

SetMIR: Multi-Interest Retrieval as Set Prediction

Xiaodong Liu
Snap Inc.
Bellevue, WA, USA
xliu9@snapchat.com

Congfei Zhang
Snap Inc.
Bellevue, WA, USA
czhang3@snapchat.com

Hsiang-wei Chao
Snap Inc.
Seattle, WA, USA
hchao@snapchat.com

Siman Wang
Snap Inc.
Bellevue, WA, USA
swang7@snapchat.com

Xiao Bai
Snap Inc.
Palo Alto, CA, USA
xbai@snapchat.com

Tong Zhao
Snap Inc.
Bellevue, WA, USA
tzhao@snapchat.com

Jingxiao Ma
Snap Inc.
Bellevue, WA, USA
jma3@snapchat.com

Wen Zhang
Snap Inc.
Bellevue, WA, USA
wzhang7@snapchat.com

Zhe Liu
Snap Inc.
Palo Alto, CA, USA
zliu11@snapchat.com

Shantanu Aggarwal
Snap Inc.
Palo Alto, CA, USA
saggarwal@snapchat.com

Di Huang
Snap Inc.
Santa Monica, CA, USA
dhuang6@snapchat.com

William Leach
Snap Inc.
Seattle, WA, USA
wleach@snapchat.com

Yunzhi Zhou
Snap Inc.
Palo Alto, CA, USA
yzhou10@snapchat.com

Yajun Wang
Snap Inc.
Palo Alto, CA, USA
ywang30@snapchat.com

Jinchao Li
Snap Inc.
Bellevue, WA, USA
jli18@snapchat.com

Yu Zhang
Snap Inc.
Palo Alto, CA, USA
yzhang3@snapchat.com

Abstract

Embedding-based retrieval is at the core of industrial recommender systems, but a single user embedding is often too limited to capture a user’s diverse interests. Multi-interest retrieval addresses this by using multiple user embeddings, yet existing methods still suffer from two issues: *interest collapse*, where different embeddings learn the same interest, and *static dispatch*, where serving uses a fixed retrieval budget even when some embeddings are unnecessary. We propose **SetMIR**, which treats multi-interest retrieval as a set prediction problem. SetMIR encodes a user’s behavior history with a transformer and uses K learnable queries to decode a set of user interests, each producing a retrieval embedding and a presence score. During training, Hungarian matching assigns targets to queries one-to-one, so matched queries learn distinct interests and the presence head learns which queries are active. At serving time, SetMIR uses presence scores and query-level Non-Maximum Suppression (NMS) to issue only active, non-redundant ANN queries. On Snap’s Dynamic Product Ads (DPA) data, SetMIR outperforms four learned multi-interest retrievers on every metric while issuing 33% fewer ANN queries per request. Deployed as a new retrieval source in the DPA production stack, SetMIR lifts overall CVR by 3.1%, while lifting CTR by 44% and CVR by 51% over the item-to-item retrieval source with the same item embeddings, ANN index, and retrieval quota.

CCS Concepts

• **Information systems** → **Retrieval models and ranking; Recommender systems.**

Keywords

Multi-interest retrieval, set prediction, recommender systems

ACM Reference Format:

Xiaodong Liu, Congfei Zhang, Hsiang-wei Chao, Siman Wang, Xiao Bai, Tong Zhao, Jingxiao Ma, Wen Zhang, Zhe Liu, Shantanu Aggarwal, Di Huang, William Leach, Yunzhi Zhou, Yajun Wang, Jinchao Li, and Yu Zhang. 2027. SetMIR: Multi-Interest Retrieval as Set Prediction. In *Proceedings of The 33rd ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD ’27, ADS Track, Under Review)*. ACM, New York, NY, USA, 9 pages.

1 Introduction

The retrieval stage of a large-scale recommender selects a few thousand candidate items per request from a corpus of millions to billions; a downstream ranker then scores those candidates with a heavier model [3, 6]. The dominant retrieval architecture is a two-tower neural network: a user tower produces one embedding, an item tower indexes the catalog, and approximate-nearest-neighbor (ANN) search [5, 7, 10, 27] retrieves the top- N items by inner product or cosine similarity. A single user embedding is often too coarse for this task. A user may interact with streetwear, baby clothes, and skincare in the same week, yet an averaged representation favors dominant categories and misses narrower, short-lived ones.

Multi-interest retrieval addresses this limitation by producing K embeddings per user, each issuing an ANN query whose results are merged [2, 4, 14]. However, current methods usually have two drawbacks. First, **interest collapse**: the standard training signal [14] assigns each target to its closest interest by $\arg \max_k \langle e_k, t \rangle$; multiple targets can collide on the same query, unmatched queries receive no gradient, and the model can converge to only a few effective interests. Second, **static dispatch**: issuing ANN searches from *all* K embeddings spreads a per-request retrieval budget N across redundant or low-quality queries, and without a learned per-query quality signal, K ANN calls may return fewer than K meaningfully distinct candidate sets while still consuming the full budget. Recent methods add auxiliary regularizers [25, 28], but they

do not learn which of the K embeddings should retrieve candidates for a particular user request.

We propose SetMIR, a set-based multi-interest retrieval framework that predicts a variable-size set from a fixed-size bank of learnable queries: the target-set size is unknown, and the queries must coordinate so each captures a distinct element. A Hungarian matcher [12] assigns engaged targets to queries one-to-one, so each matched query receives a retrieval gradient while unmatched queries receive an explicit absence target. A per-query *presence* score is supervised by the matcher (matched vs. unmatched) and gates which of the K queries dispatch ANN searches at inference time. Our contributions are summarized as follows:

- We frame multi-interest retrieval as set prediction over K learnable queries, which removes interest collapse at the training-objective level rather than through the auxiliary regularizers prior work relies on.
- We replace static dispatch with adaptive retrieval at serving time. Presence gating, followed by query-level NMS, activates $\hat{K} \leq K$ queries per request while keeping performance on par with all- K retrieval.
- Extensive offline experiments against learned multi-interest baselines, and an online A/B test in Snap DPA, demonstrate the effectiveness of SetMIR in a production recommender.

2 Related Work

Multi-interest retrieval. Multi-interest recommenders represent each user with multiple embeddings to capture the diverse interests reflected in their behavior history. Existing methods differ mainly in how these embeddings are constructed. MIND [14] uses capsule routing [20]; ComiRec uses multi-head self-attention [2]; SINE activates a sparse subset of global prototypes [22]; MVKE introduces virtual-kernel experts [26]; DCM performs differentiable clustering [4]; KuaiFormer appends learnable query tokens to a transformer over the behavior sequence [17]; and PinnerSage derives interests from offline clustering of user behaviors [19]. Subsequent work further improves these models through hard-negative mining and contrastive learning objectives [25, 28]. We refer readers to Li et al. [16] for a broader survey of multi-interest recommendation.

