Proceedings of Machine Learning Research **318** (2026)
The 39th Canadian Conference on Artificial Intelligence

Canadian AI 2026
25–29 May 2026, Vancouver, BC

Beyond Recency Bias: Combining Sequential and Global Collaborative Signals in LLM-Based Generative Recommendation for Sparse Data

Behrad Ghiasi[†,*], C. I. Ezeife [†, ‡]
[†] School of Computer Science, University of Windsor, On, Canada

Abstract

Recommender systems often use multi-stage retrieval and ranking pipelines, where errors made early cannot be fixed later, which hurts overall recommendation quality. LLM-based generative recommendation avoids this by directly generating item identifiers, but many methods represent each item as a token sequence, which creates two concrete problems: generation is slow because tokens must be produced step by step, and it can fail due to beam-search local optima, where the correct item is dropped early because its first token has low probability. SETRec addresses these issues by representing each item as an order-agnostic set of tokens and using query-guided simultaneous token generation, so the item's CF and semantic tokens are generated in parallel without intra-item token dependency. However, SETRec still uses only one collaborative filtering (CF) token, and when that CF signal comes from a sequence-aware model, it is vulnerable to recency bias, especially in sparse and cold-start settings where recent interactions dominate. CFs-SETRec reduces this recency bias by adding a second CF token chosen to capture long-term preferences, and combining it with the sequential CF signal, which preserves both short-term behavior and long-term affinity and leads to more balanced recommendations under sparse data.

Keywords: Large Language Models, Recommender Systems, Collaborative Filtering, Sequential Recommendation, Generative Models, Set Identifiers.

1. Introduction

Recommender systems have become core infrastructure for helping users navigate vast catalogs of products, media, and services by surfacing items that match their individual preferences and behaviour patterns [1]. In large-scale platforms, these systems are typically implemented as multi-stage pipelines: a lightweight retrieval model first selects a small set of candidate items from millions of possibilities, and a more expressive ranker then orders these candidates using richer features and objectives [2]. Retrieval modules often rely on coarse representations or simple heuristics, while rankers incorporate more detailed collaborative and semantic signals [3]. This separation has enabled industrial-scale deployment, but it also introduces a structural bottleneck: once the retrieval stage fails to include a relevant item, no downstream ranker, no matter how sophisticated, can recover it.

Recent advances in large language models (LLMs) have motivated a different view of recommendation, where a single generative model reads a textualized user history, side information, and task instructions, and directly produces the identifiers of recommended items [4]. In these generative recommendation frameworks, the model generates item identifiers token by token conditioned on user context, effectively unifying retrieval and ranking within one architecture [5]. It removes the hard boundary between stages and allows the model to consider the entire item universe whenever it produces the next recommendation.

The central design problem in generative recommendation is constructing *item identifiers*—the tokens that the model both consumes as input and emits as output. Simple schemes assign each item a single learned token that aggregates collaborative filtering (CF)

[*] ghiasib@uwindsor.ca
[‡] This research was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada under an Operating Grant (OGP-0194134) and a University of Windsor grant.

and semantic signals into one embedding vector [4]. This design keeps the vocabulary small and decoding fast, but risks *representation collapse*: different items, especially in sparse regions of the interaction graph, may converge to nearby embedding points, making disambiguation difficult and exacerbating cold-start issues [5], [6]. More expressive approaches describe items via sequences of multiple tokens (e.g., hierarchical or attribute-level codes) [7], which increase capacity but must be generated step-by-step with beam search, increasing latency and suffering from decoding pathologies such as generic or locally optimal outputs [8].

SETRec addresses this tension by representing each item as an *order-agnostic set* of tokens [6]. In this framework, an item's identifier consists of one collaborative token plus several semantic tokens derived from metadata, and the generative model produces the entire set without committing to a fixed token order. This preserves the flexibility of multi-token identifiers while retaining parallel decoding. However, SETRec's collaborative signal is derived from a single sequential recommender (e.g., SASRec), which is prone to recency bias in sparse and cold-start settings, where limited interaction history causes recent behaviors to dominate and long-term preferences to be underrepresented. [9]. Classical matrix factorization methods have shown that global, order-free structure in the full user–item matrix also carries valuable information, particularly under implicit feedback and data sparsity [10], [11]. Relying on a single CF branch therefore limits how well a generative identifier can capture both short-term dynamics and long-term collaborative structure.

Modern recommender systems must address concerns around fairness, interpretability, and robustness, especially when built on powerful, opaque LLM backbones [12], [13]. Token-level designs that separate different sources of information—such as sequential CF, global CF, and semantics—provide an opportunity to make generative recommenders more transparent by exposing how each signal contributes to the final identifier and recommended list.

Building on SETRec's order-agnostic identifier framework, we propose **Collaborative Filtering Signals-enhanced SETRec (CFs-SETRec)**[14], a generative recommendation model that attaches *two* complementary CF tokens to each item. One token is derived from SASRec and captures short- to medium-term sequential user behavior, which is inherently biased toward recent interactions [15], while the second token is learned via implicit Alternating Least Squares (iALS) and encodes long-term, order-free collaborative patterns across the entire user–item matrix [10], [11]. These CF tokens are fused with semantic tokens within an order-agnostic set, allowing a generative LLM to exploit short- and long-term CF structure alongside item semantics without incurring the decoding overhead of sequence-based identifiers.

