Research Article

Self-Attention Based Time-Rating-Aware Context Recommender System

Yongfu Zha, Yongjian Zhang, Zhixin Liu, and Yumin Dong

College of Computer and Information Science Chongqing Normal University, Chongqing, China

Correspondence should be addressed to Yumin Dong; dym@cqnu.edu.cn

Received 20 May 2022; Accepted 14 July 2022; Published 17 September 2022

Academic Editor: G. Thippa Reddy

Copyright © 2022 Yongfu Zha et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The sequential recommendation can predict the user’s next behavior according to the user’s historical interaction sequence. To better capture users’ preferences, some sequential recommendation models propose time-aware attention networks to capture users’ long-term and short-term intentions. However, although these models have achieved good results, they ignore the influence of users on the rating information of items. We believe that in the sequential recommendation, the user’s displayed feedback (rating) on an item reflects the user’s preference for the item, which directly affects the user’s choice of the next item to a certain extent. In different periods of sequential recommendation, the user’s rating of the item reflects the change in the user’s preference.

In this paper, we separately model the time interval of items in the user’s interaction sequence and the ratings of the items in the interaction sequence to obtain temporal context and rating context, respectively. Finally, we exploit the self-attention mechanism to capture the impact of temporal context and rating context on users’ preferences to predict items that users would click next. Experiments on three public benchmark datasets show that our proposed model (SATRAC) outperforms several state-of-the-art methods. The Hit@10 value of the SATRAC model on the three datasets (Movies-1M, Amazon-Movies, Amazon-CDs) increased by 0.73%, 2.73%, and 1.36%, and the NDCG@10 value increased by 5.90%, 3.47%, and 4.59%, respectively.

1. Introduction

With the rapid development of the Internet and big data technology, it is more difficult for people to find the items they need [1]. The emergence of a recommender system can solve this problem. It recommends different items to users according to their browsing history so that people can quickly find what they need. Nowadays, recommender systems have been widely used in e-commerce, social media platforms [2], and online platforms.

Context-based sequential recommendation [3] combines the user’s recent behavior with the recent intention to capture the next preference from their recent behavior. Among sequential recommendation techniques, the Markov chain is very special. It assumes that the next behavior is only related to the previous behavior, and can accurately predict and recommend the next item. Translation-based approaches [4] model third-order interactions with a TransRec component. Collaborative Filtering (CF) [5] is the first technical method used in recommender systems, which is often combined with Matrix Decomposition (MD). In the collaborative filtering algorithm, the method based on MD attempts to change the dimension to represent the relationship between users’ preferences and items’ attributes. Then, the inner product between user embedding and item embedding is estimated. The methods based on MD mainly include nonnegative matrix decomposition [6] and decomposition machine [7]. They learn the similarity matrix between items and then make preference recommendations according to the similarity between users’ history and their interactive items. However, the method based on the Markov chain extremely depends on the users’ last behavior. When the sequence is very long, a large amount of sequential information will be lost, resulting in inaccurate recommendations. At the same time, CF does not consider the order between sequences, cannot capture users’ short-term hobbies and long-term hobbies, and also can not recommend items that users have never browsed.
With the development of deep learning technology, recurrent neural networks, and attention mechanisms are gradually used to capture the interests of users in sequential recommendation [8, 9]. The study of [10] uses a recurrent neural network to process sequential data. Due to the recurrent neural network's ability to capture the memory of the previous step, the model has shown good results in processing sequential data [11]. Because of the great success of the attention mechanism in image processing and natural language processing [12], proposed to combine attention and collaborative filtering to make the next recommendation. [13] combine attention with context. [14] proposed the application of attention mechanism on Gated Recurrent Unit (GRU) to realize sequential recommendation and achieved good results. It classifies the input data format and applies the attention mechanism to the gates of GRU to make decisions on the importance of the information. NARM [15] employs the attention mechanism on RNN to capture users’ main purposes. STAMP [16] uses a novel attention memory network to efficiently capture both the users' general interests and current interests.