Despite these advances, two challenges remain in production retrieval. First, many methods use argmax-based target-to-query assignment [2, 14]. Multiple targets may then be assigned to the same query, while unmatched queries receive no training signal, leading to **interest collapse**. Second, most multi-interest retrievers use **static dispatch**, issuing a fixed number of ANN queries for every request regardless of how many user interests are active. Both persist in the deployed systems closest to SetMIR. KuaiFormer [17] at Kuaishou is the nearest in architecture, a transformer over the behavior sequence with learnable query tokens, and shows that this design is practical at production scale; SetMIR adopts the same family but replaces its argmax assignment and fixed ANN budget. DCM [4] reduces interest collapse through single-assignment clustering but still relies on static dispatch. SetMIR addresses both challenges by formulating multi-interest retrieval as set prediction. Hungarian matching provides one-to-one target assignment during

training, while a per-query presence head provides explicit no-interest supervision and dynamically gates ANN queries at serving time.

Set prediction, multi-vector retrieval, and selection. Set prediction produces a variable-size output set from a fixed bank of learnable queries. DETR [1] popularized this formulation for object detection; Set Transformer [13], Perceiver IO [8], and Q-Former [15] extend the learnable-query paradigm to other domains. SetMIR adapts this paradigm to retrieval-stage recommendation. Specifically, SetMIR inherits DETR’s learnable query bank, Hungarian assignment, and explicit *absence* target for unmatched queries, but applies them to a different input and a different prediction target. DETR predicts boxes in a bounded continuous space, supervised by exhaustive annotation. Here the prediction is a vector in a frozen item-embedding space, supervised contrastively, and the ground truth is a user’s engagement in a future window, so an unmatched query indicates only that no further interest appears in that window. The presence score also acts at a different point: in detection every query is decoded and the absence class filters the output, whereas here it determines how many of the K ANN searches are issued.

Unlike ColBERT-style multi-vector retrieval [11, 21], where each *document* is represented by many vectors with late interaction at scoring time, SetMIR keeps every *item* as a single vector and places multiplicity only on the *user* side, so existing single-vector ANN indices [5, 10] are reused without modification. Compared to differentiable top- M selection (e.g., Gumbel-Softmax [9]), the presence head supports a variable number of active queries per request and avoids the temperature schedule and straight-through estimator that top- M sampling requires.

3 Method

3.1 Problem statement

We frame multi-interest retrieval as set prediction over K learnable queries. Given a user’s chronological engagement history $\mathcal{H}_u = [v_1, v_2, \dots, v_L]$, where each v_i pairs an engaged product with an engagement type, the task is to recover the user’s multiple interests. The most recent high-intent products (e.g., purchase, add-to-cart, swipe-up) form the target set $\mathcal{T}_u = \{t_1, \dots, t_W\} \subset \mathcal{V}$, where $\mathcal{V}$ is the item universe. Earlier engaged products serve as model inputs. The number of targets W varies across users and is unknown at inference time.

The retrieval task is to return a top- N candidate list $C_u \subseteq \mathcal{V}$ with high recall against $\mathcal{T}_u$ under a serving budget: at most K interest embeddings may query the item index per request. SetMIR therefore predicts a set of interest representations $\mathcal{P}_u = \{(e_k, p_k)\}_{k=1}^K$, where $e_k \in \mathbb{R}^d$ is an interest embedding and $p_k \in [0, 1]$ is a presence score indicating whether query k corresponds to a current user interest.

3.2 Architecture

Figure 1 illustrates the overall framework: an input embedder, a history encoder, and an interest decoder with two per-query output heads, detailed in the following paragraphs.

Input embedder. Each history product v_i is embedded by a frozen pretrained content-embedding model from its metadata (title, brand, category, image) into a content embedding $\bar{x}_i \in \mathbb{R}^d$. We add a

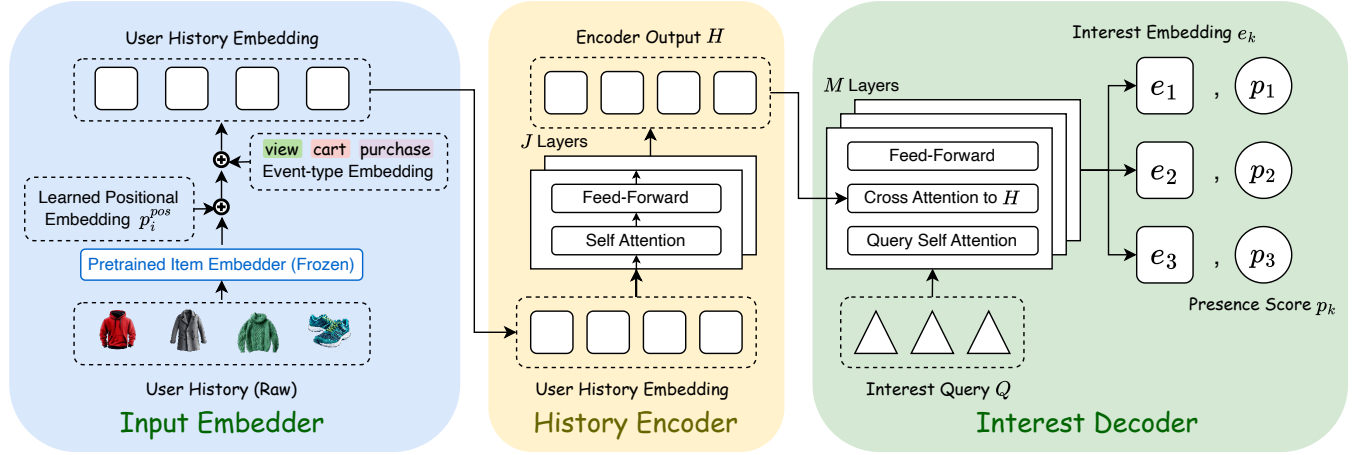


Figure 1: SetMIR pipeline. A transformer encoder reads the user history, a K -query transformer decoder emits per-query retrieval embeddings and presence scores, and training uses Hungarian matching to align engaged targets with queries. At inference, presence gating and query-level NMS select the active ANN queries, whose results are merged by maximum score.

learned positional embedding p_i^{pos} and an α -gated event-type embedding $\alpha c(\text{type}_i) \in \mathbb{R}^d$ to form the per-item input. The event-type embeddings are learned from scratch, and the scalar α is initialized at 0 and learned directly, so the model starts from the pure content-embedding space and gradually incorporates engagement signals:

$$x_i = \bar{x}_i + p_i^{\text{pos}} + \alpha c(\text{type}_i) \in \mathbb{R}^d. \quad (1)$$