The contributions of this work are threefold. **First**, we introduce a dual-CF identifier design for generative recommendation that separates sequential and global CF information into distinct tokens within an order-agnostic set, improving identifier expressiveness while preserving parallel generation. **Second**, we conduct an empirical study across datasets with different sparsity levels to analyse how short- and long-term CF signals interact under the generative identifier framework and how this interaction influences cold-start behaviour. **Third**, we compare CFs-SETRec against the original SETRec and other strong generative baselines, demonstrating that enriching identifiers with parallel CF signals is particularly beneficial in sparse e-commerce scenarios, while remaining competitive in denser domains. The code is available at https://github.com/bradghiasi/CFs-SETRec/.

The remainder of the paper is organized as follows. Section 2 reviews background on collaborative filtering, sequential recommendation, and LLM-based generative recommendation. Section 3 presents the CFs-SETRec architecture, including the construction of dual CF tokens and their integration with semantic identifiers. Section 4 describes the experimental setup and reports results across multiple datasets, with a detailed analysis of sparsity

and cold-start effects. Section 5 discusses limitations and potential extensions, and Section 6 concludes.

2. Related Work

Classical collaborative filtering models first framed recommendation as predicting user preferences from past behaviour [1]. Matrix factorization became the dominant approach by learning low-dimensional latent factors for users and items that scale well to large catalogs [11], [16]. Weighted-Regularized Matrix Factorization (WR-MF) handles implicit feedback (clicks, views, purchases) by weighting observed interactions more heavily than unobserved ones [10], [17]. However, these models compress each item into a single embedding, treat user history as an unordered set, and suffer from popularity bias and selection bias [18], [19], motivating sequence-aware and context-aware models that capture evolving preferences.

Sequence-aware recommender systems extend CF by considering interaction order. Early work combines item-to-item similarity with Markov chains to capture both long-term co-occurrence and short-term transition patterns [20]. Convolutional models such as Caser treat interaction sequences as matrix representations and apply filters to capture local patterns [21]. Recurrent neural networks (RNNs) model longer-range dependencies in session-based recommendation [22], [23]. Transformer-based models such as SASRec use self-attention over the full interaction sequence and have become strong baselines [15]. However, these models rely on single ID embeddings per item and are typically deployed only in ranking stages after upstream retrieval, making their performance dependent on retrieved candidates.

In large-scale applications, recommendation uses multi-stage pipelines where retrieval quickly selects a small candidate set from the full catalog using approximate nearest-neighbour search or lightweight models, then ranking stages apply expressive models to re-score candidates [2], [3]. This architecture is efficient but creates a structural bottleneck: if retrieval fails to include relevant items, ranking models cannot recover them. Sequential models like SASRec are often confined to ranking, while retrieval relies on simplified signals, limiting the system's ability to use sequential patterns and semantic information when forming candidates.

Large language models (LLMs) have enabled researchers to re-frame recommendation as a generative problem. LLMs can encode user behaviour, item content, and instructions in a unified text space and generate recommendation lists, explanations, or reasoning traces [24], [4]. Generative retrieval approaches let models generate item identifiers token by token conditioned on user context, unifying retrieval and ranking without fixed candidate sets [25]. Diffusion-style recommenders corrupt preference vectors with noise and learn to denoise them, becoming robust to sparse or noisy user histories [5]. These frameworks depend strongly on item identifier design.

Recent work focuses on identifier design for generative recommendation. Single-token methods learn one compact token per item from interaction data [26], with efficient search indexes to map generated codes back to catalog items at scale [7]. Collaborative signals from historical user–item interactions remain crucial for accuracy in sparse settings [24], [4]. DreamRec uses guided diffusion to generate an oracle embedding for the next item and maps this vector to the closest real item embedding [25]. E4SRec pretrains a sequential model to obtain fixed item ID embeddings, projects them into a frozen LLM, and maps LLM output back to the embedding space for ranking [27]. Both methods do not directly feed CF signals during pretraining, limiting their modeling of short-term sequential patterns and long-term global preferences.

Multi-token identifier models make items more expressive using multiple tokens. Semantic-ID methods map items to short sequences of learned codewords based on content [28]. SETRec uses an order-agnostic set of tokens—one CF token from a pretrained sequential

model plus several semantic tokens—with sparse attention and parallel decoding to reduce representation collapse, speed generation, and help cold-start items[6]. However, the use of a single CF token per item constrains the model's expressive power in sparse and cold-start settings. BIGRec generates a natural-language description of the ideal next item from user history, then embeds and matches this text against real items, using the description as a multi-token bridge between the LLM and item ID space meaning it does not directly use CF signals [29].

3. Prerequisites and Problem Formulation

Given a dataset of user–item interaction sequences and item metadata, the goal is to learn a generative recommendation model that accurately predicts the next item a user will interact with, particularly under cold–start conditions where users have sparse interaction histories. Formally, for a user u with interaction history $S_u = \{s_1, s_2, \ldots, s_t\}$, where each s_i represents an item, the task is to generate a ranked list of candidate items $\{c_1, c_2, \ldots, c_k\}$ that maximizes the likelihood of the user's next interaction. Traditional single CF signal approaches face limitations when **(1)** user sequences are either too sparse to capture meaningful sequential patterns or when **(2)** sequential dependencies become insufficient due to the problem known as recency bias. To address these challenges, our approach integrates two complementary collaborative filtering signals that capture different temporal scales of user behavior, combined with semantic item representations to form rich, multi–faceted item identifiers.

