Retraction

Retracted: A Self-Attention Mask Learning-Based Recommendation System

Abeer Aljohani; Mohamed Ali Rakrouki; Nawaf Alharbe; Reyadh Alluhaibi

IEEE Access

10.1109/ACCESS.2022.3202637

Notice of Retraction A. Aljohani, M. A. Rakrouki, N. Alharbe, and R. Alluhaibi, “A self-attention mask learning-based recommendation system,” IEEE Access, vol. 10, pp. 93017–93028, 2022, doi: 10.1109/ACCESS.2022.3202637. After careful and considered review of the content of this article by a duly constituted expert committee, this article has been found to have violated IEEE publication principles. Specifically, this article copied portions of content from the following source without appropriate reference: Huaiwen He et al, “GAT4Rec: Sequential recommendation with gated recurrent unit and transformers” IEEE Access, Submitted Jun. 2021. Therefore, IEEE has retracted the content of this article from Xplore. When informed of the retraction, the authors did not respond.

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 7 August 2022, accepted 25 August 2022, date of publication 29 August 2022, date of current version 9 September 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3202637

RESEARCH ARTICLE

A Self-Attention Mask Learning-Based Recommendation System

ABEER ALJOHANI[1], MOHAMED ALI RAKROUKI[1,2,3], NAWAF ALHARBE[1], AND REYADH ALLUHAIBI[4]

[1]Applied College, Taibah University, Madinah 1089, Saudi Arabia
[2]Ecole Supérieure des Sciences Économiques et Commerciales de Tunis, University of Tunis, Tunis 1938, Tunisia
[3]Business Analytics and Decision Making Laboratory (BADEM), Tunis Business School, University of Tunis, Tunis 1938, Tunisia
[4]Department of Computer Science, College of Computer Science and Engineering, Taibah University, Madinah 1089, Saudi Arabia

Corresponding author: Abeer Aljohani (aahjohani@taibahu.edu.sa)

ABSTRACT The primary purpose of sequence modeling is to record long-term interdependence across interaction sequences, and since the number of items purchased by users gradually increases over time, this brings challenges to sequence modeling to a certain extent. Relationships between terms are often overlooked, and it is crucial to build sequential models that effectively capture long-term dependencies. Existing methods focus on extracting global sequential information, while ignoring deep representations from subsequences. We argue that limited item transfer is fundamental to sequence modeling, and that partial substructures of sequences can help models learn more efficient long-term dependencies compared to entire sequences. This paper proposes a sequence recommendation model named GAT4Rec (Gated Recurrent Unit And Transformer For Recommendation), which uses a Transformer layer that shares parameters across layers to model the user's historical interaction sequence. The representation learned by the gated recurrent unit is used as a gating signal to filter out better substructures of the user sequence. The experimental results demonstrate that our proposed GAT4Rec model is superior to other models and has a higher recommendation effectiveness.

INDEX TERMS Recommendation algorithm, machine learning, sequence recommendation model.

I. INTRODUCTION

In daily life, the products that users buy online are usually based on historical experience or current interests, so many e-commerce companies provide us with suggestions based on users' historical shopping information. Compared with recommendation based on user or item similarity, sequence recommendation [1], [2], [3] considers the relationship between items and can better model user history information. In addition, due to privacy concerns, user ids are not always available, and sequence recommendation is more suitable for such scenarios than other methods. The purpose of sequence recommendation is to capture the transition paradigm in the user-item interaction sequence, and take the product set with high probability as the list to be recommended according to the learned hidden layer representation [4].

To obtain sequential patterns of user-item interactions more efficiently, different approaches have been proposed to learn complex representations. Researchers have attempted to apply mathematical models to recommendation methods, such as Markov Chains (MC) [5], [6]. MC is a strong hypothesis model that stipulates that the next behavior depends only on the first N behaviors. This method can achieve good results on short sequences, but it performs poorly on longer sequences and is difficult to capture internal deep relationships. Deep learning can automatically extract deep-level features of sequence information, and can combine various features to fuse all information into low-dimensional vectors. Recently, with the new wave of research in deep learning, Recurrent Neural Network (RNN)-based methods [7], [8], [9], [10] have also made great progress in the field of recommendation. The recommendation algorithm based on RNN mainly uses its natural time series structure, which can effectively model the sequence behavior, so as to obtain the

The associate editor coordinating the review of this manuscript and approving it for publication was Alberto Cano.

hidden layer representation containing the context information of the historical behavior, and predict the possible user interaction at the next moment according to the obtained information. Convolutional Neural Network (CNN)-based methods [1], [11], [12] use convolution kernels of different sizes to obtain features with different amounts of informative predictors. In addition, many recent works have used self-attention mechanisms [13], [14] to replace RNNs and CNNs and demonstrated their advantages in processing sequential information. Compared with RNN, the method based on self-attention mechanism has a high degree of parallelism, so it has a natural advantage in training speed, and at the same time, compared with CNN, this method can process longer sequences.

However, the current models have the following shortcomings: (1) these methods do not effectively use the auxiliary information of the sequence, they only model the product serial number without considering how to combine other features; (2) ignore the changes in user interests, may negatively affect the model; (3) Although the whole sequence as the input of the model can reflect the authenticity of the user's behavior, we believe that the representation of the user's interest only depends on a part of the sequence. The optimal subsequence of user behavior is crucial to make targeted recommendations for users.

To address the above problems, i.e., the problems of user interest change, loss of sequence information, and difficulty in model expansion, a sequence recommendation model GAT4Rec based on self-attention mask learning is proposed, which can capture user interest change signals with auxiliary information. The model extracts item side information as the gating signal of the gated filtering layer, and adds positional coding to the filtered subsequences so that the model can effectively learn the weight ratio between items. The random addition of mask symbols to the model input sequence enables the model to speculate and learn more complex item feature representations combined with bidirectional information. In addition, a joint loss function is designed to simultaneously train the parameters of Transformer and GRU modules. Finally, comparative experiments are conducted on four real datasets, and the GAT4Rec model outperforms some selected state-of-the-art benchmarks.

The main contributions of this paper are as follows:

- Propose a bidirectional self-attention model GAT4Rec incorporating auxiliary information. In order to capture the user's dynamic interest, the GRU module is used to model the category information of the sequence, and the learned hidden layer variables are used as the user's intention representation.
- A gating module is designed to utilize the learned latent representation to obtain the optimal subsequence. The distance between the item's embedding vector and the user's intent representation determines the similarity between the two, and the gating module determines the composition of subsequences according to the threshold.
- Experiments on public datasets demonstrate that GAT4Rec outperforms state-of-the-art methods on both recall and ranking tasks. Later ablation experiments show how the selected components affected the outcomes of the experiment.

The rest of this paper is organized as follows. Relevant related works are presented in Section II. Section III describes the structure of each part of the model and how it is trained. Section IV introduces the experimental setup, and presents the experimental results of the proposed approach, including comparative experiments, hyperparameter influence experiments, and ablation experiments. Finally, the content of this paper is summarized.

