SIN-CHN SCIENTIFIC PRESS PTE.LTD.

Molecular & Cellular Biomechanics 2025, 22(3), 1339.
https://doi.org/10.62617/mcb1339

Article

TIGNN-RL: Enabling time-sensitive and context-aware intelligent decision-making with dynamic graphs in recommender systems and biomechanics knowledge

Hui Yang, Changchun Yang*

School of Computer Science and Artificial Intelligence, Changzhou University, Changzhou 213164, China
*** Corresponding author:** Changchun Yang, ycc@cczu.edu.cn

CITATION

Yang H, Yang C. TIGNN-RL: Enabling time-sensitive and context-aware intelligent decision-making with dynamic graphs in recommender systems and biomechanics knowledge. Molecular & Cellular Biomechanics. 2025; 22(3): 1339. https://doi.org/10.62617/mcb1339

ARTICLE INFO

Received: 10 January 2025
Accepted: 26 January 2025
Available online: 13 February 2025

Abstract: Intelligent decision-making in dynamic recommender systems is crucial for capturing temporal user preferences and optimizing long-term user satisfaction. Traditional recommender systems often rely on static modeling, neglecting the temporal dynamics of user-item interactions. To address this limitation, we propose a novel framework, Temporal Interpretability Graph Neural Network with Reinforcement Learning (TIGNN-RL), which integrates dynamic graph neural networks (DGNNs) and Proximal Policy Optimization (PPO) to optimize personalized recommendations. Specifically, our method models user-item interactions as dynamic graphs and utilizes temporal interpretability modules to encode both temporal features and node-specific static features. The temporal interpretability module assigns time-aware and interactions weights to user-item, enabling more time-sensitive and explainable dynamic embeddings. This TIGNN dynamic graph sequential embedding is processed by some LSTM modules to be used as the state of the deep reinforcement learning agent and states. We take a joint approach to training, earn graph embeddings that enable better PPO policy. To evaluate the proposed framework, we conduct experiments on three benchmark datasets: Last.fm 1K, MovieLens 1M, and Amazon Product Review. Results show that TIGNN-RL outperforms state-of-the-art baselines, which use GNNs for augmenting DRL-based RS, in terms of accuracy (NDCG@K) and diversity (ILD@K@K), demonstrating its effectiveness in dynamic and interpretable recommendation scenarios. In this research, some biomechanics knowledge is integrated to further enhance the understanding and application of the proposed framework in scenarios where user behavior is influenced by physical factors.

Keywords: recommender systems; subgraphs extraction; dynamic graph neural network; deep reinforcement learning; intelligent decision-making; LSTM; biomechanics

1. Introduction

Intelligent decision-making has become a cornerstone of modern recommender systems, enabling them to provide more accurate, adaptive, and context-aware recommendations [1–4]. As users interact with systems across multiple sessions, their preferences evolve dynamically, creating new challenges for recommendation models [5]. Conventional recommender systems frequently depend on static user profiles or aggregated historical behaviors, which are inadequate in capturing the temporal intricacies of user preferences or the intricate dependencies among items. This limitation severely restricts the capacity of recommender systems to make intelligent, context-sensitive decisions, particularly in multi-round interaction scenarios where long-term user satisfaction and engagement are of paramount importance [6–8]. An existing study have demonstrated through experiments that using static models in

traditional recommendation systems results in an accuracy of predicting user interests of no more than 65% [9].

Graph-based data representations have emerged as a powerful tool for modelling the complex relational structures inherent in recommender systems, such as user-item interactions and item-item associations. Graph Neural Networks (GNNs) have advanced the field by enabling rich feature extraction and representation learning on such graph structures. However, the majority of GNN-based methods are constrained to static graphs, thus disregarding the dynamic nature of user preferences and the temporal evolution of item relationships. These dynamic aspects are critical for intelligent decision-making, as they reflect the underlying processes that drive user behavior and item popularity over time [10–12].

Recent studies have demonstrated the considerable potential of Reinforcement Learning (RL) in the domain of recommender systems. By conceptualizing recommendation tasks within the framework of sequential decision-making problems, RL has exhibited notable efficacy. RL-based approaches have the capacity to optimize long-term objectives, such as user retention or lifetime value, through the implementation of meticulously designed reward mechanisms. However, existing RL methods frequently encounter limitations in state representation, relying on simplistic or static inputs that fail to account for the dynamic nature of user-item interactions [10,13–17]. This shortcoming reduces their ability to make informed decisions across multiple recommendation rounds. Furthermore, the sparsity of user feedback and the vast action space in real-world recommendation scenarios exacerbate the challenges of effective decision-making.

In the context of user interactions with recommendation systems, biomechanics can provide valuable insights. For example, in applications related to physical activities such as fitness apps or sports equipment recommendations, understanding the biomechanics of human movement can help athletes to improve their mechanical insights and enhance their performance. Gongbing Shan [18] explores the role of multidisciplinary collaboration on enhancing human physical ability. By integrating such biomechanics knowledge into our TIGNN-RL framework, we can enhance the understanding of user behavior and improve the recommendation process.

In order to address the aforementioned issues, a novel framework has been proposed. This is known as TIGNN-RL (Temporal Interpretability Graph Neural Network-Reinforcement Learning), and it integrates Dynamic Graph Neural Networks (DGNNs) with RL in order to enable intelligent decision-making in personalized recommendation systems. TIGNN-RL captures the temporal evolution of user preferences and user-item interaction features through DGNNs, generating time-sensitive graph-structured state representations that reflect the dynamic nature of the recommendation environment [10,19,20]. These representations form the basis of an RL-based decision-making strategy, which aims to balance short-term accuracy with long-term user engagement across multiple rounds of interaction.

The primary innovation of TIGNN-RL lies in its capacity to integrate dynamic graph modelling with intelligent decision-making for recommendation tasks. The utilization of DGNNs within the framework facilitates a more nuanced comprehension of the temporal variations in user preferences and item characteristics. This dynamic state modelling, in combination with RL's capacity for sequential optimization,

enables TIGNN-RL to make more informed, context-aware decisions. The intelligent decisions made by TIGNN-RL have the potential to enhance not only the immediate recommendation relevance but also to optimize long-term metrics such as user loyalty, diversity, and overall satisfaction [21–22].

In summary, intelligent decision-making in recommender systems requires a combination of dynamic modelling and strategic optimization. The TIGNN-RL system addresses this need by integrating DGNNs and RL to create a unified framework capable of adapting to temporal changes and optimizing multi-round recommendation strategies. The integration of biomechanics knowledge further enriches the framework, allowing for more accurate predictions in scenarios where physical factors influence user behavior.

The subsequent sections of this paper offer a detailed exposition of the design, implementation, and evaluation of TIGNN-RL. This provides a compelling case study in demonstrating the effectiveness of the proposed framework in propelling the state of the art in intelligent recommendation systems. In this study, it provides a comprehensive overview of the related work in the field of TIGNN-RL, including the development of deep neural networks (DGNNs), research on reinforcement learning (RL) in recommender systems, and the current state-of-the-art research on the integration of dynamic graphs and RL. Besides, the methodology and architecture of DGNNs, discussion of the experimental results, performance evaluation, discussion of the results of the work, and the suggestion of future research directions are described in this research.

