Journal of Computer Science and Artificial Intelligence
ISSN: 3078-8242 (Print), ISSN: 3078-2309 (Online) | Vol. 4, No. 1, 2025

Machine Unlearning Scheme for Recommendation System Based on Gradient Attribution

Weiping Peng, Quanjiang Liu [*], Di Ma, Cheng Song, Yinmeng Jiang

College of Computer Science and Technology, Henan Polytechnic University, JiaoZuo, China

[*] **Corresponding author**: QuanJiang Liu

Abstract: Addressing the current data privacy leakage challenges faced by BERT-based sequential recommendation systems, this study innovatively proposes a gradient attribution-based machine unlearning scheme for recommendation systems. The scheme first leverages gradient attribution techniques to precisely evaluate the contribution of each intermediate neuron to the prediction of user behavior information. Based on these evaluation results, we adopt a filtering strategy designed to identify and remove redundant neurons while retaining those "privacy neurons" closely associated with specific sensitive user behavior information. Subsequently, the activation values of these identified privacy neurons are set to zero, thereby completely eliminating their contribution to model prediction. This effectively achieves the goal of erasing specific categories of user behavior information from the recommendation system model. Compared to traditional model retraining methods, this scheme offers a significant advantage in unlearning speed and has a minimal impact on the overall performance of the recommendation system, providing an efficient and practical new approach to addressing data privacy issues in sequential recommendation systems.

Keywords: Machine Unlearning; Gradient Attribution; Recommendation System; Bert; Privacy Protection Technology.

1. Introduction

In recent years, with the development of Internet technology, the era of big data has arrived, where information explosion is occurring. Recommender systems serve as an effective approach to address the problem of information overload, playing an increasingly important role in various fields such as e-commerce [1, 2], social networks [3, 4], location-based services, and more. Unlike traditional content-based and collaborative filtering recommender systems, sequence-based recommender systems aim to understand and model users' continuous behavior to more accurately describe their context, intent, and consumption trends of items, ultimately achieving more accurate, personalized, and dynamic recommendations. A well-trained recommender system model may have implicitly memorized user data. Firstly, considering privacy issues, large models such as recommender systems [5] and natural language models [6] may collect, use, and leak personal information without user permission. Users hope to make the model forget their sensitive data. Secondly, considering system utility, user preferences are time-sensitive and need to be updated continuously [7]. For example, a user who needs a certain product will not be interested in recommendations about that product for some time after completing the purchase. In this case, users would like to modify some data to enable the system to provide more accurate recommendations. Therefore, the recommender system model needs to forget certain sensitive data and its complete context in many cases. Machine unlearning aims to achieve the goal of unlearning sensitive data in the recommender system model without affecting performance, by erasing traces of specific individuals or data points in the model. The most primitive method of machine unlearning is to remove the data to be forgotten from the training data and retrain the model, but this method brings significant computational and time costs. Currently, there are some efforts dedicated to addressing the inefficiency of machine unlearning. Bourtoule et al. [8] introduced the Unlearning Framework SISA, which uses dividing the training data into multiple balanced slices to accelerate the retraining process through data fragmentation. Liu et al. [23] studied deleting data from the training model in the federated learning scenario. Similar to SISA [8], the retraining process can be accelerated by caching the intermediate parameters of the model, but these methods cannot be directly applied to sequence-based recommender systems.

To achieve efficient machine unlearning in the recommendation system, Yuan et al. [10] first studied the problem of machine unlearning in federated recommendation systems and proposed a negative sampling method and an update selection mechanism based on the importance of storing only important model updates. The method calibrates historical model updates to quickly recover the federated recommendation system model. Although this method achieves efficient machine unlearning, it is only applicable to recommendation system scenarios under federated learning. Subsequently, Chen et al. [24] to maintain the collaborative information of the data, divided the recommendation data into balanced slices and proposed an attention-based adaptive aggregation method to further improve the unlearning efficiency. Although this data division method can improve the unlearning efficiency, it damages the sequential recommendation performance and is not suitable for the sequential recommendation scenario. Li et al. [25] continued to use the idea of dividing data and proposed a general framework for unlearning recommendation system models, LASER, by dividing users into balanced groups based on the similarity of collaborative embeddings. Then, following the idea of curriculum learning, the model was trained for all groups sequentially to achieve efficient machine unlearning while ensuring system utility. However, applying this framework to the sequential recommendation system based on Bert is more complex. In summary, existing solutions have

improved the unlearning efficiency of different recommendation architectures, but they are not suitable for the sequential recommendation system based on Bert.

Addressing the limitations of existing solutions, this paper proposes a gradient-based attribution machine unlearning scheme for BERT-based sequential recommendation systems. This scheme aims to tackle the unlearning problem by first identifying and filtering privacy neurons—those expressing user behavior information—using a gradient-based attribution method. When specific categories of user behavior data need to be unlearned, these selected privacy neurons are directly set to zero. This not only achieves the objective of unlearning user behavior data and protecting user privacy information but also eliminates the impact of target data on the recommendation system model. Experimental results validate the rationality and effectiveness of this scheme.