II. LITERATURE REVIEW

Sequence recommendation is a subfield of recommender systems, and the difference from other recommendation tasks is that sequence recommendation considers the order dependencies in interaction sequences. For example, for recommended methods such as matrix decomposition, the position of the item in the sequence is not important, only the interaction item and the non-interaction item need to be considered. In the sequence recommendation task, the transfer paradigm between items can reflect the change of users' interests, and capturing sequence signals is beneficial to make more accurate recommendations for users. At present, in the field of sequence recommendation based on deep learning, the most widely used models are: RNN, CNN, and self-attention module. The sequence recommendation algorithm based on the recurrent neural network tries to find the transformation paradigm between the items in the sequence through the user sequence information, so as to find the item that may appear in the next item in the sequence [15], [16], [17]. Hidasi *et al.* [18] proposed GRU4Rec, which aims to leverage GRU modules to learn session-based latent representations for sequences. Subsequently, GRU4Rec+ [19] was proposed as an improved version of GRURec, which models item sequences at the feature level while optimizing the BPRMAX loss function. CNN-based methods have also achieved great success, and the sliding window in CNN can ensure that the model learns the local sequence information within the window. Tang *et al.* [1] proposed Caser, which regarded the embedding combination of the model input as a ''picture'', used the filters in the CNN to search for local sequential pattern information, and captured the important information in the hidden layer features through max pooling, and finally The item features and user features are spliced to obtain the probability distribution of the interaction between the current user and each item at the next moment through the softmax operation. Yuan *et al.* [11] believed that Caser's maximization pooling scheme may lose position information when dealing with long sequences, so stacked one-dimensional dilated convolutional layers were used to obtain long-term dependencies of sequences.

People only need to pay attention to certain elements of the visual input in order to understand or recognize them since the attention mechanism is developed from the human visual attention influence mechanism. In recent years, natural language processing field has also begun to combine the attention mechanism with tasks such as machine translation, so that the model can automatically assign weights to items during coding. Vaswani *et al.* [13] proposed a new structure Transformer based on self-attention mechanism, which can replace sequential neural network structures such as RNN to process sequence information. Many works have begun to try to apply the powerful feature extraction capabilities of Transformer to recommender systems. Kang *et al.* [20] proposed a sequence recommendation model named SASRec, which uses a one-way Transformer layer to capture the hidden layer output corresponding to each item of the sequence. The results show that compared with the sequence model based on RNN, SASRec has better sequence information capture ability. Sun *et al.* [21] believe that the learning ability of the one-way model of SASRec is limited, and not all sequences are completely in line with the actual situation. Therefore, the author proposes the BERT4Rec sequence recommendation model, which uses a deep bidirectional self-attention mechanism to model user behavior sequences, and adopts the self-supervised learning method of Cloze task. Chen *et al.* [22] used Transformer to capture the sequence signal in the historical interaction sequence of user items, and combined other features of the item to predict the click probability of the target item.

The main difference between our proposed approach and the above mentioned work are summarized in Table 1.

TABLE 1. Comparison of our proposed model with the proposed approaches of the literature.

Proposed approaches	Approach features			
	Auxiliary information of the sequence	Changes in user interests	User behavior changes	Model expansion
[21], [22]			x	
[11], [13]	x			x
[15]–[17], [19], [20]		x		
[1]	x	x		
[18]	x			
Our proposed approach	x	x	x	x

III. SELF-ATTENTION MASK LEARNING MODEL

A. PROBLEM DESCRIPTION

First, let user $\mathcal{U} \in \{u_1, u_2, \ldots, u_N\}$, item $\mathcal{I} \in \{i_1, i_2, \ldots, i_M\}$, category $\mathcal{C} \in \{c_1, c_2, \ldots, c_K\}$, the history of user $\mathcal{U}$ purchase record $S_u = \{s_u^1, s_u^2, \ldots, s_u^T\}$, where $s_u^t = (i_m, c_q)_u, (i_m, c_q)_u$ represents a 2-tuple composed of the item i_m purchased by the user $\mathcal{U}$ at time t and the category c_q to which it belongs. Given a historical sequence S_u, where the items in S_u are arranged in chronological order to predict the next possible item. The probability value of user $\mathcal{U}$ buying each item i_m of category c_q at time $t+1$ can be obtained:

$$P(s_u^{t+1} = (i_m^{t+1}, c_q^{t+1})_u | S_u) \tag{1}$$

Sort the probability values corresponding to each item from large to small, and generate a Top-k candidate set for the user according to the selection of the k value.

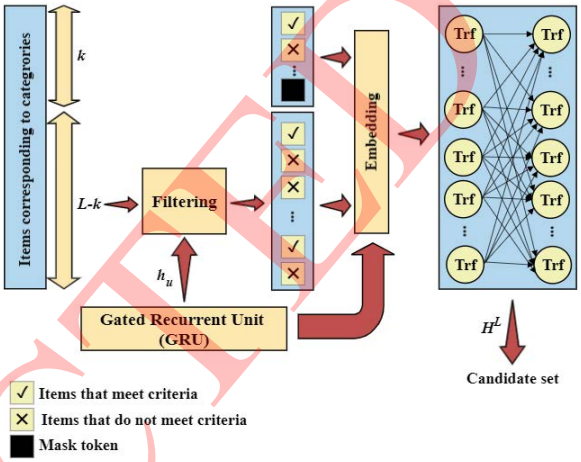


FIGURE 1. Model overview.

B. MODEL OVERVIEW

In this paper, a model called GAT4Rec is introduced to model the target type. The model is based on Transformer [13] and Gated Recurrent Unit (GRU) [23]. As shown in Figure 1, the entire model is divided into user interest coding layer, gated filtering layer, and Transformer layer. In the user interest coding layer, the category of the subsequence S_u^s consisting of the nearest k items is selected as its input. According to the research in the literature [8], it can be known that users' interest tendencies are mostly concentrated on recently purchased products. For example, when choosing a mobile phone, they usually tend to look at other mobile phones as a reference. In order to take into account the diversity and generality, this model selects the category of the purchased item as the user's interest tendency representation. According to the feature representation of the user's interest tendency, in the gated filtering layer, we can filter out the historical items that can support the current vector, that is, the embedding of the corresponding category of the historical item and the learned user's interest tendency representation are more isotropic in the vector space. It can be seen that the filtered item sequence is input to the coding layer composed of L layers of Transformers, each layer is composed of H Transformers, and the layers are fully connected to each other. Unlike RNN, Transformer can guarantee parallel training of the entire model. Each layer is equivalent to re-encoding the input items. Finally, the output of the corresponding items of the mask tokens is used to predict the final recommended product set.

C. USER EMBEDDING LAYER

Considering the concealment of user ids (anonymous users), modeling the id directly may have a negative effect on the model due to null values, so we need to use other information to help build the user representation. The category of the product is easier to obtain than other features. While using it to represent the user's interest, we also make the representation as the user's embedding. Because compared with the user embedding learned from the product sequence, the feature representation based on category information can ensure that the number of users is small, which is beneficial to the scalability of the model and the diversity of recommended products. Here, considering that the length of k is limited and GRU is more efficient than LSTM in training, we use GRU, a variant of RNN, to model the category sequence to obtain the current user representation.

