electronics

MDPI

Article

Local-Aware Hierarchical Attention for Sequential Recommendation

Jiahao Hu *, Qinxiao Liu and Fen Zhao

School of Artificial Intelligence, Chongqing University of Technology, Chongqing 401135, China; liuqinxiao@cqut.edu.cn (Q.L.); zhaof@cqut.edu.cn (F.Z.)
* Correspondence: jhaohu@stu.cqut.edu.cn

Abstract: Modeling the dynamic preferences of users is a challenging and essential task in a recommendation system. Taking inspiration from the successful use of self-attention mechanisms in tasks within natural language processing, several approaches have initially explored integrating self-attention into sequential recommendation, demonstrating promising results. However, existing methods have overlooked the intrinsic structure of sequences, failed to simultaneously consider the local fluctuation and global stability of users' interests, and lacked user information. To address these limitations, we propose LHASRec (Local-Aware Hierarchical Attention for Sequential Recommendation), a model that divides a user's historical interaction sequences into multiple sessions based on a certain time interval and computes the weight values for each session. Subsequently, the calculated weight values are combined with the user's historical interaction sequences to obtain a weighted user interaction sequence. This approach can effectively reflect the local fluctuation of the user's interest, capture the user's particular preference, and at the same time, consider the user's general preference to achieve global stability. Additionally, we employ Stochastic Shared Embeddings (SSE) as a regularization technique to mitigate the overfitting issue resulting from the incorporation of user information. We conduct extensive experiments, showing that our method outperforms other competitive baselines on sparse and dense datasets and different evaluation metrics.

Keywords: sequential recommendation; local fluctuation; global stability; Stochastic Shared Embeddings

check for updates

Citation: Hu, J.; Liu, Q.; Zhao, F. Local-Aware Hierarchical Attention for Sequential Recommendation. *Electronics* **2023**, *12*, 3742. https://doi.org/10.3390/electronics12183742

Academic Editors: Domenico Ursino, Chao Zhang, Wentao Li, Huiyan Zhang and Tao Zhan

Received: 7 August 2023
Revised: 25 August 2023
Accepted: 26 August 2023
Published: 5 September 2023

1. Introduction

In recent years, personalized recommendation tasks have become increasingly important. As the volume of information grows on the Internet, providing accurate recommendations based on changes in the user's interests has become a challenge. To address this challenge, researchers have regarded the user's historical interaction behaviors as ordered sequences, aiming to capture the dynamic changes in the user's interests from these sequences and predict the next interactive item they may be interested in. This prediction is crucial for providing personalized recommendations, which helps the platform better meet user needs and improve user experience.

To model the user's dynamic interests, researchers have explored various modeling strategies and algorithms [1–3] for the sequential features of the user's historical interaction behavior. In the early stages, Markov chain models [1,4] were commonly used to capture the transition of the user's preferences from their interaction history with items. FPMC [5] integrated the idea of matrix factorization with Markov chains, storing information about user transition matrices in a three-dimensional matrix. With the popularity of deep learning, Recurrent Neural Networks (RNNs) [6] and Convolutional Neural Networks (CNNs) [2] have gradually been applied to the sequential recommendation. Compared to Markov chains, RNNs can more effectively capture temporal relationships in the user's behavior sequences due to their inherent structural characteristics. CNNs can capture the user's interests from sequences with complex relationships. Additionally, some methods based on self-attention mechanisms [3,7,8] introduced weight adaptive adjustment mechanisms to model the importance of different elements in the sequence dynamically. These methods can more accurately capture changes and associations in the user's interests, thereby improving the accuracy and personalization of recommendation results.

In sequential recommendation, the user's preferences are typically influenced by a combination of long-term and short-term factors. Long-term preferences reflect the user's general interests, which are relatively stable and less prone to change within the sequence. On the contrary, short-term preferences reflect the user's transient "special" interests that may deviate from their general interests and exhibit the relative fluctuation within the sequence. For example, a user may prefer comedy movies as his/her favorite genre. However, due to the influence of his/her friends, he/she may develop a temporary fondness for art films for a certain period. However, traditional sequential recommendation models often treat the user's historical interaction sequence as a homogeneous entity, lacking simultaneous consideration of the local fluctuation and global stability of the user's interests. This can potentially impact the model's ability to learn the user's preferences and subsequently affect the effectiveness of recommendations.

Regarding the above issues, in this paper, we propose a Local-Aware Hierarchical Attention recommendation system that combines the local fluctuation and global stability of the user's interests. This model enhances the model's ability to model the user's behavior preferences, enabling the model to more accurately reflect the user's personalized interests and preferences. Consequently, it can more comprehensively understand the user's behavior patterns and provides users with more targeted recommendation results. Furthermore, we also consider the user information from SSE-PT [7] and employ the Stochastic Shared Embedding regularization technique to handle user and item embeddings in the input and prediction parts, alleviating overfitting issues. In summary, our main contributions are as follows:

- We comprehensively consider the local fluctuation and global stability of the user's interests to better capture users' long-term and short-term preferences.
- We employ the Stochastic Shared Embedding regularization technique to handle user embeddings and item embeddings in the input and prediction parts to alleviate the overfitting problem.
- We conducted extensive experiments on the MovieLens, Steam, and Beauty datasets, and the experimental results demonstrate that our model outperforms other competitive baselines.

2. Related Work

The user's behavior is a time-ordered behavior sequence, and their interests also dynamically change over time. Therefore, extracting temporal information from sequential data can provide valuable information. Early sequential recommendation models utilized Markov chains (MCs) [9] to capture the correlations within the sequential data. Shani et al. used the Markov chain [1] to mine the correlation between users' short-term behaviors, thus achieving a good recommendation effect. Rendle et al. combined the idea of Matrix Factorization (MF) with Markov chains [5,10] by storing user transition matrices in a three-dimensional matrix and explored the temporal information in the user's short-term behavior sequences by predicting the user's interests in other items. However, due to the scalability issue of Markov chains, the time and space complexity of the models significantly increase when dealing with longer sequences, leading to suboptimal recommendation performance.

Compared to Markov chains, Recurrent Neural Networks (RNNs) [11,12], benefiting from their distinctive structure, are more effective in handling sequential data. B. Hidasi et al. first applied RNNs to sequential recommendation [6] and proposed the session-based sequential recommendation model. It divided the user's behavior into multiple sessions based on a certain time interval, modeled each session's behavior using an RNN, and predicted the next item the user interacted with. To further improve sequential recommendation performance, Hidasi et al. introduced the parallel RNN session-based recommendation model [13], which used RNN to find the dependencies between items in the session and parallel RNN to model other attribute characteristics of items in the session, which improved the effect of sequential recommendation. Zhang et al. proposed an RNN-based sequential search click prediction model [14] that not only modeled the user's click events but also incorporated features of users and items and the information of the user's dwell time after clicking an item, resulting in the improved predictive capability of the model.