Recently, the transformer model [17] has achieved excellent performance and effect on machine translation [18]. It is different from the recurrent neural network (RNN) and convolutional neural network. It is completely based on the self-attention mechanism. [19] proposed the sequential recommendation of the self-attention mechanism, which considers the relative position of items and achieved good results. However, in the process of sequential recommendation, users’ preferences and choices will be changed dynamically with the update of the timestamp. [20] models based on absolute time. [21] proposed the time interval aware self-attention of temporal recommendation, which takes the time interval between any two items into account. [22] considers the self-attention of the relative position of items. DT4SR [23] represents items as distributions, which proposes the mean and covariance embedding to model uncertainty in items. [24] proposed a sequential temporal order recommendation based on a temporal graph collaborative transformer. Inspired by the success of graph neural networks (GNN) in other tasks, [25] proposed a framework based on dynamic graph collaborative filtering to capture the relation between items and users at the same time. [26] proposes a memory augmented graph neural network (MA-GNN) to capture users’ long-term and short-term hobbies. [27–29] applied a graph neural network to a sequential recommendation.

Although these methods have achieved convincing results, we believe that they still have some shortcomings. In different periods of sequential recommendation, the user’s rating of the item reflects the change in the user’s preference, which may affect the user’s choice of the next item. Specifically, in different periods, the user’s different ratings for the same item or the same type of items show feedback on the dynamic changes of the user’s preferences. For example, the user has given low/high ratings to emotional movies at different periods, which may indicate that the user may not like/like watching emotional movies recently. The current research does not take this into account, so this paper proposes a self-attention based time-rating-aware contextual recommendation model to solve this problem. The model considers the influence of users’ rating information on items in different periods on users’ preferences. The main contributions of this paper are as follows:

(i) We use a gated recurrent neural network to model the time interval between items and items in the user interaction sequence and use the user information as the initial hidden information to capture the influence of the temporal information on the user’s preferences to obtain the temporal context.

(ii) We fuse user interaction sequence information and rating information of items in the interaction sequence to obtain rating context. Then, this paper utilizes the self-attention mechanism to model the temporal context and the rating context to better capture the user’s dynamically changing preferences.

(iii) Finally, we conduct experimental studies on three datasets. Extensive experiments show that it outperforms state-of-the-art models.

2. Related Work

In this section, we separately review three tasks related to our work: contextual recommendation, GRU, and self-attention.

2.1. Contextual Recommendation. The study of [30] pointed out that users’ preferences are affected by different types of contexts, and the reasonable use of contextual information by the recommender system is beneficial to the recommended performance. Therefore, they propose the concept of context-aware recommender systems. The study of [14] proposed an attention-based context-aware recommender system. [31] proposed location-aware contextual attention based on session recommendation, which improves recommended performance. Therefore, they propose the concept of context-aware recommender systems. The study of [14] proposed an attention-based context-aware recommender system. The study of [31] proposed location-aware contextual attention based on session recommendation, which improves recommended performance. Therefore, they propose the concept of context-aware recommender systems. The study of [14] proposed an attention-based context-aware recommender system. The study of [31] proposed location-aware contextual attention based on session recommendation, which improves recommended performance. Therefore, they propose the concept of context-aware recommender systems. The study of [14] proposed an attention-based context-aware recommender system. The study of [31] proposed location-aware contextual attention based on session recommendation, which improves recommended performance. Therefore, they propose the concept of context-aware recommender systems. The study of [14] proposed an attention-based context-aware recommender system. The study of [31] proposed location-aware contextual attention based on session recommendation, which improves recommended performance. Therefore, they propose the concept of context-aware recommender systems. The study of [14] proposed an attention-based context-aware recommender system. The study of [31] proposed location-aware contextual attention based on session recommendation, which improves recommended performance. Therefore, they propose the concept of context-aware recommender systems. The study of [14] proposed an attention-based context-aware recommender system. The study of [31] proposed location-aware contextual attention based on session recommendation, which improves recommended performance. Therefore, they propose the concept of context-aware recommender systems. The study of [14] proposed an attention-based context-aware recommender system. The study of [31] proposed location-aware contextual attention based on session recommendation, which improves recommended performance. Therefore, they propose the concept of context-aware recommender systems. The study of [14] proposed an attention-based context-aware recommender system. The study of [31] proposed location-aware contextual attention based on session recommendation, which improves recommended performance. Therefore, they propose the concept of context-aware recommender systems. The study of [14] proposed an attention-based context-aware recommender system. The study of [31] proposed location-aware contextual attention based on session recommendation, which improves recommended performance. Therefore, they propose the concept of context-aware recommender systems. The study of [14] proposed an attention-based context-aware recommender system. The study of [31] proposed location-aware contextual attention based on session recommendation, which improves recommended performance. Therefore, they propose the concept of context-aware recommender systems. The study of [14] proposed an attention-based context-aware recommender system. The study of [31] proposed location-aware contextual attention based on session recommendation, which improves recommended performance. Therefore, they propose the concept of context-aware recommender systems.
2.2. GRU and Self-Attention. The study of [14] proposed an attention-based context-aware sequential recommendation model by redefining the update gate and reset gate of the GRU unit. The study of [35] reconstructed GRU and proposed a neural template explanation model for recommendation generation based on GFRU. MTAM [36] exploits temporal awareness for a sequential recommendation. MTAM utilizes GRU to learn short-term intentions and self-attention to learn long-term intentions to capture users’ hobbies. CoSAN [37] utilizes self-attention to learn long-term dependencies between collaborative items and generate collaborative session representations. TiSASRec [21] utilizes self-attention to model the absolute positions of items and the time interval between them. This paper utilizes GRU to capture the effect of the time factor on user preferences. Different from these models, the user’s information is used as the initial hidden information of the GRU to obtain the temporal context. The user’s rating on the item directly reflects the user’s liking for the item, which will affect the user’s next choice of item. In different periods, the user’s rating of the item reflects the change in the user’s liking. We utilize a self-attention mechanism to capture the effect of item ratings on users’ preferences over different periods.