3.1. First Collaborative Filtering Signal: SASRec

SASRec uses self-attention mechanisms to model temporal patterns in user interaction sequences. For each user, the model extracts their interaction sequence and tokenizes it into fixed-length windows of size $L_{\max}$, applying zero-padding for shorter sequences. Each item is mapped to an embedding vector $e_i \in \mathbb{R}^d$, where d is the embedding dimension. The model applies multiple self-attention layers with causal masking to ensure that the representation at position t depends only on items at positions $\{1, 2, \ldots, t\}$, maintaining the temporal ordering constraint. For each position t, the attention mechanism computes:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right)V,$$

where Q, K, and V are query, key, and value matrices derived from input embeddings through learned linear projections.

3.2. Contextualized Representation Extraction

After processing the interaction sequence through multiple self-attention layers, SASRec produces contextualized embeddings for each item. We extract the final-layer embeddings as our sequential collaborative filtering signal, denoted as E_i^{SASRec} for item i. However, on sparse datasets, SASRec's attention mechanism heavily weights recent items, ranking them in top-K results over 99% of the time and reducing recommendation quality by up to 43% compared to balanced recommendations [9]. Several methods have attempted to address this recency bias with limited success. BERT4Rec treats all temporal contexts uniformly without distinguishing short-term and long-term patterns, while hierarchical models like SHAN use fixed weighting schemes that cannot adapt to dataset sparsity [30]. To address this, our model employs two complementary collaborative filtering signals: SASRec captures short-term sequential patterns (though with recency bias on sparse data), while iALS captures stable global preferences, balancing the sequential bias [31].

3.3. **Second Collaborative Filtering Signal: iALS**

To complement the sequential signal from SASRec, this method employs implicit Alternating Least Squares (iALS) to generate long-term, sequence-agnostic collaborative filtering embeddings. iALS captures global co-occurrence patterns across the entire user–item interaction matrix, providing stable representations independent of temporal dynamics.

The iALS approach performs matrix factorization on the implicit feedback matrix by optimizing user and item latent factors. Given a sparse user–item interaction matrix where $r_{ui} = 1$ indicates that user u interacted with item i, confidence weights c_{ui} are introduced to reflect preference signal strength, defined as $c_{ui} = 1 + \alpha \cdot \text{count}_{ui}$, where count_{ui} represents interaction frequency and α is a hyperparameter. The optimization objective for each user u minimizes the weighted squared error:

$$\min_{x_u} \sum_i c_{ui} \big(p_{ui} - x_u y_i\big)^2 + \lambda \|x_u\|^2 \tag{3.1}$$

where p_{ui} is the binary preference indicator and λ is the regularization parameter.

The iALS algorithm alternately fixes one set of embeddings (users or items) and solves for the other using closed-form solutions. For a fixed item embedding matrix Y, the optimal user embedding is:

$$x_u = \big(A + Y^\top C_u Y + \lambda I\big)^{-1} Y^\top C_u p_u \tag{3.2}$$

where $C_u = \text{diag}(c_{u1}, c_{u2}, \ldots, c_{uI})$ is the diagonal confidence matrix for user u, and p_u is the preference vector. The same update rule applies symmetrically when fixing user embeddings and solving for item embeddings. After convergence, each item i is represented by an embedding $E_i^{\text{iALS}} \in \mathbb{R}^d$ that reflects its global co-occurrence patterns across the entire interaction history.

3.4. **Multi-Token Item Representation with Semantic Fusion**

The model integrates dual collaborative filtering signals with semantic embeddings derived from item metadata. For each item i, textual features from its title and description are extracted and encoded using pre-trained language models to produce semantic embedding vectors $\{E_i^{S1}, E_i^{S2}, \ldots, E_i^{SK}\}$, where each E_i^{Sk} represents a different semantic facet. Since the collaborative filtering embeddings from SASRec and iALS may reside in different dimensional spaces, learned linear projection layers map them into a unified latent space of dimension d_{unified}. Each item is then represented as a collection of token embeddings: $\text{ID}_i = \{E_i^{\text{SAS}}, E_i^{\text{iALS}}, E_i^{S1}, E_i^{S2}, \ldots, E_i^{SK}\}$. Given a user history $\{i_1, i_2, \ldots, i_r\}$, the multi-token identifier of each item is retrieved and stacked to form the history representation: $\text{History} = [\text{ID}_{i_1}; \text{ID}_{i_2}; \ldots; \text{ID}_{i_r}]$ For generation, learnable query embeddings $(Q^{\text{SAS}}, Q^{\text{iALS}}, Q^{S1}, Q^{S2}, \ldots, Q^{SK})$, are appended to the encoded history to form the complete input sequence. Following the order-agnostic masking strategy of SETRec [6], a block mask M_{block} is defined such that each pair (t, t') of token positions receives:

$$M_{\text{block}}[t, t'] = \begin{cases} 0, & \text{if the tokens belong to different items,} \\ -\infty, & \text{if the tokens belong to the same item.} \end{cases} \tag{3.3}$$

