TaDGSR: A Time-adaptive Prediction Dynamic Graph Neural Network for Sequential Recommendation

Wanzhi Chen *, Zhuo Li

***Abstract*—Sequential recommendation aims to predict users' future behaviors by analyzing their historical activity data, with a focus on extracting dynamic user preferences. Current graph neural network methods often overlook user behavior patterns and fail to consider time gaps between consecutive interactions, leading to reduced prediction accuracy. To address these limitations, we propose a Time-adaptive Prediction Dynamic Graph Neural Network for Sequential Recommendation. Our approach introduces three key innovations: First, we model user behavior sequences as dynamic graphs and implement a bucket-based adaptive threshold time prediction module to capture high-dimensional dynamic connections, thereby enhancing time data utilization. Second, we develop a dynamic memory attention module that effectively captures both long-term and short-term dependencies in user behavior, enabling rapid adaptation to environmental changes and flexible memory management. Finally, we employ a dual-branch dynamic weighted activation strategy to enhance the model's expressiveness, addressing the gradient vanishing issue during preference extraction and enabling precise, adaptive updates of user preferences. Extensive experiments on Amazon datasets (Beauty, Games, and CDs) demonstrate the superiority of our model, with Hit@10 improvements of 7.25%, 1.89%, and 3.11%, and NDCG@10 enhancements of 9.88%, 3.35%, and 4.36% respectively. These results validate that our algorithm significantly improves the model's perception of sequential time information and achieves superior performance in capturing users' dynamic preferences for sequential recommendation tasks.**

***Index Terms*—Sequential recommendation, Graph neural network, Bucket-based adaptive threshold, Dynamic memory attention, Dual-branch dynamic weighted activation**

I. Introduction

SEQUENTIAL recommendation (SR) algorithms leverage users' historical behaviors to predict their future actions, playing an important part in a number of real-world applications, including news recommendation, ad delivery, click- through rate prediction, and e-commerce. Unlike traditional recommendation algorithms that model user preferences statically, sequential recommendation captures users' dynamic and evolving preferences. For instance, while a user may generally dislike sports news (a long-term preference), they might show interest during the Olympics (a medium-term preference). Additionally, their preference may further vary based on their favorite team's performance (a short-term preference).

Several established methods for sequential recommendation have been proposed. For instance, sequential recommendation algorithms utilizing Markov Chain (MC) [1] models generate suggestions by examining the user's most recent k interactions. Models leveraging Recurrent Neural Networks (RNNs) [2],[3] effectively capture sequence dependencies using Gated Recurrent Units (GRUs) [4] or Long Short-term Memory (LSTM) networks. Furthermore, Convolutional Neural Networks (CNNs) [5] and attention networks [6] have been employed to model user sequences, incorporate interaction sequences, and capture item correlations to infer user intent. Recent advances in Graph Neural Networks (GNNs) [7] have enabled researchers to develop sequence models [8]-[11] that analyze complex item transitions and extract behavior patterns from temporal sequences. These models generate spatio-temporal embeddings by integrating temporal and structural information, improving predictions of users' next interests. Sequential recommendation problems have made considerable advancements through the application of deep learning; however, several limitations and challenges continue to persist.

First, as illustrated in Fig. 1 (where the percentage of interactions represents the proportion of a user's total interactions occurring on a specific day), user interactions provide substantial information about time intervals. However, existing sequential recommendation methods [10], often disregard timestamps and focus solely on the order of items in the sequence. This approach implicitly assumes that consecutive items in a sequence occur at uniform time intervals. In time series modeling, most methods only rely on the order of user access to encode time series information sequentially, without considering the long and short-time intervals of user access to items. Notably, when the user sequence is sparse, encoding only the time order and ignoring the time interval information can lead to significant information loss. For example, if two users share an identical sequence of interactions, but one completes these interactions within an hour while the other takes a week, the impact of their interactions on the subsequent item differs, despite occupying the same position in the sequence. However, current sequential recommendation methods focus solely on temporal order and treat these scenarios as identical.

Second, in e-commerce and social media systems, user records continuously accumulate, leading to excessively long behavior sequences. Although long behavior sequences

Manuscript received January 12, 2025; revised April 28, 2025.

This work was supported by the General Project of scientific research funds of the Liaoning Provincial Department of Education (Grant No.2021LJKZ0327); the GPU Resource Support Project of Liaoning Technical University (2024-02).

Wanzhi Chen is an associate professor of School of Software, Liaoning Technical University Huludao, 125105, China. (corresponding author to provide phone: +86 135-919-96866; e-mail: chenwanzhi@lntu.edu.cn).

Zhuo Li is a postgraduate student of School of Software, Liaoning Technical University, Huludao, 125105, China. (e-mail: 1357622766@qq.com).

contain rich information, traditional recommendation algorithms struggle to compute optimally for these sequences. Standard sequence learning architectures (e.g., RNNs, CNNs, and attention networks) often fail to capture long-term dependencies effectively. Current methods classify input sequences into long-term and short-term categories, extracting both temporal and long-term user preferences [12]. While this approach can predict the next activity, it has high computational complexity and is influenced by early interactions, which leads to forgetting long-term knowledge. Thus, creating a sophisticated model that specifically takes into account both short-term and long-term goals is essential. However, capturing and maintaining long-term dependencies remains unstable, and processing sequence data is still limited. To enhance model performance, it is essential to update and refine memory embeddings. By leveraging dynamic memory, the model can selectively emphasize different aspects of inputs during decision-making, thereby enabling more precise predictions of user intent.

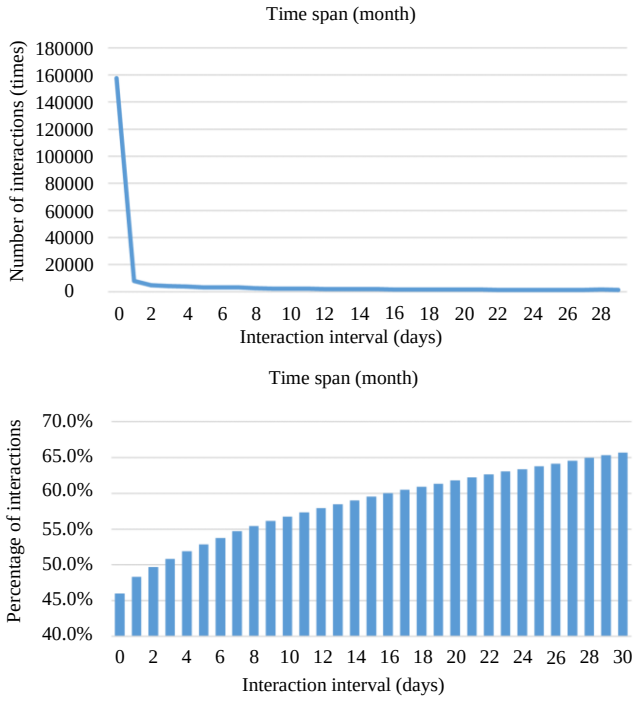

Fig. 1. The number of interactions in a month and their percentage of total data.

Finally, traditional activation mechanisms are static, while user characteristics are dynamic. Static models cannot adapt to changes in input data, limiting their performance with varying inputs. Additionally, a single processing branch cannot fully explore complex feature relationships, thus reducing the model's expressive power and prediction accuracy. This static, single-branch approach is inadequate for handling high-dimensional features and large-scale data.

