LEARNING WHERE TO LOOK: GEOMETRIC MULTI-INTEREST RETRIEVAL FOR RECOMMENDER SYSTEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

The retrieval stage of recommender systems must efficiently identify a small set of items relevant to a given user from a large item corpus. Despite their differences, many existing retrieval methods share a common structure: *query-based* nearest-neighbor retrieval in an item embedding space. From this perspective, retrieval involves two fundamental questions: *where to place the query vectors* and *how far to search*. In this work, we develop a geometric framework for multi-interest retrieval. We show that cosine-similarity retrieval corresponds to searching *spherical caps* on the embedding sphere, and formulate efficient retrieval as a geometric covering problem. Our theoretical analysis shows that multiple retrieval regions can substantially reduce the retrieval cost for users with diverse interests. We further identify two sources of query placement error: *sampling error*, caused by estimating interests from limited historical interactions, and *drift error*, caused by changes in user interests over time. To address these errors, we learn multiple retrieval regions directly from user histories, with each region parameterized by a query center and an angular radius. The model is trained to cover future target items while minimizing the total retrieval region. Experiments on four recommendation datasets show that our approach consistently improves top-ranked retrieval quality over strong single-interest and multi-interest baselines, while maintaining compact retrieval regions.

1 INTRODUCTION

The retrieval stage of an industrial recommender system transforms a user's interaction history into the few hundred to few thousand items that the user is most likely to engage with, and passes them to the downstream ranking stage. Many modern retrieval sources, including user-to-item retrieval (Covington et al., 2016), item-to-item retrieval (Barkan & Koenigstein, 2016), creator retrieval (El-Kishky et al., 2022), and topic retrieval (Satuluri et al., 2020), can be viewed through a common lens: *query-based* nearest-neighbor retrieval over an item embedding space. Under this view, the key design choices are *where to place the queries* and *how far to search from each query*.

Both choices have a natural geometric interpretation. Dense retrieval scores items by cosine similarity, so the items returned by a query under a score threshold are exactly those lying within a certain angle of the query vector, forming a *spherical cap* (Figure 1a). To achieve high recall while minimizing the number of retrieved candidates, we want to cover the target items with as little area as possible. Retrieval can therefore be viewed as a geometric covering problem: among all collections of caps that contain the target items, find the one with the smallest total area.

A single query opens one cap. However, a user typically has multiple interests (Li et al., 2019; Cen et al., 2020; Li et al., 2026), causing the target items to form several groups. A *single* cap covering all groups must be wide enough to span them, whereas *multiple* caps can each focus on one group and achieve a much smaller total area. Theorem 1 formalizes this intuition: relative to a single cap, the cost of covering the target set with multiple caps decreases significantly with the embedding dimension. This provides a geometric justification for using multiple queries in retrieval.

Multiple caps are useful only when they are placed appropriately. A natural approach is to cluster the user's interaction history and use the resulting cluster centers as query vectors, as in existing multi-interest methods (Li et al., 2019; Cen et al., 2020; Pal et al., 2020). However, this approach suffers from two sources of error (Theorem 2). First, each interest may have only a few historical

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.

interactions, leading to *sampling error*. Second, user interests evolve over time, so the historical center may differ from the center of future interactions, resulting in *drift error*. In high-dimensional embedding spaces, even small errors in query placement can substantially increase retrieval cost.

We therefore propose to *learn* where to place the query vectors and how far to search from them. Given a user's interaction history, our Transformer-based model produces multiple retrieval regions, each specified by a center and an angular radius. Learning across users allows the model to infer an interest from more than the few interactions associated with that interest in a single user's history, reducing sampling error. More importantly, the model is trained against future interactions rather than the historical window, allowing it to anticipate interest drift. The training objective balances two goals: covering as many target items as possible while minimizing the total region probed by the retriever. In this way, the model jointly learns *where to look* and *how far to look*.

Experiments on four recommendation datasets demonstrate the effectiveness of our geometric formulation. Our method consistently delivers the strongest top-ranked retrieval performance, achieving the best HR@10 and NDCG@10 across all datasets, with especially large gains on the internal short-video dataset. Additional analyses show that the learned caps adapt their locations and sizes to different user interaction patterns, which substantially reduces the amount of the item space that needs to be probed.

2 A Geometric View of Retrieval

Each item i is represented by a content embedding $\boldsymbol{e}_i$ of dimension d, normalized to unit length, so that $\boldsymbol{e}_i$ lies on the unit sphere $\mathcal{S}^{d-1} = \{\boldsymbol{x} \in \mathbb{R}^d : \|\boldsymbol{x}\| = 1\}$. A user u has an interaction history sequence $\mathcal{H}_u$, consisting of items they have already engaged with, and a target set $\mathcal{T}_u$, consisting of items they engage with during a future window. For notational simplicity, we write $\mathcal{H}$ and $\mathcal{T}$ when referring to a generic user.

We focus on the retrieval stage of a recommendation pipeline, where the goal is to recover as much of $\mathcal{T}$ as possible from the observed history $\mathcal{H}$. For the theoretical analysis, we consider the idealized case of 100% recall: the retriever must return every item in $\mathcal{T}$ while minimizing the number of additional items. This corresponds to maximizing $P@R100$ in recommendation.[1] The downstream ranking stage then receives all target items while having fewer irrelevant items to score.

In dense retrieval, a single query vector $\boldsymbol{q} \in \mathcal{S}^{d-1}$ scores each item by the inner product $\boldsymbol{q}^\top \boldsymbol{e}_i$. For unit vectors, this score equals $\cos\angle(\boldsymbol{q}, \boldsymbol{e}_i)$, where $\angle(\cdot,\cdot) \in [0, \pi]$ denotes the angle between two vectors. Thus, ranking items by decreasing similarity is equivalent to ranking them by increasing angular distance from $\boldsymbol{q}$. A nearest-neighbor retriever therefore returns the items within some angular threshold θ of $\boldsymbol{q}$, namely the items in $\mathcal{C}(\boldsymbol{q}, \theta) = \{\boldsymbol{x} \in \mathcal{S}^{d-1} : \angle(\boldsymbol{x}, \boldsymbol{q}) \le \theta\}$, which we call the *spherical cap* centered at $\boldsymbol{q}$ with angular radius θ.

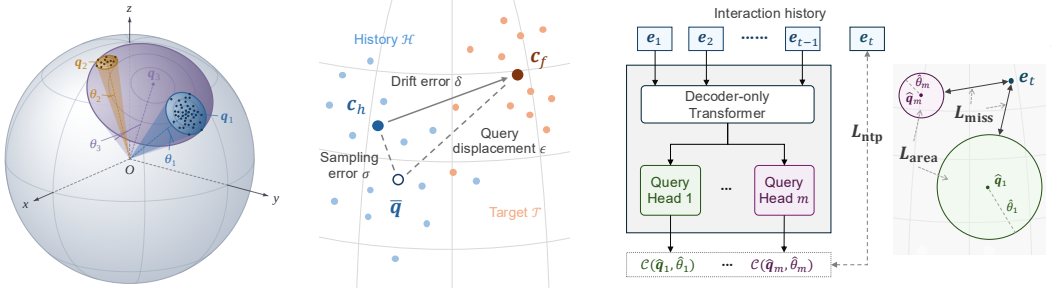
A single query vector can therefore retrieve one spherical cap. With multiple query vectors $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_m \in \mathcal{S}^{d-1}$, the retriever retrieves the union of m caps. We deduplicate the retrieved items and re-rank the resulting candidates before passing them to the downstream ranking stage.

2.1 One Query Vector Is Expensive

A *retriever's cost* is the minimal number of items it must retrieve to recover all of $\mathcal{T}$. A single query vector returns one cap, whose number of items is approximately proportional to its "area".[2] The area of a cap on the unit sphere depends only on its angular radius, so we write $A(\theta)$ for the fraction of the sphere occupied by a cap of angular radius θ, with $A(0) = 0$ and $A(\pi) = 1$. Formally, $A(\theta)$ is the uniform probability measure on $\mathcal{S}^{d-1}$ evaluated on such a cap. Under the density assumption above, retrieval cost is proportional to the area probed, so comparing retrievers reduces to comparing the areas of the caps they open.

[1] The 100% recall assumption is for analytical simplicity. The same framework can be extended to a target recall of $r < 1$, with slightly more complex expressions.

[2] A sufficient condition here is that items are spread uniformly over $\mathcal{S}^{d-1}$, which may not hold, as trained embeddings are known to concentrate in a narrow cone (Gao et al., 2019; Ethayarajh, 2019). However, global uniformity is stronger than necessary: it is sufficient that item density be reasonably stable across the regions where items actually occur. We empirically examine the local density of item embedding space in Section 4.2.



(a) One vs. multiple queries (b) Sampling and drift errors (c) Learning retrieval regions

Figure 1: A geometric view of multi-interest retrieval. A query retrieves items within an angular radius, forming a spherical cap whose area represents retrieval cost. **(a)** Multiple narrow caps can cover the same target items as a single wide cap with substantially less total area (Theorem 1). **(b)** Clustering the history introduces the sampling error σ and the drift error δ, displacing the query from the center of the future interest (Theorem 2). **(c)** Our method learns multiple retrieval regions from the user's history, with each region defined by a query center and an angular radius (Section 3).

We now ask how the number of query vectors affects the lowest cost a retriever can reach. Let $\text{cost}_m^{\mathcal{T}}$ denote the smallest total area of m caps that cover $\mathcal{T}$: $\text{cost}_m^{\mathcal{T}} = \min\left\{\sum_{j\le m} A(r_j) : \mathcal{T} \subseteq \bigcup_{j\le m} \mathcal{C}(\boldsymbol{q}_j, r_j)\right\}$, where the caps have radii $r_1, \ldots, r_m$. Let $\theta_m^{\mathcal{T}} = \max_{j\le m} r_j$ denote the radius of the widest cap.[3] Since a single cap of radius $\theta_1^{\mathcal{T}}$, the smallest cap containing $\mathcal{T}$, already covers the entire target set, no cap in the optimal m-cap cover needs to be wider than $\theta_1^{\mathcal{T}}$. Hence $\theta_m^{\mathcal{T}} \le \theta_1^{\mathcal{T}}$.