However, RNN models assume that any adjacent interactions in a sequence are mutually dependent, while in reality, there exist intricate and complex relationships among the user's consecutive actions. Linearly modeling the user's historical behavior makes it difficult to capture their true interests within a sequence with complex relationships. As a result, researchers have begun exploring the domains of CNN and self-attention mechanisms. Tang et al. proposed Caser [2], a sequential recommendation model based on convolutional embeddings, which applies CNN concepts to sequential recommendation. Caser employed multiple convolutional filters with different weights to extract various sequential pattern information from the sequence, thereby improving the accuracy of personalized recommendations. Similarly, Yuan et al. extended the Caser model with a future-oriented recommendation framework called NextItNet [15], utilizing dilated convolutions to capture more complex dependencies in the user's behavior sequence. Kang et al. introduced SASRec [3], a sequential recommendation model based on self-attention mechanisms, to address the dependency issue of RNNs. Additionally, Li et al. proposed MIND [16], which reflected a user's multidimensional interests by using multiple vectors to represent each user and predicted the match between the candidate items and the user's interests in each dimension.

Nevertheless, some existing models ignore the internal structure of the sequence, do not consider the local fluctuation and global stability of the user's interest simultaneously, and lack user information. In our study, we first divide the user's historical interaction sequence into multiple sessions based on a certain time interval. By comprehensively considering the weak correlation between sessions and the strong correlation among items within each session, we accurately reflect the local fluctuation of the user's interests. Next, we combine the calculated weights of each session with the user's historical interaction sequence to achieve the influence of the local fluctuation on the global stability of the user's interests. Additionally, we introduce user information into the model and employ the Stochastic Shared Embedding regularization technique to mitigate the overfitting problem that may arise from incorporating user information.

3. Method

We propose a hierarchical attention-based sequential recommendation model, LHASRec. The model consists of an embedding layer, a local-aware layer, a global attention layer, and a prediction layer. This section will describe how to construct this sequential recommendation model. The architecture of LHASRec is illustrated in Figure 1.

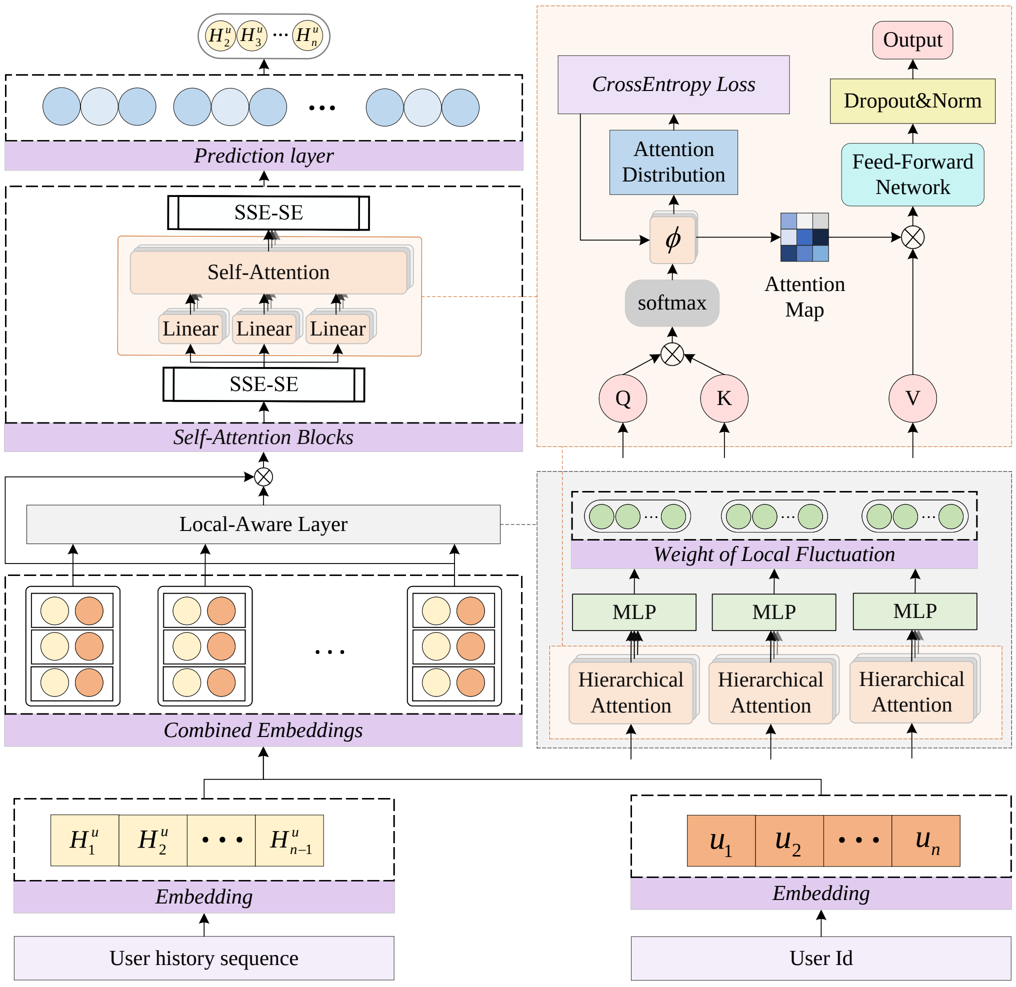

Figure 1. The overall framework of LHASRec. The model primarily consists of an embedding layer, a local-aware layer, a global attention layer, and a prediction layer, and the input and output of the global attention layer are handled using SSE regularization. (a) The target user's historical behavior sequence is combined with user information and divided into multiple sessions based on time intervals. (b) Each session is individually processed by the local-aware layer to generate local attention weights, which are then combined with the sequences containing item information and user information to serve as the input for the global attention layer. (c) The SSE regularization technique is applied to the input matrix. (d) The global attention layer captures the representation of the user's local and global preferences. (e) The output matrix is regularized using SSE. (f) Recommendations are made based on the target user's local and global preferences.

3.1. Sequential Recommendation Target

In the sequential recommendation task of this section, we define the user's historical behavior sequence as $H^u = (H_1^u, H_2^u, \cdots, H_{|H^u|}^u)$, where $u \in U, H_i^u \in I$. During the model training process, at time step t, the model predicts the next item the target user will likely interact with based on the preceding t items. In other words, we use the user's historical behavior sequence $(H_1^u, H_2^u, \cdots, H_{|H^u|-1}^u)$ and the user's information as inputs to the model, with the expected output denoted as $(H_2^u, H_3^u, \cdots, H_{|H^u|}^u)$. The symbols used in our study are summarized in Table 1.

Table 1. Notation.