The main contributions of this paper can be summarized in three aspects:

(1) We propose a sequential recommendation system machine unlearning method based on gradient attribution, which identifies intermediate neurons storing user behavior information through gradient attribution.

(2) After identifying the neurons, we analyze and screen them to obtain privacy neurons and then erase them.

(3) Experimental results on different datasets demonstrate that the proposed machine unlearning strategy can effectively eliminate the impact of user-specified behavior information on the recommendation system.

2. Preliminary

2.1. Bert

Bidirectional Encoder Representations from Transformers (BERT) is a pre-trained natural language processing model. It consists of multiple Transformer encoders, each of which is composed of multiple self-attention layers and feedforward neural network layers. Let $X \in \mathbb{R}^{n\times d}$ represent the input matrix, and the two modules can be expressed as follows:

$$Q_h = XW_h^Q, K_h = XW_h^K, V_h = XW_h^V \quad (1)$$

$$\text{Self} - \text{Att}_h(X) = \text{softmax}\left(Q_h K_h^T\right)V_h \quad (2)$$

$$\text{FFN}(H) = \text{gelu}(HW_1)W_2 \quad (3)$$

Where $W_h^Q, W_h^K, W_h^V, W_1, W_2$ is the parameter matrix; $\text{Self} - \text{Att}_h(X)$ calculates a single attention head; H represents the hidden state; gelu denotes the GELU activation function. The self-attention layer helps the model understand the context information in the input text, while the feedforward neural network layer helps the model learn the semantic information in the input text.

With the success of BERT, some research has combined BERT with recommendation systems [17, 18]. For example, to provide more accurate recommendations, Islek and Oguducu [17] use BERT to extract project embedding vectors from unstructured project description text. However, these methods only use BERT to process text input. In contrast, the Bert4Rec [19] model has a unique positioning, focusing on sequential recommendation tasks, aiming to predict the next project that the user may interact with.

2.2. Gradient Integration Algorithm

Sundararajan et al. [20] attribute the predictions of deep networks to their input features and define two basic axioms that attribution methods should satisfy: sensitivity and implementation invariance. Based on these two axioms, a new attribution method called Integrated Gradients is proposed. The Integrated Gradients algorithm combines the design ideas of direct gradients and reverse-propagation-based attribution techniques, Let the input value be x, the baseline value be x', and the function mapping be represented by F. The integral gradient definition for the i-th dimension of the input is as follows:

$$\text{IntegratedGrads}_i(x) ::= (x_i - x_i') \times \int_{\alpha=0}^{1} \frac{\partial F\left(x' + \alpha \times (x - x')\right)}{\partial x_i} d\alpha \quad (4)$$

Where $F: R^{-n} \to [0, 1]$ represents the deep network, $x \in R^n$ represents the input, and $x' \in R^n$ represents the baseline input. The attribution of the i-th component x can be regarded as the cumulative sum of all gradients on the straight path from the baseline x' to the input x. That is, the attribution of component i is the integral of the gradient path from x' to x. Here, $\frac{\partial F(x)}{\partial x_i}$ is the gradient of $F(x)$ along the i-th dimension. Hao et al. [21] propose a self-attention attribution method to explain the information interaction inside the Transformer and demonstrate that the attribution results can be used as adversarial patterns to achieve non-targeted attacks on BERT. Dai et al. [27] introduce the concept of knowledge neurons and explore how implicit knowledge is stored in pre-trained large models.

3. System Model

The main title, appearing on the first page, should be centered and positioned beneath the figure at the top of the page. It must be set in 14-point Times New Roman, boldface. Capitalize the first letter of all nouns, pronouns, verbs, adjectives, and adverbs; do not capitalize articles, coordinating conjunctions, or prepositions unless the title begins with such a word. Leave a 10.5-point space above the title and a 14-point space below it.

The unlearning strategy can be generalized to most existing sequence recommendation systems based on BERT. In this paper, we consider selecting the sequence recommendation model Bert4Rec proposed by Alibaba [19].

In the sequence recommendation system, let $U = \{u_1, u_2, \ldots, u_{|U|}\}$ be the set of users, $V = \{v_1, v_2, \ldots, v_{|V|}\}$ be the set of items, and $N_u = \left[v_1^{(u)}, \ldots, v_t^{(u)} \ldots, v_{n_u}^{(u)}\right]$ be the user's historical behavior sequence. The goal of sequence recommendation is to predict the probability of user interaction with each item at the next time step based on the historical behavior sequence: $p\left(v_{n_{u+1}}^u = v \mid N_u\right)$.

The overall framework of the proposed sequential recommendation system is schematically represented in Figure 1, comprising four interconnected components: the input layer, embedding layer, Transformer module (Trm Block), and prediction layer.

