Local Large Language Models for Recommendation

Yujin Jeon
jyj950309@snu.ac.kr
Seoul National Univ.
Seoul, Korea

Jooyoung Kim
gracekim15237@snu.ac.kr
Seoul National Univ.
Seoul, Korea

Joonseok Lee*
joonseok@snu.ac.kr
Seoul National Univ.
Seoul, Korea

Abstract

Unlike traditional classification tasks, recommendation is inherently subjective—whether an item should be suggested depends not only on user preferences and item semantics, but also on latent behavioral patterns and contextual cues. While recent LLM-based recommenders excel at modeling semantics and intent through generative reasoning, they often fail to capture collaborative signals and suffer from inefficiencies when applied globally across large interaction spaces. We propose **Local Large Language Models for Recommendation (L^{3}Rec)**, a novel model-agnostic framework that integrates collaborative filtering (CF) with generative LLMs through localized modeling. Our approach first applies a lightweight CF model to derive user and item embeddings, then clusters them into behaviorally coherent subgroups. Each cluster is assigned a dedicated generative LLM—referred to as a *local LLM*—trained only on its corresponding data subset. This enables fine-grained personalization while improving training efficiency through parallelism. At inference time, predictions from local models are aggregated via a fusion strategy, with a global CF fallback when needed. To the best of our knowledge, this is the first LLM-based recommendation framework to incorporate local collaborative structure. Experiments show that it achieves state-of-the-art performance with significantly better scalability and efficiency.

CCS Concepts

• **Information systems** → **Recommender systems**.

Keywords

Collaborative Filtering, Recommendation, Large Language Models

ACM Reference Format:
Yujin Jeon, Jooyoung Kim, and Joonseok Lee. 2025. Local Large Language Models for Recommendation. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management (CIKM '25), November 10–14, 2025, Seoul, Republic of Korea.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3746252.3761280

1 Introduction

Recommender systems are fundamental to modern information retrieval, powering personalized content delivery across domains such as e-commerce, streaming, education, and social platforms. Collaborative filtering (CF) [4, 8, 15, 16, 18, 25, 41] have long served as the backbone of recommender systems, offering effective modeling of user–item interactions through shared latent spaces. However, these methods often fall short in capturing the full complexity of user behavior, particularly in scenarios involving rich semantic contexts that go beyond user co-occurrence statistics.

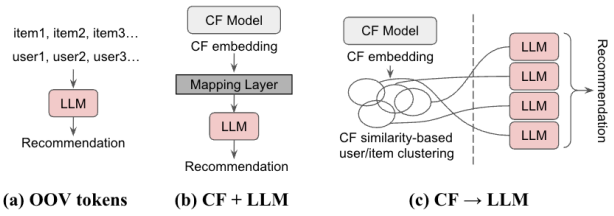


Figure 1: Different approaches to integrate collaborative filtering (CF) signals into large language models (LLMs) for recommendation.

In recent years, large language models (LLMs) have emerged as powerful tools for unifying and advancing recommendation tasks [7, 13, 17, 24, 27, 39, 50, 52, 54]. Their ability to understand and generate natural language enables prompt-based recommendation, multi-task generalization, and rich inference over textual metadata and interaction histories. Despite their promise, however, most existing LLMs are optimized for semantic understanding and generative fluency, but fall short in leveraging collaborative signals. Since users and items are typically represented as plain text tokens rather than distinct identifiers, these models often fail to capture fine-grained user–item co-preference patterns. Consequently, they tend to perform well in cold-start settings, but underperform in warm-start scenarios compared to traditional CF models [50].

In an effort to address this challenge, recent studies have explored incorporating collaborative signals into LLM-based recommenders. One straightforward approach is to introduce additional out-of-vocabulary (OOV) tokens into the LLM's vocabulary to represent each user and item explicitly, as illustrated in Fig. 1(a). This approach allows the model to directly learn distinct representations. However, adding separate tokens for each user and item significantly enlarges the token space, posing scalability issues for real-world applications. Specifically, the vocabulary size of a typical LLM (*e.g.*, 32,000 for T5) is quickly overwhelmed even by a recommendation system of moderate scale with $O(10^6)$ users and items, potentially degrading the quality of the pretrained vocabulary and leading to suboptimal performance [13, 48, 50].

An alternative strategy is to explicitly inject CF signals into the LLM recommendation pipeline as shown in Fig. 1(b). This typically involves learning user and item embeddings with a CF model and projecting them into the LLM representation space, avoiding OOV

*Corresponding author

CIKM '25, Seoul, Republic of Korea

ACM ISBN 979-8-4007-2040-6/2025/11
https://doi.org/10.1145/3746252.3761280

token inflation. However, this approach often suffers from misalignment issues between the CF embedding space and the LLM's semantic space [39, 52]. Since the two spaces are trained by fundamentally different objectives (user-item interaction modeling *vs.* language modeling), projection from one to the other often fails to bridge the gap and to preserve the relational structure between users and items. Moreover, without explicit user/item identifiers, it is difficult to precisely map CF signals into the LLM space, further reducing the efficacy of this integration [31, 50].

To overcome these limitations, we propose **Local Large Language Models for Recommendation (L^{3}Rec)**, a model-agnostic framework that leverages CF signals to guide the construction of localized LLMs tailored to coherent sub-communities of users and items, illustrated in Fig. 1(c). Our work is inspired by prior success in localized CF methods [5, 20, 22], which rest on the local low-rank assumption: user–item interactions are more predictable within a coherent and focused subgroup defined by shared interests or topics. This principle has been shown to improve the recommendation performance with matrix factorization [20, 22] and autoencoder-based models [5], and we extend this insight to LLM-based recommenders.

Specifically, we first obtain user and item similarity structure by training a lightweight CF model. Based on the learned similarity of the CF-based embeddings, we group users and items into coherent communities. Then, each resulting community is assigned a dedicated *local LLM*, trained solely on its corresponding subset of interactions. Unlike the global model covering all users and items at the same time, each local LLM needs only the users and items within the community it models. This allows each model to specialize in the collaborative and semantic nuances of its assigned group, removing nosiy signals from dissimilar users or items. Moreover, unlike prior approaches that rely on *explicit* token injection or embedding projection, our L^{3}Rec *implicitly* integrates collaborative signals through the discovered local community structure, avoiding the need to expand the token space or align the embedding spaces of CF model and LLM. Additionally, since the local LLMs are trained independently, the training process can be parallelized, further enhancing efficiency.

At inference, each user may be associated with one or more local models based on affinity, and predictions from these models are generated in parallel. The final recommendation is obtained by aggregating the outputs from the relevant local models. For users or items that do not belong to any identified cluster, a global CF model is employed as a fallback to ensure complete coverage.

Our contributions are summarized as follows:

- We propose a novel model-agnostic framework that integrates collaborative filtering–based local community assignment with LLMs to model fine-grained user-item preferences.
- The proposed framework significantly enhances scalability and efficiency by distributing training and inference across multiple local models, each operating on a small, focused subset of interactions.
- Through extensive experiments, we verify the effectiveness of the proposed method on both top-N and sequential recommendation tasks.