Notation	Description
U, I	user and item set
H^u	historical interaction sequence for the user u
$t_d \in \mathbb{N}$	division time interval
$n \in \mathbb{N}$	maximum sequence length
$n_s \in \mathbb{N}$	length of each session
$k \in \mathbb{N}$	number of sessions
$b \in \mathbb{N}$	number of stacked temporal attention blocks
$d \in \mathbb{N}$	latent vector dimension of the model
$d_i, d_u \in \mathbb{N}$	latent dimension of item and user
$M^I \in \mathbb{R}^{\|I\| \times d_i}$	item embedding matrix
$M^U \in \mathbb{R}^{\|U\| \times d_u}$	user embedding matrix
$\hat{S}_1, \hat{S}_2, \cdots, \hat{S}_k \in \mathbb{R}^{n_s \times d}$	input embedding matrix of local-aware layer
$\hat{E} \in \mathbb{R}^{n \times d}$	input embedding matrix of global attention layer
$c_1, c_2, \cdots, c_k \in \mathbb{R}$	fluctuation coefficient of each session
$A \in \mathbb{R}^{n \times d}$	output of the self-attention layer
$F \in \mathbb{R}^{n \times d}$	output of the point-wise feed-forward network

3.2. Embedding Layer

For the historical behavior sequence of the target user, denoted as $(H_1^u, H_2^u, \cdots, H_{|H^u|-1}^u)$, we fix its length to a specific value $n \in \mathbb{N}$ to obtain the sequence, denoted as $(h_1, h_2, \cdots, h_n)$, where n represents the maximum sequence length. The rule for fixing the length sequence is as follows:

$$\begin{cases} padding & H_{|H^u|-1}^u < n \\ unchange & H_{|H^u|-1}^u = n \\ cutting & H_{|H^u|-1}^u > n \end{cases} \tag{1}$$

It is worth noting that when the length of the original sequence is smaller than n, we pad the left side of the sequence with zeros. When the length of the original sequence is greater than n, we only consider the most recent n interactions. We construct the item embedding matrix $M^I \in \mathbb{R}^{|I| \times d_i}$ and the user embedding matrix $M^U \in \mathbb{R}^{|U| \times d_u}$, where $d_i, d_u \in \mathbb{N}$ represent the latent embedding dimensions for items and users, respectively. From these two embedding matrices, we retrieve the user information embedding for the target user and the embeddings of each item in the user's input sequence. These embeddings are combined to obtain the input embedding $E \in \mathbb{R}^{n \times d}$, where $d = d_i + d_u$:

$$E = \begin{bmatrix} \left[M_{h_1}^I; M_u^U\right] \\ \left[M_{h_2}^I; M_u^U\right] \\ \cdots \\ \left[M_{h_n}^I; M_u^U\right] \end{bmatrix} \tag{2}$$

where $\left[M_{h_i}^I; M_u^U\right]$ is the concatenation of the embedding vectors for item h_i and user u. We believe that when the time intervals between several user interactions are close, it indicates that the user is selecting items of the same kind of interest, indicating a strong correlation between these items. On the other hand, when the time interval between two adjacent items is large, it may suggest a change in the user's interest, resulting in a change in the selected items' category and indicating a weak correlation between these two items. In the user's historical behavior, the local fluctuation is generated with the change of the user's interest. To capture these fluctuations, we introduce a time interval threshold, denoted as $t_d \in \mathbb{N}$, and examine the time intervals between every two adjacent items. If the interval exceeds t_d, we split the input sequence accordingly. Following this approach, we divide the input sequence E into k sessions, i.e., local interaction sequences. Within each session, the items exhibit strong correlations, while there are weak correlations between items from different sessions. As the divided sessions have different lengths, we apply the same fixed-length rule for each session $S_i \in \mathbb{R}^{n_s \times d}$ to adjust its length to a specific value, denoted as $n_s \in \mathbb{N}$:

$$S_1 = \begin{bmatrix} \left[M^I_{h_1^{S_1}}; M^U_u\right] \\ \left[M^I_{h_2^{S_1}}; M^U_u\right] \\ \cdots \\ \left[M^I_{h_{n_s}^{S_1}}; M^U_u\right] \end{bmatrix}, \cdots, S_k = \begin{bmatrix} \left[M^I_{h_1^{S_k}}; M^U_u\right] \\ \left[M^I_{h_2^{S_k}}; M^U_u\right] \\ \cdots \\ \left[M^I_{h_{n_s}^{S_k}}; M^U_u\right] \end{bmatrix} \tag{3}$$

where $M^I_{h_j^{S_i}} \in \mathbb{R}^d$ represents the embedding of item h_j in the i-th session. Since the self-attention mechanism is unaware of the positional relationship of items in the sequence, we introduced learnable position embeddings for each session:

$$\hat{S}_1 = \begin{bmatrix} \left[M^I_{h_1^{S_1}}; M^U_u\right] + p_1^{S_1} \\ \left[M^I_{h_2^{S_1}}; M^U_u\right] + p_2^{S_1} \\ \cdots \\ \left[M^I_{h_{n_s}^{S_1}}; M^U_u\right] + p_{n_s}^{S_1} \end{bmatrix}, \cdots, \hat{S}_k = \begin{bmatrix} \left[M^I_{h_1^{S_k}}; M^U_u\right] + p_1^{S_k} \\ \left[M^I_{h_2^{S_k}}; M^U_u\right] + p_2^{S_k} \\ \cdots \\ \left[M^I_{h_{n_s}^{S_k}}; M^U_u\right] + p_{n_s}^{S_k} \end{bmatrix} \tag{4}$$

where $p_j^{S_i} \in \mathbb{R}^d$ represents the embedding of the j-th position in the i-th session.

3.3. Local-Aware Layer

We employ hierarchical attention to capture the strong correlations among items within each session, where each layer utilizes the self-attention mechanism [17] defined as follows:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d}}\right)V \tag{5}$$

where Q represents the query matrix, and K and V denote the key and value matrices, respectively. $\sqrt{d}$ is a scaling factor used to mitigate the problem of large inner products when the dimension is high.

We feed each session of the user separately into different attention layers to avoid mutual interference between items with weak correlations, ultimately obtaining the fluctuation coefficient corresponding to each session, thereby capturing the local fluctuations of the user's interests. Specifically, for the i-th session, it is linearly projected into three matrices, which are then fed into the i-th attention layer:

$$A_i^S = HA(\hat{S}_i) = Attention(\hat{S}_i W^Q_{S_i}, \hat{S}_i W^K_{S_i}, \hat{S}_i W^V_{S_i}) \tag{6}$$

where $W^Q_{S_i}, W^K_{S_i}, W^V_{S_i} \in \mathbb{R}^{d \times d}$ represent the projection matrices of Q, K, and V, respectively, for the matrix S_i. Due to the strong correlations among items within each session, we consider that their sequential order can be ambiguous, allowing subsequent keys to be connected to the current query to fully capture the representation power of the self-attention mechanism.

