Exact and Efficient Unlearning for Large Language Model-based Recommendation

Zhiyu Hu, Yang Zhang, Minghao Xiao, Wenjie Wang, Fuli Feng and Xiangnan He

Abstract—Recent years have witnessed the trend of enhancing recommender systems with large language models (LLMs), namely, LLMRec. A common way is to fine-tune the LLMs with the instruction data transformed from user behaviors, stimulating the recommendation ability of LLMs. Similar to traditional recommender systems, integrating user data into LLMs raises privacy concerns. Users desire a tool to erase the impacts of their sensitive data from the trained models. To meet this user demand, LLMRec unlearning becomes pivotal to enable the removal of unusable data (*e.g.,* historical behaviors) from established LLMRec models. However, existing methods mostly focus on partition strategies and approximate unlearning. These methods are not well-suited for the unique characteristics of LLMRec due to computational costs or incomplete removal. In this study, we propose the Adapter Partition and Aggregation (APA) framework for exact and efficient LLMRec unlearning while maintaining recommendation performance. APA achieves this by retraining PEFT adapters using data partitioning, constructing adapters for partitioned training data shards, and retraining only the affected adapters. To preserve recommendation performance and avoid significant inference costs, APA incorporates balanced and heterogeneous data partitioning, and parameter-level adapter aggregation with sample-adaptive adapter attention for each testing sample. Extensive experiments demonstrate the effectiveness and efficiency of our method.

Index Terms—Large Language Models, Recommender System, Machine Unlearning, Recommendation Unlearning.

1 INTRODUCTION

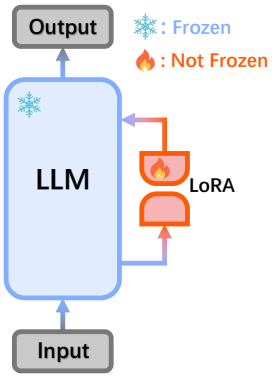
LARGE language models have demonstrated exceptional capabilities in content comprehension and generation, sparking interest in applying them in Web applications [1–6]. Recommender systems, as a primary channel for personalized content distribution, can also benefit from these capabilities in understanding items and users [7], pushing the emergence of large language model-based recommendation paradigm. The current standard approach for specializing LLMs for recommendation is parameter-efficient fine-tuning [7–10] using recommendation data. However, incorporating recommendation data (*e.g.,* historical behaviors) increases the risk of personal data leakage due to the vulnerability of LLMs [11–14]. To safeguard the privacy of users, particularly vulnerable populations, LLMRec unlearning becomes crucial, which targets the timely and effective removal of some personal data [15] (termed *unusable data* [16]) from developed LLMRec models.

To the best of our knowledge, there is currently limited research on LLMRec unlearning. Despite the significant advancements of recommendation unlearning for traditional models [16, 17], these approaches are unfeasible for LLMRec models due to the high computational cost associated with handling billions of model parameters. Some very recent studies [18–21] investigate efficient unlearning techniques for information encoded in a LLM by extending traditional methods [20] or utilizing in-context learning [18]. However, applying these methods for LLMRec will encounter the risk of incomplete removal due to their approximate nature. Although [21] provides valuable insights into unlearning in the LLM-based recommendation setting, it does not target exact unlearning. In contrast, LLMRec unlearning requires complete removal of the unusable data to comply with relevant regulations such as the General Data Protection Regulation [22]. Additionally, LLMRec unlearning must maintain overall recommendation performance to ensure a satisfactory user experience.

Achieving desirable LLMRec unlearning hinges on retraining PEFT adapters using data partitioning. Inspired by traditional unlearning methods [15, 23, 24], retraining has proven to be a reliable method for ensuring exact unlearning. By employing a partitioning strategy that divides the training data into disjoint shards and training sub-models for each shard, unlearning efficiency can be maintained as only the sub-models affected by unusable data are retrained. Since personal data is exclusively stored in the PEFT adapter (*e.g.,* LoRA [25]), retraining adapters on relevant shards incurs relatively low costs. Additionally, the PEFT adapter can quickly learn from a minimal number of examples, enabling further reduction in shard size and retraining costs. Considering these factors, we propose leveraging an adapter partition-empowered retraining approach for LLMRec unlearning.

The partition strategy in LLMRec presents a distinct challenge in terms of inference latency since aggregating prediction results from different adapters is necessary to integrate knowledge for maintaining high recommendation performance [15, 23, 24]. However, such aggregation becomes infeasible for LLMRec models due to the computa-

This work is supported by the National Natural Science Foundation of China (U24B20180, 62121002). (Corresponding Authors: Yang Zhang and Xiangnan He.)
Zhiyu Hu, Minghao Xiao, Wenjie Wang, Fuli Feng, and Xiangnan He are with the University of Science and Technology of China, China. Email: zhiyuhu@mail.ustc.edu.cn, xiaominghao@mail.ustc.edu.cn, wenjiewang96@gmail.com, fulifeng93@gmail.com, xiangnanhe@gmail.com.
Yang Zhang is with the National University of Singapore. Email: zyang1580@gmail.com.



Instruction Input	
Instr.:	Given the user's preference and unpreference, ...
Input:	User preference: ... User unpreference: ... Target new movie: ...
Instruction Output	
Output:	Yes.

Fig. 1: The left diagram illustrates the classic structure of LoRA, while the right table provides a sample for recommendation instruction data.

tionally expensive nature of LLM inference [26]. Generating predictions from K adapters would result in a K times increase in the inference cost of LLMs, leading to substantial rises in energy consumption and service latency. For this challenge, we consider adapter aggregation at parameter level to enable a single-pass inference. Additionally, the partition and aggregation should be carefully designed to ensure a recommendation performance comparable to adapter retraining without partitioning [27].

To this end, we introduce the *Adapter Partition and Aggregation* framework for exact and efficient LLMRec unlearning while maintaining overall recommendation performance. APA trains individual adapters on partitioned training data shards and leverages adapter weight aggregation during inference. As to partition, APA divides the training data into balanced and heterogeneous shards based on semantic characteristics [9] to facilitate keeping recommendation performance [27]. As to aggregation, we adopt a sample-adaptive approach for each testing sample that assigns adapter attention based on the performance of adapters on similar validation samples. This prioritizes higher-performing adapters, thereby enhancing overall performance. Notably, additional training is unnecessary for our adaptive aggregation, unlike traditional unlearning methods [23, 24], thus avoiding extra unlearning costs.

The main contributions can be summarized as follows:

- New problem: To our knowledge, this is the first study to formulate and explore exact unlearning within the realm of LLMRec.
- New technique: We introduce the APA method, an extension of partition-based unlearning, tailored to scenarios where inference involves high computational costs.
- Experiments: We conduct extensive experiments on three real-worlds, verifying the effectiveness of the proposal.

2 PRELIMINARIES

In this section, we first briefly introduce the prerequisite knowledge of PEFT and LLMRec. Then, we give the problem formulation.

2.1 PEFT

Fine-tuning LLMs with domain-specific data is an effective method to tailor LLMs for domain-specific tasks. Given that LLMs typically comprise billions of parameters, full tuning is a resource-intensive and time-consuming process. Recent work [28] shows that LLMs have a low intrinsic dimension that can match the performance of the full parameter space. PEFT provides a solution to this challenge by keeping the most of model weights frozen and only updating a part of the parameters. These learnable parameters are controlled by an adaptation module (termed adapter).

