# ClusRefineRec: Cluster-Guided Embedding Refinement for Sequential Recommendation

Jiangnan Gu[1], Xinru Liu[1*], Shengjun Liu[1]
[1] School of Mathematics and Statistics, HNP-LAMA, Central South University, Changsha, China
242112016@csu.edu.cn, *liuxinru@csu.edu.cn, shjliu.cg@csu.edu.cn

***Abstract*—While deep learning has greatly improved sequential recommendation, most methods still rely only on back-propagation to learn item embeddings. When interaction data is sparse, these models struggle to capture the relationships between items and their semantic consistency. As a result, the learned embeddings are not expressive enough. To address this issue, we propose ClusRefineRec, a clustering-based framework for embedding refinement. We introduce a clustering loss that is separate from the main recommendation objective. This loss encourages consistency among positive item embeddings. After each training epoch, we further refine the embedding space using the learned cluster centroids. We also design a dynamic state fusion module. It adaptively combines hidden states from all time steps. This helps produce a stronger sequential representation. Extensive experiments show that ClusRefineRec improves both accuracy and robustness. At the same time, it keeps a plug-and-play design, so it can be easily integrated into many existing sequential recommendation models. Implementation: https://github.com/XiaoG408/ClusRefineRec.**

***Index Terms*—Sequential recommendation, Clustering, Embedding refinement, State fusion**

## I. Introduction

Sequential Recommendation (SR) is an important branch of recommender systems. It has demonstrated strong potential in e-commerce, online services, and content delivery [1]–[3]. SR aims to predict users' future preferences and interaction intentions. It achieves this by modeling behavioral patterns and latent dependencies in historical interaction sequences [4]. Deep learning has driven rapid progress in this field. It has enabled the development of a wide range of SR models. These models are based on Recurrent Neural Networks (RNNs) [5], Convolutional Neural Networks (CNNs) [6], Transformers [7], [8], and State Space Models (SSMs) [9], [10]. These advanced architectures significantly enhance modeling capacity. They also improve predictive performance. Meanwhile, many studies aim to further boost performance. They refine existing architectures. They also propose more expressive and robust modeling approaches [11], [12].

Although deep recommendation models have achieved significant progress recently, most approaches rely on back-propagation to implicitly learn item embeddings. They do not explicitly model or impose structural constraints on the embedding space [7]–[10]. Under sparse interaction data, these models often struggle to capture latent semantic characteristics of items, leading to insufficiently expressive embeddings. [13], [14]. Some studies attempt to alleviate popularity bias caused by long-tail distributions through embedding normalization or enhancement [15], [16]. Others introduce adversarial training to mitigate representation degradation [17]. However, these methods largely overlook a fundamental issue: the latent relational structure among items and their semantic consistency are not sufficiently captured. Therefore, semantically similar items are often distributed loosely in the embedding space. Furthermore, many architectures rely solely on the final timestep to represent user preferences, overlooking earlier interactions and limiting sequence expressiveness.

To overcome these limitations, we propose ClusRefineRec, a plug-and-play framework for embedding refinement. It incorporates a dynamic state fusion mechanism that adaptively aggregates hidden states across all time steps, producing richer and more informative final sequence representations. In addition, we introduce a clustering loss that is decoupled from the primary recommendation objective. It learns trainable cluster centroids and encourages compactness among positive item embeddings, thereby reinforcing their latent semantic relations. After each training epoch, the system performs a soft clustering based iterative refinement of all item embeddings under the guidance of the learned centroids. This process progressively enhances the latent relational structure of the entire embedding space, extending beyond positive items alone.

The main contributions of this work are summarized below:

- A dynamic state fusion mechanism adaptively aggregates hidden states across all time steps, enhancing the expressiveness of the final sequence representation.
- By introducing a clustering loss that is fully decoupled from the main recommendation objective during training, the model promotes compactness among positive item embeddings, thereby enhancing the semantic consistency of the latent representation space.
- After each training epoch, the model performs progressive refinement of the entire embedding space under the guidance of the learned cluster centroids, continuously enhancing the latent relational structure among items, rather than being limited to positive items alone.
- The proposed framework, **ClusRefineRec**, integrates these techniques into a unified, plug-and-play architecture, demonstrating consistent gains in both accuracy and robustness through extensive experiments.

* is the corresponding author.

## II. Related Work

### A. Sequential Recommendation

With the rise of deep learning, Recurrent Neural Networks (RNNs) [5] and Convolutional Neural Networks (CNNs) [6] were introduced into sequential recommendation, laying the foundation for modeling users' evolving interests. Transformer-based models [7], [8] later became dominant for their strong ability to capture long-range dependencies, but their quadratic complexity with respect to sequence length limits scalability. To address this, lightweight MLP-based architectures [18] have been explored, offering linear complexity with competitive performance. More recently, Mamba [19] has shown strong sequence modeling capabilities across domains, and its adaptations to recommendation [9], [10] further underscore its efficiency and effectiveness. Unlike these methods, we propose a plug-and-play framework that more effectively leverages embedding information to enhance model capability.

### B. Embedding Optimization in Sequential Recommendation

A growing line of research focuses on improving embedding quality to address various representation deficiencies in SR. TTEN [15] mitigates popularity bias caused by head items by refining the relevance modeling process, while EGDE [16] enhances post-trained embeddings to counteract representation quality degradation stemming from item popularity. ATSRec [17] tackles embedding degeneration by injecting adversarial perturbations into the embedding layer, thereby improving robustness. However, these methods overlook the latent relational structure among items. In contrast, ClusRefineRec introduces a clustering-based perspective to explicitly capture and exploit such inter-item semantic relationships.