To address these difficulties, the present study introduces a sequence recommendation optimization algorithm that combines a Dynamic Graph Neural Network (DGNN) with a Dynamic Memory Attention Module (DMAM), termed the Time- adaptive Prediction Dynamic Graph Neural Network for Sequential Recommendation (TaDGSR). Firstly, a bucket-based Adaptive Threshold Time Prediction Module (ATTP) is proposed for sequential recommendation tasks. This module incorporates a time-bucket strategy to dynamically adjust prediction thresholds. This allows the recommendation system to effectively track changes in user behavior across different time spans, addressing the issue where traditional time-processing methods fail to flexibly respond to dynamic user behavior. Secondly, in sequential recommendation, a dynamic memory attention module selectively retrieves and emphasizes relevant historical user interactions from a dynamic memory bank. This procedure improves the model's ability to adaptively focus on significant contextual behaviors and capture long-term dependencies, enabling more accurate and personalized predictions. Finally, a Dual-branch Dynamic Weighted Activation (DDA) mechanism is proposed, which captures the user's multidimensional preferences through a dual-branch structure and dynamically adjusts the weights to flexibly respond to the diversity of user preferences. These advancements aim to enhance the recommendation system's personalization, diversity, and accuracy. Furthermore, the model contributes to addressing the cold-start issue in recommendation systems.

II. Related Works

A. Sequential Recommendation

Sequential recommendation tasks utilize users' historical behavioral sequences to predict their next actions, with a primary focus on modeling the evolution of dynamic user preferences. In one of the earliest works in this field, Rendle et al. [1] employed Markov chains to model behavioral sequence transitions, assuming that the most recently clicked items reflect users' dynamic preferences. Subsequent advancements introduced deep learning-based methods, including attention networks [5], [6] and recurrent neural networks [2], to more efficiently extract critical features from user behavior sequences, thereby accommodating complex behavioral patterns. For example, the Transformer model [13] efficiently processes input sequences in parallel by employing positional encodings and self-attention mechanisms, capturing relationships between elements regardless of their sequential distance.

Recent studies [14] have explored graph neural networks for sequential recommendation due to their exceptional capability in processing graph-structured data. Ren et al. [15] learned inter-domain sequence-level item representations from temporal dependencies and collaborative preference representations from user-item bipartite graphs. In contrast, Ma et al. [16] proposed Memory-Augmented Graph Neural Networks, integrating graph-based user-item interactions with memory modules to enhance the representation of long-term preferences. However, while these models have achieved progress in capturing user interests within sequences, they often overlook inter-sequence item relationships. Inspired by Li et al. [17], this study incorporates time interval information into dynamic graph neural networks to complement embedding representations, addressing the critical issue of neglected time intervals in user interaction sequences.

B. Dynamic Graph Neural Networks

Graph neural networks provide a framework for applying deep learning to graph-structured data, achieving remarkable performance through specific edge and node processing

policies. GNNs have been extensively utilized in recommender systems, leading to significant performance improvements and enabling the adoption of larger-scale networks. Early approaches, such as Graph Convolutional Networks (GCNs) [7], employed spectral graph convolutions to propagate and aggregate node information. Ying et al. [18] extended this by combining random walk-based sampling with graph convolutional networks, marking the first industrial application of GCNs in recommendation systems. Further innovations include Huang et al. [19], who introduced location-augmented and time-aware graph convolution operations to learn dynamic user and item representations on bipartite graphs.

Recent research has shifted toward addressing the limitations of static network embeddings. Huang et al. [14] introduced a dynamic network embedding model that leverages the evolution of high-order similarity within dynamic triples. Xu et al. [20] developed an inductive representation learning framework specifically tailored for temporal graphs, capturing structural and temporal dynamics to generalize effectively to unseen nodes and time steps. Sankar et al. [21] utilized multi-headed self-attention to model structural dynamics in dynamic graphs, studying both structured neighbors and temporal patterns. Recent studies, such as [16], have proposed memory-augmented temporal-adaptive graph convolution methods, which effectively capture long-term dependencies in sequential recommendation tasks.

Although Dynamic Graph Neural Networks (DGNNs) have demonstrated considerable potential in sequential data processing, their practical effectiveness in sequential recommendation remains constrained. Existing models fail to fully exploit the temporal richness of data, particularly in modeling long-range temporal dependencies. This deficiency leads to significant information loss when capturing long-term user preferences, especially in scenarios with sparse interaction sequences. Furthermore, the vanishing gradient problem becomes increasingly pronounced as sequence length increases, exacerbating the difficulty of extracting long and short-term preferences and compromising training stability. To address these limitations, this research explores the applicability of dynamic graph embedding techniques within sequential recommendation contexts, with a focus on enhancing temporal adaptability and capturing dynamic user preferences through time-bucket mechanisms.

III. Methodology

A. Overall Framework

The proposed TaDGSR model introduces a novel architecture for sequential recommendation that systematically captures temporal dynamics through three coordinated components. As shown in Fig. 2, the framework comprises: a user sequence module for temporal pattern extraction, a dynamic graph construction module for interaction modeling, and a dynamic graph recommendation network for preference integration.

First, User Sequence Module captures the historical user-item interactions and generates time-series data [22]. It serves as the foundation for constructing the dynamic graph by providing the necessary temporal and interaction information. Secondly, the dynamic graph construction module translates temporal interaction sequences into evolving graph representations. This component implements two key strategies: subgraph sampling for computational efficiency and time-bucket partitioning for temporal resolution control. Through continuous graph updates, the module maintains an adaptive representation of user preferences that evolves with behavioral changes while mitigating noise from sparse interactions. Finally, Dynamic Graph Recommendation Network leverages long-term and short-term memory mechanisms to model user preferences across different time spans. It employs a dual-branch dynamic weighted activation strategy to enhance the accuracy of preference capture. The network integrates dynamic memory attention to further refine the final prediction results by combining both long-term and short-term user preferences.

The overall system aims to overcome the shortcomings of current sequential recommendation techniques by integrating temporal dynamics and adaptive processes, improving the recommendations' accuracy and personalization.

1) Dynamic Graph Construction

A pivotal stage in the TaDGSR model is creating the dynamic graph, which depicts the changes in user-item interactions over time. Unlike traditional static graphs, the dynamic graph accounts for the temporal dependencies between interactions, enabling the model to gradually adjust to modifications in user behavior. The dynamic graph is constructed based on user interaction quintuples (u, i, t, o_u^i, o_i^u), where: u represents the user, i represents the item, t is the timestamp of the interaction, o_u^i denotes the user's interaction behavior embedding with item i (e.g., click, purchase, add-to-cart), o_i^u represents the item's interaction behavior embedding with user u (e.g., being viewed, purchased, rated).

Each interaction quintuple is used to update the dynamic graph, ensuring that the graph reflects the latest user behaviors. The edges in the graph are not static but evolve over time, capturing the strength and type of interactions between users and items. This temporal evolution features the dynamic graph, which can capture user preference fluctuations across timescales. The dynamic graph is constructed incrementally by aggregating subgraphs from different time intervals. This approach ensures that the graph remains up-to-date and can adapt to changes in user behavior. By continuously updating the graph, the model can make more accurate predictions about future user interactions.

2) Sub-Graph Sampling

With the growth in user numbers and interactions, the size of the dynamic graph expands rapidly, leading to high computational costs and potential noise in the target sequence S^u. A subgraph sampling technique is employed to address this issue, effectively reducing computational complexity while maintaining model accuracy. During the subgraph sampling process, a selection of nodes and edges from the entire dynamic graph is identified as part of the procedure. Specifically, the model samples multi-hop neighbors of the target user u through an iterative sampling algorithm, resulting in a subgraph $G_n^m(t_k)$ of order m, where m is a hyperparameter controlling the size of the subgraph.