Theorem 1 (Cost of full recall with one versus multiple query vectors)**.** *Assume* $\theta_1^{\mathcal{T}} < \pi/2$. *Since* $\text{cost}_1^{\mathcal{T}} = A(\theta_1^{\mathcal{T}})$ *and* $\text{cost}_m^{\mathcal{T}} \le mA(\theta_m^{\mathcal{T}})$,

$$\frac{\text{cost}_m^{\mathcal{T}}}{\text{cost}_1^{\mathcal{T}}} \le \frac{mA(\theta_m^{\mathcal{T}})}{A(\theta_1^{\mathcal{T}})} \le \frac{m}{\cos\theta_m^{\mathcal{T}}} \left(\frac{\sin\theta_m^{\mathcal{T}}}{\sin\theta_1^{\mathcal{T}}}\right)^{d-1}. \tag{1}$$

Theorem 1 is proved in Appendix A. Since $\theta_m^{\mathcal{T}} \le \theta_1^{\mathcal{T}}$, using more query vectors can only make the widest cap narrower. Eq. (1) shows that even a modest reduction in this radius can lead to a large reduction in retrieval cost, because the ratio of the radii enters exponentially through the $(d-1)$-th power, while the number of query vectors contributes only a linear factor. Since d is typically in the hundreds, this exponential effect can easily outweigh the additional factor of m. Thus, *if the query vectors can be placed appropriately, multiple query vectors can be substantially more cost-efficient than a single one*.

2.2 Clustering the History Is Not Enough

Theorem 1 shows that multiple query vectors can substantially reduce retrieval cost when they are placed well, but it does not tell us where to place them. This is challenging because $\mathcal{T}$ consists of future interactions and is unknown at query time. A natural solution is to cluster $\mathcal{H}$, the observed interaction history, and use the resulting cluster centers as query vectors. However, a center estimated from the history may be misaligned with the corresponding interest in the future for two reasons. First, an interest may contribute only a limited number of historical interactions, so its center can only be estimated with finite-sample error, which we call *sampling error*. Second, the interest itself may move over time, so its center in the history may differ from its center in the future, which we call *drift error*. We next show that even small errors of either type can lead to a large increase in retrieval cost. We empirically evaluate this natural history-clustering strategy in our experiments.

We call an *interest* a group of a user's engagements lying within some angle of a center on $\mathcal{S}^{d-1}$. For an interest that appears in both the history and future windows, let $\boldsymbol{c}_h, \theta_h$ and $\boldsymbol{c}_f, \theta_f$ denote its center and radius in the two windows, respectively.

[3] Covering a point set with m balls while minimizing total cost is known as the *min-sum-radii* problem, typically with radius as the cost (Charikar & Panigrahy, 2001; Gibson et al., 2012). Here, we instead measure the cost of each spherical cap by its area. The problem is NP-hard in general.

Theorem 2 (Cost of a query vector computed from the history). *Fix an interest, and let $\delta = \angle(\boldsymbol{c}_h, \boldsymbol{c}_f)$ be how far its center moves between the history and future windows. Let $\bar{\boldsymbol{q}}$ be the query vector obtained by clustering the n history items in $\mathcal{H}$ associated with this interest, and let $\sigma = \angle(\bar{\boldsymbol{q}}, \boldsymbol{c}_h)$ denote the error in estimating $\boldsymbol{c}_h$. The angle between $\bar{\boldsymbol{q}}$ and $\boldsymbol{c}_f$ then satisfies*

$$\epsilon = \angle\big(\bar{\boldsymbol{q}}, \boldsymbol{c}_f\big) = \sqrt{\underset{\substack{\uparrow \\ \text{sampling} \\ \text{error}}}{\sigma^2} + \underset{\substack{\uparrow \\ \text{drift} \\ \text{error}}}{\delta^2}}\,\big(1 + O(d^{-1/2})\big) \gtrsim \max(\sigma, \delta), \tag{2}$$

where $\sigma = \Theta(\theta_h/\sqrt{n})$. A query vector located at $\boldsymbol{c}_f$ can cover the future interest with radius θ_f. Because $\bar{\boldsymbol{q}}$ is displaced by ϵ, it must instead probe to radius $\theta_f + \epsilon$, increasing its cost by at least

$$\frac{A(\theta_f + \epsilon)}{A(\theta_f)} \geq \cos\theta_f \left(\frac{\sin(\theta_f + \epsilon)}{\sin\theta_f}\right)^{d-1}. \tag{3}$$

Theorem 2 is proved in Appendix B. It shows that both sampling error and drift error can displace a history-derived query from the future interest center. The resulting retrieval cost grows rapidly with this displacement, especially in high-dimensional embedding spaces. To keep the cost increase within a small constant factor, the displacement ϵ must therefore be very small, imposing stringent requirements on both the amount of historical data and the stability of user interests. For example, in a 128-dimensional embedding space, the history may need more than ten thousand interactions from the same interest, while the interest center could drift by no more than 1% of the interest radius. Detailed quantitative analysis is provided in Appendix C. Therefore, *simply clustering the history is generally insufficient: both sampling error and interest drift can misplace the query relative to future interests, and even small placement errors can lead to a sharp increase in retrieval cost.*

3 Learning to Place the Query Vectors

3.1 Query Generation

Theorem 2 shows that clustering $\mathcal{H}$ can place the query vectors poorly for the future window. We therefore *learn* the placement directly, by fitting a map $M : \mathcal{H} \mapsto \{\mathcal{C}(\hat{\boldsymbol{q}}_j, \hat{\theta}_j)\}_{j \leq m}$ whose output approximates a low-cost m-cap cover of the future target set $\mathcal{T}$.

This learned mapping addresses the two sources of error identified in Theorem 2. First, clustering estimates an interest center from only the n items associated with that interest in one user's history, leading to the sampling error $\sigma = \Theta(\theta_h/\sqrt{n})$. In contrast, a model trained across users can learn patterns shared across users and use them to infer an interest even when it is represented by only a few interactions in the current user's history, leading to more accurate query placement. Second, the model is trained to predict future interactions rather than reconstruct the historical window. It can therefore learn to place the query toward $\boldsymbol{c}_f$ rather than $\boldsymbol{c}_h$, reducing the effect of interest drift δ.

As $\mathcal{H}$ is a sequence, we implement M with a sequence model. Transformer (Vaswani et al., 2017) is well suited to this setting: a user's future interests may depend on only a subset of the interaction history, and self-attention allows the model to selectively aggregate information from relevant interactions. We therefore use a Transformer backbone. Given a history $\mathcal{H}$, let $\boldsymbol{h}_t$ denote the hidden state after processing the first t items, $\mathcal{H}_{\leq t}$. Each of the m heads maps $\boldsymbol{h}_t$ to a retrieval region, parameterized by a center and an angular radius (Figure 1c):

$$\hat{\boldsymbol{q}}_j = \boldsymbol{W}_j \boldsymbol{h}_t \,/\, \|\boldsymbol{W}_j \boldsymbol{h}_t\|, \quad \hat{\theta}_j = \mathrm{softplus}\big(\boldsymbol{w}_j^\top \boldsymbol{h}_t\big), \quad j = 1, \ldots, m, \tag{4}$$

where $\boldsymbol{W}_j$ and $\boldsymbol{w}_j$ are learned parameters. All m heads share the same hidden state, allowing the model to generate all retrieval regions in a single forward pass.

The resulting parameterization is similar in form to a variational encoder (Kingma & Welling, 2014), which likewise represents a distribution through a center and a spread. Here, however, the spread has a direct retrieval interpretation: $\hat{\theta}_j$ determines how far the retriever searches from $\hat{\boldsymbol{q}}_j$ and is optimized according to retrieval cost rather than regularized toward a prior.

3.2 Training

The goal of M is to predict a set of retrieval regions that covers the target items $\mathcal{T}$ at low cost. A direct approach would be to first compute the minimum-cost m-cap cover of $\mathcal{T}$ and use it as the

training target. However, as noted in Footnote 3, finding the optimal cover is NP-hard. We therefore avoid constructing an explicit cover and instead optimize the predicted caps directly toward covering the targets while keeping their total area small. This objective involves two competing requirements: the retrieval regions should cover as many future items as possible while probing as little area as possible. Accordingly, we introduce a miss term to encourage target coverage and an area term to control the size of the retrieval regions.

The miss term. For an item $\boldsymbol{e}$, we define its *slack* with respect to cap j as $s_j(\boldsymbol{e}) = \angle(\hat{\boldsymbol{q}}_j, \boldsymbol{e}) - \hat{\theta}_j$, which measures how far $\boldsymbol{e}$ is from being covered by cap j. The slack is negative when the item lies inside the cap and positive otherwise. We penalize only positive slack using the hinge loss

$$L_{\text{miss}} = \max\Big(0,\ \text{softmin}_{j\le m}\, s_j(\boldsymbol{e})\Big). \tag{5}$$

Here, we use a smooth softmin: $\text{softmin}_j\, x_j = \sum_j x_j \exp(-x_j/\tau)/\sum_j \exp(-x_j/\tau)$, instead of the hard minimum. A hard minimum would assign each item entirely to its nearest cap, causing unused caps to receive little or no gradient and potentially remain poorly positioned. The soft minimum instead distributes the learning signal across all caps according to their distances from the item, helping every cap learn a useful retrieval region.

The area term. The miss loss alone can be minimized by simply expanding the caps, so we additionally penalize the area probed by the retrieval regions. The total area is $\sum_j A(\hat{\theta}_j)$, which, by Eqs. (12) – (14) in Appendix B, is proportional to $\sum_j \sin^{d-1}\hat{\theta}_j$. We therefore compute the following area loss term:

$$L_{\text{area}} = \Big(\frac{1}{m}\sum\nolimits_{j=1}^{m} \sin^{d-1}\hat{\theta}_j\Big)^{1/(d-1)}. \tag{6}$$

The $(d-1)$-th root keeps it on the same scale as the radii. Because d is typically large, directly evaluating Eq. (6) can cause $\sin^{d-1}\hat{\theta}_j$ to underflow in `fp32`. We therefore compute the exactly equivalent form: $\mathcal{L}_{\text{area}} = \exp\Big(\frac{1}{d-1}\big[\log\sum_{j=1}^{m}\exp\big((d-1)\log\sin\hat{\theta}_j\big) - \log m\big]\Big)$, The inner $\log\sum\exp(\cdot)$ is evaluated using the standard numerically stabilized form, which subtracts $\max_j(d-1)\log\sin\hat{\theta}_j$ before exponentiation and adds it back afterwards. This avoids numerical underflow even when the predicted radii are very small.