## III. Preliminary

### A. Problem Statement

Let $\mathcal{U} = \{u_1, u_2, \ldots, u_{|\mathcal{U}|}\}$ denote the set of users, and $\mathcal{V} = \{v_1, v_2, \ldots, v_{|\mathcal{V}|}\}$ denote the set of items. For each user $u \in \mathcal{U}$, the interaction sequence in chronological order is represented as $S^{(u)} = [v_1^{(u)}, v_2^{(u)}, \ldots, v_n^{(u)}]$, where each $v_i^{(u)} \in \mathcal{V}$ denotes the item that user $u$ has interacted with at time step $i$. Given the sequence $S^{(u)}$ , sequential recommendation aims to predict the most likely next item $v^*$ at step $n+1$:

$$\underset{v^* \in \mathcal{V}}{\operatorname{argmax}} P_\theta \left( v_{n+1}^{(u)} = v^* \mid S^{(u)} \right), \tag{1}$$

where $\theta$ represents the model parameters. This objective selects the candidate item with the highest predicted probability as the recommendation.

### B. Sequential Recommendation Model

*1) Embedding Layer:* For each item, we initialize a trainable embedding matrix $E_{item} \in \mathbb{R}^{|\mathcal{V}| \times D}$, where $|\mathcal{V}|$ denotes the total number of items and $D$ is the embedding dimension. Given a user interaction sequence $S^{(u)} = [v_1^{(u)}, v_2^{(u)}, \ldots, v_n^{(u)}] \in \mathbb{R}^n$, the embedding layer maps this sequence to a corresponding sequence of item embedding vectors $E^{(u)} = [e_1^{(u)}, e_2^{(u)}, \ldots, e_n^{(u)}] \in \mathbb{R}^{n \times D}$, where $e_i^{(u)}$ denotes the embedding corresponding to item $v_i^{(u)}$.

*2) Sequential Recommendation Model:* To capture temporal dependencies in user interaction sequences, sequential recommendation model $f(\cdot)$ is applied to encode the item embedding sequence $E^{(u)}$ into contextualized hidden states:

$$H^{(u)} = f\Big(E^{(u)}\Big) = [\, h_1^{(u)}, h_2^{(u)}, \ldots, h_n^{(u)} \,]. \tag{2}$$

Typically, the final state $h_n^{(u)}$ serves to summarize the user's current preference for the purpose of next-item prediction.

*3) Prediction Layer and Training:* The contextual hidden state at the final time step, $h_n^{(u)} \in \mathbb{R}^{1 \times D}$, is used as the input to the prediction layer to produce a probability distribution over all candidate items. The next-item probability is computed as $P_\theta\Big(v_{n+1}^{(u)} = v^*\Big) = \operatorname{softmax}\Big(h_n^{(u)} E_{item}^\top\Big)$, where $E_{item}^\top \in \mathbb{R}^{D \times |\mathcal{V}|}$ denotes the transposed item embedding matrix. The model parameters are optimized by minimizing the Cross-Entropy (CE) loss [20]:

$$\mathcal{L}_{\mathrm{CE}} = -\sum_{u \in \mathcal{U}} \log P_\theta \Big( v_{n+1}^{(u)} = v^* \Big). \tag{3}$$

## IV. Methodology

Figure 1 illustrates the overall architecture of ClusRefineRec. The left part, Base Recommender, presents the standard pipeline of sequential recommendation and incorporates dynamic state fusion to enhance the final sequence representation. The middle part, Compact Positive Embeddings, introduces a clustering loss based on positive samples and learnable cluster centroids to improve semantic consistency in the embedding space. The right part, Iterative Embedding Refinement, progressively refines the overall embeddings under the guidance of the learned centroids.

### A. Dynamic State Fusion

Introduce a learnable weighting mechanism to enhance the expressiveness of the final sequence representation. A linear projection followed by a sigmoid activation is applied to the output states $H^{(u)}$ to produce non-negative importance scores for each position:

$$w^* = \sigma\Big(H^{(u)} W^\top + b\Big), \tag{4}$$

where $\sigma$ denotes the sigmoid function. A positional mask is then applied to remove invalid time steps introduced by 0 padding, producing the filtered weights $w^{mask}$. The valid weights are subsequently renormalized to form a stable and uniformly scaled distribution $w$ across the sequence:

$$w_i = \frac{w_i^{mask}}{\sum_{i=1}^{n} w_i^{mask} + \varepsilon}, i = 1, \ldots, n, \tag{5}$$

where $\varepsilon$ is a small constant added for numerical stability. Finally, the user representation for next-item prediction is obtained via a weighted aggregation of the hidden states:

$$z^{(u)} = \sum w \odot H^{(u)}. \tag{6}$$

Dynamic state fusion allows the model to adaptively emphasize informative time steps while suppressing padding noise, thereby producing a more expressive and discriminative representation for next-item recommendation.