The subgraph $G_n^m(t_k)$ covers the sequence S^u and its relevant nodes, ensuring that the model captures the most

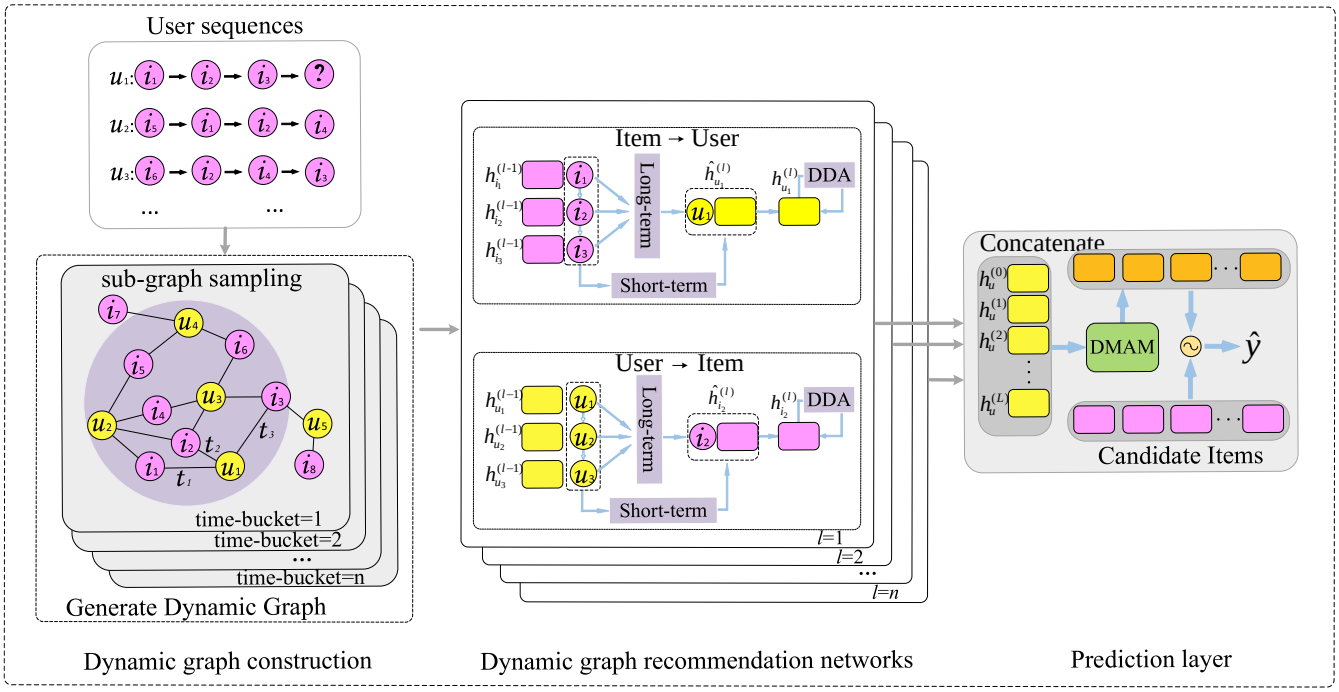

Fig. 2. Model framework diagram.

important interactions for the target user. The sampling strategy allows the model to focus on the most relevant parts of the graph, reducing noise and improving the efficiency of training and prediction. By using subgraph sampling, the model can handle large-scale datasets more effectively, ensuring that the dynamic graph remains manageable while still capturing the essential temporal and interaction patterns needed for accurate recommendations.

3) Bucket-Based Adaptive Threshold Time Prediction

In order to improve the model's capacity to represent temporal dynamics, a ATTP is introduced. This module divides user interactions into several time intervals, or 'buckets,' based on the user's historical interaction patterns. Each bucket represents a specific time range, facilitating the model's ability to capture user preferences across multiple time intervals. The ATTP module employs an adaptive threshold mechanism to dynamically adjust time-bucket boundaries by analyzing user behavior characteristics such as interaction frequency and duration within specific periods. Specifically, it initializes with default temporal boundaries and progressively optimizes these thresholds through iterative neural network parameter updates. This learnable optimization process enables flexible control over time partitioning granularity, automatically refining temporal resolution in response to evolving behavioral patterns. Such adaptive adjustments ensure optimal alignment with dynamic temporal features in sequential data while enhancing precision in capturing time-sensitive user preferences. For recent interactions, the module assigns higher weights to capture short-term preferences, while for older interactions, the weights are reduced to reflect their diminished influence on current preferences.

The model's ability to adapt to changes in user behavior is ensured by the adaptive threshold method. By segmenting user interactions into time-buckets, the model can more effectively capture temporal dynamics in user preferences, resulting in recommendations that are both more precise and timely. The ATTP module is integrated into the dynamic graph construction process, ensuring that the graph reflects the latest user behaviors and preferences. Through this interaction, the model is able to adapt to changing user behaviors, thereby enhancing the accuracy and customization of recommendations over time.

B. Dynamic Graph Recommendation Networks

The central component of the TaDGSR model is the dynamic graph recommendation network, which integrates GNNs and RNNs to effectively capture both short-term and long-term user preferences. To enhance the model's ability to learn from dynamic user-item interactions, this section elaborates on the messaging systems and node update algorithms employed.

1) *Messaging Mechanisms*

The message passing mechanism encodes both long-term and short-term information through distinct architectural components.

Long-term information propagation: In conventional graph neural networks and recurrent architectures, long-term information propagation typically focuses on either node-neighbor relationships or sequential dependencies.

The architecture integrates the GCN framework to enable direct aggregation of neighboring node embeddings. The core formula is represented as follows:

$$h_u^L = \frac{1}{|N_u|}\sum_{i\in N_u} W_1^{(l-1)} h_i^{(l-1)} \tag{1}$$

$$h_i^L = \frac{1}{|N_i|}\sum_{u\in N_i} W_2^{(l-1)} h_u^{(l-1)} \tag{2}$$

where $W_1^{(l-1)}$, $W_2^{(l-1)}$ are learnable weight matrices, and N_u and N_i and represent the sets of neighboring nodes for user u and item i, respectively.

For recurrent neural networks, GRU networks effectively

model sequential dependencies to compute long-term features of user-item nodes. The long-term feature extraction operates as follows:

$$h_u^L = GRU_U^{(l)}(h_{i_1}^{(l-1)}, \cdots, h_{i_{|N_u|}}^{(l-1)}), i \in N_u \quad (3)$$

$$h_i^L = GRU_I^{(l)}(h_{u_1}^{(l-1)}, \cdots, h_{u_{|N_i|}}^{(l-1)}), u \in N_i \quad (4)$$

where $h_u^{(l-1)}$ and $h_i^{(l-1)}$ represent the embeddings of user u and item i at the previous layer, respectively.

Graph attention networks (GATs) have been widely applied for node representation learning in graph-structured data. Building on this foundation, our study further enhances the processing of time-series information within dynamic graphs. While traditional GNNs excel at capturing relationships between a central node and its neighboring nodes, they often struggle to effectively process neighboring nodes that include temporal sequence information. To overcome this drawback, this study incorporates a graph attention mechanism that constructs a user behavior graph and dynamically aggregates information from neighboring nodes through attention mechanisms. By combining this approach with sequential encoding, our model captures both global relationships and high-order dependencies, enabling more accurate and context-aware recommendations.

