

EGR: Embedding-Native Generative Retrieval with a Shared LLM

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

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

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

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

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

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

Wenfeng Zhuo
Snap Inc.
Palo Alto, CA, USA
wzhuo@snapchat.com

Zichu Li
Snap Inc.
Palo Alto, CA, USA
zli12@snapchat.com

Jitin Krishnan
Snap Inc.
Bellevue, WA, USA
jkrishnan2@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

Generative retrieval is increasingly popular in large-scale recommendation and advertising systems, yet current methods introduce practical complications. Semantic-ID methods rely on quantization, mutable identifier vocabularies, and token-to-item grounding; embedding-based pipelines train the item encoder separately from the query generator, which limits user-item alignment. We propose **EGR**, an **E**mboding-native **G**enerative **R**etrieval framework for recommendation and advertising. EGR uses a single shared LLM to learn item representations from item metadata and user representations from interaction histories in one embedding space. Items are indexed directly as dense vectors, and user histories are encoded as dense retrieval queries. Joint contrastive training groups related items and aligns queries with their target items. We evaluate EGR on public benchmarks, industrial data, and live deployment. EGR outperforms published baselines on Amazon Reviews; on Snap DPA, it scales with data, handles cold-start items, and benefits from multi-modal input. In production, EGR delivers a +2.91% conversion-rate lift, simplifying system design while improving retrieval quality and ad performance.

CCS Concepts

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

Keywords

Generative retrieval, recommender systems, large language models

1 Introduction

Large-scale candidate retrieval is a foundational layer in modern recommendation and advertising systems [2, 3, 20, 21]. For each request, a retriever must select a compact set of relevant items

from catalogs that may contain hundreds of millions of items and change continuously, all under strict latency and reliability budgets. A useful retriever must therefore combine personalization with catalog coverage, freshness, and operational simplicity.

Industrial retrieval stacks are commonly built around two-tower embedding models and their sequential extensions [7, 14]. These systems encode queries and items into a shared vector space and retrieve candidates with approximate nearest-neighbor (ANN) search. Generative retrieval (GR) offers a different route: a sequence model conditions on user history and produces the next target, either as a sequence of item identifiers or as a dense retrieval representation. This paradigm has shown strong results in academic benchmarks [13, 22] and in recent industrial deployments [1, 5].

Existing GR methods fall into two main families. *SID-based* methods [4, 5, 13, 18, 19] map items to discrete semantic identifier sequences and train a generator to emit those sequences. They require an item encoder, a quantizer, a generator, and a grounding layer, and catalog updates require assigning identifiers to new or changed items. *Embedding-based* methods [1, 8] generate continuous vectors for ANN search and avoid discrete identifiers, but they train the item encoder separately from the query generator. In that setting, the indexed item space is fixed before the query side is trained, which can make user-item alignment a post-hoc problem.

We propose **EGR**, an embedding-native generative retrieval framework (Figure 1). EGR uses one shared LLM backbone in two paths. The item path encodes item metadata into dense vectors that are written directly to an ANN index. The user-history path consumes a chronological sequence of item engagements and emits a dense query vector. As the two paths share parameters and are optimized jointly, the indexed item space and the user-conditioned query space are learned together rather than stitched together after separate training. Our contributions are:

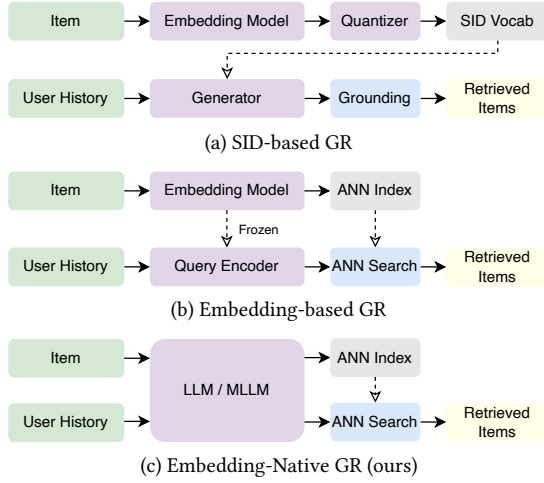


Figure 1: Comparison of generative retrieval families. (a) SID-based GR encodes items, quantizes them into a semantic-ID vocabulary, and trains a generator to emit SID token sequences that are grounded back to items at serve time. (b) Embedding-based GR uses separate item and query encoders; the item embedding model is pretrained or trained first, then fixed during query encoder training, so the indexed item embeddings are not optimized for the generative retrieval objective. (c) Embedding-Native GR (ours) uses one shared LLM backbone for both items and user histories, jointly trained so the indexed item space and the query side evolve together.