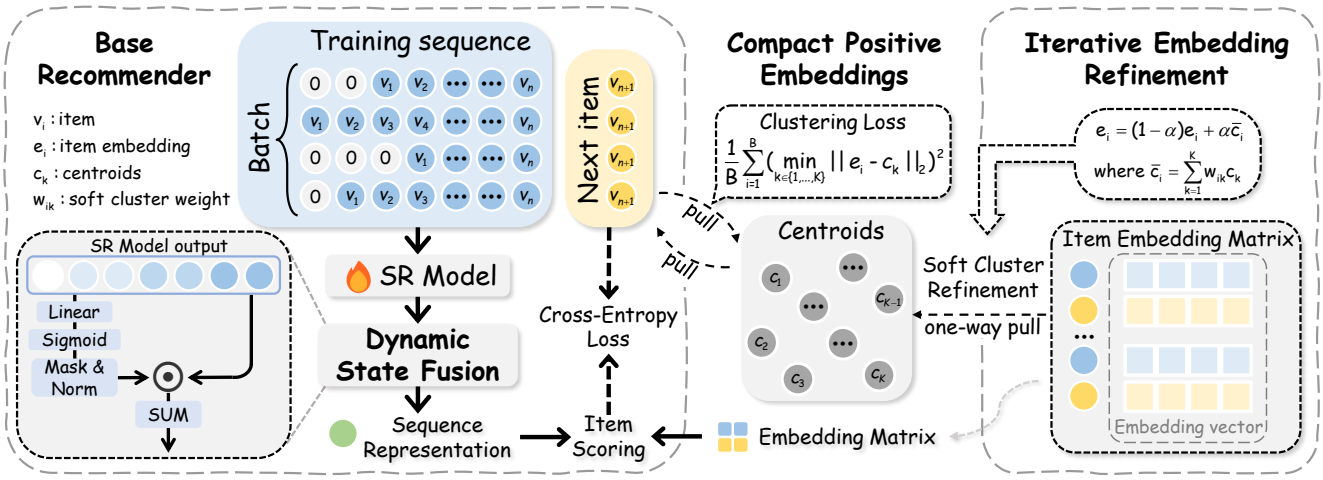



Fig. 1: Illustration of our proposed ClusRefineRec. (1) Dynamically assign weights to and fuse the output states generated by the SR model. (2) Introduce a clustering loss to jointly optimize positive item embeddings and cluster centroids, thereby uncovering the underlying semantic associations. (3) After each epoch, refine the global item embeddings through slight semantic alignment guided by the learned centroids.

### *B. Compact Positive Embeddings*

In this module, we define positive items as the ground-truth next items of each user sequence within a batch; consequently, the clustering process is naturally dominated by high-frequency and semantically stable items. Specifically, we introduce a set of $K$ learnable cluster centroids to further enhance the structural quality of positive item representations in the latent space. By explicitly optimizing the geometric distance between positive item embeddings and their nearest centroids, the model promotes stronger local compactness and semantic coherence among positive samples. Moreover, the clustering objective is fully decoupled from the main recommendation loss and optimized using an independent optimizer. This design effectively prevents gradient interference between recommendation and clustering signals, thereby improving optimization stability and enhancing target controllability.

Concretely, for a batch of positive item embeddings $\{e_i\}_{i=1}^{B}$, the pairwise distances to all cluster centroids $\{c_k\}_{k=1}^{K}$ are computed, and the minimum distance is used as the clustering feedback. The resulting Clustering Loss (CL) is formulated as:

$$\mathcal{L}_{\mathrm{CL}} = \frac{1}{B}\sum_{i=1}^{B}\left(\min_{1\le k\le K} \|e_i - c_k\|_2\right)^2, \tag{7}$$

where $e_i$ denotes the embedding of the positive item $v_i$, and $c_k$ denotes the $k$-th cluster centroid. In practice, the weight of the Clustering Loss is typically set to 0.1 to prevent drastic shifts in the embedding space.

This design enables the model to construct a semantically clearer and more discriminative representation subspace for positive items without interfering with the gradient flow of the main recommendation objective. Consequently, it provides a more robust representational foundation for downstream recommendation tasks. In practice, the clustering operation is applied exclusively to positive items rather than the entire set of item embeddings. This choice is motivated by the presence of sparsely interacted items in the full item set, whose embeddings are often insufficiently trained and exhibit limited semantic consistency. Incorporating such embeddings into the clustering process would introduce irrelevant variations and increase optimization noise, potentially leading to centroid drift and degraded convergence behavior.

From a representation learning perspective, embeddings of positive items are continuously refined under consistent user feedback, enabling them to capture stable and semantically meaningful latent structures. In contrast, items with sparse interactions suffer from insufficient supervisory signals, leading to under-trained and semantically ambiguous embeddings, which consequently exhibit greater uncertainty and significantly higher variance in the embedding space:

$$\mathrm{Var}(\mathcal{E}^-) \gg \mathrm{Var}(\mathcal{E}^+). \tag{8}$$

Here $\mathcal{E}^-$ denotes the embeddings of sparse items, while $\mathcal{E}^+$ represents the embeddings of positive items. Incorporating embeddings with heterogeneous distributions into a unified clustering process weakens the semantic structure learned from positive samples and perturbs the formation of centroids, causing the resulting cluster centers to deviate from regions that encode meaningful item similarity. Therefore, restricting clustering to positive items enables a more effective enhancement of semantic cohesion among truly related items.

### *C. Iterative Embedding Refinement*

In pursuit of enhancing the semantic structure of the embedding space, especially for long-tail items, we propose an iterative embedding refinement mechanism. This mechanism leverages cluster centroids as semantic anchors and dynamically aligns all item embeddings through a soft clustering strategy. Additionally, a chunk-wise updating scheme is adopted to

TABLE I: Performance improvements on three datasets. The entries labeled "w/ ours" denote the results of integrating ClusRefineRec into the corresponding models. "Improv." represents the relative improvement over the original performance.