The fundamental basis for determining the relative order r_u^i of item i in relation to the most recently interacted item by user u is organized through each interaction quintuple (u, i, t, o_u^i, o_i^u) within the dynamic graph attention mechanism. To effectively encode order information, a unique embedding vector Q_r^K is allocated to each relative order r. Subsequently, user node u and its neighboring item nodes i are then compared to determine the attention coefficients via the attention mechanism, represented as follows:

$$h_{it}^{(l-1)} = W_1^{(l-1)} h_i^{(l-1)} \quad (5)$$

$$e_{ui} = \frac{(W_2^{(l-1)} h_u^{(l-1)})^T (h_{it}^{(l-1)} + Q_{r_u^i}^K)}{\sqrt{d}} \quad (6)$$

where d represents the embedding dimension. The denominator $\sqrt{d}$ prevents the dot product from becoming too large and accelerates convergence, thereby stabilizing the attention scores. By applying the softmax function, weight scores a_{ui} for neighboring nodes are obtained. This enables the aggregation of user's long-term preferences through weighted combination of neighboring nodes, as represented as follows:

$$\alpha_{ui} = \text{softmax}(e_{ui}) \quad (7)$$

$$h_u^L = \sum_{i \in N_u} \alpha_{ui} (h_{it}^{(l-1)} + Q_{r_u^i}^V) \quad (8)$$

where Q_r^V is a relative order embedding that captures the order information for user nodes when aggregating neighbor information. Similarly, Q_r^V is another relative order embedding that captures the order information for item nodes when aggregating neighbor information. The long-term preference representation h_i^L of item node i is obtained by weighted summation of user node embeddings and is represented as follows:

$$h_u^{(l-1)} = W_2^{(l-1)} h_u^{(l-1)} \quad (9)$$

$$e_{iu} = \frac{(W_1^{(l-1)} h_i^{(l-1)})^T (h_u^{(l-1)} + Q_{r_i^u}^K)}{\sqrt{d}} \quad (10)$$

$$\beta_{iu} = \text{softmax}(e_{iu}) \quad (11)$$

$$h_i^L = \sum_{u \in N_i} \beta_{iu} (h_u^{(l-1)} + Q_{r_i^u}^V) \quad (12)$$

Short-term Information Processing: Short-term information reflects the latest behaviors and interest changes of nodes. In recommendation systems, capturing users' immediate needs via short-term information is crucial. Traditional systems often use the embedding of the most recent interaction as the short-term preference representation, yet this approach overlooks the impact of historical information. The propagation of short-term information is typically achieved through attention mechanisms, which model short-term preferences. Here, $\hat{\alpha}, \hat{\beta}$ are the attention coefficients between users and items. The calculation process for the short-term preferences of user node u and the short-term preferences of item node i is represented as follows:

$$\hat{\alpha}_{ui} = \text{softmax}\left(\frac{(W_3^{(l-1)} h_{i_{|N_u|}}^{(l-1)})^T (W_2^{(l-1)} h_i^{(l-1)})}{\sqrt{d}}\right) \quad (13)$$

$$\hat{\beta}_{iu} = \text{softmax}\left(\frac{(W_4^{(l-1)} h_{u_{|N_i|}}^{(l-1)})^T (W_1^{(l-1)} h_u^{(l-1)})}{\sqrt{d}}\right) \quad (14)$$

$$h_u^S = \sum_{i \in N_u} \hat{\alpha}_{ui} h_i^{(l-1)} \quad (15)$$

$$h_i^S = \sum_{u \in N_i} \hat{\beta}_{iu} h_u^{(l-1)} \quad (16)$$

The primary function of W_3, W_4 are to regulate the weight allocation of the most recent interactive item within the attention mechanism, ensuring that short-term preferences are accurately captured and integrated into the recommendation process.

2) Node Update

The node update process is crucial for maintaining the dynamic nature of the graph. It aims to generate new node embeddings by integrating long-term and short-term information with embeddings from the previous layer, accurately reflecting the evolving preferences of users and items over time.

The DDA technique is employed in the user node update process to effectively integrate both short-term and long-term preferences. The DDA approach dynamically adjusts the weights assigned to short-term and long-term information based on the user's current behavior, thereby ensuring the model's adaptability in response to fluctuations in user preferences. The update rule for user nodes is represented as follows:

$$g_u^{(l)} = \sigma(W_5^{(l)}[h_u^L \parallel h_u^S \parallel h_u^{(l-1)}]) \quad (17)$$

$$h_u^{(l)} = g_u^{(l)} \odot \text{ReLU}(W_L^{(l)} h_u^L) + (1 - g_u^{(l)}) \odot \text{ReLU}(W_S^{(l)} h_u^S) + h_u^{(l-1)} \quad (18)$$

where $h_u^{(L)}$ is the gating coefficient, controlling the fusion degree of long-term and short-term information. $W_5^{(l)}$ is the weight matrix used to compress the connection vector. Independent parameter matrices $W_L^{(l)}$ and $W_S^{(l)}$ then process the long-term and short-term features separately. Preventing feature mixing while enhancing the model's capacity to pick up on various user preferences is the aim. The update rule for item nodes is represented as follows:

$$g_i^{(l)} = \sigma(W_5^{(l)}[h_i^L \,||\, h_i^S \,||\, h_i^{(l-1)}]) \tag{19}$$

$$h_i^{(l)} = g_i^{(l)} \odot \mathrm{ReLU}(W_L^{(l)} h_i^L) + (1 - g_i^{(l)}) \odot \mathrm{ReLU}(W_S^{(l)} h_i^S) + h_i^{(l-1)} \tag{20}$$

C. Adjustment and Optimization

The adjustment and optimization process are crucial for ensuring that the TaDGSR model can effectively capture dynamic user preferences and make accurate recommendations. This section details the dynamic memory attention module and the prediction layer, which aim to improve the model's capacity to adjust to modifications in user behavior and improve recommendation accuracy.

1) Dynamic Memory Attention Module

To precisely capture the dynamic evolution of user preferences, we propose a DMAM that synergistically integrates differentiable memory storage with gated attention mechanisms. As illustrated in Fig. 3, the module operates through four coordinated phases of memory management.

The dynamic memory attention module effectively stores and updates the evolving states of user behaviors, thereby enhancing the model's ability to recognize long-term dependencies. Its update process comprises four key steps: erase, write, update, and read. The structure is illustrated in Fig. 3.

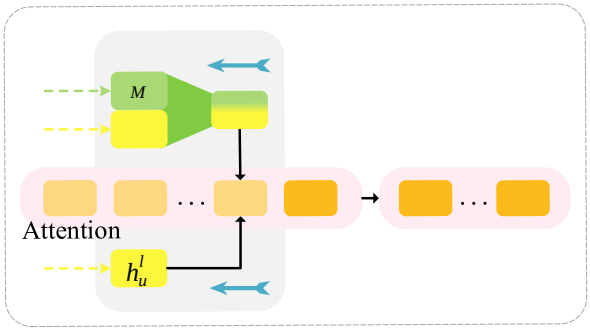

Fig. 3. Dynamic Memory Attention Module.

Erase Operation: This step removes information in the memory that is irrelevant to the current interaction by computing an elimination signal e_t. The elimination signal is computed as follows:

$$e_t = \mathrm{Sigmoid}(W_e h_u^{(l)} + b_e) \tag{21}$$

where W_e represents the weight matrix, and b_e represents the bias term. The Sigmoid activation function restricts the values of e_t to the range [0,1], indicating the proportion of memory to be erased.

Write Operation: This step integrates new interaction information into the memory by calculating an addition signal a_t, which adds new user node information to the memory module. The addition signal is computed as follows:

$$a_t = \mathrm{Tanh}(W_a h_u^{(l)} + b_a) \tag{22}$$

where W_a represents the weight matrix, and b_a represents the bias term. The Tanh activation function ensures the values of a_t fall within the range [-1,1], representing the content to be added to the memory.