3. Methodology

In this section, we will define the problem of sequential recommendation, along with a description of the embedding layer, the temporal context module, and the SATRAC model. Before that, we briefly introduced the computational recommendation, along with a description of the embedding layer. In this section, we will define the problem of sequential recommendation, along with a description of the embedding layer.

Table 1: Notation.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>U, I, R, T</td>
<td>Set of users, items, ratings and time</td>
</tr>
<tr>
<td>I_u</td>
<td>Item sequence for user u</td>
</tr>
<tr>
<td>T_u</td>
<td>Time sequence of items for user u</td>
</tr>
<tr>
<td>R_u</td>
<td>Rating sequence of items for user u</td>
</tr>
<tr>
<td>m</td>
<td>The maximum length of the sequence</td>
</tr>
<tr>
<td>d</td>
<td>Embedding dimension</td>
</tr>
<tr>
<td>C^u_t</td>
<td>Time context information of user u</td>
</tr>
<tr>
<td>C^u_r</td>
<td>Rating context information of user u</td>
</tr>
<tr>
<td>O_t</td>
<td>Output of the model at time t</td>
</tr>
</tbody>
</table>

Different from other modeling using recurrent neural networks, we let the initial hidden layer of GRU input user information. By incorporating item information and time interval information into the network, the effect of time on current user preferences is captured as the time interval changes. The self-attention mechanism uses parallel computing and does not need to rely on the hidden layer computation at the previous moment. It is defined as follows:

$$ \text{Attention} = \text{softmax} \left( \frac{QK^T}{\sqrt{d}} \right) V, $$

where Q, K and V represent query, keyword and value, respectively. In SATRAC model, Q stands for rating context, K stands for temporal context, and V stands for item information.

3.1. Problem Formulation. Let U, I, S, and T denote the set of user, item, rating, and time, respectively. The purpose of the sequential recommendation is to predict the item \( I_u^k \) that the user clicks at the next time \( k \) when the user’s historical item sequence \( (I_u^1, I_u^2, \ldots, I_u^k) \) is known. In the SATRAC model, for each user \( u \in U \), the historical interaction sequence of user \( u \) is \( I_u = (I_u^1, I_u^2, \ldots, I_u^k) \), the corresponding temporal sequence and rating sequence are \( T_u = (T_u^1, T_u^2, \ldots, T_u^k) \) and \( S_u = (S_u^1, S_u^2, \ldots, S_u^k) \), respectively. Generally speaking, the sequential recommendation will limit the length of the sequence, and we will explain how to deal with the length of the sequence later. In the training process, at time \( t \), our input is \( I_t = (I_t^1, I_t^2, \ldots, I_t^{t-1}) \), \( T_t = (T_t^1, T_t^2, \ldots, T_t^{t-1}) \), and \( S_t = (S_t^1, S_t^2, \ldots, S_t^{t-1}) \). At time \( t + 1 \), the next item output we expect is \( I_{t+1} = (I_{t+1}^1, I_{t+1}^2, \ldots, I_{t+1}^k) \).

