-mixture representation of Equation (5). All variants share the same backbone and non-table hyperparameters.

Information available to each variant. Table 1 compares the item-table variants under three settings. In FROM SCRATCH, the flat and CAP-Mix tables are randomly initialized and learn only from interactions. In OFFLINE SID, the hard table uses the tokenizer's stored codes as fixed assignments, whereas CAP-Mix uses its posterior responsibilities as the assignment prior in Equation (8); neither receives the original content embedding. In PCA-TEXT, every variant initializes its item-specific vectors from the same PCA projection. The hard table uses the tokenizer's fixed codes, whereas CAP-Mix initializes its prototypes to zero, making its initial table identical to the flat table.

Prototype sharing helps without pretrained content. In FROM SCRATCH, CAP-Mix improves HR@5 ↑ over the flat table by +2.08% on Beauty, +4.73% on Industrial, and +0.72% on Steam. Thus, shared prototypes learn useful cross-item structure from interactions alone, producing item embeddings that are more effective for next-item ranking than independently parameterized rows.

Table 2: **Main results**, HR@5 ↑ and NDCG@5 ↑ (mean±std, ×100). For each model–dataset pair, both metrics come from one run selected by their validation geometric mean; [s] marks scratch, [i] offline SID, and [p] PCA-text. **Bold**/underline mark best/second; Δ compares CAP-Mix with the best baseline.

	Beauty		Indus.		Steam		ML-1M		Yelp		Office	
	HR@5 ↑	NDCG@5 ↑	HR@5 ↑	NDCG@5 ↑	HR@5 ↑	NDCG@5 ↑	HR@5 ↑	NDCG@5 ↑	HR@5 ↑	NDCG@5 ↑	HR@5 ↑	NDCG@5 ↑
FlowRec	6.27[p] ±0.18	4.03 ±0.03	3.62[p] ±0.02	2.04 ±0.03	8.28[s] ±0.33	5.37 ±0.30	15.36[p] ±0.20	10.14 ±0.12	2.98[p] ±0.05	1.86 ±0.03	2.90[p] ±0.02	1.67 ±0.02
TIGER	3.51[s] ±0.07	2.54 ±0.05	2.05[p] ±0.03	1.34 ±0.02	8.43[i] ±0.10	5.01 ±0.03	12.59[s] ±0.40	8.47 ±0.28	1.29[p] ±0.07	0.80 ±0.04	1.52[p] ±0.01	0.98 ±0.01
MaskGR	5.06[i] ±0.13	3.28 ±0.12	3.46[p] ±0.03	2.24 ±0.03	8.64[p] ±0.00	**5.57** ±0.01	15.64[s] ±0.40	10.23 ±0.28	2.99[s] ±0.04	1.87 ±0.03	2.29[s] ±0.02	1.53 ±0.01
CAP-Mix	**6.79**[p] ±0.06	**4.73** ±0.04	**3.79**[p] ±0.01	**2.34** ±0.03	**8.69**[i] ±0.29	**5.57** ±0.19	**17.25**[p] ±0.13	**11.71** ±0.11	**3.18**[p] ±0.02	**2.03** ±0.02	**3.22**[p] ±0.01	**2.28** ±0.01
Δ (%)	+8.3%	+17.4%	+4.7%	+4.5%	+0.6%	0.0%	+10.3%	+14.5%	+6.4%	+8.6%	+11.0%	+36.5%

Table 3: **Inference efficiency** across five datasets (min–max from unrounded values). Hot-parameter ratio ↑ and MFLOP/query ratio ↑ are baseline/CAP-Mix; throughput ratio ↑ is CAP-Mix/baseline.

baseline	hot-parameter ratio ↑	MFLOP/query ratio ↑	throughput ratio ↑
TIGER	0.9–10.0×	23–265×	1.9–7.1×
MaskGR	1.4–5.4×	3.7–2,654×	1.4–161×

An offline SID is more useful as a prior than as a fixed assignment. The hard SID table exceeds the scratch anchor on all three datasets, confirming that the tokenizer carries useful information. Given the same tokenizer, CAP-Mix improves HR@5 ↑ over the hard table by 4.45%–14.90%, showing that differentiable assignments use this information more effectively than frozen one-hot assignments.

Prototype sharing complements the full content embedding. In PCA-TEXT, content seeding already strengthens the flat table on the Amazon datasets. Starting from the same embeddings, CAP-Mix improves it by $+1.49\%$ on Beauty and $+2.09\%$ on Steam while essentially preserving its Industrial result (-0.52%). In contrast, the hard SID table changes HR@5 ↑ by -3.89% and -7.61% relative to the seeded flat table on the Amazon datasets. Thus, prototype mixtures add cross-item sharing while retaining the unquantized initialization improvement.

Across all three regimes, prototype mixtures improve how continuous item embeddings use the information available by introducing cross-item pooling without quantizing their representations.

4.3 RQ2: Comparison with generative recommenders

We compare CAP-Mix with continuous FlowRec (Li et al., 2026b), autoregressive TIGER (Rajput et al., 2023), and masked-diffusion MaskGR (Shah et al., 2025) in all three information regimes, selecting one configuration per model and dataset by the validation geometric mean of HR@5 ↑ and NDCG@5 ↑. CAP-Mix and FlowRec use three-item histories; TIGER and MaskGR retain their released 120-token limit and decode one top-20 list with beam search. Appendix B specifies the tokenizers, initializations, resulting item-level history lengths, and decoding protocol.

Table 2 reports the complete-system comparison at $k{=}5$, with results at $k{=}10$ and 20 in Appendix G. CAP-Mix achieves the highest HR@5 ↑ on all six datasets and the highest or tied-for-highest NDCG@5 ↑ on all six. Averaged across datasets, it improves over the strongest evaluated baseline for each dataset and metric by $+6.9\%$ in HR@5 ↑ and $+13.6\%$ in NDCG@5 ↑. For CAP-Mix, validation selects PCA-TEXT on five of the six datasets and OFFLINE SID on Steam, suggesting that direct content initialization is generally preferable when the original embeddings are available.

Inference efficiency. At inference, CAP-Mix materializes the composed item embeddings once, so its soft assignments add no per-query computation. Table 3 shows that CAP-Mix consistently requires fewer MFLOP/query ↓ and achieves higher throughput ↑ than both discrete baselines across the five profiled datasets. The advantage is largest when the baselines generate multi-code SIDs and narrows for atomic-token configurations. The parameter advantage depends on catalog size: on Yelp, the continuous item table makes CAP-Mix slightly larger than TIGER. Appendix B documents the measurement protocol, and Table 5 reports the full per-dataset measurements.

4.4 RQ3: Robustness and diagnostics

Motivating controls. Appendix F compares hard semantic assignments with random assignments matched in depth, codebook size, and codeword vectors. Without content initialization, semantic assignments win on all three datasets, confirming that they carry useful structure beyond parameter sharing. With PCA initialization, they underperform the flat and random controls on both Amazon datasets, showing that this structure can become restrictive when frozen.

Complementarity with the retrieval prior. Without the co-visitation prior, prototype sharing is beneficial in all regimes on both Amazon datasets and retains a $+6.91\%$ HR@5 $\uparrow$ advantage over hard SID on Steam (Table 7), confirming that prototype sharing and the prior are complementary.

How do gains vary with item frequency? In the from scratch scenario, the frequency-stratified results are mixed. However, when pretrained tokenizer or content information is available, relative gains are larger for rare targets than frequent ones in 11 of 12 dataset–comparison–metric combinations for HR@5 $\uparrow$ and NDCG@10 $\uparrow$, consistent with transfer to less-observed items (Table 8).

5 Related Work

Discrete generative recommenders and semantic IDs. The discrete branch commonly replaces an atomic item identifier with a *semantic ID* (SID) by quantizing a pretrained content embedding (Ju et al., 2025). This is generally done through residual schemes that cluster either directly in the embedding space (RQ-KMeans and RQ-GMM (Deng et al., 2025; Tong et al., 2026)) or in a learned latent space (RQ-VAE (van den Oord et al., 2017; Lee et al., 2022)). A recommender then predicts the resulting tuple autoregressively, as in TIGER (Rajput et al., 2023), OneRec (Deng et al., 2025; Zhou et al., 2025), and MiniOneRec (Kong et al., 2025), or through masked discrete diffusion, as in MaskGR (Shah et al., 2025). SIDs provide two distinct benefits. The first, *parameter sharing*, arises because reusing codewords ties parameters across items and pools supervision from their interactions. The second, *semantic transfer*, comes from deriving the assignments from pretrained content embeddings, which places semantically related items under shared parameters. Together, these benefits improve generalization (Ju et al., 2025; Singh et al., 2024). However, an SID exposes only hard code assignments and preserves a partition of the content space rather than its full continuous geometry.

Diffusion and flow recommenders. Continuous generative recommenders replace discrete token prediction with generation in a continuous space. DiffRec (Wang et al., 2023) denoises a user's interaction vector, whereas DreamRec (Yang et al., 2023) and DiffuRec (Li et al., 2023) diffuse a target item embedding for next-item prediction. FMRec (Liu et al., 2025b) and FlowRec (Li et al., 2026b) use flow matching (Lipman et al., 2023) to transport a source distribution to the target embedding, with FlowRec conditioning the source on the user's history. These methods improve the generative dynamics while retaining a flat item table, thereby avoiding discretization into code sequences but forgoing the cross-item parameter sharing induced by shared SID codewords. CAP-Mix retains continuous generation while introducing that sharing through co-adaptive mixtures of prototypes.