After the self-attention layer, we employ two MLP layers to model the non-linear relationships among items within the session to obtain the fluctuation coefficient c_i corresponding to the i-th session:

$$L_i = A_i^S W_i^M + b_i^M \tag{7}$$

$$c_i = GELU\left((L_i)^T W_i^c + b_i^c\right) \tag{8}$$

where $W_i^M \in \mathbb{R}^{d\times 1}, W_i^c \in \mathbb{R}^{n_s\times 1}, b_i^M \in \mathbb{R}^{n_s\times 1}, b_i^c \in \mathbb{R}$ are learnable parameters. Instead of the ReLU function, we utilize the smoother GELU [18] function for activation.

3.4. Global Attention Layer

For each session, we utilize its fluctuation coefficient to reflect the local interest fluctuation of the user, where there is a strong correlation among items within the same session. However, no single session can fully represent the user's global preferences, and there may also exist connections between different local interests. Therefore, we combine the local fluctuation coefficients with the overall input sequence to achieve global stability. Specifically, for the input sequence E, we extract the fluctuation coefficients of each item according to the fluctuation coefficient corresponding to the session in which the item belongs, resulting in a local fluctuation sequence $C = (c^{h_1}, c^{h_2}, \cdots, c^{h_n})$, where c^{h_i} is the fluctuation coefficient corresponding to the session where item h_i belongs. We introduce the fluctuation coefficients as weights to combine the input sequence that incorporates item and user information, along with learnable position embeddings, yielding a new input embedding $\hat{E} \in \mathbb{R}^{n\times d}$:

$$\hat{E} = \begin{bmatrix} c^{h_1}\left[M^I_{h_1}; M^U_u\right] + P_1 \\ c^{h_2}\left[M^I_{h_2}; M^U_u\right] + P_2 \\ \cdots \\ c^{h_n}\left[M^I_{h_n}; M^U_u\right] + P_n \end{bmatrix} \tag{9}$$

where $P_i \in \mathbb{R}^d$ represents the embedding for the i-th position.

Attention layer: We perform linear projections on the input embedding $\hat{E}$ to obtain three matrices, then feed into the attention layer:

$$A = HA(\hat{E}) = Attention(\hat{E}W^Q, \hat{E}W^K, \hat{E}W^V) \tag{10}$$

where $W^Q, W^K, W^V \in \mathbb{R}^{d\times d}$ is the projection matrix. Similar to [3], we introduce a mask to prevent any connection between Q_i and K_j $(j > i)$ to prevent subsequent items from affecting the current item to be predicted.

Point-Wise Feed-Forward Network: To introduce non-linearity and consider interactions among different latent dimensions, similar to [3,8], we apply the same two-layer Point-Wise Feed-Forward Network (with shared weights) to A and use the ReLU activation function:

$$F = FFN(A) = ReLU(AW^{(1)} + b^{(1)})W^{(2)} + b^{(2)} \tag{11}$$

where $W^{(1)}, W^{(2)} \in \mathbb{R}^{d\times d}, b^{(1)}, b^{(2)} \in \mathbb{R}^d$ represents the learnable parameters. As the number of parameters in the network increases, several issues may arise, including overfitting, unstable training process (such as gradient vanishing), and longer training time. Similar to [3,8], we employ layer normalization, residual connections, and dropout regularization techniques after the attention layer and Point-Wise Feed-Forward Network to alleviate these issues:

$$f(x) = x + Dropout(f(LayerNorm(x))) \tag{12}$$

where $f(x)$ represents the self-attention layer or Point-Wise Feed-Forward Network. The definition of layer normalization is as follows:

$$LayerNorm(x) = \alpha \odot \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta \tag{13}$$

where x is a vector containing all the features of the samples, μ and σ denote the mean and variance, α is a learnable scale factor, and β is a bias term.

We merge one layer of the self-attention and one layer of the Feed-Forward Network into one attention module. To capture the user's preferences more accurately, we stack b attention modules to learn more complex item transformations, ultimately obtaining the representation of the user's preferences.

3.5. Prediction Layer

After b attention modules, the model obtains the global representation of the target user's preferences. Using this representation, at time step t, we predict the next item that the user may interact with:

$$r_{ti} = F_t\left[M^I_{h_i}; M^U_u\right] \tag{14}$$

where r_{ti} represents the score of item h_i given the previous t items, i.e., the possibility that the next item is h_i. $\left[M^I_{h_i}; M^U_u\right]$ represents the combined feature embedding that incorporates both item information and user information. At time step t, for each positive sample item $i = h_{t+1}$, we randomly sample a negative sample $m \notin H^u$. Due to the faster weight update rate of the binary cross-entropy loss function compared to the mean squared error loss function, we use binary cross-entropy as the loss function:

$$-\sum_{H^u \in H} \sum_{t \in [1,2,\cdots,n]} \left[log(\sigma(r_{ti})) + \sum_{m \notin H^u} log(1 - \sigma(r_{tm})) \right] \tag{15}$$

where $\sigma(\cdot)$ is the sigmoid function. Because ADAM demonstrates greater robustness in handling noise and outliers compared to the stochastic gradient descent algorithm (SGD), we optimize the model using the ADAM optimizer [19]. The top-K recommendations for the target user at time step t can be obtained by sorting the scores of all items, and the top K items in the sorted list are the recommended items.

4. Experiments

In this section, we will present our experimental setup and show the results of our experiments. The experiments conducted aim to answer the following research questions:

RQ1: Can our proposed method outperform the state-of-the-art baselines?

RQ2: Does the choice of different time interval values for sequence dividing affect the model's ability to capture the local fluctuation of the user's interests?

RQ3: How do parameters such as maximum sequence length and the number of attention blocks impact the model's performance?

4.1. Datasets

We evaluated LHASRec on four datasets. These datasets cover different domains, sizes, and sparsity levels, and all of them are publicly available:

Movielens: https://grouplens.org/datasets/movielens/ (accessed on 25 August 2023) This dataset is sourced from the GroupLens Research project at the University of Minnesota. It is a widely used benchmark dataset. We utilized the Movielens-1M version, which consists of 1 million ratings from 6040 users on 3900 movies.

Amazon: http://jmcauley.ucsd.edu/data/amazon/ (accessed on 25 August 2023) We utilized the users' purchase and rating dataset from the e-commerce platform Amazon, which was collected by McAuley et al. [20]. To enhance the usability of the dataset, the researchers divided it based on high-ranking categories on Amazon. Specifically, we selected the **"Beauty"** and **"Video Games"** categories for our study.

Steam: https://cseweb.ucsd.edu/~jmcauley/datasets.html#steam_data (accessed on 25 August 2023) It originates from the popular digital game distribution platform, Steam.

The dataset captures users' behaviors, such as game purchases, game ratings, and game social interactions on the Steam platform.

These four datasets all include timestamps of users' interactions. We followed the methods described in [3,7] to preprocess the data. Firstly, we sorted the user–item interactions in ascending order based on the timestamps. To ensure the validity of the data, we excluded those cold-start users and those with less than three user–item interactions. Similar to the approach in [3], we used the last item in the interaction sequence (i.e., the most recent item interacted with by the user) as the test set, the second-to-last item as the validation set, and the remaining items as the training set. Through these preprocessing steps, we reduced redundant information while preserving the data's original meaning, facilitating further research and algorithm evaluation in the recommendation system domain. Table 2 provides an overview of these datasets, highlighting their characteristics. Among them, Movielens-1M is the densest dataset, with fewer users and items. On the other hand, the Steam dataset is the sparsest, containing relatively fewer interactions.