This mask eliminates intra-item attention for all tokens, whether they belong to history items or query embeddings.Then, the sparse attention mechanism is applied across multiple transformer layers. For each layer, the sparse attention is computed as:

$$\text{Attention}_{\text{sparse}}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d}} + M_{\text{block}}\right) V \tag{3.4}$$

During the softmax operation, the $-\infty$ entries become zeros after exponentiation. Residual connections between layers and layer normalization are applied following standard transformer architecture. After N_{layers} layers of sparse attention, the model produces contextualized embeddings for each token. The query tokens, which can attend to all history tokens but not to each other, generate predictions for all token dimensions simultaneously in parallel rather than autoregressively.

3.5. **Grounding to Catalog Items**

The generated token embeddings do not necessarily correspond to any real item in the catalog. To ground the prediction, we compute similarity scores between the generated set and all item identifiers in the vocabulary. For each candidate item j with identifier $\text{ID}_j = \{E_j^{\text{SAS}}, E_j^{\text{iALS}}, E_j^{S_1}, E_j^{S_2}, \ldots\}$ we calculate a weighted similarity score:

$$s_j = (1-\beta)\Big[\alpha_1 \operatorname{sim}\big(\hat{E}^{\text{SAS}}, E_j^{\text{SAS}}\big) + \alpha_2 \operatorname{sim}\big(\hat{E}^{\text{iALS}}, E_j^{\text{iALS}}\big)\Big] + \beta \sum_k \operatorname{sim}\big(\hat{E}^{S_k}, E_j^{S_k}\big) \quad (3.5)$$

where $\hat{E}$ denotes the generated embeddings, $\operatorname{sim}(\cdot,\cdot)$ is the cosine similarity function, and β, α_1, and α_2 are tunable hyperparameters controlling the relative importance of different signals.

3.6. **Generation Loss with Dimension-Specific Similarity**

During training, for each sample in the dataset, the model compares generated token embeddings against the ground-truth next item's tokens using dimension-specific similarity:

$$\mathcal{L}_{\text{Gen}} = -\frac{1}{|\mathcal{D}|} \sum_{S_u \in \mathcal{D}} \sum_{k \in \mathcal{F}} \log \frac{\exp\big(\operatorname{sim}(\hat{\mathbf{z}}_k, \mathbf{z}_k)\big)}{\sum_{\mathbf{z} \in \mathcal{Z}_k} \exp\big(\operatorname{sim}(\hat{\mathbf{z}}_k, \mathbf{z})\big)} \quad (3.6)$$

where $\mathcal{D} = \{S_u \mid u \in \mathcal{U}\}$ is the dataset of user interaction sequences with $S_u = [i_1^u, i_2^u, \ldots, i_L^u]$, $\mathcal{F} = \{\text{CF}_1, \ldots, \text{CF}_m, S_1, \ldots, S_n\}$ is the set of all information dimensions, $\hat{\mathbf{z}}_k$ is the generated embedding for dimension k, $\mathbf{z}_k$ is the target token in dimension k, and $\mathcal{Z}_k$ is the token corpus for dimension k. This loss encourages generated embeddings to align with their corresponding target tokens while being pushed away from other tokens within the same dimension.

3.7. **Hyperparameter Tuning**

The weighting hyperparameters play a critical role in adapting the model to different data characteristics. The parameter β controls the trade-off between collaborative filtering and semantic signals: when data is sparse, higher β values increase reliance on semantic information, while lower β values emphasize collaborative patterns when interaction histories are rich. Within the collaborative filtering component, α_1 and α_2 separately weight the SASRec and iALS signals, with the constraint $\alpha_1 + \alpha_2 = 1$. In datasets where sequential information is weak due to sparse interactions or large temporal gaps, the model adaptively increases α_2 to rely more heavily on the global iALS signal. Conversely, when sequential patterns are reliable, α_1 increases to emphasize the recency-aware SASRec embeddings. These hyperparameters are tuned through grid search on the validation set, and the optimal configuration varies by dataset density and user behavior patterns.

Algorithm 1: CFs-SETRec Training and Inference (Balanced References)

Input: Interaction sequences $\mathcal{S} = \{S_u \mid u \in \mathcal{U}\}$; item set $\mathcal{I}$; metadata $\mathcal{M}_i$; precomputed tables $\mathbf{V}^{\text{SAS}}$ and $\mathbf{V}^{\text{iALS}}$; fusion weights $\alpha_1, \alpha_2, \beta$ (used in Eq. (3.3)); learning rate η; epochs N