- We introduce an embedding-native GR design that removes the quantization and grounding of SID-based methods and the separation of item embedding and user embedding in embedding-based methods, while preserving standard ANN serving.
- We use a shared LLM that natively aligns user and item representations in one embedding space, leveraging pretrained priors for multimodal and cold-start generalization, jointly trained with item-pair and history-to-target contrastive objectives.
- We evaluate EGR on public benchmarks, industrial-scale data, and a live online deployment. EGR outperforms the evaluated published baselines on Amazon Reviews. On industrial Snap DPA data, EGR (i) scales monotonically with more training data, (ii) remains strong on cold-start items, and (iii) benefits from multimodal input; in a live A/B test, it yields a +2.91% CVR lift.

2 Related Work

Sequential recommendation. Sequential recommenders such as SASRec [7] and BERT4Rec [14] established transformer-based next-item prediction as a strong pre-generative baseline. They use learned item-ID embeddings as model parameters rather than content-based item encoders.

SID-based generative retrieval. TIGER quantizes item-content embeddings and trains a transformer to predict the next item’s identifier sequence [13]. PLUM scales this direction to industrial YouTube recommendation with LLM adaptation and improved semantic IDs [5]. Subsequent systems refine identifier construction, alignment, and serving. DAS aligns user and ad semantic IDs

through dual quantization [19], IDGen learns identifier construction jointly with the recommendation objective [15]. OneRec unifies retrieve-and-rank through SID generation with preference alignment [4], and GR4AD and GPR adapt generative recommendation to advertising with value-aware training; GR4AD additionally adds production serving optimizations and GPR proposes a one-model paradigm [18, 23]. Recent work also moves beyond purely numeric SIDs. For example, GRLM proposes structured term IDs drawn from the LLM’s native vocabulary to reduce grounding hallucination during generation [24]. A parallel line of work recognizes that item identifiers should not be learned independently of the downstream generator: PIT co-evolves item tokenization with the generative recommender for dynamic industrial streams [17], and the Spotify semantic-ID study shows that semantic IDs built from jointly fine-tuned bi-encoder embeddings transfer well across search and recommendation [11]. These methods differ in how they build, align, and serve identifiers, but they retain a discrete identifier layer between item representation and retrieval. EGR removes that layer and trains the retrieval embedding space directly.

Embedding-based generative retrieval. PinRec [1] generates dense query representations for Pinterest’s Homefeed, Search, and Related Pins surfaces and retrieves through ANN, avoiding semantic-ID quantization altogether. It uses a two-model design with a separate item encoder and query generator, together with outcome-conditioned generation and temporal multi-token inference. Other systems in this family include TBGRecall [8], which uses a session-level representation for ANN retrieval in e-commerce. EGR differs from PinRec in two central ways. First, a single shared LLM backbone produces both item and query embeddings through the same parameters and projection head. Second, joint training updates the indexed item space and the query side together, rather than freezing the item encoder before training the query generator.

3 Method

3.1 Overall Framework

Figure 2 shows the EGR architecture. EGR learns a shared embedding space for items and user-conditioned queries. Each item is represented by its metadata, including title, brand, category, and image; each user is represented by a chronological sequence of engagement events, where each event contains a conversion type and an item. A shared LLM backbone encodes both inputs, followed by a projection head that compresses the backbone’s hidden state to a lower-dimensional retrieval embedding for compact ANN serving. The item path maps item metadata to an indexed item embedding, and the user-history path maps recent engagements to a query embedding. Because both paths share parameters and emit vectors in the same space, next-item prediction becomes dense retrieval: at inference time, EGR emits a query vector and retrieves items by maximum inner product. This design is *embedding-native*: it never predicts item-specific tokens, quantizes item vectors, or grounds generated identifiers back to items. The two paths are optimized jointly with the contrastive objectives described below.

3.2 Item Representation Learning (IRL)

Item encoder. Let each item i have metadata x_i (title, brand, category, and image). IRL trains item embeddings with co-engagement

