RAIE: Region-Aware Incremental Preference Editing with LoRA for LLM-based Recommendation

Jin Zeng*
zengj255@mail2.sysu.edu.cn
School of Cyber Science and Technology, Shenzhen Campus of Sun Yat-sen University
Shenzhen, Guangdong, China

Yupeng Qi*
qiyp7@mail2.sysu.edu.cn
School of Cyber Science and Technology, Shenzhen Campus of Sun Yat-sen University
Shenzhen, Guangdong, China

Hui Li
hui@xmu.edu.cn
Department of Computer Science and Technology, Xiamen University
Xiamen, Fujian, China

Chengming Li
licm@smbu.edu.cn
Artificial Intelligence Research Institute, Shenzhen MSU-BIT University
Shenzhen, Guangdong, China

Ziyu Lyu†
lvzy7@mail.sysu.edu.cn
School of Cyber Science and Technology, Shenzhen Campus of Sun Yat-sen University
Shenzhen, Guangdong, China

Lixin Cui
cuilixin@cufe.edu.cn
School of Information, Central University of Finance and Economics
Beijing, China

Lu Bai
bailu@bnu.edu.cn
School of Artificial Intelligence, Beijing Normal University
Beijing, China

Abstract

Large language models (LLMs) are increasingly adopted as the backbone of recommender systems. However, user–item interactions in real-world scenarios are non-stationary, making preference drift over time inevitable. Existing model update strategies mainly rely on global fine-tuning or pointwise editing, but they face two fundamental challenges: (i) imbalanced update granularity, where global updates perturb behaviors unrelated to the target while pointwise edits fail to capture broader preference shifts; (ii) unstable incremental updates, where repeated edits interfere with prior adaptations, leading to catastrophic forgetting and inconsistent recommendations. To address these issues, we propose **R**egion-**A**ware **I**ncremental **E**diting (RAIE), a plug-in framework that freezes the backbone model and performs region-level updates. RAIE first constructs semantically coherent preference regions via spherical k-means in the representation space. It then assigns incoming sequences to regions via confidence-aware gating and performs three localized edit operations—Update, Expand, and Add—to dynamically revise the affected region. Each region is equipped with a dedicated Low-Rank Adaptation (LoRA) module, which is trained only on the region's updated data. During inference, RAIE routes each user sequence to its corresponding region and activates the region-specific adapter for prediction. Experiments on two benchmark datasets under a time-sliced protocol that segments data into Set-up (S), Finetune (F), and Test (T) show that RAIE significantly outperforms state-of-the-art baselines while effectively mitigating forgetting. These results demonstrate that region-aware editing offers an accurate and scalable mechanism for continual adaptation in dynamic recommendation scenarios. Our code is available at https://github.com/fengaogao/RAIE.

CCS Concepts

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

Keywords

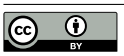
LLM-based Recommender Systems, Knowledge Editing, Preference Drift

ACM Reference Format:
Jin Zeng, Yupeng Qi, Hui Li, Chengming Li, Ziyu Lyu, Lixin Cui, and Lu Bai. 2026. RAIE: Region-Aware Incremental Preference Editing with LoRA for LLM-based Recommendation. In *Proceedings of the ACM Web Conference 2026 (WWW '26), April 13–17, 2026, Dubai, United Arab Emirates.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3774904.3792451

1 INTRODUCTION

Recommender systems are widely deployed in online services such as e-commerce, video platforms and advertising to help users discover relevant content and reduce information overload [39, 41]. Among existing techniques, sequential recommendation models user behaviors as an ordered interaction sequence and predicts the next item by capturing temporal dependencies. Representative methods include SASRec [8], applying self-attention to model sequential patterns; TiSASRec [17], incorporating time-interval

*Both authors contributed equally to this research.
†Corresponding author.



WWW '26, Dubai, United Arab Emirates

ACM ISBN 979-8-4007-2307-0/2026/04
https://doi.org/10.1145/3774904.3792451

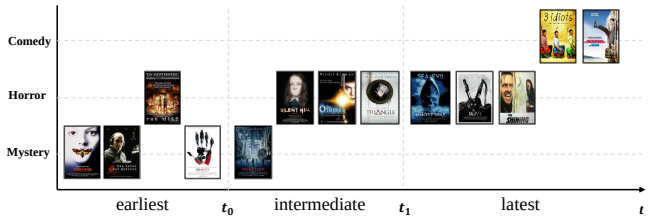


Figure 1: Illustrated example of user preference drift in the movie domain.

signals; and BERT4Rec [35], learning bidirectional dependencies via masked-item prediction. Despite their effectiveness, most sequential recommenders still rely on ID-based item representations, which encode limited semantic information and generalize poorly to unseen items or new domains.

Large language models (LLMs) alleviate the semantic limitations of ID-based recommenders [4, 22] by leveraging item content to learn semantically grounded representations. Existing LLM-based recommenders can be broadly organized into three paradigms: (i) feature-based augmentation [30], where LLMs enrich user and item representations with semantic signals, (ii) generative recommendation [1, 15, 24], where LLMs are used to directly generate recommendation content via prompt, (iii) agentic recommenders [7], where LLMs act as agents that iteratively use tools and memory for recommendation. However, most prior work is trained and evaluated under static or near-stationary distributions. In practice, user interactions arrive continuously and data distributions evolve over time, leading to preference drift. When an LLM-based recommender is trained on historical data, it effectively learns a time-invariant mapping from context to relevance; as user interests shift, this mapping becomes stale, resulting in degraded recommendation performance. Therefore, dynamic recommendation requires drift-aware mechanisms that can stably adapt to evolving preferences.

Figure 1 illustrates a preference-drift example. For clarity, we partition a user's interactions into three consecutive time slices: earliest, intermediate and latest. In the earliest slice, the user mainly prefers *mystery*; in the intermediate slice, the preference shifts toward *horror*, which remains dominant in the latest slice, while the latest slice also shows a short-term increase in *comedy*. If a static LLM trained only on the earliest slice is used to make recommendations for the latest slice, it will often overestimate the user's interest in *mystery*, failing to capture both the dominant preference and the transient shifts observed in the latest slice. A straightforward solution is periodic retraining, but it is computationally expensive and can overwrite valid long-term preferences under distribution shifts, leading to catastrophic forgetting [10, 29]. Parameter-efficient fine-tuning (PEFT) offers a more efficient alternative by updating a small number of parameters while freezing the backbone. For example, LoRA updates low-rank matrices with low training and storage costs [6], and adapter-based methods also support efficient transfer and fast adaptation [5]. In practice, many PEFT pipelines rely on a single global adapter to handle distributional shifts. Although effective, such global updates may perturb stable preferences and introduce interference across behavioral patterns, resulting in a process that lacks specificity and precise control. Recent multi-adapter variants (e.g. dual-LoRA [32] or mix of experts LoRA [37]) enhance adaptation capacity by assigning different adapters to different tasks. However, such methods often route data to adapters using task-level heuristics, and their parameter updates are weakly aligned with user preference changes, making it difficult to precisely adapt to the specific drifting regions.