Table 2. Dataset statistics.

Dataset	Users	Items	Avg. Sequence Length	Sparsity
MovieLens-1M	6040	3706	163.6	95.58%
Beauty	22,363	12,101	6.88	99.94%
Games	24,303	10,672	7.54	99.92%
Steam	144,051	11,153	3.49	99.97%

4.2. Compared Methods

We compared LHASRec with various methods, including the classic recommendation approach (BPR) and recommendation models based on different techniques. Among them, we considered methods based on first-order Markov chains (such as FMC, FPMC, TransRec), transformer-based methods (such as SASRec, SSE-PT, TiSASRec), convolutional neural network-based methods (Caser, TARN), fusion model-based methods (BAR), and multilayer perceptron-based methods (FMLP-Rec).

BPR [21]: Bayesian personalized ranking model (BPR) is a traditional recommendation method that employs matrix factorization for the recommendation.

FPMC [5]: Factorizing personalized Markov chains model (FPMC) amalgamates matrix factorization with the initial-order Markov chain technique, enabling the model to encompass both users' long-term preferences and the dynamic transitions of items.

TransRec [22]: Translation-based recommendation model (TransRec) represents a first-order sequential recommendation approach, where items undergo embedding within a transformational domain, while users are depicted as translation vectors that encapsulate shifts from the present item to the subsequent one.

SASRec [3]: Self-attentive sequential recommendation model (SASRec) is the first transformer-based model that extracts context from all past interactions like recurrent neural networks while making predictions based on a limited number of interactions, similar to Markov chains.

SSE-PT [7]: Sequential recommendation via personalized transformer model (SSE-PT) integrates the embedding vector of the user ID and employs a novel regularization approach.

TiSASRec [8]: Time interval aware self-attention for sequential recommendation model (TiSASRec) leverages the advantage of attention mechanisms to handle items at different ranges in different datasets adaptively and adjusts the weights based on different items, absolute positions, and time intervals.

TARN [23]: Neural time-aware recommendation network (TARN) simultaneously captures users' static and dynamic preferences by fusing a feature interaction network with a convolutional neural network.

BAR [24]: Behavior-aware recommendation model (BAR) integrates behavioral information into the representation module and employs the innovative module across diverse backbone models.

FMLP-Rec [25]: Filter-enhanced MLP model (FMLP-Rec) is a pure MLP architecture model that encodes user sequences using learnable filters.

4.3. Implementation Details

We implemented the LHASRec using PyTorch, with the same number of transformer encoding blocks as SASRec and SSE-PT (i.e., $b = 2$). To optimize the model, we chose ADAM as the optimizer with a learning rate of 0.001 and a momentum decay rate of $\beta_1 = 0.9, \beta_2 = 0.98$. The batch size was set to 128. For the Movielens-1M dataset, we set the dropout rate to 0.2, while for the other three datasets, it was set to 0.5. Regarding the maximum length of the sequences, we set it to 190 for the Movielens-1M dataset and 50 for the other three datasets. Additionally, to further enhance the effectiveness of personalized recommendations, we fine-tuned two parameters of the SSE to improve its performance.

4.4. Evaluation Metrics

To assess the effectiveness of all the models, we employed *HR@N* and *NDCG@N* as evaluation metrics [26], defined as follows:

$$HR@N = \frac{1}{M}\sum_{i=1}^{M} hits(i) \tag{16}$$

$$NDCG@N = \frac{1}{M}\sum_{i=1}^{M} \frac{1}{log_2(p_i + 1)} \tag{17}$$

where M is the number of users, $hits(i)$ indicates whether the item interacted with by the i-th user is present in the recommendation list of length N, and p_i represents the position of the item interacted with by the i-th user in the recommendation list. In our experiments, we set the length N of the recommendation list to 10. To evaluate the performance of the recommendation algorithms, we employed *HR*@10 and *NDCG*@10 as the two metrics. Specifically, we appended 100 negative samples [27] randomly after each user's actual items and calculated the metric values based on the rankings of these 101 items. It is worth noting that higher values of *HR*@10 and *NDCG*@10 indicate better model performance.

4.5. Recommendation Performance (RQ1)

Table 3 presents the recommendation performance of various methods on the four datasets **(RQ1)**. For the dense dataset, TiSASRec outperforms other baseline methods. Its advantage lies in the effective utilization of attention mechanisms, and it can dynamically adjust the weights according to different items, absolute positions, and time intervals to adapt to variations in dataset ranges. For the sparse dataset, FMLP-Rec demonstrates superior recommendation performance compared to other baseline methods. Replacing the complex Transformer architecture with MLP layers in the frequency domain effectively addresses the overfitting issue caused by insufficient available information in sparse datasets. Neural network-based methods (Caser) excel at capturing long-term sequential patterns, thus performing well on dense datasets. In contrast, methods based on Markov chains (such as FMC, FPMC, and TransRec) focus more on item transitions, resulting in better performance on sparse datasets. Furthermore, the TARN approach, which concurrently captures both users' dynamic and static preferences, achieves superior performance compared to the SASRec technique, which focuses solely on a single type of preference, across all the datasets. Moreover, the BAR technique demonstrates superior performance compared to its underlying model, SASRec, across all datasets, highlighting the effectiveness of segregating the user's historical interaction sequence into item sequence and behavior sequence as a productive modeling strategy.

Table 3. Recommended performance. We have bolded the best-recommended method in each row and underlined the second-best-performing approach in each row.

Methods	Beauty		Games		ML-1M		Steam	
	Hit@10	NDCG@10	Hit@10	NDCG@10	Hit@10	NDCG@10	Hit@10	NDCG@10
BPR	0.3775	0.2183	0.4853	0.2875	0.5781	0.3287	0.7061	0.4436
FMC	0.3771	0.2477	0.6358	0.4456	0.6983	0.4676	0.7731	0.5193
FPMC	0.4310	0.2891	0.6082	0.4680	0.7599	0.5176	0.7710	0.5011
TransRec	0.4607	0.3020	0.6838	0.4557	0.6413	0.3969	0.7624	0.4852
Caser	0.4264	0.2547	0.5282	0.3214	0.7886	0.5538	0.7874	0.5381
SASRec	0.4663	0.3080	0.6843	0.4602	0.8285	0.5982	0.7867	0.5108
SSE-PT	0.4963	0.3159	0.6955	0.4677	0.8346	0.6163	0.7885	0.5369
TiSASRec	0.4981	0.3329	0.7080	0.467	0.8359	0.6156	0.8053	0.5523
TARN	0.4979	0.3324	0.6996	0.4698	0.8325	0.6139	0.7985	0.5476
BAR	0.4995	0.3336	0.7073	0.4704	0.8351	0.6192	0.8039	0.5492
FMLP-Rec	0.5029	0.3351	0.7091	0.4773	0.8291	0.5333	0.8031	0.5470
LHASRec	**0.5150**	**0.3402**	**0.7359**	**0.5072**	**0.8396**	**0.6197**	**0.8218**	**0.5611**

LHASRec outperforms the leading benchmark techniques in recommendation performance across all the datasets. This achievement can be attributed to two key factors. Firstly, in the case of sparse data, the introduction of user information embedding enhances the correlation between users and items, thereby improving data representation. This enables LHASRec to capture the user's preferences better and achieve more accurate personalized recommendations. Secondly, the model considers both the local fluctuation and global stability of the user's interests, demonstrating the ability to model the user's behavior accurately. This comprehensive modeling approach helps capture users' long-term and short-term preferences more accurately.