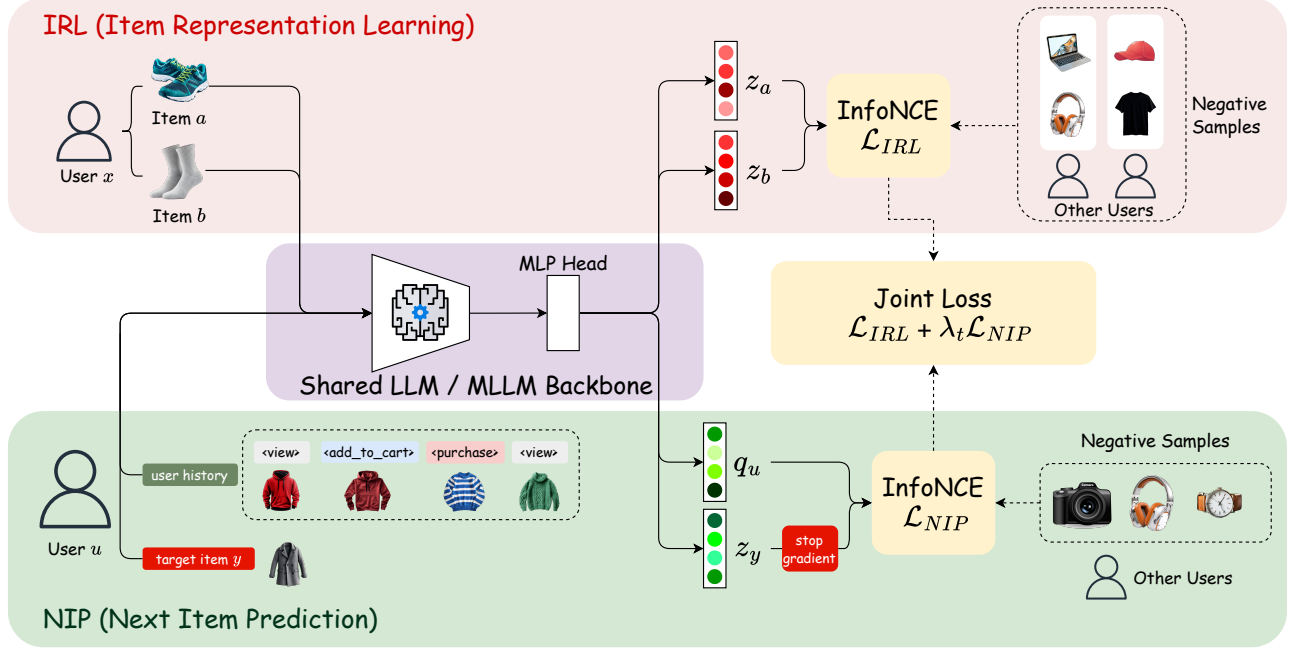


Figure 2: Overall framework of EGR. A shared LLM with a projection head encodes both item and user histories. IRL applies a symmetric InfoNCE on the item pair (p_a, p_b) , shaping the indexed item space. NIP applies a history-to-target InfoNCE from the user-conditioned query embedding to the target item embedding, with stop-gradient on the target. Joint training combines the two losses as $\mathcal{L}_{\text{IRL}} + \lambda_t \mathcal{L}_{\text{NIP}}$.

contrastive learning. The encoder serializes x_i into a fixed prompt and passes it through the shared LLM backbone f_θ followed by a projection head h_ϕ , producing an L2-normalized vector

$$z_i = \frac{h_\phi(f_\theta(x_i))}{\|h_\phi(f_\theta(x_i))\|_2}, \quad (1)$$

which is inserted directly into the ANN index. Unlike semantic-ID systems, z_i is not quantized and requires no token-to-item grounding; a new or updated item can enter the index as soon as its meta-data is encoded.

Pair contrastive objective. For a consecutive co-engaged item pair (p_a, p_b) from the same user’s chronological sequence, IRL applies a symmetric InfoNCE loss [16]:

$$\mathcal{L}_{\text{IRL}} = \frac{1}{2} [\text{InfoNCE}(z_a, z_b) + \text{InfoNCE}(z_b, z_a)], \quad (2)$$

where $\text{InfoNCE}(a, b) = -\log \frac{\exp(a^\top b / \tau)}{\sum_{b' \in \{b\} \cup \mathcal{N}} \exp(a^\top b' / \tau)}$, $\mathcal{N}$ is the set of in-batch negatives, τ is a learned temperature, and all vectors are L2-normalized. IRL uses a co-engagement structure to shape the item space that will later be indexed for retrieval.

3.3 Next Item Prediction (NIP)

History encoder. Let user u be represented by a chronological history $H_u = (e_1, \dots, e_T)$ of (conversion-type, item) events. NIP trains the user-history encoder to produce a query that retrieves the target item. EGR serializes H_u in oldest-to-newest order, prefixing each event with a conversion-type token (`<view>`, `<add_to_cart>`,

`<purchase>`) that reflects different user intents and joins events with separators. The serialized history is encoded through the same backbone and projection head:

$$q_u = \frac{h_\phi(f_\theta(H_u))}{\|h_\phi(f_\theta(H_u))\|_2}. \quad (3)$$

At serving time, q_u retrieves items by maximum inner product against the indexed z_i .