Update Operation: The dynamic memory attention module M updates its content using weight matrices and bias parameters. The update process is represented as follows:

$$M_t = M_{t-1}(1 - W_w e_t) + W_w a_t \tag{23}$$

where W_w represents the weight matrix, indicating the impact of current interactions on the memory matrix. This allows the memory module to dynamically adjust its content to reflect the user's latest behavior patterns.

Read Operation: This step retrieves the latest state of the user node from the module by computing weights through the product of module M_t and the current interaction node information $h_u^{(l)}$. The weights are then compressed and activated via a neural network layer to produce the final output $h_u^{(l)}$. The computation process is represented as follows:

$$\tilde{h}_u^{(l)} = \mathrm{Sigmoid}(W_r (M_t h_u^{(l)}) + b_r) \tag{24}$$

where W_r is the reading weight matrix, and b_r is the bias term. The output $\tilde{h}_u^{(l)}$ represents the user node output, providing an interpretable memory mechanism for the model.

2) Prediction Layer

The prediction layer propagates through multi-layer dynamic graph neural network modules to obtain multiple embedding vectors for user nodes at different timestamps. These vectors reflect user preference features at various time points. The model concatenates these multi-layer embedding vectors to derive the final embedding vector for user nodes, which is represented as follows:

$$h_u = h_u^{(0)} \,||\, h_u^{(1)} \,||\, \cdots \,||\, h_u^{(L)} \tag{25}$$

The model employs a link prediction function to assess the probability of user-item interactions for a specified candidate item, which is represented as follows:

$$S_{ui} = h_u^T W_Q e_i \tag{26}$$

where W_Q represents the trainable transformation matrix that aligns user and item embeddings within a shared space. S_{ui} denotes the score assigned by user u to candidate item i, while e_i refers to the feature vector associated with item i.

The model organizes the S_{ui} scores of all candidate items into a vector, thereby generating a score vector for the user corresponding to each item. Subsequently, the model normalizes the score vector S_u through the softmax function to derive the normalized probability distribution of user node u over the candidate items, which is represented as follows:

$$\hat{y}_u = \mathrm{softmax}(S_u) \tag{27}$$

To optimize the model parameters, we employ the cross-entropy loss function as our objective function. The One-Hot encoding vector representing the next item that user u interacts with is denoted by the symbol y_{ui}. The formulation of the objective function is presented as follows:

$$loss = -\sum_{S} \sum_{i=1}^{|I|} y_{ui} \ln(\hat{y}_{ui}) + (1 - y_{ui}) \ln(1 - \hat{y}_{ui}) + \lambda \|\Theta\|_2 \tag{28}$$

where Θ denotes all trainable parameters within the model, $\|.\|_2$ signifies the L2 norm used for parameter regularization, and λ represents a hyperparameter governing regularization intensity, which modulates the impact of the regularization term.

IV. Experiment

A. Experimental Environment and Experimental Dataset

Experiments were conducted on three benchmark datasets from the Amazon platform [23]: Beauty, Games, and CDs. These datasets vary in category diversity, sequence length, and sparsity, as summarized in Table I.

TABLE II
PERFORMANCE COMPARISON OF THE MODELS ON THREE DATASETS

Method	Beauty		Games		CDs	
	NDCG@10	Hit@10	NDCG@10	Hit@10	NDCG@10	Hit@10
FPMC	0.2891	0.4310	0.4680	0.6802	0.3355	0.5122
GRU4Rec+	0.2642	0.4398	0.4564	0.6715	0.4452	0.6784
Caser	0.2547	0.4264	0.4593	0.6883	0.4585	0.6865
SASRec	0.3219	0.4854	0.5360	0.7398	0.4923	0.7132
SR-GNN	0.3233	0.4862	0.5325	0.7349	0.4895	0.6963
TiSASRec	0.3045	0.4687	0.5019	0.7185	0.4897	0.7100
HGN	0.3247	0.4863	0.4934	0.7142	0.4934	0.7142
HyperRec	0.2326	0.3471	0.4896	0.7124	0.4716	0.7102
DGSR	0.3290	0.4871	0.5618	0.7618	0.5118	0.7213
TaDGSR	0.3615	0.5224	0.5806	0.7762	0.5341	0.7437

An Intel Xeon Platinum 8352V CPU facilitated computational support for the PyTorch implementation of the model, which was executed on an NVIDIA GeForce RTX 4090 GPU. All experiments were conducted five times using different random seeds to ensure reproducibility, and the average metrics are reported. An embedding size of 50, a batch size of 50, and a learning rate of 0.01 tuned using the Adam algorithm were important hyperparameters [24]. To mitigate overfitting, L2 regularization with a coefficient of 1×10^{-4} was employed. The network architecture consistently comprises two layers, and the time-buckets are divided into 4. The maximum sequence length was fixed at 50 to balance computational efficiency and information retention.

TABLE I
STATISTICS OF THE EXPERIMENTAL DATASET

Dataset	Beauty	Games	CDs
Number of Users	52204	31013	17052
Number of Items	57289	23715	35118
Interactions	394908	287107	472265
Average sequence length	7.59	9.26	27.70
Density	0.01%	0.04%	0.08%

B. Evaluation Metrics

For the evaluation of top-K recommendations, we utilize two widely recognized metrics [25]:

Hit@K: This metric indicates whether the target item appears among the top-K suggestions.

$$\text{Hit@K} = \frac{1}{|\mathcal{N}|}\sum_{i\in\mathcal{N}} f(Rank_{r_{s_{n+1}}} \le \text{K}) \tag{29}$$

where $\mathcal{N}$ is the test set, and $f(.)$ is an indicator function.

NDCG@K: By assigning greater weights to higher positions, this metric assesses the quality of the rankings.

$$\text{NDCG@K} = \frac{1}{|\mathcal{N}|}\sum_{i\in\mathcal{N}} \frac{2^{f(Rank_{r_{s_{n+1}}}<\text{K})}-1}{\log_2(Rank_{r_{s_{n+1}}}+1)} \tag{30}$$

C. Performance Comparison

We conduct a comprehensive evaluation of the proposed TaDGSR model by comparing it with the following state-of-the-art techniques:

FPMC [1]. A hybrid model for sequential recommendation that integrates Markov chains and matrix factorization.

GRU4Rec+ [3]. An enhanced GRU-based model designed for session-based recommendations, optimized using ranking loss.

Caser [5]. CNN-based sequential model capturing high-order item transition patterns.

SASRec [6]. Self-attention sequential recommender with positional encoding.

SR-GNN [8]. This model employs graph neural networks for session recommendation, effectively capturing item transition dynamics.

DGSR [10]. The essence of this approach is to employ dynamic graph neural networks to effectively capture the evolving interactions within the user-item graph.

TiSASRec [17]. Time interval-aware self-attention model for temporal interaction modeling.

HGN [26]. Hierarchical gating network for multi-scale user interest extraction.

HyperRec [27]. Hypergraph-based sequential recommendation model capturing high-order item relationships.

D. Analysis of Experimental Results

1) Evaluation Index Analysis

The performance comparison among the models is presented in Table II. After analysis, it is evident that all baseline models are surpassed by the proposed approach. Notably, this approach improves performance by 7.25%, 1.89%, and 3.11% in Hit@10, and 9.88%, 3.35%, and 4.36% in NDCG@10 across three datasets in comparison to the highest-performing baseline model, DGSR. There are two primary causes for this dominance. First, the introduction of the time-bucket mechanism allows for better utilization of user interaction time information in dynamic graph neural networks. Second, the adoption of the dynamic memory attention network effectively addresses the limitations of early interactions in traditional graph neural networks, thereby significantly enhancing prediction accuracy.