Specifically, we have a sequence $S_u = \{s_u^1, s_u^2, \ldots, s_u^T\}$, and its corresponding category $C = \{c_{T-k+1}, c_{T-k+2}, \ldots, c_T\}$, where c_i represents the category corresponding to the i^{th} item in the sequence S_u. Through the mapping transformation, we can get the embedding of the category sequence $E_c = \{e_c^{T-k+1}, e_c^{T-k+2}, \ldots, e_c^T\}$, where $e_c^i \in R^c$. We input this sequence of categories into a structure composed of GRUs to obtain the embedded representation of the user [24]. The node update for GRU looks like this:

$$r_t = \sigma(W_r \cdot [h_{t-1}, e_c^t]) \tag{2}$$

$$z_t = \sigma(W_z \cdot [h_{t-1}, e_c^t]) \tag{3}$$

$$\tilde{h}_t = \tanh(W_{\tilde{h}} \cdot [r_t \odot h_{t-1}, e_c^t]) \tag{4}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{5}$$

where r_t regulates how much of the past state information is forgotten, z_t controls the amount of information from the previous state that is incorporated into the current state, h_{t-1} is the output state at time $t-1$, W_r, W_z, W_h are weight matrices to be learned, $\odot$ represents the Hadamard product, and σ represents sigmoid activation function. Finally, we use the hidden layer representation obtained by the k GRU unit as the user's interest tendency representation, and its expression is:

$$h_u = h_k \tag{6}$$

where $h_u \in R^{d_c}$ is the potential interest representation of the user. The network parameters of the GRU are trained by feeding h_i to the multilayer perceptron and softmax operation. Considering the complexity of the overall network, the training time of the network needs to be reduced to prevent it from overfitting.

D. GATED FILTER LAYER

With the passage of time, the user's historical purchase items gradually increase. If the full sequence is modeled, many historical items cannot be well matched with the user's current interests and preferences, which will have side effects on the recommendation effect. So we need to filter the historical sequence to find the item that matches the current interest. In recent years, attention mechanisms [13] have been widely used in sequence modeling, such as self-attention, soft-attention, and hard-attention. Hard-attention is usually used to emphasize that a certain item in the sequence is very important. The GAT model draws on the idea of hard-attention and needs to find the more important items in the sequence, which are in line with the current user's interest tendency. For the obtained hi, we can filter out the eligible history options, where $index(\text{item}) \leq T - k$, that is, filter the items whose index is less than $T - k$. The historical sequence $S_u^l = \{s_u^1, s_u^2, \ldots, s_u^{T-k}\}$ corresponds to the category sequence $C_u^l = \{c_1, c_2, \ldots, c_{T-k}\}$, according to the mapping table we can get the corresponding Category word embedding $E_c^l = \{e_c^1, e_c^2, \ldots, e_c^{T-k}\}$. Cai [25] proposed to select the items corresponding to the result values of Top-k $softmax(e_c^i \cdot h_i)$ as the input of the model, but this method has limitations: 1. There may be more than k items in the historical sequence and user interest tendencies are represented in the potential Approach in the direction in space; 2. The length of the historical sequence may be less than k. Here we set a hyperparameter λ with the condition: $sigmoid(h_u^T \cdot e_c^i) > \lambda$. We let the subsequence of item items whose calculation result is greater than λ be the input to the model.

E. PRODUCT EMBEDDING LAYER

The number of products is often in the thousands, and one-hot encoding is used to label the product ids in the integer field, but this will greatly increase the number of parameters of the model. Usually, we map the one-hot encoding to a low-dimensional embedding vector, which not only achieves the purpose of dimensionality reduction, but also improves the representation learning ability of the model. Let $|V|$ be the size of the dataset, and d the dimension of the embedding vector, then $|V| \times d$ is the number of parameters that the model should learn. Here we adopt embedded factorization [26] to further minimize the number of parameters of the model, which is beneficial to the expansion of the model. In simple terms, embedded factorization adds another layer to the original mapping matrix. Assuming that the newly added embedding dimension is E, the amount of parameters is $|V| \times E + E \times d = E \times (|V| + d)$. If $E \ll d$, the amount of parameters decreases significantly. Therefore the item embedding can be expressed as $E_p = \{e_p^1, \ldots, e_p^{|V|}\} \in R^{|V| \times d}$.

The sequence order of products can reflect changes in user behavior, but the Transformer module does not come with timing information like a recurrent neural network, so additional sequence coding is required to ensure that the model can learn the importance of the position. The position encoding $P = \{p_1, \ldots, p_L\}$, L is the maximum length of the sequence, here we choose to use the model to learn the encoding.

In order to make full use of the hidden layer information in the user representation, this paper chooses to add this vector to the Transformer layer to learn together. The combination

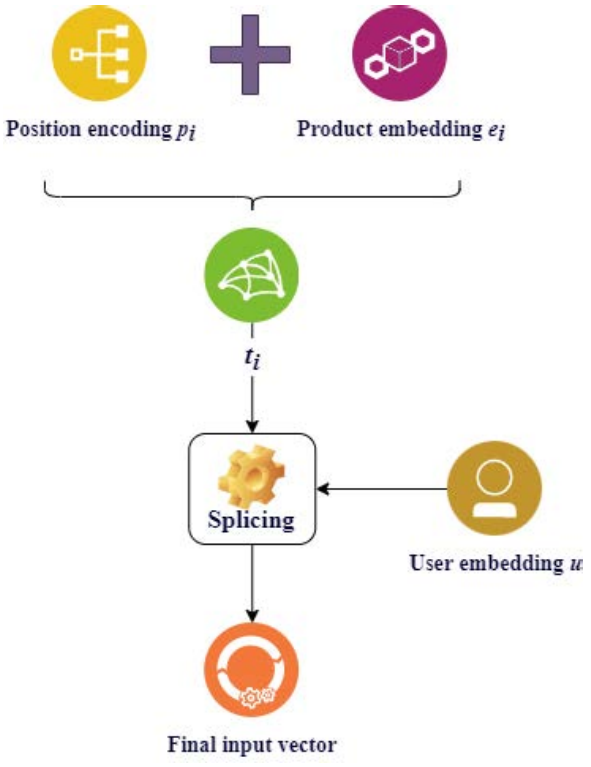


FIGURE 2. Long-term dependence on the input of the encoding layer.

process of vectors is shown in Figure 2. First, add the product embedding e_i and the position code p_i to get t_i, then splicing t_i with the user embedding u and obtain the final input vector into the model through linear transformation.

F. TRANSFORMER LAYER

Learning from the sequence to the transfer paradigm between products is the primary purpose of sequence recommendation. Compared with the feature extraction network based on RNN, Transformer can not only perform better in processing long sequence tasks, but also its parallel processing ability ensures its performance. Training speed is better than RNN. This model's core structure, the Transformer layer, is made up of three layers: a normalization layer, a feed-forward network layer, and a multi-head attention layer. The key to this module lies in the multi-head attention layer, which is based on the self-attention mechanism and assisted by multiple heads to learn representation vectors in different subspaces. The basic flow of the attention mechanism [27] is shown in Equation 7:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right) \cdot V \tag{7}$$

Among them, Q, K, and V represent query, key, and value, respectively, that is, the degree of association between query and key determines the weight of the current value. Due to the self-attention mechanism adopted in this paper, Q, K, V are generated from the same input, $Q = HW^Q$, $K = HW^K$, $V = HW^V$. Multi-head attention uses multiple self-attention modules to learn different hidden layer representations, which are specifically defined as follows:

$$\text{head}_i = \text{Attention}(H^{L-1}W_i^Q, H^{L-1}W_i^K, H^{L-1}W_i^V) \tag{8}$$

$$H^L = [\text{head}_1; \text{head}_2; \ldots; \text{head}_{n-1}; \text{head}_n]W^0 \tag{9}$$

where H^L is the output of the hidden layer representation of the L^{th} layer, each head can calculate its corresponding attention weight distribution, and then generate a new parameter matrix. where $W_i^Q \in R^{d\times d/n}$, $W_i^K \in R^{d\times d/n}$, $W_i^V \in R^{d\times d/n}$ are independent weight matrices that are not shared by each head. Finally, the obtained n heads are spliced together, and the multi-head attention output of the L layer is obtained through the weight matrix transformation.