2 Related Work

LLM-based Recommendation. Large language models (LLMs) have recently gained attention in the recommender systems community for their ability to unify diverse tasks under a text-to-text generation paradigm. Early studies [11, 27, 36, 43, 51] explored leveraging pretrained language models such as GPT and BERT by reformatting user-item interactions as natural language sequences. For instance, GPTRec [36] utilized GPT-2 for generative sequential recommendation, representing item IDs as sub-token sequences to perform next-item prediction in an autoregressive fashion. Other works [12, 27, 47] demonstrated that LLMs can serve as recommender models, even in zero-shot or prompt-based settings.

While these approaches highlight the potential of generative language models for recommendation, they also expose several key limitations. A primary concern is that most existing approaches [1, 3, 7, 11, 12, 24, 27, 32, 36, 45, 51] heavily depend on item textual content, representing users and items purely through natural language descriptions rather than unique identifiers. This reliance on semantic representations limits the use of CF signals—an essential component for modeling implicit user preferences—thereby weakening personalization in recommendation scenarios.

To overcome these limitations, recent works have sought to incorporate collaborative signals more directly into LLM-based recommenders. One approach involves introducing Out-Of-Vocabulary (OOV) tokens [13, 54] or employing vector quantization techniques [10, 37, 38, 42, 52] to encode each user and item as unique identifiers. However, these strategies significantly increase the token vocabulary size, leading to scalability challenges and degraded model efficiency. Other lines of research [2, 26, 29, 30, 39, 46, 50] integrate explicit collaborative filtering (CF) models alongside LLMs. For instance, CoLLM [50] learns user and item embeddings using a CF model and maps them into the LLM space via a multilayer perceptron (MLP). However, such approaches often face alignment challenges between the CF embedding space and the LLM representation space due to the fundamentally different training objectives of the two models [31].

Another major concern is the high computational cost of LLM-based recommenders, which poses challenges for real-time deployment and scalability [27, 28]. The large size of LLMs results in slow inference times and high computational costs during training. POD [24] attempts to alleviate this by introducing task-alternated training and using continuous task-specific tokens at inference. However, training a global LLM on the full interaction set remains resource-heavy and can hinder the model's ability to learn fine-grained preference patterns.

Our framework addresses these challenges by constructing local user-item communities based on the embeddings learned from a CF model, and training a dedicated local LLM for each community using only its subset of interactions. This not only reduces the training time, but also allows us to incorporate collaborative signals effectively without requiring an enlarged token vocabulary or complex alignment mechanisms. In contrast to global LLMs that serve all users uniformly, our localized modeling approach enables specialization over user segments with similar preferences, resulting in both improved efficiency and enhanced recommendation performance.

Local Collaborative Filtering. Traditional collaborative filtering methods, such as matrix factorization [18], typically assume a global latent space in which all users and items are embedded. This implies that a single set of latent dimensions is used to represent the preferences of all users and the characteristics of all items. While effective for capturing broad trends, this global assumption often fails to model the rich diversity of user behaviors—particularly when preferences are non-uniformly distributed or multi-modal. As a result, global models tend to overfit popular preferences and underperform on niche interests or minority user groups.

Local collaborative filtering (CF) methods [23] were introduced to address the limitations of global models. These approaches learn multiple specialized models over subsets of similar users or items, allowing each model to capture fine-grained patterns within localized regions of the interaction space. A foundational example is LLORMA [19–21], which approximates the rating matrix using a set of overlapping low-rank models, each trained on a neighborhood of users and items. This framework relaxes the global low-rank assumption and allows for more expressive modeling of local preferences. While this improved accuracy by capturing localized variation, its performance gains were largely attributed to ensemble effects and it lacked global latent factors, limiting its ability to generalize to sparse regions.

To mitigate this, Subsequent research proposed hybrid models such as GLSLIM [35] and sGLSVD [6] which introduce global-local frameworks, combining a shared global model with user cluster–specific components to balance generalization and specialization. However, they still rely on linear modeling and hard clustering, which may not fully capture overlapping or nonlinear structures in user behavior.

LOCA [5] advanced this direction by introducing local autoencoders and a coverage-based community discovery mechanism. This ensemble of small, coherent models enables nonlinear representation learning while improving coverage and personalization. Building on this motivation, our work extends the notion of locality into the domain of large language models (LLMs) for recommendation. Instead of relying on autoencoders, we train a dedicated generative LLM for each user–item community, discovered via collaborative filtering (CF) embeddings. Unlike LOCA, which relies on continuous latent representations, our approach uses pretrained language models and leverages textual reasoning over interaction data, achieving enhanced recommendation quality while maintaining efficiency through localized training.

3 Problem Formulation

Given a set of n users $\mathcal{U} = \{u_1, \ldots, u_n\}$ and a set of m items $\mathcal{V} = \{v_1, \ldots, v_m\}$, each user interacts with a subset of items over time. The recommendation task aims to predict the N most probable items that a user is likely to interact with next, given their historical behavior. Here, the historical user interactions can be provided either as an unordered set or as a temporal sequence. The former setting has been studied under the name of *top-N recommendation*, while the latter is called *sequential recommendation* task.

Formally, each user $u \in \mathcal{U}$ is associated with an interaction sequence $\mathcal{S}_u = [v_1, v_2, \ldots, v_t]$, where each $v_i \in \mathcal{V}$ denotes an item previously engaged by the user. The objective is to conditionally generate a ranked list $\hat{R}_u$ of items from $\mathcal{V}$ that are most relevant to u based on $\mathcal{S}_u$. The top-N recommendation task takes $\mathcal{S}_u$ as a set, instead of an ordered sequence. We evaluate our proposed approach under both experimental settings.

4 The Proposed Method

Our proposed framework takes a divide-and-conquer strategy on the recommendation task, following the seminal papers in local recommendation models [5, 20, 22]. That is, we divide the entire set of users and items into several subsets of them, sharing common tastes and characteristics, derived through collaborative filtering and a tailored community assignment strategy (Divide; Sec. 4.1). Then, we train a specialized generative LLM for each localized sub-community of users and items, using only the interactions observed within this community (Conquer; Sec. 4.2). Once multiple local models are trained, we predict preference by a user on an item by aggregating predictions from both local and global models (Combine; Sec. 4.3). An overview of the entire framework is illustrated in Fig. 2.

4.1 Local Community Discovery

We discover local communities sharing similar tastes in a bottom-up fashion, following [5, 20]. That is, a user is chosen as an anchor, and other users relevant to the anchor participate in the local community centered on this anchor point. In order to collect these relevant users, we begin by training a lightweight collaborative filtering (CF) model to obtain initial embeddings. This model can be any standard CF method. (In our experiments, we adopt MultVAE [25].) The resulting model also serves as the global backbone $\mathcal{M}_{\text{global}}$ for users not assigned to any local community during the local assignment process.

To construct q local user communities, we first select q anchor users $\mathcal{A} = \{a^{(1)}, \ldots, a^{(q)}\}$ to serve as representatives. Anchors are selected using a greedy coverage-based algorithm [5] that iteratively selects users connected to the largest number of uncovered users in a similarity graph, thereby maximizing the coverage of local models.