The next-token prediction term. To stabilize training, we retain a standard next-token prediction objective. For each item $\boldsymbol{e}$, we define its retrieval score as the negative slack of the best-matching cap: $f(\boldsymbol{e}) = -\min_{j\le m} s_j(\boldsymbol{e})$. Thus, an item receives a higher score when it lies closer to, or deeper inside, at least one cap. Given the ground-truth next item $\boldsymbol{e}$, we compute the next-token loss over the negative item set $\mathcal{N}$:

$$L_{\text{ntp}} = -\log\frac{\exp\big(f(\boldsymbol{e})/\tau\big)}{\sum_{\boldsymbol{e}'\in\mathcal{N}}\exp\big(f(\boldsymbol{e}')/\tau\big)}. \tag{7}$$

We use all items as negatives on small datasets and in-batch negatives on large datasets.

The overall training objective. The overall training objective combines the three loss terms:

$$L = \frac{1}{|\mathcal{D}|}\sum_{\mathcal{H}\in\mathcal{D}}\frac{1}{|\mathcal{H}|}\sum_{i\in\mathcal{H}}\big(L_{\text{miss}} + \lambda_{\text{area}}L_{\text{area}} + \lambda_{\text{ntp}}L_{\text{ntp}}\big), \tag{8}$$

where $\mathcal{D}$ is the training dataset, λ_{area} and λ_{ntp} are balancing hyperparameters. The model is trained autoregressively: for each item i in a user's interaction sequence $\mathcal{H}$, the preceding sequence $\mathcal{H}_{<i}$ is used to generate the retrieval caps, and the three loss terms are then computed against the target item i. Thus, each position in the sequence provides a training example.

3.3 Serving

At serving time, a single forward pass produces the centers and radii of all m caps. The retrieval procedure depends on the size of the item set and the available retrieval infrastructure.

Small-scale offline evaluation. For datasets with a relatively small item corpus, we exhaustively evaluate all items against the predicted caps. For each item, we compute its slack with respect to every cap and use the minimum slack as its retrieval score. This score jointly captures the item's

angular distance from the cap centers and the corresponding cap radii. We then rank all items by their minimum slack and select the top-K items as the retrieved candidates.

Large-scale online serving. In a production-scale setting, exhaustively scoring all items is infeasible, so retrieval is performed with an approximate-nearest-neighbor (ANN) infrastructure. Standard ANN systems are primarily designed for top-k nearest-neighbor search rather than radius-based queries. We therefore use the m predicted centers as independent ANN queries and allocate the total candidate budget according to the predicted retrieval radii. This procedure makes the predicted retrieval radius operational: larger retrieval regions receive a larger ANN budget, while compact regions are probed with fewer candidates. We merge and deduplicate the candidate lists, assigning each item its maximum score across the queries that retrieved it. The deduplicated candidates are then ranked by this score and passed to the downstream ranking stage.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate our approach on four datasets: *MovieLens-1M*, *MovieLens-20M* (Harper & Konstan, 2015), *KuaiRand-27K* (Gao et al., 2022), and an *internal short-video dataset*. MovieLens-1M and MovieLens-20M are movie recommendation datasets based on explicit rating histories, where each rating is treated as a positive interaction. KuaiRand-27K is a large-scale short-video recommendation dataset, and we use interactions from its main recommendation surface and retain users with sufficient positive interactions. The internal short-video dataset consists of short-video user interaction histories collected from our company's recommendation platform. For this dataset, we retain positive interactions from each user's most recent $1{,}024$ interactions and apply a bilateral 5-core filter. Detailed preprocessing, filtering, item representation, and dataset statistics are provided in Appendix D.

Baseline methods. We compare with both single-interest and multiple-interest approaches. *SASRec* (Kang & McAuley, 2018) and *HSTU* (Zhai et al., 2024) represent each user's history with a single query vector, while *MIND* (Li et al., 2019) and *ComiRec* (Cen et al., 2020) explicitly extract multiple interest vectors for retrieval. We also include *UIH k-means*, which applies k-means to the observed interaction history and uses the resulting cluster centers as multiple query vectors, as discussed in Section 2.2. All methods use the same item embeddings, data splits, negative sampling strategy, model dimensions, and evaluation protocol. More details on the baselines and their implementations are provided in Appendix E.

Implementation. Each item in a user's history is represented as one input token to a Llama-style decoder-only Transformer with 4 causal decoder layers, hidden dimension 256, 2 attention heads, feed-forward dimension $1{,}024$, RMSNorm, rotary position embeddings, SwiGLU feed-forward layers, and grouped-query attention. The item embedding dimension is 256. The model is trained from scratch with dropout 0.3 and a maximum sequence length of 200. The hidden state at each position is projected into 3 retrieval caps by default. We use a sampled-softmax (InfoNCE) objective with temperature 0.05, learning rate 0.002 with cosine scheduling, weight decay 0.01, $\lambda_{\text{area}} = 0.1$ and $\lambda_{\text{ntp}} = 1$. For MovieLens-1M, which contains only $3{,}883$ items, the full item corpus is scored at each training step. For the other datasets, we sample 128 negatives per positive.

Evaluation metrics. All methods are evaluated using the same full-corpus leave-one-out protocol. For each user, we use $\mathcal{H}_{<|\mathcal{H}|}$ as the input history and hold out $\mathcal{H}_{|\mathcal{H}|}$ as the target item. The target is ranked against the entire item corpus. We report HR@k and NDCG@k for $k \in \{10, 200\}$, averaged over all evaluation users. Because exactly one item is held out for each user, HR@k is equivalent to Recall@k. We use HR@k for consistency with the baseline literature. The two cutoffs measure retrieval quality at different depths: $k = 10$ emphasizes the top of the ranking, while $k = 200$ measures whether the target is retrieved within a larger candidate set.

4.2 Empirical Study

Local item density. As noted in Footnote 2, using spherical area as a proxy for retrieval cost does not require globally uniform item embeddings. It only requires item density to be reasonably stable across the regions where items occur. We examine this property in our internal short-video dataset

Methods	MovieLens-1M				MovieLens-20M				KuaiRand-27k				Internal Short-Video			
	H@10	H@200	N@10	N@200	H@10	H@200	N@10	N@200	H@10	H@200	N@10	N@200	H@10	H@200	N@10	N@200
SASRec	28.3	74.9	15.8	24.8	28.4	76.5	15.9	25.0	1.45	10.9	0.74	**2.35**	3.09	18.1	1.60	4.20
HSTU	29.8	**75.9**	16.5	25.6	32.4	78.0	18.7	27.6	1.49	**11.5**	0.72	2.33	3.82	20.3	1.97	4.84
MIND	8.21	51.5	4.06	11.6	8.50	50.1	4.26	11.6	0.60	6.87	0.27	1.31	0.43	3.69	0.21	0.76
ComiRec-SA	21.8	68.7	11.7	20.5	15.1	61.5	7.88	16.3	0.50	3.50	0.24	0.76	0.58	5.20	0.30	1.07
ComiRec-DR	11.1	53.0	5.70	13.1	7.87	49.1	3.80	11.1	0.41	3.51	0.19	0.73	0.59	4.27	0.28	0.90
UIH k-means	2.91	23.3	1.34	4.80	7.31	47.1	3.59	10.6	0.16	1.81	0.07	0.35	0.54	5.06	0.26	1.00
Ours	**31.0**	75.6	**17.6**	**26.2**	**33.2**	**78.2**	**19.0**	**28.8**	**1.56**	10.4	**0.79**	2.29	**4.90**	**24.5**	**2.59**	**6.01**

Table 1: Results of hit rate (H@10, H@200) and NDCG (N@10, N@200) on four datasets. The best results are highlighted in **bold**.

by randomly sampling 2,000 items and computing the mean distance to each item's 100 nearest neighbors as a measure of local density. We compare the resulting distribution with two synthetic references: uniformly distributed points and a highly non-uniform clustered catalog. As shown in Figure 3, the real item space has a substantially narrower density distribution than the clustered reference and is much closer to the uniform case. This indicates that local item density varies within a relatively limited range, supporting the use of spherical area as a proxy for retrieval cost. Detailed experimental settings are provided in Appendix F.1.

Multiple interests and interest drift. User histories often contain multiple local groups and exhibit substantial temporal changes. As illustrated in Figure 4, engaged items from the same user can form several local groups in the item embedding space, while items from different time periods may occupy noticeably different regions. These patterns suggest that a single query may be insufficient to efficiently cover a user's interests, while a query inferred from past interactions may be misaligned with future interests. Such misalignment can arise from both the limited number of historical interactions and temporal interest drift, as formalized in Theorem 2. A more detailed visualization and analysis is provided in Appendix F.2.

4.3 Main Results

Table 1 compares our method with single-interest and multi-interest baselines across four datasets. Our method achieves the best result on 13 of the 16 reported metrics, including the highest HR@10 and NDCG@10 on all four datasets. The gains are particularly significant on the internal short-video dataset, where HR@10 and NDCG@10 reach 4.90 and 2.59, compared with 3.82 and 1.97 for HSTU. On KuaiRand-27K, HSTU performs relatively strongly at larger cutoffs, which may be partly due to HSTU's design for long and non-stationary recommendation sequences. Overall, the results demonstrate the effectiveness of modeling multiple retrieval regions across both movie and short-video recommendation.

Compared with existing multi-interest methods, our approach achieves substantially higher retrieval quality across all four datasets. In particular, UIH k-means performs poorly despite using multiple query vectors, suggesting that directly clustering the observed history does not provide effective query locations for future retrieval. These results are consistent with our analysis of sampling and drift errors, and highlight the importance of learning both the location and size of retrieval regions.

4.4 Ablation Study

Loss terms. Our objective combines three complementary terms: L_{miss} encourages the retrieval caps to cover the target, L_{area} constrains their size, and L_{ntp} provides the representation and ranking signal. We ablate each term while keeping the others fixed. We evaluate both *full-corpus ranking*, which ranks all items, and *cap-restricted ranking*, which ranks only items inside the union of the predicted caps. We also report *Scan*, the percentage of the item corpus contained in this union. As shown in Table 2, the ablations reveal distinct roles for the three terms. Removing L_{miss} collapses the caps to nearly zero area, leaving no target covered by the retrieval region. Removing L_{area} preserves nearly the same ranking quality but expands the scanned region from 9.8% to 38.5% of the corpus. Finally, removing L_{ntp} causes the representation to collapse, resulting in near-random ranking and caps covering almost the entire corpus. Together, the three terms balance representation quality, target coverage, and retrieval efficiency.