History encoder. A standard J -layer transformer encoder $\mathcal{E}_\theta$ [24] processes the history of per-item inputs $[x_1, \dots, x_L]$ into contextualized representations:

$$H = \mathcal{E}_\theta([x_1, \dots, x_L]) \in \mathbb{R}^{L \times d}. \quad (2)$$

Interest decoder with K learnable queries. SetMIR maintains K learnable interest queries, shared across users, $Q^{(0)} = [q_1, \dots, q_K] \in \mathbb{R}^{K \times d}$, randomly initialized and trained jointly with the rest of the model. An M -layer transformer decoder $\mathcal{D}_\phi$ alternates between self-attention among queries and cross-attention to the encoded history. Omitting residual connections and layer normalization for compactness, one decoder layer is

$$\tilde{Q}^{(\ell)} = \text{SelfAttn}(Q^{(\ell-1)}), \quad (3)$$

$$\bar{Q}^{(\ell)} = \text{CrossAttn}(\tilde{Q}^{(\ell)}, H, H), \quad (4)$$

$$Q^{(\ell)} = \text{FFN}(\bar{Q}^{(\ell)}), \quad (5)$$

for $\ell = 1, \dots, M$. Query self-attention lets interest queries exchange information before reading the history, which encourages different queries to specialize rather than independently chase the same dominant signal. The decoder output $Z = Q^{(M)} \in \mathbb{R}^{K \times d}$ is passed to two per-query heads.

Two output heads per query. For each decoder-output row z_k of $Z = Q^{(M)}$, SetMIR predicts a normalized retrieval embedding e_k and a presence score p_k :

$$e_k = \frac{W_e \text{LN}(z_k) + b_e}{\|W_e \text{LN}(z_k) + b_e\|_2} \in \mathbb{R}^d, \quad (6)$$

$$p_k = \sigma(w_p^\top \text{LN}(z_k) + b_p) \in [0, 1], \quad (7)$$

where $\sigma(\cdot)$ is the logistic sigmoid. The L_2 normalization makes training and serving consistent. The scalar presence head estimates whether the query is active for this request.

3.3 Set-prediction training

For a training example, target product identifiers are deduplicated by product id (pid). If the same pid appears with multiple event types, we keep the strongest-intent event type (priority order: purchase > add-to-cart > swipe-up). The target products are then capped at $W_{\max} = 15$ by strongest-intent priority. The model produces K interest embeddings $\{e_k\}_{k=1}^K$, and the same frozen pre-trained model from §3.2 encodes each target product t_w into a target embedding $\bar{t}_w \in \mathbb{R}^d$, giving the target set $\{\bar{t}_w\}_{w=1}^W$ where $W \leq W_{\max}$ and may exceed K .

Hungarian matching. We compute a one-to-one assignment between predicted queries and target items. Let $\mathcal{A}_{K,W}$ be the set of matchings $A \subseteq [K] \times [W]$ with $|A| = \min(K, W)$ and no repeated query or target. We solve

$$\hat{A} = \arg \min_{A \in \mathcal{A}_{K,W}} \sum_{(k,w) \in A} \Gamma_{k,w}, \quad (8)$$

$$\Gamma_{k,w} = -\lambda_{\text{emb}} e_k^\top \bar{t}_w - \lambda_{\text{cls}} \log \sigma(\ell_k),$$

where ℓ_k is the pre-sigmoid presence logit. The embedding term prefers geometrically close query-target pairs; the class term biases the matcher toward queries the model already predicts as present. We use $\lambda_{\text{emb}} = 1$ and $\lambda_{\text{cls}} = 0.5$, with the class term linearly warmed up over the first 10% of training steps so that early matching is driven primarily by embedding geometry. For $K = 10$, each user's assignment is a small $K \times W$ problem and the solve itself is negligible. Let $\mathcal{M}_u = \{k : (k, w) \in \hat{A}\}$ denote the matched query indices, with $|\mathcal{M}_u| = \min(W, K)$, and let $\mathcal{U}_u = [K] \setminus \mathcal{M}_u$ denote unmatched ("no-interest") queries.

InfoNCE retrieval loss. For each matched pair $(k, w) \in \hat{A}$, we apply an in-batch sampled-softmax (InfoNCE [23]) loss. The assigned target $\bar{t}_w$ is the positive, and all valid target embeddings gathered

across the data-parallel mini-batch form the contrastive pool $\mathcal{N}$, which therefore contains $\bar{t}_w$ itself:

$$\mathcal{L}_{\text{InfoNCE}}(u) = -\frac{1}{|\hat{A}|} \sum_{(k,w) \in \hat{A}} \log \frac{\exp(e_k^\top \bar{t}_w / \tau_r)}{\sum_{\bar{t}' \in \mathcal{N}} \exp(e_k^\top \bar{t}' / \tau_r)}, \quad (9)$$

where τ_r is the InfoNCE temperature and padded target entries are masked out. Only matched queries participate in $\mathcal{L}_{\text{InfoNCE}}$; unmatched queries are not pulled toward any target direction.

Presence loss. The matcher identifies which queries are used on the example, and the presence head learns to predict that assignment. Given presence score p_k and binary label $y_k = 1[k \in \mathcal{M}_u]$,

$$\mathcal{L}_{\text{Pres}}(u) = -\frac{1}{K} \sum_{k=1}^K [y_k \log p_k + (1 - y_k) \log(1 - p_k)]. \quad (10)$$

The retrieval and presence losses provide complementary supervision. Matched queries are pulled toward their assigned targets, whereas unmatched queries receive only a $y_k=0$ presence target. This absence signal is the mechanism missing from argmax-trained multi-interest models: when targets independently choose their closest query, unmatched queries receive no direct supervision and can collapse or drift.

Margin diversity loss. We add a light pairwise repulsion term to discourage active queries from becoming nearly identical. Let $a_k = 1[p_k > 0.5]$ be a detached hard activity mask; gradients do not flow through the threshold. We define

$$\mathcal{L}_{\text{Div}}(u) = \frac{\sum_{i \neq j} a_i a_j \max(0, e_i^\top e_j - m)}{\max(1, \sum_{i \neq j} a_i a_j)}, \quad (11)$$

with margin $m = 0.3$. The loss is zero when fewer than two queries are active. Otherwise, it penalizes only active pairs whose cosine similarity exceeds m , leaving already separated queries unchanged. Unlike ComiRec’s controllability term [2], which is applied during inference, this term shapes the learned retrieval geometry during training.

Total loss. For a mini-batch of users $\mathcal{B}$, the total training loss is

$$\mathcal{L} = \frac{1}{|\mathcal{B}|} \sum_{u \in \mathcal{B}} [\lambda_R \mathcal{L}_{\text{InfoNCE}}(u) + \lambda_P \mathcal{L}_{\text{Pres}}(u) + \lambda_D \mathcal{L}_{\text{Div}}(u)], \quad (12)$$

with loss weights $\lambda_R, \lambda_P, \lambda_D$ specified in §4.1.