Output: Trained parameters Θ

Training;
Initialize model parameters Θ (projection, sparse Transformer, semantic encoder);
for *epoch* $= 1$ **to** N **do**
  for *each user* $u \in \mathcal{U}$ **do**
    Load user history S_u;
    $\mathbf{E}^{\text{SAS}} \leftarrow \text{Lookup}(\mathbf{V}^{\text{SAS}}, S_u)$; `// SASRec signal`
    $\mathbf{E}^{\text{iALS}} \leftarrow \text{Lookup}(\mathbf{V}^{\text{iALS}}, u)$; `// iALS signal`
    $\mathbf{E}^{S_1}, \ldots, \mathbf{E}^{S_K} \leftarrow \text{SemanticEncode}(\mathcal{M}_i \text{ for } i \in S_u)$;
    Project all embeddings into a unified latent space and build a multi-token history;
    Construct block-sparse mask $\mathbf{M}_{\text{block}}$ (Eq. (3.3));
    Encode history using sparse attention (Eq. (3.4));
    Generate next-item token embeddings for all dimensions in parallel;
    Compute generation loss using dimension-specific similarity (Eq. (3.6));
    Update $\Theta \leftarrow \Theta - \eta \nabla \mathcal{L}_{\text{gen}}$;
Inference;
Given S_u, run the embedding, encoding, and generation steps to obtain generated tokens;
for *each candidate item* $j \in \mathcal{I}$ **do**
  Compute grounding score using the fusion rule (Eq. (3.5));
Return top-N items ranked by the grounding score;

4. Time Complexity Analysis

Implicit Alternating Least Squares (iALS). It is a standard matrix factorization algorithm for implicit-feedback recommenders, but its scalability is strongly shaped by its computational cost. For a dataset with M users, N items, $|R|$ observed interactions, and d latent factors, one iteration of iALS has time complexity $\mathcal{O}\big((M+N)d^3 + |R|d^2\big)$ [10], [32]. The cubic term in d comes from inverting $d \times d$ matrices when solving the per-user and per-item least-squares subproblems, which becomes a bottleneck for high-dimensional embeddings. The $|R|d^2$ term reflects the cost of updating gradients over all observed interactions and grows quadratically with the number of factors [32]. Although faster variants such as element-wise ALS reduce the dependence on d by avoiding full matrix inversions, they trade off some convergence speed and vectorization efficiency [32]. Despite these costs, iALS remains attractive because it handles very large, sparse catalogs without explicit negative sampling and is widely used for implicit-feedback data [33].

SETRec. The time complexity of SETRec is analyzed explicitly by its authors in Section 3.2.2 of their SIGIR 2025 paper [6]. SETRec uses an order-agnostic set identifier and a sparse attention mask to encode user histories and generate all item tokens simultaneously. With M information dimensions (tokens per item), sequence length L, and hidden size d, dense self-attention over all tokens and dimensions would require $\mathcal{O}(M^3L^2d)$ operations during batched generation. However, by enforcing sparse block masking that removes intra-item attention, SETRec reduces this to $\mathcal{O}(M^2L^2d)$ for simultaneous generation [6].

CFs-SETRec. When iALS is added as a second collaborative filtering signal on top of SETRec, each item receives one additional CF token, increasing the number of tokens per item from M to $M+1$. Under the same sparse attention design, the time complexity for both pretraining and inference becomes $\mathcal{O}\big((M+1)^2L^2d\big)$, compared to $\mathcal{O}(M^2L^2d)$ for the original SETRec [6]. This is only a constant-factor increase. Moreover, the iALS model is trained offline as a separate matrix factorization step, so its precomputation cost does not directly affect CFs-SETRec's online training or inference time [10], [33].

5. Results and Analysis

5.1. Experimental Setup

Experiments were conducted on a server with an AMD EPYC 7742 processor (8 cores, 3.2 GHz), 128 GB RAM, and one NVIDIA A16 GPU with 15 GB VRAM, using Debian GNU/Linux 12. Training employed a batch size of 512 with micro-batches of 64 for efficient GPU utilization, with a learning rate of 1×10^{-3}. The dataset was split into training, validation, and test sets in an 8:1:1 ratio. Model optimization used AdamW over 20 epochs, and Mean Squared Error (MSE) loss was adopted for tokenization.

5.2. Datasets and Performance

We compare CFs-SETRec with several baseline models that mainly differ in how they represent items for recommendation. Specifically, DreamRec [25] refines ID embeddings via diffusion, E4SRec [27] ranks items using pretrained CF embeddings, BIGRec [29] uses item titles as identifiers, IDGenRec [34] learns compact textual tags, CID [7] constructs identifiers through co-occurrence-based clustering, SemID [7] derives them from category hierarchies, TIGER [5] quantizes semantic features with RQ-VAE, and LETTER [35] integrates semantic and CF information to learn richer identifiers. Moreover, The Amazon datasets were taken from the public Amazon review dataset source[1], and Steam was adopted from prior work.[2]

CFs-SETRec achieves its largest gains over SETRec on the Amazon Toys dataset, which contains 19,412 users, 11,924 items, and 167,597 interactions with 0.07 density and 5-core filtering. Recall@5 improves from 0.0443 to 0.0542 (+22.35%) and Recall@10 from 0.0812 to 0.0887 (+9.24%). NDCG@5 rises from 0.0310 to 0.0386 (+24.52%) and NDCG@10 from 0.0445 to 0.0510 (+14.61%), yielding a 17.68% average improvement across all metrics, showing that the model both retrieves more relevant items and ranks them higher in this sparse, cold-start-prone setting.