	Full Corpus		Cap-Restricted		Scan
	HR@10	HR@200	HR@10	HR@200	
Full objective	31.0	75.6	31.0	73.0	9.76%
w/o L_{miss}	28.2	74.6	0.00	0.00	0.00%
w/o L_{area}	28.0	74.8	28.0	74.6	38.5%
w/o L_{ntp}	0.28	4.16	0.28	4.16	98.7%

Table 2: Effect of ablating the three loss terms on MovieLens-1M.

$L \times H$	Params	HR@10	HR@200	NDCG@10	NDCG@200
2×64	0.13M	21.3	67.4	11.7	20.4
1×128	0.26M	21.6	66.5	11.8	20.2
2×128	0.52M	26.6	71.4	15.0	23.6
3×192	1.77M	29.7	73.5	16.9	25.4
4×256	4.20M	**31.0**	**75.6**	**17.6**	**26.2**
6×256	6.29M	30.5	70.4	17.4	25.3

Table 3: Effect of the backbone model capacity on MovieLens-1M.

Model capacity. We study the effect of backbone model capacity by varying the number of layers L and hidden dimensions H while keeping other hyperparameters fixed. As shown in Table 3, increasing the model size substantially improves both head and tail retrieval quality up to 4.20M parameters. Beyond this point, accuracy saturates: the 6.29M model slightly underperforms the 4.20M model on all four metrics. Interestingly, additional capacity beyond this accuracy saturation point still reduces the retrieval cost by producing smaller retrieval regions, as analyzed in Appendix G.1. We therefore use the 4×256 configuration as the default throughout the paper.

Number of query heads. We study the effect of the number of query heads m on MovieLens-20M. As shown in Table 4, performance is stable when $m \leq 8$, with $m = 3$ achieving the best overall results, while performance drops substantially at $m = 16$. Examining the training dynamics reveals that, as m increases, supervision becomes increasingly concentrated on a subset of caps, leaving others with few assignments and weak gradients (see Appendix G.2 for details). This creates a self-reinforcing effect in which underused caps become less likely to attract future targets. At the same time, increasing m consistently reduces the retrieval cost: *Scan* decreases from 24.1% with $m = 1$ to 9.76% with $m = 3$, and further to 1.12% with $m = 16$. Thus, additional query heads can substantially reduce the retrieved region, but excessive splitting can hurt retrieval quality. A moderate number of query heads therefore provides a better balance between retrieval quality and cost.

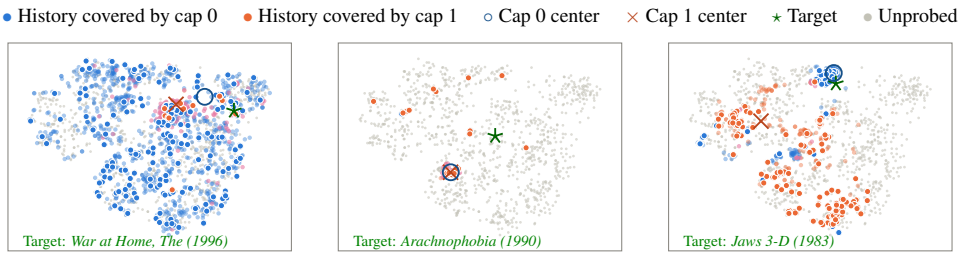
m	HR@10	HR@200	NDCG@10	NDCG@200	Scan
1	31.1	73.4	17.9	26.9	24.1%
2	31.3	76.0	**19.1**	27.2	15.3%
3	**33.2**	**78.2**	19.0	**28.8**	9.76%
4	32.8	75.4	18.5	26.5	5.21%
6	31.1	72.5	18.9	26.8	3.58%
8	31.0	72.6	18.7	26.6	2.45%
16	27.0	68.4	15.8	23.5	1.12%

Table 4: Effect of the number of query heads m on MovieLens-20M.

4.5 Case Study

We examine three representative cases from MovieLens-1M, where the model predicts two retrieval caps for each user. Figure 2 shows one failure and two successful cases, visualized by t-SNE (Van der Maaten & Hinton, 2008). The reported angles are computed in the original item embedding space. (a) *Failure on a dispersed history*. Figure 2a shows a user with 200 movies spread across a broad region of the item space. The two caps are correspondingly large, with radii 1.25 and 1.16 radians, and together probe 29.3% of the item pool. The target is covered but ranked only 437. This example shows that broad coverage alone does not guarantee good ranking: the caps can find the target while still failing to place it near the top. (b) *Successful retrieval with a concentrated history*. In Figure 2b, the 25 historical movies are concentrated in a small region, so the two caps are nearly redundant, with centers only 0.13 radians apart and a Jaccard overlap of 0.87. Nevertheless, the target is ranked 7 while probing only 1.2% of the item pool. Thus, multiple caps can safely overlap when the user's history is concentrated. (c) *Successful retrieval for multiple interests*. Figure 2c shows a user whose history separates into two distinct groups. The two cap centers are 0.89 radians apart, close to the 0.94-radian separation between the centers of the two groups. The target item is ranked 1 while the caps probe only 8.7% of the item pool. This is the case where multiple caps are most useful: they can separately cover distinct interests instead of placing a single query between them.

These examples show that the learned caps adapt to the structure of each user's history rather than always behaving as multiple independent queries. For dispersed histories, the caps spread out to improve coverage; for concentrated histories, they overlap and avoid unnecessary expansion; and for users with distinct interests, they separate to cover different regions. Thus, multiple caps provide a flexible representation that can adjust both their locations and coverage according to the user's specific interest pattern.



(a) Dispersed history (b) Concentrated history (c) Two distinct interests

Figure 2: Three representative users from MovieLens-1M, each with two predicted retrieval caps. Points are shown in the shared 256-d item embedding space using t-SNE. Colored points are covered by the caps, while gray points are unprobed. Historical items are highlighted, with blue and orange indicating their assigned cap. The star marks the target item.

5 Related Work

Multi-interest user modeling. Representing users with multiple vectors is well established in recommendation (Li et al., 2019; Cen et al., 2020; Hao et al., 2021; Wu et al., 2025), motivated by the limitation of single embeddings in capturing multiple preferences (Guzman-Rivera et al., 2012; Rupprecht et al., 2017; Yang et al., 2018). See Li et al. (2026) for a recent survey. PinnerSage (Pal et al., 2020) clusters historical actions into multiple interests, while other methods use routing or attention to extract interest vectors. Despite different architectures, these approaches generally construct query vectors directly from historical interactions, motivating our analysis of the sampling and drift errors of history-based query construction in Theorem 2.

Multi-vector retrieval and the limits of one embedding. In information retrieval, multi-vector retrieval typically represents each document with multiple vectors, as in ColBERT (Khattab & Zaharia, 2020), with subsequent work addressing its computational and storage costs (Lee et al., 2023; Dhulipala et al., 2024; MacAvaney et al., 2025). In contrast, we use a single embedding per item and study the geometric cost of using multiple query vectors to cover a target set. Related work also studies the representational limitations of a single embedding, including the representation of top-k results (Weller et al., 2026), required dimensionality (Wang et al., 2026), single- versus multi-vector representations (Agarwal et al., 2026a;b), and quantization (Okajima & Yoshida, 2026). More closely related to our formulation, Li et al. (2024) represent relations with spheres, while Chen et al. (2025) use multiple query vectors for multi-answer questions. These works focus on representation rather than explicitly modeling the retrieval region and its cost.

Generative recommendation. Generative recommendation uses sequence models to generate item identifiers or continuous retrieval representations from user history. Most methods generate discrete identifiers (Geng et al., 2022; Rajput et al., 2023; Zhai et al., 2024; Deng et al., 2025), while recent approaches generate continuous representations (Yang et al., 2025b; Qu et al., 2026; Yang et al., 2025a) or use diffusion to model multiple interests (Wang et al., 2023; Yang et al., 2023). Wu et al. (2024) further formulate generative retrieval as multi-vector dense retrieval. In contrast, our model directly generates explicit retrieval regions, whose placement and size can be jointly optimized for retrieval cost and target coverage.

6 Conclusion

We presented a geometric method for retrieval in recommender systems, viewing cosine-similarity retrieval as searching spherical caps in the item embedding space. We formulated efficient multi-query retrieval as a geometric covering problem and identified sampling and drift as key sources of query placement error. Based on these insights, we proposed learning multiple retrieval caps directly from user histories, jointly optimizing their locations and sizes for future-item coverage and retrieval efficiency. Experiments across four datasets show consistent improvements over both single-interest and multi-interest baselines, while maintaining compact retrieval regions. These results demonstrate the benefit of learning *where to look* and *how far to search* for efficient personalized retrieval. We discuss the limitations of our approach and directions for future work in Appendix H.

AI USE STATEMENT

We used generative AI tools only after completing the research, methodology, experiments, analysis, and initial manuscript preparation. Their use was limited to improving the wording, grammar, clarity, and readability of the manuscript, including revising sentences and improving the presentation of technical descriptions. Generative AI was not used to develop the research ideas, design the methodology or experiments, implement the models, analyze or interpret experimental results, or generate any scientific claims or results. All AI-assisted edits were reviewed and verified by the authors. The authors remain fully responsible for the accuracy, originality, and final content of the manuscript.

ETHICS STATEMENT

This work studies recommendation using user interaction data and does not involve direct interaction with human participants. Our experiments use publicly available recommendation datasets as well as an internal short-video dataset collected and processed in accordance with applicable company policies, privacy requirements, and data governance procedures. The paper reports aggregate experimental results and does not disclose personally identifying information or individual user identities. We have taken care not to include sensitive user-level information in the manuscript or released materials. Any future release of internal data or derived resources will be subject to the applicable privacy, security, and data-sharing requirements, including anonymization where necessary.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the proposed method and experimental setup throughout the paper and appendices, including the model architecture, retrieval formulation, training objectives, serving procedure, dataset preprocessing, evaluation protocol, and baseline implementations. The ablation studies and additional analyses further document the main design choices and their effects on retrieval quality and retrieval cost. Where appropriate, we also provide the relevant hyperparameters, model configurations, and implementation details needed to reproduce the reported experiments. We plan to release the source code and, to the extent permitted by applicable company policies and data-sharing requirements, the internal dataset or an appropriately anonymized version of the data. These materials will be released with the goal of enabling independent reproduction of our main results while respecting privacy, security, and confidentiality constraints.