4.6. Local-Aware Ability (RQ2)

When utilizing sequential models for handling the user's historical interaction sequences, it is often prone to overlooking the impact of the local fluctuation of the user's interests on global stability. To delve deeper into this issue, we conducted a series of experiments. We divided the user's historical interaction sequences into multiple sessions based on different division time interval values (t_d) between adjacent items and compared the performance across four datasets. As shown in Table 4, selecting excessively small t_d (resulting in numerous sessions) or tremendous t_d (resulting in too few sessions) led to a certain degree of decline in the model's recommendation capability. Specifically, when t_d is too small, the model's local-aware ability becomes excessively strong, focusing excessively on the user's short-term interests and disregarding the strong correlations among items in the sequence, thus affecting the accuracy of recommendations. Conversely, when t_d is too large, the model's local-aware ability becomes weak, making it challenging to capture the user's short-term specific interests and failing to promptly reflect changes in the user's interests. We found that the model performed best on the Movielens-1M dataset when the value of t_d was 30 min. For the Games, Beauty, and Steam datasets, the model achieved the best recommendation results when the value of t_d was 15 min. In order to gain a deeper insight into the factors influencing the model's performance, we explored the possibility that it might be due to the additional information used by the model. Consequently, we modified the LHASRec by removing specific user attributes and compared the resulting model with the baseline model. As shown in Table 5, it is evident that even after removing specific user attributes, LHASRec maintains superior performance on the Beauty, Games, and Steam datasets. While its performance on the MovieLens-1M dataset is slightly below that of the optimal model, it still remains significant. This indicates that dividing the user's historical interaction sequence into a reasonable number of sessions is meaningful. It allows for a comprehensive consideration of the user's short-term special interests and the strong correlations among items within sessions, thus better reflecting the local fluctuation of the user's interests and laying a solid foundation for achieving global stability of the user's interests.

Table 4. Impact of different division time interval values on the recommendation performance of the models across four datasets. We have bolded the best-recommended method in each row.

t_d (min)	Beauty		Games		ML-1M		Steam	
	Hit@10	NDCG@10	Hit@10	NDCG@10	Hit@10	NDCG@10	Hit@10	NDCG@10
1	0.5052	0.3239	0.7214	0.4868	0.8214	0.5881	0.8141	0.5575
15	**0.5150**	**0.3402**	**0.7359**	**0.5072**	0.8242	0.5964	**0.8218**	**0.5611**
30	0.5137	0.3367	0.7310	0.5005	**0.8396**	**0.6198**	0.8116	0.5601
45	0.5086	0.3297	0.7258	0.4916	0.8235	0.5909	0.8075	0.5522
60	0.5059	0.3266	0.7218	0.4877	0.8225	0.5892	0.8001	0.5450

Table 5. Recommended performance. We removed specific user attributes from LHASRec and compared the resulting model with the baseline model for analysis. We have bolded the best-recommended method in each row and underlined the second-best-performing approach in each row.

Methods	Beauty		Games		ML-1M		Steam	
	Hit@10	NDCG@10	Hit@10	NDCG@10	Hit@10	NDCG@10	Hit@10	NDCG@10
BPR	0.3775	0.2183	0.4853	0.2875	0.5781	0.3287	0.7061	0.4436
FMC	0.3771	0.2477	0.6358	0.4456	0.6983	0.4676	0.7731	0.5193
FPMC	0.4310	0.2891	0.6082	0.4680	0.7599	0.5176	0.7710	0.5011
TransRec	0.4607	0.3020	0.6838	0.4557	0.6413	0.3969	0.7624	0.4852
Caser	0.4264	0.2547	0.5282	0.3214	0.7886	0.5538	0.7874	0.5381
SASRec	0.4663	0.3080	0.6843	0.4602	0.8285	0.5982	0.7867	0.5108
SSE-PT	0.4963	0.3159	0.6955	0.4677	0.8346	0.6163	0.7885	0.5369
TiSASRec	0.4981	0.3329	0.7080	0.467	**0.8359**	0.6156	<u>0.8053</u>	<u>0.5523</u>
TARN	0.4979	0.3324	0.6996	0.4698	0.8325	0.6139	0.7985	0.5476
BAR	0.4995	0.3336	0.7073	0.4704	0.8351	**0.6192**	0.8039	0.5492
FMLP-Rec	<u>0.5029</u>	<u>0.3351</u>	<u>0.7091</u>	<u>0.4773</u>	0.8291	0.5333	0.8031	0.5470
LHASRec	**0.5092**	**0.3387**	**0.7280**	**0.4996**	<u>0.8354</u>	<u>0.6167</u>	**0.8166**	**0.5556**

4.7. Stochastic Shared Embeddings

In the process of stacking self-attention modules and incorporating user information, the model is prone to overfitting. To alleviate this issue, we conducted a series of experiments with various regularization methods and compared their performance on the MovieLens-1M dataset (Table 6). Through the analysis of Table 6, we found that Stochastic Shared Embeddings (SSE) is a more effective regularization method compared to existing techniques such as Dropout and weight decay. Specifically, we investigated the recommendation performance when using Dropout and L2 regularization alone and their combination. The results showed that in the LHASRec, the overfitting problem was mitigated to some extent by adopting the SSE regularization method, which randomly replaces embedding matrices. Compared to Dropout or L2 regularization alone, the recommendation performance of the LHASRec on the MovieLens-1M dataset improved by approximately 3% and 8%, respectively. Overall, considering the results of our experiments, the combination of SSE, Dropout, and weight decay is the optimal choice for regularization. This comprehensive approach effectively reduces the risk of overfitting and improves the model's performance in recommendation tasks. Therefore, we recommend adopting this combined regularization strategy in practical applications for better recommendation results.

Table 6. Impact of different regularization methods on the recommendation effect on the MovieLens-1M. We have bolded the best-recommended method in each row and underlined the second-best-performing approach in each row.