History-to-target objective. The NIP loss is a history-to-target InfoNCE objective with a stop-gradient on the target embedding:

$$\mathcal{L}_{\text{NIP}} = \text{InfoNCE}(q_u, \text{sg}(z_y)), \quad (4)$$

where y is the target item and $\text{sg}(\cdot)$ denotes a stop-gradient. The target embedding z_y is recomputed by the current backbone at every step, but the stop-gradient prevents $\mathcal{L}_{\text{NIP}}$ from updating the target branch for that example. Because the backbone is shared, NIP gradients still update the model through the history branch and can influence future item embeddings through the shared parameters. The stop-gradient prevents per-example target pulls from destabilizing the indexed item space.

3.4 Joint Training

Total objective. EGR optimizes IRL and NIP jointly on the same shared backbone:

$$\mathcal{L} = \mathcal{L}_{\text{IRL}} + \lambda_t \mathcal{L}_{\text{NIP}}. \quad (5)$$

Warm-up schedule. Training proceeds in two phases. During the first 10% of steps, only $\mathcal{L}_{\text{IRL}}$ is active ($\lambda_t = 0$), giving the item

space its initial co-engagement structure. After warm-up, λ_t ramps linearly from 0.1 to 1.0.

Joint training dynamics. IRL keeps the backbone anchored to the item-pair structure while NIP trains user histories to retrieve future items. As a result, the indexed item space and the query side evolve together, rather than forcing the query side to align after the item space has been frozen. Section 4.4.1 compares this configuration with alternatives that ablate task coupling, the stop-gradient mechanism, and backbone sharing.

3.5 Serving Architecture

Offline indexing. EGR separates item-index construction from query generation. Item embeddings can be computed periodically or in near real-time, depending on the cost and freshness tradeoff, and are written to an ANN index; because embeddings are L2-normalized, maximum-inner-product search is equivalent to cosine retrieval.

Query retrieval. EGR supports two query-serving modes. In our deployed pipeline (Section 4.5), each user’s query embedding is generated once per day through the user-history path and stored in the serving layer. For each request, the system loads this precomputed embedding, performs online ANN search against the current product index, and forwards the top- K products to downstream ranking. For surfaces where intent changes within a session, the same encoding path can run at request time: the system serializes the user’s recent item engagements, runs the user-history path online, retrieves top- K items from the index, and forwards candidates to downstream ranking at a higher per-request serving cost.

Practical advantages. This architecture has three operational advantages: (i) a richer item encoder adds no per-request cost when items are indexed offline; (ii) the output is a dense vector, so serving can use standard ANN infrastructure rather than a token-generation stack; and (iii) new or updated items can be re-encoded and inserted into the index without rebuilding a semantic-ID vocabulary or retraining a generator over item-specific tokens.

4 Experiments

4.1 Experimental Setup

Datasets. For public-benchmark evaluation, we use the Amazon Reviews datasets [10] on three categories: Beauty, Sports, and Toys. Following [9, 13], we apply 5-core filtering and use leave-one-out splitting. For industrial evaluation, we use Snap Dynamic Product Ads (DPA) logs of user-product engagement events (we use “product” and “item” interchangeably throughout this section), and hold out 1M user sequences for evaluation, using the remaining sequences for training.

Metrics. We report Recall@10 (R@10) for offline retrieval. For the online deployment, we report relative lifts in impressions, click-through rate (CTR), and conversion rate (CVR). Per Snap’s policy, experiments on Snap data are reported only as relative numbers.

Inputs. Both paths use the model’s chat template. The product path pairs the system instruction “Represent this product for retrieval.” with a user message of the form Title: . . . , Brand: . . . , Category: The user-history path pairs the system instruction “Predict the next product this user will interact with.” with a message that lists each engagement as `<conv_token>` followed

Table 1: Recall@10 on Amazon Reviews. Best per column in bold.

Method	Beauty	Sports	Toys
BERT4Rec [14]	0.0396	0.0175	0.0332
SASRec [7]	0.0607	0.0301	0.0626
TIGER [13]	0.0623	0.0347	0.0547
HSTU [22]	0.0652	0.0343	0.0566
EGR (ours)	0.0655	0.0362	0.0807

by product text, with events joined by ‘|’ separators. For example: `<view>` Title: Galaxy Watch, Brand: Samsung, Category: Electronics | `<add_to_cart>` Title: Coffee Maker, Brand: Philips, Category: Kitchen | `<purchase>` Title: Running Shoes, Brand: Nike, Category: Footwear.