REFERENCES

Mihir Agarwal, Viraj Agrawal, Sabyasachi Basu, Ankit Garg, and Kirankumar Shiragur. Retrieval needs multivectors: An exponential separation. *arXiv preprint arXiv:2608.21494*, 2026a.

Mihir Agarwal, Ankit Garg, Neeraj Kayal, Kirankumar Shiragur, et al. On strengths and limitations of single-vector embeddings. *arXiv preprint arXiv:2603.29519*, 2026b.

Oren Barkan and Noam Koenigstein. Item2vec: Neural item embedding for collaborative filtering. In *2016 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6. IEEE, 2016. doi: 10.1109/MLSP.2016.7738886.

Yukuo Cen, Jianwei Zhang, Xu Zou, Chang Zhou, Hongxia Yang, and Jie Tang. Controllable multi-interest framework for recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2942–2951, 2020.

Moses Charikar and Rina Panigrahy. Clustering to minimize the sum of cluster diameters. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC)*, pp. 1–10, 2001.

Hung-Ting Chen, Xiang Liu, Shauli Ravfogel, and Eunsol Choi. Beyond single embeddings: Capturing diverse targets with multi-query retrieval. *arXiv preprint arXiv:2511.02770*, 2025.

Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pp. 191–198, 2016.

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

Laxman Dhulipala, Majid Hadian, Rajesh Jayaram, Jason Lee, and Vahab Mirrokni. MUVERA: Multi-vector retrieval via fixed dimensional encoding. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 101042–101073, 2024. arXiv:2405.19504.

Ahmed El-Kishky, Thomas Markovich, Serim Park, Chetan Verma, Baekjin Kim, Ramy Eskander, Yury Malkov, Frank Portman, Sofía Samaniego, Ying Xiao, and Aria Haghighi. Twhin: Embedding the twitter heterogeneous information network for personalized recommendation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2842–2850. ACM, 2022. doi: 10.1145/3534678.3539080.

Kawin Ethayarajh. How contextual are contextualized word representations? comparing the geometry of BERT, ELMo, and GPT-2 embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019. arXiv:1909.00512.

Chongming Gao, Shijun Li, Yuan Zhang, Jiawei Chen, Biao Li, Wenqiang Lei, Peng Jiang, and Xiangnan He. Kuairand: An unbiased sequential recommendation dataset with randomly exposed videos. In *CIKM*, 2022.

Jun Gao, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tie-Yan Liu. Representation degeneration problem in training natural language generation models. In *International Conference on Learning Representations (ICLR)*, 2019. arXiv:1907.12009.

Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5). In *Proceedings of the 16th ACM conference on recommender systems*, pp. 299–315, 2022.

Matt Gibson, Gaurav Kanade, Erik Krohn, Imran A. Pirwani, and Kasturi R. Varadarajan. On clustering to minimize the sum of radii. *SIAM Journal on Computing*, 41(1):47–60, 2012.

Abner Guzman-Rivera, Dhruv Batra, and Pushmeet Kohli. Multiple choice learning: Learning to produce multiple structured outputs. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

Junmei Hao, Jingcheng Shi, Qing Da, Anxiang Zeng, Yujie Dun, Xueming Qian, and Qianying Lin. Diversity regularized interests modeling for recommender systems. *arXiv preprint arXiv:2103.12404*, 2021.

F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems*, 5(4):1–19, 2015.

Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*, pp. 197–206. IEEE, 2018. doi: 10.1109/ICDM.2018.00035.

Omar Khattab and Matei Zaharia. ColBERT: Efficient and effective passage search via contextualized late interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014. arXiv:1312.6114.

Jinhyuk Lee, Zhuyun Dai, Sai Meher Karthik Duddu, Tao Lei, Iftekhar Naim, Ming-Wei Chang, and Vincent Zhao. Rethinking the role of token retrieval in multi-vector retrieval. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 15384–15405, 2023. arXiv:2304.01982.

Chao Li, Zhiyuan Liu, Mengmeng Wu, Yuchi Xu, Pipei Huang, Huan Zhao, Guoliang Kang, Qiwei Chen, Wei Li, and Dik Lun Lee. Multi-interest network with dynamic routing for recommendation at tmall. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM)*, pp. 2615–2623, 2019.

Zihao Li, Yuyi Ao, and Jingrui He. SpherE: Expressive and interpretable knowledge graph embedding for set retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2024. arXiv:2404.19130.

Zihao Li, Qiang Chen, Lixin Zou, Aixin Sun, and Chenliang Li. Multi-interest recommendation: A survey. *ACM Transactions on Information Systems*, 44(4):1–38, 2026. arXiv:2506.15284.

Sean MacAvaney, Antonio Mallia, and Nicola Tonellotto. Efficient constant-space multi-vector retrieval. In *European Conference on Information Retrieval (ECIR)*, pp. 237–245, 2025. arXiv:2504.01818.

Koki Okajima and Tsukasa Yoshida. What limits does quantization place on dense top-k retrieval? a theoretical study. *arXiv preprint arXiv:2606.11780*, 2026.

Aditya Pal, Chantat Eksombatchai, Yitong Zhou, Bo Zhao, Charles Rosenberg, and Jure Leskovec. PinnerSage: Multi-modal user embedding framework for recommendations at Pinterest. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2020. arXiv:2007.03634.

Haohao Qu, Shanru Lin, Yujuan Ding, Yiqi Wang, and Wenqi Fan. Diffusion generative recommendation with continuous tokens. In *Proceedings of the ACM Web Conference*, pp. 7259–7270, 2026. doi: 10.1145/3774904.3792428.

Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan H. 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 (NeurIPS)*, 2023.

Christian Rupprecht, Iro Laina, Robert DiPietro, Maximilian Baust, Federico Tombari, Nassir Navab, and Gregory D. Hager. Learning in an uncertain world: Representing ambiguity through multiple hypotheses. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

Venu Satuluri, Yao Wu, Xun Zheng, Yilei Qian, Brian Wichers, Qieyun Dai, Gui Ming Tang, Jerry Jiang, and Jimmy Lin. Simclusters: Community-based representations for heterogeneous recommendations at twitter. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 3183–3193. ACM, 2020. doi: 10.1145/3394486.3403370.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

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*, pp. 832–841, 2023.

Zihao Wang, Hang Yin, Lihui Liu, Hanghang Tong, Yangqiu Song, Ginny Wong, and Simon See. $\mathbb{R}^{2k}$ is theoretically large enough for embedding-based top-k retrieval. *arXiv preprint arXiv:2601.20844*, 2026.

Orion Weller, Michael Boratko, Iftekhar Naim, and Jinhyuk Lee. On the theoretical limitations of embedding-based retrieval. In *International Conference on Learning Representations (ICLR)*, 2026.

Shiguang Wu, Wenda Wei, Mengqi Zhang, Zhumin Chen, Jun Ma, Zhaochun Ren, Maarten de Rijke, and Pengjie Ren. Generative retrieval as multi-vector dense retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2024.

Zhibo Wu, Yunfan Wu, Quan Liu, Lin Jiang, Ping Yang, and Yao Hu. GemiRec: Interest quantization and generation for multi-interest recommendation. *arXiv preprint arXiv:2510.14626*, 2025.

Liu Yang, Fabian Paischer, Kaveh Hassani, Jiacheng Li, Shuai Shao, Zhang Gabriel Li, Yun He, Xue Feng, Nima Noorshams, Sem Park, Bo Long, Robert D. Nowak, Xiaoli Gao, and Hamid Eghbalzadeh. Unifying generative and dense retrieval for sequential recommendation. *Transactions on Machine Learning Research*, 2025a.

Yuhao Yang, Zhi Ji, Zhaopeng Li, Yi Li, Zhonglin Mo, Yue Ding, Kai Chen, Zijian Zhang, Jie Li, Shuanglong Li, and Lin Liu. Sparse meets dense: Unified generative recommendations with cascaded sparse-dense representations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025b.

Zhengyi Yang, Jiancan Wu, Zhicai Wang, Xiang Wang, Yancheng Yuan, and Xiangnan He. Generate what you prefer: Reshaping sequential recommendation via guided diffusion. *Advances in Neural Information Processing Systems*, 36:24247–24261, 2023.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. Breaking the softmax bottleneck: A high-rank RNN language model. In *International Conference on Learning Representations (ICLR)*, 2018.

Jiaqi Zhai, Lucy Liao, Xing Liu, Yueming Wang, Rui Li, Xuan Cao, Leon Gao, Zhaojie Gong, Fangda Gu, Michael He, Yinghai Lu, and Yu Shi. Actions speak louder than words: Trillion-parameter sequential transducers for generative recommendations. In *International Conference on Machine Learning (ICML)*, 2024.

A Proof of Theorem 1

Proof. We establish the theorem in four steps. First, we characterize the cost of a single query and that of m queries in terms of cap measures. We then derive a bound on the ratio of two cap measures and combine the results.