3.2. Embedding Layer. Before the input of the model, we must convert our data into an embedding representation. For user \( u \), use \( T_u = (T_u^1, T_u^2, \ldots, T_u^{t_{\text{max}}}) \) to represent the temporal sequence corresponding to the items in the user interaction sequence, and use \( I_u = (I_u^1, I_u^2, \ldots, I_u^{t_{\text{max}}}) \) represents the user’s item sequence, \( S_u = (S_u^1, S_u^2, \ldots, S_u^{t_{\text{max}}}) \) represents the rating sequence corresponding to the item in the user interaction sequence. In the process of embedding representation, we fix the length of our input to \( m \). If the length of the user’s behavior sequence is less than \( m \), then we will add a padding item 0 on the left side. If the length of the user’s behavior sequence is greater than \( m \), then we will select the most recent \( m \) historical behaviors as our input.

The embedding layer is as follows:

$$ E_u^I = \text{Embedding}(I_u), $$

$$ E_u^T = \text{Embedding}(T_u), $$

$$ E_u^S = \text{Embedding}(S_u), $$

$$ E_u^O = \text{Embedding}(O_u), $$

where \( E \) is the embedding layer.
where $E_u^t \in \mathbb{R}^{m \times d}$ represents the embedding of the corresponding temporal sequence in the user $u$ interaction sequence, $E_i^t \in \mathbb{R}^{m \times d}$ represents the embedding of the item corresponding to user $u$, $E_u^t \in \mathbb{R}^{1 \times d}$ represents the embedding of user $u$, $E_i^t \in \mathbb{R}^{m \times d}$ represents the embedding of the corresponding rating sequence in the user $u$ interaction sequence.

3.3. Temporal Context Module. The method adopted by [19] discards the timestamp, assuming that the time interval between all items is the same, and only retains the user’s behavior sequence. Recently, [21] achieved good results by modeling the absolute positions of items and their time intervals in the sequence. [14] proposed to use the attention mechanism on the GRU and make improvements to use time in the recommendation. We build a network through the Gated Recurrent Unit (GRU) to capture the influence of the temporal sequence on the user’s behavior. In the processing of time, the difference between the time of two adjacent items is used to represent the time information. We also input the user’s information into the GRU network, because we think that for different users, even if the behavior sequence and time series are the same, the final recommendation should be different. The temporal context module is shown in Figure 1.

The following are the initial values of the input and hidden layers:

$$h_0 = E_u^t,$$

$$x^t = e_i^t + e_u^t,$$  \hspace{1cm} (4)

where $e_i^t$ represents the time embedding of the user at time $t$, and $e_i^t$ represents the item embedding at time $t$. We can get a time context $C^t_{G}$ through the network:

$$C^t_{G} = (h_1, h_2, \ldots, h_m),$$  \hspace{1cm} (5)

where $C^t_{G}$ is the temporal context that contains user information and item time information.

3.4. SATRAC Model. The user’s rating of an item can be seen as displayed feedback. The rating can reflect the user’s hobby for the current item, which greatly affects the user’s next choice of item. First, we fuse the rating information of the item with the user interaction sequence information to obtain the rating context. In different periods, to better capture the impact of rating on users. We utilize a self-attention mechanism to capture the effects of temporal context and rating context on user hobby. Figure 2 shows the structure of the SATRAC model.