The purpose of the feedforward network layer (FFN) is to enable the model to have nonlinear modeling capabilities, using the Gelu activation function [28]. Compared with Relu, Gelu activation function introduces random regularization, and the convergence speed is improved. Its activation expression is shown in Equation 10 and Equation 11:

$$\text{FFN}(x) = \text{Gelu}(xW^{f1} + b_{f1})W^{f2} + b_{f2} \tag{10}$$

$$\text{Gelu}(x) = x\Phi(x) \tag{11}$$

where Φ is the cumulative distribution function of the standard Gaussian distribution, $W^{f1} \in R^{d\times 4d}$, $W^{f2} \in R^{4d\times d}$, $b_{f1} \in R^{4d}$, $b_{f2} \in R^d$ are the learned parameters and are shared in each Transformer.

In the normalization layer (LN), this paper uses the residual network to ensure the learning effect of the deep network parameters. Combined with the multi-head attention layer (MH) and the feedforward network layer, the overall process of the Transformer is as follows:

$$\text{AN}_1^L = \text{LN}(H^{L-1} + \text{MH}(H^{L-1})) \tag{12}$$

$$\text{FFN}(\text{AN}_1^L) = \text{Gelu}(\text{AN}_1^L W^{f1})W^{f2} + b_{f2} \tag{13}$$

$$\text{AN}_2^L = \text{LN}(\text{FFN}(\text{AN}_1^L) + \text{AN}_1^L) \tag{14}$$

$$H^L = \text{AN}_2^L \tag{15}$$

The entire long-term dependency encoding layer is composed of many Transformers, and the parameters between the layers are shared, which greatly reduces the overall parameter amount of the model and provides the possibility for model expansion.

Finally, the Transformer output H_T^L corresponding to the end of the sequence is selected as the characterization vector of the product for subsequent recommendation:

$$H^O = H_T^L \tag{16}$$

Algorithm 1 describes the execution flow of GAT4Rec:

G. MODEL TRAINING

Due to the bidirectional modeling characteristics of the model, it needs to be trained by mask learning. The general idea is that for the sequence $S = \{s_1, s_2, \ldots, s_n\}$, then sample $\rho\%$ and do for mask operation, 80% in the sampling is replaced by *mask_token*, 10% is replaced by any item in the product set, and 10% remains unchanged.

Algorithm 1 Sequence Recommendation Model Based on Self-Supervised Mask Learning

Input: k: User interest network input length; λ: gating signal; T: Training set (X, C), X is the user, C is the corresponding category of the item in the sequence; η: learning rate; E: number of learning iterations; B: Number of batches; L: maximum length of sequence

Output: The Parameters of Model θ_m //Model weights

```
Initialize parameters of model θ_m;
for iter in E do
  for b in b_1 ··· b_{|T|/B} do
    h_u ← GRU(I_{l−k}, ··· I_L)
    for c_i in C_1 ··· C_{L−K} do
      if softmax(h_u^T · c_i) < λ then
        Remove I_i from sequence;
      end if
    end for
    Padding and rearrange E_mask ← f(E, ρ) //Add mask_token according to ρ;
    for e_i in e_1 ··· e_L do
      e_i ← map(h_u ⊕ e_i) + p_i
    end for
    O ← Transformer(E_mask, L) //L-Layer converter encoding processing
    argmax_{O_{1:|O_mask|}} Σ_{i=1}^{|O_mask|} P(L_i|O_i) ;
    Update θ_m;
  end for
end for
return θ_m;
```

After the mask operation, the model input becomes $S_m = \{s_1, [mask\_token], [mask\_token], s_4, \ldots, s_n\}$. The purpose of this model is to predict the original corresponding item of the masked item through the obtained output feature. Therefore, the learning of the model can be regarded as a multi-classification task. This paper chooses to use the softmax function to obtain the probability value of each item in the product set. Considering that the model only focuses on whether the predicted item is a true label, the training on the encoding layer adopts the cross-entropy loss function:

$$L_t = \frac{1}{|\text{sub}(S_m)|} \sum_{j=1}^{N_s} \sum_{i \in \text{sub}(S_m)} -\log P(i = i' | H_i^O) \quad (17)$$

where $|\text{sub}(S_m)|$ represents the number of items operated by mask, N_s is the total number of input sequences, and i' is the label value of item i, that is, the corresponding item before mask operation. In addition, the GRU module needs to be trained to predict the next category information. The category label may contain multiple categories, that is, this type of task belongs to multi-label classification, so the binary cross entropy loss function is used:

$$L_c = -\sum_{j=1}^{N_s} \sum_{i=1}^{N_c} y_i' \log y_i = -(1 - y_i') \log(1 - y_i) \quad (18)$$

where N_c is the number of categories contained in the training samples. Therefore, the loss function consists of the loss of the coding layer and the loss of category prediction. α is the weight value of the two loss functions, which can be adjusted according to the actual training situation. The loss function expression is as follows:

$$L = \alpha L_t + (1 - \alpha) L_c \quad (19)$$

IV. EXPERIMENTAL RESULTS

This section provides the statistics of the dataset used in the experiments, the experimental metrics, and the specific parameter settings of the experiment. Then we provide the comparison results with some state-of-the-art approaches, investigate the impact of important hyperparameters on the model, and conduct ablation experiments to understand how much each module affects model learning.

A. EXPERIMENTAL PLATFORM

This experiment is carried out on two platforms of personal computer and server at the same time. The hardware environment of personal computer is as follows:

- Processor: Intel i7 9700 CPU @ 4.70GHz
- Memory: DDR4 24G
- Development environment: Anaconda3 + PyTorch + Pandas

The server hardware environment is as follows:

- Processor: Intel Xeon Silver 4116 CPU @ 2.10GHz
- Memory: DDR4 128G
- Development environment: Anaconda3 + PyTorch + Pandas

B. DATASET AND PREPROCESSING

For the design of the model, this paper selects four real datasets: MovieLens-1M (ML-1M), MovieLens-20M (ML-20M), Taobao, Taobao_m, and the statistical information is as follows:

TABLE 2. Dataset statistics.

Dataset	#Users	#Items	#Categories	#Actions	Avg. Act/User
ML-1M	6040	3706	18	1.0M	165.5
ML20M	138493	26744	19	20.M	144.4
Taobao	6469	45393	2034	0.4M	66.2
Taobao_m	17719	36716	1702	0.8M	44.5

ML-1M and ML-20M are two public movie sequence datasets [29] containing 1 million and 20 million user-item interactions (#Actions), respectively. These datasets provide a set of movie ratings from the MovieLens website that offers movie recommendations. Taobao and Taobao_m are

both subsets of the Tianchi competition dataset. Taobao is a randomly selected sequence of 6469 users, and retains user-item interactions that appear less frequently. Taobao_m represents the modified dataset, which extracts the top 20k user interaction sequences in the original sample, and filters out interactions whose user sequence length is less than 20 and the total number of item and category occurrences is less than 10. MovieLens datasets have fewer categories and a longer average interaction length of user items. Taobao's two sub-datasets have more categories, and the user's interaction sequence length is much smaller than the other two groups. This experiment retains four groups of information in the data set: user serial number, product serial number, timestamp, and product category.