LoRA. In this work, we focus on *Low-Rank Adaptation* (LoRA) [25], a prominent and widely adopted PEFT solution. To make fine-tuning more efficient, LoRA adds pairs of rank-decomposition weight matrices to existing weights of the LLM in a plug-in manner and only trains the newly added weights for learning tasks. The rank-decomposition design would ensure that the addition of weight matrices introduces only a small number of learnable parameters, thereby expediting the fine-tuning process. More specifically, for a matrix multiplication layer within LLMs, a LoRA module adds weight matrices as follows:

$$o = W_0 x + BAx, \tag{1}$$

where x and o represent the input and the resulting output, respectively. $W_0 \in \mathcal{R}^{d_1 \times d_2}$ denotes the original model weight matrix, while $B \in \mathcal{R}^{d_1 \times r}$ and $A \in \mathcal{R}^{r \times d_2}$ constitute the pair of rank-decomposition weight matrices, with d_1, d_2, and r representing the dimensions involved. Notably, $r \ll \min(d_1, d_2)$, meaning that the number of parameters introduced by BA is significantly fewer than that of W_0 because $d_1 r + r d_2 \ll d_1 d_2$. During the fine-tuning process, only A and B are adjustable. In a similar way, a LoRA module (also called a LoRA adapter) is generally applicable to any LLM layer desired for updating.

2.2 LLMRec

PEFT with recommendation data has emerged as a de facto standard to specialize LLMs for recommendation tasks. Numerous studies have appeared and showcased the effectiveness of such LLMRec methods. In our research, we focus on the two widely popular LLMRec methods called TALLRec [8] and Rella [29], considering their broad applicability and representation of the general paradigm in the field of LLMRec.

TALLRec. TALLRec employs LoRA tuning techniques to align LLMs with the recommendation task using recommendation instruction data. This approach involves the conversion of user-item interaction data into language instructions, as exemplified in Figure 1. Each instruction comprises both an input and an output component. Within the instruction input, TALLRec represents items using their titles and user preferences or non-preferences are conveyed by referencing historical item titles. It also instructs LLMs to respond with either "Yes" or "No" to indicate the user's preference for a target item. The response is included in the instruction output. With the instruction data, TALLRec performs fine-tuning of the LLM using a LoRA adapter to learn the recommendation task. Let $\mathcal{D}$ represent the set of all converted instruction data for training, and then the optimization problem can be formulated as follows:

$$\max_{\Phi} \sum_{(x_i, y_i) \in \mathcal{D}} \sum_{t=1}^{|y|} log(P_{\Theta_0 + \Phi}(y_{it} | x_i, y_{i<t})), \tag{2}$$

where x and y represent the instruction input and output of a data sample in $\mathcal{D}$, y_t represents the t-th text token of y, $y_{<t}$ denotes the text tokens that precede y_t, and $P_{\Theta_0+\Phi}(y_t|x, y_{<t}))$ signifies the predictive probability of y_t by the LLM. Θ_0 refers to the existing parameters of the original LLM, and Φ encompasses all model parameters within the LoRA adapter, including the A and B as defined in Equation (1) for all layers. Notably, only the LoRA adapter parameters Φ would be updated.

Rella. Rella, like TALLRec, utilizes LoRA tuning and instruction data based on user-item interactions to align large language models (LLMs) with recommendation tasks. However, Rella introduces a key enhancement through Retrieval-Enhanced Instruction Tuning (ReiT), which improves the model's ability to process long behavior sequences. Instead of simply truncating the most recent K behaviors, ReiT selects the most semantically relevant K behaviors in relation to the target item, enhancing the quality of the input data. Similar to TALLRec, Rella instructs LLMs to respond with "Yes" or "No" to indicate user preferences for a target item. Additionally, Rella adopts the same optimization problem (1) as TALLRec, focusing on maximizing the log-probability of generating correct outputs based on the input data. This allows Rella to improve recommendation accuracy by leveraging more relevant user behavior information.

2.3 Problem Formulation

Let $\mathcal{D}_{-r} \subset \mathcal{D}$ represent the data that a user wishes to remove from a PEFT LLMRec model f that was initially trained with $\mathcal{D}$. Following previous work, we assume the size of $\mathcal{D}_{-r}$ is very small, *e.g.*, $|\mathcal{D}_{-r}| = 1$. We try to obtain a retrained model using only the remaining data, denoted as $\mathcal{D}_r = \mathcal{D} - \mathcal{D}_{-r}$, to achieve exact unlearning. Simultaneously, this unlearning process needs to be efficient in order to respond to the user's request promptly. Finally, we aim to minimize any performance degradation after implementing the unlearning designs to ensure that users remain satisfied with the recommendation quality.

3 METHODOLOGY

In this section, we commence with presenting an overview of our approach, encompassing the model framework and the unlearning process. We provide a detailed discussion of the pivotal components of our method.

3.1 Overview

To enable exact and efficient unlearning based on retraining, our APA framework employs a partitioning strategy to train and construct the LLMRec model. Our APA framework, as illustrated in Figure 2, encompasses three key phases:

1) Data and Adapter Partition: We partition the training data $\mathcal{D}$ into K balanced and disjoint shards, denoted as $\{\mathcal{D}_1, \ldots, \mathcal{D}_K\}$. Given that the LLM relies on text semantics for predictions, we perform the partition based on the text semantics of the samples, utilizing a K-means clustering method. Once the data shards are obtained, we proceed to train an individual LoRA adapter (a sub-adapter) for each shard in TALLRec. For the k-th data shard $\mathcal{D}_k$, we train a LoRA adapter parameterized with Φ_k according to Equation (4) (replacing $\mathcal{D}$ and Φ in the equation with $\mathcal{D}_k$ and Φ_k).
2) Adapter Aggregation: At the serving stage, we perform adapter aggregation, which involves merging the weights of different LoRA adapters to create a unified adapter. We just use the aggregated LoRA adapter for inference. Importantly, we employ a sample-adaptive aggregation strategy, tailoring the aggregated adapter to the specific sample for improved performance. This part is the key to ensuring performance and inference efficiency.

Unlearning. When a user requests to erase data $\mathcal{D}_{-r}$, only the sub-LoRA adapters affected by $\mathcal{D}_{-r}$ need to be retrained, obviating the need to retrain the entire model and facilitating acceleration. In theory, we only need to invest a $\frac{|\mathcal{D}_{-r}|}{K}$ cost for full retraining to achieve precise unlearning. With the support of two considerations, $\frac{|\mathcal{D}_{-r}|}{K}$ can be kept at a low value, resulting in significant acceleration: Firstly, it is often assumed that user requests arrive in a streaming manner. Usually, only one sample needs to be unlearned at a time. Secondly, given the few-shot learning capabilities of TALLRec (a few hundred samples are adequate to train an effective LoRA), the data partition can be fine-grained, allowing for a relatively high value of K.

After the LLMRec has been constructed, the unlearning process is simple and straightforward. Therefore, the essence of the process lies in our partition and aggregation stages. We now delve into the details of these two phases.

3.2 Partition

The partitioning phase is the key to training a LLMRec model, involving two key parts: 1) data partition, and 2) training a sub-adapter module for each partitioned data shard. We next elaborate on the two parts.

3.2.1 Data Partition

For data partitioning, the crucial factor is ensuring that data within the same shard share related knowledge, creating homogeneity of knowledge within a shard and heterogeneity across shards. This aids in the effective learning of sub-adapters with limited data and their subsequent aggregation. Prior methods for recommendation unlearning relied on collaborative embedding to perform partitioning, utilizing the K-means algorithm to group samples with similar collaborative information within the same shard. Similarly, given that LLMRec relies on text semantics for prediction, we propose partitioning data based on semantics.