2. Related work

2.1. DGNNs

DGNNs have emerged as a powerful tool for modelling time-evolving graph-structured data. In contradistinction to static graphs, which assume a fixed structure, dynamic graphs capture temporal changes in nodes, edges, and their features, rendering them particularly suitable for dynamic environments such as recommender systems. Existing DGNN models can be broadly categorized into two approaches: discrete-time and continuous-time modeling.

In discrete-time DGNNs, the temporal evolution of the graph is divided into snapshots, where each snapshot represents the graph's state at a specific time. The employment of techniques such as temporal convolutional networks (TCN) and recurrent neural networks (RNN) is prevalent in the modelling of dependencies across snapshots. To illustrate this point, consider the Evolved GCN model, which updates GCN parameters over time using an RNN, thereby facilitating the capture of temporal changes in graph structure.

In contrast, continuous-time DGNNs treat graph evolution as a continuous process, often using event-driven methods to model changes in the graph. Notable examples in this category include Graph Neural Ordinary Differential Equations (GraphODE) and Temporal Graph Networks (TGNs). These methods capture finer-grained temporal dynamics by modelling edge creation, deletion, or attribute updates as asynchronous events. While continuous-time models are more expressive, they are computationally intensive and challenging to scale for large graphs.

Notwithstanding the advances achieved, the implementation of DGNNs within the domain of recommender systems poses distinctive challenges. Primarily, user-item interactions are inherently sparse and dynamic, necessitating efficient techniques to handle real-time updates. Secondly, DGNNs must be capable of capturing the evolving preferences of users and the changing characteristics of items while maintaining scalability. Addressing these challenges remains an active area of research.

2.2. RL in recommender systems

The field of RL has garnered considerable attention within the domain of recommender systems, primarily due to its capacity to enhance the efficacy of sequential decision-making processes. In contradistinction to conventional methodologies that emphasize static recommendations, RL-based approaches are oriented towards the maximization of long-term objectives, including, but not limited to, user satisfaction, retention, and lifetime value.

One of the foundational RL applications in the field of recommendation is the use of deep Q-networks (DQN) to model user interactions as a Markov decision process (MDP). To illustrate this point, RL agents are able to learn optimal recommendation strategies by maximizing cumulative rewards based on user feedback (e.g., clicks, purchases). The development of advanced versions, such as Double DQN and Dueling DQN, has led to significant improvements in the stability and accuracy of Q-value estimation in recommendation tasks\cite{lei2020reinforcement}.

Another popular RL paradigm in recommender systems is policy gradient methods, including REINFORCE, A2C, and PPO. These methods directly optimize the recommendation policy by learning a probability distribution over actions, thereby enabling more flexible and adaptive strategies. To illustrate this point, consider the use of Proximal Policy Optimization (PPO) in conversational recommender systems. This approach enables the dynamic adaptation of recommendations based on user responses in multi-turn interactions.

Nevertheless, RL methods in recommender systems encounter numerous challenges. State representation is frequently dependent on static features or aggregated histories, which are unable to account for dynamic user preferences. Additionally, the reward signal is typically sparse and delayed, which hinders the evaluation of the immediate impact of recommendations. Furthermore, the extensive action space, corresponding to a vast catalog of items, introduces scalability issues for traditional RL algorithms. Addressing these limitations through improved state modelling and efficient exploration strategies is an ongoing area of research.

2.3. Integration of DGs and RL

The integration of Dynamic Graphs (DGs) and RL is a promising avenue for enhancing intelligent decision-making in recommender systems. By utilizing the temporal modelling capabilities of DGNNs and the sequential optimization strengths of RL, these approaches have the potential to address the limitations of static modelling and enhance recommendation performance over time.

Recent studies have begun to explore this intersection. For instance, Graph-based Q-learning methods have incorporated GNNs to extract relational features from graphs,

while RL agents have been employed to optimize recommendation strategies based on these features. A notable extension is the use of DGNNs to dynamically update graph representations, enabling the RL agent to adapt to evolving user-item interactions. These frameworks facilitate the capture of both the structural dependencies within graphs and the temporal evolution of relationships.

In a similar manner, policy gradient methods have been combined with graph-based state representations to enhance multi-round recommendation systems. For instance, attention-based DGNNs have been used to encode user-item interaction sequences, thereby providing RL agents with time-sensitive state representations. This approach has been shown to facilitate more informed decision-making, especially in scenarios where user preferences are subject to rapid change over time.

Notwithstanding the aforementioned advances, the integration of DGNNs and RL remains challenging. Primarily, the computational intricacy of dynamic graph updates and RL training can be substantial, particularly in the context of large-scale recommender systems. Secondly, the design of effective reward mechanisms that align with both immediate user feedback and long-term engagement is non-trivial. Finally, the question of how to ensure the interpretability of decisions made by such integrated frameworks remains unresolved.

The proposed TIGNN-RL framework aims to address these challenges by combining the strengths of DGNNs and RL into a unified system. By leveraging DGs modelling for state representation and reinforcement learning for sequential optimization, TIGNN-RL seeks to advance the frontiers of intelligent decision-making in recommender systems. In addition, by considering biomechanics knowledge, we can further enhance the understanding of user behavior and improve the performance of the framework in relevant application scenarios. For example, in a fitness recommendation system, biomechanics principles can help in understanding how different exercises affect the body and predict user preferences based on their physical capabilities and goals [1].

3. Materials and methods

3.1. Problem formulation

The capacity for intelligent and context-aware decision-making in recommender systems necessitates the capability to adapt to dynamic environments, wherein user preferences and item features undergo evolution over time. This dynamic environment is represented by a temporal user-item interaction graph $G_t = (V_t, E_t)$, where V_t includes users and items as nodes, and E_t represents their interactions at time t. Each interaction $e_t \in E_t$ is associated with contextual featyres such as timestamps, interaction types, and user-specific data. These interactions capture the complex, evolving relationships that form the basis of personalized recommendations.

In relevant biomechanics scenarios, such as in fitness or sports-related recommendations, the interactions can also be influenced by factors such as the physical demands of an activity and the user's physical capabilities. For example, a user's past interactions with different types of fitness equipment may be related to their body strength, flexibility, and injury history. The system incorporates biomechanical

contextual features in dynamic environments by leveraging measurable physical ability indicators, such as joint range of motion, muscle strength, and postural stability. Understanding these biomechanical factors can help in providing more accurate recommendations. Incorporating user-specific data related to biomechanics can enable the system to better learn and understand these features.

The core challenge lies in designing a system that balances immediate user engagement with long-term satisfaction. To address this, we model the recommendation process as a Deep RL, where the state s_t represents the system's knowledge at time t, derived from the temporal graph G_t. This state encapsulates user behaviors, item properties, and temporal dynamics, ensuring a comprehensive understanding of the current context. The system generates recommendations as actions a_t, selecting a subset of items from the catalog to maximize engagement. The user's feedback on these recommendations influences the transition to a new state s_{t+1}, as the interaction graph evolves with newly observed behaviors and relationships.

To quantify the effectiveness of the recommendations, a reward function r_t is defined to integrate both immediate feedback, such as clicks or purchases, and long-term objectives, such as user retention or repeat engagement. The reward is designed to reflect the system's dual goals: optimizing short-term relevance while fostering sustained user satisfaction. This balance is achieved by combining immediate and long-term rewards, weighted by a parameter λ, allowing the system to prioritize contextually appropriate strategies. The cumulative reward over multiple rounds is then maximized using a policy $\pi(a_t \mid s_t)$, which maps states to actions and evolves dynamically with user preferences.