Inspired by knowledge editing for LLMs [25, 26, 28], we propose RAIE, a plug-in, region-aware incremental editing framework for LLM-based recommendation to address preference drift. A key observation is that preference drift is often localized: a user may remain stable on some interests while drifting within others [36]. This calls for a structured update scheme that groups interactions into interest groups and localizes adaptation to the interest group experiencing drift, thereby preserving stable preferences elsewhere. We therefore introduce knowledge regions, defined as semantically coherent clusters of a user's historical interactions in a shared representation space, each summarized by a centroid and an effective radius. By organizing interactions into regions, RAIE can localize both routing and updates: when drift occurs, it updates only the affected region, reducing interference with stable regions. RAIE consists of three components: (a) **knowledge region construction**, which partitions the user's historical interactions into multiple regions and assigns a LoRA adapter to each region; (b) **region-aware editing and LoRA adaptation**, which uses confidence-aware routing to trigger edit operations and trains the corresponding regional adapter on updated region data; and (c) **region-aware routing**, which activates the region-specific adapter at inference to reduce interference across regions. Focusing on component (b), RAIE supports three regional edit operations to track preference evolution: Update refines the region's representation under fixed boundaries; Expand relaxes the boundary moderately and applies the same refinement to cover mild outward shifts; Add instantiates a new region when a novel pattern emerges. Refinement uses exponential-moving-average (EMA) updates for stability. A similarity-based router assigns new sequences to candidate regions, and a confidence threshold determines whether an edit is triggered. Each knowledge region is paired with a LoRA adapter initialized uniformly. After editing, the regional data are used to train its adapter, aligning the adapter's parameters with the region's current sub-distribution.

In summary, our contributions are highlighted as follows:

- We formalize user preference drift adaptation in LLM-based recommendation as a region-aware incremental editing problem, highlighting the core trade-off between localized adaptation and global stability. This formulation provides a structured approach to manage evolving user preferences while maintaining model coherence.
- We introduce the Region-Aware Incremental Editing (RAIE) framework, which integrates three key components: Knowledge Region Construction, Region-Aware Preference Editing and LoRA Adaptation, while reducing cross-region interference via region-specific adapters and routing.
- Extensive experiments on MovieLens-10M and Yelp demonstrate that RAIE consistently outperforms state-of-the-art baselines, while effectively adapting to emerging user preferences and mitigating forgetting in cross-phase incremental learning scenarios.

2 RELATED WORK

2.1 Continual learning

User preferences are inherently non-stationary, evolving over time due to changing contexts. This preference drift causes models trained on static historical data to become misaligned with current user interests [13]. While periodic full retraining of conventional sequential recommenders (e.g., SASRec [8], BERT4Rec [35]) is common, it becomes prohibitively costly for LLM-based backbones due to massive computational and storage overhead. Moreover, retraining from scratch can exacerbate catastrophic forgetting, disrupting previously learned stable preferences. Continual learning offers a framework to balance stability (retaining old knowledge) and plasticity (acquiring new knowledge) [40]. Representative strategies include: (i) regularization-based methods (e.g., EWC [10], ℓ_2-SP[18]) that constrain parameter updates to protect important weights, and (ii) replay-based methods [11, 33] that revisit past data to mitigate forgetting. Despite these advances, applying continual learning to LLM-based recommenders remains challenging in practice. Regularization-based methods still require repeated gradient updates on a large backbone, while replay-based methods incur non-trivial storage and computation overhead and can amplify training instability as updates accumulate.

2.2 Parameter-Efficient Fine-Tuning for LLM-based Recommendation systems

To enable feasible adaptation of large models, Parameter-Efficient Fine-Tuning (PEFT) has emerged as a key paradigm [3]. Techniques like low-rank adaptation (LoRA) [6], adapters [5] and prefix tuning [19] freeze the pre-trained backbone and only optimize a small set of injected parameters, drastically reducing update costs. To enhance capacity and specialization, recent work introduces routing mechanisms. For instance, instance-level LoRA (iLoRA [12]) and Mixture of LoRA Experts (MoLE [37]) dynamically select or combine multiple adapters to tailor the model's behavior. These PEFT and routing techniques are increasingly integrated into modern LLM-for-Rec pipelines [30, 38]. However, adapter selection is often driven by task-level heuristics, and the resulting updates are not explicitly tied to how user interests evolve over time, which makes it hard to apply targeted adaptations without introducing unnecessary interference.

2.3 Knowledge Editing for LLMs and Recommenders

Knowledge editing aims to make precise, localized changes to a model's behavior without full retraining [42]. Parametric editing methods (e.g., ROME [25] and MEMIT [26]) directly modify specific model weights to alter factual associations, while external memory-based methods (SERAC [28], MEND [27]) store edits in a side network to override the base model's predictions. In recommendation, editing concepts have been applied to correct user-item interactions [14] or update knowledge graphs [31]. Lifelong editing frameworks like ELDER [16] explore LoRA-based routing to manage sequential edits. Nevertheless, most work remains focused on editing factual knowledge or discrete interaction records, rather than modeling and editing continuous preference subspaces. We advocate a preference-oriented editing perspective, whose central goal is to calibrate the user's dynamically evolving interests. Our proposed RAIE framework instantiates this perspective: it organizes user behaviors into semantically coherent regions, treats each preference region as an editable unit, assigns it a dedicated LoRA module, and strictly localizes updates to regions experiencing drift, thereby bridging the gap between knowledge editing and preference adaptation in recommendation.

3 Problem Definition

Interaction Sequences and Temporal Split. In the recommendation scenario, we have a user set $\mathcal{U}$ and an item set $\mathcal{V}$. For a user $u_i \in \mathcal{U}$, their time-ordered interaction history is denoted as $S_{u_i} = [v_{i,1}, v_{i,2}, \ldots, v_{i,L_{u_i}}]$, where each item $v_{i,j}$ is associated with a timestamp $t_{i,j}$. In order to perform incremental learning and updates, we split each user's sequence into three temporal phases: S^S (Set-up phase), S^F (Incremental Fine-tune phase), and S^T (Inference Test phase). Specifically, S^S contains interactions with $t \le t^S$, S^F contains those with $t^S < t \le t^F$, and S^T contains the remaining interactions with $t > t^F$.