The SATRAC model includes self-attention layers, residual connections, layer normalization, and pointwise feedforward neural networks.

$$C^t_S = E^t_S + E^t_I,$$  \hspace{1cm} (6)

where $C^t_S \in \mathbb{R}^{m \times d}$ represents the rating context that fuses the item information in the interaction sequence and the corresponding rating information of the item in the interaction sequence.

$$Z^t = \text{Attention}(C^t_S, C^t_I, E^t_I),$$  \hspace{1cm} (7)

where $Z^t \in \mathbb{R}^{m \times d}$, $Q$ represents the rating context, $K$ represents the temporal context, and $V$ represents the item information.

3.4.1. Residual Connection. In the past, to better extract features, more often they used the superposition of multi-layer neural networks [39]. However, in some cases, multilayer neural networks did not have better performance until the emergence of residual networks [40]. The main idea of the residual network is to transfer low-dimensional features to high-dimensional through residual connection. In this paper, the rating context information and the information captured by the self-attention mechanism are residually connected.

$$Z^t = Z^t + C^t_S.$$  \hspace{1cm} (8)

3.4.2. Layer Normalization. To enhance the capabilities of the model and accelerate neural network training [41], the statistical data used in layer standardization is independent of other samples in the same batch. The operation is defined as follows:
where LayerNorm \( (x) \) is defined as follows:

\[
\text{LayerNorm} (x) = \alpha \odot \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta,
\]

where \( \odot \) is the elementwise product, \( \mu \) and \( \sigma \) are the mean and variance of \( x \), and \( \alpha \) and \( \beta \) are the scale factors and bias terms, respectively.

**3.4.3. Pointwise Feed-Forward Network.** Although self-attention can aggregate temporal and rating contexts with adaptive weights, it is ultimately a linear model. To give the model nonlinearity and consider the interaction between different potential dimensions, we adopted the GELU function [42]. To solve the model’s overfitting, we used the dropout technique [43].

\[
\text{FFN} (Z^u) = \text{Dropout} (\text{GELU} (Z^u)).
\]

Finally, after stacking self-attention layers, we obtain the following equation:

\[
Z^u = (z_1^u, z_2^u, \ldots, z_m^u),
\]

where \( Z^u \) represents the embedded information of user \( u \) representing the click on the next item.

**3.5. Prediction Layer.** Like the [19, 21] model, to predict the next item the user clicks, we rank the items using the following computation:

\[
\text{Rank}^u = Z^u M^i,
\]

where \( M^i \in \mathbb{R}^d \) is the embedding of item \( i \). Finally, get the ranking of \( m \) items, and then recommend the top-ranked items to user \( u \).

**3.6. Model Training and Loss Function.** The sequence of items input for our model training is: \( I^u = (I_1^u, I_2^u, \ldots, I_{|I^u|-1}^u) \), and then a fixed length \( I^u = (I_1^u, I_2^u, \ldots, I_m^u) \), we expect the output at time \( t \) to be:

\[
O_t = \begin{cases} 
\langle \text{pad} \rangle & \text{if } I_t^u \text{ is a padding item} \\
I_{t+1} & 1 \leq t < m \\
I_m^u & t = m 
\end{cases}
\]

\[
\text{Table 2: Dataset statistics.}
\]

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Users</th>
<th>Items</th>
<th>Interactions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Movies-1M</td>
<td>6040</td>
<td>3416</td>
<td>999611</td>
</tr>
<tr>
<td>Amazon-Movies</td>
<td>40928</td>
<td>37654</td>
<td>1163413</td>
</tr>
<tr>
<td>Amazon-CDs</td>
<td>26875</td>
<td>42779</td>
<td>770188</td>
</tr>
</tbody>
</table>

where \( < \text{pad} > \) represents the padding item. \( O_t \) represents the output we expect. We use binary cross entropy loss as the objective function:

\[
- \sum_{t=1}^{m} \sum_{j \in \mathcal{F}_m} \left[ \log \sigma (R_{O_t, j}) + \sum_{j \notin \mathcal{F}_m} \log (1 - \sigma (R_{j, t})) \right].
\]

For padded items, ignore its loss. Our model uses Adam optimizer [44].

**4. Experiment**

In this section, we validate the effectiveness of the SATRAC model on three datasets. First, we introduce our experimental setup and compare it with current state-of-the-art baselines. Then, we discuss the effect of some hyperparameters on the SATRAC model. Finally, we discuss the impact of temporal information and rating information on the SATRAC model.