Step 1: Cost of one query. Scores are cosine similarities, so thresholding the score of a query vector $\boldsymbol{q}$ is equivalent to retrieving all items within an angular radius r of $\boldsymbol{q}$, i.e., within the spherical cap $\mathcal{C}(\boldsymbol{q}, r)$. Thus, retrieving all items in $\mathcal{T}$ with a single query requires $\mathcal{T} \subseteq \mathcal{C}(\boldsymbol{q}, r)$. By definition of $\theta_1^{\mathcal{T}}$, no cap of radius smaller than $\theta_1^{\mathcal{T}}$ can cover $\mathcal{T}$. Hence $r \geq \theta_1^{\mathcal{T}}$, and since $A(\theta)$ is increasing, the probed area is at least $A(\theta_1^{\mathcal{T}})$. This radius is achievable by the definition of $\theta_1^{\mathcal{T}}$, giving

$$\mathrm{cost}_1^{\mathcal{T}} = A(\theta_1^{\mathcal{T}}). \tag{9}$$

Step 2: Cost of m queries. Let $(\boldsymbol{q}_j, r_j)_{j=1}^m$ be an optimal set of caps covering $\mathcal{T}$, so that $\mathrm{cost}_m^{\mathcal{T}} = \sum_{j=1}^m A(r_j)$. By definition of $\theta_m^{\mathcal{T}}$, every optimal radius satisfies $r_j \leq \theta_m^{\mathcal{T}}$. Since $A(\theta)$ is increasing,

$$\mathrm{cost}_m^{\mathcal{T}} = \sum_{j=1}^{m} A(r_j) \leq m\, A(\theta_m^{\mathcal{T}}). \tag{10}$$

Combining this with Step 1 gives

$$\frac{\mathrm{cost}_m^{\mathcal{T}}}{\mathrm{cost}_1^{\mathcal{T}}} \leq m\, \frac{A(\theta_m^{\mathcal{T}})}{A(\theta_1^{\mathcal{T}})}. \tag{11}$$

We will use Eq. (15) below, which requires $\theta_m^{\mathcal{T}} \leq \theta_1^{\mathcal{T}}$. A single cap of radius $\theta_1^{\mathcal{T}}$ covers $\mathcal{T}$, so it is also a feasible solution for the m-cap problem by assigning zero radius to the remaining $m-1$ caps. Therefore, $\mathrm{cost}_m^{\mathcal{T}} \leq A(\theta_1^{\mathcal{T}})$. If any optimal radius r_j were larger than $\theta_1^{\mathcal{T}}$, then $\mathrm{cost}_m^{\mathcal{T}} \geq A(r_j) > A(\theta_1^{\mathcal{T}})$, a contradiction. Hence $\theta_m^{\mathcal{T}} \leq \theta_1^{\mathcal{T}}$.

Step 3: Ratio of two cap measures. By rotation invariance, the normalized measure of a spherical cap can be written as

$$A(\theta) = \frac{\int_0^{\theta} \sin^{d-2} t \, \mathrm{d}t}{\int_0^{\pi} \sin^{d-2} t \, \mathrm{d}t}. \tag{12}$$

The factor $\sin^{d-2} t$ arises because the points at angular distance t from the cap center form a $(d-2)$-sphere of radius $\sin t$. Since the denominator is independent of θ, it cancels in any ratio of cap measures. It therefore suffices to bound

$$I(\psi) = \int_0^{\psi} \sin^{d-2} t \, \mathrm{d}t. \tag{13}$$

We use

$$\frac{\mathrm{d}}{\mathrm{d}t} \sin^{d-1} t = (d-1) \sin^{d-2} t \cos t.$$

Thus,

$$I(\psi) = \frac{1}{d-1} \int_0^{\psi} \frac{1}{\cos t} \, \mathrm{d}\big(\sin^{d-1} t\big).$$

For $0 \leq t \leq \psi < \pi/2$, we have $1 \leq 1/\cos t \leq 1/\cos \psi$. Applying these bounds inside the integral gives

$$\frac{\sin^{d-1} \psi}{d-1} \leq I(\psi) \leq \frac{\sin^{d-1} \psi}{(d-1)\cos \psi}. \tag{14}$$

Now let $0 < r' \leq r < \pi/2$. Applying the lower bound in Eq. (14) to $I(r)$ and the upper bound to $I(r')$, and vice versa, yields

$$\cos r' \left(\frac{\sin r}{\sin r'}\right)^{d-1} \leq \frac{A(r)}{A(r')} \leq \frac{1}{\cos r} \left(\frac{\sin r}{\sin r'}\right)^{d-1}. \tag{15}$$

The two bounds differ only by the factor $1/(\cos r \cos r')$, which is independent of d. Thus, the dominant dependence on the dimension is captured by $(\sin r / \sin r')^{d-1}$.

Step 4: Combining the bounds. Applying the left-hand side of Eq. (15) with $r = \theta_1^{\mathcal{T}}$ and $r' = \theta_m^{\mathcal{T}}$ gives

$$\frac{A(\theta_m^{\mathcal{T}})}{A(\theta_1^{\mathcal{T}})} \leq \frac{1}{\cos\theta_m^{\mathcal{T}}} \left(\frac{\sin\theta_m^{\mathcal{T}}}{\sin\theta_1^{\mathcal{T}}}\right)^{d-1}.$$

Combining this with Step 2 yields

$$\frac{\mathrm{cost}_m^{\mathcal{T}}}{\mathrm{cost}_1^{\mathcal{T}}} \leq \frac{m}{\cos\theta_m^{\mathcal{T}}} \left(\frac{\sin\theta_m^{\mathcal{T}}}{\sin\theta_1^{\mathcal{T}}}\right)^{d-1}, \tag{16}$$

which is Eq. (1). □

B Proof of Theorem 2

Proof. We work in the tangent space to the sphere at $\boldsymbol{c}_h$, which has dimension $d-1$. For angles well inside a hemisphere, angular distances on the sphere are proportional to Euclidean distances in this tangent space up to constant factors, so it suffices to analyze the corresponding tangent-space vectors.

Step 1: Sampling error. Each of the n historical items associated with the interest lies within θ_h of $\boldsymbol{c}_h$. Write the corresponding tangent-space displacement as $\boldsymbol{v}_i$, with $\|\boldsymbol{v}_i\| \leq \sin\theta_h$. The clustered query is obtained from their normalized mean, so its tangent-space displacement is proportional to $\bar{\boldsymbol{v}} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{v}_i$. Therefore, $\sigma = \|\bar{\boldsymbol{v}}\|$ up to a constant factor.

Because $\boldsymbol{c}_h$ is the true center, we assume $\mathbb{E}[\boldsymbol{v}_i] = 0$. Under independence,

$$\mathbb{E}\|\bar{\boldsymbol{v}}\|^2 = \frac{1}{n}\mathbb{E}\|\boldsymbol{v}\|^2.$$

Since $\|\boldsymbol{v}_i\| \leq \sin\theta_h$, we have

$$\mathbb{E}\|\boldsymbol{v}\|^2 \leq \sin^2\theta_h.$$

Conversely, if the interest mass is not concentrated arbitrarily close to its center, there exists a constant $c > 0$ such that

$$\mathbb{E}\|\boldsymbol{v}\|^2 \geq c\sin^2\theta_h.$$

Hence the sampling error scales as

$$\sigma = \Theta\left(\frac{\theta_h}{\sqrt{n}}\right).$$

In practice, interactions within an interest may be correlated, which reduces the effective sample size and can make the sampling error larger. Using the center of the minimum enclosing cap instead of the mean, as in the min-sum-radii formulation of Section 2.1, does not change the scaling over the range of interest. Empirically, at $d = 128$ and $\theta_h = 5^\circ$, the two errors remain within a factor of two for n in the hundreds.

Step 2: Adding interest drift. Let $\boldsymbol{u}$ be a unit vector in the tangent space in the direction of the drift. Then the future center can be written locally as $\boldsymbol{c}_f = \boldsymbol{c}_h + \delta\boldsymbol{u}$. The displacement between the history-derived query and the future center is therefore $\bar{\boldsymbol{v}} - \delta\boldsymbol{u}$. Hence,

$$\epsilon^2 = \|\bar{\boldsymbol{v}} - \delta\boldsymbol{u}\|^2 = \sigma^2 + \delta^2 - 2\delta\langle\bar{\boldsymbol{v}}, \boldsymbol{u}\rangle.$$

The direction of the sampling error is unrelated to the direction of the future drift, so we model the direction of $\bar{\boldsymbol{v}}$ as approximately uniform in the $(d-1)$-dimensional tangent space. Its inner product with any fixed unit direction is then $O(d^{-1/2})$, giving

$$\epsilon^2 = \sigma^2 + \delta^2 + O\left(\sigma\delta\, d^{-1/2}\right).$$

This gives Eq. (2). In particular, $\epsilon \gtrsim \max(\sigma, \delta)$, so the two sources of error cannot substantially cancel in high dimensions.

Step 3: Retrieval cost of the displaced query. The future interest lies within angular radius θ_f of $\boldsymbol{c}_f$, while the history-derived query is at angular distance ϵ from $\boldsymbol{c}_f$. By the triangle inequality, every future item is therefore within $\theta_f + \epsilon$ of the history-derived query. Thus, probing to radius $\theta_f + \epsilon$ is sufficient to cover the entire future interest. It is also necessary in the worst case: an item located on the opposite side of $\boldsymbol{c}_f$ from the displaced query is exactly $\theta_f + \epsilon$ away.

In contrast, a query placed at the future center $\boldsymbol{c}_f$ needs only radius θ_f. Applying Eq. (15) with $r = \theta_f + \epsilon$ and $r' = \theta_f$ therefore gives

$$\frac{A(\theta_f + \epsilon)}{A(\theta_f)} \geq \cos\theta_f \left(\frac{\sin(\theta_f + \epsilon)}{\sin\theta_f}\right)^{d-1}, \tag{17}$$

which is Eq. (3). □

C Deriving the Accuracy Requirement for History-Based Queries

We derive the accuracy requirement implied by Theorem 2. For small θ_f and ϵ, we have $\cos\theta_f \approx 1$, $\sin(\theta_f + \epsilon) \approx \theta_f + \epsilon$, and $\sin\theta_f \approx \theta_f$. Therefore, Eq. (3) becomes approximately

$$\frac{A(\theta_f + \epsilon)}{A(\theta_f)} \gtrsim \left(1 + \frac{\epsilon}{\theta_f}\right)^{d-1}. \tag{18}$$

For this factor to remain bounded by a constant as d grows, we require

$$\frac{\epsilon}{\theta_f} = O\left(\frac{1}{d-1}\right). \tag{19}$$

For example, setting $\epsilon/\theta_f = 1/(d-1)$ gives

$$\frac{A(\theta_f + \epsilon)}{A(\theta_f)} \gtrsim \left(1 + \frac{\epsilon}{\theta_f}\right)^{d-1} = \left(1 + \frac{1}{d-1}\right)^{d-1} \approx e. \tag{20}$$

Thus, a displacement of $\epsilon = \theta_f/(d-1)$ already incurs a cost increase of at least $e \approx 2.7$. Maintaining a constant-factor increase in retrieval cost therefore requires

$$\epsilon \lesssim \frac{\theta_f}{d-1}. \tag{21}$$

By Eq. (2), $\epsilon \gtrsim \max(\sigma, \delta)$, so both the sampling and drift errors must satisfy

$$\sigma \lesssim \frac{\theta_f}{d-1}, \qquad \delta \lesssim \frac{\theta_f}{d-1}. \tag{22}$$

Because $\sigma = \Theta(\theta_h/\sqrt{n})$, the sampling requirement becomes

$$n \gtrsim (d-1)^2 \left(\frac{\theta_h}{\theta_f}\right)^2, \tag{23}$$

while the drift requirement is

$$\delta \lesssim \frac{\theta_f}{d-1}. \tag{24}$$

These requirements are stringent in high-dimensional embedding spaces. For example, at $d = 128$, assuming the interest width remains unchanged ($\theta_h = \theta_f$), Eq. (23) requires $n \gtrsim 127^2 \approx 16{,}000$ historical interactions from the same interest. Meanwhile, Eq. (24) allows the interest center to drift by at most $\theta_f/127$, which is roughly two orders of magnitude smaller than the interest radius itself.