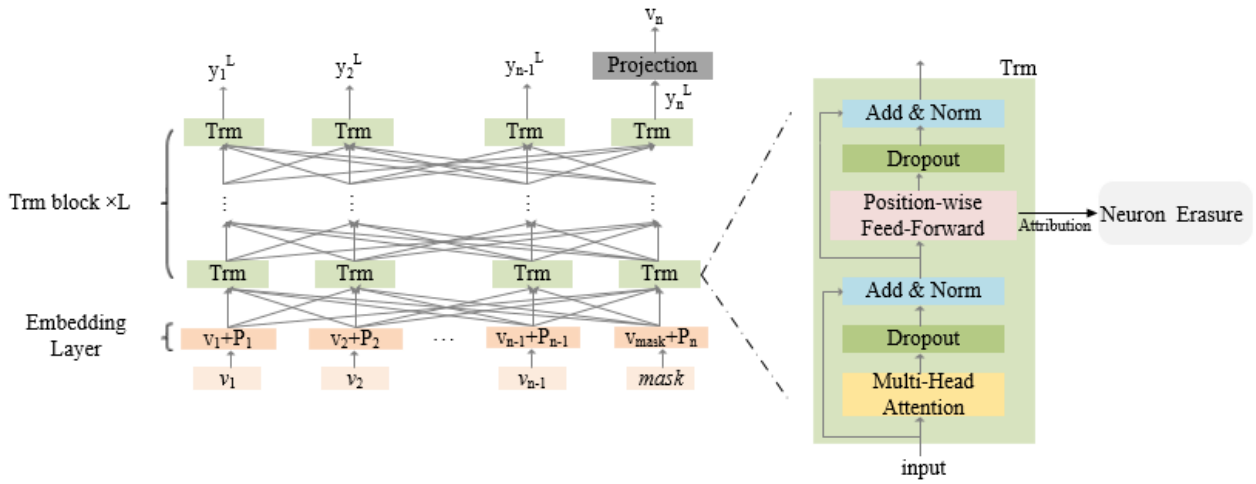

Figure 1. The overall model training process of our proposed architecture

The operational workflow initiates at the input layer, which serves as the primary entry point for raw user interaction data. The user's chronological sequence of interactions, such as clicks, purchases, or browsing behaviors, is collected and organized into an ordered list, denoted as $V = \{v_1, v_2, \ldots, v_{|V|}\}$. This sequence directly constitutes the initial input for subsequent model processing and prediction, ensuring that all relevant user behavior data is accurately captured, thus laying a robust foundation for complex analyses.

The raw interaction sequence proceeds to the embedding layer. This layer's fundamental function is to transform discrete item identifiers into continuous, semantically meaningful vector representations—a critical prerequisite for deep learning models handling non-numeric data. Within this layer, each item $v_{|V|}$ in the sequence is mapped into a dense vector space, yielding its corresponding item embedding. Crucially, recognizing that the Transformer module is inherently insensitive to the sequential order of input items, the embedding layer strategically introduces positional encodings (P_i). Each item's embedding vector is then element-wise added to its corresponding positional encoding P_i, based on its position within the sequence. This additive operation enables the model to discern both the relative and absolute positions of items, thereby effectively leveraging the intrinsic sequential nature of user interactions.

The processed sequence, now rich in semantic and positional information, is subsequently fed into the Transformer module (Trm Block), which forms the core of the system. This module is responsible for deep feature extraction and intricate pattern learning on the embedded sequence. Comprising L stacked bidirectional Transformer layers, this architecture is designed to capture long-range and complex global dependencies within the sequence, while precisely incorporating the positional information of each item. Each Transformer layer is further dissected into two key sub-components. The Multi-Head Attention

Mechanism, a cornerstone of the Transformer architecture, operates by running multiple attention heads in parallel. Each head independently learns information from a distinct representation subspace,

with their outputs then aggregated. This parallelism allows the model to simultaneously focus on diverse and complex relationships among various positions in the input sequence, leading to a more comprehensive understanding of item dependencies crucial for handling long sequences and capturing non-local relationships. Following the attention

mechanism, the Position-wise Feed-Forward Neural Network (PFFN) performs position-wise transformations on the hidden representations. This involves independently processing the representation vector at each sequence position through a series of nonlinear transformations, refining feature representations, enhancing the expressiveness of each positional vector, and improving the model's understanding of abstract patterns in the user behavior sequence.

Through the coordinated operation of Multi-Head Attention and PFFN, the Transformer module efficiently converts sequential information into highly abstract and context-rich representations, establishing a strong foundation for the ultimate prediction task. Finally, these sophisticated sequence representations generated by the Transformer module are passed to the prediction layer, the system's concluding stage. This layer typically includes one or more linear layers and applies an activation function (e.g., Softmax) to map the abstract features into a probability distribution over all possible items. Based on these probabilities, the system recommends the next item the user is most likely to interact with, thereby completing the recommendation process. In summary, this model framework, through its hierarchical processing and modular design, effectively transforms user interaction history into accurate predictions of future behavior, showcasing the powerful potential of Transformer-based architectures in the domain of sequential recommendation systems.

4. Methodology

Geva et al. [26] propose that each layer of the FFN is composed of multiple key-value pairs. By analogy with their results, we speculate that user behavior information is stored in specific neurons in the recommendation system.