The experimental results of the proposed algorithm, in comparison to alternative methods, are presented in Fig. 4 and Fig. 5 for the Beauty, Games, and CDs datasets. The proposed methodology consistently outperforms competing strategies across all three datasets under various evaluation metrics, as illustrated by the figures. Notably, it demonstrates exceptional performance on both the Beauty and CDs datasets. This can be attributed to the fact that the Beauty dataset contains a limited number of interactions, with many users and items engaging infrequently. In this scenario, the proposed method's use of time-bucket embedding effectively mitigates the data sparsity problem. The preliminary analysis of the CDs dataset indicates that users have relatively long average sequence lengths. The dynamic memory attention network demonstrates its ability to capture temporal dynamics, successfully addressing limitations caused by

TABLE III
ABLATION EXPERIMENTS

#	DGNN	ATTP	DMAM	DDA	Hit@5	Hit@10	Hit@20	NDCG@5	NDCG@10	NDCG@20
1	√	×	×	√	0.5721	0.6907	0.7969	0.4376	0.4761	0.5031
2	√	√	×	×	0.6042	0.7113	0.8067	0.4853	0.5129	0.5463
3	√	×	√	×	0.6387	0.7516	0.8257	0.5110	0.5518	0.5641
4	√	×	√	√	0.6722	0.7688	0.8500	0.5424	0.5737	0.5944
5	√	√	×	√	0.6759	0.7711	0.8516	0.5484	0.5793	0.5996
6	√	√	√	×	0.6771	0.7754	0.8519	0.5482	0.5798	0.6001
7	√	√	√	√	0.6794	0.7762	0.8561	0.5498	0.5806	0.6010

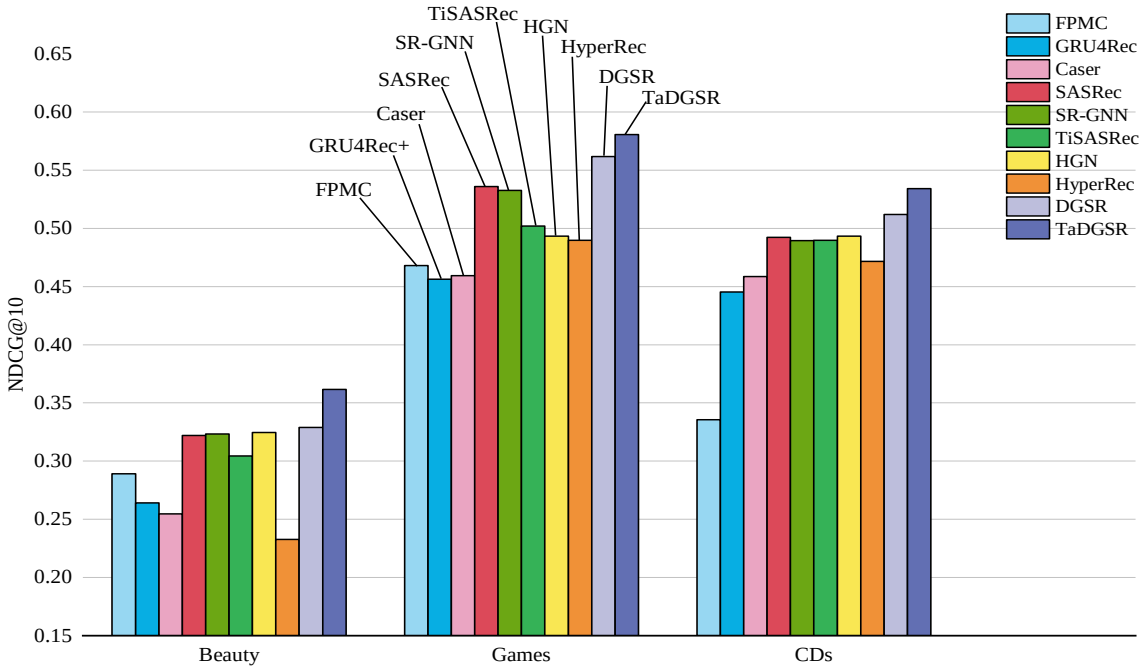

Fig. 4. The NDCG@10 of the methods being compared on different dataset.

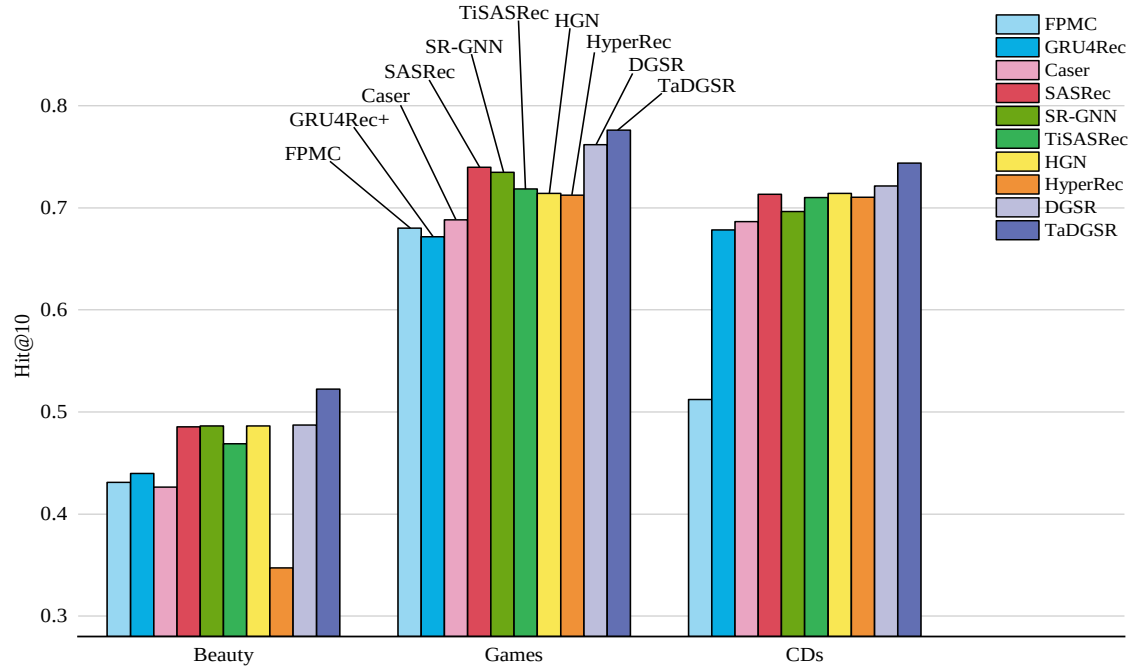

Fig. 5. The Hit@10 of the methods being compared on different datasets.

sequence length and computational complexity in the model.

2) Ablation and Effectiveness Analyses

To validate the effectiveness of the proposed modules in sequential recommendation tasks, ablation experiments were performed on the Games dataset. As shown in Table III, '√' indicates that the specified algorithm policy is adopted. Ablation experiments 1, 2, and 3 represent the use of individual methods.