The objective of the system is formalized as finding an optimal policy π^* that maximizes the expected cumulative reward over a finite horizon T. This is expressed mathematically as:

$$\pi^* = \arg \max_{\pi} \mathbb{E}_{\pi}[\sum_{t=0}^{T} \gamma^t r_t].$$

where γ is a discount factor that balances the importance of immediate versus future rewards. By incorporating this temporal perspective, the system is equipped to make intelligent, adaptive decisions that respond to both current user needs and anticipated future behaviors.

The framework under discussion is predicated on the state representation, which integrates multi-faceted information to enable intelligent decision-making. The graph topology captures the relational structure of user-item interactions, including both direct links and higher-order dependencies. Temporal encoding techniques are utilized to embed recency and frequency of interactions, thereby reflecting shifts in user interests. Additionally, the system incorporates static features, alongside dynamic embeddings derived from graph neural networks. The integration of these components ensures that the state representation provides a rich, context-aware foundation for generating recommendations.

The transition dynamics further enhance the system's adaptivity by updating the interaction graph based on user feedback. Positive responses, introduce new edges or update existing ones, while changes in item features or user profiles modify node features. This evolving graph structure enables the system to remain responsive to the

continuous interplay between user preferences and item relevance, ensuring that the recommendations remain both timely and personalized.

The formulation of the recommendation task as an DRL, in conjunction with the dynamic interaction graph, facilitates a balance between immediate user engagement and long-term satisfaction. This intelligent, context-aware approach provides a robust foundation for multi-round decision-making, integrating dynamic graph modeling and reinforcement learning to optimize the recommendation process in real time. Incorporating biomechanics knowledge into the state representation and reward function can further improve the decision-making process. And it also allows the system to take into account physical factors that may influence user preferences and behaviors.

3.2. DGs representation

In the recommender systems, the interactions between users and items are subject to dynamic evolution, reflecting complex temporal dependencies. The objective of DGs representation is to capture these temporal changes in user preferences, item features, and their relationships, constructing a graph-structured representation of the system's state in real time.

In **Figure 1**, We use a snapshot approach to describe the process of dynamic graph G_t over time, with each subgraph view G_t then state at different points in time. Each subgraph contains not only nodes and edges, but also node features, edge features, and timestamps.

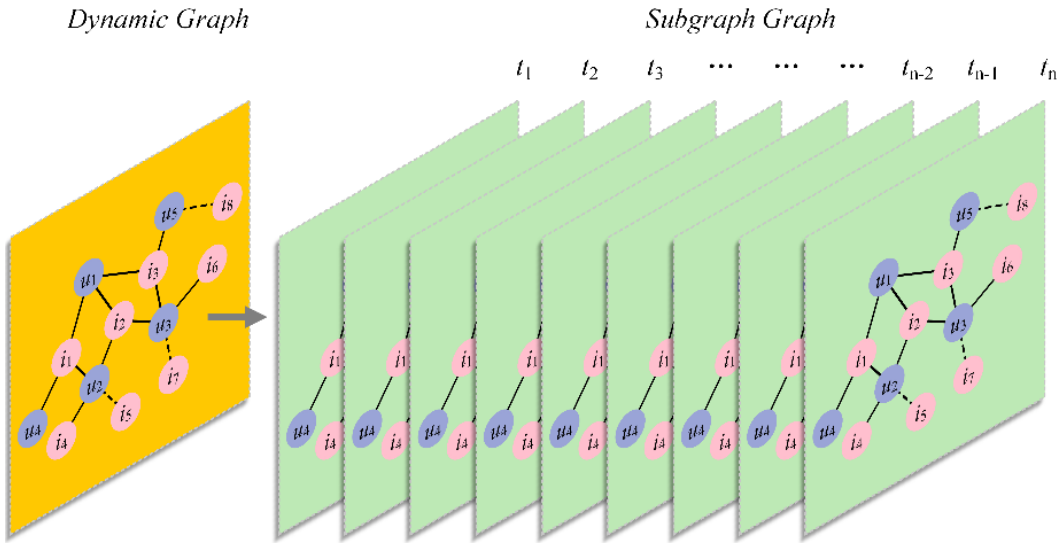


Figure 1. Temporal user-item interaction graph.

3.2.1. Subgraph and edge features

When using DGs as input to the DGNN, if the node features contain dynamic features, and each dynamic feature is time-dependent sequential data, it will correspond to multiple values. If the dynamic features of each node are directly expanded into multiple rows, this will result in the repetition of user nodes in the feature matrix. This repetition may interfere with the embedding processing of the DGNN, since traditional graph neural networks assume that each node corresponds to

one row in the feature matrix. We avoid this problem by using the method of extracting dynamic subgraphs, where the dynamic features are not directly embedded into the node feature matrix, but are modeled as dynamic subgraphs to capture time-dependent information.

The core of the extracting dynamic subgraphs is a temporal user-item interaction graph $G_t = (V_t, E_t)$, where V_t consists of user u and item i nodes, and E_t represents interactions at time t as **Figure 1** show. Each node $v \in V_t$ has a feature vector x_v that includes static features (e.g., user demographics or item metadata) and dynamic features (e.g., recent behaviors or current popularity). Each interaction $e_t = (u, i, t) \in E_t$ includes a timestamp t and optional features x_e, such as interaction type or contextual features. As described in Table 1.

Table 1. Part of temporal subgraph at time point t_n, timestamps are examples.

optional x_e	user node u	item node i	timestamp t
Like	u_1	i_1	2024-01-01 10:00
Buy	u_1	i_2	2024-01-01 10:05
Favorite	u_2	i_2	2024-01-01 10:10
Dislike	u_2	i_4	2024-01-01 10:15
Rate	u_3	i_2	2024-01-01 10:20
Share	u_3	i_3	2024-01-01 10:25

3.2.2. Temporal encoding

As new interactions occur, the graph evolves dynamically, with edges being added or updated to reflect changes in user preferences and item relevance. We use timestamps to represent the temporal state of the edges. Timestamps are usually represented as continuous numerical values, and in order to enable GNNs to effectively recognize and utilize temporal information and understand it's while significance in the temporal dimension, and thus significantly improve the embedding ability of GNNs for dynamic graphs, we temporally encode timestamps.

We encode the timestamp t of each edge as a feature vector t_e:

$$\mathrm{PE}(\Delta t) = \begin{cases} \sin\left(\frac{t - t_{\mathrm{prev}}}{10000^{2i/d}}\right), & 2i \\ \cos\left(\frac{t - t_{\mathrm{prev}}}{10000^{2i/d}}\right), & 2i + 1 \end{cases}.$$

where $\$\Delta t = t - t_{\mathrm{prev}}\$$, t_{prev} is the timestamp of the previous interaction. This enables the model to incorporate the influence of temporal changes on user preferences and item relationships. Edge temporal features and edge semantic features can be combined into complete edge feature vectors:

$$x_e^{\mathrm{final}} = \mathrm{Concat}\,(x_e, t_e).$$

Since we use a snapshot extraction of dynamic subgraphs to provide inputs to the DGNNs, the dynamic features in the node features can be considered without temporal encoding, since their dynamic node features can be considered invariant in each dynamic subgraph.