For each anchor user $a^{(j)}$, we compute the similarity between its embedding $\mathbf{a}^{(j)}$ and each user $u \in \mathcal{U}$ based on the scaled arccosine distance:

$$s_u^{(j)} = \text{dist}(\mathbf{a}^{(j)}, \mathbf{u}) = \arccos\left(\frac{\mathbf{a}^{(j)} \cdot \mathbf{u}}{\|\mathbf{a}^{(j)}\| \cdot \|\mathbf{u}\|}\right). \tag{1}$$

Then, we compute a non-negative affinity weight $w_u^{(j)}$ between user u and anchor $a^{(j)}$ using a kernel function K_h. This weight reflects how closely the user u is associated with the anchor $a^{(j)}$. Formally,

$$w_u^{(j)} = K_h(s_u^{(j)}) \propto (1 - (s_u^{(j)})^2)\, \mathbf{1}[s_u^{(j)} < h], \tag{2}$$

where $\mathbf{1}[\cdot]$ denotes the indicator function. The choice of the kernel K_h is a hyperparameter, and we adopt the Epanechnikov kernel following [20]. The similarity threshold h determines the neighborhood size, allowing only users with sufficient similarity to contribute to the community. Users with non-zero affinity to an anchor are grouped into the corresponding community $\mathcal{U}_j = \{u \mid w_u^{(j)} > 0\}$.

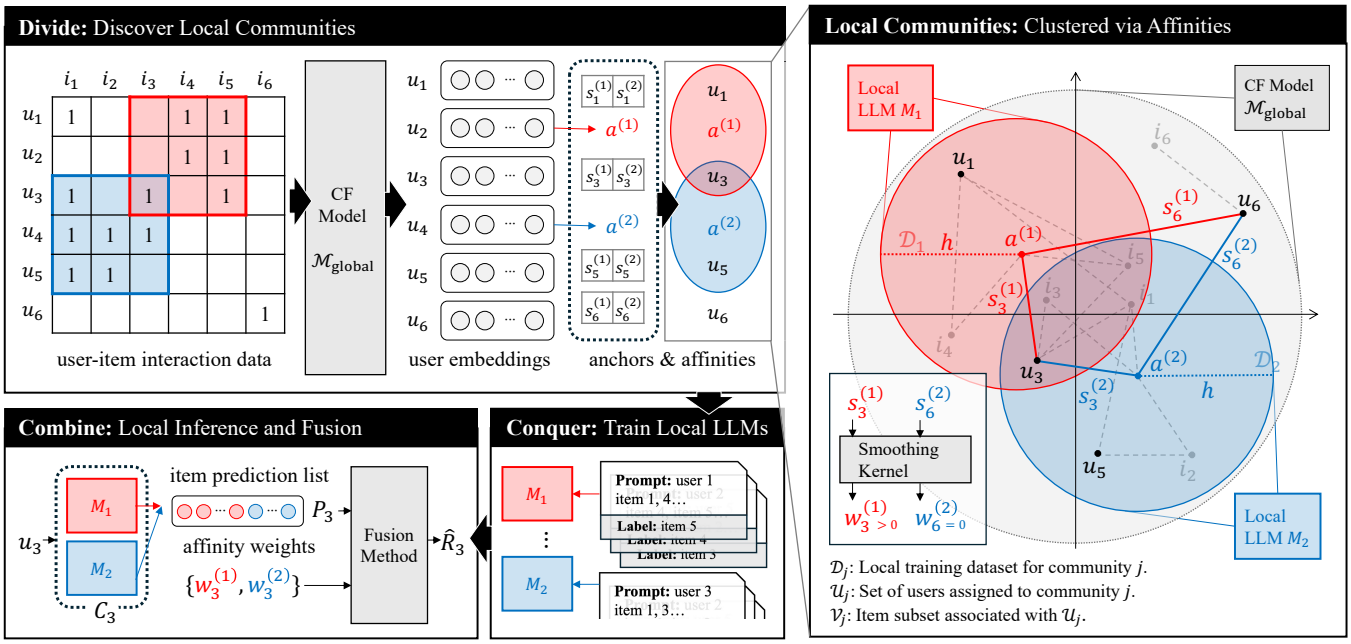
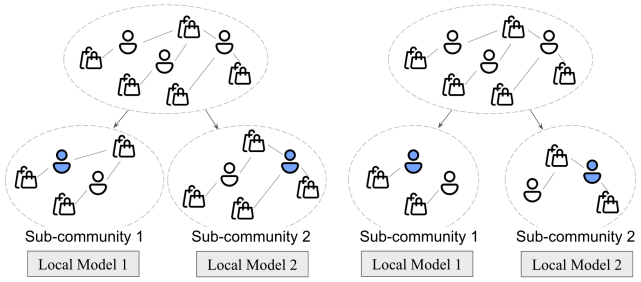


Figure 2: Overview of the proposed L^{3}Rec framework, following a divide–conquer–combine strategy.



(a) User-only assignment **(b) User-and-Item assignment**

Figure 3: Two variants of local community assignment. The user in blue indicates the anchor user.

This procedure results in q potentially overlapping user communities $\{\mathcal{U}_1, \ldots, \mathcal{U}_q\}$, each corresponding to a distinct local LLM that handles training and inference for its assigned users. The affinity weights $w_u^{(j)}$ can be reused at inference time to enable personalized prediction fusion across the local models (Sec. 4.3.2).

In addition to assigning users to local communities, our approach can be further extended to the local community construction with items as well. Specifically, for each user community $\mathcal{U}_j$, we optionally define a corresponding item set $\mathcal{V}_j$ which consists of items that are similar to the anchor user $a^{(j)}$. (Recall that users and items are embedded in the common latent space with collaborative filtering.) This results in more fine-grained and focused local datasets, where both users and items are related in a collaborative manner. Because both user and item spaces are narrowed, this variant produces more specialized local subgroups.

Note that assigning both users and items to local communities limits the model's ability to perform sequential recommendation. Since each local dataset resulting from this assignment strategy contains only a subset of a user's interaction history, the continuity of item sequences is disrupted, hindering the temporal modeling essential for sequential tasks.

Accordingly, we define two variants of our framework, illustrated in Fig. 3:

- **User-only assignment**: each local model is trained on all interactions of users within the community, including their full item histories. This supports both Top-N and sequential recommendation.
- **User-and-item assignment**: each local model is trained only on interactions between users and a subset of similar items. This variant is applicable only to Top-N recommendation, as it sacrifices temporal coherence.

We evaluate both strategies in Sec. 5.3 to investigate the trade-off between task generality and the compactness of local training sets.

4.2 Training Local LLMs

For the j-th local community constructed in Sec. 4.1 (with $j = 1, \cdots, q$), we define a corresponding local interaction dataset $\mathcal{D}_j$ and fine-tune a local LLM $\mathcal{M}_j$ on it. Note that any generative LLM recommendation model with support for fine-tuning can be adopted as $\mathcal{M}_j$.