Our data partition method is detailed in Algorithm 1, consisting of three key parts. Initially, we utilize the original LLM (without fine-tuning) to derive the hidden representations (denoted as h_i) of the input instructions x_i for each training sample (x_i, y_i) in $\mathcal{D}$, capturing the text semantics (line 2). Subsequently, we employ K-means on the obtained hidden representations, resulting in K clusters and K clustering centers (denoted as $a_1, \ldots, a_K$) (line 3). The clusters generated directly by K can be highly unbalanced, potentially making unlearning inefficient for large shards. Therefore, we take further steps to balance the clusters (lines 4-10). Instead of directly assigning a sample to the nearest cluster, we take into account the cluster size: if the size of the closest cluster exceeds a certain threshold t, we assign the sample to the nearest cluster whose size is still below

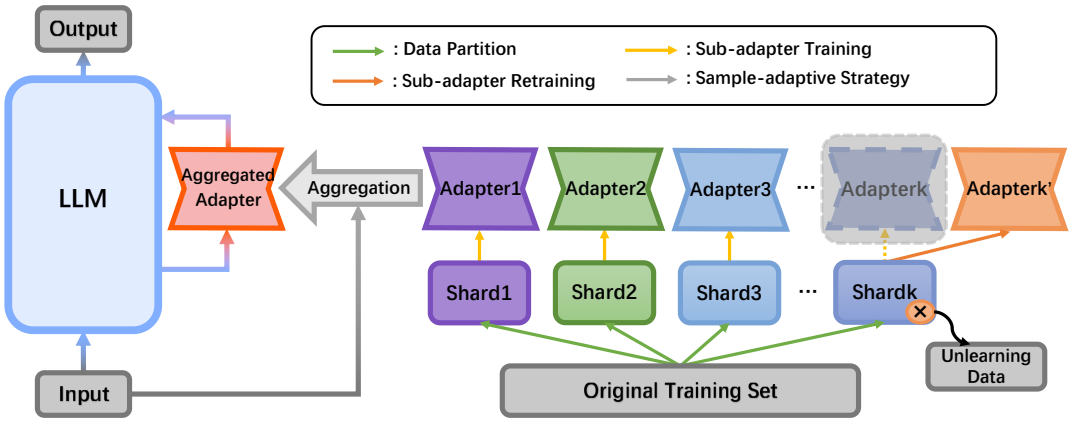


Fig. 2: Illustration of our APA framework, which consists of three parts: Data Partition, Adapter training, and Adapter aggregation. When a user requests to erase data D_{-r}, only the sub- LoRA adapters affected by D_{-r} need to be retrained.

that threshold. Formally, for all samples, we calculate their cosine distance to each cluster center as follows:

$$dist(h_i, a_k) = -cos(h_i, a_k). \tag{3}$$

We store all distances in a list, denoted as $F = \{(x_i, y_i, dist(h_i, a_k))|i \leq |\mathcal{D}|, k \leq K\}$, and then sort the list based on the $dist()$ function. We refer to the sorted list as F_s. Subsequently, we orderly examine each element $(x_{i'}, y_{i'}, dist(h_{i'}, a_{k'}))$ in F_s to achieve balanced clustering. If (x'_i, y'_i) has not yet been assigned to any cluster, we assign it to the k'-th cluster, denoted as $\mathcal{D}_{k'}$.

3.2.2 Sub-adapter Training

After obtaining the partitioned data, we proceed to train an individual LoRA adapter for each data shard, following the approach of TALLRec. Formally, for the k-th data shard, the optimization objective is as follows:

$$\max_{\Phi_k} \sum_{(x_i,y_i)\in\mathcal{D}_k} \sum_{t=1}^{|y|} log(P_{\Theta_0+\Phi_k}(y_i|x_i)), \tag{4}$$

where Φ_k denotes the model parameters of the k-th LoRA for the k-th data shard, $P_{\Theta_0+\Phi_k}(y_i|x_i)$ denotes the prediction probability of LLMRec with the k-th LoRA for y_i.

3.3 Aggregation

During the serving prediction stage, aggregating knowledge from the sub-models is essential to enhance the overall prediction quality. Typically, prediction result aggregation is a commonly used approach. However, this method necessitates performing LLM inference K times, as it requires computing $P_{\Theta_0+\Phi_k}(y_i|x_i)$ for k=1, 2, . . . , K. To address this challenge, we introduce aggregation at the LoRA adapter model weight level, *i.e.*, adapter aggregation. This technique combines the weights of multiple LoRA adapters, creating a single LoRA adapter that allows for one-pass prediction.

Given that a weight matrix in the original LLM corresponds to a pair of rank-decomposition weight matrices, as shown in Equation 1, we consider two levels of aggregation:

- Decomposition level: At this level, each model weight of LoRA serves as the unit for model aggregation. We directly aggregate the A and B matrices defined in Equation (1) from different LoRA adapters using weight averaging. Formally, an aggregated LoRA layer can be defined as follows:

$$\begin{aligned} \bar{o} &= W_0 x + \bar{B}\bar{A}x, \\ \bar{B} &= \sum_{k=1}^{K} \omega_k B_k,\ \bar{A} = \sum_{k=1}^{K} \omega_k A_k, \end{aligned} \tag{5}$$

where $\bar{B}$ represents the aggregated B matrix, B_k represents the B matrix of the k-th sub-adapter, similarly for those of A; $\bar{o}$ denotes the layer output in the aggregated LoRA adapter, and ω_k is the aggregation weight for the k-th sub-adapter, where a higher value indicates higher attention. The method for assigning ω_k is described later.

- Non-decomposition level: At this level, the weight unit of the original LLM serves as the aggregation unit for the adapter aggregation. Then, we aggregate the "BA" result defined in Equation (1) from all sub-LoRA adapters using weight averaging. Formally, an aggregated LoRA layer can be formulated as follows:

$$\begin{aligned} \bar{o} &= W_0 x + \overline{BA}x, \\ \overline{BA} &= \sum_{k=1}^{K} \omega_k B_k A_k, \end{aligned} \tag{6}$$

where $\overline{BA}$ represents the aggregated LoRA weights, and other symbols have the same meanings as in Equation (5).

Sample-adaptive Strategy. Different testing samples require varying levels of knowledge from different sub-models for accurate prediction, suggesting the need for an adaptive attention allocation when aggregating sub-adapters. To avoid introducing additional training and unlearning, we explore a heuristic approach to assign aggregation weights to different sub-adapters. Based on our partition, one straightforward solution is to use text semantic similarity to determine these weights, giving higher priority to the adapter corresponding to the data shards with greater similarity to the sample. However, selecting an adapter solely based on input similarity doesn't guarantee better prediction accuracy.

Algorithm 1: Balanced Semantic-aware Data Partition

Input: training instruction data $\mathcal{D}$, cluster number K, and maximum size of each shard t
Output: The Shards $\{\mathcal{D}_1, ..., \mathcal{D}_K\}$

Initialize $\mathcal{D}_1, \ldots, \mathcal{D}_K$;
Compute hidden representation h_i of x_i in the original LLM for each training sample $(x_i, y_i) \in \mathcal{D}$;
Running the K-means with all hidden representations $\{h_i | i \leq |\mathcal{D}|\}$, obtaining cluster centers: $\{a_1, a_2, ..., a_K\} = K\text{-}means(\{h_i | i \leq |\mathcal{D}|\}, K)$;
For each sample (x_i, y_i) and each cluster center a_k, compute their cosine distance using h_i, *i.e.*, $dist(h_i, a_k)$, storing $(x_i, y_i, dist(h_i, a_k))$ in a list F;
Sort F in ascending order to get F_s;
for *each* $(x'_i, y'_i, dist(h_{i'}, a_{k'}))$ *in* F_s **do**
  if $|\mathcal{D}_{k'}| < t$ *and* (x_i, y_i) *has not been assigned* **then**
    $\mathcal{D}_{k'} \leftarrow \mathcal{D}_{k'} \cup (x_i, y_i)$
  end
end
return $\{\mathcal{D}_1, ..., \mathcal{D}_K\}$;

To address this concern, we devise a method that leverages validation prediction errors to enhance the assignment mechanism. In essence, for each testing sample, we rely on the prediction errors of the most similar validation samples to assess the suitability of a particular adapter for that specific testing sample and allocate the attention weights accordingly. This approach ensures more accurate weight assignments for effective prediction.