The Amazon Beauty dataset (22,363 users, 12,101 items, 198,502 interactions, 0.07 density, 5-core) shares Toys' sparsity level but exhibits more moderate gains over SETRec, indicating that sequential patterns are already well captured by SASRec. Recall@5 increases from 0.0384 to 0.0447 (+16.41%), while Recall@10 improves from 0.0761 to 0.0801 (+5.26%), suggesting diminishing advantages as the candidate set widens. NDCG@5 improves from 0.0280 to 0.0313 (+11.79%) and NDCG@10 from 0.0413 to 0.0452 (+9.44%), for an average improvement of 10.72%. The smaller gains compared with Toys suggest that Beauty products rely more on semantic features, making iALS less beneficial.

The Amazon Sports dataset (35,598 users, 18,357 items, 296,337 interactions, 0.045 density, 5-core) is the sparsest and presents the most challenging conditions. Despite this, CFs-SETRec delivers consistent improvements over SETRec: Recall@5 increases from 0.0341 to 0.0369 (+8.21%) and Recall@10 from 0.0595 to 0.0657 (+10.42%). NDCG@5 rises from 0.0233 to 0.0245 (+5.15%) and NDCG@10 from 0.0323 to 0.0361 (+11.76%), giving an average gain of 8.88%, the lowest among the Amazon datasets. Absolute scores remain below

[1] https://jmcauley.ucsd.edu/data/amazon/
[2] https://github.com/kang205/SASRec

Toys and Beauty (Recall@5 in the 0.03–0.04 range vs. 0.04–0.05), confirming that extreme sparsity fundamentally limits performance regardless of architectural improvements, even though consistent gains across all metrics show measurable benefits under these conditions.

The Steam Video Games dataset (2,567 users, 5,155 items, 41,554 interactions, 0.5 density) provides a contrasting dense setting with roughly 36.7 games per user. In this regime, CFs-SETRec shows slight but consistent degradation relative to SETRec: Recall@5 drops from 0.0313 to 0.0308 (–1.60%) and Recall@10 from 0.0572 to 0.0565 (–1.22%), while NDCG@5 decreases from 0.0248 to 0.0244 (–1.61%) and NDCG@10 from 0.0342 to 0.0338 (–1.17%). The average degradation of –1.40% may suggest that, in this dense dataset, where a single CF signal already captures much of the user preference information, adding a second CF signal may introduce unnecessary complexity and weaken generalization.

Dataset	**Model**	**R@5**	**R@10**	**N@5**	**N@10**
Amazon Toys	DreamRec	0.0066	0.0168	0.0045	0.0082
	EASRec	0.0065	0.0122	0.0056	0.0078
	BIGRec	0.0278	0.0360	0.0196	0.0223
	IDGenRec	0.0318	0.0589	0.0236	0.0335
	CID	0.0359	0.0411	0.0047	0.0066
	SemID	0.0307	0.0507	0.0220	0.0292
	TIGER	0.0315	0.0555	0.0228	0.0314
	LETTER	0.0383	0.0395	0.0115	0.0190
	SETRec	0.0443	0.0812	0.0310	0.0445
	CFs-SETRec	**0.0542**	**0.0887**	**0.0386**	**0.0510**
Amazon Beauty	DreamRec	0.0078	0.0161	0.0065	0.0094
	EASRec	0.0072	0.0118	0.0065	0.0077
	BIGRec	0.0106	0.0251	0.0095	0.0151
	IDGenRec	0.0187	0.0350	0.0186	0.0224
	CID	0.0087	0.0183	0.0071	0.0104
	SemID	0.0260	0.0465	0.0178	0.0255
	TIGER	0.0190	0.0325	0.0130	0.0178
	LETTER	0.0251	0.0410	0.0241	0.0285
	SETRec	0.0384	0.0761	0.0280	0.0413
	CFs-SETRec	**0.0447**	**0.0801**	**0.0313**	**0.0452**
Amazon Sports	DreamRec	0.0045	0.0108	0.0032	0.0049
	EASRec	0.0031	0.0093	0.0019	0.0030
	BIGRec	0.0069	0.0301	0.0043	0.0061
	IDGenRec	0.0181	0.0302	0.0130	0.0177
	CID	0.0082	0.0149	0.0037	0.0049
	SemID	0.0254	0.0439	0.0192	0.0243
	TIGER	0.0190	0.0326	0.0119	0.0159
	LETTER	0.0241	0.0404	0.0241	0.0285
	SETRec	0.0341	0.0595	0.0233	0.0323
	CFs-SETRec	**0.0369**	**0.0657**	**0.0245**	**0.0361**
Steam Video Games	DreamRec	0.0017	0.0029	0.0003	0.0008
	EASRec	0.0006	0.0010	0.0001	0.0002
	BIGRec	0.0099	0.0197	0.0129	0.0127
	IDGenRec	0.0151	0.0311	0.0039	0.0070
	CID	0.0060	0.0107	0.0016	0.0025
	SemID	0.0207	0.0422	0.0107	0.0117
	TIGER	0.0192	0.0336	0.0112	0.0150
	LETTER	0.0161	0.0260	0.0107	0.0147
	SETRec	0.0313	0.0572	0.0248	0.0342
	CFs-SETRec	**0.0308**	**0.0655**	**0.0244**	**0.0338**

Table 1. Cold-Start Results Across All Datasets.