| Dataset | Metric | GRU4Rec | **w/ours** | **Improv.** | SASRec | **w/ours** | **Improv.** | FMLP-Rec | **w/ours** | **Improv.** | FeaRec | **w/ours** | **Improv.** | SIGMA | **w/ours** | **Improv.** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Beauty | HR@5 | 0.3754 | 0.3871 | **3.12%** | 0.4075 | 0.4499 | **10.40%** | 0.4096 | 0.4397 | **7.35%** | 0.4085 | 0.4445 | **8.81%** | 0.4291 | 0.4451 | **3.73%** |
| | HR@10 | 0.4722 | 0.4893 | **3.62%** | 0.5037 | 0.5529 | **9.77%** | 0.5044 | 0.5376 | **6.58%** | 0.5043 | 0.5498 | **9.02%** | 0.5283 | 0.5471 | **3.56%** |
| | NDCG@5 | 0.2844 | 0.2935 | **3.20%** | 0.3076 | 0.3392 | **10.27%** | 0.3173 | 0.3385 | **6.68%** | 0.3118 | 0.3345 | **7.28%** | 0.3313 | 0.3406 | **2.81%** |
| | NDCG@10 | 0.3157 | 0.3265 | **3.42%** | 0.3387 | 0.3725 | **9.98%** | 0.3478 | 0.3701 | **6.41%** | 0.3427 | 0.3685 | **7.53%** | 0.3634 | 0.3735 | **2.78%** |
| | MRR@5 | 0.2543 | 0.2625 | **3.22%** | 0.2745 | 0.3026 | **10.24%** | 0.2867 | 0.3050 | **6.38%** | 0.2797 | 0.2980 | **6.54%** | 0.2988 | 0.3060 | **2.41%** |
| | MRR@10 | 0.2672 | 0.2761 | **3.33%** | 0.2873 | 0.3163 | **10.09%** | 0.2993 | 0.3179 | **6.21%** | 0.2924 | 0.3120 | **6.70%** | 0.3121 | 0.3195 | **2.37%** |
| Sports | HR@5 | 0.3855 | 0.4064 | **5.42%** | 0.3876 | 0.4189 | **8.08%** | 0.4012 | 0.4232 | **5.48%** | 0.4185 | 0.4309 | **2.96%** | 0.4009 | 0.4394 | **9.60%** |
| | HR@10 | 0.5122 | 0.5364 | **4.72%** | 0.5204 | 0.5556 | **6.76%** | 0.5255 | 0.5492 | **4.51%** | 0.5471 | 0.5628 | **2.87%** | 0.5380 | 0.5700 | **5.95%** |
| | NDCG@5 | 0.2823 | 0.2957 | **4.75%** | 0.2763 | 0.3032 | **9.74%** | 0.2963 | 0.3101 | **4.66%** | 0.3082 | 0.3136 | **1.75%** | 0.2875 | 0.3210 | **11.65%** |
| | NDCG@10 | 0.3232 | 0.3377 | **4.49%** | 0.3191 | 0.3473 | **8.84%** | 0.3364 | 0.3508 | **4.28%** | 0.3498 | 0.3563 | **1.86%** | 0.3316 | 0.3631 | **9.50%** |
| | MRR@5 | 0.2482 | 0.2592 | **4.43%** | 0.2397 | 0.2651 | **10.60%** | 0.2617 | 0.2728 | **4.24%** | 0.2718 | 0.2749 | **1.14%** | 0.2600 | 0.2819 | **8.42%** |
| | MRR@10 | 0.2651 | 0.2765 | **4.30%** | 0.2572 | 0.2832 | **10.11%** | 0.2781 | 0.2895 | **4.10%** | 0.2890 | 0.2925 | **1.21%** | 0.2782 | 0.2992 | **7.55%** |
| Toys | HR@5 | 0.3720 | 0.3881 | **4.33%** | 0.4086 | 0.4287 | **4.92%** | 0.3968 | 0.4264 | **7.46%** | 0.3955 | 0.4227 | **6.88%** | 0.4263 | 0.4394 | **3.07%** |
| | HR@10 | 0.4805 | 0.4935 | **2.71%** | 0.5116 | 0.5362 | **4.81%** | 0.4982 | 0.5278 | **5.94%** | 0.4979 | 0.5329 | **7.03%** | 0.5277 | 0.5450 | **3.28%** |
| | NDCG@5 | 0.2821 | 0.2927 | **3.76%** | 0.3097 | 0.3266 | **5.46%** | 0.3064 | 0.3250 | **6.07%** | 0.2974 | 0.3170 | **6.59%** | 0.3281 | 0.3337 | **1.71%** |
| | NDCG@10 | 0.3171 | 0.3267 | **3.03%** | 0.3431 | 0.3613 | **5.30%** | 0.3392 | 0.3577 | **5.45%** | 0.3304 | 0.3525 | **6.69%** | 0.3607 | 0.3679 | **2.00%** |
| | MRR@5 | 0.2525 | 0.2611 | **3.41%** | 0.2771 | 0.2927 | **5.63%** | 0.2765 | 0.2914 | **5.39%** | 0.2650 | 0.2820 | **6.42%** | 0.2956 | 0.2987 | **1.05%** |
| | MRR@10 | 0.2669 | 0.2750 | **3.03%** | 0.2908 | 0.3070 | **5.57%** | 0.2900 | 0.3049 | **5.14%** | 0.2786 | 0.2965 | **6.42%** | 0.3090 | 0.3128 | **1.23%** |

progressively refine all item embeddings, thereby effectively avoiding out-of-memory (OOM) issues caused by full-batch updates. Specifically, at the end of each training epoch, we compute soft assignment weights based on the cosine similarity between item embeddings and cluster centroids, and subsequently derive a weighted centroid $\bar{c}_i$, which serves as the directional guidance for embedding refinement. The computation process is as follows:

$$\bar{c}_i = \sum_{k=1}^{K} w_{ik} c_k, \quad w_{ik} = \frac{\exp\left(\frac{\cos(e_i, c_k)}{\tau}\right)}{\sum_{j=1}^{K} \exp\left(\frac{\cos(e_i, c_j)}{\tau}\right)}, \tag{9}$$

where $c_k$ denotes the $k$-th centroid, $w_{ik}$ is the soft assignment weight, and $\tau$ is the temperature factor controlling distribution sharpness — a smaller $\tau$ yields a more peaked distribution, emphasizing the closest centroids, while a larger $\tau$ produces a smoother, more uniform weighting across all centroids.