Task Definition. The incremental recommendation process consists of three stages: set-up phase, dynamic fine-tuning phase on incoming sequences, and final inference test phase. As time progresses, new data splits are continuously generated. The system works dynamically by performing incremental fine-tuning on each new training splits, followed by inference testing on the corresponding test splits. For simplicity, we state the problem formulation with the three phases defined above, i.e., S^S, S^F, S^T.

Set-up Phase (S). We construct the initial model using the early-stage user sequences S^S. We adopt a base recommendation model $\mathcal{M}(\theta)$, which can be traditional sequential recommendation methods or LLM-based methods[1]. This model is used to obtain initial user sequence representations from S^S. For LLM-based recommenders, we define a prompt construction function BuildPrompt$(\cdot)$ that maps a user subsequence to a textual prompt[2]. In addition, we construct a set of initial preference regions $\Phi = (c_k, R_k)_{k=1}^K$, where K is the initial number of regions, c_k is the center of region k, and R_k defines its effective radius in the representation space.

Incremental Learning Finetune Phase (F). This is the core phase of incremental recommendation. Its primary goal is to discover the fine-grained preference changes within the existing preference regions Φ and to perform incremental preference updating to the recommendation model $\mathcal{M}$ (i.e., $\mathcal{M}(\theta) \rightarrow \mathcal{M}(\theta')$) based on the new user sequences S^F.

Inference Test Phase (T). The phase is to perform user preference prediction. Based on the updated model $\mathcal{M}(\theta')$, the new preference regions Φ, and the test user sequence S^T, the model performs next-item prediction for each test sequence to evaluate the effectiveness of the incremental adaptation.

[1] In the method section, we mainly introduce that the sequence representation extraction from LLM-based recommendation methods. But our proposed framework is plug-in method that is flexible for traditional recommendation methods. We present experimental results with diverse base models to demonstrate this generality.

[2] Prompt template: Here is the purchase history of user_{user_id}: item {history}. I wonder what is the next recommended item for the user. Answer:

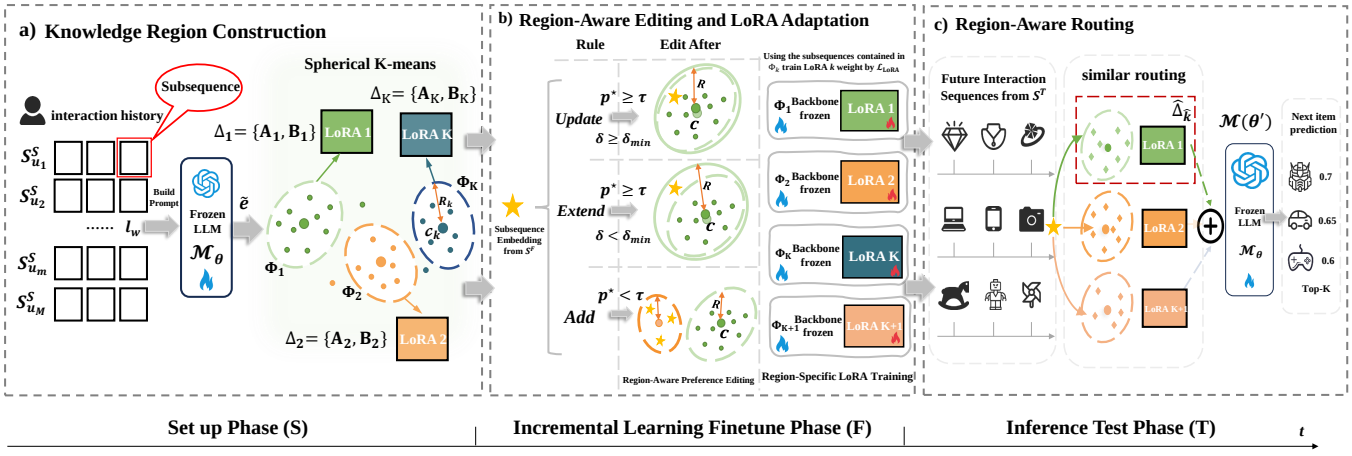


Figure 2: The overall architecture of our proposed RAIE. It consists of three core modules, respectively. a) Knowledge Region Construction at the set-up phase (S); b) Region-aware Editing and LoRA Adaption in the incremental learning finetune phase (F); and the Region-ware Routing for the inference test phase (T).

4 Method

To address the challenges of incremental preference updating, we propose Region-Aware Incremental Editing (RAIE), a framework that integrates low-rank adaptation for LLM-based recommendation. The overall architecture of RAIE is shown in Figure 2. Our method consists of three key components: (1) Knowledge Region Construction, (2) Region-Aware Editing and LoRA Adaptation, and (3) Region-Aware Routing. RAIE operates dynamically across three phases: set-up (S), incremental fine-tuning (F), and inference (T). During the set-up phase, initial knowledge regions are constructed, each paired with a LoRA-based parameter-efficient module. The region-aware editing and LoRA adaptation module serves as the core mechanism in the incremental fine-tuning phase. It locates the corresponding region for each new preference sequence and performs incremental editing through three operations: Update, Expand, and Add. Based on the edited regions, we finetune the corresponding LoRA adapters to enable efficient incremental learning. During inference, the region-aware routing mechanism activates the appropriate adapter for next-item prediction. Algorithm 1 summarizes the overall procedure of method.

4.1 Knowledge Region Construction

We perform knowledge region construction using user sequence S^s from the set-up phase. This process consists of three steps: (1) segmenting user sequences into subsequences and extracting their representations, (2) discovering preference regions via clustering, and (3) mapping each region to a dedicated LoRA adapter.

4.1.1 Subsequence Segmentation and representation. Since user sequences are often long and contain multiple intents over time, capturing fine-grained preferences directly from the full sequence is difficult. Therefore, we first segment each user sequence into a series of overlapped subsequences and obtain their representations. Specifically, we adopt a sliding window approach [9] with window length l_w and stride n. For the user sequences S^s, this yields subsequences $[A_1, A_2, \ldots, A_{\lfloor (T_u - L_w)/n \rfloor + 1}]$.[3] This fragment-level view emphasizes temporal locality and isolates short-term intents, highlighting fine-grained user preferences.

After subsequence segmentation, we extract the embedding representation for each subsequence using an LLM-basd backbone. We construct the user preference prompt by filing the user subsequence into a template: $\mathbf{x}_u = \text{BuildPrompt}(A)$ and obtain the embedding representation from the frozen backbone $\mathcal{M}(\mathbf{x}_u)$. We take the last-token hidden state of the last hidden layer as the subsequence preference representation $\mathbf{e} \in \mathbb{R}^d$. For further region clustering in the spherical space, we apply ℓ_2-normalization on the preference represenation $\mathbf{e}$, and finally obtain the normalized representation $\tilde{\mathbf{e}} = \mathbf{e}/\|\mathbf{e}\|_2$.