Methods	Value	NDCG@10	Hit@10
L2	0.0005	0.5049	0.7447
	0.001	0.5083	0.7550
Dropout	0.4	0.5457	0.7990
	0.6	0.5192	0.7770
SSE-SE	-	0.5616	0.8035
L2+ Dropout	0.001 + 0.4	0.5523	0.8089
	0.001 + 0.6	0.5418	0.7952
	0.0005 + 0.4	0.5427	0.7980
	0.0005 + 0.6	0.5382	0.7892
L2 + SSE-SE	0.0005 + SSE-SE	0.5600	0.7982
	0.001 + SSE-SE	0.5641	0.8096
Dropout + SSE-SE	0.4 + SSE-SE	0.5512	0.8002
	0.6 + SSE-SSE	<u>0.5649</u>	<u>0.8103</u>
L2 + Dropout + SSE-SE	0.001 + 0.4 + SSE-SE	**0.5877**	**0.8243**

4.8. Ablation Study (RQ3)

The influence of the maximum sequence length on the model: Considering the different average sequence lengths of the datasets, we set different maximum sequence lengths based on the principle that each dataset's average sequence length is roughly proportional to the model's maximum sequence length. Through experimental observations, we found that the model's recommendation results are notably influenced by the maximum sequence length. Generally, a longer maximum sequence length leads to better recommendation performance. However, when the maximum sequence length exceeds a certain threshold, the recommendation performance of the model starts to decline. In the experimental data shown in Figure 2, we illustrate the variation in recommendation performance of the LHASRec under different maximum sequence lengths. The results indicate that the recommendation performance of the LHASRec improves as the maximum sequence length increases and reaches its optimum at a certain length (e.g., 190 for MovieLens-1M, 50 for Beauty and Games datasets, and 30 for the Steam dataset). However, the recommendation performance starts to deteriorate when the maximum sequence length surpasses this critical value. This is because an excessively long maximum sequence length may introduce irrelevant noise to the recommendation task, affecting the model's ability to utilize limited information for recommendations effectively. In conclusion, the model's recommendation performance is notably affected by the maximum sequence length.

The influence of the number of attention blocks on the model: The number of attention blocks in the model has a significant impact on the recommendation results. Generally, more self-attention blocks can improve the model's ability to fit the data. However, when the number of blocks is too low, the model tends to underfit, while an excessive number of blocks increases the model's complexity, resulting in a long time for fitting the data, and may lead to overfitting, thereby reducing the recommendation performance. In our experiments, we investigated the impact of using different numbers of self-attention blocks in the LHASRec on the recommendation results across four datasets. Based on the experimental data shown in Figure 3, we found that when $b = 1$, the model fails to fit the data well, resulting in poor recommendation performance. When $b = 2$, the model achieves the best performance and optimal recommendations. However, as the number of blocks exceeds 2, the model gradually starts to overfit, and the performance declines. Based on these experimental results, we select two self-attention blocks as the optimal setting across all datasets to balance the model's fitting ability and complexity, thereby obtaining better recommendation performance.

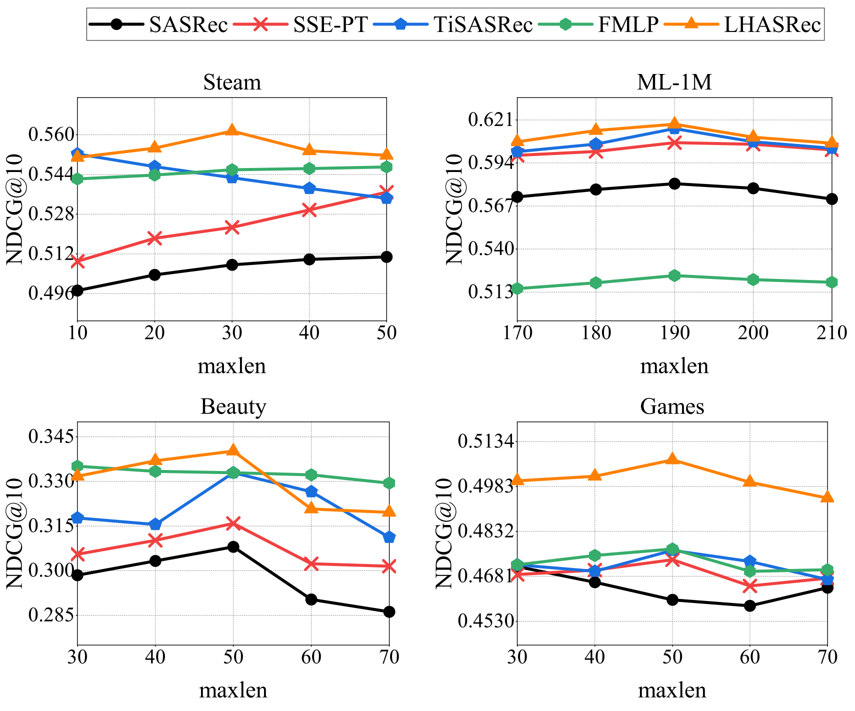

Figure 2. Influence of maximum sequence length on ranking performance (*NDCG*@10).

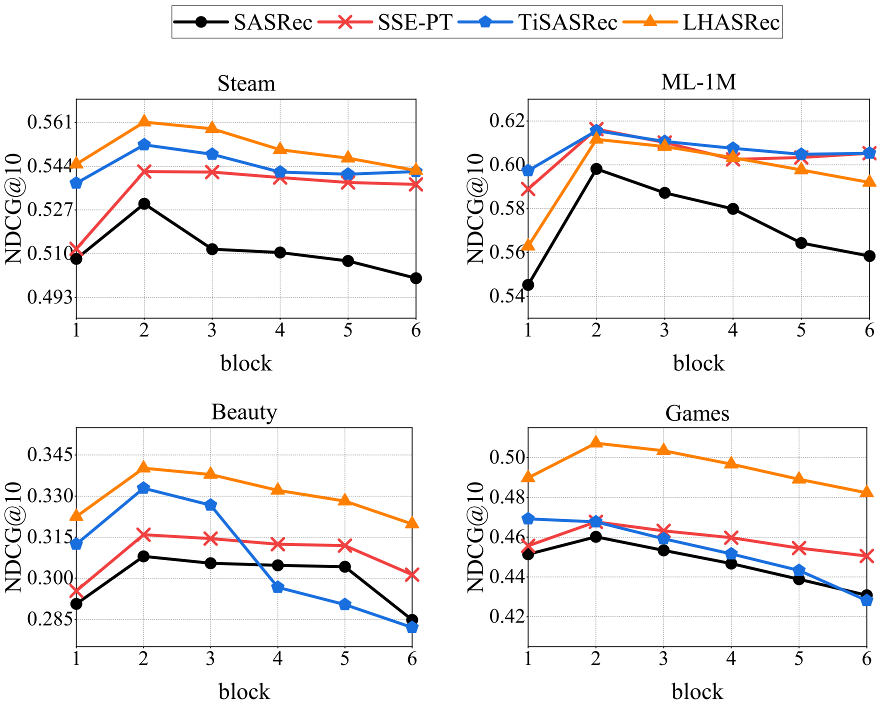

Figure 3. Influence of the number of attention blocks on ranking performance (*NDCG*@10).