3.2.3. Subgraph extraction method

In order to improve computational efficiency and focus recommendation on users, we use user nodes as target nodes to extract subgraphs, which include the target users and their associated higher-order neighbors like **Figure 2**. This local subgraph retains context while reducing computational overhead:

$$G_t(u) = (V_u, E_u).$$

The nodes and edges in the subgraph $G_t(u)$ are directly or indirectly related to the target node u. It is imperative that the subgraph $G_t(u)$ guarantees the establishment of connections between nodes, with the objective of capturing contextual information between users and items. Furthermore, the subgraph $G_t(u)$ should be designed to reflect temporal evolution, including recent interactions and time-sensitive relationships.

We focus on the target user node for subgraph extraction to improve computational efficiency (**Figure 2**).

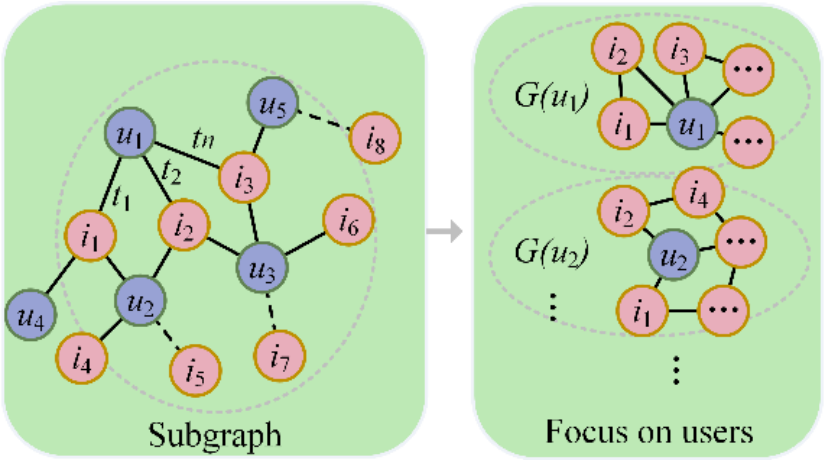


Figure 2. Target user node and subgraph extraction.

The local neighborhood of the query vertex u is identified based on its direct and higher-order neighbors. Include all nodes directly connected to u through edges in E_t:

$$\mathcal{N}_1(u) = i \mid (u, i) \in E_t.$$

where $i \in I$ represents items interacted with by user u. Extend the neighborhood to second-order or higher-order connections to capture indirect relationships:

$$\mathcal{N}_2(u) = j \mid \exists i \in \mathcal{N}_1(u), (i, j) \in E_t.$$

$$\mathcal{N}_k(u) = v \in V_t \mid d(u, v) \leq k.$$

where $d(u, v)$ represents the shortest path distance between nodes u and v, and k is the range of hops. This allows the inclusion of items co-interacted with by other users, enriching the subgraph with collaborative signals.

After selecting the neighborhood, further filter the time-sensitive edge set E_u^t associated with the target node u, defined as:

$$E_u = (v, w, t) \in E_t \mid v, w \in \mathcal{N}_k(u), t \geq t_{\text{threshold}}.$$

where $t_{\text{threshold}}$ is the set time threshold used to filter out early interactions. To ensure relevance, temporal constraints only interactions within a specified time window Δt are considered. This reduces noise and focuses on recent, contextually meaningful relationships.

Further optimize the node set V_u and edge set E_u of the subgraph based on the features values of the nodes or edges. Define the attribute filtering function:

$$V_u' = v \in V_u^t \mid \text{Attr}\,(v) \in \mathcal{C}_v.$$

$$E_u' = e \in E_u^t \mid \text{Attr}\,(e) \in \mathcal{C}_e.$$

where $\mathcal{C}_v$ and $\mathcal{C}_e$ are sets of filter conditions for node and edge features.

For example, node properties could be category labels for items, and filtering could keep only items in certain categories. Edges could be filtered based on interaction types for high-value relationships.

Combine the selected node V_u' and edge E_u' to generate the final subgraph $G_t(u) = (V_u', E_u')$. The generation of subgraphs can be achieved by cropping the adjacency matrix:

$$\mathrm{A}_u = \mathrm{A}[V_u', V_u'].$$

where A is the adjacency matrix of the original complete graph and A_u is the adjacency matrix of the extracted subgraph.

For the DGs scenario, the subgraph needs to be updated in real time to reflect the latest interactions and relationships as time t changes. When a new edge e is added to the full graph G_t, determine whether the filter conditions of the subgraph are satisfied. If so, add (v, w, t) to $G_t(u)$. Remove edges that exceed the $t_{\text{threshold}}$ from the subgraph. When the features of nodes or edges change, re-evaluate whether they satisfy the filter conditions, and add or delete nodes or edges from the subgraph.

The attribute evaluation function of a node is used to determine whether a node meets the filtering conditions of the current subgraph, the evaluation function can be defined as:

$$f_{\text{node}}\,(v, t) = \begin{cases} 1, & \text{if } \mathbf{x}_v(t) \in \mathcal{C}_v \\ 0, & \text{otherwise} \end{cases}.$$

where $x_v(t)$ represents the dynamic features value of node v at time t, such as the current preferences of a user or the status of an item. $\mathcal{C}_v$ is a constraint set of node features, such as the user's preference for a specific category of items or whether the item is valid at the current time. Recalculate $f_{\text{node}}\,(v, t)$ when the features of a node change. If the evaluation changes from 1 to 0, remove the node from the subgraph.

Add new or previously removed nodes to the subgraph if their updated features satisfy $\mathcal{C}_v$.

The features evaluation function of an edge is used to determine whether edge $e = (v, w, t)$ satisfies the filter conditions of the current subgraph. The evaluation function is defined as follows:

$$f_{\text{edge}}\,(e, t) = \begin{cases} 1, & \text{if } \mathrm{x}_e(t) \in \mathcal{C}_e \text{ and } t \geq t_{\text{threshold}} \\ 0, & \text{otherwise} \end{cases}.$$

where $x_e(t)$ represents the dynamic features value of edge e at time t, such as the interaction type, weight etc. $\mathcal{C}_e$ is the constraint set of edge features, such as filtering purchases or interactions with a frequency above a certain threshold. $t_{\text{threshold}}$ is a temporal constraint used to filter expired or historical interactions. Recalculate $f_{\text{edge}}(e,t)$ when the edge features $x_e(t)$ or the timestamp t is updated. If the evaluation value changes from 1 to 0, remove the edge; if the evaluation value changes from 0 to 1, add the edge to the subgraph. Adding new interactions automatically evaluates the features to ensure that they meet the filter conditions before adding them to the subgraph.

During subgraph extraction, the features adjustments of nodes and edges need to be considered comprehensively with respect to contextual constraints. In order to maintain the connectivity of the subgraph, a comprehensive evaluation function is introduced using the evaluation functions of nodes and edges:

$$f_{\text{subgraph}}(G_u,t)=\frac{1}{|V_u|+|E_u|}\left(\sum_{v\in V_u} f_{\text{node}}(v,t)+\sum_{e\in E_u} f_{\text{edge}}(e,t)\right).$$

where $|V_u|$ and $|E_u|$ are the total numbers of subgraph nodes and edges, respectively. This formula is used to measure the extent to which a subgraph as a whole satisfies the property constraints by normalizing the evaluation scores of nodes and edges. When the value of $f_{\text{subgraph}}(G_u,t)$ is lower than the set threshold (for example, 0.8), it indicates that a large number of nodes or edges in the subgraph do not satisfy the constraints, and the entire subgraph needs to be rebuilt or re-extracted. Through this reasoning process, the extracted subgraph not only captures the local relationships of the target node, but also generates a contextually relevant graph structure by combining temporal evolution and attribute information. This dynamic subgraph provides high-quality input for subsequent DGNNs modeling.