Implementation. We use Qwen3-VL-Embedding 2B [12] as the default backbone with last-token pooling, and add a two-layer MLP projection head (LayerNorm, GELU, dropout) trained from scratch to produce L2-normalized 512-dimensional embeddings. LoRA [6] adapters use $r=128$, $\alpha=256$ on the q/k/v/o attention projections, yielding approximately 41M trainable parameters, mostly in the LoRA matrices and the projection head. We optimize the LoRA matrices, projection head, and learned temperature with AdamW at learning rate 2×10^{-5} , using a short learning-rate warm-up followed by cosine decay. The canonical recipe trains for one epoch on 8 H100 GPUs with per-GPU batch size 64 and maximum history length 20, corresponding to approximately 460 input tokens on average. In-batch negatives are gathered across all Distributed Data Parallel (DDP) ranks, giving $B \times W$ negatives per anchor for local batch size B and W GPUs.

4.2 Public Amazon Reviews Benchmark

We evaluate EGR on three public Amazon Reviews categories: Beauty, Sports, and Toys, and compare against representative sequential recommendation baselines. Table 1 reports R@10 on each category.

- **BERT4Rec [14].** A bidirectional transformer trained with a masked-item (cloze) objective that predicts items via softmax over the item vocabulary.
- **SASRec [7].** A causal self-attention sequence model trained for next-item prediction with learned item embeddings as model parameters.
- **TIGER [13].** An SID-based generative retriever that quantizes item content embeddings into discrete semantic identifier sequences and trains a transformer to generate them autoregressively.
- **HSTU [22].** A hierarchical sequential transducer for generative recommendation at industrial scale.

We find that EGR outperforms the baseline methods on all three categories: it is essentially tied with HSTU on Beauty (0.0655 vs. 0.0652), improves over TIGER on Sports (0.0362 vs. 0.0347), and obtains a larger margin over the next-best baseline on Toys (0.0807 vs. 0.0626).

Table 2: EGR retrieval, stratified by each target product’s training-input frequency. Count = 0: the target never appeared in training input. R@10 normalized to the head bucket (freq > 15) at 100% (higher is better).

Count	% of test samples	rel R@10
0	31.3%	94.9%
1	8.0%	99.1%
2	6.0%	111.5%
3	4.3%	112.7%
4	3.6%	117.6%
5	2.8%	115.4%
6–10	9.4%	119.1%
11–15	5.3%	116.8%
> 15	29.2%	100.0%

4.3 Snap DPA Offline Results

4.3.1 Data Scaling. We train EGR on four matched-event subsets ranging from 10K to 10M users. Figure 3 reports R@10, normalized to its value at 10K training users.

Findings. R@10 increases monotonically with training data, growing by 27% from 10K to 10M users with no observed saturation. More training data continues to improve retrieval quality at industrial scale.

4.3.2 Cold-Start and Frequency Strata. We stratify the offline test set by the training-input frequency of each target product, defined as the number of times the target product appeared in the training data. The cold-start stratum (frequency 0) contains products the model never saw as training inputs; the head stratum (> 15) contains the most frequently seen products. Table 2 reports relative R@10, normalized so the head bucket equals 100%.

Findings. EGR retrieves never-seen products with limited degradation: cold-start R@10 trails the head bucket by only 5.1%. Mid-frequency strata (2–15) score highest. This non-monotonic pattern suggests that factors beyond target frequency affect retrieval quality. Because the buckets are defined only by target frequency, this experiment does not isolate those factors. Thus, the strong cold-start result indicates robustness to unseen products but not parity with the best-performing mid-frequency strata.

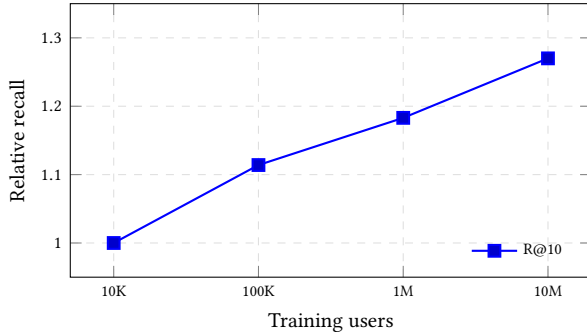


Figure 3: Data scaling of EGR. R@10 normalized to its value at 10K training users.

Table 3: Multimodal vs. text-only product input on Qwen3-VL-Embedding 2B. R@10 normalized to text-only at 100% (higher is better).

Modality	rel R@10
Text-only	100.0%
Text+image	106.9%

Table 4: Joint-training ablation. R@10 normalized to the full EGR recipe at 100% (higher is better). JT: IRL and NIP optimized jointly in a shared backbone. SG: stop-gradient on the target embedding in NIP.

Method	IRL	NIP	JT	SG	rel R@10
EGR (ours)	✓	✓	✓	✓	100.0%
EGR w/o stop-grad	✓	✓	✓	—	95.3%
NIP only	—	✓	—	—	93.7%
IRL only	✓	—	—	—	26.6%
Two models	✓	✓	—	—	86.2%

4.3.3 Multimodal Input. The default Qwen3-VL-Embedding 2B backbone supports multimodal product input through its vision tower. We compare two configurations on the same backbone: *text-only*, which encodes each product using title, brand, and category, and *multimodal*, which additionally feeds the product image through the vision tower. Table 3 reports R@10 normalized to the text-only configuration at 100%.