6 Conclusion

We presented CAP-Mix, a flow-based recommender whose item table combines differentiable assignments over shared codebooks with an item-specific vector, all trained jointly with the recommendation objective. The shared codewords introduce explicit cross-item parameter coupling, while the item-specific vectors retain unrestricted per-item degrees of freedom. Across 6 public benchmarks, CAP-Mix improves average HR@5 $\uparrow$ and NDCG@5 $\uparrow$ relative to the strongest evaluated baselines, while providing higher throughput $\uparrow$ than the evaluated discrete baselines.

AI use statement

In this work, we used generative AI tools to assist in implementing methods and cleaning and reformatting datasets. We have not used generative AI tools to help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving

mathematical claims, design or provide feedback on research methodology or experiments, propose or refine hypotheses, or interpret results, and the use of generative AI tools to generate synthetic data sets, assist in the writing of proofs, assist with translation, and support qualitative and thematic data analysis was not applicable to this work. Additionally, we used generative AI tools to create or modify scientific figures or images, create or edit software code, draft parts of a research paper, edit a research paper to improve readability, and identify relevant literature. We have reviewed all AI-assisted work. We used an LLM primarily to improve the readability of the paper. We reviewed its suggestions for correctness and ensured that all claims and content remained faithful. We also reviewed and tested AI-assisted code for correctness. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

Ethics statement

This work studies next-item retrieval on six public benchmark datasets that contain no personally identifying information beyond standard interaction logs. In deployment, next-item recommenders may amplify popularity bias and feedback loops, reduce exposure diversity, or reproduce biases in interaction histories and item metadata. Our offline evaluation does not assess these risks. Thus, deployment would require application-specific privacy, fairness, and diversity evaluation.

Reproducibility statement

The method and training objective are specified in Section 3, the experimental protocol and model-selection procedure in Section 4.1, and dataset processing, implementation, tokenizer, and prior details in Appendices A–D. All recommendation-quality results are three-seed means. Configurations are selected using the three-seed validation geometric mean of HR@5 $\uparrow$ and NDCG@5 $\uparrow$, and test results are reported only after selection. We will release code, processed datasets, and processing scripts to reproduce training and evaluation upon acceptance.

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019. doi: 10.1145/3292500.3330701. URL https://doi.org/10.1145/3292500.3330701.

David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pp. 1027–1035, USA, 2007. Society for Industrial and Applied Mathematics. ISBN 9780898716245. URL https://dl.acm.org/doi/10.5555/1283383.1283494.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011. URL https://proceedings.neurips.cc/paper_files/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf.

Paul S. Bradley, Kristin P. Bennett, and Ayhan Demiriz. Constrained k-means clustering. Technical Report MSR-TR-2000-65, Microsoft Research, 2000. URL https://www.microsoft.com/en-us/research/publication/constrained-k-means-clustering/.

Gilles Celeux and Gérard Govaert. A classification EM algorithm for clustering and two stochastic versions. *Computational Statistics & Data Analysis*, 14(3):315–332, 1992. doi: 10.1016/0167-9473(92)90042-E. URL https://doi.org/10.1016/0167-9473(92)90042-E.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B*, 39:1–38, 1977. URL http://web.mit.edu/6.435/www/Dempster77.pdf.

Jiaxin Deng, Shiyao Wang, Kuo Cai, Lejian Ren, Qigen Hu, Weifeng Ding, Qiang Luo, and Guorui Zhou. Onerec: Unifying retrieve and rank with generative recommender and iterative preference alignment, 2025. URL https://arxiv.org/abs/2502.18965.

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*, SIGIR '26, pp. 369–379, New York, NY, USA, 2026. Association for Computing Machinery. ISBN 9798400725999. doi: 10.1145/3805712.3809641. 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*, RecSys '22, pp. 299–315, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450392785. doi: 10.1145/3523227.3546767. URL https://doi.org/10.1145/3523227.3546767.

F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4), December 2015. ISSN 2160-6455. doi: 10.1145/2827872. URL https://doi.org/10.1145/2827872.

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*, WWW '16, pp. 507–517, Republic and Canton of Geneva, CHE, 2016. International World Wide Web Conferences Steering Committee. ISBN 9781450341431. doi: 10.1145/2872427.2883037. URL https://doi.org/10.1145/2872427.2883037.

Yupeng Hou, Jiacheng Li, Xiangjun Fu, Zhankui He, An Yan, Xiusi Chen, and Julian McAuley. Bridging language and items for retrieval and recommendation: Benchmarking LLMs as semantic encoders. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3251–3265, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.147. URL https://aclanthology.org/2026.acl-long.147/.

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)*, 2025. doi: 10.1145/3746252.3761612. URL https://doi.org/10.1145/3746252.3761612.

Wang-Cheng Kang and Julian McAuley. Self-Attentive Sequential Recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*, pp. 197–206, Los Alamitos, CA, USA, November 2018. IEEE Computer Society. doi: 10.1109/ICDM.2018.00035. URL https://doi.ieeecomputersociety.org/10.1109/ICDM.2018.00035.

Xiaoyu Kong, Leheng Sheng, Junfei Tan, Yuxin Chen, Jiancan Wu, An Zhang, Xiang Wang, and Xiangnan He. Minionerec: An open-source framework for scaling generative recommendation, 2025. URL https://arxiv.org/abs/2510.24431.

Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. URL https://openaccess.thecvf.com/content/CVPR2022/html/Lee_Autoregressive_Image_Generation_Using_Residual_Quantization_CVPR_2022_paper.html.

Jialei Li, Yang Zhang, Yimeng Bai, Shuai Zhu, Ziqi Xue, Xiaoyan Zhao, Dingxian Wang, Frank Yang, Andrew Rabinovich, and Xiangnan He. Unigrec: Unified generative recommendation with soft identifiers for end-to-end optimization, 2026a. URL https://arxiv.org/abs/2601.17438.

Li Li, Mingyue Cheng, Yuyang Ye, Zhiding Liu, and Enhong Chen. Flowrec: Prior-informed flow matching for efficient sequential recommendation generation. In Hyungsoo Jung, Tianzheng Wang, Masashi Toyoda, Hyuk-Yoon Kwon, and Jae-woong Lee (eds.), *Database Systems for Advanced Applications - 31st International Conference, DASFAA 2026, Jeju, South Korea, April 27-30, 2026, Proceedings, Part I*, volume 16535 of *Lecture Notes in Computer Science*, pp. 206–222. Springer, 2026b. doi: 10.1007/978-981-92-0363-5_13. URL https://doi.org/10.1007/978-981-92-0363-5_13.

Zihao Li, Aixin Sun, and Chenliang Li. Diffurec: A diffusion model for sequential recommendation. *ACM Trans. Inf. Syst.*, 42(3), December 2023. ISSN 1046-8188. doi: 10.1145/3631116. URL https://doi.org/10.1145/3631116.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Enze Liu, Bowen Zheng, Cheng Ling, Lantao Hu, Han Li, and Wayne Xin Zhao. Generative recommender with end-to-end learnable item tokenization. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '25, pp. 729–739, New York, NY, USA, 2025a. Association for Computing Machinery. ISBN 9798400715921. doi: 10.1145/3726302.3729989. URL https://doi.org/10.1145/3726302.3729989.

Feng Liu, Lixin Zou, Xiangyu Zhao, Min Tang, Liming Dong, Dan Luo, Xiangyang Luo, and Chenliang Li. Flow matching based sequential recommender model. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence*, IJCAI '25, 2025b. ISBN 978-1-956792-06-5. doi: 10.24963/ijcai.2025/346. URL https://doi.org/10.24963/ijcai.2025/346.

S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2): 129–137, 1982. doi: 10.1109/TIT.1982.1056489. URL https://doi.org/10.1109/TIT.1982.1056489.

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*, SIGIR '15, pp. 43–52, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450336215. doi: 10.1145/2766462.2767755. URL https://doi.org/10.1145/2766462.2767755.

Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Q. Tran, Jonah Samost, Maciej Kula, Ed H. Chi, and Maheswaran Sathiamoorthy. Recommender systems with generative retrieval. In *Advances in Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=BJ0fQUU32w.

Kulin Shah, Bhuvesh Kumar, Neil Shah, and Liam Collins. Masked diffusion for generative recommendation, 2025. URL https://arxiv.org/abs/2511.23021.

Anima Singh, Trung Vu, Nikhil Mehta, Raghunandan Keshavan, Maheswaran Sathiamoorthy, Yilin Zheng, Lichan Hong, Lukasz Heldt, Li Wei, Devansh Tandon, Ed Chi, and Xinyang Yi. Better generalization with semantic ids: A case study in ranking for recommendations. In *Proceedings of the 18th ACM Conference on Recommender Systems*, RecSys '24, pp. 1039–1044, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400705052. doi: 10.1145/3640457.3688190. URL https://doi.org/10.1145/3640457.3688190.