3.3. DGNNs modeling

DGNNs play a critical role in capturing the temporal and structural dynamics of user-item interaction graphs in recommender systems. In this study, we integrate advanced DGNN models, such as STGCN (design a temporal convolution layer to capture dynamic behaviors) and DySAT (introduces self-attention mechanism), while introducing innovations in attention mechanisms and memory modules to enhance embeddings interpretability.

The dynamic graph is a spatial-temporal graph, and we consider spatial features by designing a Temporal Interpretability Graph Neural Network (TIGNN) that focuses on temporal coding as well as node and interaction features during aggregation, using the LSTM mechanism to focus on temporal of dynamic subgraphs' sequences (**Figure 3**).

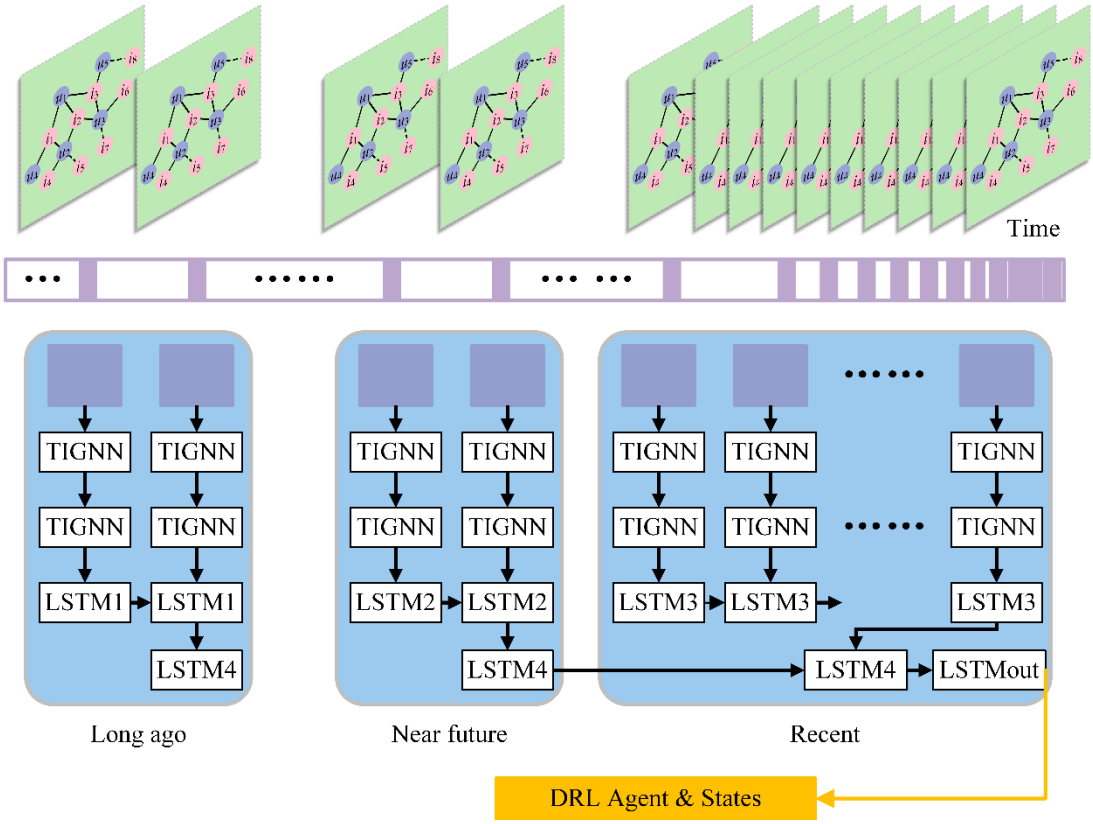


Figure 3. Temporal Interpretability Graph Neural Network (TIGNN).

3.3.1. Message propagation

For each interaction quaternion (u, i, e, t_e), the TIGNN we design takes into account both the user nodes u and item nodes i interactions feature vector e as well as the temporal features of the time-encoded vector e_t in message propagation stage, in order to increase the temporal interpretability.

Considering interpretability, we use a cosine similarity function $cs: \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ to compute the similarity π_e^u between a user and an interaction:

$$\mathrm{cs}\,(u, e) = \frac{u \cdot e}{\|u\|\|e\|}.$$

where $u \in \mathbb{R}^d$ and $e \in \mathbb{R}^d$ are the embedding of features vector of user u and interaction e, and d is the dimension of embedding. We compute the normalized similarity of user u 's neighborhood:

$$\tilde{\pi}_{e_{u,i}}^u = \frac{\exp(\pi_{e_{u,i}}^u)}{\sum_{i \in \mathcal{N}(u)} \exp\,(\pi_{e_{u,i}}^u)}.$$

where i is the embedding of item features vector. $\tilde{\pi}_{e_{u,i}}^u$ act as interpretability personalized filters when computing the user's neighborhood embedding, since we aggregate the items with bias with respect to these user-interactions similarity.

So, we define an interpretability attention mechanism to differentiate the importance weight $w_{u,i}$ of dynamic subgraph users interaction. The weight $w_{u,i}$ taking $l-1$-th layer node embedding $\mathrm{h}_u^{(l-1)}$ and $\mathrm{h}_i^{(l-1)}$ as the input, formulated as:

$$w_{u,i} = \frac{(W_1^{(l-1)} \cdot h_u^{(l-1)})^T \cdot (W_2^{(l-1)} \cdot h_i^{(l-1)} + \tilde{\pi}_{e_{u,i}}^u)}{\sqrt{d}}.$$

where $h_u^{(0)}$ and $h_i^{(0)}$ are the user u features vector embedding and item i features vector embedding, respectively. d is the embedding dimension, and the scale factor $\sqrt{d}$ is used to speed up convergence by avoiding excessively large dot products. The weighting scores between users and their neighbors are obtained via the SoftMax function:

$$\alpha_{u,i} = \text{softmax}\,(w_{u,i}).$$

Thus, the interpretability preference of user can be obtained by aggregating the information from its all neighbors adaptively:

$$h_u^I = \sum\nolimits_{i \in \mathcal{N}_u} (\alpha_{ui}(W_1^{(l-1)} h_i^{(l-1)}) + \tilde{\pi}_{e_{u,i}}^u).$$

The temporal interpretability attention weights $\alpha_{u,i}$ have already been calculated using $\tilde{\pi}$ in the weight formula $w_{u,i}$. At this point, the interactions features are no longer mixed with the features of neighboring nodes, but is used as a global correction term. The final feature of the target node is adjusted directly by the interaction's features, rather than propagating through neighboring nodes, which prevents the influence of the interactions feature from being amplified and even introduces redundancy.