Therefore, history-based clustering requires both a very large number of historical interactions and extremely stable interests to maintain low retrieval cost.

D More Details on Datasets

MovieLens-1M and MovieLens-20M. The MovieLens datasets (Harper & Konstan, 2015) consist of explicit-rating logs from the MovieLens film recommender. We treat each rating event as a positive signal without thresholding its star value. MovieLens-1M contains 6,040 users and 3,883 films, with a mean user history length of 165.6, while MovieLens-20M contains 138,493 users and 27,278 films, with a mean user history length of 144.4. Both datasets guarantee at least 20 ratings per user, so no additional user filtering is applied. We represent each film using the string "$\langle$*title (year)*$\rangle$. *Genres:* g_1, g_2, ...", which is encoded with a frozen Sentence-T5-base model to obtain initial item embedding.

KuaiRand-27K. KuaiRand (Gao et al., 2022) is a sequential recommendation dataset collected from Kuaishou short-video platform. We use the KuaiRand-27K subset, which contains a random sample of 27,285 users and approximately 32 million videos. We use user ID, video ID, video-ID history, click history, like history, and long-video-view (LVV) history from the available features. We restrict interactions to `tab=1`, corresponding to the app's dominant recommendation surface and accounting for 65.9% of impressions. We use `is_click` as the positive signal and retain users with at least 11 positive interactions. After filtering, the dataset contains 27,065 users, 95.8 million interactions, and 9.7 million distinct videos, with a median sequence length of 2,884.

Internal Short-Video Dataset. We additionally evaluate on an internal short-video recommendation dataset constructed from user interaction logs in our company's recommendation platform. Each active user on the collection day contributes one user-interaction history (UIH), stored as time-ordered parallel arrays. We define an interaction as positive if it contains at least one of seven engagement actions: `like`, `share`, `comment`, `follow`, `save`, `complete view`, or `long watch` (watch time $\geq$ 60 seconds). We retain positive interactions from each user's most recent 1,024 interactions. Each retained position contains the interacted media ID and its associated side information: event `timestamp`, a 7-bit engagement bitmask (`action`), bucketized inter-event gap (`time_delta`, 32 buckets), and bucketized dwell time (`time_spent`, 20 buckets). We then apply a bilateral 5-core reduction to the user–media bipartite graph, requiring every retained user to have at least 5 distinct media and every retained media to be associated with at least 5 distinct users. After filtering, the dataset contains 10.9 million users, 11.4 million distinct media, and 2.1 billion positive interactions. The mean UIH length is 193.1.

E More Details on Baseline Methods

We consider both single-interest and multiple-interest baselines. The single-interest baselines are SASRec (Kang & McAuley, 2018) and HSTU (Zhai et al., 2024), which represent each user with a single query vector. The multiple-interest baselines are MIND (Li et al., 2019), ComiRec (Cen et al., 2020), and UIH k-means, which represent a user with multiple query vectors. For a fair comparison, we use the same item pool, leave-one-out split, sampled-softmax objective with uniform local negatives, embedding width, depth, head count, and evaluation metrics for all baselines. Thus, the main differences between methods lie in how they represent the user's history and how many query vectors they use for retrieval.

SASRec. SASRec (Kang & McAuley, 2018) models the user's interaction sequence with causal self-attention, allowing each position to selectively attend to previous items. For next-item prediction, the hidden representation at the current position summarizes the relevant interaction history and is used to score candidate items. Following this formulation, we use the hidden representation at the final position as the user's query vector. Thus, regardless of how the attention distributes information across the history, the model ultimately represents the user with a single query vector.

HSTU. HSTU (Zhai et al., 2024) is designed for large-scale recommendation with long, high-cardinality, and non-stationary interaction sequences. It replaces the standard Transformer attention block with a recommendation-oriented architecture that uses a combined projection for the attention components, a SiLU-based attention mechanism, relative position and time biases, and a gated output transformation. In our setting, we use the resulting representation at the final position as a single query vector, making HSTU a single-interest baseline.

MIND. MIND (Li et al., 2019) explicitly models a user as a collection of interest vectors rather than a single representation. Its multi-interest extractor uses capsule dynamic routing to group historical behaviors into different interest representations, where each behavior can contribute to multiple interests and the routing weights are iteratively refined based on their agreement with the interest representations. During training, MIND further uses label-aware attention to identify the interest most relevant to the target item and construct the training representation. At inference, the resulting interest vectors serve as independent retrieval queries, and an item is scored by its best match among the user's interests. MIND also introduces a dynamic mechanism for adjusting the number of interest vectors according to the user's history length.

ComiRec. ComiRec (Cen et al., 2020) also represents each user with multiple interest vectors, but introduces a controllable multi-interest framework with two alternative interest extractors. ComiRec-SA uses self-attentive aggregation to extract multiple interest vectors from the behavior sequence, while ComiRec-DR uses the capsule dynamic-routing mechanism adopted by MIND. For training, ComiRec uses hard assignment: the target item is associated with the interest vector having the highest relevance to the target, so the training signal is concentrated on the most relevant interest. At inference, each interest vector can independently retrieve candidate items, and the scores from the multiple interests can be aggregated to control the trade-off between recommendation accuracy and diversity. In our experiments, we use the multi-interest vectors directly for retrieval and score an item by its best match across the interests.

UIH k-means. The analysis in Section 2.2 shows that placing queries using the observed interaction history can suffer from sampling and drift errors. We therefore include UIH k-means as a natural, training-free baseline for history-based query placement. For each user, we apply k-means ($k = 3$) to the embeddings of their observed interaction history and use the resulting m unit cluster centers as query vectors. An item is scored by its maximum similarity to these centers, following the same max-over-interests scoring used by the learned multiple-interest models. Because UIH k-means introduces no trainable parameters, it provides a direct reference for how well clustering the observed history can place multiple queries without learning a dedicated multi-interest encoder.

Baseline implementation. We run all baselines ourselves and tune their hyperparameters following the recommendations in the original papers.

For HSTU and SASRec, we use the open-source `generative-recommenders` library[4]. The two models share the same training code and differ only in the sequence encoder, selected through the `encoder_type` switch. Thus, the item-embedding table, dot-product similarity module, learnable positional pre-processing, ℓ_2-normalized output layer, sampled-softmax objective, and evaluation code are identical. HSTU uses two blocks with one attention head and $d_{qk} = d_v = 50$. SASRec uses `nn.MultiheadAttention`, which splits the model width across heads, together with a position-wise feed-forward layer of width d and ReLU activation. Its attention-specific hyperparameters d_{qk}, d_v, and attention dropout are therefore not used.

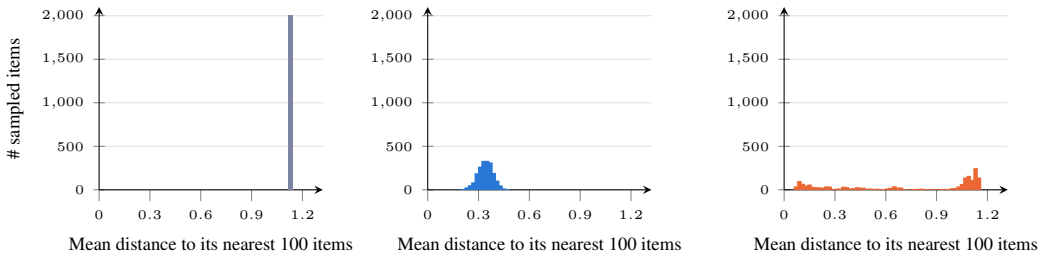
MIND and ComiRec are implemented in PyTorch using the same training and evaluation framework. They use the same interaction data, leave-one-out split, item pool, sampled-softmax objective, and evaluation metrics as HSTU and SASRec, so the main difference is the user encoder. For all multi-interest baselines, we use $K = 3$ interest vectors and three dynamic-routing iterations, following the original papers. The interest vectors are ℓ_2-normalized, and the user-item score is the maximum inner product over the K interests.

F More Details on Empirical Study

F.1 Local Item Density

As noted in Footnote 2, global uniformity of item embeddings over the sphere is a sufficient condition for using spherical area as a proxy for the number of items returned by a cap, but it is stronger than necessary. We only require the item density to be reasonably stable across the regions where items actually occur. The reason is as follows.

[4] https://github.com/meta-recsys/generative-recommenders.



(a) Synthetic uniform point set (b) Real item embeddings (c) Synthetic non-uniform point set

Figure 3: Distribution of local item density, measured by the mean distance from each sampled item to its 100 nearest neighbors. We randomly sample 2,000 items from each catalog, with 11.4 million points in 128 dimensions. The uniform catalog has a tightly concentrated distribution, indicating nearly constant local density. The highly non-uniform catalog has a much broader distribution due to substantial density variation. The real item space is much closer to the uniform case, suggesting that item density is reasonably stable across the regions where items occur.

Every comparison in Section 2.1 is a ratio. Suppose the density stays within $[\rho_{\min}, \rho_{\max}]$ wherever items lie, and write $\kappa = \rho_{\max}/\rho_{\min}$. A cap of radius θ then holds between $\rho_{\min} A(\theta)$ and $\rho_{\max} A(\theta)$ items, so the ratio of the item counts of any two caps agrees with the ratio of their areas to within a factor of κ. Carrying κ through Theorem 1 would multiply its bound by that constant and change nothing else, since κ depends on neither m nor d while the geometric factor $(\sin\theta_m^{\mathcal{T}}/\sin\theta_1^{\mathcal{T}})^{d-1}$ falls exponentially in the dimension. We therefore leave it out of the statements. What the argument does need is that κ be a moderate constant, and that is what we measure below.