Specifically, for each testing sample (x, y), we initially identify the top-n most similar samples from the validation set, calculating the similarities similar to Equation (3). These identified similar samples are denoted as $\mathcal{N}_v$. Next, we measure the average prediction error among $\mathcal{N}_v$ for each sub-adapter as follows:

$$error_k = \frac{1}{|\mathcal{N}_v|} \sum_{(x_i, y_i) \in \mathcal{N}_v} error(y, P_{\Theta + \Phi_k}(y_i | x_i)), \tag{7}$$

where $|\mathcal{N}_v|$ represents the size of $\mathcal{N}_v$, and $error_k$ stands for the average prediction error of the k-th sub-LoRA adapter. Subsequently, for this testing sample, we assign higher attention weights to the sub-LoRA adapter with lower prediction errors. Formally, the attention weight ω_k for the k-th sub-LoRA adapter is calculated as follows:

$$\omega_k = \frac{exp(\tau \cdot -error_k)}{\sum_{k'=1}^{K} exp(\tau \cdot -error_{k'})}, \tag{8}$$

where τ represents the temperature parameter, controlling the strength of the assignment mechanism. When $\tau = 0$, the mechanism becomes ineffective, allocating equal attention weights for all sub-LoRA adapters.

3.4 Discussion

Extension to Other PEFT-based LLMRec Models. It is worth mentioning that our method is developed specifically for LoRA-based LLMRec. However, since the PEFT method commonly incorporates adaptation modules, we can directly extend our method to LLMRec models developed using other PEFT techniques like Adapter Tuning [30]. Because Adapter Tuning shares structural similarities with the LoRA architecture, featuring two decomposable parts of an up-down Feed-Forward network (FFN), it enables the aggregation of network parameters in our approach. This flexibility allows us to apply our method to a wider range of LLMRec architectures.

Unlearning Efficiency. Our method remains efficient when more than one data point requires removal, as it allows simultaneous retraining of the affected sub-models. However, when a large number of unlearning requests arrive in a short time, repeated retraining can result in significant cumulative cost—an inherent limitation of data partition-based approaches. A potential solution is batch unlearning, which handles multiple requests together to avoid redundant updates. Inspired by [31], one can also use inference consistency checks to decide when retraining is truly necessary, reducing overhead in practice. Future work could explore adaptive batch unlearning strategies that dynamically determine the optimal grouping of unlearning requests based on their impact scope and timing.

Robustness to Validation Set Quality. Our sample-adaptive strategy indeed relies on validation samples to guide weight assignment. We construct the dataset by applying a sliding window over each user's historical interactions. The training, validation, and test sets are then randomly partitioned, ensuring consistency across these splits. While selecting validation samples that closely resemble the test set may enhance aggregation precision, such an approach implicitly relies on future data and diverges from real-world application scenarios. For real-world applications, it may be important to adopt more effective time-aware validation strategies, such as using the most recent historical data for validation and dynamically updating it as new data becomes available. This approach ensures that the method is performed on up-to-date data, better capturing current user data distribution and controlling the potential distribution shifts between the validation and test sets. We leave a more in-depth exploration of validation strategies to future work.

4 EXPERIMENTS

In this section, we conduct a series of experiments to answer the following research questions:

- **RQ1**: How does APA perform in terms of recommendation performance and unlearning efficiency compared to the state-of-the-art exact unlearning methods for LLMRec?
- **RQ2**: How does APA perform in terms of inference efficiency?
- **RQ3**: How do different components of the proposed APA influence its effectiveness?
- **RQ4**: How do different hyper-parameters affect the performance of APA?

We first present experimental settings, followed by results and analyses to answer each research question.

4.1 Experimental Settings

4.1.1 Datasets

We conduct experiments on three widely used real-world datasets in the recommender systems domain. As we focus

on the CTR (Click-Through Rate) prediction task, we follow the standard preprocessing approach [32] to convert explicit ratings into implicit feedback. The details of the three datasets are as follows:

- **Book**. We utilize the BookCrossing dataset [33], which consists of user ratings ranging from 1 to 10. This dataset also provides textual descriptions of books, including attributes like "book author" and "book title". To reduce the sparsity of the dataset, we filtered out users who interacted with items fewer than three times. Furthermore, we binarize the ratings based on a threshold of 5. That is, ratings exceeding 5 are considered as "like", while ratings below 5 are labeled as "dislike".
- **Movie**. We utilize the MovieLens100K [34] benchmark dataset, which comprises user ratings ranging from 1 to 5, following [35, 36], we treat the ratings as "like" (corresponding to the "Yes" in the TALLRec instruction out) if the ratings are higher than 3 and otherwise "dislike". The dataset includes comprehensive textual attributes of the movies, such as "title" and "director".
- **Game**. We utilize the Amazon Video Games dataset [37], which contains user ratings ranging from 1 to 5. We binarize the ratings by treating scores higher than 3 as "like" and scores of 3 or below as "dislike". The dataset also provides rich textual information for each game, including attributes such as "title" and "brand".

We strictly adhere to the pre-processing procedures outlined in the TALLRec paper [8] for data filtering, data splitting, and instruction data construction for all three datasets. In particular, considering TALLRec and Rella's capability to efficiently learn recommendations and yield good performance with a minimal number of training samples, we constrain our training size to 1024 (larger than the maximum size of 256 mentioned in the TALLRec paper). Similar to the TALLRec setting, both the validation and test sets consist of 1,000 samples for each of the Movie, Book, and Game datasets.

4.1.2 Compared Methods

We focus on exact unlearning, but there is currently no specific work designed for LLM. Therefore, to serve as a baseline, we consider extending the following traditional exact unlearning baselines to TALLRec and Rella:

- **Retraining** This represents the straightforward retraining approach, *i.e.,* retraining the entire model from scratch while excluding the unusable data. We implement it by retraining TALLRec from scratch, excluding the unusable data. This method serves as the gold standard in terms of recommendation performance.
- **SISA [15]** is the earliest known partition-enhanced retraining method. It randomly divides data and aggregates sub-model predictions through methods such as averaging or majority voting. We extend this approach to TALLRec, employing its average-based aggregation.
- **RecEraser [23]** is a recommendation-specific unlearning method, sharing similarities with SISA but incorporating unique partitioning strategies to preserve collaborative information. We adapt it for LLMRec based on its UBP version. Notably, its prediction aggregation involves training with K TALLRec, requiring overmuch computational resources. In our adaptation, we directly replace it with the aggregation strategy of SISA.
- **GraphEraser [24]** is an unlearning method designed for graph-structured data (including the bipartite graph structure of interaction data). It employs node clustering techniques, namely BEKM and BLPA, for graph data partitioning. We extend GraphEraser to TALLRec using the BEKM-based partition. Similar to RecEraser, we adopt the aggregation strategy of SISA for it.

Regarding our APA, we implement two versions using different levels of aggregation, as defined in Section 3.3. We denote the version with decomposition-level aggregation as APA(D) and the version with non-decomposition-level aggregation as APA(ND).

4.1.3 Evaluation Setting

Our objective is to achieve precise and efficient unlearning for LLMRec while preserving recommendation performance. Therefore, our evaluation focuses on three aspects: 1) the completeness of data removal, 2) unlearning efficiency, and 3) recommendation performance. Since all compared methods are built on retraining (from scratch) without the unusable data, the first aspect is inherently maintained. Following the RecEraser paper [23], we do not consider this aspect for evaluation. To assess recommendation performance, we use the Area under the ROC Curve (AUC) metric, following the settings in TALLRec and Rella. Additionally, we introduce another performance loss metric Δ to measure the recommendation performance loss of a method relative to the Retraining method. This metric is calculated by the difference in AUC between the method and the Retraining method, with higher values indicating less performance loss. For evaluating unlearning efficiency, we directly utilize the unlearning time (retraining time). Additionally, considering the inference cost for LLMs, we further compare the inference time.

4.1.4 Implementation Details

As all methods utilize TALLRec and Rella as the backbone recommendation model, we apply the same hyper-parameter settings for them to learn the recommendation model, following the original configuration outlined in both papers. For the LoRA configuration, we follow previous works [8] and set the parameters as follows: the LoRA rank is set to 8, the LoRA alpha is set to 16, and the LoRA dropout is set to 0.05. The LoRA module is applied to the query and value projection matrices of attention blocks. Concerning the unlearning setting, for all partition-based base models, we set the shard size to 256 for the data partition, resulting in $K = 4$ training data shards. For the specific hyper-parameters of the baselines, we tune them in accordance with the settings provided in the original papers, whenever available for our extension. Regarding the proposed APA, we set the threshold parameter τ (in Equation (8)) to 500, 1500, and 600 for the Movie, Book, and Game datasets, respectively. For the neighbor size $|\mathcal{N}_v|$ in (7), we set it to 20 for Movie, 100 for Book, and 40 for Game. All these hyper-parameters are tuned on the validation set, and all experiments are conducted on the same machine equipped with the NVIDIA A40 GPUs.