4.1.2 Preference Region Discovery. We treat all the subsequences from all users in the set-up phase as clustering samples, and apply the spherical k-means method [2] on the preference representation of each sample to perform preference region discovery. The spherical k-means method will cluster the similar preference into a preference region, with the intuition that the samples within a cluster share similar preferences, and the samples from difference clusters have different preferences. The optimization objective is defined as in Equation 1, the initial number of regions is set as k.

$$\max_{\{C_k, c_k\}_{k=1}^{K}} \sum_{k=1}^{K} \sum_{\tilde{\mathbf{e}} \in C_k} c_k^\top \tilde{\mathbf{e}} \quad \text{s.t.} \quad \|c_k\|_2 = 1. \tag{1}$$

where C_k denote the set of samples for each preference region, c_k is center prototype of each cluster region.

When the clustering algorithm finishes, we obtain K preference regions $\Phi = \{(c_k, R_k)\}_{k=1}^{K}$. Each region is characterized by a prototype representation c_k and a radius R_k. The radius quantifies the intra-region cohesion, parameterizing coverage and locality.

[3] The same segmentation is applied to sequences in the finetuning and inference phases for consistent processing.

Algorithm 1: Region-Aware Incremental Editing

Input : User set $\mathcal{U}$; frozen encoder $\mathcal{M}_\theta$; data splits S^S, S^F, S^T
Output: top-k item recommendation list

for *each* $S_u^S \in S^S$ **do**
  Segment user sequence S_u^S to obtain subsequences A
  Build textual prompt $\mathbf{x}_u$ and encode it using $\mathcal{M}(\theta)$ to obtain $\tilde{\mathbf{e}}$
end
// 1. Knowledge Region Discovery
Apply spherical k-means clustering to obtain centers c and radius R
Form knowledge region set $\Phi = \{(c_k, R_k)\}_{k=1}^K$
// 2. Region-Aware Preference Editing
for *each incoming subsequence* $S_u^F \in S^F$ **do**
  perform subsequence segment and obtain the vector $\tilde{\mathbf{e}}$
  Select candidate region $\hat{\Phi}_k$ via cosine similarity Eq. (3)
  if $p^\star \geq \tau$ *and* $\delta \geq \delta_{\min}$ **then**
    Update region center and radius via Eqs. (5)
  else if $p^\star \geq \tau$ *and* $\delta < \delta_{\min}$ **then**
    Expand boundary using Eqs. (6)
  else
    Create new region by Eqs.(1)
  end
end
// 3. Region-Specific LoRA Training
for *each region* $k = 1, \ldots, K$ **do**
  Insert low-rank adapters $\Delta\theta_k$
  Train $\Delta\theta_k$ on $S_k = S_k^{(S)} \cup S_k^{(F)}$ using next-item loss Eq. (7)
end
Activate trained LoRA adapters and score candidates via Eq.(10)
return *top-k item recommendation list*

4.1.3 *Regional LoRA Mapping.* After obtaining the preference regions, we aim to conduct region-aware incremental learning for the recommender model. We adopt a region-specific LoRA strategy: each preference region is mapped to a dedicated LoRA adapter, which is updated in a parameter-efficient manner to capture localized preference knowledge during the F phase. Here we demonstrate the regional LoRA mapping mechanism.

In general, LoRA-based parameter-efficient learning for a recommender model is defined as:

$$\max_{\Delta\theta} \sum_{(\mathbf{x}_u, v) \in S} \sum_{t=1}^{L} \log \mathcal{M}_{\theta+\Delta\theta}(v_t | \mathbf{x}_u, v_{<t}) \tag{2}$$

where $\Delta\theta$ is decomposed into two low-rank matrices $\mathbf{A} \in \mathbb{R}^{r \times d_{\text{in}}}$, $\mathbf{B} \in \mathbb{R}^{d_{\text{out}} \times r}$. When performing finetuning or incremental learning on the new data, we only update the low-rank matrices, instead of the full set of model parameters, ensuring parameter efficiency.

In order to perform the fine-grained and localized preference updating, we construct the region-specific LoRA, and initially map each preference regions with a region specific LoRA. Namely, for each preference region Φ_k is associated with region specific LoRA matrices $\mathbf{A}^k$ and $\mathbf{B}^k$. In the incremental learning phase, as the number of preference regions increases, a new region-specific LoRA is constructed. Incremental learning then proceeds in a parameter-efficient manner, whereby each region is updated via its corresponding LoRA module as defined in Equation 2.

4.2 Region-Aware Preference Editing and LoRA Adaptation

After the set-up phase, the core part of our method is to perform region-aware preference editing and incremental learning via LoRA adaption, using the new user sequence S^F.

4.2.1 Region-Aware Preference Editing.

Region-aware Preference Region Localization. Before preference editing, the preliminary process is to determine which regions to edit. For a new user preference subsequence A^F in the F phase, we have the preference representation $\tilde{\mathbf{e}}$ (same operation as in Section 4.1). We then calculate the similarities between the preference represenation and each region's center representation:

$$Score_k = c_k^\top \tilde{\mathbf{e}}, \quad p_k = \frac{\exp(Score_k)}{\sum_{j=1}^K \exp(Score_j)} \tag{3}$$

where $Score_k$ denotes the similarity score between the new preference representation with a region center, p_k indicates the confidence score. We can sort the regions based on the confidence scores, and choose the preference regions with the highest confidence scores as the candidate matching regions.

Preference Region Editing Mechanism. When getting the candidate matching regions, we define some rules to select the editing operation for each preference region as in Equation 4. We denote $p^\star = p_{\hat{k}}$ be the highest confidence, and $\delta = p^\star - \max_{j \neq \hat{k}} p_j$ be the upper confidence margin between the highest confidence score and the second highest confidence score. If the top confidence is smaller than a threshold τ, it indicates there is no similar region and requires to add a new region with the **Add** operation. But if the top confidence is larger than a threshold τ, it indicates there are some similar regions, perform the **Update** operation or the **Expand** operation on the matching regions. If upper confidence margin is large ($\delta_{\min} > 0$ is the minimum margin), we should **Update** the matching region, otherwise perform the **Expand** operation.

$$\text{Act}(\tilde{\mathbf{e}}) = \begin{cases} \texttt{update}, & \text{if } p^\star \geq \tau \text{ and } \delta \geq \delta_{\min}, \\ \texttt{expand}, & \text{if } p^\star \geq \tau \text{ and } \delta < \delta_{\min}, \\ \texttt{add}, & \text{if} p^\star < \tau \end{cases} \tag{4}$$

The three editing operations are defined as follows:

- **Update:** updating the center and radius as follows:

$$R_{\hat{k}} \leftarrow (1-\beta) R_{\hat{k}} + \beta \arccos(\mathbf{c}_{\hat{k}}^\top \tilde{\mathbf{e}})$$

$$\tilde{\mathbf{c}}_{\hat{k}} = (1-\gamma)\mathbf{c}_{\hat{k}} + \gamma \tilde{\mathbf{e}}, \quad \mathbf{c}_{\hat{k}} \leftarrow \frac{\tilde{\mathbf{c}}_{\hat{k}}}{\|\tilde{\mathbf{c}}_{\hat{k}}\|_2}. \tag{5}$$

  where β and γ are smoothing coefficients.
- **Expand:** Expand the boundary of the preference regions while keeping semantically aligned:

$$R_{\hat{k}} \leftarrow R_{\hat{k}} + \lambda \big(\arccos(\mathbf{c}_{\hat{k}}^\top \tilde{\mathbf{e}}) - R_{\hat{k}} \big)_+, \quad R_{\hat{k}} \leq R_{\max}$$

$$\mathbf{c}_{\hat{k}} \leftarrow \frac{(1-\alpha)\mathbf{c}_{\hat{k}} + \alpha\ \tilde{\mathbf{e}}}{\|(1-\alpha)\mathbf{c}_{\hat{k}} + \alpha\ \tilde{\mathbf{e}}\|_2}. \tag{6}$$

  where $\lambda > 0$ controls the expansion rate, $\alpha \in (0, 1)$ and $R_{\max}$ caps the maximum radius.

- **Add** : When the exsiting regions are not similar with the new user preference, we need to create a new region via the add operation. We design the bath-level new region construction. Namely, we construct a buffer pool $\mathcal{B}$ and fill the new user preference sample into the buffer pool when it triggers the **Add** operation. When $|\mathcal{B}_{\text{add}}|$ exceeds a threshold, we perform spherical k-means on $\mathcal{B}$ as defined in Equation 1.

4.2.2 Region-Specific LoRA Training. After performing preference editing, we train the corresponding region-specific LoRA adapters. As mentioned above, each preference region Φ_k corresponding to a Region-Specific LoRA with two low-rank matrices $\mathbf{A}^k$ and $\mathbf{B}^k$. We perform Region-Specific LoRA Training on $S_k = S_k^{(S)} \cup S_k^{(F)}$ by optimizing the following loss function $\mathcal{L} = \mathcal{L}_{\text{LoRA}} + \mathcal{L}_p$:

$$\mathcal{L}_{\text{LoRa}}(\Delta\theta_k; S_k) = -\sum_{(\mathbf{x}_u, v)\in S_k} \sum_{t=1}^{L} \log \mathcal{M}_{\theta+\Delta\theta_k}(v_t|\mathbf{x}_u, v_{<t}) \quad (7)$$

$$\mathrm{d}_{i,j} = \left(R_i + R_j - \|\mathbf{c}_i - \mathbf{c}_j\|_2\right)_+, \qquad \mathcal{L}_p = \lambda_{\text{sep}} \sum_{i<j} {\mathrm{d}_{i,j}}^2. \quad (8)$$

where $\mathcal{L}_{\text{LoRA}}$ is the LoRA training loss and $\mathcal{L}_p$ is the a penalty term, which control the potential ovelap between different regions.

4.3 Region-Aware Routing.

In the inference phase, we have the trained region-specific LoRA adapters $\{\widehat{\Delta}_k\}_{k=1}^K$. Given a new user subsequence, we obtain its preference representation $\tilde{\mathbf{e}}$ and select the most compatible preference region $\Phi_{\hat{k}}$ based on the similarity:

$$\Phi_{\hat{k}} = \underset{k\in\{1,\ldots,k\}}{\arg\max}\ \tilde{\mathbf{e}}^\top c_k. \quad (9)$$

We then attach the corresponding adapter $\widehat{\Delta}_{\hat{k}}$ to the frozen backbone $\mathcal{M}_\Theta$. Let $f(v \mid \mathbf{x}_u, \mathcal{M}_\Theta, \widehat{\Delta}_{\hat{k}})$ denote the routed model's logit for item v. The predictive distribution over the candidate set $\mathcal{V}$ is

$$p(v \mid \mathbf{x}_u, \mathcal{M}, \widehat{\Delta}_{\hat{k}}) = \frac{\exp\left(f(v \mid \mathbf{x}_u, \mathcal{M}, \widehat{\Delta}_{\hat{k}})\right)}{\sum_{j\in\mathcal{V}} \exp\left(f(j \mid \mathbf{x}_u, \mathcal{M}, \widehat{\Delta}_{\hat{k}})\right)}. \quad (10)$$

The top-k items under this distribution are returned as the final recommendation list.

5 EXPERIMENTS

To evaluate the performance of RAIE (a plugin-in module for existing recommenders), we conduct experiments to address the following research questions:

• **RQ1**: How does RAIE compare with state-of-the-art plug-in baselines across different backbone models and datasets?
• **RQ2**: What is the contribution of each key component to overall performance?
• **RQ3**: How sensitive is RAIE to major hyperparameters, and are the trends consistent across datasets?
• **RQ4**: Does RAIE provide intuitive interpretability reflecting pre-edit and post-edit changes in preference structure?

Table 1: Statistics of the datasets under the S→F→T protocol.

Stage	Metric	MovieLens-10M	Yelp
Set-up (S)	#Users	43,021	41,072
	#Items	4,784	28,404
	#Interactions	2,502,840	702,212
Finetune (F)	#Users	18,168	39,631
	#Items	8,305	29,657
	#Interactions	1,501,707	421,328
Test (T)	#Users	14,450	30,274
	#Items	9,264	27,437
	#Interactions	1,001,137	280,885

5.0.1 Datasets. We evaluate on two public datasets: **MovieLens-10M**[4]and **Yelp**[5] (statistics in Table 1). Following common practice for implicit recommendation, we binarize feedback by marking ratings ≥ 4 as positive and discarding the rest [21]. To reduce sparsity and noise, we apply k-core filtering (remove users and items with fewer than k interactions): k=5 for MovieLens-10M and k=10 for Yelp. Then, we sorted the interaction records of each user in chronological order to obtain the item sequences. To enforce a consistent temporal protocol, we first compute two dataset-level timestamp quantiles, $q^S = 0.5$ and $q^F = 0.8$, and map them to cutoffs t^S and t^F. For each user, the chronologically ordered interactions are then split into three disjoint segments: Set-up (S) with t < t^S, Finetune (F) with $t^S \le$ t < t^F, and Test (T) with t ≥ t^F. Within each segment, we form training examples via right-aligned sliding windows: the most recent items serve as context and the immediate next item from the same segment is the prediction target; any window that would cross a segment boundary is discarded to prevent leakage.