3.4 Inference

At serving time, SetMIR performs a single forward pass that encodes the user history and decodes the K interest queries, yielding $\{(e_k, p_k)\}_{k=1}^K$. It then converts the fixed query bank into a dynamic set of $\tilde{K} \leq K$ ANN queries through three lightweight steps that together replace static dispatch with dynamic $\tilde{K}$ retrieval (see Figure 2).

Step 1: Presence gating. We first retain queries whose presence exceeds a threshold τ :

$$\mathcal{S}_1 = \{k \in [K] : p_k > \tau\}. \quad (13)$$

If no query passes the threshold, we keep the query with the largest presence score as a fallback.

Step 2: Query-level non-maximum suppression. We sort $\mathcal{S}_1$ by descending p_k and apply NMS over query-query cosine similarity. A query is removed if its cosine similarity to any higher-presence

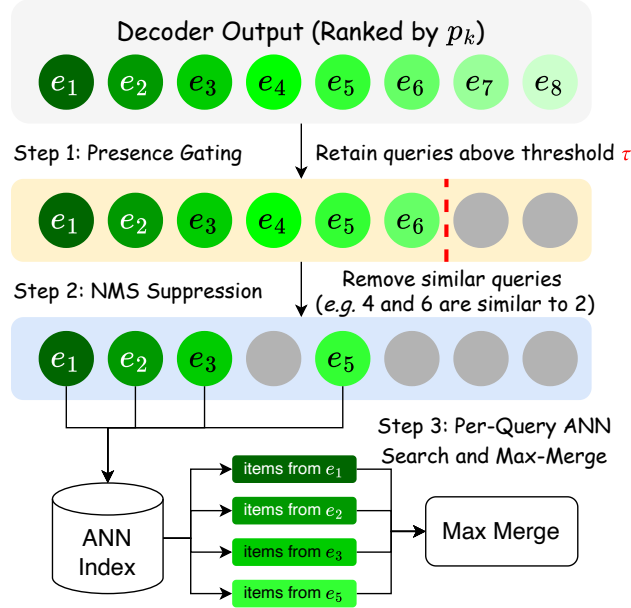


Figure 2: Inference-time gating and per-query ANN. Presence gating at threshold τ retains active queries, cosine-NMS at δ_{NMS} removes redundant survivors, each survivor issues one ANN search, and returned items are merged by max-score.

retained query exceeds δ_{NMS} ; we use $\delta_{\text{NMS}} = 0.9$. The surviving set is $\mathcal{S}_2 \subseteq \mathcal{S}_1$, with $|\mathcal{S}_2| = \tilde{K}$.

Step 3: Per-query ANN search and max-merge. Each surviving query $k \in \mathcal{S}_2$ retrieves its top- M_q items from the item index. We set $M_q = \lceil N/\tilde{K} \rceil$ so that the total retrieval depth remains approximately constant as $\tilde{K}$ changes. Items returned by multiple queries are merged by maximum score:

$$s(t | u) = \max_{k \in \mathcal{S}_2} e_k^\top \bar{t}, \quad (14)$$

and the top- N items by $s(t | u)$ form the candidate list.

4 Experiments

4.1 Experimental Setup

Dataset. We evaluate SetMIR on user-product engagement logs from Snap DPA. The advertiser catalog contains hundreds of millions of products, and the event stream includes views, swipe-ups, add-to-carts, and purchases. For each user, we form the target set $\mathcal{T}_u$ from all high-intent events (such as purchase, add-to-cart, and swipe-up) that fall within the last 3-day window. Targets are deduplicated by product id and capped at $W_{\text{max}}=15$ items, which retains all high-intent targets for most users in the training data. The history sequence is every event whose timestamp lies strictly before the start of the window, truncated to the most recent $L=40$ events. Users are partitioned into disjoint train, validation, and test sets. Item embeddings are computed offline by a separately pretrained content-embedding model, then frozen and stored as L_2 -normalized vectors throughout SetMIR training and evaluation.

Table 1: Comparison of SetMIR against four *learned* multi-interest retrieval methods. All rows share the same training data, the same frozen item embeddings, $K=10$, the same optimizer and batch size, and the same per-request retrieval budget; they differ only in the interest-extraction architecture and the target-to-query assignment. Baselines have no presence concept, so they dispatch all K queries ($\tilde{K}=K$), which is the static dispatch they practice in production. Every method is trained until it converges, each to its own point of convergence rather than to a shared step count, since the model classes differ substantially in how long they keep improving. The SetMIR (argmax) row replaces Hungarian matching with argmax assignment and is analyzed in §4.2.2. Each metric is normalized so that SetMIR equals 100%.

Method	$\tilde{K}$	R@1	R@5	R@10	R@100	HR@100	NDCG@100	MRR
MIND [14]	10.00	45.7%	50.7%	52.9%	61.7%	58.2%	54.3%	44.4%
ComiRec-SA [2]	10.00	67.4%	83.5%	89.8%	99.9%	99.6%	89.8%	80.0%
DCM [4]	10.00	26.0%	34.5%	37.4%	49.3%	48.8%	39.3%	29.1%
KuaiFormer [17]	10.00	43.0%	47.6%	52.0%	62.8%	64.1%	52.3%	42.2%
SetMIR (argmax)	2.16	62.1%	63.2%	63.5%	69.1%	73.2%	63.1%	56.4%
SetMIR	6.70	100%	100%	100%	100%	100%	100%	100%

Evaluation pool. All offline results are reported on a held-out *test* set of around 1M users, yielding a target pool of around 5.3M products. The validation split also includes around 1M users and is used to choose the presence threshold τ , the NMS threshold δ_{NMS} , and each method’s converged checkpoint, so no reported number informs those choices.

Metrics. We report R@ N (recall at top- N retrieved products) for $N \in \{1, 5, 10, 100\}$, HR@100 (hit rate), NDCG@100, and MRR (mean reciprocal rank). For serving analysis we additionally report $\tilde{K}$, the average number of ANN queries per request after presence gating and NMS. Per Snap’s policy, we report only relative metric lifts.

Implementation. The history encoder is a 2-layer transformer with 8 attention heads, feed-forward width $d_{\text{ff}} = 1024$, and dropout 0.1; the interest decoder has 6 layers with the same width, heads, and dropout, with hidden size $d = 128$ and approximately 3M learnable parameters.