To examine whether this weaker condition holds in practice, we conduct an empirical study of local item density in the internal short-video dataset. We randomly sample 2,000 items and, for each sampled item, find its 100 nearest neighbors and compute their average distance to the sampled item. A smaller average distance indicates a denser local neighborhood, while a larger distance indicates a sparser one. We plot the distribution of these average distances in Figure 3. As references, we construct two synthetic point sets with the same number of points and the same embedding dimension. The first distributes points uniformly over the unit sphere, representing an ideally homogeneous item space. The second is designed to have highly heterogeneous density: we place points into 1,000 clusters, with cluster sizes following a power-law distribution and cluster widths spanning two orders of magnitude. This creates both densely packed cluster cores and much sparser halos around them, providing a reference for a strongly non-uniform item space.

The uniformly distributed points exhibit a tightly concentrated distribution of local distances, with a 90th-to-10th percentile ratio of 1.01, while the highly non-uniform reference has a much broader distribution with a ratio of 9.60. The real item embeddings lie between the two, but are substantially closer to the uniform reference, with a ratio of 1.41. Thus, although the item embeddings are not globally uniform, their local density varies within a relatively narrow range across the regions where items actually occur. This supports using spherical area as a reasonable proxy for the number of items covered by a retrieval region.

F.2 Multiple Interests and Interest Drift

Two properties of a user's history are important for our analysis: whether the items form local groups, and whether these groups change over time. Figure 4 illustrates both properties. Each panel shows one randomly sampled user's history (embeddings of engaged items) projected into two dimensions using t-SNE from the internal short-video dataset. The leftmost panel shows the same number of points distributed evenly over the sphere as a reference without meaningful structure. For the four user histories, gray points represent the middle 80% of the history, while blue and orange points represent the earliest and latest 10%, respectively, which are about 25 days apart.

User histories form groups. The reference forms a relatively uniform cloud, with no clear gaps or dense regions. In contrast, all four user histories contain visible groups of nearby items separated

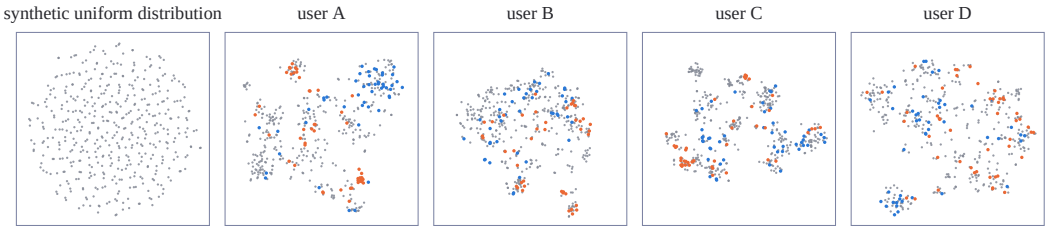


Figure 4: Examples of user histories in the item embedding space. Each panel shows 460 items from one user's history after projecting item embeddings with t-SNE in the internal short-video dataset. The leftmost panel is a uniformly distributed reference with the same number of points. In the user histories, gray denotes the middle 80% in time, while blue and orange denote the earliest and latest 10%, respectively. These two portions are about 25 days apart.

by sparser regions. These groups vary substantially in size and compactness: some contain many closely related videos, while others contain only a few loosely related ones. The boundaries between groups are also gradual rather than sharply separated, and larger groups often contain smaller subgroups. Thus, user histories are neither a single homogeneous cluster nor a small number of cleanly separated clusters. This structure creates an opportunity for multiple retrieval caps: several small caps can cover different regions of the history, whereas a single cap would need to cover the gaps between them as well.

User interests change over time. The blue and orange points also occupy noticeably different regions of the embedding space. Across all four examples, the content represented in the earliest 10% of the history is largely different from that in the latest 10%, even though the two ends are separated by less than a month. The content of these changes is also easy to interpret. For example, one user moves from American football, bodybuilding, and video games toward music and performing arts; another moves from country music and cinema toward politics, health, and alternative medicine; and a third moves from celebrity culture toward career development and employment. A query center inferred from the past history can therefore deviate from the center of the user's future interest, making it difficult to cover future interactions effectively. This prediction error arises from both the finite sampling of past interactions and the drift of user interests over time, as formalized in Theorem 2.

G More Experimental Results

G.1 Model Capacity

Here we provide a more detailed analysis of the capacity ablation. The complete results are shown in Table 5, including retrieval cost and the additional dropout ablation.

The retrieval cost continues to decrease beyond the point where accuracy saturates. Among the dropout-0.3 configurations, #scan decreases from 597 to 252 items as the model grows from 0.13M to 6.29M parameters, reducing the cost per recovered target from 714 to 333. The effect is particularly clear from 4.20M to 6.29M: #scan drops from 377 to 252 despite a slight decrease in ranking accuracy. Thus, additional capacity can make the predicted caps more compact even after it no longer improves ranking quality.

A smaller #scan, however, does not necessarily imply better retrieval. The 6.29M model has the lowest #scan but also the highest miss rate, 24.4%, compared with 18.6% for the default model. Its caps therefore become sufficiently small to exclude more targets, limiting the benefit of the reduced candidate set. This illustrates the trade-off between retrieval cost and coverage.

We also compare two dropout settings at the same 0.52M capacity. Increasing dropout from 0.1 to 0.3 improves NDCG@10 from 14.6 to 15.0 while reducing #scan from 470 to 392. The corresponding reduction in #scan suggests that better regularization can simultaneously improve ranking quality and produce more compact retrieval regions. Overall, these results show that model capacity affects not only ranking quality but also the geometry and serving cost of the learned retrieval regions.

$L \times H$	Params	Drop	HR@10	HR@200	NDCG@10	NDCG@200	#Scan	Miss	#Scan / Hit
2×64	0.13M	0.3	21.3	67.4	11.7	20.4	597	**16.4%**	714
1×128	0.26M	0.3	21.6	66.5	11.8	20.2	531	19.1%	656
2×128	0.52M	0.3	26.6	71.4	15.0	23.6	392	19.0%	484
2×128	0.52M	0.1	26.5	70.8	14.6	23.2	470	17.4%	569
3×192	1.77M	0.3	29.7	73.5	16.9	25.4	410	17.4%	496
4×256	4.20M	0.3	**31.0**	**75.6**	**17.6**	**26.2**	377	18.6%	463
6×256	6.29M	0.3	30.5	70.4	17.4	25.3	**252**	24.4%	**333**

Table 5: Detailed model capacity ablation on MovieLens-1M. Ranking is performed over the 3,883-item corpus. *#Scan* is the mean number of items in the union of the retrieval caps, *Miss* is the fraction of held-out targets outside this union, and *#Scan / Hit* is defined as $\#\mathrm{scan}/(1 - \mathrm{miss})$, measuring the mean number of scanned items per covered target.

m	Fair share	Min	Median	Max	Median / Fair
2	0.500	0.455	0.500	0.545	1.00
3	0.333	0.210	0.353	0.437	1.06
4	0.250	0.140	0.251	0.359	1.00
6	0.167	0.056	0.176	0.236	1.05
8	0.125	0.036	0.116	0.210	0.93
16	0.063	0.014	0.038	0.232	0.60

Table 6: Distribution of training targets across retrieval caps on MovieLens-20M. Each row corresponds to one setting of m, and *Min*, *Median*, and *Max* report the smallest, median, and largest assignment shares among its m caps. The assignments are measured during training, directly reflecting how the supervision signal is distributed across caps.

G.2 The Number of Query Heads

We further analyze how the number of retrieval caps affects both training dynamics and the geometry of the learned retrieval regions. Table 6 reports the assignment statistics for different numbers of caps on MovieLens-20M. For each training target, we assign it to the cap with the highest retrieval score and measure the fraction of targets assigned to each cap. With a well-balanced set of caps, the expected share for each cap is approximately $1/m$.

For $m \leq 6$, the assignment distribution remains reasonably balanced, whereas $m = 16$ exhibits substantially stronger imbalance. In particular, the median cap receives only 0.60 times the ideal share, indicating that under-utilized caps become common rather than being confined to a small number of outliers. This imbalance can create a self-reinforcing training dynamic: caps that receive fewer assignments obtain weaker gradients, making them less likely to attract future targets. The $m = 16$ model is also not fully converged, so the observed degradation may reflect both cap starvation and the additional optimization difficulty of learning many caps.

We also examine the geometry of the learned caps. As illustrated in the case study in Section 4.5, a user's interests do not necessarily form well-separated clusters in the item embedding space. Different interests can overlap and lie close to one another, with ambiguous boundaries between them. Consistent with this observation, the learned caps tend to overlap rather than form disjoint regions. Thus, multiple caps need not correspond one-to-one to distinct semantic interests. Instead, they provide additional degrees of freedom to represent different components of the user's future interaction distribution. As m increases, these additional caps allow the model to cover such regions with greater geometric flexibility.

Overall, increasing m provides greater geometric flexibility, but also makes the caps harder to train because the available supervision must be distributed across more queries. The results suggest that a moderate number of caps provides a useful balance between expressive retrieval geometry and stable optimization.

H Limitations and Future Work

Our framework also leaves several directions open for further investigation.

Geometric modeling assumptions. We use spherical caps and spherical area to obtain a tractable geometric formulation and analysis. In practice, item embedding spaces may have non-uniform local density, and user interests may not form perfectly spherical regions. These assumptions simplify the theory but are not essential to the practical effectiveness of the proposed method: our empirical study finds reasonably stable local density, and the learned caps provide effective retrieval regions across datasets. A natural direction for future work is to develop a more general geometric formulation that can account for heterogeneous density or more flexible region shapes.

Practical retrieval settings. Our theoretical analysis considers full-recall covering, while practical systems typically target a fixed recall level and rely on approximate nearest-neighbor indexes. In our experiments, we approximate cap retrieval by using the predicted centers as ANN queries and re-ranking the retrieved candidates. Future work could more directly connect the geometric objective to target-recall optimization and the serving cost of specific ANN systems.

Adaptive number of retrieval regions. We use a fixed number of retrieval regions for simplicity, although users may differ in the complexity and diversity of their future interests. Our experiments suggest that a moderate number of caps provides a useful balance between retrieval flexibility and optimization stability. An adaptive mechanism that predicts the number of regions from the user's history could further improve this trade-off.

More broadly, our formulation provides a geometric interface between user modeling and retrieval. Future work could explore richer region parameterizations and more explicit modeling of temporal interest drift.