TABLE 1: Comparison of different unlearning methods on recommendation performance based on the TALLRec and Rella backbones, where '**APA(D)**'/'**APA(ND)**' represents APA implemented with decomposition/non-decomposition level aggregation, and $\triangle$ represents the gap between retraining and the unlearning method in terms of AUC. 'Bef. Agg.' represents the average AUC of the sub-model.

Datasets	Methods	TallRec			Rella		
		Beff. Agg.	AUC	delta	Beff. Agg.	AUC	delta
Book	**Retraining**	-	0.6738	-	-	0.7085	-
	SISA	0.6561	0.6731	-0.0007	0.6794	0.6928	-0.0157
	GraphEraser	0.6393	0.6646	-0.0092	0.6810	0.6923	-0.0162
	RecEraser	0.6525	0.6719	-0.0019	0.6734	0.6889	-0.0196
	APA(D)	0.6578	0.6738	0	0.6779	0.7023	-0.0062
	APA(ND)	0.6578	0.6741	0.0003	0.6779	0.7026	-0.0059
Movie	**Retraining**	-	0.7428	-	-	0.7716	-
	SISA	0.7003	0.7055	-0.0373	0.7208	0.7252	-0.0464
	GraphEraser	0.6732	0.6885	-0.0543	0.7130	0.7214	-0.0502
	RecEraser	0.6699	0.6918	-0.0510	0.6721	0.7391	-0.0325
	APA(D)	0.6874	0.7264	-0.0164	0.7192	0.7414	-0.0302
	APA(ND)	0.6874	0.7262	-0.0166	0.7192	0.7414	-0.0302
Game	**Retraining**	-	0.7383	-	-	0.7982	-
	SISA	0.6571	0.6832	-0.0551	0.7035	0.7220	-0.0762
	GraphEraser	0.6111	0.6919	-0.0464	0.6744	0.6991	-0.0991
	RecEraser	0.6488	0.6856	-0.0527	0.6787	0.7287	-0.0695
	APA(D)	0.6171	0.7076	-0.0307	0.6615	0.7744	-0.0238
	APA(ND)	0.6171	0.7069	-0.0314	0.6615	0.7725	-0.0257

4.2 Main Results (RQ1)

In this subsection, we evaluate all unlearning methods based on two criteria: recommendation performance and unlearning efficiency. It is essential to note that all compared methods inherently achieve complete removal of unusable data. Therefore, we omit the comparison in the aspect of exact unlearning [23].

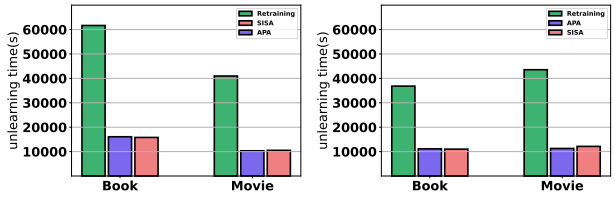
4.2.1 Accuracy Comparison

To compare the accuracy of APA with the baselines, we conducted experiments using two different backbone models to assess their ability to maintain recommendation performance during unlearning. Higher ability is indicated by less performance loss compared to the Retraining method. The comparison results for both backbones are summarized in Table 1, where we additionally include the averaged performance of the sub-models for the partition-based methods (denoted as "Bef. Agg."). We also report $\triangle$, which reflects the AUC difference from Retraining (smaller values indicate better performance). The following observations are drawn from the table:

- All methods with aggregation demonstrate improved AUC compared to the averaged AUC of their corresponding sub-models. This underscores the significance of aggregating knowledge from sub-models to enhance performance.
- Compared to the baselines, APA exhibits less performance loss compared to the reference Retraining method and, on TALLRec's Book dataset, can even bring improvements. These results highlight the superior ability of our method to maintain recommendation performance during unlearning. This superiority can be attributed to the compatibility between our partitioning method and aggregation, as well as the adaptive aggregation approach based on validation performance, which pays more attention to high-performance sub-adapters.
- In contrast, SISA, RecEraser, and GraphEraser show much inferior performance compared to the reference method Retraining. Particularly on the movie dataset, these baselines exhibit a significant decline in recommendation performance, with the performance loss reaching -0.0373, -0.0510, and -0.0543, respectively. This suggests that the direct application of traditional exact unlearning methods to TALLRec results in a substantial compromise in recommendation performance.
- The two versions of APA with different levels of adapter aggregation (APA(D) and APA(ND)) demonstrate similar performance. This indicates that treating the LoRA rank-decomposed parameter as the aggregation unit or the original LLM parameter unit as the aggregation unit does not affect the effectiveness of our adaptive aggregation method.
- Our APA(D) and APA(ND) demonstrate better unlearning performance across three evaluation datasets and two backbone LLM-based recommendation models, confirming their broad applicability.
- Rella consistently outperforms TALLRec in terms of AUC across all three datasets, indicating its superior capability in handling recommendation tasks by leveraging retrieval-enhanced instruction tuning.

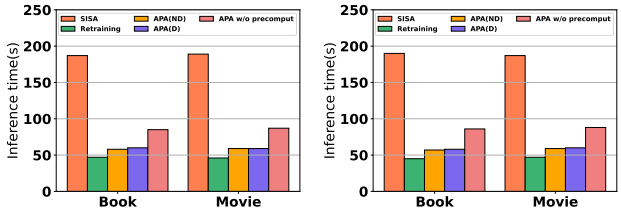
4.2.2 Unlearning Efficiency Comparison

We next conduct experiments to explore the unlearning efficiency of our APA. We fully follow the efficiency evaluation experiment setting in the RecEraser paper [23], ensuring that only one sub-model needs to be retrained for unlearning each time. We primarily compare our method with both the Retraining and SISA methods, as other baselines theoret-



(a) **Unlearning time - TALLRec** (b) **Unlearning time - Rella**

Fig. 3: (a) (b) Comparison of unlearning time among Retraining, APA, and SISA across both backbones.



(a) **Inference time - TALLRec** (b) **Inference time - Rella**

Fig. 4: (a) (b) Inference time of Retraining, SISA, APA(D), and APA(ND) on both backbones.

ically incur similar unlearning efficiency costs to those of APA and SISA. The results are shown in Figure 3(a) and Figure 3(b). We omit the results on the Game dataset here, as they exhibit a similar trend to those on the Book and Movie datasets. The results demonstrate that APA significantly improves unlearning efficiency. For example, on the movie dataset, APA only took 10,335 seconds and 11,280 seconds to achieve the optimal performance for TALLRec and Rella, making it about 3.96 times and 3.86 times faster (approximately $=$ K) than the Retraining method. The Retraining method is time-consuming as it is trained on the whole dataset. In contrast, APA only requires retraining the specific sub-model responsible for the unlearned data. From the figures, we can also observe that APA and SISA exhibit similar unlearning times, as both adopt data-partition-based unlearning strategies. Moreover, when the training data is large, we can keep a small shard size to allow for a large number of data shards K, considering that TALLRec and Rella can effectively learn recommendations with few samples. In this case, APA could achieve greater acceleration as long as only a few sub-adapters are affected by unusable data.

4.3 Inference Time Comparison (RQ2)

In the previous section, we explored how our APA method can significantly reduce time during the unlearning process. In this section, we investigate whether APA can improve efficiency during the inference stage compared to baselines. We randomly selected 500 samples from the testing set and measured the total inference time for these samples, ensuring that only one LLM inference could be executed at a time. Specifically, in our default APA method, embeddings and validation errors are precomputed and cached. As a result, online inference with APA only requires similarity computation and attention-based model aggregation, which are the primary costs we measure. However, to provide a more comprehensive evaluation, we also consider a variant where embeddings and validation errors are computed during testing, denoted as "APA w/o precomput." For baseline comparisons, we include SISA and Retraining, and omit other baselines due to their similar computational costs to the SISA method.