We train with AdamW [18] (peak learning rate 5×10^{-4} , 4000 warm-up steps, weight decay 0.01, cosine decay) on 8 A100-40GB GPUs with batch size 1536 users per GPU and bfloat16 mixed precision. The in-batch InfoNCE pool gathers target embeddings across GPUs and masks padded target entries. The InfoNCE temperature is $\tau_r = 0.05$, the diversity margin is $m = 0.3$, and the loss weights are $\lambda_R = 1.0$, $\lambda_P = 0.5$, $\lambda_D = 0.1$, with the Hungarian class term $\lambda_{\text{cls}} = 0.5$ linearly warmed up over the first 10% of steps. Unless otherwise stated, inference uses presence threshold $\tau = 0.3$, NMS threshold $\delta_{\text{NMS}} = 0.9$, and $K=10$ queries.

4.2 Offline Experiments

4.2.1 Comparison with learned multi-interest baselines. We compare SetMIR against four learned multi-interest retrievers spanning the designs surveyed in §2: MIND [14], ComiRec-SA [2], DCM [4], and KuaiFormer [17] on the Snap DPA evaluation pool. Because SetMIR consumes a *frozen* pretrained item tower whereas all four learn item embeddings jointly with the user side, this is a controlled comparison of interest-extraction mechanisms rather than an end-to-end reproduction of published results. For the experiment comparison, ComiRec-SA and MIND are ported from the authors’ reference implementation¹; DCM and KuaiFormer have no public

code and are implemented from their papers. All four use the same output head as SetMIR, and MIND’s ReLU output layer is dropped because a frozen item tower would otherwise confine queries to the non-negative orthant. Each method is trained until it converges on the validation set, so the training length is per method rather than shared: MIND, ComiRec-SA and DCM peak within 60K steps, KuaiFormer at roughly 100K and SetMIR at 160K.

SetMIR is strongest on every metric, and the margins are widest at the top of the ranking, which is what the downstream ranker consumes: the best baseline, ComiRec-SA, reaches 67.4% of SetMIR’s R@1 and 80.0% of its MRR, yet becomes competitive at depth (99.9% of R@100), so attention-based routing recovers a similar candidate *pool* but orders it less well. SetMIR does this while issuing 6.70 ANN queries per request against the baselines’ 10, a 33% smaller budget.

MIND and DCM fall furthest behind, at 52.9% and 37.4% of SetMIR’s R@10. Both construct interests geometrically, by capsule routing and by differentiable clustering respectively, and both were designed to learn the item tower jointly with the user side; DCM additionally had to be reimplemented from its paper. We therefore read these rows as evidence about how those mechanisms behave under a frozen item space, not as a verdict on the published systems.

KuaiFormer is worth isolating: it is the closest architecture to SetMIR, a transformer over the history with learnable query tokens, yet reaches only 52.0% of SetMIR’s R@10. Both are deep transformers of comparable capacity on identical data and item embeddings, so the gap points to how the queries are supervised, which the SetMIR (argmax) row isolates directly (§4.2.2).

4.2.2 Component ablation. Table 2 removes one mechanism at a time: the target-to-query assignment, the presence loss, and the diversity loss. Replacing Hungarian matching with per-target argmax costs 36.5% of R@10 and 43.6% of MRR and drops the mean number of presence-active queries from 6.70 to 2.16 of 10: targets compete for whichever query is already closest, the rest never receive a retrieval gradient, and the model settles on roughly two effective interests, illustrating the impact of interest collapse.

Both loss terms also cost recall when removed: R@10 falls to 91.4% without the presence loss and 93.2% without diversity. Each term does distinct work. The presence loss is what makes gating

¹<https://github.com/THUDM/ComiRec>

Table 2: Component ablation. Each row removes exactly one mechanism; architecture, data and the remaining losses are held fixed, on the same converged schedule as Table 1. Rows are normalized to SetMIR and $\tilde{K}$ is absolute at $\tau=0.3$. Each mechanism is discussed in §4.2.2.

Variant	$\tilde{K}$	R@1	R@5	R@10	R@100	HR@100	NDCG@100	MRR
SetMIR (full)	6.70	100%	100%	100%	100%	100%	100%	100%
– presence loss	8.23	97.1%	90.4%	91.4%	91.3%	88.4%	92.9%	97.4%
– diversity loss	7.55	88.5%	90.6%	93.2%	91.8%	91.5%	93.3%	95.7%
argmax assignment	2.16	62.1%	63.2%	63.5%	69.1%	73.2%	63.1%	56.4%

Table 3: K sweep experiment. All rows share architecture, optimization, and training duration; only K varies. Each metric is normalized so that the single-query ($K=1$) configuration equals 100%. The second column reports $\tilde{K}$, the average number of presence-active queries.

K	$\tilde{K}$	R@1	R@5	R@10	R@100	HR@100	NDCG@100	MRR
1	1.00	100%	100%	100%	100%	100%	100%	100%
3	3.00	108.4%	133.5%	137.5%	137.2%	110.8%	130.7%	112.4%
5	4.72	113.2%	145.8%	151.1%	148.6%	113.3%	141.6%	117.2%
7	5.75	112.3%	147.6%	154.7%	151.9%	114.2%	144.2%	117.8%
10	6.70	113.0%	148.3%	155.0%	152.8%	114.1%	145.8%	118.9%
15	7.59	112.8%	148.6%	155.1%	152.6%	114.1%	145.9%	118.4%

Table 4: Ablation of the α -gated event-type embedding: percentage change from a no-event-type-embedding baseline at three training durations. All other architecture and training settings are held fixed.

Steps	$\Delta R@1$	$\Delta R@5$	$\Delta R@10$	$\Delta R@100$	$\Delta HR@100$	$\Delta NDCG@100$	ΔMRR
30K	+5.1%	+2.6%	+3.0%	+1.7%	+0.5%	+2.9%	+3.0%
60K	+7.6%	+5.6%	+4.0%	+1.8%	+0.5%	+4.3%	+5.3%
120K	+8.2%	+6.9%	+5.0%	+3.2%	+1.6%	+5.2%	+5.6%

learnable: left unsupervised, $\sigma(\ell_k)$ carries no information about whether a query is warranted, so thresholding at $\tau=0.3$ admits 8.23 queries rather than 6.70 and still loses recall, which is more calls for less return. The diversity term acts on query redundancy, and mean pairwise cosine among the K query embeddings rises from 0.298 to 0.321 when it is removed. Across the three, the assignment dominates: it costs 36.5% of R@10 against 8.6% and 6.8% for the presence and diversity terms, so one-to-one matching carries the retrieval quality while the two losses contribute smaller, separate effects.

4.2.3 K sweep. Having established that SetMIR improves on prior learned multi-interest retrievers, we now vary the number of decoder queries K while holding the rest of the configuration fixed (Table 3). The $K=1$ row serves as the single-query configuration.