Ziye Tong, Jiahao Liu, Weimin Zhang, Hongji Ruan, Derick Tang, Zhanpeng Zeng, Qinsong Zeng, Peng Zhang, Tun Lu, and Ning Gu. Rq-gmm: Residual quantized gaussian mixture model for multimodal semantic discretization in ctr prediction. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '26, pp. 4187–4192, New York, NY, USA, 2026. Association for Computing Machinery. ISBN 9798400725999. doi: 10.1145/3805712.3809968. URL https://doi.org/10.1145/3805712.3809968.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/7a98af17e63a0ac09ce2e96d03992fbc-Paper.pdf.

Huanjie Wang, Xinchen Luo, Honghui Bao, Zhang Zixing, Lejian Ren, Yunfan Wu, Hongwei Zhang, Liwei Guan, and Guang Chen. Pit: A dynamic personalized item tokenizer for end-to-end generative recommendation, 2026. URL https://arxiv.org/abs/2602.08530.

Wenjie Wang, Yiyan Xu, Fuli Feng, Xinyu Lin, Xiangnan He, and Tat-Seng Chua. Diffusion recommender model. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '23, pp. 832–841, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450394086. doi: 10.1145/3539618.3591663. URL https://doi.org/10.1145/3539618.3591663.

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*, CIKM '24, pp. 2400–2409, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704369. doi: 10.1145/3627673.3679569. URL https://doi.org/10.1145/3627673.3679569.

Zhengyi Yang, Jiancan Wu, Zhicai Wang, Xiang Wang, Yancheng Yuan, and Xiangnan He. Generate what you prefer: Reshaping sequential recommendation via guided diffusion. In *Advances in Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=oFpBnt6bgC.

Yelp Inc. Yelp open dataset. https://business.yelp.com/data/resources/open-dataset/, 2022.

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

Wayne Xin Zhao, Shanlei Mu, Yupeng Hou, Zihan Lin, Yushuo Chen, Xingyu Pan, Kaiyuan Li, Yujie Lu, Hui Wang, Changxin Tian, Yingqian Min, Zhichao Feng, Xinyan Fan, Xu Chen, Pengfei Wang, Wendi Ji, Yaliang Li, Xiaoling Wang, and Ji-Rong Wen. Recbole: Towards a unified, comprehensive and efficient framework for recommendation algorithms. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, CIKM '21, pp. 4653–4664, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450384469. doi: 10.1145/3459637.3482016. URL https://doi.org/10.1145/3459637.3482016.

Guorui Zhou, Jiaxin Deng, Jinghao Zhang, Kuo Cai, Lejian Ren, Qiang Luo, Qianqian Wang, Qigen Hu, Rui Huang, Shiyao Wang, Weifeng Ding, Wuchao Li, Xinchen Luo, Xingmei Wang, Zexuan Cheng, Zixing Zhang, Bin Zhang, Boxuan Wang, Chaoyi Ma, Chengru Song, Chenhui Wang, Di Wang, Dongxue Meng, Fan Yang, Fangyu Zhang, Feng Jiang, Fuxing Zhang, Gang Wang, Guowang Zhang, Han Li, Hengrui Hu, Hezheng Lin, Hongtao Cheng, Hongyang Cao, Huanjie Wang, Jiaming Huang, Jiapeng Chen, Jiaqiang Liu, Jinghui Jia, Kun Gai, Lantao Hu, Liang Zeng, Liao Yu, Qiang Wang, Qidong Zhou, Shengzhe Wang, Shihui He, Shuang Yang, Shujie Yang, Sui Huang, Tao Wu, Tiantian He, Tingting Gao, Wei Yuan, Xiao Liang, Xiaoxiao Xu, Xugang Liu, Yan Wang, Yi Wang, Yiwu Liu, Yue Song, Yufei Zhang, Yunfan Wu, Yunfeng Zhao, and Zhanyu Liu. Onerec technical report, 2025. URL https://arxiv.org/abs/2506.13695.

A DATASET DETAILS

Sources. Most datasets are taken as RecBole atomic files (Zhao et al., 2021) from the RecSysDatasets[1] distribution. Amazon's Industrial & Scientific and Office are taken directly from the raw Amazon Reviews 2023 website[2]. Regardless of the source, all datasets are reduced to `(user, item, timestamp)` triples and then processed through the same code path.

Table 4: **Datasets after iterative 5-core filtering**. *Avg. length* is interactions per user, *Median inter./item* is the median number of interactions a catalog item receives, and sparsity is $1 - \#\text{interactions}/(\#\text{users}\cdot\#\text{items})$. *Eff. rank* is $\exp H(p)$ over the normalized eigenvalue spectrum of the content embedding, out of 2560, and *Var. kept* is the share of that variance surviving the projection to D.

	Catalog						Content embedding	
Dataset	#Users	#Items	#Interactions	Avg. length	Median inter./item	Sparsity (%)	Eff. rank	Var. kept (%)
Amazon Beauty	22,363	12,101	198,502	8.9	9	99.927	219	73.2
Amazon Industrial & Scientific	51,322	25,923	417,422	8.1	9	99.969	240	73.5
Amazon Office	223,996	77,655	1,810,031	8.1	11	99.990	213	75.0
MovieLens-1M	6,040	3,416	999,611	165.5	146	95.155	326	66.0
Yelp-2022	272,547	111,260	4,185,885	15.4	14	99.986	279	71.3
Steam	25,256	4,087	325,320	12.9	18	99.685	304	67.6

Preprocessing. Every dataset follows the same procedure. First, we apply an *iterative* k-core filter ($k = 5$). Removing a low-count user can push an item below the threshold and vice versa, so we alternately drop users with fewer than k interactions and items with fewer than k interactions until a pass removes nothing. Second, each user's interactions are sorted chronologically by timestamp. Third, we collapse consecutive repeats of the same item and then *re-run* the k-core filter, iterating both to a fixed point. Iterating guarantees every surviving user has at least k distinct items and therefore appears in all three splits. In our experiments, datasets vary from 1 to 4 collapse-and-re-core rounds. We then assign dense user and item IDs. Finally, each user's sequence $(i_1, \ldots, i_n)$ is split using a leave-one-out scheme: the test instance is the full sequence with target i_n, the validation instance is $(i_1, \ldots, i_{n-1})$ with target i_{n-1}, and training uses $(i_1, \ldots, i_{n-2})$. All families consume identical item sequences.

Text embedding. Every dataset has a deterministic per-item text template, filled from the item metadata and left empty where a field is missing: Amazon uses `Title; Brand; Categories; Price`, following TIGER (Rajput et al., 2023); MovieLens uses `Title; Release year; Genres` plus the most frequent community tags; Yelp uses `Name; City, State; Categories`; and Steam uses `Title; Developer; Publisher; Genres; Tags`. Each string is encoded once with a frozen Qwen3-Embedding-4B (Zhang et al., 2025) using last-token pooling, giving $\mathbf{q}_i \in \mathbb{R}^{2560}$.

SID pretraining. The offline SIDs are produced by the balanced RQ-GMM of Appendix C. We fit 4 levels with $K \in \{256, 512\}$, plus a fifth de-duplication digit that makes identifiers unique. The hard RQ-SID baseline consumes the resulting codes directly, while CAP-Mix consumes the per-level posterior responsibilities as the prior in Equation (8).

Evaluation. Evaluation ranks the target against the *full* catalog, with no candidate sampling and no filtering of previously seen items, so every model scores all $|\mathcal{I}|$ items for every test user. We report hit rate and normalized discounted cumulative gain. Hit rate (HR@k ↑) is the fraction of users whose held-out item appears among the k highest-ranked items. Normalized discounted cumulative gain (NDCG@k ↑) additionally rewards placing that item higher within the top k. We report $k \in \{5, 10, 20\}$. The two generative baselines decode differently from CAP-Mix but are scored on the same ranked list: TIGER produces its top-20 by autoregressive beam search over code positions, and MaskGR by beam search over the positions it unmasks. Every number is a mean over three seeds, taken from the checkpoint with the best validation performance in each run.

[1] `https://github.com/RUCAIBox/RecSysDatasets`

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

B IMPLEMENTATION DETAILS

Architecture and optimization. The history encoder is a bidirectional Transformer with rotary positions, RMSNorm, and pre-norm blocks (4 layers, 4 heads, and hidden dimension D=128), reading the m=3 most recent items as one token each. The time code $\varphi(t)$ is produced by passing a 128-dimensional sinusoidal embedding through a two-layer MLP with SiLU activations. The velocity field is a second two-layer SiLU MLP with hidden width $2D$=256, applied to the concatenation of the modulated state and $\varphi(t)$. We use neither normalization nor FiLM conditioning:

$$v_\theta(\mathbf{x}_t, t) = \mathrm{MLP}([g(\mathbf{x}_t); \varphi(t)]), \qquad g(\mathbf{x}_t) = \begin{cases} \boldsymbol{\lambda} \odot \mathbf{x}_t, & \lambda_j \overset{\text{iid}}{\sim} \mathcal{N}(\delta, \delta^2) \quad \text{during training,} \\ \delta \mathbf{x}_t, & \text{during evaluation,} \end{cases} \tag{19}$$

where $\delta = 10^{-3}$. The modulation vector is redrawn for every call during training and replaced by its mean during evaluation. Item features have dimension D_f=32. The assignment head in Equation (7) uses a separate affine input projection and output projection for each codebook. With $T = 1$, the S codebook tokens are processed jointly by a single-layer, single-head bidirectional self-attention block. With a residual SID, each of the S chains contains T tokens and is processed independently with causal attention following the tokenizer's level order; the depth-t prior is shared across chains. We optimize with AdamW at learning rate 10^{-3} and no weight decay, use dropout 0.3 on item embeddings and 0.5 on hidden states, and stop early with patience 50. Inference uses N=3 Euler steps.