Each training sample $(x, y) \in \mathcal{D}_j$ consists of a textual prompt x that encodes the user's interaction history and the target item y that the user has actually interacted after the given history. The training objective is to minimize the cross-entropy loss commonly used in generative LLMs:

$$\mathcal{L}_{\text{local}}^{(j)} = - \sum_{(x_j, y_j) \in \mathcal{D}_j} \log p_{\mathcal{M}_j}(y_j \mid x_j). \tag{3}$$

Algorithm 1 Inference with Local LLMs and Weighted Fusion

Require: Batch of user embeddings $\mathcal{U} = \{\mathbf{u}_1, \ldots, \mathbf{u}_B\}$; local models $\{\mathcal{M}_1, \ldots, \mathcal{M}_q\}$; affinity weights $\{w_i^{(j)}\}$; global fallback model $\mathcal{M}_{\text{global}}$; fusion function `Fuse()`

Ensure: Final recommendations $\{\hat{R}_1, \ldots, \hat{R}_B\}$

for each user embedding $\mathbf{u}_i \in \mathcal{U}$ **do**
  $C_i \leftarrow$ `GetRelevantCommunities`$(\mathbf{u}_i)$ // local communities with $w_i^{(j)} > 0$
  Initialize item prediction list $P_i \leftarrow \emptyset$
  for all communities $j \in C_i$ **in parallel do**
    $x \leftarrow$ `ConstructPrompt`$(\mathbf{u}_i)$
    $\hat{y}_j \leftarrow \mathcal{M}_j(x)$
    Extract predicted items and ranks from $\hat{y}_j$ and add to P_i
  end for
  if $P_i \neq \emptyset$ **then**
    $\hat{R}_i \leftarrow$ `Fuse`$(P_i, \{w_i^{(j)}\})$
  else
    $\hat{R}_i \leftarrow \mathcal{M}_{\text{global}}(\mathbf{u}_i)$
  end if
end for
return $\{\hat{R}_1, \ldots, \hat{R}_B\}$

Since our framework is model-agnostic, the exact form of the loss can be adapted depending on the specific LLM-based recommender used. As each local model $\mathcal{M}_j$ does not require information from other local models, we can train all of them in parallel. Since each training set $\mathcal{D}_j$ is relatively smaller than the entire data, training each model is usually faster. Under a highly distributed training setup, training our model can be even faster than training a huge single global model [20]. We empirically demonstrate this in Sec. 5.4. To sum up, this strategy maintains scalability while allowing each model to specialize in its respective user–item distribution.

4.3 Local Inference and Fusion

At inference, each local model predicts preference of candidate items by the target user based on their local information in parallel (Sec. 4.3.1), and those estimated scores are aggregated by one of the proposed fusion strategies below (Sec. 4.3.2).

4.3.1 Parallel Inference with Local LLMs. Each user u can be associated with one or more communities based on their embedding similarity from the community assignment process in Sec. 4.1. Let $C(u)$ denote the set of relevant communities for user u. For each relevant community in $C(u)$, a dedicated local LLM $\mathcal{M}_j$ is assigned. Each local model $\mathcal{M}_j$ generates a recommendation score $p_j(u, i)$ for item $i \in \mathcal{V}_j$ conditioned on the prompt derived from the interaction history of user u:

$$p_j(u, i) = P_{\mathcal{M}_j}(i \mid \text{Prompt}(u)) \tag{4}$$

All relevant local models are queried in parallel, and each produces a ranked list of items based on their generation outputs.

4.3.2 Fusion Strategy for Final Recommendation. To obtain a unified Top-N recommendation list, we aggregate the outputs from multiple local models using either standard Reciprocal Rank Fusion (RRF) or our proposed weighted variant (wRRF), which incorporates user–community affinity.

Reciprocal Rank Fusion (RRF). For each item i retrieved from any local model $\mathcal{M}_j \in C(u)$, the RRF score is computed as:

$$\text{RRF}_u(i) = \sum_{j \in C(u)} \frac{1}{k + \text{rank}_j(u, i)}, \tag{5}$$

where $\text{rank}_j(u, i)$ is the rank of item i assigned by model $\mathcal{M}_j$, and k is a damping constant to prevent overly favoring top-ranked items.

Weighted Reciprocal Rank Fusion (wRRF). To account for the user's affinity to each community, each rank contribution is weighted by the community-specific affinity score $w_u^{(j)}$:

$$\text{wRRF}_u(i) = \sum_{j \in C(u)} w_u^{(j)} \cdot \frac{1}{k + \text{rank}_j(u, i)}. \tag{6}$$

Global Model Fallback. If a user u has no associated communities, we fall back to the global CF model $\mathcal{M}_{\text{global}}$. The global score is computed by

$$p_{\text{global}}(u, i) = P_\theta(\mathbf{i} \mid \mathbf{u}), \tag{7}$$

where $\mathbf{u}$ and $\mathbf{i}$ are the CF representations of user u and item i of $\mathcal{M}_{\text{global}}$.

Final Recommendation. The final recommendation list is constructed by aggregating all available scores and selecting the top-N items:

$$\hat{R}_u = \text{Top-}N \text{ items ranked by Fusion}_u(i) \tag{8}$$

This fusion framework ensures robust coverage, personalization, and sensitivity to user-specific community membership. The overall inference process is summarized in Algorithm 1.

5 Experiments

We conduct extensive experiments to answer the following research questions:

- RQ1: How does the proposed approach L^{3}Rec perform compared to existing methods?
- RQ2: Does the proposed local training strategy reduce computational cost while maintaining or improving accuracy?
- RQ3: How does each component of our framework contribute to the overall recommendation performance?

5.1 Experimental Settings

5.1.1 Datasets. We conduct experiments on three Amazon[1] [34] datasets: Sports & Outdoors, Beauty, and Toys & Games. Each record in this dataset is composed of a user ID, an item ID, a rating, a textual review, and the corresponding timestamp. The statistics of each dataset are presented in Tab. 1. To ensure consistency in

Table 1: Statistics of the datasets

Dataset	# User	# Item	# Interaction
Sports	35,598	18,357	296,337
Beauty	22,363	12,101	198,502
Toys	19,412	11,924	167,597

[1] http://jmcauley.ucsd.edu/data/amazon/

Table 2: Average number of users in each local community

Dataset	# User	Coverage
Sports	31,911	89.6%
Beauty	20,659	92.3%
Toys	16,471	84.8%

evaluation, we follow the data preparation and train-test splitting protocol established by previous works [7, 24].

5.1.2 Evaluation Metrics. To measure the performance of our recommendation models, we adopt two widely used ranking-based evaluation metrics: Hit Ratio (HR@K) and Normalized Discounted Cumulative Gain (NDCG@K), where $K \in \{5, 10\}$. These metrics are computed based on the ranked list of predicted items for each user and are standard in evaluating both top-K and sequential recommendation tasks.

5.1.3 Implementation Details. We utilize POD [24] as the local LLMs and MultVAE [25] as the global collaborative filtering model. Following prior baselines, we adopt T5-small as the backbone of the language models and train using a learning rate of 0.0005. All experiments are conducted in a parallel setting using NVIDIA A6000 GPU with 32GB of memory. Unless otherwise specified, we set the hyperparameters to $h = 1.0$ and $k = 0$ across all experiments. We fix the number of local models to $q = 10$, and report the average number of users assigned to each local model in Tab. 2.

5.2 Baselines

We compare the performance of our method with the following groups of baselines.