The experimental results are presented in Figure 4(a) and Figure 4(b). We do not show results on the Game dataset for brevity, as the inference efficiency trends are consistent with those observed on the Book and Movie datasets. From the results, we have the following observations: 1) Our APA method exhibits small gaps in time efficiency compared to a single model inference (Retraining), and the delay for each sample is just approximately 0.02 seconds, which is entirely acceptable in real-world scenarios. 2) The two versions of APA with different levels of adapter aggregation (APA(D) and APA(ND)) demonstrate similar inference costs, indicating that the two levels of aggregation have comparable time complexity. 3) SISA has much higher inference time costs compared to Retraining and APA. This is because SISA performs prediction-level aggregation for sub-adapters, which involves additional time for inference cost. 4) "APA w/o precomput" requires additional computation due to the need for calculating the testing set embeddings, but even with this extra overhead, our APA's inference efficiency remains significantly higher than that of SISA. These results demonstrate the effectiveness of APA's aggregation designs in enhancing inference efficiency.

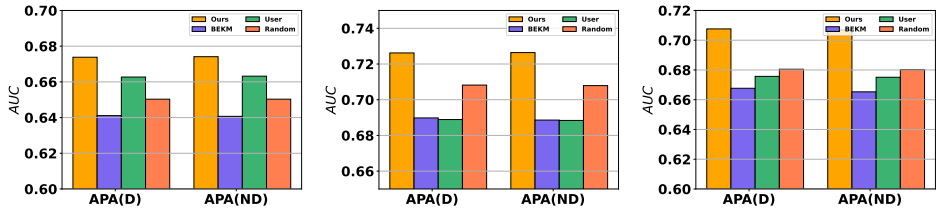
4.4 In-depth Studies(RQ3)

4.4.1 *Effect of the Data Partition Methods*

To validate the effectiveness of our proposed data partition method, we compare it with three other grouping methods: random partition, the user-based partition of RecEraser [23], and BEKM partition [24], denoted as 'Random,' 'User,' and 'BEKM', respectively. We replace the original partition method with the three methods in our APA, respectively, and then compare their performances with the original one. The experimental findings are illustrated in Figure 5. We conduct experiments on all three datasets and draw the following observations. Replacing the original semantic-aware partitioning with any of the three alternative methods leads to performance degradation. For instance, on the Book dataset, the semantic-aware method achieves an AUC of 0.6738, whereas random partitioning, user-based partitioning, and BEKM only achieve 0.6503, 0.6627, and 0.6411, respectively. These results highlight the importance of semantic-guided partitioning in ensuring data shard heterogeneity, which facilitates more effective model aggregation and improved recommendation performance.

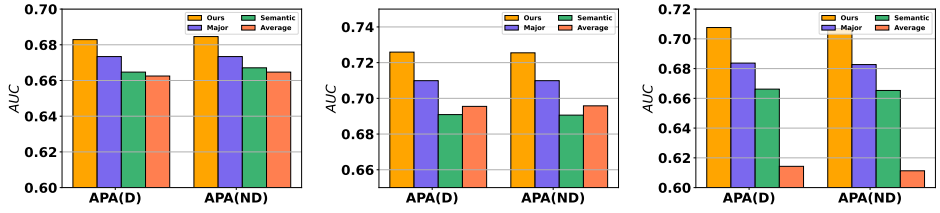
4.4.2 *Effect of the Sample-Adaptive Method*

We proceed to assess the model utility of different aggregation weight assignment methods to demonstrate the effectiveness of our proposed sample-adaptive method. We compare our methods with the following three choices: 1) average-based, assigning equal weight for each sub-adapter, 2) major-based, assigning all weights to the one with the highest ω_k computed by our method, 3) semantic-based,



(a) **Book** (b) **Movie** (c) **Game**

Fig. 5: Performance comparison of different data partition methods on Book, Movie, and Game datasets. APA(D) and APA(ND).



(a) **Book** (b) **Movie** (c) **Game**

Fig. 6: Performance comparison of different aggregation weight assignment methods on Book, Movie, and Game datasets.

which assigns weight according to the semantic similarity of the sample to the center of different shards. We conduct a comparison between the APA implemented with our assignment method and the variants of APA implemented using the three different methods. The experimental results are presented in Figure 6, with 'Average,' 'Major,' and 'Semantic' denoting the compared three choices, respectively. Here are some observations we found: 1) The average-based method underperforms our sample-adaptive method on both datasets, highlighting the importance of assigning different weights for different adapters; 2) The semantic-based method also exhibits worse recommendation performance than our original method, confirming the effectiveness of utilizing validation performance information; 3) Using only the best sub-model choice by our weight assignments can maintain relatively high recommendation performance, but there is still a gap compared to our method, as shown by the results of the major-based method. These results emphasize the effectiveness of our weight assignments, and meanwhile, the importance of aggregating knowledge from different sub-adapters.

4.5 Hyper-parameters Studies(RQ4)

We study how the hyper-parameters affect the recommendation performance of APA on the TALLRec backbone, more specifically, the shard size, neighbor size, and temperature parameter.

4.5.1 *Impact of the Shard Size*

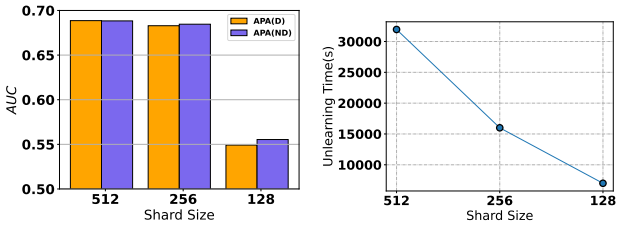
We investigate the influence of the shard size on the Book dataset. We configure the shard size as $\{512, 256, 128\}$ and evaluate unlearning efficiency and recommendation performance. The experimental results are displayed in Figure 7. We find that 1) as the shard size decreases, the unlearning time significantly reduces; 2) In terms of recommendation performance, as the shard size decreases, it remains relatively stable before decreasing. For example, when the shard size is 512 and 256, the performance of APA remains very close, and it only significantly decreases at 128. Although each sub-model trained on 128 samples can still reach the normal performance level of TALLRec, the increased number of sub-models introduces greater complexity in aggregation. This makes it harder to assign accurate attention weights, ultimately leading to a decline in the overall model performance. This indicates that within a certain range, we can improve unlearning efficiency by reducing the data shard size (increasing the number of shards) while maintaining comparable recommendation performance. In this way, on the one hand, the cost of retraining individual shards decreases, and on the other hand, increasing the number of shards may keep the proportion of adapters requiring retraining relatively low, thereby enhancing unlearning efficiency.

4.5.2 *Impact of the Neighbor Size* $|\mathcal{N}_v|$

We delve deeper into exploring the impact of the neighbor size $|\mathcal{N}_v|$ on the Book dataset. We fix the temperature parameter at 1500 and adjust the size of the neighbor from $\{10, 50, 100, 150, 200\}$. As shown in the figure 8(a), APA achieves optimal performance when $|\mathcal{N}_v| = 100$. Furthermore, as increases, there is a noticeable declining trend in the recommendation performance, indicating that the neighbor size needs to be kept at a reasonable magnitude. Too few similar samples can result in a lack of knowledge, while too many samples can introduce redundant knowledge. In this case, using a suitable neighbor size is beneficial.

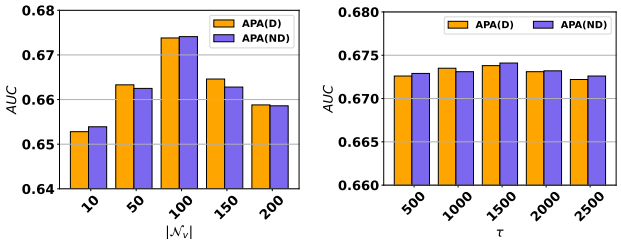
4.5.3 *Impact of the Temperature Parameter* τ

To explore the impact of the temperature parameter on APA performance, we fix the neighbor size at 100 and adjust the temperature parameter from $\{500, 1000, 1500, 2000, 2500\}$.