Since there is textual information in the dataset, it needs to be converted into numerical information for model learning. The two datasets of MovieLens contain 18 and 19 categories respectively, and some movies have multiple categories. To speed up model data preprocessing, category information is converted into multihot vectors and saved in pkl files. Taobao's two datasets both contain thousands of categories, and there is a one-to-one relationship between products and categories, but if it is converted to a onehot vector for saving, it will take up a lot of space, so the category information is saved as the corresponding Id.

In addition, the user serial numbers, product serial numbers, and category serial numbers of these four datasets are discrete and discontinuous, and they need to be compressed into discrete continuous values. Finally, the data set is sorted according to the (user, timestamp) tuples from small to large.

C. EVALUATION METRICS

In order to confirm the effectiveness of the sequence recommendation model, this experiment uses Recall, nDCG (normalized Discounted Cumulative Gain), and MRR (Mean Reciprocal Rank) as the evaluation metrics of the model on recall and ranking tasks.

Each user has only one ground-truth item, where Recall refers to whether the ground-truth is in the candidate list, if it exists, then Recall is 1, otherwise it is 0. The expression for Recall R is as follows:

$$\mathrm{R@}K_j = \begin{cases} 0 & \text{ground-truth} \notin C_{1:K} \\ 1 & \text{ground-truth} \in C_{1:K} \end{cases} \tag{20}$$

$$\mathrm{R@}K = \frac{1}{K}\sum_{j=1}^{K} \mathrm{R@}K_j \tag{21}$$

where $C_{1:K}$ represents the top-K items in the candidate set. MRR and nDCG consider the effect of ranking, MRR is the average of the reciprocal ranking of ground-truth, and nDCG consists of two parts: DCG and IDCG. DCG is a combined version of cumulative gain (CG) and position weights, and IDCG refers to the ideal maximum value of DCG. In the sequence recommendation task, since the correlation of the recommendation results is all 1, the IDCG is also equal to 1, and the nDCG only focuses on the position of the ground-truth in the candidate set. The formulas for MRR and nDCG are as follows:

$$\mathrm{nDCG} = \frac{1}{K}\sum_{j=1}^{K} \frac{1}{\log_2(I_j + 1)} \tag{22}$$

$$\mathrm{MRR} = \frac{1}{K}\sum_{j=1}^{K} \frac{1}{I_j} \tag{23}$$

I_j refers to the index position of the j^{th} prediction (starting at 1). Compared with Recall, the two metrics, MRR and nDCG, focus more on the order of ground-truth in the candidate set. This experiment uses nDCG@K, R@K, MRR, where $K = \{1, 5, 10\}$. In order to avoid huge computational consumption, this experiment adopts the same strategy in [20] and [21], randomly selecting 100 uninteracted items as negative samples and forming the candidate set together with ground-truth.

D. BASELINE DESCRIPTION

To verify the effectiveness of this method, five representative models were selected as benchmarks:

- PopRec: This method simply ranks based on popularity, which is based on the amount of user interaction with the item.
- NCF: Model users and items using multilayer perceptrons instead of matrix factorization to learn interaction probability values for user items.
- GRU4Rec: Models a session-based sequence using the GRU module to predict the next item as its training target.
- SASRec: A one-way self-attention-based Transformer module is used to capture the sequence information of user behavior, and its effect is better than the sequence model based on RNN/CNN.
- BERT4Rec: This model uses a feature-representation-based bidirectional Transformer module at the end of the sequence to recommend the next step, which has better information acquisition ability than the unidirectional model.

PopRec is the mainstream comparison model, and NCF, GRU4Rec, SASRec, and BERT4Rec are the deep learning models proposed in recent years. Both SASRec and Bert4Rec use the source code provided by the author, and the corresponding PyTorch version used by NCF and GRU4Rec. In addition to the PopRec model, we adopt latent dimensions $d \in \{32, 64, 128, 256\}$, learning rate $lr \in \{0.0005, 0.001, 0.001, 0.01, 0.1, 1\}$, $dropout \in \{0, 0.1, 0.2, 0.3, 0.4, 0.5\}$, and other hyperparameters are set to the default configuration recommended by the author. The data presented in this chapter are the optimal values obtained by each model.

E. PARAMETER SETTINGS

This experiment was completed with PyTorch. To make sure that the experiment's results are fair, some super participants

are consistent with SASRec and Bert4Rec. Adam is selected as the model optimizer, the learning rate of Transformer and GRU are is 0.001 and 0.01, respectively. Also, batch_size is 64, learning epoch is equal to 100, dropout is 0.1, max_len is 100 for MovieLens and 50 on Taobao dataset. Table 3 presents the hyperparameters that need to be experimentally verified in this experimental part.

TABLE 3. Experimental parameter values.

Parameter name	Experimental value range
Hidden_dimension_GRU(d_G)	$[8, 16, 32, 64, 128, 256, 512]$
Intent_num(k)	$[1, 2, 3, 4, 5, 6, 7, 8]$
Threshold(λ)	$[0.1, 0.2, 0.4, 0.6, 0.8, 0.9, 1.0]$|| $[0.01, 0.03, 0.05, 0.1, 0.2, 0.4, 1.0]$

F. EXPERIMENTAL RESULT ANALYSIS

1) OVERALL COMPARISON

A comparison report of our best results with the 5 baselines is shown in Table 4. The last item in the table describes the improvement rate of GAT4Rec compared to each best metric. From the table we can conclude that PopRec, the method for recommending based on popularity, performs the worst. NCF and GRU4Rec outperform PopRec on all datasets, where GRU4Rec>NCF, indicating that RNN is better than multi-layer perceptron in capturing sequence information. Furthermore, the performance of SASRec and BERT4Rec is better than NCF and GRU4rec, which also demonstrates that the ability of the attention-based Transformer module to extract sequence information is far superior to the RNN or its variant modules. Among them, Bert4rec surpassed SASRec in all indicators on the two datasets of MovieLens, and some indicators were worse than SASRec on the two datasets of Taobao. The small number of masks leads to the training effect of the model is not as good as the effect of long sequences.

GAT4Rec outperforms other methods in almost all metrics, and these improvements may come from:

1) Utilization of metadata: Sas4rec and Bert4Rec only use the interaction sequence between users and items, and do not model the concerns of each user, which will undoubtedly bring a certain degree of loss. GAT4Rec models the categories of recently interacted items, and combines the obtained user interest hidden layer variables with the user sequence to make recommendations that are more in line with the current user interests.
2) Relevance: Models and algorithms determine the lower bound of learning, while data determine the upper bound of the model. Most of the existing models use the entire user interaction sequence as the learning sample of the model, but not all items can improve the current recommendation. Usually people's attention is always related to the recent interactions. Using this information to filter out "irrelevant items in the sequence can help the model to better extract sequence information.

2) THE EFFECT OF HYPERPARAMETERS ON THE MODEL

This section is mainly to verify the influence of some important hyperparameters on the model. Here, the hidden layer dimension (d_G) of the GRU, the window size (k) of the user embedding, and the gating signal (λ) are selected.