Findings. Adding the product image improves R@10 by +6.9% over text-only on the same backbone. EGR can therefore use multimodal product evidence without changing the retrieval architecture.

4.4 Snap DPA Ablation Study

We study the joint-training strategy and the NIP loss-weight schedule.

4.4.1 Effective Joint Training. We compare EGR against four alternative configurations. Table 4 reports R@10 normalized so the full EGR recipe equals 100%.

- **EGR (ours).** The shared backbone is trained on IRL and NIP jointly, with stop-gradient on the target embedding in NIP.
- **EGR w/o stop-grad.** Same as EGR (ours) but without the stop-gradient in NIP.
- **NIP only.** The same backbone is trained on NIP alone.
- **IRL only.** The same backbone is trained on IRL alone.
- **Two models.** Model 1 is trained on IRL and frozen; Model 2 is trained on NIP using embeddings produced by Model 1 as input.

Findings. Table 4 shows four effects. First, removing IRL while keeping NIP reduces relative R@10 from 100.0% to 93.7%, indicating that item-pair contrastive learning remains useful even when the history-to-target loss is active. Second, IRL alone reaches only 26.6%, so co-engagement structure is a useful item-space anchor but is not sufficient for next-item retrieval. Third, removing the stop-gradient on the NIP target branch lowers R@10 to 95.3%, consistent with the stop-gradient stabilizing the indexed item space. Fourth,

Table 5: Ablation of training loss schedules. Results are normalized Recall@10 relative to the Joint schedule.

Variant	IRL Schedule	NIP Schedule	rel R@10
Joint (ours)	fixed at 1	λ_t : 0.1 $\rightarrow$ 1.0	100.0%
Balanced	$1 - \lambda_t$	λ_t	97.9%
Sequential	first 50% only	last 50% only	74.4%

Table 6: Online A/B test in Snap DPA over two weeks (10% / 10% user traffic split). Relative lifts of the EGR-enabled treatment arm over the control arm; absolute values are withheld per Snap’s policy.

Metric	Relative lift
Impressions	+0.15%
CTR	+0.23%
CVR	+2.91%

freezing the item side and training a separate query model reaches only 86.2%. Jointly shaping the item and query spaces, with stop-gradient on the target branch, is more effective than ablating either objective or aligning a query generator to a frozen item encoder.

4.4.2 Loss Schedule. We compare three loss-schedule variants for the joint training objective. *Joint* (ours): the IRL weight stays at 1 throughout, while the NIP weight λ_t ramps linearly from 0.1 to 1.0 after warm-up. *Balanced*: IRL weight decays as $1 - \lambda_t$ while NIP weight ramps as λ_t , so total loss weight stays constant. *Sequential*: IRL is active only during the first half of training; NIP is active only during the second half, with no overlap phase. Table 5 reports R@10 normalized to Joint at 100%.

Findings. Maintaining IRL throughout joint training is important. *Balanced* trails *Joint* by only about 2%, suggesting that decayed IRL still preserves enough product-space supervision for the indexed space to remain mostly intact. *Sequential* trails by about 26%: when IRL is turned off after the first half of training, NIP must train against a product space that is no longer directly anchored by co-engagement structure. The overlap phase in which both losses are active is therefore central to the training recipe.

4.5 Online A/B Test

We conducted a two-week online A/B test in Snap DPA. The treatment arm included EGR as an additional retrieval source while holding the total candidate quota fixed; the control arm retained the existing production retrieval stack without EGR. Each arm received 10% of traffic through Snap’s experimentation platform. EGR operated as a hybrid batch-and-online retrieval pipeline: each user’s query embedding was computed once per day and stored for serving; at request time, the system used this embedding to perform real-time ANN search and forward the retrieved candidates to downstream ranking. Downstream ranking, auction, policy, and rendering were held fixed across arms. We report relative lifts over the control arm on Snap’s standard DPA engagement-quality metrics (impressions, CTR, CVR); absolute values are withheld per Snap’s policy.

Findings. Adding EGR to the production retrieval stack increases impressions by +0.15%, CTR by +0.23%, and CVR by +2.91% (Table 6). Because the total candidate quota is held constant across arms, the CTR and CVR gains reflect improved retrieval quality rather than additional candidate allocation. Of all final impressions delivered to users, 5.3% come from EGR. At the source level, EGR’s CTR and CVR are 1.55 $\times$ and 1.53 $\times$ the respective averages across all production retrieval sources, including user-to-item (U2I), item-to-item (I2I), and non-personalized retrievers. After the A/B test, EGR was rolled out across DPA and now serves 100% of users.