TABLE IV
THE EFFECT OF EACH ALGORITHM ON THE COLD-START DATA SET

Algorithm	Beauty				Games			
	Recall@10	NDCG@10	Recall@20	NDCG@20	Recall@10	NDCG@10	Recall@20	NDCG@20
SASRec	0.1623	0.0912	0.2035	0.1093	0.1912	0.1105	0.2116	0.1029
SR-GNN	0.1867	0.0987	0.2293	0.1288	0.2156	0.1231	0.2437	0.1195
DGSR	0.1988	0.1092	0.2416	0.1376	0.2301	0.1328	0.2609	0.1253
TaDGSR	0.2431	0.1365	0.2975	0.1723	0.2789	0.1589	0.3212	0.1645

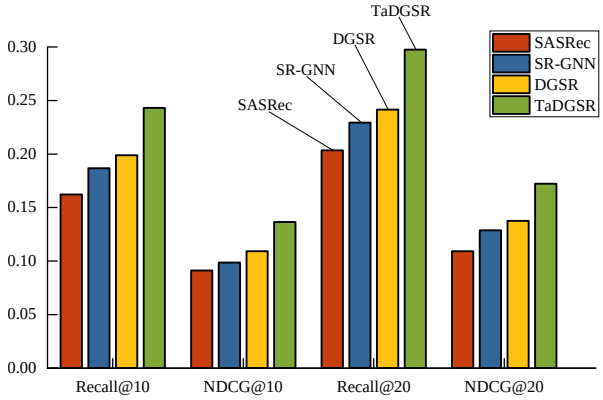

Fig. 6. Performance metrics on the Beauty cold-start dataset

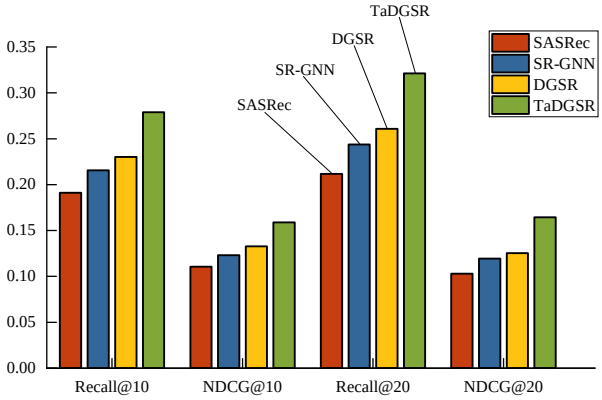

Fig. 7. Performance metrics on the Games cold-start dataset

Experiment 1 integrates DDA into a dynamic graph neural network, mainly to enhance the flexibility of the model in handling complex user and item interactions.

Experiment 2 employs ATTP, optimizing performance by leveraging sequential data and user behavior changes, demonstrating strong capabilities in handling temporal dependencies.

Experiment 3 incorporates DMAM, thereby augmenting the model's capacity to effectively capture long-term dependencies and intricate interactions within dynamic graphs.

Experiments 4, 5, and 6, which use two modules, show significant performance improvements. Experiment 7 shows marked enhancements over Experiment 4 in both Hit@10 and NDCG@10, demonstrating that the absence of ATTP significantly reduces the accuracy of sequential recommendations. Experiment 5, which excludes DMAM, underperforms Experiment 7, primarily because DMAM captures richer temporal dynamics and is particularly effective in complex time-series tasks. In Experiment 6, the absence of DDA results in inferior performance compared to Experiment 7. This indicates that DDA, by providing independent dynamic adjustment paths, alleviates gradient vanishing, thereby improving learning stability and generalization. In Experiment 7, the combined use of ATTP, DMAM, and DDA further elevates performance, highlighting the interdependence and necessity of these modules in sequential recommendation tasks.

3) Performance in Cold-start Scenario

The cold-start problem holds a significant position in recommendation system experiments. In practical applications, recommendation systems often encounter new users or items that lack sufficient historical interaction data. This makes it difficult for traditional recommendation algorithms, which rely on historical behavior, to accurately predict preferences or features, thereby affecting recommendation effectiveness. Thus, the ability to provide accurate recommendations for new users or items quickly in cold-start scenarios has become a key challenge in evaluating recommendation system performance.

In order to confirm that the TaDGSR algorithm effectively reduces cold-start problems for users and items, experiments were conducted on the Beauty and CDs datasets. In performance evaluation, standard ranking evaluation metrics for top-K recommendations were adopted, including Recall@K and NDCG@K. A few items and corresponding users were selected for interaction in cold-start experiments. Cold-start users are defined as individuals who have engaged in fewer than 20 interactions within the test set, and a dedicated cold-start test set was constructed. The results are shown in Table IV and visualized in Fig. 6 and Fig. 7 for easy comparison. The TaDGSR model outperformed other models in all metrics across the two datasets, indicating its significant role in addressing cold-start problem in recommendation systems.

4) Parameter Sensitivity Analysis

We conducted experiments focusing on several key hyperparameters in the proposed method, specifically the number of layers in the dynamic graph network, the number of time-buckets, the maximum sequence length, and the embedding size. These parameters were examined to assess their impact on model performance. Fig. 8 to Fig. 11 present the results of our model's evaluation using the Games dataset.

Impact of TaDGSR Network Layers: To evaluate the effect of different network layers, we experimented with varying numbers of layers in the TaDGSR network, ranging from 0 to 4. As shown in Fig. 8, increasing the number of layers significantly improves performance, demonstrating that leveraging higher-order user sequence information effectively enhances recommendation quality. The optimal performance was achieved with 2 layers. Further increasing the number of layers led to performance degradation, most likely as a result of the over-smoothing problem in graph neural networks that arises from having too many propagation layers.

Impact of Time-Bucket Count: To evaluate the influence of

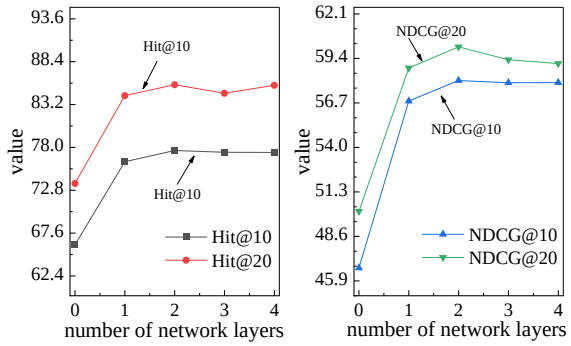

Fig. 8. Impact of the number of network layers.

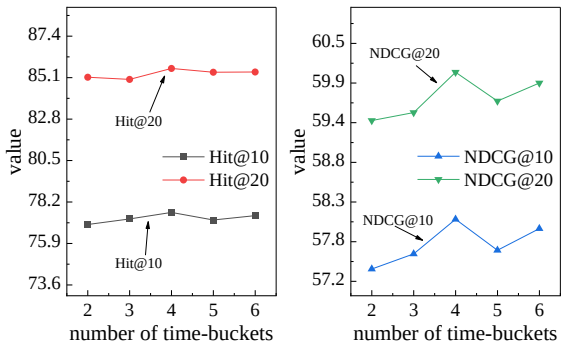

Fig. 9. Impact of the number of time-buckets.

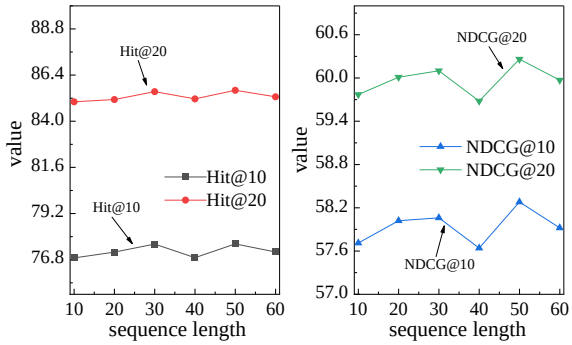

Fig. 10. Impact of the maximum sequence length.