In this model, k is an important hyperparameter that determines how many recent interactions are involved in modeling user interests. Let $\lambda = 0.4$ on the MovieLens datasets and $\lambda = 0.003$ on the Taobao datasets, and the rest of the parameters are set to default values. From Figure 3, it can be seen that the initial experimental effect gradually increases with the increase of k, and the model's performance exhibits a declining trend when k reaches a specific value. The possible reason is that the initial number of categories is small, and the model filters more input sequences, so that the sequence information obtained by the model is lost. As k increases, the model learns a better sequence substructure. However, when k is too large, the user embedding contains too much information, which may lead to over-fitting, causing the model's performance to decline as a result. Specifically, on the two datasets of MovieLens, when $k = 4$, the model can perform at its peak, and then k shows a downward trend as a whole. On Taobao's two datasets, the corresponding k value is 6.

Figure 4 shows the effect of λ on each dataset. λ is one of the most influential parameters in the whole model, which can directly act on the user sequence and affect the input of the model. When $\lambda = 0$, the model cannot filter the sequence information, but only uses the user's interest tendency. Considering the difference in the number of categories of the datasets, the values of λ are different on Taobao and MovieLens datasets. It can be observed from Figure 4 that when $\lambda = 0.4$, the nDCG@10 obtained by the ml1m dataset and the ml20m dataset are 0.6507 and 0.8575, respectively, which are the optimal λ values for these two datasets. In addition, when $\lambda = 0.03$, the optimal nDCG@10 values of Taobao dataset and Taobao_m dataset are 0.5376 and 0.5357, respectively. As λ gradually increases, the performance of the model decreases greatly, which may be because the entire sequence is basically regarded as noise and filtered out, the loss of sequence information is too much, and the learning ability of the model decreases accordingly.

The influence of the hidden layer dimension of user embedding is shown in Figure 5. The hidden layer dimension d_G of nDCG@10 and R@10 increases sequentially from 8 to 512. It can be observed that when the hidden layer dimension is small, nDCG@10 is in four The data sets can achieve better results. With the increase of d_G, the overall performance of the model shows a downward trend. The possible reasons are: (1) the number of product categories is limited, and the information learned by the larger-dimensional model is more sparse; (2) The input of the Transformer is the vector of the splicing and mapping of the product embedding and the user embedding. The larger dimension of the user embedding may bias the learning of the model to the user embedding side, thus affecting the modeling ability of the product sequence.

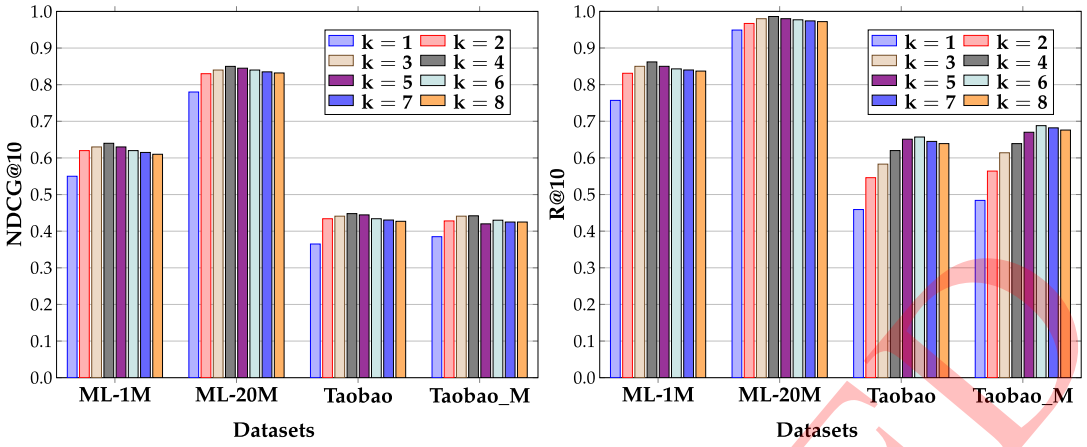


FIGURE 3. The effect of hyperparameter *k* on experimental results.

TABLE 4. Experimental index results.

Dataset	Metric	PopRec	NCF	GRU4Rec	SASRec	BERT4Rec	GAT4Rec	Improvement
ML-1M	nDCG@5	0.0482	0.3637	0.4417	0.5573	*0.5834*	**0.6225**	6.70%
	nDCG@10	0.0703	0.4167	0.4852	0.5883	*0.6152*	**0.6507**	5.77%
	MRR	0.0496	0.3484	0.4235	0.5227	*0.5557*	**0.5938**	6.86%
	R@1	0.0141	0.1934	0.2675	0.3662	*0.4143*	**0.4552**	9.87%
	R@5	0.0547	0.5242	0.5751	0.7248	*0.7292*	**0.7646**	4.85%
	R@10	0.1027	0.6881	0.7125	0.8202	*0.8277*	**0.8517**	2.89%
ML-20M	nDCG@5	0.0744	0.6399	0.6618	0.8094	*0.8332*	**0.8453**	1.45%
	nDCG@10	0.1074	0.6738	0.7025	0.8127	*0.8452*	**0.8566**	1.35%
	MRR	0.0655	0.6008	0.6359	0.7841	*0.8055*	**0.8202**	1.82%
	R@1	0.0161	0.4383	0.5677	0.7013	*0.7051*	**0.7278**	3.22%
	R@5	0.0892	0.7142	0.8065	0.8579	*0.9343*	**0.9478**	1.44%
	R@10	0.1606	0.8179	0.8372	0.8977	*0.9717*	**0.9787**	0.72%
Taobao	nDCG@5	0.0078	0.3888	0.3985	0.4548	*0.4549*	**0.5132**	12.82%
	nDCG@10	0.0089	0.4161	0.4473	0.4827	*0.4741*	**0.5386**	11.58%
	MRR	0.0056	0.3826	0.3961	0.4531	*0.4612*	**0.5163**	11.95%
	R@1	0.0012	0.2869	0.3152	0.3657	*0.4044*	**0.4991**	23.42%
	R@5	0.0109	0.4784	0.5057	0.5371	*0.5024*	**0.5731**	6.70%
	R@10	0.0134	0.5633	0.5826	0.6237	*0.5619*	**0.6517**	4.49%
Taobao_m	nDCG@5	0.0087	0.3514	0.4205	0.4733	*0.4953*	**0.5082**	2.60%
	nDCG@10	0.0113	0.3902	0.4598	0.5071	*0.5263*	**0.5357**	1.79%
	MRR	0.0074	0.3451	0.4134	0.4644	*0.4982*	**0.5074**	1.85%
	R@1	0.0021	0.2284	0.3374	*0.4549*	0.4132	**0.4435**	-2.51%
	R@5	0.0099	0.4639	0.5015	*0.5808*	0.5723	**0.5895**	1.50%
	R@10	0.0158	0.5856	0.6176	*0.6854*	0.6691	**0.6827**	-0.39%

G. ABLATION EXPERIMENTS

Ablation experiments are performed on the GAT4Rec model to understand how each model component affects the experimental results. To more clearly and intuitively see how each element affects the overall results, only the nDCG@10 and R@10 indicators are used here. The experimental objects include user embeddings (UE), gated filtering layers (FL), and end mask processing (MP).

The user embedding is also added to the Transformer module as the gating signal for joint learning with the sequence embedding. According to Table 5, it can be concluded that the user embedding is removed and only the item embedding is used for learning. On the four datasets, the average decrease of nDCG@10 is 1.84%, and the average decrease of R@10 is 0.94%. This shows that the user embedding can provide additional hidden layer information, reflecting the current interest tendency of the user, so the model can better learn the sequence representation vector for the next recommendation.