Three patterns stand out. *First, increasing K produces large recall gains.* $K=10$ reaches 155% of the $K=1$ R@10 and 153% of the $K=1$ R@100. The largest jump is from $K=1$ to $K=3$, indicating that even a small number of additional interest queries captures most of the available gain. *Second, gains are larger at deeper retrieval depths.* The lift from $K=1$ to $K=10$ is wider on R@10/R@100 (+55%/+53%) than on R@1/HR@100 (+13%/+14%), consistent with multi-interest retrieval broadening the candidate surface rather than only improving the top hit. *Third, gains gradually saturate as K increases.* $K=7$ already reaches 99.8% of $K=10$'s R@10, and $K=15$ is within $\pm 0.5\%$

of $K=10$ across all columns. The $\tilde{K}$ column makes this explicit: $\tilde{K}$ grows sublinearly with K (4.72 at $K=5$, 6.70 at $K=10$, only 7.59 at $K=15$), so even when 15 queries are available the model commits to fewer than 8 on average; the effective number of interests is bounded by the data, not by K .

4.2.4 Event-type embedding. Table 4 isolates the α -gated event-type embedding of §3.2. The event signal helps consistently, and its effect grows with training duration. Because α is initialized at zero, the model first matches the no-event baseline and then gradually learns how much event-type information to add: the R@10 gain rises from +3.0% to +5.0% between the shortest and longest schedule. The gain is strongest near the top of the ranking ($\Delta R@1 = +8.2\%$ vs. $\Delta R@100 = +3.2\%$), which complements the K sweep, where the multi-interest gain is more visible at depth.

4.2.5 Data scaling. Table 5 sweeps the number of training users. Each metric is normalized so that the smallest scale (100K users) equals 100%. Performance scales smoothly with training data: 1M users lifts the R@100 to 1.66 $\times$ and HR@100 to 1.45 $\times$, while 10M users further improves the R@100 and HR@100 to 2.18 $\times$ and 1.69 $\times$, respectively.

4.2.6 Inference-time gating and dispatch reduction. Table 6 tests whether gating and NMS deliver the dynamic $\tilde{K} \leq K$ dispatch that replaces static all- K retrieval, serving one $K=10$ checkpoint under

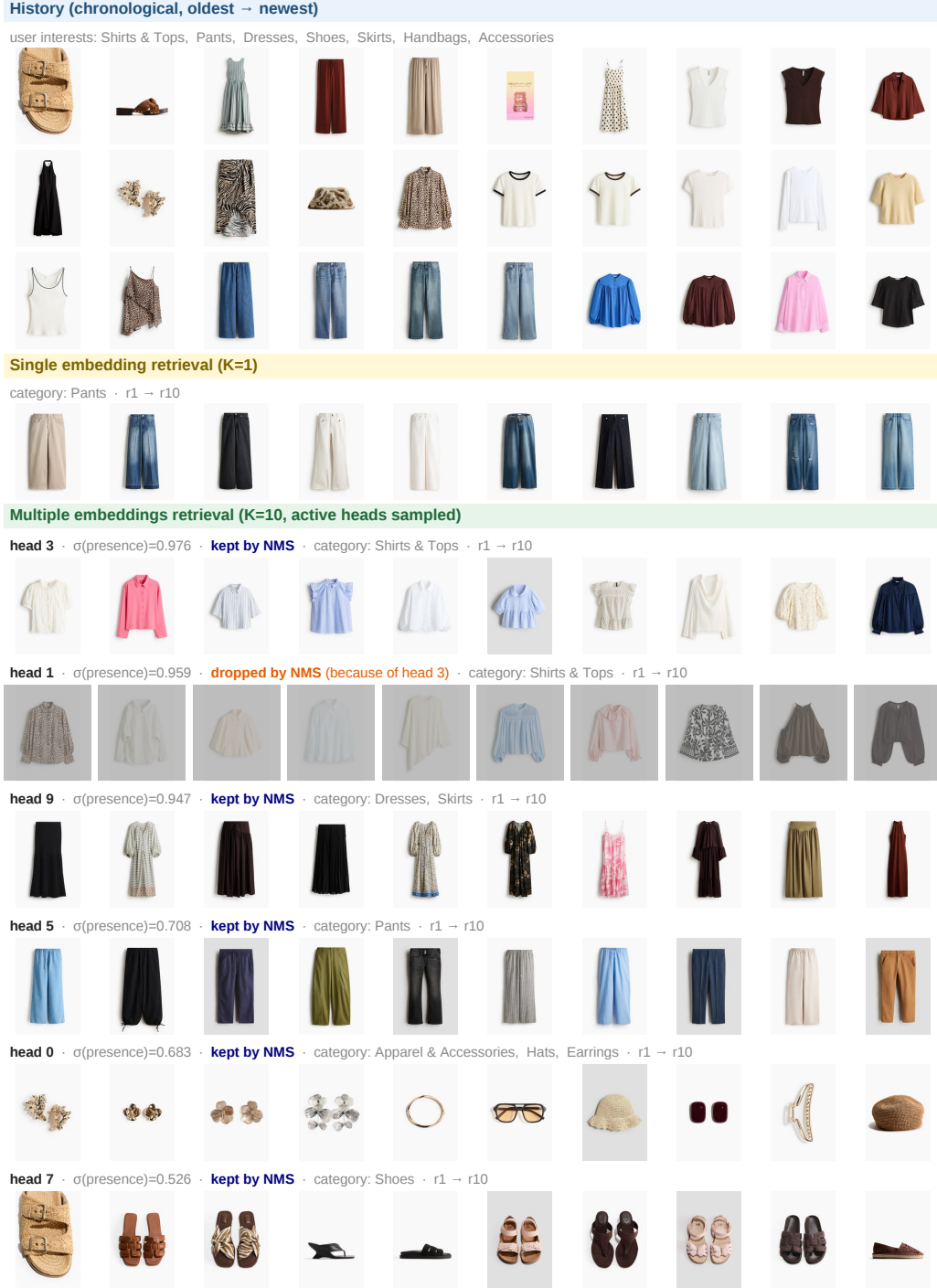


Figure 3: Qualitative retrieval comparison for one user. The figure labels “Single embedding ($K=1$)” and “Multiple embeddings ($K=10$)” correspond to the single-query baseline and SetMIR respectively; per-query rows are labeled “head k ”, each corresponding to one of SetMIR’s K interest queries with its $\sigma(\text{presence})$ score from the gating step. The user’s history spans shirts & tops, pants, dresses, shoes, skirts, handbags, and accessories. The single-query baseline collapses onto pants; SetMIR’s five active queries each specialize in a distinct category, while a near-duplicate query (head 1) is removed by NMS for overlapping with head 3.

Table 5: Training-data scaling sweep. All rows share architecture and training recipe, and each is trained to convergence; only the number of training users varies. Each metric is normalized so that the smallest scale (100K users) equals 100%.