5 Conclusion

We presented EGR, an embedding-native framework for generative retrieval. Rather than generating discrete item identifiers or training separate item and query encoders, EGR uses a single shared LLM backbone to produce both indexed item embeddings and user-conditioned query embeddings. The model is trained jointly with a symmetric item-pair contrastive loss and a stop-gradient history-to-target loss, while retaining the operational simplicity of standard ANN serving.

Experimental results demonstrate that EGR outperforms prior sequential recommendation and generative retrieval baselines on Amazon Reviews. On Snap DPA, EGR scales monotonically with training data, remains strong on cold-start items, and benefits from multimodal input. In a live production deployment, incorporating EGR into the retrieval stack yields a +2.91% conversion-rate improvement. Together, these results show that embedding-native generative retrieval can simplify system design while improving both offline retrieval quality and online advertising performance.

References

- [1] Prabhat Agarwal, Anirudhan Badrinath, Laksh Bhasin, Jaewon Yang, Edoardo Botta, Jiajing Xu, and Charles Rosenberg. 2025. PinRec: Outcome-Conditioned, Multi-Token Generative Retrieval for Industry-Scale Recommendation Systems. arXiv:2504.10507 [cs.LG] <https://arxiv.org/abs/2504.10507>
- [2] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishu Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. 2016. Wide & Deep Learning for Recommender Systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (DLRS 2016)*. ACM, New York, NY, USA, 7–10. doi:10.1145/2988450.2988454
- [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] Jiaxin Deng, Shiyao Wang, Kuo Cai, Lejian Ren, Qigen Hu, Weifeng Ding, Qiang Luo, and Guorui Zhou. 2025. OneRec: Unifying Retrieve and Rank with Generative Recommender and Iterative Preference Alignment. arXiv:2502.18965 [cs.LG] <https://arxiv.org/abs/2502.18965>
- [5] Ruining He, Lukasz Heldt, Lichan Hong, Raghunandan Keshavan, Shifan Mao, Nikhil Mehta, Zhengyang Su, Alicia Tsai, Yueqi Wang, Shao-Chuan Wang, Xinyang Yi, Lexi Baugher, Baykal Kakici, Ed Chi, Cristos Goodrow, Ningren Han, He Ma, Romer Rosales, Abby Van Soest, Devansh Tandon, Su-Lin Wu, Weilong Yang, and Yilin Zheng. 2025. PLUM: Adapting Pre-trained Language Models for Industrial-scale Generative Recommendations. arXiv:2510.07784 [cs.LG] <https://arxiv.org/abs/2510.07784>
- [6] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *The Tenth International Conference on Learning Representations (ICLR 2022)*. OpenReview.net, Virtual Event, 13 pages. <https://openreview.net/forum?id=nZeVKeeFYf9>
- [7] Wang-Cheng Kang and Julian McAuley. 2018. Self-Attentive Sequential Recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, Singapore, 197–206. doi:10.1109/ICDM.2018.00035
- [8] Zida Liang, Changfa Wu, Dunxian Huang, Weiqiang Sun, Ziyang Wang, Yuliang Yan, Jian Wu, Yuning Jiang, Bo Zheng, Ke Chen, Silu Zhou, and Yu Zhang. 2025.