The removal of the gated filter layer has a greater impact on the three datasets ML-1M, ML-20M, and Taobao, and Taobao_m is relatively less affected. This shows that some noise items have been removed from the pre-filtered dataset, and the gated filtering layer can help find a better sequence substructure. In addition, it can be known that on the two datasets of MovieLens, the decline ratio of nDCG@10 is greater than that of R@10, and on the two datasets of Taobao, the impact of R@10 is greater than that of nDCG@10, that is, the MovieLens datasets. The recall task is more sensitive

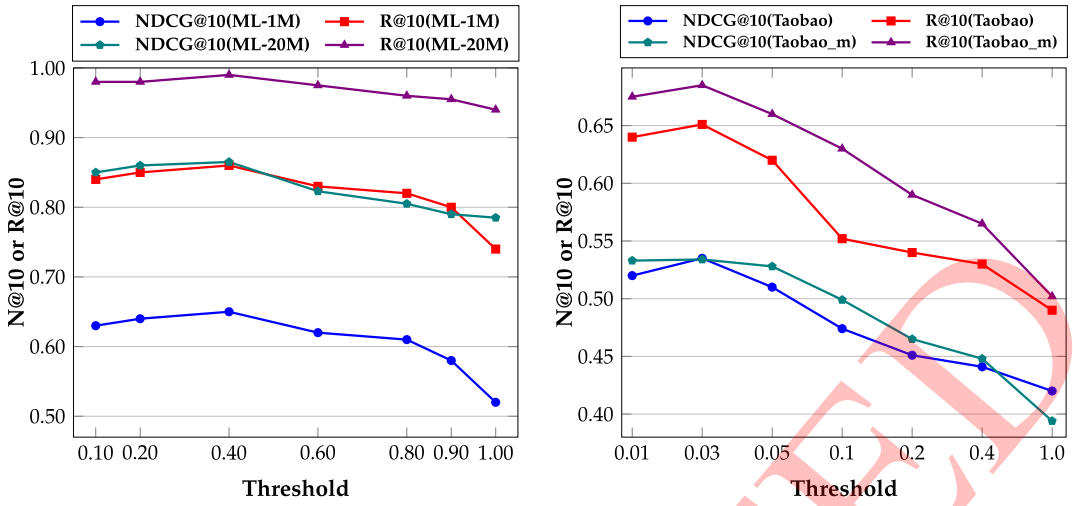


FIGURE 4. The effect of hyperparameter λ on experimental results.

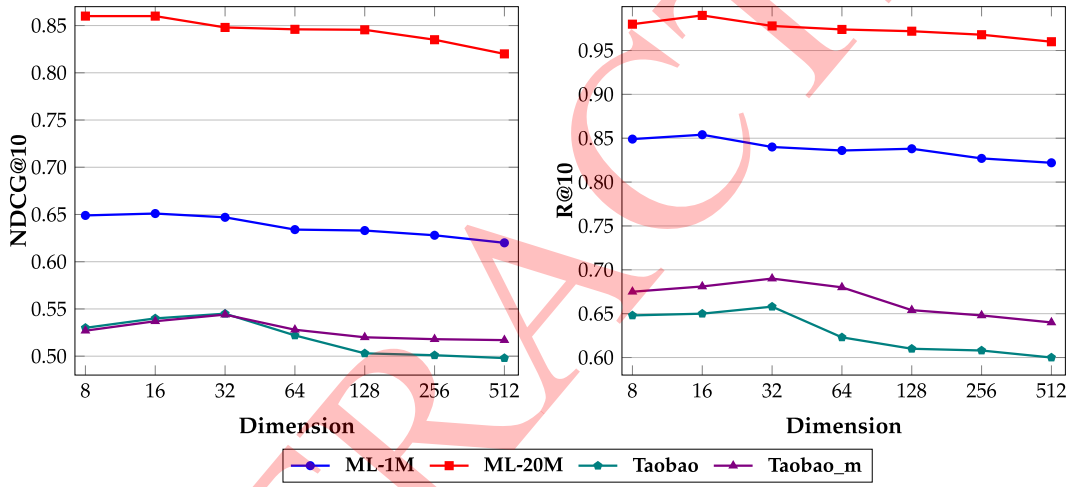


FIGURE 5. The effect of hyperparameter d_G on experimental results.

TABLE 5. Ablation experiment results.

Architecture	Metrics	Dataset			
		ML-1M	ML-20M	Taobao	Taobao_m
Default	nDCG@10	0.6507	0.8566	0.5386	0.5357
	R@10	0.8517	0.9787	0.6517	0.6827
Without UE	nDCG@10	0.6331	0.8526	0.5228	0.5289
	R@10	0.8452	0.9723	0.6388	0.6801
Without FL	nDCG@10	0.6301	0.8491	0.5137	0.5304
	R@10	0.8477	0.9769	0.6136	0.6759
With MP	nDCG@10	0.6281	0.7407	0.5346	0.5294
	R@10	0.8423	0.9262	0.6509	0.6744

than the ranking task, and the Taobao dataset is the opposite. This may be related to the length of the sequence.

The GAT4Rec model is a bidirectional Transformer based mask model that adds *mask_token* to the end of the sequence in the validation phase to predict the corresponding item. Since the task of sequence recommendation is to forecast whether the next item is ground-truth, the item corresponding to the *mask_token* at the end is calculated. Add *mask_token* to the end of each sequence as a new data loading method. According to the results in the table, it can be known that the effect of the model does not increase but decreases. This shows that the mask model can learn more effective representation vectors by using random processing preprocessing, and the end mask processing leads to overfitting in model learning.

V. CONCLUSION

A sequence recommendation model GAT4Rec based on unsupervised mask learning is proposed in this paper, in order to address the problem of insufficient information utilization and neglect of user interest tendency in sequence recommendation. The user's interest embedding is obtained through GRU and used as the gate of the gated filtering layer. Control

signals to filter out the categories with higher correlation and embed the corresponding items. The Transformer layer is learned in an unsupervised mask way, and a self-attention mechanism is used between its module units to learn the correlation information between items. On four public datasets, subsequent tests compare the performance of GAT4Rec with various baseline models, and the results demonstrate that the model outperforms nearly all baseline models. Furthermore, the influence of important hyperparameters on the model is studied, and ablation experiments of some components are carried out.

Although the proposed recommendation algorithm in this paper has improved in performance indicators, there are still some shortcomings that need to be improved. GAT4Rec does not fully utilize the sequence information, and truncates the input sequence exceeding the length according to the limited maximum sequence length, which may cause loss of information and thus affect the recommendation effect. Future work will try to learn from the strategy of XLNET [30], select a node to save the segment features of the previous sequence, and ensure that the entire sequence is connected.

REFERENCES

[1] J. Tang and K. Wang, "Personalized top-N sequential recommendation via convolutional sequence embedding," in *Proc. 11th ACM Int. Conf. Web Search Data Mining*, Feb. 2018, pp. 565–573.

[2] S. Bai, J. Zico Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," 2018, *arXiv:1803.01271*.

[3] Q. Liu, S. Wu, D. Wang, Z. Li, and L. Wang, "Context-aware sequential recommendation," in *Proc. IEEE 16th Int. Conf. Data Mining (ICDM)*, Dec. 2016, pp. 1053–1058.