**4.1. Dataset.** We evaluate our method on three real datasets. These datasets have different sizes, and different sparsity, and are all common datasets. As shown in Table 2.

- **Movies-1M** https://www.kaggle.com/datasets/odedgolden/movielens-1m-dataset: it contains 1 million movie records rated by users.
- **Amazon** https://www.kaggle.com/datasets/deovcs/amazon-dataset: we use the Amazon-Movies and Amazon-CDs datasets from the series of datasets presented in [45].

All three datasets include user ratings and timestamps, and we filter out cold-start users and users with less than 3 interaction sequences. When calculating the time interval between adjacent items, we set a maximum time interval value. If the time interval of adjacent items in the user interaction sequence is greater than the time interval value we set, then we set the time interval of adjacent items to a fixed value. On the three datasets, the maximum interval value we set is 2048. As proposed in [19], for each user, we use the latest
item in the interaction record for testing, the second-to-last item for validation, and the remaining items for training.

4.2. Evaluation Index. We use two frequently used metrics Hit@10 and NDCG@10 to evaluate the performance of the model [46]. Hit@10 calculates the ratio of basic fact items in the top 10 items. NDCG@10 will consider location and assign higher weights to higher locations. For each user $u$, we randomly sample $m - 1$ negative items and rank these items with basic fact items. We calculate Hit@10 and NDCG@10 based on the ranking of these $m$ items (This is slightly different from [19, 21]).

4.3. Baselines. To demonstrate the effectiveness of the SATRAC model, we compare SATRAC with the following baselines, which include some classic baselines and some state-of-the-art baselines.

(i) **BPR** [47]: the bayesian ranking is personalized and often used as a matrix factorization recommender.

(ii) **GRURec** [11]: it uses RNN to model user action sequences for session-based recommendations. Regard each user’s feedback sequence as a conversation, and then perform a recommended algorithm.

(iii) **Caser** [48]: a convolution-based algorithm designed to capture the general preferences and order patterns of users.

(iv) **SASRec** [19]: a sequential recommendation model based on a self-attention mechanism.

(v) **MARank** [49]: this method considers recent items and applies multi-order attention to capture single and joint-level item relevance.

(vi) **HGN** [50]: a sequential model that includes feature selection and instance selection modules combined with Bayesian personalized ranking (BPR) to obtain long-term and short-term user interests.

(vii) **TiSASRec** [21]: a recently proposed state-of-the-art model, this model mainly considers self-attention sequential recommendation for time interval modeling.

(viii) **SATRAC-NR**: a variant of the SATRAC model. SATRAC-NR does not add temporal context information and replaces temporal context information with rating context information for calculation. This is beneficial to show the impact of temporal context information on the SATRAC model.

(ix) **SATRAC-NS**: a variant of the SATRAC model. SATRAC-NS does not add rating information and replaces rating context information with time context information for calculation. This is beneficial to show the impact of rating context information on the SATRAC model.

For some comparative baselines, we used the same dataset and we directly report the experimental results of their paper. For other baselines using the self-attention mechanism, we reproduce the model under the premise that the model parameters are set optimally to obtain the best performance of the model. For the fairness of the comparison, our model uses the common framework pytorch and the same optimization algorithm adam. All datasets used a learning rate of 0.001 and were trained the same number of times. The number of self-attention heads and self-attention layers is both set to 1, the value of dropout is 0.2, and we set both $m$ and $d$ to 50. To discuss the impact of embedding dimension $d$ and sequence length $m$ on the model recommended performance, we do some comparative experiments. The $m$ value is selected as [10, 20, 30, 40, 50], and the potential dimension $d$ is selected as [10, 20, 30, 40, 50], and different values are compared. If all metrics are lower than the last evaluation metrics, the training is terminated. The maximum number of training sessions is 200. During implementation, the batch size used is 128, and all experiments are performed with a GTX-1650 Ti graphics processor. This paper discusses the impact of embedding dimension $d$ and sequential length $m$ on the model recommended performance in Figures 3 and 4, respectively.