(a) **Recommendation Performance** (b) **Unlearning Time**

Fig. 7: Impact of the shard size on the unlearning efficiency and performance on Book dataset. (a) shows the recommendation performance and (b) shows the unlearning time cost.



(a) **Neighbor Size** (b) **Temperature Parameter**

Fig. 8: Impact of the neighbor size and temperature parameter on recommendation performance on Book dataset.

It is worth noting that large temperature values are used because sub-model errors are typically small. Without proper scaling, softmax becomes too smooth, assigning similar weights to all sub-models. A larger temperature sharpens the distribution, better highlighting high-performing sub-models during aggregation. As illustrated in the figure 8(b), the model performance gradually improves as the temperature parameter increases. However, after reaching a relatively large temperature coefficient, a decline begins to occur. This is related to when the parameter is too small, critical adapters cannot receive sufficient attention. Conversely, when the coefficient is too large, similar to the major-based approach mentioned in section 4.4.2, it overlooks the influence of other adapters, leading to a decrease in recommendation performance. These results suggest that during aggregation, it is essential to have a reasonable temperature parameter to allocate appropriate attention to each adapter.

5 RELATED WORK

5.1 Machine Unlearning

• **Machine Unlearning**. Machine unlearning, the process of removing partial training data information from trained machine learning models, is essential in various domains, including recommendation, for reasons such as privacy and security concerns [38–40]. This concept is known as machine unlearning [15]. In traditional machine learning, two main technique lines for unlearning have emerged: approximate unlearning and exact unlearning [41, 42]. Approximate unlearning aims for unlearning without retraining, using techniques like influence functions [16, 43] and data augmentation [44, 45] for extreme efficiency, but it often involves incomplete removal of the data. On the other hand, exact unlearning [15] typically involves retraining, ensuring complete unlearning but in a time-costly manner. Existing work, like SISA [23, 24, 31, 46], focuses on partition strategies, building individual sub-models for partitioned training data shards to retrain only partial sub-models. Our method, while also based on the partition strategy, addresses new challenges posed by the large scale and high inference cost of Large Language Models (LLMs). This makes our work distinct from existing methods.

• **LLM Unlearning**. The challenges presented by Large Language Models (LLMs), particularly their large scale, bring forth new considerations for unlearning. Previous efforts [18–20] have explored unlearning for LLMs, but they often involve approximate methods. For instance, [19] simulates data labels to approximate the next-token predictions of a model that has not been trained on the unusable data, and then fine-tunes the LLM on these simulated labels for unlearning. [18] proposes "In Context Unlearning", which leverages in-context learning by flipping labels of unusable data to achieve approximate unlearning. [20] leverages the gradient ascent to erase the influence of unusable data on a trained model with fine-tuning. In the context of LLM-based recommender systems, [21] introduces a dual-teacher framework to balance forgetting and retention by aligning the unlearned model with teacher outputs. Yet, it still falls under approximate unlearning. In contrast, our approach focuses on LLMRec and strives for exact unlearning, considering the significant impact of incomplete removal of sensitive data.

5.2 Model Aggregation in LLM

To aggregate the different models, there are two strategies: 1) output aggregation and 2) model weight aggregation. Output aggregation has been widely studied, and applied for aggregation process for partition-based unlearning, but could introduce inefficiency for LLM. Regarding the model weight aggregation, existing work focuses on leveraging it to better finish tasks like image classification, multi-domain learning [47], Cross-lingual information extraction [48], etc. To our knowledge, we are the first to leverage it for the unlearning task. Meanwhile, from the technical view, our method has significant differences from existing work on the aggregation weight assignment. To achieve weight assignment, previous works usually considered 1) average aggregation [27], 2) greedy aggregation [27], and 3) learning-based aggregation like simulating Mixture-of-Experts (MoE) [47, 48]. Differently, we innovatively assign weights to different sub-models based on the prediction quality of similar verification samples, which can achieve effective adaptive weight assignments without learning.

6 CONCLUSION

In this work, we introduce a novel and efficient framework, which, to the best of our knowledge, is the first exact unlearning method designed for large language model-based recommendation (LLMRec). To achieve efficient unlearning while preserving high recommendation performance, we propose an Adapter Partition and Aggregation

(APA) approach, which consists of 1) a balanced semantic-aware data partition method, and 2) a sample-adaptive method. Additionally, we employ parameter-level adapter aggregation to create an aggregated adapter to mitigate the high inference cost associated with traditional methods. We carry out comprehensive experiments on three real-world datasets, offering insightful analysis of the effectiveness and efficiency of our approach in removing interaction data.

In our future endeavors, we aim to expand our unlearning paradigm to encompass other PEFT methods (*e.g.,* Adapter Tuning [30], and Prompt Tuning [49]), thus widening the adaptability of our method across diverse LLMRec architectures. At present, how to efficiently unlearn using the data partition framework in batch settings is still an open problem [15, 31]. We are exploring methods to enhance unlearning efficiency in batch settings, enabling the management of more unlearn data.

REFERENCES

[1] K. Singhal, S. Azizi, T. Tu, S. S. Mahdavi, J. Wei, H. W. Chung, N. Scales, A. Tanwani, H. Cole-Lewis, S. Pfohl *et al.*, "Large language models encode clinical knowledge," *Nature*, vol. 620, no. 7972, pp. 172–180, 2023.

[2] P. Gao, J. Han, R. Zhang, Z. Lin, S. Geng, A. Zhou, W. Zhang, P. Lu, C. He, X. Yue *et al.*, "Llama-adapter v2: Parameter-efficient visual instruction model," *arXiv preprint arXiv:2304.15010*, 2023.

[3] Z. Zhang, A. Zhang, M. Li, H. Zhao, G. Karypis, and A. Smola, "Multimodal chain-of-thought reasoning in language models," *Transactions on Machine Learning Research*, vol. 2024, 2024.

[4] X. Zhao, J. You, Y. Zhang, W. Wang, H. Cheng, F. Feng, S.-K. Ng, and T.-S. Chua, "Nextquill: Causal preference modeling for enhancing llm personalization," *arXiv preprint arXiv:2506.02368*, 2025.

[5] X. Zhao, L. Wang, Z. Wang, H. Cheng, R. Zhang, and K.-F. Wong, "Pacar: Automated fact-checking with planning and customized action reasoning using large language models," in *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, 2024, pp. 12 564–12 573.

[6] X. Zhao, Y. Deng, W. Wang, H. Cheng, R. Zhang, S.-K. Ng, T.-S. Chua *et al.*, "Exploring the impact of personality traits on conversational recommender systems: A simulation with large language models," *arXiv preprint arXiv:2504.12313*, 2025.

[7] Y. Zhang, F. Feng, J. Zhang, K. Bao, Q. Wang, and X. He, "Collm: Integrating collaborative embeddings into large language models for recommendation," *IEEE Transactions on Knowledge and Data Engineering*, 2025.

[8] K. Bao, J. Zhang, Y. Zhang, W. Wang, F. Feng, and X. He, "Tallrec: An effective and efficient tuning framework to align large language model with recommendation," in *Proceedings of the 17th ACM Conference on Recommender Systems, RecSys 2023*, 2023, pp. 1007–1014.

[9] K. Bao, J. Zhang, W. Wang, Y. Zhang, Z. Yang, Y. Luo, C. Chen, F. Feng, and Q. Tian, "A bi-step grounding paradigm for large language models in recommendation systems," *ACM Transactions on Recommender Systems*, vol. 3, no. 4, pp. 1–27, 2025.