Rather than updating the centroids, we refine the item embeddings in place through a controlled, small-step semantic alignment toward their weighted centroids:

$$e_i = (1 - \alpha) e_i + \alpha \bar{c}_i, \tag{10}$$

where $\alpha$ is a small alignment coefficient that ensures stable refinement, typically set in the range $10^{-4}$ to $10^{-3}$, preventing excessive perturbations while enabling precise adaptation of the embedding space.

This procedure progressively imposes semantic compactness and improves representation quality while maintaining diversity. More importantly, it particularly benefits long-tail and infrequently interacted items, whose embeddings are generally under-optimized in conventional training. Moreover, the refinement process is computationally lightweight, differentiable-free, and can be seamlessly integrated into existing recommendation models as a plug-and-play module.

## V. Experiments

### A. Experiment Setting

**Datasets.** We conduct comprehensive experiments on three real-world datasets: *Amazon Beauty* [21] (22,363 users, 12,101 items, 198,502 interactions), *Amazon Sports* [21] (35,598 users, 18,357 items, 296,337 interactions), and *Amazon Toys* [21] (19,412 users, 11,924 items, 167,597 interactions).

**Evaluation Metrics.** We evaluate model performance using the *leave-one-out* strategy: for each user, we reserve the last interaction for testing, the second last for validation, and use the remaining interactions for training. Following common practice [7], [22], [23], each test item is paired with 100 uniformly sampled negative items that the user has not interacted with. For ranking evaluation, we report top-$k$ recommendation performance using Hit Rate (HR@$k$), Normalized Discounted Cumulative Gain (NDCG@$k$), and Mean Reciprocal Rank (MRR@$k$), where $k$ is set to 5 and 10.

**Baselines.** We select five representative state-of-the-art deep sequential recommendation models as baselines, including the RNN-based model **GRU4Rec** [5], the pure MLP-based model **FMLP-Rec** [18], the classical attention-based model **SASRec** [7], the Transformer-derived model **FeaRec** [8], and the Mamba-based model **SIGMA** [10].

**Implementation Details.** All baseline models are implemented using the widely adopted open-source RecBole framework [24]. To ensure a fair comparison, the embedding dimension is consistently set to 16 across all models. All experiments are conducted on a single NVIDIA RTX 4090 GPU with 24 GB of memory and a batch size of 2048. Other implementation details strictly follow the original source code.

### B. Overall Performance Comparison

The experimental results for the original sequential recommendation models and their enhanced versions with our method are reported in Table I. As shown, integrating the

TABLE II: Model performance on different interaction histories. The entries labeled "w/ ours" denote the results of integrating our proposed ClusRefineRec into the corresponding models. "Improv." represents the relative improvement over the original performance. The backbone network is SASRec.

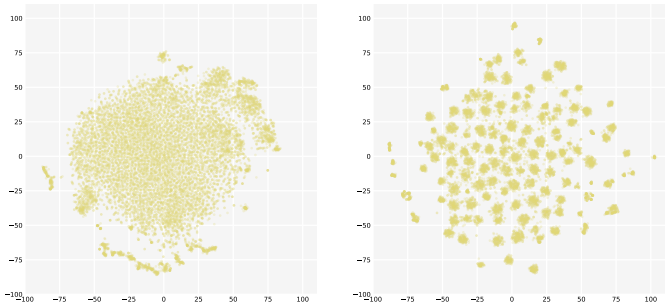
| Sequence Length | | Sparse ($\|s_u\| = 5$) | | | Dense ($\|s_u\| > 5$) | | |
|---|---|---|---|---|---|---|---|
| Dataset | Metric | Original | **w/ours** | **Improv.** | Original | **w/ours** | **Improv.** |
| Beauty | HR@5 | 0.4062 | 0.4370 | **7.58%** | 0.4578 | 0.4761 | **4.00%** |
| | HR@10 | 0.4993 | 0.5392 | **7.99%** | 0.5552 | 0.5819 | **4.81%** |
| | NDCG@5 | 0.3120 | 0.3304 | **5.90%** | 0.3477 | 0.3482 | **0.14%** |
| | NDCG@10 | 0.3421 | 0.3635 | **6.26%** | 0.3791 | 0.3826 | **0.92%** |
| | MRR@5 | 0.2808 | 0.2951 | **5.09%** | 0.3112 | 0.3129 | **0.55%** |
| | MRR@10 | 0.2932 | 0.3087 | **5.29%** | 0.3241 | 0.3251 | **0.31%** |
| Sports | HR@5 | 0.3714 | 0.4133 | **11.28%** | 0.3648 | 0.3887 | **6.55%** |
| | HR@10 | 0.5021 | 0.5487 | **9.28%** | 0.4936 | 0.5285 | **7.07%** |
| | NDCG@5 | 0.2640 | 0.2989 | **13.22%** | 0.2540 | 0.2757 | **8.54%** |
| | NDCG@10 | 0.3061 | 0.3426 | **11.92%** | 0.2955 | 0.3208 | **8.56%** |
| | MRR@5 | 0.2286 | 0.2611 | **14.22%** | 0.2175 | 0.2385 | **9.66%** |
| | MRR@10 | 0.2459 | 0.2791 | **13.50%** | 0.2346 | 0.2681 | **14.28%** |
| Toys | HR@5 | 0.4036 | 0.4264 | **5.65%** | 0.4009 | 0.4221 | **5.29%** |
| | HR@10 | 0.5079 | 0.5328 | **4.90%** | 0.5191 | 0.5415 | **4.32%** |
| | NDCG@5 | 0.3064 | 0.3224 | **5.22%** | 0.2998 | 0.3070 | **2.40%** |
| | NDCG@10 | 0.3400 | 0.3568 | **4.94%** | 0.3396 | 0.3457 | **1.80%** |
| | MRR@5 | 0.2742 | 0.2879 | **5.00%** | 0.2610 | 0.2689 | **3.03%** |
| | MRR@10 | 0.2881 | 0.3021 | **4.86%** | 0.2785 | 0.2850 | **2.33%** |