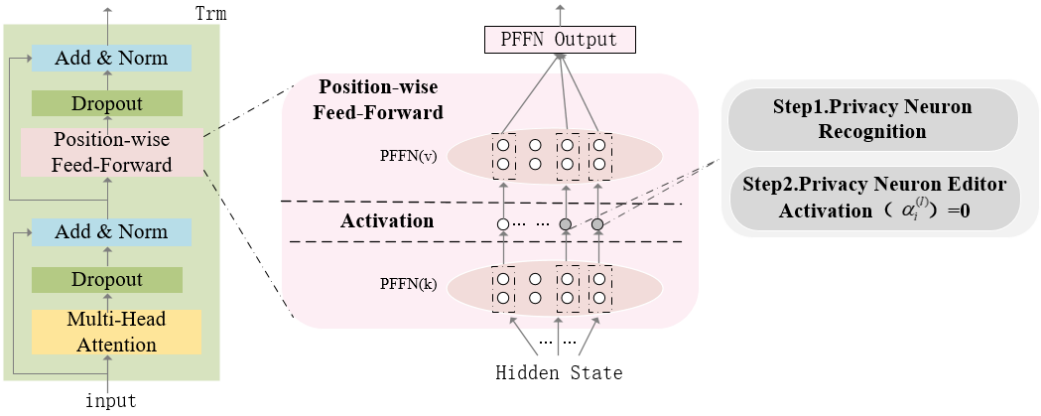

Figure 2. Trm layer structures

As shown in Figure 2, the first linear layer of the PFFN is analogous to the key, and the second linear layer is analogous to the value. User behavior information is stored in the middle neurons of the PFFN layer. Machine unlearning is achieved by editing neurons based on gradient integration. Our machine unlearning scheme consists of two steps: first, identify and select the proprietary neurons that contribute significantly to the final recommendation results as privacy neurons; second, edit the privacy neurons to erase their impact on the final recommendation.

4.1. Neuron Identification

The sequential recommendation aims to predict the project interacted by user u at time step $n_u + 1$ based on their historical interaction sequence N_u. It can be formulated as modeling the probability of all possible projects for user u at time step $n_u + 1$:

$$\mathrm{Attr}\left(\alpha_i^{(l)}\right) = \bar{\alpha}_i^{(l)} \int_{\chi=0}^{1} \frac{\partial P_{S_u}\left(\overset{-(l)}{\chi\alpha_i}\right)}{\partial \alpha_i^{(l)}} d\chi \tag{5}$$

Where $\overset{-(l)}{\alpha_i}$ represents the value calculated by the pre-trained model for the neuron; $P_{S_u}\left(\overset{-(l)}{\chi\alpha_i}\right)$ represents the probability of the recommendation system predicting the correct next project when the value of neuron $\alpha_i^{(l)}$ is $\overset{-(l)}{\chi\alpha_i}$; $\frac{\partial P_{S_u}\left(\overset{-(l)}{\chi\alpha_i}\right)}{\partial \alpha_i^{(l)}}$ calculates the gradient of the model output with respect to $\alpha_i^{(l)}$. Intuitively, as χ varies from 0 to 1, by integrating the gradient, $\mathrm{Attr}\left(\alpha^{(l)}\right)$ accumulates the output probability changes caused by $\alpha_i^{(l)}$ changes. If a neuron has a significant impact on the prediction of a specific category of projects, the gradient will significantly increase, resulting in a large integration value. Therefore, the attribution value can measure the degree of contribution of neuron $\alpha_i^{(l)}$ s to the specific category prediction. Algorithm 1 is the complete process of identifying neurons.

To identify the key neural units responsible for specific recommendation outcomes, we propose an attribution-based neuronal identification algorithm. The objective is to extract a subset of neurons that significantly influence the model's decision regarding a given target category. This process facilitates targeted analysis, interpretability, or downstream tasks such as selective forgetting. The procedure is as follows:

Sequence Recommendation Function Definition

The algorithm begins by defining the Sequence Recommendation function, which is designed to model and compute the probability that a user will interact with a particular item at the next time step $n_u + 1$ given their historical interaction sequence N_u. The goal is to estimate the likelihood of the next interaction $v_{n_{u+1}}^{u}$ based on prior behaviors.

Gradient-Based Attribution Computation

For each intermediate neuron $\overset{-(l)}{\alpha_i}$, the next step involves quantifying its cumulative influence on the model's output probability. This is achieved through Integrated Gradients, a widely-used attribution technique that accumulates the gradient of the output probability $P_{S_u}\left(\overset{-(l)}{\chi\alpha_i}\right)$ with respect to the scaled neuron value $\overset{-(l)}{\chi\alpha_i}$.

Initialization of Key Neuron Set

An empty set S is initialized to store the subset of neurons identified as significantly contributing to the prediction of the target category. This set will be populated through the following selection procedure.