5.2.1 Sequential Recommendation.

- **CASER [44]:** Convolutional Sequence Embedding Recommendation (CASER) treats a user's recent item interactions as an image-like matrix and applies horizontal and vertical convolutional filters to capture sequential patterns.
- **HGN [33]:** Hierarchical Gating Networks (HGN) models both long- and short-term user preferences through gated recurrent units and a hierarchical attention mechanism to dynamically combine historical behaviors.
- **GRU4Rec [9]:** GRU4Rec is one of the earliest RNN-based recommenders that applies Gated Recurrent Units to model session-based sequences for next-item prediction.
- **BERT4Rec [43]:** BERT4Rec reformulates sequential recommendation as a bidirectional Masked Language Modeling (MLM) task, using Transformer encoders to learn item dependencies in both directions.
- **FDSA [49]:** Feature Disentangled Self-Attention (FDSA) enhances Transformer-based models by disentangling feature types (*e.g.*, temporal, positional) in self-attention layers to better model user behavior patterns.
- **SASRec [14]:** Self-Attentive Sequential Recommendation (SASRec) is a Transformer-based model that applies self-attention to capture dependencies between past items for next-item prediction.
- **S^3-Rec [53]:** Self-Supervised Sequential Recommendation (S^3-Rec) integrates auxiliary self-supervised objectives (*e.g.*, item masking, sequence permutation) into SASRec to enhance generalization under sparse data.
- **P5 [7]:** Pretrain, Personalized Prompt, and Predict Paradigm (P5) unifies various recommendation tasks under a language modeling framework by framing them as natural language prompts and responses.
- **POD [24]:** PrOmpt Distillation (POD) improves LLM-based recommendation by learning soft continuous prompts for each task, reducing the need for full model finetuning and enabling efficient multi-task learning.

5.2.2 Top-N Recommendation.

- **MF [18]:** Matrix Factorization (MF) is a foundational collaborative filtering approach that models user-item interactions through the dot product of latent embeddings. The training objective is optimized using Bayesian Personalized Ranking (BPR) [40], which is designed to enhance ranking performance by favoring observed interactions over unobserved ones.
- **MLP [4]:** Multi-Layer Perceptron (MLP) serves as a neural alternative to MF, projecting user/item representations into a shared space through multiple non-linear transformations. Similar to MF, the model is trained with the BPR loss.
- **P5 [7] :** Again, P5 reformulates the recommendation task as a natural language generation problem. It is trained to support five distinct recommendation scenarios, including both sequential and top-N recommendation tasks.
- **POD [24] :** Again, building upon P5, POD improves training and inference efficiency by introducing continuous prompts.

5.3 Overall Performance (RQ1)

We first compare the overall performance of our model with the aforementioned baselines on the top-N and sequential recommendation tasks. For our method, we report the results under two different community assignment strategies: the user-only assignment (U) and user-item assignment (UI). For the top-N recommendation task, we report results for both variants to examine the impact of the community assignment strategies. Note that for the UI variant, the hyperparameter h is tuned to 1.1, 1.0, and 1.3 for the Sports, Beauty and Toys dataset respectively. For the sequential recommendation task, we report results only for the user-only setting as discussed earlier in Sec. 4.1, due to the disruption of temporal interaction sequences when item clustering is applied. To aggregate predictions from local models, we employ Reciprocal Rank Fusion (RRF) as our integration strategy.

Tab. 3-4 compare the performance of competing methods in the sequential and top-N recommendation tasks, respectively. As shown in the tables, our method consistently achieves the state-of-the-art performance across all three datasets and evaluation metrics in both recommendation tasks. The improvements are particularly notable in the sequential recommendation task, where our model exhibits substantial gains - especially on the Beauty domain with over 20% improvement across all metrics. The improvement in the Toys domain is relatively modest, possibly due to its sparser user–item interactions, weaker collaborative signals, and higher variability in user intent and item semantic. Nevertheless, our model achieves the best performance across all evaluation metrics. This result demonstrates the effectiveness of our method in capturing

Table 3: Overall performance comparison on sequential recommendation (%)

Method	Sports				Beauty				Toys			
	HR@5	NDCG@5	HR@10	NDCG@10	HR@5	NDCG@5	HR@10	NDCG@10	HR@5	NDCG@5	HR@10	NDCG@10
Caser	1.16	0.72	1.94	0.97	2.05	1.31	3.47	1.76	1.66	1.07	2.70	1.41
HGN	1.89	1.20	3.13	1.59	3.25	2.06	5.12	2.66	3.21	2.21	4.97	2.77
GRU4Rec	1.29	0.86	2.04	1.10	1.64	0.99	2.83	1.37	0.97	0.59	1.76	0.84
BERT4Rec	1.15	0.75	1.91	0.99	2.03	1.24	3.47	1.70	1.16	0.71	2.03	0.99
FDSA	1.82	1.22	2.88	1.56	2.67	1.63	4.07	2.08	2.28	1.40	3.81	1.89
SASRec	2.33	1.54	3.50	1.92	3.87	2.49	6.05	3.18	4.63	3.06	6.75	3.74
S^3-Rec	251	1.61	3.85	2.04	3.87	2.44	6.47	3.27	4.43	2.94	7.00	3.76
P5	2.72	1.69	3.61	1.98	5.03	3.70	6.59	4.21	6.48	5.67	7.09	5.87
POD	4.96	3.96	5.76	4.19	5.37	3.95	6.88	4.43	6.91	5.99	7.42	6.10
L^3Rec (U)	**5.54**	**4.47**	**6.46**	**4.77**	**6.62**	**4.97**	**8.37**	**5.53**	**7.21**	**6.29**	**7.67**	**6.44**
Gain	11.7%	12.9%	12.2%	13.8%	23.3%	25.8%	21.7%	24.8%	4.3%	5.0%	3.4%	5.6%

Table 4: Overall performance comparison on top-N recommendation (%)

Method	Sports				Beauty				Toys			
	HR@5	NDCG@5	HR@10	NDCG@10	HR@5	NDCG@5	HR@10	NDCG@10	HR@5	NDCG@5	HR@10	NDCG@10
MF	14.04	8.48	25.63	12.20	14.26	8.57	25.73	12.24	10.66	6.41	20.03	9.40
MLP	15.20	9.27	26.71	12.96	13.92	8.48	25.42	12.15	11.42	6.88	20.77	9.40
P5	15.14	10.49	21.96	12.69	15.66	10.78	23.17	13.18	13.22	8.89	20.23	11.14
POD	20.86	15.06	28.73	17.56	19.26	13.91	26.70	16.29	14.33	10.09	20.82	12.15
L^3Rec (U)	23.19	16.99	30.88	19.47	21.80	15.95	29.70	18.49	**16.44**	**11.61**	**22.82**	**13.65**
L^3Rec (UI)	**26.88**	**19.68**	**35.19**	**22.35**	**22.28**	**16.07**	**30.12**	**18.59**	14.67	9.52	21.85	11.83
Gain	28.5%	30.6%	22.4%	27.3%	16.3%	16.8%	13.8%	15.6%	14.7%	15.1%	9.6%	12.3%

Table 5: Comparison of training efficiency

Method	Sports		Beauty		Toys	
	Time	Epochs	Time	Epochs	Time	Epochs
POD	6h 40m	29	4h 29m	26	2h 51m	23
L^3Rec	5h 50m	23	3h 49m	24	2h 32m	20
Gain	12.5%	20.7%	14.9%	7.7%	11.1%	13.0%

temporal dynamics and modeling user preferences within localized interaction patterns.

In the top-N recommendation setting, our method outperforms the baselines in all scenarios, with the L^3Rec (UI) variant achieving the best results in most cases. This confirms that modeling both users and items that are similar within local communities enhances the ability to capture fine-grained collaborative signals, leading to more precise ranking–especially for the top-ranked items, as reflected in the strong HR@5 and NDCG@5 scores. In the Toys domain, on the other hand, we observe that L^3Rec (U) outperforms L^3Rec (UI). This can be attributed to the higher sparsity of user-item interactions in this dataset, which makes item clustering less stable or meaningful. In such cases, restricting clustering to users helps preserve more complete interaction histories and avoids over-fragmentation of the data. Nonetheless, our model still consistently outperforms all baselines, indicating strong robustness and generalizability even in data-scarce environments.