- TBGRcall: A Generative Retrieval Model for E-commerce Recommendation Scenarios. arXiv:2508.11977 [cs.IR] <https://arxiv.org/abs/2508.11977>
- [9] Zhanyu Liu, Shiyao Wang, Xingmei Wang, Rongzhou Zhang, Jiaxin Deng, Honghui Bao, Jinghao Zhang, Wuchao Li, Pengfei Zheng, Xiangyu Wu, Yifei Hu, Qigen Hu, Xinchun Luo, Lejian Ren, Zixing Zhang, Qianqian Wang, Kuo Cai, Yunfan Wu, Hongtao Cheng, Zexuan Cheng, Lu Ren, Huanjie Wang, Yi Su, Ruiming Tang, Kun Gai, and Guorui Zhou. 2025. OneRec-Think: In-Text Reasoning for Generative Recommendation. arXiv:2510.11639 [cs.IR] <https://arxiv.org/abs/2510.11639>
- [10] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-Based Recommendations on Styles and Substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. Association for Computing Machinery, New York, NY, USA, 43–52. doi:10.1145/2766462.2767755
- [11] Gustavo Penha, Edoardo D’Amico, Marco De Nadai, Enrico Palumbo, Alexandre Tamborrino, Ali Vardasbi, Max Lefarov, Shawn Lin, Timothy Heath, Francesco Fabbri, and Hugues Bouchard. 2025. Semantic IDs for Joint Generative Search and Recommendation. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems (RecSys ’25)*. ACM, New York, NY, USA, 6 pages. doi:10.1145/3705328.3759300
- [12] Qwen Team. 2025. Qwen3-VL-Embedding-2B. Hugging Face model card. <https://huggingface.co/Qwen/Qwen3-VL-Embedding-2B>
- [13] 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. 2023. Recommender Systems with Generative Retrieval. arXiv:2305.05065 [cs.IR] <https://arxiv.org/abs/2305.05065>
- [14] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM ’19)*. ACM, New York, NY, USA, 1441–1450. doi:10.1145/3357384.3357895
- [15] Juntao Tan, Shuyuan Xu, Wenyue Hua, Yingqiang Ge, Zelong Li, and Yongfeng Zhang. 2024. IDGenRec: LLM-RecSys Alignment with Textual ID Learning. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR ’24)*. ACM, New York, NY, USA, 355–364. doi:10.1145/3626772.3657821
- [16] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation Learning with Contrastive Predictive Coding. arXiv:1807.03748 [cs.LG] <https://arxiv.org/abs/1807.03748>
- [17] Huanjie Wang, Xinchun Luo, Honghui Bao, Zixing Zhang, Lejian Ren, Yunfan Wu, Hongwei Zhang, Liwei Guan, and Guang Chen. 2026. PIT: A Dynamic Personalized Item Tokenizer for End-to-End Generative Recommendation. arXiv:2602.08530 [cs.IR] <https://arxiv.org/abs/2602.08530>
- [18] Ben Xue, Dan Liu, Lixiang Wang, Mingjie Sun, Peng Wang, Pengfei Zhang, Shaoyun Shi, Tianyu Xu, Yunhao Sha, Zhiqiang Liu, Bo Kong, Bo Wang, Hang Yang, Jieting Xue, Junhao Wang, Shengyu Wang, Shuping Hui, Wencai Ye, Xiao Lin, Yongzhi Li, Yuhang Chen, Zhihui Yin, Quan Chen, Shiyang Wen, Wenjin Wu, Han Li, Guorui Zhou, Changcheng Li, Peng Jiang, and Kun Gai. 2026. Generative Recommendation for Large-Scale Advertising. arXiv:2602.22732 [cs.IR] <https://arxiv.org/abs/2602.22732>
- [19] Wencai Ye, Mingjie Sun, Shaoyun Shi, Peng Wang, Wenjin Wu, and Peng Jiang. 2025. DAS: Dual-Aligned Semantic IDs Empowered Industrial Recommender System. arXiv:2508.10584 [cs.IR] <https://arxiv.org/abs/2508.10584>
- [20] Xinyang Yi, Ji Yang, Lichan Hong, Derek Zhiyuan Cheng, Lukasz Heldt, Aditee Kumthekar, Zhe Zhao, Li Wei, and Ed H. 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
- [21] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD ’18)*. ACM, New York, NY, USA, 974–983. doi:10.1145/3219819.3219890
- [22] Jiaqi Zhai, Lucy Liao, Xing Liu, Yueming Wang, Rui Li, Xuan Cao, Leon Gao, Zhaojie Gong, Fangda Gu, Jiayuan He, Yinghai Lu, and Yu Shi. 2024. Actions Speak Louder than Words: Trillion-Parameter Sequential Transducers for Generative Recommendations. In *Proceedings of the 41st International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 235)*. PMLR, Vienna, Austria, 58484–58509. <https://proceedings.mlr.press/v235/zhai24a.html>
- [23] Jun Zhang, Yi Li, Yue Liu, Changping Wang, Yuan Wang, Yuling Xiong, Xun Liu, Haiyang Wu, Qian Li, Enming Zhang, Jiawei Sun, Xin Xu, Zishuai Zhang, Ruoran Liu, Suyuan Huang, Zhaoxin Zhang, Zhengkai Guo, Shuojin Yang, Meng-Hao Guo, Huan Yu, Jie Jiang, and Shi-Min Hu. 2025. GPR: Towards a Generative Pre-trained One-Model Paradigm for Large-Scale Advertising Recommendation. arXiv:2511.10138 [cs.IR] <https://arxiv.org/abs/2511.10138>
- [24] Zhiyang Zhang, Junda She, Kuo Cai, Bo Chen, Shiyao Wang, Xinchun Luo, Qiang Luo, Ruiming Tang, Han Li, Kun Gai, and Guorui Zhou. 2026. Unleashing the Native Recommendation Potential: LLM-Based Generative Recommendation via Structured Term Identifiers. arXiv:2601.06798 [cs.IR] <https://arxiv.org/abs/2601.06798>