Hyperparameter search. We fix λ_{cfm}=10 and λ_{align}=1, use Optuna's TPE sampler (Bergstra et al., 2011; Akiba et al., 2019), and select configurations by the three-seed validation geometric mean of HR@5 ↑ and NDCG@5 ↑. We independently sample λ_{match} log-uniformly from $[10^{-4}, 4]$, λ_{bal} log-uniformly from $[10^{-4}, 1]$, and K_{eff}/K log-uniformly from $[0.02, 1]$. We search $S \in \{1, 2, 4\}$ in all settings; $T = 1$ by default, whereas $T \in \{1, 2, 3\}$ is selected jointly with S in the offline-SID setting.

PCA-text initialization. Within each run, we fit PCA on the catalog's 2560-dimensional content embeddings and project them to D=128. We then standardize each projected coordinate across the catalog to zero mean and unit variance, preventing any principal direction from dominating the initialization. For the continuous models, these standardized vectors initialize the trainable item-specific vectors $\mathbf{r}_i$. CAP-Mix initializes every prototype to zero, so Equation (5) gives the same initial item table as the flat PCA-initialized model. For TIGER and MaskGR, we apply the same fitted PCA projection and coordinatewise standardization to the RQ centroids and use the resulting vectors to initialize each level's codeword embeddings. The de-duplication level has no centroid and remains randomly initialized.

Baseline implementations. We use MaskGR's official implementation[3]. Because its default hyperparameters differ from those reported in the paper, we train both configurations and select between them using the validation criterion from Section 4.1. For TIGER, we use the implementation[4] provided in GRID (Ju et al., 2025). We supply each with our own SIDs and interaction splits and otherwise keep the published training code untouched. We could not find an official implementation for FlowRec, so we use our own implementation following Li et al. (2026b) for the flow objective. For fair comparison, we use the same encoder, velocity field and decoder as CAP-Mix. We fix m=3 for CAP-Mix and FlowRec and do not tune it. TIGER and MaskGR use their released implementations' maximum input length of 120 tokens. Because an item occupies one token per SID level, including the de-duplication level, the corresponding maximum number of history items varies with the validation-selected SID depth.

Generative baselines across the three scenarios. TIGER and MaskGR consume items as tokens, so a scenario is a choice of tokenizer and of how its token table is initialized. OFFLINE SID uses the RQ-GMM tuple. FROM SCRATCH uses an atomic tokenizer, one level assigning item i the token i, so the token table is $|\mathcal{I}|$ rows learned from interactions alone. PCA-TEXT uses the RQ-GMM tuple with the PCA-initialized codeword rows described above.

[3] https://github.com/snap-research/MaskGR

[4] https://github.com/snap-research/GRID

Table 5: **Inference and training cost** at matched input and output: top-20 lists from 3 history items, with TIGER and MaskGR using beam search with width 20. *Hot params* are those a served model must hold, split as *table* + *backbone* (every backbone is constant across datasets). Subscripts give each baseline relative to CAP-Mix and are computed from unrounded measurements. Throughput is measured on one H200 with a batch size of 128 in FP32. *Train GPU-h* is the estimated cost of training the reported model, computed by dividing the optimizer-step count at the selected checkpoint by a separately measured training rate.

	ML-1M	**Steam**	**Beauty**	**Indus.**	**Yelp**
$\lvert\mathcal{I}\rvert$	3.4k	4.1k	12k	25k	111k
Hot params (M) ↓: table + backbone					
CAP-Mix	0.4+0.9	0.5+0.9	1.5+0.9	3.3+0.9	14.2+0.9
TIGER	0.9+12.9 $_{10\times}$	0.3+12.9 $_{9.1\times}$	3.1+12.9 $_{6.5\times}$	0.3+12.9 $_{3.1\times}$	0.5+12.9 $_{0.9\times}$
MaskGR	0.4+6.8 $_{5.4\times}$	0.3+6.8 $_{5.0\times}$	0.2+6.8 $_{2.8\times}$	0.3+6.8 $_{1.7\times}$	14.2+6.8 $_{1.4\times}$
MFLOP / query ↓					
CAP-Mix	6.4	6.6	8.6	12.2	34.0
TIGER	195 $_{30\times}$	1,742 $_{265\times}$	199 $_{23\times}$	1,742 $_{143\times}$	1,747 $_{51\times}$
MaskGR	43.6 $_{6.8\times}$	17,435 $_{2,654\times}$	17,037 $_{1,976\times}$	10,399 $_{855\times}$	126 $_{3.7\times}$
Queries/s ↑					
CAP-Mix	26,934	28,752	29,240	27,864	7,688
TIGER	5,568 $_{1/4.8}$	4,054 $_{1/7.1}$	7,242 $_{1/4.0}$	4,121 $_{1/6.8}$	4,028 $_{1/1.9}$
MaskGR	5,634 $_{1/4.8}$	182 $_{1/158}$	182 $_{1/161}$	242 $_{1/115}$	5,656 $_{1/1.4}$
Train GPU-h ↓					
CAP-Mix	1.46	0.21	0.06	0.08	1.99
TIGER	2.01 $_{1.4\times}$	0.20 $_{1.0\times}$	0.53 $_{8.9\times}$	1.08 $_{13\times}$	2.21 $_{1.1\times}$
MaskGR	2.06 $_{1.4\times}$	10 $_{49\times}$	17 $_{286\times}$	47 $_{560\times}$	7.59 $_{3.8\times}$
Train examples/s ↑					
CAP-Mix	40,623	53,331	37,149	31,406	17,385
TIGER	1,616 $_{1/25}$	2,796 $_{1/19}$	4,541 $_{1/8.2}$	4,551 $_{1/6.9}$	1,311 $_{1/13}$
MaskGR	5,525 $_{1/7.4}$	5,547 $_{1/9.6}$	1,344 $_{1/28}$	5,552 $_{1/5.7}$	5,995 $_{1/2.9}$

Measuring inference cost. All five dataset columns of Table 5 describe the same query: every model is given the m=3 most recent items, which for a token-based baseline means an input sequence of length $3n_h$. *Hot params* are the parameters a served model must hold, determined by which modules actually fire during an inference batch rather than by the checkpoint size: CAP-Mix materializes its item table $\mathbf{x}_i$ once into $[|\mathcal{I}|, D]$ and discards the codebooks, assignment head and item features, so it serves at the size of a flat table, where a generative decoder has nothing to fold away. *MFLOP/query* is counted with PyTorch's flop formulas. *Queries/s* is wall-clock throughput on one H200 with a batch size of 128 in FP32, measured the same way for every model. We run three independent processes per cell; each contributes one steady-state estimate after warm-up, and we report the median of the three. Within a process we time the decode alone, excluding import and checkpoint load, over 100 batches for the baselines and a 12 s window for CAP-Mix.

Measuring training cost. Table 5 also reports training cost. Optimizer *steps* come from the checkpoint that produced the reported result; the *rate* is measured separately from the elapsed-time difference between training-only runs stopped at N_1 and N_2 steps, excluding validation. GPU-hours are estimated by dividing the selected checkpoint's optimizer-step count by this rate. All cells use one H200 with identical dataloader-worker and CPU allocations, and we report *examples/s* to account for differences in effective batch size. GPU-hours are implementation-bound: for TIGER and MaskGR, we use the unmodified implementations.

C BALANCED RQ-GMM

Why the balanced variant. RQ-GMM (Tong et al., 2026) fits each residual level by soft EM (Dempster et al., 1977), which places no constraint on how item mass is distributed over the K components, and in some of our experiments a few components absorb most of the catalog. Because an item's identifier is the L-tuple $(c_i^{(1)}, \ldots, c_i^{(L)})$, an unbalanced level shrinks the effective

identifier space, so more items collide on the same tuple and the clustering the SID is meant to provide degrades. We therefore replace the soft EM by a hard-assignment classification EM, which imposes the constraint where the decision is actually made. The soft responsibility then plays no part in the fit at all: it survives only as a read-time quantity, the prior in Equation (23).