5.4 Training Efficiency (RQ2)

We compare the overall training time and the number of training epochs on each dataset with the POD baseline. For our method, the reported training time represents the total duration required to train all $q = 10$ local models in parallel under user-only assignment setting. The early stopping criterion was uniformly set to 5 epochs across all experiments.

As shown in Tab. 5, our proposed method takes even less training time compared to the POD baseline, across all domains. The number of training epochs is reduced by 3 on average, accounting for the reduced complexity of the interaction patterns within each sub-community.

Naturally, the training time per epoch is also shorter. These consistent reductions highlight the efficiency of our method in terms of both scalability and resource utilization, making it a practical choice for real-world deployment without compromising performance.

Table 6: Effect of different fusion methods for sequential recommendation performance (%) on Amazon Sports

Fusion	HR@5	NDCG@5	HR@10	NDCG@10
Count	5.32	4.29	6.23	4.58
RRF	5.54	4.47	6.46	4.77
wRRF(CF)	5.47	4.43	6.39	4.73
wRRF(LLM)	**6.01**	**5.05**	**6.80**	**5.30**

5.5 Ablation Study (RQ3)

5.5.1 Effect of the Fusion Strategy. We evaluate four fusion strategies: Count, Reciprocal Rank Fusion (RRF), weighted RRF with CF-based weights (wRRF(CF)), and weighted RRF with LLM-based weights (wRRF(LLM)). The Count method aggregates local predictions by summing item occurrences, while RRF applies rank-based aggregation. The wRRF variants extend RRF by incorporating user-specific weights. In wRRF(CF), the weight corresponds

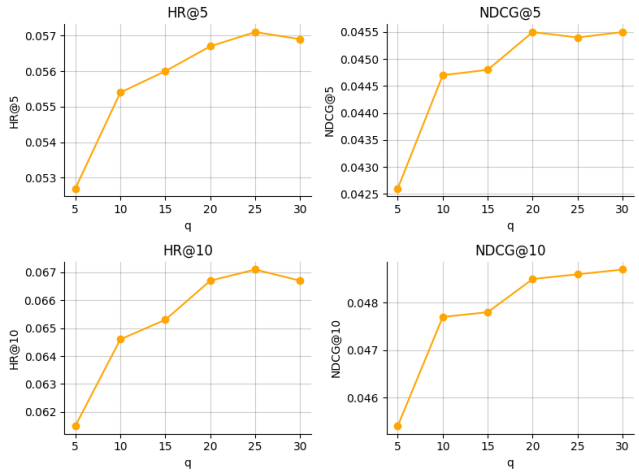


Figure 4: Sequential recommendation performance on Amazon Sports with various number of local models (q).

to the user's affinity score for each local model computed from collaborative filtering (CF) embeddings, as described in Sec. 4.3.2. In contrast, wRRF(LLM) uses the accuracy of each local model for each user—measured by NDCG@5 on the user's held-out interactions—as the weight. These accuracy scores are normalized across local models for each user via a softmax function.

As shown in Tab. 6, wRRF(LLM) achieves the best performance across all metrics, outperforming both the unweighted RRF and the CF-weighted variant. The weaker results of wRRF(CF) compared to wRRF(LLM) can be attributed to the representational gap between CF embeddings and LLM prediction space. CF-derived affinity scores may not align well with the semantic and generative reasoning patterns in LLM outputs, leading to suboptimal weight assignments. By contrast, wRRF(LLM) directly leverages evaluation-based accuracy signals, ensuring that models with stronger recent performance for a given user are prioritized in the fusion process.

These findings highlight that, while personalization in fusion can be beneficial, the source of the weighting signal is crucial. Using LLM-relevant performance metrics offers a more adaptive and model-aligned approach than relying solely on external embedding similarities. Future work could explore hybrid strategies that combine accuracy-based weighting with semantic similarity measures from LLM representations to further enhance adaptive fusion.

5.5.2 Effect of the Number of Local LLMs. We examine the impact of the number of local LLMs (q) on recommendation performance. The reported metrics in this section correspond to the RRF setting. As illustrated in Fig. 4, all evaluation metrics steadily improve as q increases, demonstrating the effectiveness of fine-grained user-item partitioning. By allowing each local model to specialize in a more homogeneous subset of users and items, the framework captures collaborative signals more effectively.

However, the improvement rate begins to plateau beyond $q = 20$, with only marginal gains observed between $q = 25$ and $q = 30$, aligned with the observations in traditional local models [22]. This diminishing return with a large number of local models implies a practical upper bound on the benefits of further partitioning.

These findings point to a trade-off between model accuracy and scalability. While increasing q improves the recommendation accuracy, the computational and management overhead also grows. Notably, even with a small number of local models (*e.g.*, $q = 5$), our method already yields substantial performance improvements over the baselines. This balance between efficiency and effectiveness makes our approach well-suited for scalable real-world deployment.

5.5.3 Effect of k. In Reciprocal Rank Fusion (RRF) and its weighted variant (wRRF), the damping constant k determines how much influence lower-ranked items exert in the final aggregated score. To assess its impact, we evaluate the sequential recommendation performance across three Amazon domains (Sports, Beauty, and Toys), varying k from 0 to 40. In the wRRF setting, the user-specific weights are derived from collaborative filtering (CF) models.

As illustrated in Fig. 5, increasing k generally leads to a decline in performance across all metrics and datasets, with the degradation being most pronounced in early precision metrics such as HR@5 and NDCG@5. RRF consistently outperforms wRRF across most k values, maintaining stronger and more stable performance. In contrast, wRRF shows greater sensitivity to the damping constant; its performance declines more steeply as k increases, particularly on metrics that emphasize rank position.

These findings reveal a trade-off between adaptivity and robustness. Although wRRF introduces user-specific weighting to enhance personalization, it appears more vulnerable to noise when the contribution from lower-ranked items is amplified. RRF, in contrast, provides more consistent performance across a wide range of k values. Furthermore, the superior results achieved with smaller k values underscore the importance of accurately ranking top items in recommendation scenarios. In practice, choosing a small k (*e.g.*, 0 or 10) offers a favorable balance between ranking precision and stability.

6 Case Study

We illustrate an example to demonstrate how our L^{3}Rec framework effectively incorporates collaborative signals into LLM-based prediction in Fig. 6. We focus on User 25 from the Amazon Beauty dataset, who has interacted with several items, predominantly perfume products. The ground truth target, Item 233, is a makeup kit from the brand *Smashbox*, which is semantically distinct from the user's prior interactions. The baseline model POD fails to retrieve this item within its top-10 predictions, while our model successfully identifies this preferred item.