[10] J. Fu, F. Yuan, Y. Song, Z. Yuan, M. Cheng, S. Cheng, J. Zhang, J. Wang, and Y. Pan, "Exploring adapter-based transfer learning for recommender systems: Empirical studies and practical insights," *The 17th ACM International Conference on Web Search and Data Mining*, 2024.

[11] A. G. Carranza, R. Farahani, N. Ponomareva, A. Kurakin, M. Jagielski, and M. Nasr, "Privacy-preserving recommender systems with synthetic query generation using differentially private large language models," *arXiv preprint arXiv:2305.05973*, 2023.

[12] H. Duan, A. Dziedzic, N. Papernot, and F. Boenisch, "Flocks of stochastic parrots: Differentially private prompt learning for large language models," *Advances in Neural Information Processing Systems*, vol. 36, pp. 76 852–76 871, 2023.

[13] S. Zanella-Béguelin, L. Wutschitz, S. Tople, V. Rühle, A. Paverd, O. Ohrimenko, B. Köpf, and M. Brockschmidt, "Analyzing information leakage of updates to natural language models," in *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*, 2020, pp. 363–375.

[14] N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson *et al.*, "Extracting training data from large language models," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 2633–2650.

[15] L. Bourtoule, V. Chandrasekaran, C. A. Choquette-Choo, H. Jia, A. Travers, B. Zhang, D. Lie, and N. Papernot, "Machine unlearning," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 141–159.

[16] Y. Zhang, Z. Hu, Y. Bai, J. Wu, Q. Wang, and F. Feng, "Recommendation unlearning via influence function," *ACM Transactions on Recommender Systems*, vol. 3, no. 2, pp. 1–23, 2024.

[17] W. Liu, J. Wan, X. Wang, W. Zhang, D. Zhang, and H. Li, "Forgetting fast in recommender systems," *arXiv preprint arXiv:2208.06875*, 2022.

[18] M. Pawelczyk, S. Neel, and H. Lakkaraju, "In-context unlearning: language models as few-shot unlearners," in *Proceedings of the 41st International Conference on Machine Learning*, 2024, pp. 40 034–40 050.

[19] R. Eldan and M. Russinovich, "Who's harry potter? approximate unlearning in llms," *arXiv preprint arXiv:2310.02238*, 2023.

[20] Y. Yao, X. Xu, and Y. Liu, "Large language model unlearning," *Advances in Neural Information Processing Systems*, vol. 37, pp. 105 425–105 475, 2024.

[21] H. Wang, J. Lin, B. Chen, Y. Yang, R. Tang, W. Zhang, and Y. Yu, "Towards efficient and effective unlearning of large language models for recommendation," *Frontiers of Computer Science*, vol. 19, no. 3, p. 193327, 2025.

[22] G. D. P. Regulation, "General data protection regulation (gdpr)," *Intersoft Consulting, Accessed in October*, vol. 24, no. 1, 2018.

[23] C. Chen, F. Sun, M. Zhang, and B. Ding, "Recommendation unlearning," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 2768–2777.

[24] M. Chen, Z. Zhang, T. Wang, M. Backes, M. Humbert,

and Y. Zhang, "Graph unlearning," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 499–513.

[25] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen *et al.*, "Lora: Low-rank adaptation of large language models." *ICLR*, vol. 1, no. 2, p. 3, 2022.

[26] J. Kaddour, J. Harris, M. Mozes, H. Bradley, R. Raileanu, and R. McHardy, "Challenges and applications of large language models," *CoRR*, 2023.

[27] M. Wortsman, G. Ilharco, S. Y. Gadre, R. Roelofs, R. Gontijo-Lopes, A. S. Morcos, H. Namkoong, A. Farhadi, Y. Carmon, S. Kornblith *et al.*, "Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time," in *International Conference on Machine Learning*. PMLR, 2022, pp. 23 965–23 998.

[28] A. Aghajanyan, S. Gupta, and L. Zettlemoyer, "Intrinsic dimensionality explains the effectiveness of language model fine-tuning," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2021, pp. 7319–7328.

[29] J. Lin, R. Shan, C. Zhu, K. Du, B. Chen, S. Quan, R. Tang, Y. Yu, and W. Zhang, "Rella: Retrieval-enhanced large language models for lifelong sequential behavior comprehension in recommendation," in *Proceedings of the ACM on Web Conference 2024*, 2024, pp. 3497–3508.

[30] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for nlp," in *International Conference on Machine Learning*. PMLR, 2019, pp. 2790–2799.

[31] Y. Hu, J. Lou, J. Liu, W. Ni, F. Lin, Z. Qin, and K. Ren, "Eraser: Machine unlearning in mlaas via an inference serving-aware approach," in *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, 2024, pp. 3883–3897.

[32] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," in *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, 2009, pp. 452–461.

[33] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen, "Improving recommendation lists through topic diversification," in *Proceedings of the 14th international conference on World Wide Web*, 2005, pp. 22–32.

[34] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *Acm transactions on interactive intelligent systems (tiis)*, vol. 5, no. 4, pp. 1–19, 2015.

[35] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "Lightgcn: Simplifying and powering graph convolution network for recommendation," in *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, 2020, pp. 639–648.

[36] Y. Zhang, T. Shi, F. Feng, W. Wang, D. Wang, X. He, and Y. Zhang, "Reformulating ctr prediction: learning invariant feature interactions for recommendation," in *Proceedings of the 46th international ACM SIGIR conference on research and development in information retrieval*, 2023, pp. 1386–1395.

[37] J. Ni, J. Li, and J. McAuley, "Justifying recommendations using distantly-labeled reviews and fine-grained aspects," in *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, 2019, pp. 188–197.

[38] Y. Cao and J. Yang, "Towards making systems forget with machine unlearning," in *2015 IEEE symposium on security and privacy*. IEEE, 2015, pp. 463–480.

[39] N. G. Marchant, B. I. Rubinstein, and S. Alfeld, "Hard to forget: Poisoning attacks on certified machine unlearning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 7, 2022, pp. 7691–7700.

[40] Y. Li, X. Feng, C. Chen, and Q. Yang, "A survey on recommendation unlearning: Fundamentals, taxonomy, evaluation, and open questions," *arXiv preprint arXiv:2412.12836*, 2024.

[41] H. Xu, T. Zhu, L. Zhang, W. Zhou, and P. S. Yu, "Machine unlearning: A survey," *ACM Comput. Surv.*, vol. 56, no. 1, aug 2023.

[42] T. T. Nguyen, T. T. Huynh, Z. Ren, P. L. Nguyen, A. W.-C. Liew, H. Yin, and Q. V. H. Nguyen, "A survey of machine unlearning," *ACM Trans. Intell. Syst. Technol.*, 2025.

[43] Z. Izzo, M. A. Smart, K. Chaudhuri, and J. Zou, "Approximate data deletion from machine learning models," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2021, pp. 2008–2016.

[44] S. Shan, E. Wenger, J. Zhang, H. Li, H. Zheng, and B. Y. Zhao, "Fawkes: Protecting privacy against unauthorized deep learning models," in *29th USENIX security symposium (USENIX Security 20)*, 2020, pp. 1589–1604.

[45] A. K. Tarun, V. S. Chundawat, M. Mandal, and M. Kankanhalli, "Fast yet effective machine unlearning," *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[46] W. Qian, C. Zhao, H. Shao, M. Chen, F. Wang, and M. Huai, "Patient similarity learning with selective forgetting," in *2022 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, 2022, pp. 529–534.

[47] S. Diao, T. Xu, R. Xu, J. Wang, and T. Zhang, "Mixture-of-domain-adapters: Decoupling and injecting domain knowledge to pre-trained language models' memories," in *The 61st Annual Meeting Of The Association For Computational Linguistics*, 2023.

[48] T. Li, Z. Wang, L. Chai, J. Yang, J. Bai, Y. Yin, J. Liu, H. Guo, L. Yang, H. Zine el abidine *et al.*, "Mt4crossoie: Multi-stage tuning for cross-lingual open information extraction," *Expert Systems with Applications*, vol. 255, p. 124760, 2024.

[49] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2021, pp. 4582–4597.