proposed ClusRefineRec consistently boosts the performance of all five backbone models across all three datasets and evaluation metrics. Overall, ClusRefineRec delivers consistent relative gains, with average improvements ranging from 3% to 10%. Notably, the performance gains are particularly pronounced on attention-based models. For instance, SASRec exhibits an average improvement of around 10% across multiple metrics on the Beauty and Sports dataset. In contrast, models such as GRU4Rec benefit less from our method. We attribute this to the fact that ClusRefineRec builds upon meaningful embedding representations and further refines them through iterative optimization, while attention mechanisms are inherently better at capturing rich and more semantically meaningful information in item embeddings. This complementary relationship enables stronger synergy between our refinement strategy and attention-based architectures.

These observations demonstrate that ClusRefineRec not only effectively enhances model performance but also serves as a universally applicable and scalable refinement module that can be seamlessly integrated into mainstream sequential recommendation architectures.

### C. Performance under Sparse and Dense Interaction Histories

In line with previous studies [25], [26], we evaluate the effectiveness of our method under sparse and dense interaction history settings within the same dataset. The results are reported in Table II. Specifically, in the sparse-history setting ($|s_u| = 5$), all interaction sequences are truncated to retain the most recent five items. In the dense-history setting ($|s_u| > 5$), we exclude sequences with fewer than five interactions. As shown, ClusRefineRec consistently improves model performance across all three datasets under both settings, demonstrating the applicability of our refinement strategy.



(a) SASRec (b) ClusRefineRec

Fig. 2: t-SNE of Item Embeddings

TABLE III: Evaluation of local embedding quality using mean and variance of $k$ nearest neighbor similarity ($k = 50$).

| Model | SASRec | | ClusRefineRec | |
|---|---|---|---|---|
| Metric | $\bar{s}$ | $\sigma_s^2$ | $\bar{s}$ | $\sigma_s^2$ |
| Beauty | 0.7687 | 0.0043 | 0.8203 | 0.0047 |
| Sports | 0.8801 | 0.0041 | 0.9279 | 0.0033 |
| Toys | 0.7605 | 0.0076 | 0.7906 | 0.0065 |

Moreover, performance gains are more pronounced under sparse interaction histories, particularly on the Sports dataset, where relative improvements exceed 10% across multiple metrics. With limited behavioral context, recommendation performance becomes more dependent on item embedding quality. By explicitly refining the semantic structure of item embeddings, ClusRefineRec captures latent item relationships and improves embedding quality, leading to more substantial gains in sparse-history settings.

### D. Cluster Analysis

In the original embedding space, semantically or behaviorally similar items are often dispersed due to sparse supervision and noisy co-occurrence signals. After applying ClusRefineRec, these items are pulled into cohesive micro clusters, indicating that our refinement process effectively captures latent semantic relations that were previously underrepresented, as visually illustrated in Fig. 2.

Importantly, the compact clusters do not lead to oversmoothing or loss of distinctiveness. Rather, clusters exhibit strong internal cohesion while remaining well separated, allowing the model to retain fine-grained preference distinctions and preserve meaningful semantic boundaries.

To evaluate the local clustering quality of item embeddings while preserving intra-cluster diversity, we compute the average similarity among each item's $k$ nearest neighbors. Let $e_i$ denote the embedding of item $v_i$, and $N_k(i)$ denote its top-$k$ nearest neighbors in cosine similarity. The average neighbor similarity for item $v_i$ is defined as:

$$s_i = \frac{1}{k} \sum_{j \in N_k(i)} \frac{e_i \cdot e_j}{\|e_i\| \|e_j\|}. \tag{11}$$

The global metrics are then computed as the mean and variance across all items:

$$\bar{s} = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} s_i, \quad \sigma_s^2 = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} (s_i - \bar{s})^2 \tag{12}$$

where $|\mathcal{V}|$ is the total number of items. As shown in Table III, when setting $k = 50$, the proposed method consistently improves the average intra-neighbor similarity $\bar{s}$ across all datasets, indicating stronger local semantic coherence in the embedding space. Meanwhile, the variance $\sigma_s^2$ remains comparable or slightly lower, demonstrating that the refinement does not cause over-smoothing or collapse, and the item embeddings still preserve sufficient discriminative diversity.