The quantizer. Let $\{\mathbf{q}_i\}_{i=1}^N$ be the item embeddings and $\mathbf{e}_i^{(1)} = \mathbf{q}_i$. Level s models the residual left by the levels above it with a K-component diagonal Gaussian mixture whose means form that level's codebook, and passes on $\mathbf{e}_i^{(s+1)} = \mathbf{e}_i^{(s)} - \boldsymbol{\mu}_{c_i^{(s)}}^{(s)}$. Writing $\omega_k^{(s)}$ for the mixing coefficients, the cost of placing a residual $\mathbf{e}$ in component k is its negative log-likelihood,

$$\psi_k^{(s)}(\mathbf{e}) = -\log \omega_k^{(s)} + \tfrac{1}{2}\sum_{j=1}^{d}\Big[\frac{(e_j - \mu_{k,j}^{(s)})^2}{\sigma_{k,j}^{(s)2}} + \log \sigma_{k,j}^{(s)2}\Big], \tag{20}$$

up to a constant that does not depend on k. Each level is fitted by a *classification* EM (Celeux & Govaert, 1992), which alternates between a hard assignment and a maximum-likelihood update, and it is the assignment step that carries the size constraint:

$$\min_{z\in\{0,1\}^{N\times K}} \sum_{i=1}^{N}\sum_{k=1}^{K} z_{ik}\,\psi_k^{(s)}(\mathbf{e}_i^{(s)}) \quad \text{s.t.} \quad \sum_k z_{ik} = 1 \ \ \forall i, \qquad n_{\min} \le \sum_i z_{ik} \le n_{\max} \ \ \forall k, \tag{21}$$

with $n_{\min} = \max(1, \lfloor N/K \rfloor - 1)$ and $n_{\max} = \lfloor N/K \rfloor + 1$. Without the size bounds, Equation (21) separates per item and returns the maximum a posteriori (MAP) code; with them it is a transportation problem, which we solve exactly as a minimum-cost flow. This extends the constrained k-means of Bradley et al. (2000), who impose a minimum cluster size and solve the resulting assignment step as a minimum-cost flow. We add the upper bound $n_{\max}$ and use the mixture cost Equation (20) in place of squared Euclidean distance, so the assignment remains an exact minimum-cost flow while the cost stays a likelihood.

The M-step is the maximum-likelihood update of a diagonal mixture under the hard partition $\mathcal{C}_k = \{i : z_{ik} = 1\}$,

$$\boldsymbol{\mu}_k^{(s)} = \frac{1}{|\mathcal{C}_k|}\sum_{i\in\mathcal{C}_k}\mathbf{e}_i^{(s)}, \quad \boldsymbol{\sigma}_k^{(s)2} = \frac{1}{|\mathcal{C}_k|}\sum_{i\in\mathcal{C}_k}\big(\mathbf{e}_i^{(s)} - \boldsymbol{\mu}_k^{(s)}\big)^2 + \varepsilon, \quad \omega_k^{(s)} = \frac{|\mathcal{C}_k|}{N}, \tag{22}$$

with a covariance ridge $\varepsilon = 10^{-6}$ added to every variance so Equation (20) does not send the cost to infinity on coordinates without spread. Each level is initialized with one short run of unconstrained k-means (k-means++ seeding (Arthur & Vassilvitskii, 2007), at most 50 Lloyd steps (Lloyd, 1982)) and iterated until the assigned cost $\sum_i \psi_{c_i^{(s)}}^{(s)}(\mathbf{e}_i^{(s)})$ changes by less than 10^{-4} relatively or fewer than 0.1% of items are reassigned, with a cap of 200 iterations. After reaching either stopping criterion or the iteration limit, we fix the resulting assignments, update the residuals, and fit the next level. Balance therefore holds by construction at every level: no code group differs from $\lfloor N/K \rfloor$ by more than one, whatever the scale of the embedding. A final de-duplication digit, ranking the items inside each colliding L-tuple, makes the full identifier unique.

Use as a prior. The hard RQ-SID baseline consumes the codes $c_i^{(s)}$ directly. CAP-Mix instead consumes the level-s responsibility, the posterior over the K components given the residual,

$$\gamma_i^{(s)} = \mathrm{softmax}_k\big(-\psi_k^{(s)}(\mathbf{e}_i^{(s)})\big), \qquad \rho_i^{(s)} = \mathrm{softmax}_k\big(-\psi_k^{(s)}(\mathbf{e}_i^{(s)})/\tau_\rho\big), \tag{23}$$

where $\rho_i^{(s)}$ is the tempered version Equation (8) actually reads. Since ψ sums over the d embedding dimensions, its spread across components grows with d, so τ_ρ also scales with d. We use τ_ρ=400 at d=2560. The residual path in Equation (23) is the *greedy* one, $\mathbf{e}_i^{(s+1)} = \mathbf{e}_i^{(s)} - \boldsymbol{\mu}_{k^*}^{(s)}$ with $k^* = \arg\max_k \gamma_i^{(s)}[k]$. Thus, the mode of $\gamma_i^{(s)}$ is not always the stored code $c_i^{(s)}$, although they agree for most items. For example, on Beauty, they coincide for 92–94% of items at each level. Feeding the responsibility rather than the code is what lets the tokenizer prior carry its uncertainty.

D THE TWO PRIORS

CAP-Mix injects external knowledge in two places: a *tokenizer* prior on which codes are assigned to an item (Equation (8)) and a *co-visitation* prior on which item to retrieve (Equation (12)). Both follow the same rule, which this appendix states once, justifies, and then instantiates in both settings.

The rule. Let $q_\theta \propto \exp(\ell)$ be a softmax over a finite set with learned logits ℓ, and let π be a prior belief over that same set. Multiplying the two and renormalizing,

$$p \;\propto\; \underbrace{\exp(\ell)}_{\text{evidence}} \cdot \underbrace{\pi^{\,w}}_{\text{prior}}, \qquad \text{equivalently} \qquad \log p = \ell + w \log \pi + \text{const}, \tag{24}$$

so *adding a scaled log-prior to the logits is equivalent to multiplying the two beliefs*, with w weighting the prior. This is log-linear pooling. The trust $w \geq 0$ interpolates from discarding the prior (w=0) to collapsing p onto $\arg\max \pi$ ($w \to \infty$).

Why not mix probabilities. What separates Equation (24) from the obvious alternative $p = (1-\beta)q_\theta + \beta\pi$ is that any finite w leaves the hypothesis class unchanged. That is, provided the prior has full support, Equation (8) can be overridden by evidence: the learned logits can add an arbitrary vector, so every interior point of Δ stays reachable. The persistent prior therefore biases the parameterization and optimization without restricting the reachable distributions. To ensure full support, we add ϵ_ρ inside the log of Equation (8) and mix the graph prior with the uniform distribution in Equation (25). A prior that is exactly zero on an outcome removes it at any $w > 0$. In contrast, the alternative linear mixing at a fixed $\beta > 0$ restricts p, since $p_k \geq \beta\pi_k$ no matter the learned evidence.

Instance 1: the tokenizer prior on codes. At depth t, the learned evidence for chain s is $\ell = \phi_{s,t}(\mathbf{f}_i)$. The tokenizer supplies the same prior to every chain at that depth: the responsibility that the fitted quantizer places on each of its K Gaussians for item i, denoted $\gamma_{i,k}^{(t)}$ (Appendix C). This is not a hand-built pseudo-count but an exact posterior under a fitted GMM generative model. It is read along a greedy residual path rather than the balanced assignment the quantizer was fitted with, so its mode need not be the stored code.

We use the tempered responsibility $\rho_i^{(t)}$ defined in Equation (23) as the prior in Equation (8), with α_t controlling its influence.

Instance 2: the co-visitation prior on items. Here the set is the catalog $\mathcal{I}$, the evidence is the inner product $\ell = \mathbf{u}^\top \mathbf{x}_i$ against the shared item table, and the prior is $\pi = \tilde{g}(i \mid H)$ with trust w, yielding Equation (12). Equation (12) thus makes explicit a prior that a plain dot-product softmax leaves implicit. We build $\tilde{g}$ once from the training sequences in three steps:

$$P = D^{-1}C, \quad D = \operatorname{diag}(C\mathbf{1}); \qquad \hat{g}_H = \frac{q_H\, P^{K_s}}{\|q_H\, P^{K_s}\|_1}; \qquad \tilde{g} = (1-\beta)\,\tfrac{1}{|\mathcal{I}|}\mathbf{1} + \beta\,\hat{g}_H. \tag{25}$$

First, C_{ij} counts how often items i and j occur in the same training sequence, and row-normalizing turns those counts into conditional frequencies: P_{ij} estimates how likely j is to appear given that i does, so every row is a distribution over the catalog and P is the transition kernel of a random walk on the item graph. Second, we summarize the history by a *recency-weighted* distribution q_H, which puts mass only on items with which the user interacted, weighted according to recency: the item at age a (most recent a=0) gets weight $\propto \eta^{\,a}$, repeats accumulate, and the result is L_1-normalized. Applying the kernel K_s times lets that mass flow outward along co-occurrence edges as a K_s-step random walk, so $\hat{g}_H[j]$ is large when j co-occurs strongly with what the user recently touched. Third, a raw $\hat{g}_H$ can place zero mass on an unseen (context, item) pair, which would veto that item outright, however strong the evidence. Thus, to ensure full support, we temper it toward the uniform distribution. For $0 \leq \beta < 1$, this keeps $\log \tilde{g}$ finite; β=0 ignores the graph, whereas β=1 trusts it fully but can contain zero-probability outcomes.

Only w and β are tuned; all other settings use fixed defaults that are identical everywhere and never searched: whole-sequence symmetric counts with no self-loops, η=0.5, K_s=1 (so only direct neighbours receive mass), and the same truncated history the encoder reads, which at m=3 gives $q_H = (0.571,\, 0.286,\, 0.143)$ from newest to oldest. The same log-prior is added to the training logits of both cross-entropies, so the model learns to complement the prior.