Train users	R@1	R@5	R@10	R@100	HR@100	NDCG@100	MRR
100K	100%	100%	100%	100%	100%	100%	100%
1M	179.1%	181.2%	179.2%	166.9%	145.1%	175.1%	171.0%
10M	247.5%	262.5%	258.4%	218.5%	169.7%	243.3%	231.2%

Table 6: Serving ablation. One checkpoint served under three dispatch policies; no retraining. ANN calls is the mean number of per-request ANN queries actually issued, and metrics are normalized so that static all-K dispatch equals 100%. Presence gating removes 23% of the ANN calls at 99.4% of R@10 while improving R@1, and NMS at $\delta_{\text{NMS}}=0.90$ removes a further 1.03 calls at 99.2% of R@10, so the two together issue 6.70 of 10.

Dispatch policy	ANN calls	R@1	R@10	MRR
static all-K ($\tau=0$)	10.00	100%	100%	100%
+ presence gating ($\tau=0.3$)	7.73	101.9%	99.4%	100.7%
+ query-level NMS ($\delta=0.9$)	6.70	101.8%	99.2%	100.7%

three policies. Gating accounts for most of the reduction (10 $\rightarrow$ 7.73 calls of the 10 $\rightarrow$ 6.70 total), and it slightly *improves* R@1 and MRR, because suppressing low-confidence queries removes candidates that would otherwise dilute the merged list. The two knobs are complementary rather than interchangeable: gating asks whether the user has the interest at all, which the head is trained to answer, whereas NMS applies a purely geometric test to whichever queries survive, so each removes calls the other keeps. Loosening the gate below $\tau=0.30$ buys little (99.4% $\rightarrow$ 100% R@10 for 7.73 $\rightarrow$ 10 calls), so we deploy $\tau=0.30$ with $\delta_{\text{NMS}}=0.90$.

4.3 Online Experiment

We deployed SetMIR as a new retrieval source in the Snap DPA production stack and ran a live A/B test with the downstream ranker held fixed. SetMIR uses its default inference policy and its own ANN index built from the same pretrained content-embedding model. The experiment ran on a representative live-traffic slice for about one week; exact traffic fractions, sample sizes, and significance thresholds are omitted per Snap’s policy. We report two analyses from this single A/B: (a) the group-level lift of treatment over control, capturing SetMIR’s overall contribution to the production stack; and (b) a source-level comparison of SetMIR versus the embedding-based item-to-item (I2I) matching retrieval source running alongside it in the treatment arm.

(a) Overall contribution to the production stack. The control retains the current Snap DPA retrieval mix; the treatment adds SetMIR as a new retrieval source while keeping the global retrieval quota unchanged. Adding SetMIR lifts user impressions by +0.10%, CTR by +0.21%, and CVR by +3.11% (Table 7, column (a)).

(b) Per-source comparison vs item-to-item retrieval. Item-to-item (I2I) retrieval dispatches a user’s recent engaged high-intent products directly as ANN seeds, sharing SetMIR’s input signal,

Table 7: Online A/B results from a single live experiment. (a) Group-level engagement lifts of treatment (production stack + SetMIR) over control (production stack). (b) Source-level engagement attribution within the treatment arm, comparing SetMIR against the item-to-item (I2I) retrieval source running alongside it under matched per-source retrieval quota. Downstream ranker held fixed across both analyses.

Metric	(a) Δ vs control	(b) Δ SetMIR vs I2I
CTR	+0.21%	+44%
CVR	+3.11%	+51%

frozen item embeddings, ANN index, and per-source retrieval quota; the two differ only in how the query vectors are produced. Since I2I already dispatches several seeds per request, this is a comparison between two ways of choosing a small set of query vectors, and serving cost is comparable because both issue a few single-vector ANN lookups per request. Attributing engagement to each source within the treatment arm, SetMIR achieves +44% CTR and +51% CVR over I2I (Table 7, column (b)), isolating the value of the learned encoder-decoder, Hungarian matching, and presence head over the simpler heuristic of dispatching engaged history items directly.

4.4 Qualitative Analysis

Figure 3 visualizes the interest collapse failure mode on one user: although the user’s history spans shirts & tops, pants, dresses, shoes, skirts, handbags, and accessories, the single-query ($K=1$) baseline returns only pants. SetMIR’s five active queries each specialize in a distinct interest cluster, so the merged retrieval covers most of the user’s interest mix.

5 Conclusion

We presented SetMIR for multi-interest retrieval. The model uses K learnable interest queries, Hungarian matching against engaged targets, and a presence head supervised by the matcher. Together, these components address interest collapse at training time and static dispatch at serving time: presence gating and query-level NMS turn the fixed query bank into a request-dependent set of active ANN queries, all served by a single-vector item index. On Snap DPA, SetMIR outperforms four learned multi-interest retrievers on every metric while issuing 33% fewer ANN queries per request. Deployed as a new retrieval source in the production stack, SetMIR lifts CVR by 3.11% over the existing retrieval mix and delivers 44% higher CTR and 51% higher CVR than the item-to-item source running beside it.