TABLE IV: Comparison of Positive-Only vs. Full-Item Refinement Approaches

| Dataset | Metric | SASRec | ClusRefineRec (All-Item Emb.) | ClusRefineRec | **Improv.** |
|---|---|---|---|---|---|
| Beauty | HR@10 | 0.5037 | 0.5482 | 0.5529 | **0.86%** |
| | NDCG@10 | 0.3387 | 0.3696 | 0.3725 | **0.78%** |
| | MRR@10 | 0.2873 | 0.3141 | 0.3163 | **0.70%** |
| Sports | HR@10 | 0.5204 | 0.5434 | 0.5556 | **2.25%** |
| | NDCG@10 | 0.3191 | 0.3377 | 0.3473 | **2.84%** |
| | MRR@10 | 0.2572 | 0.2743 | 0.2832 | **3.24%** |
| Toys | HR@10 | 0.5116 | 0.5315 | 0.5362 | **0.88%** |
| | NDCG@10 | 0.3431 | 0.3551 | 0.3613 | **1.75%** |
| | MRR@10 | 0.2908 | 0.3004 | 0.3070 | **2.20%** |

### *E. Impact of Positive-Only Refinement*

As shown in Table IV, refining the model using only the item embeddings from positive interactions consistently surpasses the use of all item embeddings across every dataset and evaluation metric. The advantage becomes especially clear on the sparsest dataset (Sports), where long-tail items account for a large portion of the catalog. This observation indicates that restricting refinement to positive samples is particularly beneficial in settings where many items are noisy, weakly informative, or rarely interacted with.

The performance gains are especially pronounced in MRR, reflecting notable improvements in top-ranked prediction accuracy. This suggests that positive-only refinement enhances the model's local ranking behavior and strengthens the semantic cohesion among frequently interacted items. In contrast, incorporating all item embeddings brings a large number of long-tail items into the clustering process, causing the centroids to drift away from semantically meaningful regions of the embedding space and ultimately weakening the structural consistency required for reliable recommendation.

Overall, the empirical results support the theoretical intuition: focusing on positive interactions yields cleaner semantic organization, more stable optimization, and consistently better recommendation performance, offering a simple yet broadly effective enhancement for sequential models.

TABLE V: Ablation study of SASRec on Beauty. Abbreviations in "Setting": CPE — Compact Positive Embeddings, CPE+IER — Compact Positive Embeddings with Iterative Embedding Refinement, DSF — Dynamic State Fusion.

| Setting | HR@5 | HR@10 | NDCG@5 | NDCG@10 | MRR@5 | MRR@10 |
|---|---|---|---|---|---|---|
| SASRec | 0.4075 | 0.5037 | 0.3076 | 0.3387 | 0.2745 | 0.2873 |
| only CPE | 0.4236 | 0.5269 | 0.3203 | 0.3535 | 0.2860 | 0.2997 |
| CPE + IER | 0.4296 | 0.5331 | 0.3256 | 0.3579 | 0.2902 | 0.3026 |
| only DSF | 0.4284 | 0.5305 | 0.3233 | 0.3552 | 0.2890 | 0.3018 |
| ClusRefineRec | **0.4499** | **0.5529** | **0.3392** | **0.3725** | **0.3026** | **0.3163** |

### *F. Ablation Study*

Table V reports the ablation results on SASRec using the Beauty dataset. As shown, the full ClusRefineRec framework achieves the best performance across all metrics, confirming the effectiveness and complementarity of its core components.

Applying CPE alone already yields notable improvements over the vanilla SASRec, underscoring its ability to enhance the semantic compactness of positive item embeddings. Incorporating IER (CPE+IER) further boosts performance by progressively refining embedding representations through iterative optimization. In addition, using DSF alone also leads to consistent gains, indicating its effectiveness in enhancing the expressiveness of the final sequence representations.

More importantly, integrating all components within ClusRefineRec results in the most substantial improvements, with relative gains of up to 10.40% on HR@5, 10.27% on NDCG@5, and 10.24% on MRR@5. This demonstrates that the modules are not merely additive but synergistic: DSF enhances the final representations by effectively extracting and leveraging the enriched semantic information produced. CPE then provides a compact and informative representational foundation, upon which IER progressively refines the embeddings to further elevate their overall quality.

### *G. Hyperparameter Study*

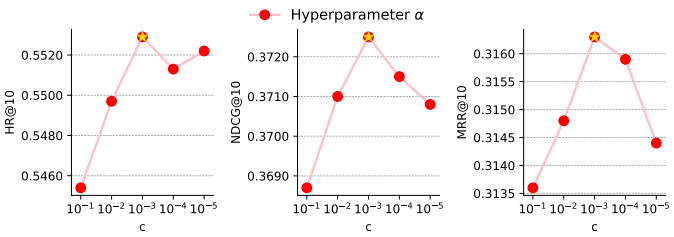



Fig. 3: Parameter study for $\alpha$ on Beauty.

In the hyperparameter sensitivity analysis on the Beauty dataset with the SASRec model, we evaluate the impact of the refinement magnitude coefficient $\alpha$, as shown in Fig. 3. Specifically, $\alpha$ is tested with values $10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}$. The results indicate that $\alpha = 10^{-3}$ achieves the best performance. It balances effective embedding refinement and training stability. Larger values (e.g., $10^{-1}$ or $10^{-2}$) cause

overly aggressive updates. These updates may disrupt the original semantic structure and harm convergence. Smaller values (e.g., $10^{-5}$) provide insufficient refinement.

In practice, $\alpha$ can be efficiently selected via a coarse-to-fine search on a logarithmic scale. Based on these observations, we recommend setting $\alpha$ in the range of $10^{-4}$ to $10^{-3}$ to balance stability and effectiveness.

## VI. Conclusion

We propose ClusRefineRec, a cluster-guided embedding refinement framework. It fuses temporal states, improves embedding consistency, and iteratively refines representations to enhance semantic coherence and preference modeling. Experiments on multiple datasets show its effectiveness, robustness, and easy integration with existing models.