E GRADIENT COUPLING INDUCED BY PROTOTYPE SHARING

Let $V = |\mathcal{I}|$, and stack the item embeddings and item-specific vectors into $\mathbf{X}, R \in \mathbb{R}^{V \times D}$. For chain s and depth t, let $A^{(s,t)} \in \mathbb{R}^{V \times K}$ and $C^{(s,t)} \in \mathbb{R}^{K \times D}$. To isolate the direct effect of prototype sharing, hold the assignments fixed and consider a loss component $\mathcal{L}(\mathbf{X})$ whose dependence on R and $C^{(s,t)}$ is only through $\mathbf{X}$. Let $G = \nabla_{\mathbf{X}}\mathcal{L}$.

$$\mathbf{X} = R + \frac{\gamma}{S}\sum_{s,t} A^{(s,t)}C^{(s,t)}, \qquad \nabla_{C^{(s,t)}}\mathcal{L} = \frac{\gamma}{S}A^{(s,t)\top}G, \qquad \nabla_R\mathcal{L} = G. \tag{26}$$

Consider one gradient-descent step with learning rate η, and let $\Delta\mathbf{X}$ denote the resulting change in the item table. Equation 26 gives

$$\Delta R = -\eta G, \qquad \Delta C^{(s,t)} = -\eta\frac{\gamma}{S}A^{(s,t)\top}G. \tag{27}$$

Because the assignments are fixed, substituting these parameter updates into the composed table gives

$$\begin{aligned}\Delta\mathbf{X} &= \Delta R + \frac{\gamma}{S}\sum_{s,t} A^{(s,t)}\Delta C^{(s,t)} \\ &= -\eta G - \eta\frac{\gamma^2}{S^2}\sum_{s,t} A^{(s,t)}A^{(s,t)\top}G \\ &= -\eta(I_V + \mathcal{K})G, \qquad \mathcal{K} = \frac{\gamma^2}{S^2}\sum_{s,t} A^{(s,t)}A^{(s,t)\top},\end{aligned} \tag{28}$$

where $I_V \in \mathbb{R}^{V \times V}$ is the identity matrix. Consequently,

$$\mathcal{K}_{ij} = \frac{\gamma^2}{S^2}\sum_{s,t}\left\langle A_i^{(s,t)}, A_j^{(s,t)}\right\rangle, \tag{29}$$

which couples item updates in proportion to assignment overlap, while the identity term I_V preserves a direct item-specific update. This analysis isolates the direct representation path with fixed assignments, while the complete model additionally learns the assignments and contains losses that depend explicitly on the prototypes.

F DIAGNOSTIC RESULTS

Table 6 separates the semantic information in the RQ-SID assignments from the parameter sharing induced by the codebook. The random control preserves the table structure, depth, codebook size, and codeword vectors while randomly permuting the item-to-code assignments.

From scratch, the hard RQ-SID table improves HR@5 ↑ over the flat table on all three datasets, whereas matched random assignments yield smaller gains or a loss. This isolates useful semantic structure in the pretrained assignments beyond parameter sharing alone. With PCA-TEXT initialization, freezing the content embeddings causes losses ranging from 14.15% to 20.21%. On the two Amazon datasets, hard semantic assignments also underperform both the trainable flat table and the random-assignment control, suggesting that a fixed semantic partition can conflict with the recommendation objective once the continuous content representation is available. This motivates using the tokenizer to initialize differentiable assignments that can be optimized jointly with the recommender. On Steam, PCA initialization itself reduces performance, while hard and random assignments produce similar gains below 1%; we hypothesize that the content embedding is less informative for next-item prediction on this dataset, causing its semantic partition to behave similarly to random sharing.

Removing the co-visitation prior. Table 7 repeats the scenario study with the co-visitation prior disabled for every variant. On Beauty and Industrial, CAP-Mix remains above the corresponding baseline in all three regimes. On Steam, it is essentially flat relative to the baseline in FROM SCRATCH and PCA-TEXT, but retains a $+6.91\%$HR@5 ↑ advantage over the hard table in OFFLINE SID. Thus, these results show that the gradient-pooling benefit from our prototype mixtures is complementary to the retrieval prior.

Table 6: Item-table variants for a continuous flow recommender. The flat table of each block is absolute test HR@5 ↑; every other cell is its % change relative to it. A hard SID beats random codes from scratch, but with the full content embedding the ordering reverses and, on the sparse catalogs, both fall below the trainable flat table.

scenario	item table	Beauty	Industrial	Steam
FROM SCRATCH	FLAT	5.77	2.75	8.29
	+ HARD RQ-SID	+0.52%	+9.82%	+0.36%
	+ RANDOM SID	−1.73%	+0.73%	+0.12%
PCA-TEXT	FLAT, trainable	6.69	3.81	8.13
	FROZEN	−18.24%	−20.21%	−14.15%
	+ HARD RQ-SID	−3.89%	−7.61%	+0.74%
	+ RANDOM SID	+0.30%	−1.84%	+0.86%

Table 7: **Controlled results without the co-visitation prior.** All values are test HR@5 ↑ (×100). Each variant's cell gives its absolute score over its relative change from the from-scratch flat table. Each *vs.* column compares CAP-Mix with that regime's baseline.

	FROM SCRATCH			OFFLINE SID			PCA-TEXT			
dataset	FLAT	CAP-Mix	*vs.*	HARD	CAP-Mix	*vs.*	FLAT	HARD	CAP-Mix	*vs.*
Beauty	5.61 *ref.*	5.67 +1.07%	+1.07%	5.76 +2.67%	5.89 +4.99%	+2.26%	6.27 +11.76%	6.27 +11.76%	6.50 +15.86%	+3.67%
Industrial	2.64 *ref.*	2.81 +6.44%	+6.44%	2.85 +7.95%	3.21 +21.59%	+12.63%	3.62 +37.12%	3.51 +32.95%	3.68 +39.39%	+1.66%
Steam	8.28 *ref.*	8.26 −0.24%	−0.24%	7.96 −3.86%	8.51 +2.78%	+6.91%	8.10 −2.17%	8.21 −0.85%	8.14 −1.69%	+0.49%

Frequency stratification. We group test users by the training frequency of their target item, without splitting users whose targets have the same frequency. The resulting approximately equal-mass groups contain 22,363 Beauty users (rare: at most 6 interactions; medium: 7–22; frequent: more than 22) and 51,322 Industrial users (rare: at most 7; medium: 8–28; frequent: more than 28). Table 8 reports the relative gain within each group.

G FULL RESULTS

Table 9 summarizes all evaluation cutoffs for the validation-selected configuration of each model and dataset. Tables 10–15 then give the complete per-dataset results across all model families, initializations, and metrics.

Table 8: **Relative gains by target-item training frequency.** Test users are grouped by their target item's frequency in the training set. Entries are the relative improvement of CAP-Mix over the indicated baseline. Relative gains use unrounded scores and may differ slightly from calculations using displayed values.

scenario	baseline	dataset and metric	rare	medium	frequent	all	larger end
FROM SCRATCH	FLAT	Beauty HR@5 ↑	+0.2%	−1.8%	+4.1%	+2.1%	frequent
		Beauty NDCG@10 ↑	−3.3%	−1.5%	+2.9%	+0.9%	frequent
		Industrial HR@5 ↑	+5.9%	+4.5%	+4.4%	+4.5%	rare
		Industrial NDCG@10 ↑	+4.6%	+1.8%	+5.1%	+4.5%	frequent
OFFLINE SID	HARD	Beauty HR@5 ↑	+8.5%	+8.8%	+8.9%	+8.8%	frequent
		Beauty NDCG@10 ↑	+10.9%	+6.6%	+4.8%	+6.1%	rare
		Industrial HR@5 ↑	+49.7%	+26.6%	+10.9%	+14.7%	rare
		Industrial NDCG@10 ↑	+56.0%	+32.3%	+17.1%	+20.8%	rare
PCA-TEXT	FLAT	Beauty HR@5 ↑	+5.2%	+3.6%	−0.3%	+1.4%	rare
		Beauty NDCG@10 ↑	+4.2%	+3.1%	−0.4%	+1.1%	rare
		Industrial HR@5 ↑	+7.1%	+2.6%	−1.4%	−0.3%	rare
		Industrial NDCG@10 ↑	+9.0%	+0.8%	−2.5%	−1.4%	rare
PCA-TEXT	HARD	Beauty HR@5 ↑	+10.7%	+3.0%	+5.4%	+5.5%	rare
		Beauty NDCG@10 ↑	+5.3%	+3.2%	+5.1%	+4.6%	rare
		Industrial HR@5 ↑	+15.3%	+8.7%	+6.9%	+7.6%	rare
		Industrial NDCG@10 ↑	+18.8%	+9.7%	+8.3%	+9.0%	rare

Table 9: **Full cross-dataset results**, HR@k ↑ and NDCG@k ↑ (mean±std, ×100). Each model and dataset uses the configuration selected by the validation geometric mean of HR@5 ↑ and NDCG@5 ↑; the same run supplies every cutoff. Superscripts denote ^{s} scratch, ^{i} offline SID, and ^{p} PCA-text. **Bold** = best, underline = 2nd per column, and Δ compares CAP-Mix with the strongest baseline.