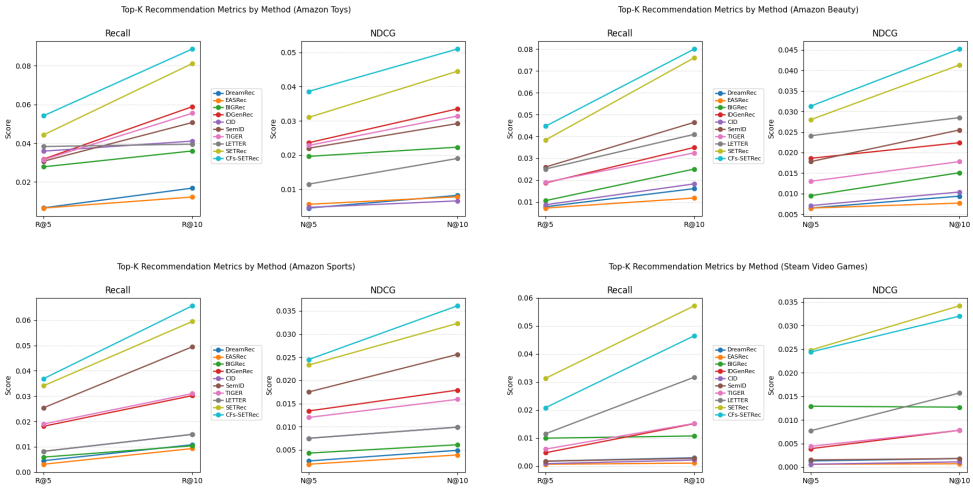

Figure 1. Recall and NDCG across K for each method on all datasets.

6. Conclusion

CFs-SETRec mitigates embedding collapse in LLM-based generative recommendation by combining two complementary collaborative filtering signals—SASRec for short- to mid-term sequential patterns and iALS for long-term global patterns—within multi-token item identifiers that also include semantic embeddings. This dual-signal design preserves representation diversity and allows the model to adapt how much it relies on each signal depending on dataset characteristics.

Experiments show clear dataset-dependent behaviour. On moderately sparse Amazon Toys and Beauty (density 0.07), CFs-SETRec outperforms SETRec with average gains of 17.68% and 10.72% across all metrics, with improvements above 20% at $K = 5$, which is especially helpful for cold-start users where top-K accuracy is crucial. On the sparsest dataset, Amazon Sports (density 0.045), the average gain drops to 8.88%, indicating that extreme sparsity fundamentally limits how much collaborative filtering can help, regardless of architecture. On the dense Steam dataset (density 0.5), CFs-SETRec slightly underperforms SETRec, showing that when interaction data are abundant and a single CF signal already captures user preferences well, adding a second CF signal yields diminishing or even negative returns.

The key insight is that an optimal sparsity range exists—neither too sparse to limit collaborative structure nor too dense to require additional signals—where dual CF modeling provides maximal benefit. For typical e-commerce platforms with densities between 0.05 and 0.10, the observed 10–20% cold-start improvements justify incorporating two CF components, particularly given that iALS can be trained offline. For dense platforms such as gaming or streaming services, simpler single-CF architectures remain preferable.

References

[1] P. Resnick and H. R. Varian. “Recommender systems”. In: *Communications of the ACM* 40.3 (1997), pp. 56–58.

[2] P. Covington, J. Adams, and E. Sargin. “Deep neural networks for youtube recommendations”. In: *Proceedings of the 10th ACM conference on recommender systems*. 2016, pp. 191–198.

[3] J. Huang, J. Chen, J. Lin, J. Qin, Z. Feng, W. Zhang, and Y. Yu. "A comprehensive survey on retrieval methods in recommender systems". In: *ACM Transactions on Information Systems* (2024).

[4] L. Li, Y. Zhang, D. Liu, and L. Chen. "Large language models for generative recommendation: A survey and visionary discussions". In: *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*. 2024, pp. 10146–10159.

[5] S. Rajput, N. Mehta, A. Singh, R. Hulikal Keshavan, T. Vu, L. Heldt, L. Hong, Y. Tay, V. Tran, J. Samost, et al. "Recommender systems with generative retrieval". In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 10299–10315.

[6] X. Lin, H. Shi, W. Wang, F. Feng, Q. Wang, S.-K. Ng, and T.-S. Chua. "Order-agnostic identifier for large language model-based generative recommendation". In: *Proceedings of the 48th international ACM SIGIR conference on research and development in information retrieval*. 2025, pp. 1923–1933.

[7] W. Hua, S. Xu, Y. Ge, and Y. Zhang. "How to index item ids for recommendation foundation models". In: *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*. 2023, pp. 195–204.

[8] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi. "The curious case of neural text degeneration". In: *arXiv preprint arXiv:1904.09751* (2019).

[9] J. Oh and S. Cho. "Measuring Recency Bias In Sequential Recommendation Systems". In: *arXiv preprint arXiv:2409.09722* (2024).

[10] Y. Hu, Y. Koren, and C. Volinsky. "Collaborative filtering for implicit feedback datasets". In: *2008 Eighth IEEE international conference on data mining*. Ieee. 2008, pp. 263–272.

[11] Y. Koren, R. Bell, and C. Volinsky. "Matrix factorization techniques for recommender systems". In: *Computer* 42.8 (2009), pp. 30–37.