5.0.2 Implementation details. We use AdamW [23] with learning rate 5×10^{-4}. Models are trained for 5 epochs in the set-up phase and 3 epochs in the fine-tuning phase. LoRA employs rank $r = 8$, scaling factor $\alpha = 16$, and dropout 0.05. For region construction, spherical k-means uses radius quantile q=0.9; region-specific adapters are trained for 3 epochs with an original-data mixing ratio of 0.7.

5.0.3 Baselines for Comparison. We benchmark RAIE against incremental learning plugins across various backbone models. All methods follow the same blocked incremental evaluation protocol.

Sequence Recommendation Methods: BERT4Rec [35]: bidirectional Transformer with masked-item prediction. **SASRec** [8]: causal self-attention for next-item prediction. **TiSASRec** [17]: self-attention enhanced with relative time intervals.

LLM-Based Recommendation Methods: OpenP5 [38]: unified LLM-for-Rec platform.

Incremental plugins : Replay [34]: rehearsal via episodic memory. **LwF** [20]: knowledge distillation to retain prior knowledge.

LLM-based Parameter Efficient Tuning: LSAT [32]: dual-LoRA (long-term stable + short-term adaptive). **MoLE** [12]: mixture of LoRA experts with learned gating.

Knowledge Editing based Recommendation Methods: E-BPR [14]: using novel editing bayesian personalized ranking loss for recommendation editing.

[4]https://grouplens.org/datasets/movielens/10m/
[5]https://www.kaggle.com/datasets/yelp-dataset/yelp-dataset

Table 2: Results on MovieLens-10M (left) and Yelp (right) under the Set-up (S) and Test (T) splits, reporting Recall@10 (R@10) and NDCG@10 (N@10). The Set-up (S) split evaluates knowledge retention (less forgetting), while the Test (T) split measures predictive adaptability to future interactions. RAIE results are highlighted.

Backbone	Variant	MovieLens-10M				Yelp			
		S. R@10	S. N@10	T. R@10	T. N@10	S. R@10	S. N@10	T. R@10	T. N@10
BERT4Rec	Base	0.1311	0.0670	0.0440	0.0216	0.0210	0.0104	0.0094	0.0047
	+LoRA	0.1103	0.0558	0.0569	0.0286	0.0194	0.0097	0.0086	0.0041
	+LoRA+Replay	0.1248	0.0644	0.0563	0.0282	0.0202	0.0101	0.0088	0.0043
	+LoRA+LwF	0.1116	0.0563	0.0573	0.0288	0.0241	0.0123	0.0145	0.0069
	+LSAT	0.1579	0.0816	0.0561	0.0279	0.0325	**0.0160**	0.0176	0.0086
	+MoLE	0.1618	0.0837	0.0467	0.0229	**0.0327**	0.0159	0.0170	0.0083
	+E-BPR	0.1310	0.0669	0.0440	0.0215	0.0255	0.0126	0.0126	0.0060
	+RAIE	**0.1795**	**0.0957**	**0.0870**	**0.0453**	0.0257	0.0126	**0.0195**	**0.0095**
SASRec	Base	0.0703	0.0335	0.0234	0.0108	0.0230	0.0112	0.0099	0.0049
	+LoRA	0.0509	0.0236	0.0366	0.0172	0.0208	0.0104	0.0119	0.0058
	+LoRA+Replay	0.0624	0.0296	0.0322	0.0150	0.0214	0.0104	0.0119	0.0059
	+LoRA+LwF	0.0524	0.0246	0.0374	0.0176	0.0210	0.0103	0.0111	0.0056
	+LSAT	0.0625	0.0295	0.0348	0.0162	0.0217	0.0106	0.0122	0.0057
	+MoLE	0.0703	0.0335	0.0234	0.0107	0.0230	0.0112	0.0099	0.0049
	+E-BPR	**0.0984**	**0.0460**	0.0344	0.0166	**0.0268**	**0.0127**	0.0083	0.0040
	+RAIE	0.0686	0.0327	**0.0449**	**0.0209**	0.0251	0.0123	**0.0125**	**0.0062**
TiSASRec	Base	0.1093	0.0589	0.0342	0.0167	0.0693	0.0321	0.0123	0.0055
	+LoRA	0.1330	0.0686	0.0449	0.0220	0.0580	0.0278	0.0130	0.0061
	+LoRA+Replay	0.1226	0.0646	0.0467	0.0225	0.0572	0.0274	0.0130	0.0062
	+LoRA+LwF	0.1303	0.0675	0.0467	0.0225	0.0561	0.0271	0.0135	0.0063
	+LSAT	0.1118	0.0583	0.0359	0.0176	0.0721	0.0333	0.0136	0.0059
	+MoLE	0.1338	0.0695	0.0467	0.0227	0.0721	0.0333	0.0136	0.0059
	+E-BPR	**0.1666**	**0.0870**	0.0443	0.0218	0.0661	0.0317	0.0127	0.0057
	+RAIE	0.1646	0.0868	**0.0483**	**0.0231**	**0.0833**	**0.0381**	**0.0135**	**0.0063**
openP5	Base	0.2390	0.1422	0.0356	0.0196	0.0123	0.0062	0.0051	0.0024
	+LoRA	0.1109	0.0570	0.0877	0.0470	0.0071	0.0033	0.0068	0.0033
	+LoRA+Replay	0.2675	0.1560	0.0887	0.0474	0.0105	0.0051	0.0079	0.0038
	+LoRA+LwF	0.2456	0.1488	0.0890	0.0477	0.0101	0.0048	0.0083	0.0040
	+LSAT	0.1107	0.0578	0.0572	0.0293	0.0003	0.0001	0.0002	0.0001
	+MoLE	0.0384	0.0183	0.0579	0.0276	0.0043	0.0021	0.0043	0.0019
	+E-BPR	0.2556	0.1503	0.0867	0.0463	0.0111	0.0053	0.0076	0.0037
	+RAIE	**0.2768**	**0.1686**	**0.0935**	**0.0486**	**0.0125**	**0.0062**	**0.0085**	**0.0043**

5.1 Overall Performance (RQ1)

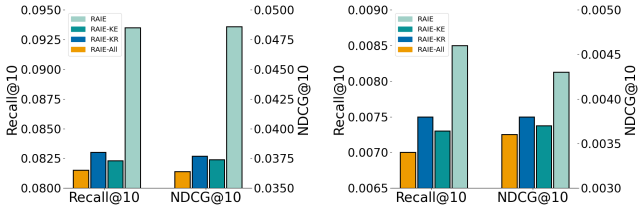
Table 2 reports Recall@10/NDCG@10 on the Set-up (S) split and the Test (T) split. We first observe that adding LoRA to the frozen backbone consistently improves T over the Base model, indicating that parameter-efficient fine-tuning enhances adaptability to preference drift; however, this global adaptation also degrades S, revealing partial forgetting of historical preferences. LoRA+Replay and LoRA+LwF mitigate this forgetting, recovering S while maintaining solid T performance, which suggests that rehearsal or consistency regularization helps retain prior knowledge, though gains on future data remain limited. LSAT and MoLE generally achieve stronger S (retention) than plain LoRA, but their improvements on T are modest and can be backbone-dependent or dataset-dependent, indicating limited generalization. E-BPR, which edits by retraining on mispredicted user–item behavior pairs, excels on S due to strong memorization of corrected interactions but generally lags on T because it lacks an explicit mechanism for modeling evolving intents and thus generalizes poorly under preference shift. Across all methods, scores are lower on Yelp than on MovieLens-10M owing to higher sparsity and noisier interactions, which weaken sequential signals and compress performance margins.