Considering temporal, in the same time, we can also define a temporal attention mechanism to differentiate the importance weight $w'_{u,i}$ of dynamic subgraph users' interaction.

$$w'_{u,i} = \frac{(W_1^{(l-1)} \cdot h_u^{(l-1)})^T \cdot (W_2^{(l-1)} \cdot h_i^{(l-1)} + P \cdot t_e)}{\sqrt{d}}.$$

P is a projection matrix for encoding temporal features, which is used to map temporal embedding information t_e to a space consistent with the node embeddings. The weighting scores between users and their neighbors are obtained via the SoftMax function:

$$\beta_{u,i} = \text{softmax}\,(w'_{u,i}).$$

Thus, the temporal preference of user can be obtained by aggregating the information from its all neighbors adaptively:

$$h_u^T = \sum\nolimits_{i \in \mathcal{N}_u} (\alpha_{ui}(W_1^{(l-1)} h_i^{(l-1)}) + P \cdot t_e).$$

3.3.2. Node updating

In this stage, we aggregate the interpretability embedding and temporal embedding to update the node's embedding of $G_t(u)$. The embedding updating rule from $l-1$-th layer to l-th layer can be formulated as:

$$h_u^{(l)} = \tanh\,(\mathrm{W}_3^{(l)}[h_u^I \parallel h_u^T \parallel h_u^{(l-1)}]).$$

where $W_3^{(l)} \in \mathbb{R}^{d \times 3d}$ is a user update matrix to control the information of h_u^I, $\$\mathrm{h}_u^T$ and $h_u^{(l-1)}$.

3.3.3. LSTM mechanism for subgraphs' sequences

LSTM is a powerful time series modelling tool that captures long-term dependencies and is therefore suitable for dealing with dynamically changing nodes or subgraphs in a sequence of snapshots. Suppose the DG is decomposed into T subgraphs $G_1, G_2, \ldots, G_T$ and the graph features of each subgraph are represented as node embedding matrix $H_t \in \mathbb{R}^{N\times d}$, where N is number of nodes, d is embedding dimensions. Use the graph features H_t as an input to the time series:

$$X_{\mathrm{LSTM}} = \{H_1, H_2, \ldots, H_T\}.$$

We design a model that uses three LSTM modules to process subgraph sequences far, middle and near in time respectively, and then use a fusion LSTM module to aggregate the outputs of these three modules, which can efficiently model the dynamical properties of different time scales. When LSTM deals with subgraph sequences, direct input of node embedding matrix leads to mismatch of input shapes if the number of nodes at different time steps. The computational complexity is higher when the number of nodes is larger. To facilitate the problem of dynamic number of nodes and reduce the computational complexity, we first use Mean Pooling processing to transform the node embedding matrix H_T into a single embedding vector at the graph level:

$$\mathrm{s}_t = \frac{1}{N}\sum_{v\in\mathcal{V}} \mathrm{h}_v.$$

Meanwhile Pooling processing for DRL-compliant dynamic recommendation tasks requires the features of the entire subgraph as a state $\mathrm{s}_t \in \mathbb{R}^d$ input. Sequences of node embeddings from different time periods are fed into three separate LSTM modules as shown in **Figure 3**, LSTM1, 2 and 3:

$$s_{\mathrm{long}} = \mathrm{LSTM}_{\mathrm{long}}\left(\{s_1, s_2, \ldots, s_k\}\right).$$

$$s_{\mathrm{mid}} = \mathrm{LSTM}_{\mathrm{mid}}\left(\{s_{k+1}, \ldots, s_m\}\right).$$

$$s_{\mathrm{short}} = \mathrm{LSTM}_{\mathrm{recent}}\left(\{s_{m+1}, \ldots, s_T\}\right).$$

where $\mathrm{s}_{long} \in \mathbb{R}^d$, $\mathrm{s}_{mid} \in \mathbb{R}^d$, $\mathrm{s}_{recent} \in \mathbb{R}^d$. Finally, the LSTM outputs of the three time periods are fed into the fusion LSTM4 as a sequence shown in **Figure 3**:

$$s_{\mathrm{final}} = \mathrm{LSTM}_{\mathrm{fusion}}\left([s_{\mathrm{long}}, s_{\mathrm{mid}}, s_{\mathrm{recent}}]\right).$$

where $\mathrm{s}_{fusion} \in \mathbb{R}^{N\times 3d}$, $\mathrm{s}_{final} \in \mathbb{R}^d$.

The LSTM updates the hidden state h_t by the input s_t at time step t. The output of the TIGNN is input to a LSTM unit, and output of the LSTM unit is the feature map. The LSTM unit can be defined as:

$$\begin{aligned}
\mathrm{f}_t &= \sigma(\mathrm{W}_f \cdot [\mathrm{h}_{t-1}, \mathrm{s}_t] + \mathrm{b}_f),\\
\mathrm{i}_t &= \sigma(\mathrm{W}_i \cdot [\mathrm{h}_{t-1}, \mathrm{s}_t] + \mathrm{b}_i),\\
\mathrm{o}_t &= \sigma(\mathrm{W}_o \cdot [\mathrm{h}_{t-1}, \mathrm{s}_t] + \mathrm{b}_o),\\
\mathrm{c}_t &= \mathrm{f}_t \odot \mathrm{c}_{t-1} + \mathrm{i}_t \odot \tilde{\mathrm{c}}_t,\\
\mathrm{h}_t &= \mathrm{o}_t \odot \tanh(\mathrm{c}_t),
\end{aligned}$$

where f_t refers to the forgotten gate, i_t refers to the input gate, o_t refers to the output gate, and h_{t-1} and c_{t-1} are the hidden and cell states of the previous time $t-$

1 step. The feature map can be used as input for subsequent intelligent decision-making tasks for DRL recommender systems.

3.4. PPO DRL for intelligent decision-making optimization

Although GNNs have advantages in modelling graph structure information, DRLs are more advantageous in dynamic recommender systems, especially in tasks that need to deal with user behavioral changes and long-term decision optimizations. DRL can automatically learn an adaptable recommendation strategy, continuously adjust the recommendation strategy based on user feedback in real-time interactions, and optimize the long-term return of the recommendation strategy, rather than just optimizing based on current feedback.

By using LSTM outputs as DRL states and inputting them into the PPO Agent, the temporal dynamics of the dynamic subgraphs can be captured, and LSTM provides a time-sensitive embedding of the graphs, while the PPO Agent optimizes the recommendation strategy on this basis, and dynamically adjusts it in conjunction with the user's feedback signals, ultimately realizing intelligent decision-making in an efficient personalized recommendation system. (**Figure 4**)