4.4. Recommended Performance. Table 3 shows the recommended performance of all baselines on the three datasets. On Movies-1M, Amazon-Movies, and Amazon-CDs datasets, the SATRAC model shows the state-of-the-art recommended performance. Among the baseline methods, MARank has state-of-the-art performance compared with other baselines. MARank can capture individual-level, and union-level user interactions and tackle sparse datasets well. On the dense dataset Movies-1M, the recommended performance of the self-attention-based baseline is significantly better than the ordinary baseline. Because the self-attention mechanism has a stronger ability to capture long-term sequential models. On Movies-1M and Amazon-Movies datasets, the TiSASRec model outperforms SASRec, which shows that utilizing temporal information can help improve recommended performance. HGN outperforms TiSASRec on the Amazon-CDs dataset, which may be because the HGN model can capture both the long-term and short-term hobbies of users. On the three datasets, SATRAC outperforms the best baseline methods on both metrics. On the one hand, SATRAC takes into account the influence of temporal information on user preferences, utilizing a gated neural network to obtain temporal context. On the other hand, we consider the impact of user ratings on items on users’ preferences. Integrate item information and rating information to obtain context. Finally, SATRAC utilizes a self-attention mechanism to capture the effects of temporal context and rating context on users’ preferences. Most of the existing baselines only consider the impact of time information on users’ preferences.

4.5. Influence of Hyperparameters. In Figure 3, we discuss the effect of the embedding dimension $d$ on Hit@10 and NDCG@10, while setting other hyperparameters to be optimal. The value of the embedding dimension $d$ is [10, 20, 30,
40, 50]. On the three datasets, the recommended performance of the SATRAC model will gradually improve. This is because as the dimension $d$ becomes larger and larger, more feature information can be represented, but when the $d$ value reaches a certain value, the $d$ value has a limited impact on the model, and the final model will tend to be stable, eventually, the model will stabilize. In Figure 4, we discuss the effect of sequential length $m$ on NDCG@10, while setting other hyperparameters to be optimal. The value of the sequential length $m$ is [10, 20, 30, 40, 50]. When the value of $m$ becomes larger, the recommended performance of the SATRAC model, SASRec model, and TiSASRec model starts to decrease. This is because a lot of noise is generated as the sequential length increases, which can lead to lower recommendation model performance. Compared with the SASRec model and the TiSASRec model, the recommended performance of the SATRAC model degrades more slowly, which shows the importance of rating information and temporal information in the sequential recommendation, and also shows the effectiveness of the SATRAC model in capturing time information and rating.

4.6. Discussion. The effects of time information and rating information on the recommended performance of the SATRAC model are shown in Table 4. On the three datasets,
the SATRAC-NT model outperforms the SASRec model. This illustrates that rating information is important for sequential recommendation tasks, and the SATRAC-NT model is effective in capturing rating information. On the three datasets, the SATRAC-NR model outperforms the TiSASRec model, which demonstrates the effectiveness of

Table 3: Comparison of recommended performance of the SATRAC model and other baselines. The best results are in bold. The underlined numbers are suboptimal results.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Metric</th>
<th>BPR</th>
<th>GRURec</th>
<th>Caser</th>
<th>MARank</th>
<th>SASRec</th>
<th>HGN</th>
<th>TiSASRec</th>
<th>SATRAC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Movies-1M</td>
<td>Hit@10</td>
<td>0.5922</td>
<td>0.5781</td>
<td>0.7487</td>
<td>0.7622</td>
<td>0.8493</td>
<td>0.8123</td>
<td>0.8529</td>
<td>0.8591</td>
</tr>
<tr>
<td></td>
<td>NDCG@10</td>
<td>0.3401</td>
<td>0.3781</td>
<td>0.5123</td>
<td>0.5169</td>
<td>0.5712</td>
<td>0.5448</td>
<td>0.5767</td>
<td>0.6107</td>
</tr>
<tr>
<td>Amazon-movies</td>
<td>Hit@10</td>
<td>0.5517</td>
<td>0.5368</td>
<td>0.4938</td>
<td>0.6546</td>
<td>0.7563</td>
<td>0.7429</td>
<td>0.7647</td>
<td>0.7856</td>
</tr>
<tr>
<td></td>
<td>NDCG@10</td>
<td>0.3464</td>
<td>0.3317</td>
<td>0.3201</td>
<td>0.4562</td>
<td>0.5238</td>
<td>0.5167</td>
<td>0.5334</td>
<td>0.5519</td>
</tr>
<tr>
<td>Amazon-CDs</td>
<td>Hit@10</td>
<td>0.5627</td>
<td>0.5364</td>
<td>0.6865</td>
<td>0.6981</td>
<td>0.7015</td>
<td>0.7045</td>
<td>0.7001</td>
<td>0.7141</td>
</tr>
<tr>
<td></td>
<td>NDCG@10</td>
<td>0.3626</td>
<td>0.3267</td>
<td>0.4585</td>
<td>0.4589</td>
<td>0.4635</td>
<td>0.4662</td>
<td>0.4604</td>
<td>0.4876</td>
</tr>
</tbody>
</table>