During the local assignment phase, User 25 is assigned to multiple local communities (1, 3, 4, 5, 6, 8, 9, and 10). Among the corresponding local LLMs trained for each community, five of them($\mathcal{M}_1, \mathcal{M}_4, \mathcal{M}_5, \mathcal{M}_8, \mathcal{M}_{10}$) were able to place the target item within the top 10 predictions. Notably, all of these five local models also include User 7347, a user who shares a collaborative signal with User 25 through a common interaction with Item 216. While User 25 does not exhibit an explicit preference for makeup products, User 7347's interaction history shows a strong inclination toward them. Through this collaborative link, the relevant local models are able to infer a hidden preference of User 25—capturing relational

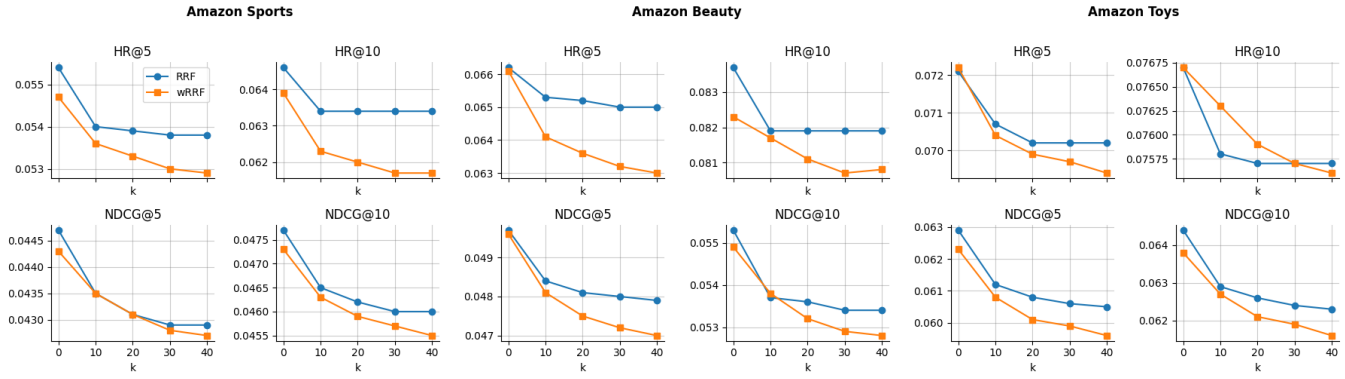


Figure 5: Effect of the damping constant k on sequential recommendation performance. Lower values of k give more weight to top-ranked items.

Local LLMs	Is User Included?		Local Prediction		Affinity Weight of User 25
	User 25	User 7347	Hit/Miss	Rank	
$\mathcal{M}_1$	✓	✓	**Hit**	**1**	0.26
$\mathcal{M}_2$	✗	-		-	-
$\mathcal{M}_3$	✓	✗	Miss	-	0.08
$\mathcal{M}_4$	✓	✓	**Hit**	**2**	0.49
$\mathcal{M}_5$	✓	✓	**Hit**	**8**	0.07
$\mathcal{M}_6$	✓	✗	Miss	-	0.08
$\mathcal{M}_7$	✗	-		-	-
$\mathcal{M}_8$	✓	✓	**Hit**	**1**	0.22
$\mathcal{M}_9$	✓	✗	Miss	-	0.21
$\mathcal{M}_{10}$	✓	✓	**Hit**	**7**	0.03

Fusion Method	Count	RRF	wRRF
Rank of Target (Item 233)	5	3	2

Figure 6: Case study of User 25 from Amazon Beauty dataset.

signals that are not apparent from content similarity alone with the help of smaller local community setting.

Furthermore, our proposed aggregation strategy enhances the final ranking of the target item. As shown in the table in Fig. 6, the rankings of Item 233 vary across local models, where $\mathcal{M}_1$, $\mathcal{M}_4$, and $\mathcal{M}_8$ ranked the highest with 1st, 2nd, and 1st, respectively. When using a simple count-based aggregation, the item is ranked only in the 5th position in the final prediction. In contrast, our rank-based strategies elevate the item into the top 3. Specifically in this case, the wRRF strategy predicts the target item with the highest rank, owing to the high user-affinity weight values assigned to the local models that assign the highest ranks to the target item.

This case highlights how our framework leverages collaborative signals in forming local communities, enabling the model to capture user–item relationships that are not apparent through semantic similarity alone. By aggregating insights from users with overlapping behavioral patterns, our method can successfully predict items that lie outside the immediate semantic space of a user's history.

7 Conclusion

In this work, we present a novel framework for recommendation using Local Large Language Models (LLMs), addressing the scalability and inefficiency challenges of existing generative LLM-based recommenders. By partitioning the interaction space into groups of similar users and items using a collaborative filtering backbone, we enable the training of local LLMs that better capture fine-grained collaborative signals within each group. This design not only allows for parallelized and faster training but also enhances recommendation quality by focusing each model on a more homogeneous subset of the data. Our experimental results highlight the potential of hybrid architectures that combine traditional recommendation techniques with the generative capabilities of LLMs. Future directions include exploring dynamic clustering strategies, adaptive fusion methods, and extending the framework to support more diverse recommendation scenarios such as multi-modal or conversational recommendation.

Acknowledgments

This work was supported by Samsung Electronics (IO240512-09881-01), Youlchon Foundation, NRF grants (RS-2021-NR05515, RS-2024-00336576, RS-2023-0022663) and IITP grants (RS-2022-II220264, RS-2024-00353131) by the government of Korea.

GenAI Usage Disclosure

ChatGPT was utilized exclusively for grammatical corrections and enhancing the fluency of the writing. We did not use generative AI tools for any other purposes.

References

[1] Arkadeep Acharya, Brijraj Singh, and Naoyuki Onoe. 2023. Llm based generation of item-description for recommendation system.

[2] Keqin Bao, Jizhi Zhang, Wenjie Wang, Yang Zhang, Zhengyi Yang, Yanchen Luo, Chong Chen, Fuli Feng, and Qi Tian. 2025. A bi-step grounding paradigm for large language models in recommendation systems. *ACM Transactions on Recommender Systems* 3, 4 (2025), 1–27.

[3] Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. 2023. Tallrec: An effective and efficient tuning framework to align large language model with recommendation.

[4] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Workshop on deep learning for recommender systems*.

[5] Minjin Choi, Yoonki Jeong, Joonseok Lee, and Jongwuk Lee. 2021. Local collaborative autoencoders. In *WSDM*.

[6] Evangelia Christakopoulou and George Karypis. 2018. Local latent space models for top-n recommendation. In *KDD*.

[7] Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. 2022. Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5).

[8] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*.

[9] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv:1511.06939* (2015).

[10] Yupeng Hou, Zhankui He, Julian McAuley, and Wayne Xin Zhao. 2023. Learning vector-quantized item representation for transferable sequential recommenders. In *WWW*.

[11] Yupeng Hou, Shanlei Mu, Wayne Xin Zhao, Yaliang Li, Bolin Ding, and Ji-Rong Wen. 2022. Towards universal sequence representation learning for recommender systems. In *KDD*. 585–593.

[12] Yupeng Hou, Junjie Zhang, Zihan Lin, Hongyu Lu, Ruobing Xie, Julian McAuley, and Wayne Xin Zhao. 2024. Large language models are zero-shot rankers for recommender systems. In *European Conference on Information Retrieval*.

[13] Wenyue Hua, Shuyuan Xu, Yingqiang Ge, and Yongfeng Zhang. 2023. How to index item ids for recommendation foundation models. In *Proc. of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*.

[14] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *IEEE international conference on data mining (ICDM)*.