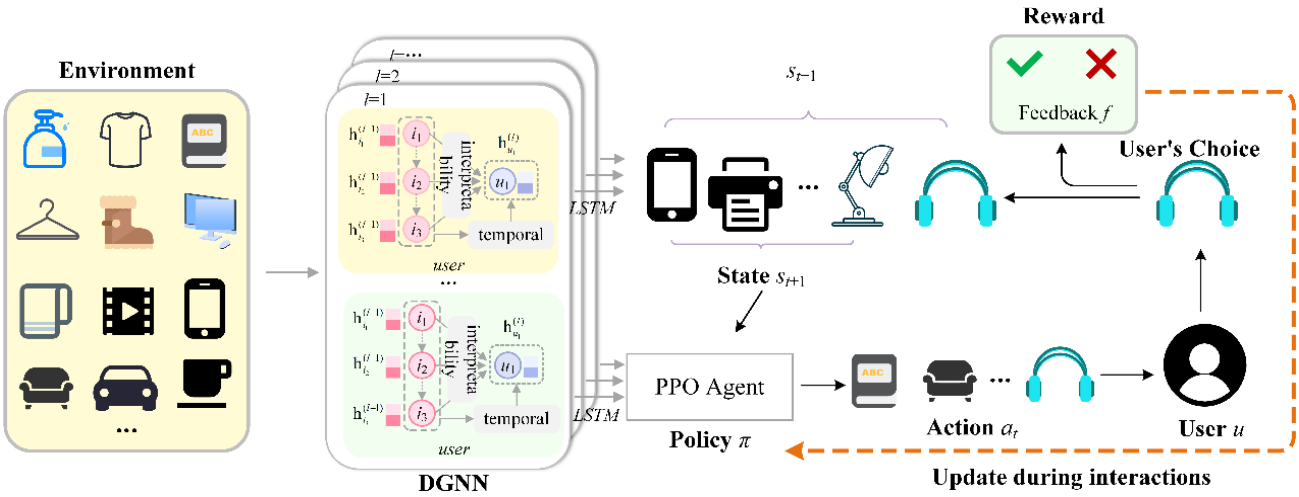


Figure 4. Temporal-aware Dynamic Graph Neural Network Recommendation Framework with PPO Optimization.

3.4.1. PPO agent

PPO uses state s_t as input, outputs recommended action distributions through a policy network, and predicts state values through a value network. The PPO agent consists of two main components: the policy network π_θ and the value network V_θ. The policy network outputs a probability distribution over actions a_t given the state s_t:

$$\pi_\theta(a_t|s_t) = \text{Softmax}(W_p \cdot s_t + b_p).$$

where W_p and b_p are learnable parameters. The value network estimates the expected return of the state s_t:

$$V_\theta(s_t) = W_v \cdot s_t + b_v.$$

where W_v and b_v are learnable parameters. The agent selects an action a_t by sampling from the policy:

$$a_t \sim \pi_\theta(a_t|s_t).$$

3.4.2. Reward mechanism

Reward r_t is defined based on user feedback signals. The reward r_t is computed based on user feedback after executing the action a_t. The reward function combines multiple feedback signals:

$$r_t = w_1 \cdot \text{Click} + w_2 \cdot \text{Purchase} + w_3 \cdot \text{Dwell\_Time}.$$

where w_1, w_2, w_3 are weights that balance the importance of each feedback type. The reward r_t is used to compute the advantage function A_t for PPO training:

$$A_t = r_t + \gamma V_\theta(s_{t+1}) - V_\theta(s_t).$$

where γ is the discount factor.

Algorithm 1 time-interpretable dynamic recommender system intelligent decision-making

1: Input: user-item interaction graph $G_t = (V_t, E_t)$, user feature vector x_v, interaction feature vector x_e, timestamp t
2: Output: action Distribution $\pi_\theta(a \mid s_t)$, state value $V_\theta(S_t)$, action a_t, reward r_t
3: 1. Temporal Encoding: $t_e = PE(t - t_p rev)$
4: $x_e = Concat(x_e, t_e)$
5: 2. Subgraph Extraction: $\mathcal{N}_k(u) = v \in V_t \mid d(u,v) \le k$
6: $E_u = (v, w, t) \in E_t \mid v, w \in \mathcal{N}_k(u), t \ge t_{\text{threshold}}$
7: 3. Subgraph Updated:
8: features evaluation function: $f_{node}(v,t)$, $f_{edge}(e,t)$ and $f_{subgraph}(G_u, t)$
9: cropping the adjacency matrix: $\mathbf{A}_u = \mathbf{A}[V_u', V_u']$
10: 4. TIGNN:
11: message propagation:
12: considering interpretability: $w_{u,i} = \frac{(\mathbf{W}_1^{(l-1)} \cdot \mathbf{h}_u^{(l-1)})^T \cdot (\mathbf{W}_2^{(l-1)} \cdot \mathbf{h}_i^{(l-1)} + \tilde{\pi}_{e_{u,i}}^{u})}{\sqrt{d}}$
13: $\mathbf{h}_u^I = \sum_{i \in \mathcal{N}_u} (\alpha_{ui} (\mathbf{W_1}^{(l-1)} \mathbf{h}_i^{(l-1)}) + \tilde{\pi}_{e_{u,i}}^{u})$
14: considering temporal: $w_{u,i}' = \frac{(\mathbf{W}_1^{(l-1)} \cdot \mathbf{h}_u^{(l-1)})^T \cdot (\mathbf{W}_2^{(l-1)} \cdot \mathbf{h}_i^{(l-1)} + \mathbf{P} \cdot t_e)}{\sqrt{d}}$
15: $\mathbf{h}_u^T = \sum_{i \in \mathcal{N}_u} (\alpha_{ui} (\mathbf{W_1}^{(l-1)} \mathbf{h}_i^{(l-1)}) + \mathbf{P} \cdot t_e)$
16: node updating: $\mathbf{h}_u^{(l)} = \tanh(\mathbf{W_3}^{(l)} [\mathbf{h}_u^I \parallel \mathbf{h}_u^T \parallel \mathbf{h}_u^{(l-1)}])$
17: 5. LSTM:
18: pooling node embedding matrix: $\mathbf{s}_t = \frac{1}{N} \sum_{v \in \mathcal{V}} \mathbf{h}_v$
19: far middle and near future: $\mathbf{s}_{\text{final}} = \text{LSTM}_{\text{fusion}}([\mathbf{s}_{\text{long}}, \mathbf{s}_{\text{mid}}, \mathbf{s}_{\text{recent}}])$
20: 6. PPO DRL:
21: PPO agent: $\pi_\theta(a_t|s_t) = \text{Softmax}(\mathbf{W}_p \cdot s_t + \mathbf{b}_p)$
22: value state: $V_\theta(s_t) = \mathbf{W}_v \cdot s_t + \mathbf{b}_v$
23: Reward: $A_t = r_t + \gamma V_\theta(s_{t+1}) - V_\theta(s_t)$
24: 7. Loss function: $L^{\text{PPO}} = L^{\text{clip}} - c_1 \cdot L^{\text{value}} + c_2 \cdot L^{\text{entropy}}$
25: 8. Joint training: $L^{\text{PPO}} \to \pi_\theta(a_t \mid s_t), V_\theta(s_t) \to s_t \to \text{TIGNN}_{\text{parameter}}$

3.5. Training and evaluation

In the absence of a distinct TIGNN loss function, the gradient can be propagated through the PPO's loss function, enabling the GNN to learn embedding vectors that are conducive to the PPO's effective execution of the recommendation task. The loss function of PPO consists of three parts: clipped surrogate loss, value function loss and

entropy regularization term. Optimize the recommendation strategy so that recommended items have a higher probability with the current strategy. Learning the value of states and evaluating the long-term cumulative benefit of the current state. Increase the exploratory nature of the strategy to prevent it from converging to a local optimum prematurely. Taken together, the total loss function of the PPO is:

$$L^{\text{PPO}} = L^{\text{clip}} - c_1 \cdot L^{\text{value}} + c_2 \cdot L^{\text{entropy}}.$$

where L^{clip} is clipped surrogate loss, L^{value} is value loss, L^{entropy} is entropy regularizations term and c_1, c_2 control the importance of losses.