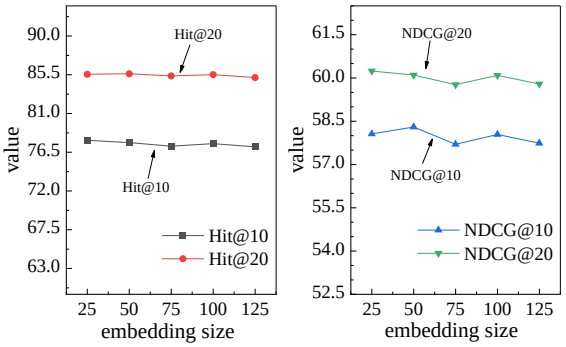

Fig. 11. Impact of the embedding size.

time-bucket count, we tested values from 2 to 6. As shown in Fig. 9, when the number of time-bucket was increased from 2 to 4, there was a significant improvement in the model's performance, reaching its peak at 4. Fewer time-buckets failed to capture fine-grained temporal dynamics, while excessive buckets disrupted behavioral patterns and introduced noise.

Impact of Maximum Sequence Length: To evaluate the effect of sequence length, we tested values from 10 to 60. As shown in Fig. 10, the model showed better recommendation performance across all tested lengths, with performance steadily increasing as the sequence length grew. Performance plateaued at a sequence length of 50, beyond which computational costs rose sharply. Therefore, we selected a sequence length of 50 as the optimal value.

Impact of Embedding Size: To evaluate the effect of embedding size, we conducted experiments with dimensions ranging from 25 to 125. As shown in Fig. 11, model performance improved with increasing embedding size. However, excessively large dimensions risked overfitting, as the model might focus too much on minor variations in training data. Choosing an appropriate embedding size is critical for balancing expressiveness and generalization. We ultimately selected an embedding size of 50, which maintained sufficient expressiveness while minimizing overfitting risks.

V. Conclusions

This paper primarily investigates the impact of the time interval of user behavior and early interactions on prediction effectiveness within the realm of sequence recommendation. Traditional sequence recommendation methods often neglect to adequately consider the temporal aspects of user behavior, leading to limited predictive accuracy. To address this issue, this study introduces a time-adaptive prediction dynamic graph neural network for sequential recommendation By innovatively incorporating time-bucket, a dynamic memory attention network, and a dual-branch dynamic weighted activation strategy, the model's capacity to capture relevant outcomes is significantly enhanced. The experimental results indicate that this model exhibits substantial performance enhancements when compared to the benchmark model index values across multiple datasets, thereby fully demonstrating its superiority in addressing sequence recommendation tasks. Further research will focus on real-time deployment and online learning to facilitate continuous model updates in dynamic environments. Extending the framework to cross-domain recommendations and incorporating causal inference for interpretable predictions are promising directions.

References

[1] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme, "Factorizing personalized Markov chains for next-basket recommendation," in Proceedings of the 19th International Conference on World Wide Web, 2010, pp811-820.

[2] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk, "Session-based recommendations with recurrent neural networks," arXiv preprint arXiv:1511.06939, 2015.

[3] Balázs Hidasi and Alexandros Karatzoglou, "Recurrent neural networks with top-k gains for session-based recommendations," in Proceedings of the 27th ACM International Conference on Information and Knowledge Management, 2018, pp843-852.

[4] Junyoung Chung, Çağlar Gülçehre, Kyunghyun Cho, and Yoshua Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," arXiv preprint arXiv:1412.3555, 2014.

[5] Jiaxi Tang and Ke Wang, "Personalized top-N sequential recommendation via convolutional sequence embedding," in Proceedings of the eleventh ACM International Conference on Web Search and Data Mining, 2018, pp565-573.

[6] Wang-Cheng Kang and Julian McAuley, "Self-attentive sequential recommendation," in 2018 IEEE International Conference on Data Mining, 2018, pp197-206.

[7] Thomas N.Kipf and Max Welling, "Semi-supervised classification with graph convolutional networks," arXiv preprint arXiv:1609.02907, 2016.

[8] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan, "Session-based recommendation with graph neural networks," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, no. 1, pp346-353, 2019.
[9] Mengqi Zhang, Shu Wu, Meng Gao, Xin Jiang, Ke Xu, and Liang Wang, "Personalized graph neural networks with attention mechanism for session-aware recommendation," IEEE Transactions on Knowledge and Data Engineering, vol. 34, no. 8, pp3946–3957, 2020.
[10] Mengqi Zhang, Shu Wu, Xueli Yu, Qiang Liu, and Liang Wang, "Dynamic graph neural networks for sequential recommendation," IEEE Transactions on Knowledge and Data Engineering, vol. 35, no. 5, pp4741-4753, 2022.
[11] Yue Teng and Kai Yang, "Research on Enhanced Multi-head Self-Attention Social Recommendation Algorithm Based on Graph Neural Network," IAENG International Journal of Computer Science, vol. 51, no. 7, pp754-764, 2024.
[12] Yangyang Guo, Zhiyong Cheng, Liqiang Nie, Yinglong Wang, Jun Ma, and Mohan Kankanhalli, "Attentive long short-term preference modeling for personalized product search," ACM Transactions on Information Systems, vol. 37, no. 2, pp1-27, 2019.
[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N.Gomez, Łukasz Kaiser, and Illia Polosukhin, "Attention is all you need," Advances in Neural Information Processing Systems, vol. 30 , 2017.
[14] Hong Huang, Zixuan Fang, Xiao Wang, Youshan Miao, and Hai Jin, "Motif-preserving temporal network embedding," in IJCAI, 2020, pp1237-1243.
[15] Hao Ren, Baisong Liu, Jinyang Sun, Qian Dong, and Jiangbo Qian, "A time and relation-aware graph collaborative filtering for cross-domain sequential recommendation," Journal of Computer Research and Development, vol. 60, no. 1, pp112-124, 2023.
[16] Chen Ma, Liheng Ma, Yingxue Zhang, Jianing Sun, Xue Liu, and Mark Coates, "Memory Augmented Graph Neural Networks for Sequential Recommendation," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 4, pp5045-5052, 2020.
[17] Jiacheng Li, Yujie Wang, and Julian McAuley, "Time interval aware self-attention for sequential recommendation," in Proceedings of the 13th International Conference on Web Search and Data Mining, 2020, pp322-330.
[18] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L.Hamilton, and Jure Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2018, pp974-983.
[19] Liwei Huang, Yutao Ma, Yanbo Liu, Bohong Danny Du, Shuliang Wang, and Deyi Li, "Position-enhanced and time-aware graph convolutional network for sequential recommendations," ACM Transactions on Information Systems, vol. 41, no. 1, pp1-32, 2023.
[20] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan, "Inductive representation learning on temporal graphs," arXiv preprint arXiv:2002.07962, 2020.
[21] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, and Hao Yang, "DySAT: Deep neural representation learning on dynamic graphs via self-attention networks," in Proceedings of the 13th International Conference on Web Search and Data Mining, 2020, pp519-527.
[22] Haibo Hu, Dan Yang, and Yu Zhang, "DPRec: Social Recommendation Based on Dynamic User Preferences," IAENG International Journal of Computer Science, vol. 50, no. 3, pp980-987, 2023.
[23] Julian McAuley, Christopher Targett, Qinfeng Shi and Anton van den Hengel, "Image-based recommendations on styles and substitutes," In Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2015, pp43-52.
[24] Diederik P.Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.
[25] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua, "Neural collaborative filtering," In Proceedings of the 26th International Conference on World Wide Web, 2017, pp173-182.
[26] Chen Ma, Peng Kang, and Xue Liu, "Hierarchical gating networks for sequential recommendation," in Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2019, pp825-833.
[27] Jianling Wang, Kaize Ding, Liangjie Hong, Huan Liu, and James Caverlee, "Next-item recommendation with sequential hypergraphs," in Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp1101-1110.