[15] Chanwoo Kim, Jinkyu Sung, Yebonn Han, and Joonseok Lee. 2025. Graph Spectral Filtering with Chebyshev Interpolation for Recommendation. In *SIGIR*.

[16] Eungi Kim, Chanwoo Kim, Kwangeun Yeo, Jinri Kim, Yujin Jeon, Sewon Lee, and Joonseok Lee. 2025. ReducedGCN: learning to adapt graph convolution for top-N recommendation. In *PAKDD*.

[17] Sein Kim, Hongseok Kang, Seungyoon Choi, Donghyun Kim, Minchul Yang, and Chanyoung Park. 2024. Large Language Models meet Collaborative Filtering: An Efficient All-round LLM-based Recommender System. In *KDD*.

[18] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[19] Joonseok Lee, Samy Bengio, Seungyeon Kim, Guy Lebanon, and Yoram Singer. 2014. Local collaborative ranking. In *WWW*.

[20] Joonseok Lee, Seungyeon Kim, Guy Lebanon, and Yoram Singer. 2013. Local low-rank matrix approximation. In *ICML*.

[21] Joonseok Lee, Seungyeon Kim, Guy Lebanon, and Yoram Singer. 2013. Matrix approximation under local low-rank assumption.

[22] Joonseok Lee, Seungyeon Kim, Guy Lebanon, Yoram Singer, and Samy Bengio. 2016. LLORMA: Local low-rank matrix approximation. *Journal of Machine Learning Research* 17, 15 (2016), 1–24.

[23] Joonseok Lee, Mingxuan Sun, Seungyeon Kim, and Guy Lebanon. 2012. Automatic feature induction for stagewise collaborative filtering. In *NIPS*.

[24] Lei Li, Yongfeng Zhang, and Li Chen. 2023. Prompt distillation for efficient llm-based recommendation. In *CIKM*.

[25] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *WWW*.

[26] Jiayi Liao, Sihang Li, Zhengyi Yang, Jiancan Wu, Yancheng Yuan, and Xiang Wang. 2023. Llara: Aligning large language models with sequential recommenders. *CoRR* (2023).

[27] Junling Liu, Chao Liu, Peilin Zhou, Renjie Lv, Kang Zhou, and Yan Zhang. 2023. Is chatgpt a good recommender? a preliminary study. *arXiv:2304.10149* (2023).

[28] Junling Liu, Chao Liu, Peilin Zhou, Qichen Ye, Dading Chong, Kang Zhou, Yueqi Xie, Yuwei Cao, Shoujin Wang, Chenyu You, et al. 2023. Llmrec: Benchmarking large language models on recommendation task. *arXiv:2308.12241* (2023).

[29] Zhongzhou Liu, Hao Zhang, Kuicai Dong, and Yuan Fang. 2024. Collaborative Cross-modal Fusion with Large Language Model for Recommendation. In *CIKM*.

[30] Sichun Luo, Yuxuan Yao, Bowei He, Yinya Huang, Aojun Zhou, Xinyi Zhang, Yuanzhang Xiao, Mingjie Zhan, and Linqi Song. 2024. Integrating large language models into recommendation via mutual augmentation and adaptive aggregation. *arXiv:2401.13870* (2024).

[31] Yucong Luo, Qitao Qin, Hao Zhang, Mingyue Cheng, Ruiran Yan, Kefan Wang, and Jie Ouyang. 2024. Molar: Multimodal LLMs with Collaborative Filtering Alignment for Enhanced Sequential Recommendation. *arXiv:2412.18176* (2024).

[32] Hanjia Lyu, Song Jiang, Hanqing Zeng, Yinglong Xia, Qifan Wang, Si Zhang, Ren Chen, Christopher Leung, Jiajie Tang, and Jiebo Luo. 2023. LLM-rec: Personalized recommendation via prompting large language models. *arXiv:2307.15780* (2023).

[33] Chen Ma, Peng Kang, and Xue Liu. 2019. Hierarchical gating networks for sequential recommendation. In *KDD*.

[34] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *SIGIR*.

[35] Xia Ning and George Karypis. 2011. Slim: Sparse linear methods for top-n recommender systems. In *ICDM*.

[36] Aleksandr V Petrov and Craig Macdonald. 2023. Generative sequential recommendation with gptrec. *arXiv:2306.11114* (2023).

[37] Haohao Qu, Wenqi Fan, Zihuai Zhao, and Qing Li. 2024. Tokenrec: learning to tokenize id for llm-based generative recommendation. *arXiv:2406.10450* (2024).

[38] Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Tran, Jonah Samost, et al. 2023. Recommender systems with generative retrieval. In *NeurIPS*.

[39] Xubin Ren, Wei Wei, Lianghao Xia, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. 2024. Representation learning with large language models for recommendation. In *WWW*.

[40] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian personalized ranking from implicit feedback. *arXiv:1205.2618* (2012).

[41] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*. 285–295.

[42] Anima Singh, Trung Vu, Nikhil Mehta, Raghunandan Keshavan, Maheswaran Sathiamoorthy, Yilin Zheng, Lichan Hong, Lukasz Heldt, Li Wei, Devansh Tandon, et al. 2024. Better generalization with semantic ids: A case study in ranking for recommendations.

[43] 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 *CIKM*.

[44] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *WSDM*.

[45] Lei Wang and Ee-Peng Lim. 2023. Zero-shot next-item recommendation using large pretrained language models. *arXiv:2304.03153* (2023).

[46] Wei Wei, Xubin Ren, Jiabin Tang, Qinyong Wang, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. 2024. LLMRec: Large language models with graph augmentation for recommendation. In *WSDM*.

[47] Likang Wu, Zhi Zheng, Zhaopeng Qiu, Hao Wang, Hongchao Gu, Tingjia Shen, Chuan Qin, Chen Zhu, Hengshu Zhu, Qi Liu, et al. 2024. A survey on large language models for recommendation. *World Wide Web* 27, 5 (2024), 60.

[48] Da Yu, Edith Cohen, Badih Ghazi, Yangsibo Huang, Pritish Kamath, Ravi Kumar, Daogao Liu, and Chiyuan Zhang. 2025. Scaling Embedding Layers in Language Models. *arXiv:2502.01637* (2025).

[49] Tingting Zhang, Pengpeng Zhao, Yanchi Liu, Victor S Sheng, Jiajie Xu, Deqing Wang, Guanfeng Liu, Xiaofang Zhou, et al. 2019. Feature-level deeper self-attention network for sequential recommendation.. In *IJCAI*.

[50] Yang Zhang, Fuli Feng, Jizhi Zhang, Keqin Bao, Qifan Wang, and Xiangnan He. 2023. Collm: Integrating collaborative embeddings into large language models for recommendation. *arXiv:2310.19488* (2023).

[51] Zizhuo Zhang and Bang Wang. 2023. Prompt learning for news recommendation. In *SIGIR*.

[52] Bowen Zheng, Yupeng Hou, Hongyu Lu, Yu Chen, Wayne Xin Zhao, Ming Chen, and Ji-Rong Wen. 2024. Adapting large language models by integrating collaborative semantics for recommendation. In *ICDE*.

[53] Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. 2020. S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *CIKM*.

[54] Yaochen Zhu, Liang Wu, Qi Guo, Liangjie Hong, and Jundong Li. 2024. Collaborative large language model for recommender systems. In *WWW*.