Traversal of Relevant Sequences and Neuron Filtering

The algorithm proceeds by iterating through all historical interaction sequences N_u associated with a predefined target category. For each such sequence, the IntegrateGradient function is invoked to compute the attribution value of every neuron $\alpha_i^{(l)}$ in the specific context of that sequence. This step ensures that the attribution reflects both the model structure and the contextual behavioral patterns of users.

Neuron Selection Criterion

All intermediate neurons $\alpha_i^{(l)}$ are evaluated based on the following selection rule:

Condition Check: If the computed attribution value of a neuron exceeds a predefined threshold θ, it is deemed to have a significant impact on predicting the target category.

Inclusion in the Set: Neurons satisfying this condition are added to the set S.

Final Output

After processing all relevant sequences and applying the filtering criteria, the algorithm returns the final set S, which contains the neurons identified as most influential in the target recommendation task.

This attribution-based selection mechanism enables fine-grained interpretability and supports further operations such as concept-aware forgetting or attention modulation within recommendation systems.

Algorithm 1: Neuronal identification

Input:

- The sequence of historical interactions of user u: N_u
- Intermediate neural network element: $\alpha_i^{(l)}$
- Constant assigned to $\alpha_i^{(l)}$: $\left(\beta_i^{(l)}\right)$
- Threshold for attributions: θ

Output:

- Set of selected neurons: S

1. function SequenceRecommendation (N_u):
2. $P_{su}\left(\beta_i^{(l)}\right) = p\left(V_{t_u+1}^u = v \mid N_u, \alpha_i^{(l)} = \beta_i^{(l)}\right)$
3. ComputeGradient $\left(Atr\left(\alpha_i^{(l)}\right)\right) = \bar{\alpha}_i^{(l)} \int_{\chi=0}^{1} \frac{\partial P_{su}\left(\chi\alpha_i^{-(l)}\right)}{\partial \alpha_i^{(l)}} d\chi$
4. $S = \{\}$
5. for each sequence N_u related to the target category:
6. attrib = IntegrateGradient$\left(Atr\left(\alpha_i^{(l)}\right), N_i\right)$
7. for each neuron $\alpha_i^{(l)}$:
8. if $attrib\left(x_i^{(l)}\right) > \theta$
9. $S.add\left(\alpha_i^{(l)}\right)$
10. return S

4.2. Privacy Neuron Filtering and Editing

To precisely locate the neurons and reduce the impact of editing operations on other categories of user behavior information, further neuron screening is performed. For the target category, we assume that different sequences at time step $n_u + 1$ share a set of privacy neurons because they predict the same category. At the same time, it is assumed that as long as the sequences have diversity, they will not share redundant neurons. Therefore, in the case of multiple different sequences being given, the privacy neuron set can be screened by only retaining the neurons shared by the majority of sequences. In Algorithm 2, we provide a complete description of the process for screening edited neurons, with the specific steps for screening neurons as follows:

For the target category, select multiple sequences.

For the multiple sequences corresponding to the target category, calculate the attribution values of the neurons through $Attr\left(\alpha^{(l)}\right)$ and filter out the neurons with attribution values greater than the preset threshold θ to form a preliminary set of neurons.

Set the privacy threshold P%, and consider the preliminary screened neuron set together. Retain the neurons shared by more than P% sequences as privacy neurons.

When a user submits an unlearning request and needs to forget the behavior information of a specific category, set the parameters (activation values) of the screened neurons to zero in the second linear layer of the PFFN.

Algorithm 2: Privacy Neuron Screening Editor

Input:

- The sequence of historical interactions of user u: N_u
- Intermediate neural network element: $\alpha_i^{(l)}$
- Percentage threshold for shared neurons: P

Output:

- Set of privacy neurons: S'

1. $S' = \{\}$
2. for each neuron $\alpha_i^{(l)}$ inS:
3. shared_percentage = CalculateSharedPercentage(N_u, $\alpha_i^{(l)}$)
4. if shared_percentage >P%
5. S' .add($\alpha_i^{(l)}$)
6. if $\alpha_i^{(l)}$ in S':
7. SetActivationValue($\alpha_i^{(l)}$, 0)
8. Activation($\alpha_i^{(l)}$)=0
9. return S'

5. Experiment

5.1. Experimental Setup

Experimental results on two publicly available real-world datasets, Movielens-1m and Steam, are presented. These datasets have been widely used for evaluating the performance of recommendation models. Users are grouped according to their interaction records, and the interaction records of users are arranged in ascending order of timestamps to form the user interaction sequence. Then, users and projects with fewer than 5 interactions are filtered out. Table 1 shows the statistics of each dataset after preprocessing.

Table 1. Dataset statistics

Dataset	#User	#Item	#Interaction	Sparsity
MovieLens-1M	6 040	3 416	163.5	95.16
Steam	281 428	13 044	12.4	99.9

For each category in the user's historical behavior sequence, the attribution threshold is set as 0.3 times the maximum attribution value. For each category, the privacy threshold is initialized to 50 and increased or decreased by 5 iteratively until the average number of knowledge neurons is within [4, 8]. Eighty percent of user interaction sequences are selected for training, and the remaining 20% are used for testing. All experiments are conducted on an Intel(R) Core (TM) i7-7700HQ CPU @ 2.80GHz, a GTX 1050Ti GPU, and 16GB RAM.