	Beauty						**Indus.**					
	HR@5	N@5	HR@10	N@10	HR@20	N@20	HR@5	N@5	HR@10	N@10	HR@20	N@20
FlowRec	6.27^{p} ±0.18	4.03 ±0.03	9.52 ±0.22	5.07 ±0.05	13.52 ±0.41	6.08 ±0.10	3.62^{p} ±0.02	2.04 ±0.03	5.54 ±0.04	2.66 ±0.04	8.03 ±0.04	3.28 ±0.04
TIGER	3.51^{s} ±0.07	2.54 ±0.05	4.92 ±0.09	2.99 ±0.05	6.90 ±0.21	3.49 ±0.08	2.05^{p} ±0.03	1.34 ±0.02	3.25 ±0.03	1.73 ±0.02	4.61 ±0.01	2.08 ±0.01
MaskGR	5.06^{i} ±0.13	3.28 ±0.12	8.11 ±0.15	4.27 ±0.12	12.10 ±0.16	5.29 ±0.09	3.46^{p} ±0.03	2.24 ±0.03	5.25 ±0.06	2.82 ±0.02	7.46 ±0.07	3.38 ±0.01
CAP-Mix	**6.79**^{p} ±0.06	**4.73** ±0.04	**9.75** ±0.06	**5.69** ±0.04	**13.53** ±0.17	**6.64** ±0.07	**3.79**^{p} ±0.01	**2.34** ±0.03	**5.77** ±0.02	**2.98** ±0.02	**8.36** ±0.08	**3.63** ±0.04
Δ (%)	+8.3%	+17.4%	+2.4%	+12.2%	+0.1%	+9.2%	+4.7%	+4.5%	+4.2%	+5.7%	+4.1%	+7.4%
	Steam						**ML-1M**					
	HR@5	N@5	HR@10	N@10	HR@20	N@20	HR@5	N@5	HR@10	N@10	HR@20	N@20
FlowRec	8.28^{s} ±0.33	5.37 ±0.30	13.54 ±0.31	7.06 ±0.29	20.61 ±0.35	8.84 ±0.30	15.36^{p} ±0.20	10.14 ±0.12	22.94 ±0.04	12.58 ±0.05	32.68 ±0.14	15.03 ±0.09
TIGER	8.43^{i} ±0.10	5.01 ±0.03	13.28 ±0.63	6.57 ±0.19	20.46 ±1.07	8.36 ±0.24	12.59^{s} ±0.40	8.47 ±0.28	19.03 ±0.37	10.54 ±0.28	27.20 ±0.68	12.61 ±0.36
MaskGR	8.64^{p} ±0.00	**5.57** ±0.01	**14.43** ±0.05	**7.42** ±0.02	**20.97** ±0.23	**9.08** ±0.07	15.64^{s} ±0.40	10.23 ±0.28	23.61 ±0.29	12.79 ±0.26	33.93 ±0.17	15.39 ±0.21
CAP-Mix	**8.69**^{i} ±0.29	**5.57** ±0.19	13.94 ±0.32	7.25 ±0.20	20.82 ±0.42	8.99 ±0.23	**17.25**^{p} ±0.13	**11.71** ±0.11	**24.71** ±0.38	**14.11** ±0.17	**33.98** ±0.34	**16.44** ±0.15
Δ (%)	+0.6%	0.0%	−3.4%	−2.3%	−0.7%	−1.0%	+10.3%	+14.5%	+4.7%	+10.3%	+0.1%	+6.8%
	Yelp						**Office**					
	HR@5	N@5	HR@10	N@10	HR@20	N@20	HR@5	N@5	HR@10	N@10	HR@20	N@20
FlowRec	2.98^{p} ±0.05	1.86 ±0.03	5.10 ±0.05	2.54 ±0.03	8.34 ±0.06	3.35 ±0.04	2.90^{p} ±0.02	1.67 ±0.02	**4.41** ±0.04	2.15 ±0.02	**6.23** ±0.02	2.61 ±0.02
TIGER	1.29^{p} ±0.07	0.80 ±0.04	2.09 ±0.09	1.06 ±0.04	3.03 ±0.10	1.30 ±0.04	1.52^{p} ±0.01	0.98 ±0.01	2.33 ±0.03	1.25 ±0.00	3.34 ±0.08	1.50 ±0.02
MaskGR	2.99^{s} ±0.04	1.87 ±0.03	5.10 ±0.07	2.54 ±0.03	8.24 ±0.10	3.33 ±0.04	2.29^{s} ±0.02	1.53 ±0.01	3.40 ±0.03	1.89 ±0.02	4.88 ±0.03	2.26 ±0.01
CAP-Mix	**3.18**^{p} ±0.02	**2.03** ±0.02	**5.25** ±0.04	**2.70** ±0.03	**8.38** ±0.05	**3.48** ±0.03	**3.22**^{p} ±0.01	**2.28** ±0.01	**4.41** ±0.01	**2.66** ±0.01	5.84 ±0.03	**3.02** ±0.01
Δ (%)	+6.4%	+8.6%	+2.9%	+6.3%	+0.5%	+3.9%	+11.0%	+36.5%	0.0%	+23.7%	−6.3%	+15.7%

Table 10: **Beauty — full results.** All families × initializations. **Bold** = best, underline = 2nd per column within an initialization block; CAP-Mix shows Δ% vs. the best baseline. FlowRec has no OFFLINE SID variant, since a flat table has no codes to consume, so the row marked † repeats its FROM SCRATCH numbers.

beauty	model	HR@5	HR@10	HR@20	N@5	N@10	N@20
scratch	FlowRec	5.61±0.02	8.01±0.05	10.90±0.20	3.86±0.02	4.63±0.03	5.36±0.06
	TIGER	3.51±0.07	4.92±0.09	6.90±0.21	2.54±0.05	2.99±0.05	3.49±0.08
	MaskGR	4.38±0.16	6.48±0.19	9.33±0.09	2.99±0.14	3.66±0.14	4.38±0.12
	CAP-Mix	**5.89**±0.04 +4.99%	**8.28**±0.05 +3.37%	**11.30**±0.01 +3.67%	**4.21**±0.01 +9.07%	**4.98**±0.01 +7.56%	**5.75**±0.01 +7.28%
SID	FlowRec†	5.61±0.02	8.01±0.05	10.90±0.20	3.86±0.02	4.63±0.03	5.36±0.06
	TIGER	3.48±0.14	5.33±0.15	8.00±0.13	2.30±0.14	2.89±0.14	3.56±0.14
	MaskGR	5.06±0.13	8.11±0.15	**12.10**±0.16	3.28±0.12	4.27±0.12	5.29±0.09
	CAP-Mix	**6.32**±0.10 +12.66%	**8.72**±0.08 +7.52%	11.96±0.10 −1.16%	**4.51**±0.05 +16.84%	**5.28**±0.03 +14.04%	**6.10**±0.00 +13.81%
PCA	FlowRec	6.27±0.18	9.52±0.22	13.52±0.41	4.03±0.03	5.07±0.05	6.08±0.10
	TIGER	3.39±0.09	5.45±0.19	8.30±0.18	2.24±0.06	2.90±0.09	3.62±0.08
	MaskGR	5.05±0.13	7.93±0.05	11.97±0.13	3.26±0.13	4.19±0.10	5.22±0.06
	CAP-Mix	**6.79**±0.06 +8.29%	**9.75**±0.06 +2.42%	**13.53**±0.17 +0.07%	**4.73**±0.04 +17.37%	**5.69**±0.04 +12.23%	**6.64**±0.07 +9.21%

Table 11: **Industrial & Scientific — full results.** Conventions as in Table 10.

industrial	model	HR@5	HR@10	HR@20	N@5	N@10	N@20
scratch	FlowRec	2.64±0.00	4.04±0.03	**5.91**±0.09	1.49±0.01	1.94±0.01	2.41±0.02
	TIGER	1.61±0.05	2.60±0.10	4.01±0.13	1.05±0.03	1.37±0.05	1.72±0.06
	MaskGR	2.36±0.02	3.63±0.02	5.47±0.02	1.58±0.01	1.99±0.01	2.45±0.01
	CAP-Mix	**2.88**±0.01 +9.09%	**4.15**±0.07 +2.72%	5.82±0.05 −1.52%	**2.00**±0.00 +26.58%	**2.41**±0.02 +21.11%	**2.83**±0.02 +15.51%
SID	FlowRec†	2.64±0.00	4.04±0.03	5.91±0.09	1.49±0.01	1.94±0.01	2.41±0.02
	TIGER	2.01±0.05	3.20±0.03	4.62±0.06	1.32±0.03	1.70±0.02	2.06±0.02
	MaskGR	3.30±0.05	**5.19**±0.07	7.41±0.07	2.14±0.02	2.75±0.03	3.31±0.03
	CAP-Mix	**3.47**±0.02 +5.15%	5.17±0.00 −0.39%	**7.50**±0.04 +1.21%	**2.29**±0.02 +7.01%	**2.84**±0.02 +3.27%	**3.43**±0.02 +3.63%
PCA	FlowRec	3.62±0.02	5.54±0.04	8.03±0.04	2.04±0.03	2.66±0.04	3.28±0.04
	TIGER	2.05±0.03	3.25±0.03	4.61±0.01	1.34±0.02	1.73±0.02	2.08±0.01
	MaskGR	3.46±0.03	5.25±0.06	7.46±0.07	2.24±0.03	2.82±0.02	3.38±0.01
	CAP-Mix	**3.79**±0.01 +4.70%	**5.77**±0.02 +4.15%	**8.36**±0.08 +4.11%	**2.34**±0.03 +4.46%	**2.98**±0.02 +5.67%	**3.63**±0.04 +7.40%