5. Conclusions

In this work, we propose a sequential model with local-aware ability (LHASRec). The model comprehensively considers the local fluctuation and global stability of the user's interests to capture the long-term and short-term preferences more accurately. Meanwhile, we enhance the user's historical interaction sequences by embedding user information.

Additionally, we employ the Stochastic Shared Embeddings regularization technique to alleviate overfitting caused by embedding a large amount of user information in the model. Through experiments conducted on sparse and dense datasets, we demonstrate the superiority of LHASRec over various state-of-the-art baseline models. These results highlight the effectiveness and superiority of our proposed approach.

Author Contributions: Conceptualization, J.H.; methodology, J.H.; software, J.H. and Q.L.; validation, J.H., Q.L. and F.Z.; formal analysis, J.H.; investigation, J.H.; resources, J.H.; data curation, J.H.; writing—original draft preparation, J.H.; writing—review and editing, J.H. and F.Z.; supervision, J.H. and Q.L.; project administration, J.H. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the Action Plan for High-Quality Development of Graduate Education of Chongqing University of Technology (No. gzlcx20232102).

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Shani, G.; Heckerman, D.; Brafman, R.I.; Boutilier, C. An MDP-based recommender system. *J. Mach. Learn. Res.* **2005**, *6*, 1265–1295.
2. Tang, J.; Wang, K. Personalized top-n sequential recommendation via convolutional sequence embedding. In Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, Marina Del Rey, CA, USA, 5–9 February 2018; pp. 565–573.
3. Kang, W.C.; McAuley, J. Self-attentive sequential recommendation. In Proceedings of the 2018 IEEE International Conference on Data Mining (ICDM), Singapore, 17–20 November 2018; pp. 197–206.
4. Hosseinzadeh Aghdam, M.; Hariri, N.; Mobasher, B.; Burke, R. Adapting recommendations to contextual changes using hierarchical hidden markov models. In Proceedings of the 9th ACM Conference on Recommender Systems, Vienna, Austria, 16–20 September 2015; pp. 241–244.
5. Rendle, S.; Freudenthaler, C.; Schmidt-Thieme, L. Factorizing personalized markov chains for next-basket recommendation. In Proceedings of the 19th International Conference on World Wide Web, Raleigh, NC, USA, 26–30 April 2010; pp. 811–820.
6. Hidasi, B.; Karatzoglou, A.; Baltrunas, L.; Tikk, D. Session-based recommendations with recurrent neural networks. *arXiv* **2015**, arXiv:1511.06939.
7. Wu, L.; Li, S.; Hsieh, C.J.; Sharpnack, J. SSE-PT: Sequential recommendation via personalized transformer. In Proceedings of the 14th ACM Conference on Recommender Systems, Virtual Event, Brazil, 22–26 September 2020; pp. 328–337.
8. Li, J.; Wang, Y.; McAuley, J. Time interval aware self-attention for sequential recommendation. In Proceedings of the 13th International Conference on Web Search and Data Mining, Houston, TX, USA, 3–7 February 2020; pp. 322–330.
9. Brémaud, P. *Markov Chains: Gibbs Fields, Monte Carlo Simulation, and Queues*; Springer Science & Business Media: Berlin, Germany, 2001; Volume 31.
10. Xue, H.J.; Dai, X.; Zhang, J.; Huang, S.; Chen, J. Deep matrix factorization models for recommender systems. In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17), Melbourne, Australia, 19–25 August 2017; Volume 17, pp. 3203–3209.
11. Hopfield, J.J. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci. USA* **1982**, *79*, 2554–2558. [CrossRef] [PubMed]
12. Elman, J.L. Finding structure in time. *Cogn. Sci.* **1990**, *14*, 179–211. [CrossRef]
13. Hidasi, B.; Quadrana, M.; Karatzoglou, A.; Tikk, D. Parallel recurrent neural network architectures for feature-rich session-based recommendations. In Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, 15–19 September 2016; pp. 241–248.
14. Zhang, Y.; Dai, H.; Xu, C.; Feng, J.; Wang, T.; Bian, J.; Wang, B.; Liu, T.Y. Sequential click prediction for sponsored search with recurrent neural networks. In Proceedings of the AAAI Conference on Artificial Intelligence, Québec City, QC, Canada, 27–31 July 2014; Volume 28.
15. Yuan, F.; Karatzoglou, A.; Arapakis, I.; Jose, J.M.; He, X. A simple convolutional generative network for next item recommendation. In Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, Melbourne, VIC, Australia, 11–15 February 2019; pp. 582–590.
16. Li, C.; Liu, Z.; Wu, M.; Xu, Y.; Zhao, H.; Huang, P.; Kang, G.; Chen, Q.; Li, W.; Lee, D.L. Multi-interest network with dynamic routing for recommendation at Tmall. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, Beijing, China, 3–7 November 2019; pp. 2615–2623.
17. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*.

18. Hendrycks, D.; Gimpel, K. Bridging Nonlinearities and Stochastic Regularizers with Gaussian Error Linear Units. *CoRR* **2016**, abs/1606.08415. Available online: https://www.bibsonomy.org/bibtex/9aaf203ef9c9e38569532ac88603af8e (accessed on 24 August 2023).
19. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
20. McAuley, J.; Targett, C.; Shi, Q.; Van Den Hengel, A. Image-based recommendations on styles and substitutes. In Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, 9–13 August 2015; pp. 43–52.
21. Rendle, S.; Freudenthaler, C.; Gantner, Z.; Schmidt-Thieme, L. BPR: Bayesian personalized ranking from implicit feedback. *arXiv* **2012**, arXiv:1205.2618.
22. He, R.; Kang, W.C.; McAuley, J. Translation-based recommendation. In Proceedings of the Eleventh ACM Conference on Recommender Systems, Como, Italy, 27–31 August 2017; pp. 161–169.
23. Zhang, Q.; Cao, L.; Shi, C.; Niu, Z. Neural time-aware sequential recommendation by jointly modeling preference dynamics and explicit feature couplings. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *33*, 5125–5137. [CrossRef] [PubMed]
24. He, M.; Pan, W.; Ming, Z. BAR: Behavior-aware recommendation for sequential heterogeneous one-class collaborative filtering. *Inf. Sci.* **2022**, *608*, 881–899. [CrossRef]
25. Zhou, K.; Yu, H.; Zhao, W.X.; Wen, J.R. Filter-enhanced MLP is all you need for sequential recommendation. In Proceedings of the ACM Web Conference 2022, Lyon, France, 25–29 April 2022; pp. 2388–2399.
26. He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; Chua, T.S. Neural collaborative filtering. In Proceedings of the 26th International Conference on World Wide Web, Perth, Australia, 3–7 April 2017; pp. 173–182.
27. Koren, Y. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, NV, USA, 24–27 August 2008; pp. 426–434.