References

- [1] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. End-to-End Object Detection with Transformers. In *Computer Vision – ECCV 2020 (Lecture Notes in Computer Science, Vol. 12346)*. Springer International Publishing, Cham, 213–229. arXiv:2005.12872 [cs.CV] doi:10.1007/978-3-030-58452-8_13
- [2] Yukuo Cen, Jianwei Zhang, Xu Zou, Chang Zhou, Hongxia Yang, and Jie Tang. 2020. Controllable Multi-Interest Framework for Recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '20)*. ACM, New York, NY, USA, 2942–2951. arXiv:2005.09347 [cs.IR] doi:10.1145/3394486.3403344
- [3] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16)*. ACM, New York, NY, USA, 191–198. doi:10.1145/2959100.2959190
- [4] Zhibo Fan, Hongtao Lin, Haoyu Chen, Bowen Deng, Hedi Xia, Yuke Yan, and James Li. 2025. Synergizing Implicit and Explicit User Interests: A Multi-Embedding Retrieval Framework at Pinterest. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '25)*. ACM, New York, NY, USA, 4396–4405. arXiv:2506.23060 [cs.IR] doi:10.1145/3711896.3737265
- [5] Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. 2020. Accelerating Large-Scale Inference with Anisotropic Vector Quantization. In *Proceedings of the 37th International Conference on Machine Learning (ICML) (Proceedings of Machine Learning Research, Vol. 119)*. PMLR, Virtual Event, 3887–3896. arXiv:1908.10396 [cs.LG]
- [6] Jui-Ting Huang, Ashish Sharma, Shuying Sun, Li Xia, David Zhang, Philip Pronin, Janani Padmanabhan, Giuseppe Ottaviano, and Linjun Yang. 2020. Embedding-based Retrieval in Facebook Search. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20)*. ACM, New York, NY, USA, 2553–2561. arXiv:2006.11632 [cs.IR] doi:10.1145/3394486.3403305
- [7] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry P. Heck. 2013. Learning Deep Structured Semantic Models for Web Search using Clickthrough Data. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management (CIKM '13)*. ACM, New York, NY, USA, 2333–2338. doi:10.1145/2505515.2505665
- [8] Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, Olivier J. Hénaff, Matthew M. Botvinick, Andrew Zisserman, Oriol Vinyals, and João Carreira. 2022. Perceiver IO: A General Architecture for Structured Inputs and Outputs. In *International Conference on Learning Representations (ICLR)*. OpenReview.net, Virtual Event, 29 pages. arXiv:2107.14795 [cs.LG]
- [9] Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical Reparameterization with Gumbel-Softmax. In *International Conference on Learning Representations (ICLR)*. OpenReview.net, Toulon, France, 12 pages. arXiv:1611.01144 [cs.LG]
- [10] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2021. Billion-Scale Similarity Search with GPUs. *IEEE Transactions on Big Data* 7, 3 (2021), 535–547. arXiv:1702.08734 [cs.CV] doi:10.1109/TBDDATA.2019.2921572
- [11] Omar Khattab and Matei Zaharia. 2020. 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 (SIGIR '20)*. ACM, New York, NY, USA, 39–48. arXiv:2004.12832 [cs.IR] doi:10.1145/3397271.3401075
- [12] Harold W. Kuhn. 1955. The Hungarian Method for the Assignment Problem. *Naval Research Logistics Quarterly* 2, 1–2 (1955), 83–97. doi:10.1002/nav.3800020109
- [13] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh. 2019. Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning (ICML) (Proceedings of Machine Learning Research, Vol. 97)*. PMLR, Long Beach, CA, USA, 3744–3753. arXiv:1810.00825 [cs.LG]
- [14] Chao Li, Zhiyuan Liu, Mengmeng Wu, Yuchi Xu, Pipei Huang, Huan Zhao, Guoliang Kang, Qiwei Chen, Wei Li, and Dik Lun Lee. 2019. Multi-Interest Network with Dynamic Routing for Recommendation at Tmall. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM '19)*. ACM, New York, NY, USA, 2615–2623. arXiv:1904.08030 [cs.IR] doi:10.1145/3357384.3357814
- [15] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023. BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models. In *Proceedings of the 40th International Conference on Machine Learning (ICML) (Proceedings of Machine Learning Research, Vol. 202)*. PMLR, Honolulu, HI, USA, 19730–19742. arXiv:2301.12597 [cs.CV]
- [16] Zihao Li, Qiang Chen, Lixin Zou, Aixin Sun, and Chenliang Li. 2025. Multi-Interest Recommendation: A Survey. *arXiv preprint arXiv:2506.15284* (2025). arXiv:2506.15284 [cs.IR]
- [17] Chi Liu, Jiangxia Cao, Rui Huang, Kai Zheng, Qiang Luo, Kun Gai, and Guorui Zhou. 2024. KuaiFormer: Transformer-Based Retrieval at Kuaishou. *arXiv preprint arXiv:2411.10057* (2024). arXiv:2411.10057 [cs.IR]
- [18] Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. In *New Orleans Conference on Learning Representations (ICLR)*. OpenReview.net, New Orleans, LA, USA, 19 pages. arXiv:1711.05101 [cs.LG]
- [19] Aditya Pal, Chantat Eksombatchai, Yitong Zhou, Bo Zhao, Charles Rosenberg, and Jure Leskovec. 2020. PinnerSage: Multi-Modal User Embedding Framework for Recommendations at Pinterest. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '20)*. ACM, New York, NY, USA, 2311–2320. arXiv:2007.03634 [cs.LG] doi:10.1145/3394486.3403280
- [20] Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. 2017. Dynamic Routing Between Capsules. In *Advances in Neural Information Processing Systems 30 (NeurIPS)*. Curran Associates, Inc., Red Hook, NY, USA, 3856–3866. arXiv:1710.09829 [cs.CV]
- [21] Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2022. ColBERTv2: Effective and Efficient Retrieval via Lightweight Late Interaction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 3715–3734. arXiv:2112.01488 [cs.IR] doi:10.18653/v1/2022.naacl-main.272
- [22] Qiaoyu Tan, Jianwei Zhang, Jiangchao Yao, Ninghao Liu, Jingren Zhou, Hongxia Yang, and Xia Hu. 2021. Sparse-Interest Network for Sequential Recommendation. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining (WSDM '21)*. ACM, New York, NY, USA, 598–606. arXiv:2102.09267 [cs.IR] doi:10.1145/3437963.3441811
- [23] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation Learning with Contrastive Predictive Coding. *arXiv preprint arXiv:1807.03748* (2018). arXiv:1807.03748 [cs.LG]
- [24] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Advances in Neural Information Processing Systems 30 (NeurIPS)*. Curran Associates, Inc., Red Hook, NY, USA, 5998–6008. arXiv:1706.03762 [cs.CL]
- [25] Yueqi Xie, Jingqi Gao, Peilin Zhou, Qichen Ye, Yining Hua, Jae Boum Kim, Fangzhao Wu, and Sunghun Kim. 2023. Rethinking Multi-Interest Learning for Candidate Matching in Recommender Systems. In *Proceedings of the 17th ACM Conference on Recommender Systems (RecSys '23)*. ACM, New York, NY, USA, 283–293. arXiv:2302.14532 [cs.IR] doi:10.1145/3604915.3608766
- [26] Zhenhui Xu, Meng Zhao, Liqun Liu, Lei Xiao, Xiaopeng Zhang, and Bifeng Zhang. 2022. Mixture of Virtual-Kernel Experts for Multi-Objective User Profile Modeling. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22)*. ACM, New York, NY, USA, 4257–4267. arXiv:2106.07356 [cs.IR] doi:10.1145/3534678.3539062
- [27] Xinyang Yi, Ji Yang, Lichan Hong, Derek Zhiyuan Cheng, Lukasz Heldt, Aditee Kumthekar, Zhe Zhao, Li Wei, and Ed Chi. 2019. Sampling-Bias-Corrected Neural Modeling for Large Corpus Item Recommendations. In *Proceedings of the 13th ACM Conference on Recommender Systems (RecSys '19)*. ACM, New York, NY, USA, 269–277. doi:10.1145/3298689.3346996
- [28] Shengyu Zhang, Lingxiao Yang, Dong Yao, Yujie Lu, Fuli Feng, Zhou Zhao, Tat-Seng Chua, and Fei Wu. 2022. Re4: Learning to Re-contrast, Re-attend, Re-construct for Multi-interest Recommendation. In *Proceedings of the ACM Web Conference 2022 (WWW '22)*. ACM, New York, NY, USA, 2216–2226. arXiv:2208.08011 [cs.IR] doi:10.1145/3485447.3512094