[4] X. Chen, H. Xu, Y. Zhang, J. Tang, Y. Cao, Z. Qin, and H. Zha, "Sequential recommendation with user memory networks," in *Proc. 11th ACM Int. Conf. Web Search Data Mining*, Feb. 2018, pp. 108–116.

[5] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized Markov chains for next-basket recommendation," in *Proc. 19th Int. Conf. World Wide Web (WWW)*, 2010, pp. 811–820.

[6] H. Zhang, W. Ni, X. Li, and Y. Yang, "Modeling the heterogeneous duration of user interest in time-dependent recommendation: A hidden semi-Markov approach," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 48, no. 2, pp. 177–194, Feb. 2018.

[7] E. Smirnova and F. Vasile, "Contextual sequence modeling for recommendation with recurrent neural networks," in *Proc. 2nd Workshop Deep Learn. Recommender Syst.*, Aug. 2017, pp. 2–9.

[8] F. Yu, Q. Liu, S. Wu, L. Wang, and T. Tan, "A dynamic recurrent model for next basket recommendation," in *Proc. 39th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2016, pp. 729–732.

[9] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, and J. Ma, "Neural attentive session-based recommendation," in *Proc. ACM Conf. Inf. Knowl. Manag.*, Nov. 2017, pp. 1419–1428.

[10] T. Donkers, B. Loepp, and J. Ziegler, "Sequential user-based recurrent neural network recommendations," in *Proc. 11th ACM Conf. Recommender Syst.*, Aug. 2017, pp. 152–160.

[11] F. Yuan, A. Karatzoglou, I. Arapakis, J. M. Jose, and X. He, "A simple convolutional generative network for next item recommendation," in *Proc. 12th ACM Int. Conf. Web Search Data Mining*, Jan. 2019, pp. 582–590.

[12] T. X. Tuan and T. M. Phuong, "3D convolutional networks for session-based recommendation with content features," in *Proc. 11th ACM Conf. Recommender Syst.*, Aug. 2017, pp. 138–146.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, Dec. 2017, pp. 5999–6009.

[14] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. Le, and R. Salakhutdinov, "Transformer-XL: Attentive language models beyond a fixed-length context," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 2978–2988.

[15] Y. Zhu, H. Li, Y. Liao, B. Wang, Z. Guan, H. Liu, and D. Cai, "What to do next: Modeling user behaviors by time-LSTM," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, Aug. 2017, pp. 3602–3608.

[16] V. Bogina and T. Kuflik, "Incorporating dwell time in session-based recommendations with recurrent neural networks," in *Proc. CEUR Workshop*, vol. 1922, 2017, pp. 57–59.

[17] B. Hidasi, M. Quadrana, A. Karatzoglou, and D. Tikk, "Parallel recurrent neural network architectures for feature-rich session-based recommendations," in *Proc. 10th ACM Conf. Recommender Syst.*, Sep. 2016, pp. 241–248.

[18] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," in *Proc. 4th Int. Conf. Learn. Represent. (ICLR)*, 2016, pp. 1–10.

[19] B. Hidasi and A. Karatzoglou, "Recurrent neural networks with top-K gains for session-based recommendations," in *Proc. 27th ACM Int. Conf. Inf. Knowl. Manag.*, Oct. 2018, pp. 843–852.

[20] W.-C. Kang and J. McAuley, "Self-attentive sequential recommendation," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2018, pp. 197–206.

[21] 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. Manag.*, 2019, pp. 1441–1450.

[22] Q. Chen, H. Zhao, W. Li, P. Huang, and W. Ou, "Behavior sequence transformer for E-commerce recommendation in Alibaba," in *Proc. 1st Int. Workshop Deep Learn. Pract. High-Dimensional Sparse Data*, Aug. 2019, pp. 1–4.

[23] K. Cho, B. V. Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*. 2014, pp. 1724–1734.

[24] F. Teng, Y. Song, G. Wang, P. Zhang, L. Wang, and Z. Zhang, "A GRU-based method for predicting intention of aerial targets," *Comput. Intell. Neurosci.*, vol. 2021, pp. 1–13, Nov. 2021.

[25] R. Cai, Q. Wang, C. Wang, and X. Liu, "Learning to structure long-term dependence for sequential recommendation," 2020, *arXiv:2001.11369*.

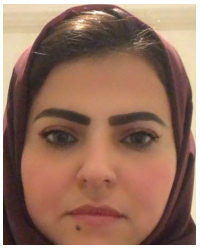
[26] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A lite BERT for self-supervised learning of language representations," 2019, *arXiv:1909.11942*.

[27] D. Chen, W. Hong, and X. Zhou, "Transformer network for remaining useful life prediction of lithium-ion batteries," *IEEE Access*, vol. 10, pp. 19621–19628, 2022.

[28] D. Hendrycks and K. Gimpel, "Gaussian error linear units (GELUs)," 2016, *arXiv:1606.08415*.

[29] F. M. Harper and J. A. Konstan, "The MovieLens datasets: History and context," *ACM Trans. Interact. Intell. Syst.*, vol. 5, no. 4, pp. 1–19, Jan. 2016, doi: 10.1145/2827872.

[30] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, "XLNet: Generalized autoregressive pretraining for language understanding," 2019, *arXiv:1906.08237*.



ABEER ALJOHANI received the B.Sc. degree in computer science from the College of Computer Science and Engineering, Taibah University, Medina, Saudi Arabia, in 2008, the M.Sc. degree in information technology for management from the Department of Computer Science, Coventry University, U.K., in 2011, and the Ph.D. degree in computer science from the University of Loughborough, in 2019. She is currently working as an Assistant Professor with the Applied Collage, Taibah University. Her research interests include sensor technology in e-health and smart health management, smart cities, data science, artificial intelligence, evolutionary computation, machine learning, deep learning, and pattern recognition.

MOHAMED ALI RAKROUKI received the B.S., M.S., and Ph.D. degrees in management information systems from the University of Tunis, Tunisia, in 2003, 2005, and 2010, respectively. From 2005 to 2008, he was a Lecturer with the Computer Science Department, University of Tunis. He has been working as an Assistant Professor with the Applied College, Taibah University, Saudi Arabia, since 2010. His research interests include machine scheduling, artificial intelligence, cloud computing, evolutionary computation, data science, and machine learning.

NAWAF ALHARBE received the B.Sc. degree in computer science from the College of Computer Science and Engineering, Taibah University, Medina, Saudi Arabia, in 2007, the M.Sc. degree in advanced computer science from the Department of Computer Science, University of Huddersfield, U.K., in 2011, and the Ph.D. degree in computer science from Staffordshire University, in 2015. He is currently working as an Associate Professor with the Applied College, Taibah University. His research interests include knowledge management systems in smart healthcare operations using emerging technology, such as RFID, ZigBee, the Internet of Things (IoT), cloud computing, artificial intelligence and machine learning, big data & data processing, knowledge engineering, knowledge harvesting, and healthcare applications.

REYADH ALLUHAIBI received the B.E. degree from Taibah University, in 2005, the M.Sc. degree from Tulsa University, in 2009, and the Ph.D. degree from Manchester University, in 2017. He was worked as a Lecturer with the Department of Computer Science, Taibah University, from 2009 to 2012, where he has been working as an Assistant Professor with the Department of Computer Science, since 2017. His research interests include machine learning, natural language processing, computational linguistics, computational semantics, knowledge representation, and temporal logic.