RAIE consistently achieves the best performance. It delivers the strongest T results across all backbones and datasets while maintaining competitive S performance, demonstrating a superior balance between retention and adaptation. We attribute these improvements to three key factors: (1) region-aware editing that confines updates to preference-coherent regions, reducing interference with preserved knowledge; (2) dynamic routing with localized LoRA adapters that activates parameters aligned with current intents for fine-grained adaptation; and (3) incremental, constrained updates that mitigate catastrophic forgetting and stabilize S.

5.2 Ablation Study (RQ2)

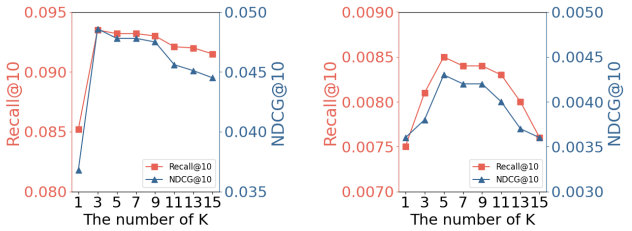
In order to validate the effectiveness of the proposed approach, we check the contribution of the main components of model to the final performance by comparing RAIE with the three variants:

- w/o KR: removing the Knowledge Region construction. comparing a single global adapter with multiple regional adapters.
- w/o KE: disabling the Region-Aware Editing. Evaluate its effectiveness for handling preference drift.
- w/o All: removing the Knowledge Region Construction and the Region-Aware Editing.



(a) MovieLens 10M (b) Yelp

Figure 3: The ablation study of RAIE.



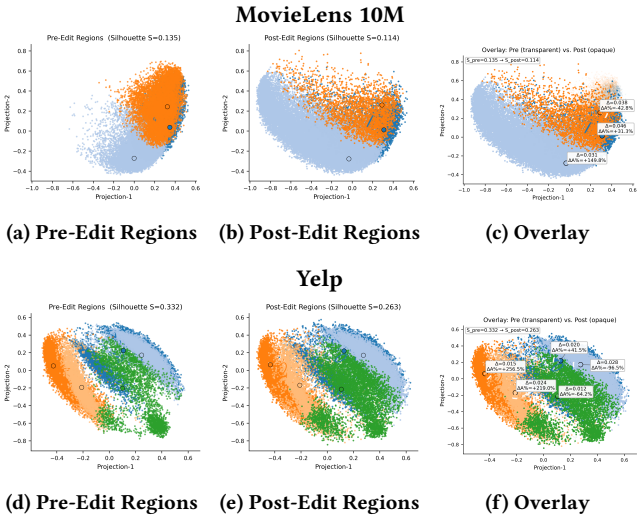
(a) MovieLens 10M (b) Yelp

Figure 4: Impact of region number K.

The experimental results are reported in Figure 3, we could summarize the following observations: (i) The proposed Region-Aware Editing contributes the most to performance enhancement. It is crucial to effectively update user preferences; moreover, region-aware editing effectively addresses preference drift. By editing the original preference representation to adapt to evolving interests, recommendation performance can be substantially improved. (ii) The variant "w/o KR" confronts conspicuous performance decay in recommendation. This indicates that constructing user preference knowledge regions effectively separates a user's different preferences. Compared with a single global adapter, multiple local adapters can capture the user's current preference from the perspective of each region and update preferences locally within that region. (iii) After removing both Knowledge Region Construction and Region-Aware Editing, the model's performance drops substantially, indicating that these two modules are both important and effective. Combining them further improves recommendation performance.

5.3 Hyperparameter Sensitivity (RQ3)

To analyze the effect of the number of regions K, we vary the K range and show the performance variation curves on both datasets in Figure 4. We found that: when there are few regions, the overall performance of model is poor. This again emphasizes the benefit of exploring multiple user preference regions. However, as the number of regions increases, the model performance shows a decreasing trend. This is because an increase in the number of regions gradually reduces the amount of useful information that each knowledge regions carries, and instead introduces more noise information, which impairs the model's performance. RAIE performs best on MovieLens-10M and Yelp with K = 3 and 5, respectively, which we attribute to the differences between the two datasets.



(a) Pre-Edit Regions (b) Post-Edit Regions (c) Overlay

(d) Pre-Edit Regions (e) Post-Edit Regions (f) Overlay

Figure 5: The visualization of region distributions.

5.4 Interpretability & Visualization (RQ4)

Finally, we examine interpretability by visualizing the knowledge regions before and after editing. Figure 5 visualizes pre-edit and post-edit knowledge regions for MovieLens and Yelp (with overlays) and annotates centroid displacement (Δ), relative boundary–area change (ΔA%), and region separability (S). Across both datasets, the overlays show that editing preserves the global partition while inducing local, targeted adjustments: centroids shift modestly and boundaries expand primarily near interfaces, reallocating coverage without re-partitioning the space. The separability metric changes in line with these boundary refinements, indicating comparable inter-region distinction after editing. Comparing domains, MovieLens exhibits larger Δ with moderate (ΔA%), whereas Yelp shows smaller centroid movement but substantially larger |ΔA%|, indicating stronger reallocation of boundary coverage. The separability metric S tends to decrease slightly after editing on both datasets, reflecting smoother inter-region transitions rather than new partitioning. Overall, RAIE adapts regional geometry to evolving preferences while maintaining an interpretable structure, consistent with the observed gains in recommendation quality.

6 Conclusion

We introduce RAIE, a plug-in, parameter-efficient framework for region-aware incremental preference editing. Its modular design is backbone-agnostic and robust under diverse drift patterns. While effective, RAIE currently relies on spherical k-means with a fixed K, making performance sensitive to this choice. Future work will focus on adaptive region discovery that learns both the number of regions and their boundaries, enhancing robustness.