5.2. Experimental Results

5.2.1. Recommendation System Performance

First, we evaluate the impact of our approach on the performance of recommendation system models. We conducted experiments on two publicly available real-world

datasets, Movielens-1m and Steam, which have been widely used for evaluating the performance of recommendation models. We use two widely used evaluation metrics for recommendation systems: Hit Ratio (HR) and Normalized Discounted Cumulative Gain (NDCG), which measure the ranking accuracy of recommendation lists and the goodness of search result lists, respectively. We evaluate these metrics with K = 1, 5, and 10. Higher values for all these metrics indicate better performance.

We compare the following recommendation models:

Original Model: The recommendation model without deleting any data.

Retrained Model: The recommendation model is retrained after deleting data.

GRU4Rec [28]: A recommendation model that uses gated recurrent units to model user sequences for session-based sequential recommendations.

Caser [29]: A recommendation model that adopts convolutional neural networks to model higher-order Markov chains for sequential recommendations.

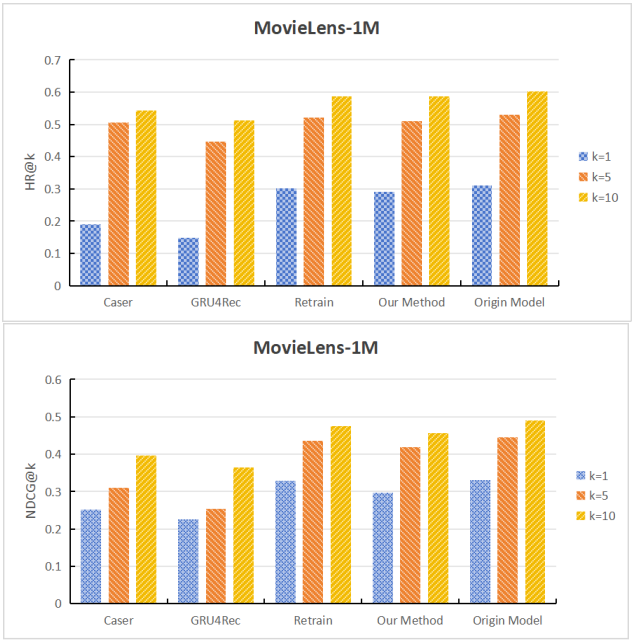

Figure 3. Performance Comparison of Models in Movielens-1m Dataset

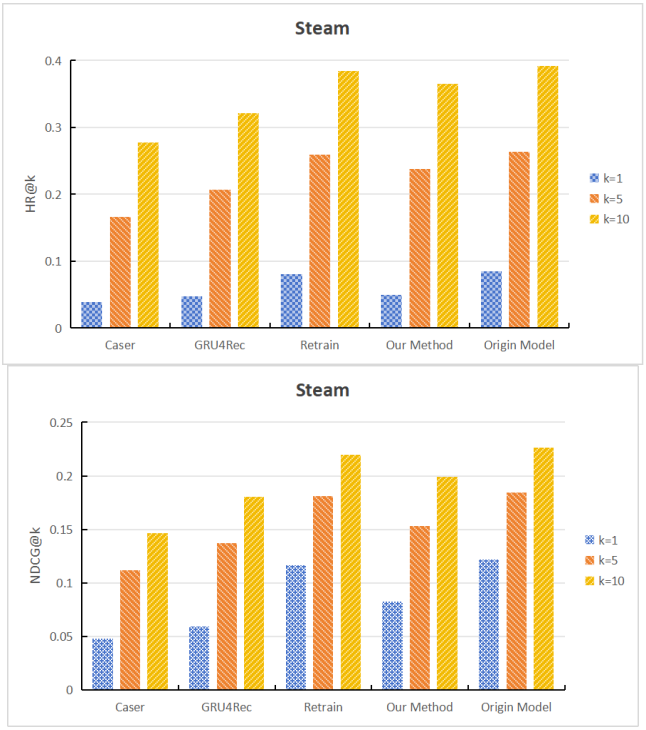

Figure 4. Performance comparison of models in the Steam dataset

As shown in the figure 3 and figure 4, the experimental results, evidenced by both Hit Rate (HR@k) and Normalized Discounted Cumulative Gain (NDCG@k) metrics, provide a comprehensive view of the performance of various sequential recommendation models and their variants.