[12] S. Geng, Z. Fu, Y. Ge, L. Li, G. De Melo, and Y. Zhang. "Improving personalized explanation generation through visualization". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2022, pp. 244–255.

[13] M. Jiang, K. Bao, J. Zhang, W. Wang, Z. Yang, F. Feng, and X. He. "Item-side fairness of large language model-based recommendation system". In: *Proceedings of the ACM Web Conference 2024*. 2024, pp. 4717–4726.

[14] B. Ghiasi. "CFs-SETRec: LLM-Based Generative Recommendation Capturing Short- to Long-Term CF User-Item Dependencies". Unsubmitted thesis manuscript (in preparation). MA thesis. Windsor, Ontario, Canada: University of Windsor, 2026.

[15] W.-C. Kang and J. McAuley. "Self-attentive sequential recommendation". In: *2018 IEEE international conference on data mining (ICDM)*. IEEE. 2018, pp. 197–206.

[16] G. Takács, I. Pilászy, B. Németh, and D. Tikk. "Matrix factorization and neighbor based algorithms for the netflix prize problem". In: *Proceedings of the 2008 ACM conference on Recommender systems*. 2008, pp. 267–274.

[17] S. Rendle. "Item recommendation from implicit feedback". In: *Recommender Systems Handbook*. Springer, 2021, pp. 143–171.

[18] A. Gunawardana and G. Shani. "A survey of accuracy evaluation metrics of recommendation tasks." In: *Journal of Machine Learning Research* 10.12 (2009).

[19] H. Steck. "Training and testing of recommender systems on data missing not at random". In: *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2010, pp. 713–722.

[20] R. He and J. McAuley. "Fusing similarity models with markov chains for sparse sequential recommendation". In: *2016 IEEE 16th international conference on data mining (ICDM)*. IEEE. 2016, pp. 191–200.

[21] J. Tang and K. Wang. "Personalized top-n sequential recommendation via convolutional sequence embedding". In: *Proceedings of the eleventh ACM international conference on web search and data mining*. 2018, pp. 565–573.

[22] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk. "Session-based recommendations with recurrent neural networks". In: *arXiv preprint arXiv:1511.06939* (2015).

[23] Q. Liu, S. Wu, D. Wang, Z. Li, and L. Wang. "Context-aware sequential recommendation". In: *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE. 2016, pp. 1053–1058.

[24] L. Wu, Z. Zheng, Z. Qiu, H. Wang, H. Gu, T. Shen, C. Qin, C. Zhu, H. Zhu, Q. Liu, et al. "A survey on large language models for recommendation". In: *World Wide Web* 27.5 (2024), p. 60.

[25] Z. Yang, J. Wu, Z. Wang, X. Wang, Y. Yuan, and X. He. "Generate what you prefer: Reshaping sequential recommendation via guided diffusion". In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 24247–24261.

[26] J. Liu, L. Collins, J. Tang, T. Zhao, N. Shah, and C. M. Ju. "Understanding Generative Recommendation with Semantic IDs from a Model-scaling View". In: *arXiv preprint arXiv:2509.25522* (2025).

[27] X. Li, C. Chen, X. Zhao, Y. Zhang, and C. Xing. "E4srec: An elegant effective efficient extensible solution of large language models for sequential recommendation". In: *arXiv preprint arXiv:2312.02443* (2023).

[28] C. M. Ju, L. Collins, L. Neves, B. Kumar, L. Y. Wang, T. Zhao, and N. Shah. "Generative Recommendation with Semantic IDs: A Practitioner's Handbook". In: *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*. 2025, pp. 6420–6425.

[29] K. Bao, J. Zhang, W. Wang, Y. Zhang, Z. Yang, Y. Luo, C. Chen, F. Feng, and Q. Tian. "A bi-step grounding paradigm for large language models in recommendation systems". In: *ACM Transactions on Recommender Systems* 3.4 (2025), pp. 1–27.

[30] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, and P. Jiang. "BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer". In: *Proceedings of the 28th ACM international conference on information and knowledge management*. 2019, pp. 1441–1450.

[31] H. Ying, F. Zhuang, F. Zhang, Y. Liu, G. Xu, X. Xie, H. Xiong, and J. Wu. "Sequential recommender system based on hierarchical attention network". In: *IJCAI international joint conference on artificial intelligence*. 2018.

[32] X. He, H. Zhang, M.-Y. Kan, and T.-S. Chua. "Fast matrix factorization for online recommendation with implicit feedback". In: *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. 2016, pp. 549–558.

[33] S. Rendle, W. Krichene, L. Zhang, and Y. Koren. "IALS++: Speeding up matrix factorization with subspace optimization". In: *arXiv preprint arXiv:2110.14044* (2021).

[34] J. Tan, S. Xu, W. Hua, Y. Ge, Z. Li, and Y. Zhang. "Idgenrec: Llm-recsys alignment with textual id learning". In: *Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval*. 2024, pp. 355–364.

[35] W. Wang, H. Bao, X. Lin, J. Zhang, Y. Li, F. Feng, S.-K. Ng, and T.-S. Chua. "Learnable item tokenization for generative recommendation". In: *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*. 2024, pp. 2400–2409.