Table 12: **Steam — full results.** Conventions as in Table 10.

steam	model	HR@5	HR@10	HR@20	N@5	N@10	N@20
scratch	FlowRec	8.28±0.33	13.54±0.31	20.61±0.35	5.37±0.30	7.06±0.29	8.84±0.30
	TIGER	7.94±0.17	13.18±0.24	**21.20**±1.59	4.90±0.07	6.59±0.07	8.58±0.38
	MaskGR	**8.52**±0.17	**14.21**±0.19	20.94±0.05	**5.58**±0.18	**7.40**±0.19	**9.11**±0.16
	CAP-Mix	8.35±0.31 −2.00%	14.15±0.28 −0.42%	21.09±0.20 −0.52%	5.16±0.17 −7.53%	7.02±0.16 −5.14%	8.77±0.14 −3.73%
SID	FlowRec†	8.28±0.33	13.54±0.31	20.61±0.35	5.37±0.30	7.06±0.29	8.84±0.30
	TIGER	8.43±0.10	13.28±0.63	20.46±1.07	5.01±0.03	6.57±0.19	8.36±0.24
	MaskGR	8.64±0.09	**14.43**±0.24	20.64±0.23	5.54±0.09	**7.40**±0.14	8.97±0.13
	CAP-Mix	**8.69**±0.29 +0.58%	13.94±0.32 −3.40%	**20.82**±0.42 +0.87%	**5.57**±0.19 +0.54%	7.25±0.20 −2.03%	**8.99**±0.23 +0.22%
PCA	FlowRec	8.10±0.24	13.45±0.11	20.70±0.22	5.18±0.23	6.88±0.19	8.72±0.22
	TIGER	8.22±0.31	13.58±0.33	18.65±0.63	5.00±0.07	6.72±0.07	8.00±0.17
	MaskGR	**8.64**±0.00	**14.43**±0.05	20.97±0.23	**5.57**±0.01	**7.42**±0.02	**9.08**±0.07
	CAP-Mix	8.30±0.20 −3.94%	14.28±0.02 −1.04%	**21.50**±0.06 +2.53%	5.07±0.09 −8.98%	6.99±0.03 −5.80%	8.81±0.04 −2.97%

Table 13: **ML-1M — full results.** Conventions as in Table 10.

ml-1m	model	HR@5	HR@10	HR@20	N@5	N@10	N@20
scratch	FlowRec	15.01±0.12	22.73±0.17	32.48±0.30	9.90±0.17	12.38±0.20	14.84±0.24
	TIGER	12.59±0.40	19.03±0.37	27.20±0.68	8.47±0.28	10.54±0.28	12.61±0.36
	MaskGR	15.64±0.40	23.61±0.29	33.93±0.17	10.23±0.28	12.79±0.26	15.39±0.21
	CAP-Mix	**17.48**±0.27 +11.76%	**24.80**±0.09 +5.04%	**34.01**±0.29 +0.24%	**11.74**±0.19 +14.76%	**14.10**±0.14 +10.24%	**16.41**±0.20 +6.63%
SID	FlowRec†	15.01±0.12	22.73±0.17	32.48±0.30	9.90±0.17	12.38±0.20	14.84±0.24
	TIGER	12.52±0.70	18.91±0.62	26.88±0.66	8.20±0.31	10.25±0.29	12.26±0.30
	MaskGR	13.04±1.22	20.09±1.31	29.45±1.75	8.43±0.72	10.70±0.74	13.05±0.85
	CAP-Mix	**16.61**±0.31 +10.66%	**24.25**±0.08 +6.69%	**33.70**±0.31 +3.76%	**11.26**±0.15 +13.74%	**13.73**±0.08 +10.90%	**16.11**±0.13 +8.56%
PCA	FlowRec	15.36±0.20	22.94±0.04	32.68±0.14	10.14±0.12	12.58±0.05	15.03±0.09
	TIGER	12.16±0.31	18.43±0.17	27.08±0.40	8.12±0.18	10.12±0.15	12.30±0.06
	MaskGR	12.58±0.55	19.81±0.56	29.10±0.39	8.27±0.43	10.60±0.42	12.95±0.39
	CAP-Mix	**17.25**±0.13 +12.30%	**24.71**±0.38 +7.72%	**33.98**±0.34 +3.98%	**11.71**±0.11 +15.48%	**14.11**±0.17 +12.16%	**16.44**±0.15 +9.38%

Table 14: **Yelp-2022 — full results.** Conventions as in Table 10.

yelp	model	HR@5	HR@10	HR@20	N@5	N@10	N@20
scratch	FlowRec	2.79±0.02	4.76±0.05	7.76±0.07	1.73±0.02	2.37±0.02	3.12±0.03
	TIGER	0.26±0.01	0.42±0.02	0.73±0.10	0.17±0.01	0.22±0.01	0.30±0.03
	MaskGR	2.99±0.04	5.10±0.07	8.24±0.10	1.87±0.03	2.54±0.03	3.33±0.04
	CAP-Mix	**3.07**±0.02 +2.68%	**5.14**±0.02 +0.78%	**8.26**±0.02 +0.24%	**1.93**±0.02 +3.21%	**2.60**±0.01 +2.36%	**3.38**±0.01 +1.50%
SID	FlowRec†	2.79±0.02	4.76±0.05	7.76±0.07	1.73±0.02	2.37±0.02	3.12±0.03
	TIGER	1.27±0.08	2.05±0.10	2.95±0.18	0.77±0.05	1.02±0.06	1.25±0.08
	MaskGR	2.63±0.01	4.37±0.05	6.79±0.07	1.65±0.01	2.20±0.02	2.81±0.03
	CAP-Mix	**3.13**±0.02 +12.19%	**5.18**±0.02 +8.82%	**8.32**±0.01 +7.22%	**1.99**±0.01 +15.03%	**2.64**±0.01 +11.39%	**3.43**±0.01 +9.94%
PCA	FlowRec	2.98±0.05	5.10±0.05	8.34±0.06	1.86±0.03	2.54±0.03	3.35±0.04
	TIGER	1.29±0.07	2.09±0.09	3.03±0.10	0.80±0.04	1.06±0.04	1.30±0.04
	MaskGR	2.53±0.02	4.26±0.04	6.60±0.04	1.59±0.01	2.14±0.01	2.73±0.02
	CAP-Mix	**3.18**±0.02 +6.71%	**5.25**±0.04 +2.94%	**8.38**±0.05 +0.48%	**2.03**±0.02 +9.14%	**2.70**±0.03 +6.30%	**3.48**±0.03 +3.88%

Table 15: **Amazon Office — full results.** Conventions as in Table 10.

office	model	HR@5	HR@10	HR@20	N@5	N@10	N@20
scratch	FlowRec	2.11±0.06	3.09±0.07	4.33±0.12	1.40±0.03	1.71±0.03	2.03±0.05
	TIGER	0.76±0.01	1.09±0.02	1.67±0.10	0.48±0.02	0.59±0.02	0.74±0.02
	MaskGR	2.29±0.02	3.40±0.03	**4.88**±0.03	1.53±0.01	1.89±0.02	2.26±0.01
	CAP-Mix	**2.72**±0.01 +18.78%	**3.50**±0.01 +2.94%	4.49±0.01 −7.99%	**2.00**±0.01 +30.72%	**2.25**±0.01 +19.05%	**2.50**±0.01 +10.62%
SID	FlowRec†	2.11±0.06	3.09±0.07	4.33±0.12	1.40±0.03	1.71±0.03	2.03±0.05
	TIGER	1.47±0.07	2.26±0.11	3.17±0.16	0.95±0.04	1.20±0.05	1.43±0.07
	MaskGR	2.39±0.01	3.56±0.04	4.88±0.05	1.59±0.02	1.96±0.02	2.30±0.03
	CAP-Mix	**2.95**±0.03 +23.43%	**4.13**±0.05 +16.01%	**5.65**±0.07 +15.78%	**2.07**±0.02 +30.19%	**2.45**±0.02 +25.00%	**2.84**±0.03 +23.48%
PCA	FlowRec	2.90±0.02	**4.41**±0.04	**6.23**±0.02	1.67±0.02	2.15±0.02	2.61±0.02
	TIGER	1.52±0.01	2.33±0.03	3.34±0.08	0.98±0.01	1.25±0.00	1.50±0.02
	MaskGR	2.16±0.02	3.24±0.01	4.48±0.01	1.42±0.01	1.77±0.01	2.08±0.01
	CAP-Mix	**3.22**±0.01 +11.03%	**4.41**±0.01 0.00%	5.84±0.03 −6.26%	**2.28**±0.01 +36.53%	**2.66**±0.01 +23.72%	**3.02**±0.01 +15.71%