## References


[1] J. Guo, P. Zhang, C. Li, X. Xie, Y. Zhang, and S. Kim, "Evolutionary preference learning via graph nested gru ode for session-based recommendation," in *Proceedings of the 31st ACM international conference on information & knowledge management*, 2022, pp. 624–634.

[2] F. Song, B. Chen, X. Zhao, H. Guo, and R. Tang, "Autoassign: Automatic shared embedding assignment in streaming recommendation," in *2022 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2022, pp. 458–467.

[3] K. Zhao, S. Liu, Q. Cai, X. Zhao, Z. Liu, D. Zheng, P. Jiang, and K. Gai, "Kuaisim: A comprehensive simulator for recommender systems," *Advances in Neural Information Processing Systems*, vol. 36, pp. 44 880–44 897, 2023.

[4] T. F. Boka, Z. Niu, and R. B. Neupane, "A survey of sequential recommendation systems: Techniques, evaluation, and future directions," *Information Systems*, vol. 125, p. 102427, 2024.

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

[6] F. Yuan, A. Karatzoglou, I. Arapakis, J. M. Jose, and X. He, "A simple convolutional generative network for next item recommendation," in *Proceedings of the twelfth ACM international conference on web search and data mining*, 2019, pp. 582–590.

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

[8] X. Du, H. Yuan, P. Zhao, J. Qu, F. Zhuang, G. Liu, Y. Liu, and V. S. Sheng, "Frequency enhanced hybrid attention network for sequential recommendation," in *Proceedings of the 46th international ACM SIGIR conference on research and development in information retrieval*, 2023, pp. 78–88.

[9] C. Liu, J. Lin, J. Wang, H. Liu, and J. Caverlee, "Mamba4rec: Towards efficient sequential recommendation with selective state space models," *arXiv preprint arXiv:2403.03900*, 2024.

[10] Z. Liu, Q. Liu, Y. Wang, W. Wang, P. Jia, M. Wang, Z. Liu, Y. Chang, and X. Zhao, "Sigma: Selective gated mamba for sequential recommendation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 12, 2025, pp. 12 264–12 272.

[11] S. Wang, L. Hu, Y. Wang, L. Cao, Q. Z. Sheng, and M. Orgun, "Sequential recommender systems: Challenges, progress and prospects," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 7 2019, pp. 6332–6338. [Online]. Available: https://doi.org/10.24963/ijcai.2019/883

[12] Q. Liu, J. Hu, Y. Xiao, X. Zhao, J. Gao, W. Wang, Q. Li, and J. Tang, "Multimodal recommender systems: A survey," *ACM Computing Surveys*, vol. 57, no. 2, pp. 1–17, 2024.

[13] R. He and J. McAuley, "Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering," in *proceedings of the 25th international conference on world wide web*, 2016, pp. 507–517.

[14] Y. Xu, G. Sun, J. Lu, L. Yang, X. Fang, and Y. Zhang, "Mmcdsr: a multimodal and cross-domain fusion framework for sequential recommendation," in *2025 International Joint Conference on Neural Networks (IJCNN)*, 2025, pp. 1–8.

[15] D. Kim, J. Park, and D. Kim, "Test-time embedding normalization for popularity bias mitigation," in *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 2023, pp. 4023–4027.

[16] G. Lee, K. Kim, and K. Shin, "Post-training embedding enhancement for long-tail recommendation," in *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, 2024, pp. 3857–3861.

[17] Y. Zhang and X. Chen, "Enhancing sequential recommendation modeling via adversarial training," in *2024 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2024, pp. 1–6.

[18] K. Zhou, H. Yu, W. X. Zhao, and J.-R. Wen, "Filter-enhanced mlp is all you need for sequential recommendation," in *Proceedings of the ACM web conference 2022*, 2022, pp. 2388–2399.

[19] A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," in *First Conference on Language Modeling*, 2024. [Online]. Available: https://openreview.net/forum?id=tEYskw1VY2

[20] Z. Zhang and M. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," *Advances in neural information processing systems*, vol. 31, 2018.

[21] J. McAuley, C. Targett, Q. Shi, and A. Van Den Hengel, "Image-based recommendations on styles and substitutes," in *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, 2015, pp. 43–52.

[22] S. M. Cho, E. Park, and S. Yoo, "Meantime: Mixture of attention mechanisms with multi-temporal embeddings for sequential recommendation," in *Proceedings of the 14th ACM Conference on recommender systems*, 2020, pp. 515–520.

[23] J. Jiang, P. Zhang, Y. Luo, C. Li, J. B. Kim, K. Zhang, S. Wang, X. Xie, and S. Kim, "Adamct: adaptive mixture of cnn-transformer for sequential recommendation," in *Proceedings of the 32nd ACM international conference on information and knowledge management*, 2023, pp. 976–986.

[24] W. X. Zhao, S. Mu, Y. Hou, Z. Lin, Y. Chen, X. Pan, K. Li, Y. Lu, H. Wang, C. Tian *et al.*, "Recbole: Towards a unified, comprehensive and efficient framework for recommendation algorithms," in *proceedings of the 30th acm international conference on information & knowledge management*, 2021, pp. 4653–4664.

[25] J. Jiang, Y. Luo, J. B. Kim, K. Zhang, and S. Kim, "Sequential recommendation with bidirectional chronological augmentation of transformer," *arXiv preprint arXiv:2112.06460*, vol. 90, 2021.

[26] Y. Dang, Y. Liu, E. Yang, G. Guo, L. Jiang, X. Wang, and J. Zhao, "Repeated padding for sequential recommendation," in *Proceedings of the 18th ACM Conference on Recommender Systems*, 2024, pp. 497–506.