7 Acknowledgements

This work is supported and sponsored by the Guangdong Basic and Applied Basic Research Foundation (2023A1515012848), National Natural Science Foundation of China (Nos. 62572410, 62576371 and T2122020), and CCF-DiDi GAIA Collaborative Research Funds.

References

[1] 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. In *Proceedings of the 17th ACM conference on recommender systems*. 1007–1014.

[2] I. S. Dhillon and D. S. Modha. 2001. Concept decompositions for large sparse text data using clustering. *Machine Learning* 42, 1 (Jan 2001), 143–175.

[3] Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. 2023. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature machine intelligence* 5, 3 (2023), 220–235.

[4] 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). In *Proceedings of the 16th ACM conference on recommender systems*. 299–315.

[5] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *International conference on machine learning*. PMLR, 2790–2799.

[6] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2022. Lora: Low-rank adaptation of large language models. *ICLR* 1, 2 (2022), 3.

[7] Chengkai Huang, Junda Wu, Yu Xia, Zixu Yu, Ruhan Wang, Tong Yu, Ruiyi Zhang, Ryan A Rossi, Branislav Kveton, Dongruo Zhou, et al. 2025. Towards agentic recommender systems in the era of multimodal large language models. *arXiv preprint arXiv:2503.16734* (2025).

[8] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*. IEEE, 197–206.

[9] Wang-Cheng Kang and Julian McAuley. 2018. Self-Attentive Sequential Recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*. 197–206.

[10] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences* 114, 13 (2017), 3521–3526.

[11] Marcus Klasson, Hedvig Kjellström, and Cheng Zhang. 2022. Learn the time to learn: Replay scheduling in continual learning. *arXiv preprint arXiv:2209.08660* (2022).

[12] Xiaoyu Kong, Jiancan Wu, An Zhang, Leheng Sheng, Hui Lin, Xiang Wang, and Xiangnan He. 2024. Customizing language models with instance-wise lora for sequential recommendation. *Advances in Neural Information Processing Systems* 37 (2024), 113072–113095.

[13] Yehuda Koren. 2009. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. 447–456.

[14] Chengyu Lai, Sheng Zhou, Zhimeng Jiang, Qiaoyu Tan, Yuanchen Bei, Jiawei Chen, Ningyu Zhang, and Jiajun Bu. 2024. Better Late Than Never: Formulating and Benchmarking Recommendation Editing. *arXiv preprint arXiv:2406.04553* (2024).

[15] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems* 33 (2020), 9459–9474.

[16] Jiaang Li, Quan Wang, Zhongnan Wang, Yongdong Zhang, and Zhendong Mao. 2025. ELDER: Enhancing Lifelong Model Editing with Mixture-of-LoRA. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 39. 24440–24448.

[17] Jiacheng Li, Yujie Wang, and Julian McAuley. 2020. Time Interval Aware Self-Attention for Sequential Recommendation. (2020), 322–330.

[18] Xuhong Li, Yves Grandvalet, and Franck Davoine. 2018. Explicit Inductive Bias for Transfer Learning with Convolutional Networks. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*. PMLR, 2825–2834.

[19] Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190* (2021).

[20] Zhizhong Li and Derek Hoiem. 2016. Learning Without Forgetting. 9908 (2016), 614–629.

[21] Yujie Lin, Chenyang Wang, Zhumin Chen, Zhaochun Ren, Xin Xin, Qiang Yan, Maarten de Rijke, Xiuzhen Cheng, and Pengjie Ren. 2023. A self-correcting sequential recommender. In *Proceedings of the ACM Web Conference 2023*. 1283–1293.

[22] 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 preprint arXiv:2308.12241* (2023).

[23] Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* (2017).

[24] Sichun Luo, Bowei He, Haohan Zhao, Wei Shao, Yanlin Qi, Yinya Huang, Aojun Zhou, Yuxuan Yao, Zongpeng Li, Yuanzhang Xiao, et al. 2025. Recranker: Instruction tuning large language model as ranker for top-k recommendation. *ACM Transactions on Information Systems* 43, 5 (2025), 1–31.

[25] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. *Advances in neural information processing systems* 35 (2022), 17359–17372.

[26] Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. 2022. Mass-editing memory in a transformer. *arXiv preprint arXiv:2210.07229* (2022).

[27] Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2021. Fast model editing at scale. *arXiv preprint arXiv:2110.11309* (2021).

[28] Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D Manning, and Chelsea Finn. 2022. Memory-based model editing at scale. In *International Conference on Machine Learning*. PMLR, 15817–15831.

[29] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. 2019. Continual lifelong learning with neural networks: A review. *Neural networks* 113 (2019), 54–71.

[30] 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. *Advances in Neural Information Processing Systems* 36 (2023), 10299–10315.

[31] Amit Rozner, Barak Battash, Lior Wolf, and Ofir Lindenbaum. 2024. Knowledge editing in language models via adapted direct preference optimization. *arXiv preprint arXiv:2406.09920* (2024).

[32] Tianhao Shi, Yang Zhang, Zhijian Xu, Chong Chen, Fuli Feng, Xiangnan He, and Qi Tian. 2024. Preliminary study on incremental learning for large language model-based recommender systems. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*. 4051–4055.

[33] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. 2017. Continual learning with deep generative replay. *Advances in neural information processing systems* 30 (2017).

[34] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. 2017. Continual Learning with Deep Generative Replay. 30 (2017).

[35] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*. 1441–1450.

[36] Chaoyong Wei, Jiang Wenjun, Kenli Li, and Jie Wu. 2025. Stability-aware Preference Modeling for Sequential Recommendation. *ACM Transactions on the Web* (2025).

[37] Xun Wu, Shaohan Huang, and Furu Wei. 2024. Mixture of lora experts. *arXiv preprint arXiv:2404.13628* (2024).

[38] Shuyuan Xu, Wenyue Hua, and Yongfeng Zhang. 2024. OpenP5: an Open-Source Platform for Developing, Training, and Evaluating LLM-based Recommender Systems. (2024).

[39] Yuhao Yang, Chao Huang, Lianghao Xia, and Chenliang Li. 2022. Knowledge graph contrastive learning for recommendation. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*. 1434–1443.

[40] Hyunsik Yoo, SeongKu Kang, and Hanghang Tong. 2025. Continual Recommender Systems. *arXiv preprint arXiv:2507.03861* (2025).

[41] Jin Zeng, Nan Wang, and Jinbao Li. 2025. Knowledge-driven hierarchical intents modeling for recommendation. *Expert Systems with Applications* 259 (2025), 125361.

[42] Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi, Shengyu Mao, Jintian Zhang, Yuansheng Ni, et al. 2024. A comprehensive study of knowledge editing for large language models. *arXiv preprint arXiv:2401.01286* (2024).