The Movielens-1m dataset, the original model, the retrained model, and the forgotten model obtained using our approach all outperform Caser and GRU4Rec in terms of NDCG@k and HR@k. The original model's NDCG@10 and HR@10 are 0.4903 and 0.6018, respectively. The performance of the retrained model is lower than the original model due to the change in the dataset caused by deleting data after retraining. Our approach uses gradient attribution to forget data from the model, resulting in a slightly lower performance due to editing neuron parameters. However, the HR@k of the forgotten model still reaches 94.03% to 96.49% of the original model, and the NDCG@k reaches 90.15% to 94.25% of the original model. From the perspective of HR@k, a consistent trend of performance improvement across all models is observed with increasing values of k. Notably, the Origin Model (the original model without any unlearning operations) consistently achieves the best performance across all tested K values. This unequivocally demonstrates the substantial accuracy advantage of the full, unpruned model. The performance of the Retrain method (model retrained from scratch) and Our Method (the proposed approach) are remarkably close, exhibiting particular stability at k=10. This indicates that Our Method effectively maintains model accuracy, closely approaching that of a complete retraining. In contrast, baseline models such as Caser and GRU4Rec show suboptimal performance, especially at HR@1 and HR@5. GRU4Rec, in particular, highlights limitations in its ability to model local patterns effectively. The observed trends for NDCG@k largely mirror those of HR@k. The Origin Model again performs optimally, underscoring its superiority in the ranking quality of recommendation results. Our Method and the Retrain model also maintain high NDCG@k values, with only marginal differences from the Origin Model at k=5 and k=10. This validates the

effectiveness of Our Method in optimizing the ranking quality of recommendations. Conversely, Caser and GRU4Rec exhibit poorer ranking performance at smaller k values, with this limitation being most pronounced at k=1.

On the Steam dataset, the performance of NDCG@k and HR@k is shown in the figure 4. Overall, the HR@k scores on the Steam dataset are slightly lower compared to those on the MovieLens dataset, which may be attributed to the increased sparsity or the presence of more long-tail items in Steam. Among all evaluated models, the Origin Model consistently achieves the best performance, reaching an HR@10 of 0.4. Our Method closely follows, with performance at k=10 nearly matching that of the Retrain model. This indicates that the proposed approach successfully enables effective forgetting while preserving recommendation accuracy. The Retrain model remains a strong baseline, serving as the only alternative that approaches the performance of the original model. In contrast, Caser and GRU4Rec yield significantly lower HR@k scores across all values of k, further demonstrating their limitations in handling complex sequential recommendation tasks. The forgotten model obtained using our approach is slightly lower than the original model and the retrained model in both metrics, but this decrease is within an acceptable range, ensuring the performance of the recommendation system model while erasing data.

5.2.2. Unlearning Efficiency

In this section, we compare our approach with the retraining method in terms of unlearning rate and running time. As shown in the table, we randomly select three classes, Action, Drama, and Comedy, for unlearning in the Movielens-1m dataset.

Table 2. Running time

Method	Removal category	Running time (minute [m]).	Speedup
Retrain	Action	10258	0
	Drama	10401	0
	Comedy	10372	0
Our	Action	987	×10.4
	Drama	943	×11.0
	Comedy	1025	×10.1

In terms of running time, the retraining method requires starting from scratch each time, leading to significant time consumption. As shown in Table 2, our approach does not need to retrain the model on a large amount of raw data but only needs to edit the privacy neurons. Therefore, compared to the retrained model, our approach reduces a significant amount of training time. Our approach achieves an acceleration ratio of ×10.4, ×11.0, and ×10.1, respectively, when unlearning the Action, Drama, and Comedy classes.

Table 3. Removal rate

Method	Removal category	Removal rate
Retrain	Action	100%
	Drama	100%
	Comedy	100%
Our	Action	74.82%
	Drama	70.53%
	Comedy	72.91%

Table 3 compares our approach and the retraining method in terms of unlearning rate. Since the retraining method deletes the target class data from the raw data and starts training the model from scratch, it can forget all the data in the target class. However, the unlearning efficiency is extremely low, especially when the model is complex and the original dataset is large. Our approach does not require retraining the model and can achieve an unlearning rate of 70.53% to 74.82% on the three target classes, deleting most of the data in the target classes. It can balance the unlearning rate and running time.

6. Conclusion

To address user privacy leakage in sequential recommendation systems, this paper proposes an efficient unlearning scheme based on gradient attribution for recommendation models. First, refined neurons are identified through attribution, then filtered and edited to erase users' specific category behavior information from the trained model. Compared to training a model from scratch without the target category data, this approach demonstrates superior unlearning speed while maintaining the recommendation system's performance.

References

[1] Fuyu Lv, Taiwei Jin, Changlong Yu, Fei Sun, Quan Lin, Keping Yang, and Wilfred Ng. 2019. SDM: Sequential Deep Matching Model for Online Large-Scale Recommender System. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management (Beijing, China). Association for Computing Machinery, New York, NY, USA, 2635–2643

[2] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah.2016. Wide & Deep Learning for Recommender Systems. In Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (Boston, MA, USA).Association for Computing Machinery, New York, NY, USA, 7–10.

[3] Chong Chen, Min Zhang, Yiqun Liu, and Shaoping Ma. 2019. Social Attentional Memory Network: Modeling Aspect-and Friend-level Differences in Recommendation. In Proceedings of WSDM.