Table 4: We discuss the impact of temporal information and rating information on SATRAC models.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Metric</th>
<th>SATRAC-NT</th>
<th>SATRAC-NR</th>
<th>SATRAC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Movies-1M</td>
<td>Hit@10</td>
<td>0.8529</td>
<td>0.8561</td>
<td>0.8591</td>
</tr>
<tr>
<td></td>
<td>NDCG@10</td>
<td>0.5812</td>
<td>0.6012</td>
<td>0.6107</td>
</tr>
<tr>
<td>Amazon-movies</td>
<td>Hit@10</td>
<td>0.7782</td>
<td>0.7717</td>
<td>0.7856</td>
</tr>
<tr>
<td></td>
<td>NDCG@10</td>
<td>0.5421</td>
<td>0.5406</td>
<td>0.5519</td>
</tr>
<tr>
<td>Amazon-CDs</td>
<td>Hit@10</td>
<td>0.7066</td>
<td>0.7005</td>
<td>0.7141</td>
</tr>
<tr>
<td></td>
<td>NDCG@10</td>
<td>0.4742</td>
<td>0.4714</td>
<td>0.4876</td>
</tr>
</tbody>
</table>

Figure 4: The effect of different sequential length \( m \) on recommended performance (NDCG@10).
the SATRAC-NR model in capturing temporal information. Compared with the TiSASRec model, the SATRAC-NR model first uses the GRU network to capture the user’s temporal interaction information and the user’s information and then uses the self-attention mechanism to further capture the impact of temporal information on the user’s preferences. The SATRAC model outperforms the SATRAC-NT model and the SATRAC-NR model, which indicates that the SATRAC model is reasonable and effective to capture the user’s time and rating information.

5. Conclusion

This paper proposes a self-attention-based time-rating-aware context recommender system model (SATRAC). The SATRAC model first uses a gated recurrent neural network to model the user interaction sequence and the time interval information of items in the interaction sequence to capture the impact of time information on user preferences, thereby obtaining the time context. Then, the user interaction sequence information and the rating information of the items in the sequence are fused. Finally, a self-attention mechanism is used to learn the effects of temporal context and rating context on user preferences. Extensive experiments on the three datasets show that the SATRAC model outperforms state-of-the-art models, which demonstrates the superiority of the SATRAC model in processing data with time information and rating information. Current research and future research must be discussed under the premise of ensuring user privacy and security. In future research, we will also explore the influence of some other item attributes (such as item description and item image) on sequence recommendation, so that the sequence recommendation model has a more powerful recommendation performance.

Data Availability

The data used to support the study are included in the paper. https://www.kaggle.com/datasets/odedgolden/movieLens-1m-dataset. https://www.kaggle.com/datasets/devocs/amazon-dataset.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The study was supported by National Natural Science Foundation of China (Grant nos. 61772295, 61572720, and 61173056), the Science and Technology Research Program of Chongqing Municipal Education Commission (Grant no. K7ZD-M202000501), Chongqing Technology Innovation and Application Development Special General Project (Grant no. cstc2020jcsx-lyjsA0002), and Chongqing Technology Foresight and System Innovation Project (Grant no. cstc2021jsyj-yzysbA0011).

References


[42] D. Hendrycks and K. Gimpel, Gaussian Error Linear Units (Gelu), 2016.