[4] Chong Chen, Min Zhang, Chenyang Wang, Weizhi Ma, Minming Li, Yiqun Liu, and Shaoping Ma. 2019. An Efficient Adaptive Transfer Neural Network for Social-aware Recommendation. In Proceedings of SIGIR. 225–234.

[5] Minxing Zhang, Zhaochun Ren, Zihan Wang, Pengjie Ren, Zhunmin Chen, Pengfei Hu, and Yang Zhang. 2021. Membership Inference Attacks Against Recommender Systems. , 16 pages.

[6] Santiago Zanella-Béguelin, Lukas Wutschitz, Shruti Tople, Victor Rühle, Andrew Paverd, Olga Ohrimenko, Boris Köpf, and Marc Brockschmidt. 2020. Analyzing Information Leakage of Updates to Natural Language Models. Association for Computing Machinery, New York, NY, USA, 363–375.

[7] Chenyang Wang, Weizhi Ma, Min Zhang, Chong Chen, Yiqun Liu, and Shaoping Ma. 2020. Toward Dynamic User Intention: Temporal Evolutionary Effects of Item Relations in Sequential Recommendation. ACM Transactions on Information Systems (TOIS) 39, 2 (2020), 1–33

[8] Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. 2021. Machine unlearning.

In 2021 IEEE Symposium on Security and Privacy (SP). IEEE, 141–159.

[9] Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. 2021. Graph Unlearning. arXiv preprint arXiv: 2103.14991 (2021).

[10] Wei Yuan, Hongzhi Yin, Fangzhao Wu, shijie zhang, Tieke He, Hao Wang. Federated Unlearning for On-Device Recommendation [C]. In: Proceedings of the 16th ACM International Conference on Web Search and Data Mining (WSDM 2023), 2023.

[11] Guy Shani, David Heckerman, and Ronen I Brafman. 2005. An MDP-based recommender system. JMLR 6, Sep (2005), 1265–1295.

[12] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized Markov chains for next-basket recommendation. In Proceedings of WWW'10. ACM, Raleigh, North Carolina, USA, 811–820.

[13] Pengfei Wang, Jiafeng Guo, Yanyan Lan, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2015. Learning Hierarchical Representation Model for NextBasket Recommendation. In Proceedings of ACM SIGIR'15. ACM, Santiago, Chile, 403–412.

[14] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In Proceedings of NIPS'14 (December 08 - 13). MIT Press, Montreal, Canada, 3104–3112.

[15] Wang-Cheng Kang and Julian McAuley. [n. d.]. Self-Attentive Sequential Recommendation. In Proceedings of ICDM. 197–206.

[16] Sun F, Liu J, Wu J, et al. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer [C]//Proceedings of the 28th ACM International Conference on Information and Knowledge Management. 2019: 1441-1450.

[17] I. Islek and S. G. Oguducu, ''A hybrid recommendation system based on bidirectional encoder representations,'' in Proc. Joint Eur . Conf. Mach.Learn. Knowl. Discovery Databases. Cham, Switzerland: Springer, 2020, pp. 225–236.

[18] B. Juarto and A. S. Girsang, ''Neural collaborative with sentence BERT for news recommender system,'' Int. J. Informat. Vis., vol. 5, no. 4, pp. 448–455, 2021.

[19] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, and P. Jiang, ''BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer,'' in Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.New York, NY, USA: Association for Computing Machinery, Nov. 2019, pp. 1441–1450.

[20] Mukund Sundararajan, Ankur Taly, and Qiqi Y an. 2017. Axiomatic attribution for deep networks. In Proceedings of the 34th International Conference on Machine Learning, ICML 2017, volume 70 of Proceedings of Machine Learning Research, pages 3319–3328. PMLR.

[21] Y aru Hao, Li Dong, Furu Wei, and Ke Xu. 2021. Self-attention attribution: Interpreting information interactions inside transformer. In The Thirty-Fifth AAAI Conference on Artificial Intelligence. AAAI Press.

[22] Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In 2015 IEEE Symposium onSecurity and Privacy, pages 463–480. IEEE, 2015. 1, 8

[23] Gaoyang Liu, Y ang Yang, Xiaoqiang Ma, Chen Wang, and Jiangchuan Liu. Federated unlearning. arXiv preprint arXiv: 2012.13891, 2020. 1, 9

[24] Chen C, Sun F, Zhang M et al. Recommendation Unlearning [J]. 2022. DOI:10.48550/arXiv.2201.06820.

[25] Li Y, Zheng X, Chen C et al. Making Recommender Systems Forget: Learning and Unlearning for Erasable Recommendation [J]. 2022. DOI:10.48550/arXiv.2203.11491.

[26] Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2020. Transformer feed-forward layers are key-value memories. CoRR, abs/2012.14913.

[27] Dai, Damai et al. "Knowledge neurons in pretrained transformers." ACL 2022.

[28] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk.2016. Session-based Recommendations with Recurrent Neural Networks. In Proceedings of ICLR.

[29] Jiaxi Tang and Ke Wang. 2018. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. In Proceedings of WSDM. 565–573.