The loss function of PPO is back-propagated into the TIGNN through the gradient of the state s_t to optimize the parameters of the TIGNN:

$$L^{\text{PPO}} \to \pi_\theta(a_t \mid s_t), V_\theta(s_t) \to s_t \to \text{TIGNN}_{\text{parameter}}.$$

The embedding s_t of the TIGNN is automatically tuned to orientations that are more effective for the PPO recommendation task. The goal of the GNN is implicit in the goal of the PPO, which is to learn graph embeddings that enable better policy π_θ.

In this section we describe in detail the time-interpretable dynamic recommender system intelligent decision-making algorithm in Algorithm 1. We will compare and analyzes our TIGNN with some representative recommendation models in section Discussion.

4. Experiments

In this section, we perform experiments on three real-world datasets to evaluate the performance of our model. The results demonstrate the effectiveness of the system in improving recommendation accuracy, personalization, and decision-making intelligence.

4.1. Datasets

To evaluate the effectiveness of our model, we conduct experiments on three datasets from real-world:

a) Last. fm1K: The dataset is annotated with timestamps for each user's music playing behaviors and is suitable for training time-sensitive recommender systems. The small size of the dataset is ideal for validating the dynamic properties of the algorithm and training the model quickly and iteratively.

b) MovieLens 1M: User rating behaviors for films tends to be temporally dynamic. The data size is medium and suitable for validating the effectiveness of time-dynamic recommendation algorithms.

c) Amazon Product Review: User purchasing behaviors is distinctly temporal and interactions can be modelled with timestamps. The data is large and needs to be preprocessed and sampled to form a subset suitable for training.

In all of the sets of data, we treat the presence of comments or ratings as implicit feedback and exclude users and items with fewer than five relevant actions. After processed, the data statistics are shown in **Table 2**.

Table 2. Basic statistics for the three datasets.

Datasets	**Last.FM 1K**	**MovieLens 1M**	**Amazon Product Review**
#of Users	992	6,039	9,985
#of Items	1,998	3,679	1,1467
#of Interactions	25,990	576,724	97,853
Average length	26.2	95.5	9.8
Density	1.31%	2.59%	0.08%

According to the characteristics of the dataset, the division by time series can ensure the time continuity of the training set and the test set, which is in line with the workflow of the recommender system in the real scenario, and can better test the performance of the dynamic recommendation model. Last.FM 1K data is sparser, can capture the time dynamic characteristics of user interest, time dynamic recommendation. MovieLens 1M data is denser, can use the time information to capture the change of user interest, long-term interest modelling. Amazon Product Review data is very sparse, the long-tail effect is obvious, suitable for long-tail recommendation.

4.2. Experiments setting

4.2.1. Baselines

We compare the proposed TIGNN with the following, all of them use GNNs for augmenting DRL-based RS, except for the first one, which is a GCN approach:

KGCN: propose a variant of GCN to learn the embedding for KG.

KERL: uses a traditional graph embedding method $TransE$ to generate the state representation for DRL-based RS.

DGN: propose a graph convolutional RL (DGN) method which integrates the GCN into the Q-learning framework for general RL problems by replacing the state encoding layer with the GCN layer.

GCQN: extend this method into the deep learning field and apply it to recommender systems.

KGRL: employs KG inside the actor-critic algorithm to help the agent learn the policy.

RLfD: adopts a Graph Attention Network (GAT) into the actor-critic network to conduct recommendation.

4.2.2. Evaluation metrics

We adopt two widely-used metrics, NDCG@K and Hit@K, to evaluate all methods.

NDCG@K is a ranking indicator, and higher NDCG means target items tend to have more top rank positions, while Hit@K only cares if the ground-truth item appears in Top-K and does not take into account the sorting quality of the other items in the recommended list.

For each test sample, we randomly sample 100 negative items, and rank these items with the ground-truth item. We evaluate Hit@K and NDCG@K based on these 101 items. By default, we set $K = 10$.

4.2.3. Parameter setup

We implement our model in DGL + PyTorch. The maximum subgraphs length n is set to 50. The embedding size is fixed to 50 for all methods. All trainable parameters are optimized by Adam algorithm, the learning rate is set to 0.01. Batch size is 50. λ is $1e-4$. The TIGNNN layer number L is set to 3 for Last.FM and MovieLens, 2 for Amazon. For each dataset, the ratio of training, evaluation, and test set is 6: 2: 2. Each experiment is repeated 3 times, and the average performance is reported. For the compared methods, we use the default hyperparameters except for dimensions.

4.3. Results

4.3.1. Performance comparison

We use **Table 3** to present the sorting quality of recommended lists NDCG@10 values and recommended list hit rate Hit@10 values for the recommender system TOP-10 predictions, respectively (other variants are not plotted in the graphs for clarity). We have the following observations:

Table 3. Performance of TIGNN and compared methods in terms of Hit@10 and NDCG@10.

Datasets	Metric	KGCN	KERL	DGN	GCQN	KGRL	RLfD	TIGNN	Gain
Last.FM	NDCG@10	22.10	29.00	33.10	32.80	32.95	31.00	36.50	10.27%
	Hit@10	38.00	44.00	49.10	48.80	49.00	47.00	53.00	7.95%
MovieLens	NDCG@10	29.00	47.00	54.00	53.90	50.10	51.20	56.80	5.19%
	Hit@10	38.00	68.00	74.00	73.50	72.20	72.50	76.00	2.70%
Amazon	NDCG@10	37.00	34.00	50.00	49.50	50.30	50.20	52.00	3.40%
	Hit@10	57.00	52.00	72.00	70.50	72.50	71.50	73.80	1.80%

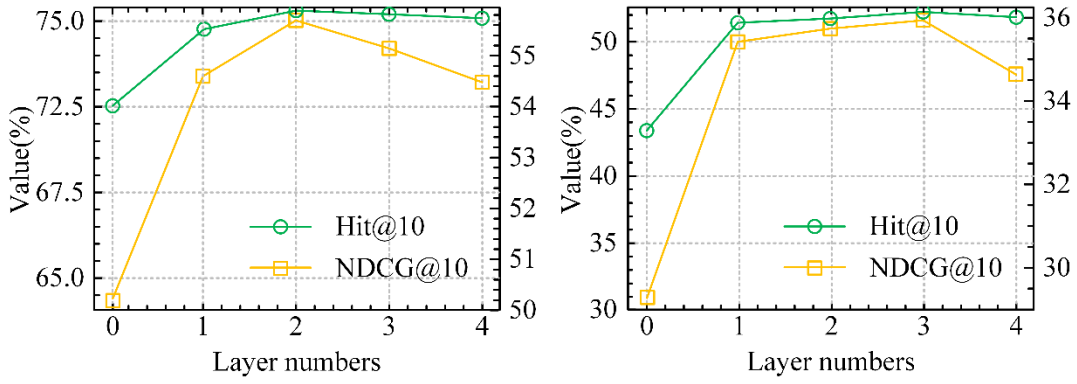
Note: "Gain" means the improvement over the best compared methods.

TIGNN-RL achieves the best performance on three datasets with most evaluation metrics. In particular, DGSR improves over the strongest baselines w.r.t NDCG@10 by 10.27%, 5.19%, 3.40% in Music, Movies, and Shopping, respectively. Notably, Last.FM is the most sparse and short dataset, so many users and items only have a few interactions. In our model, the high-order connectivity of a dynamic graph alleviates this issue. So, there is a significant improvement in Last.FM. By stacking the TIGNN layers, TIGNN can utilize time-sensitive and context-aware information explicitly to provide more auxiliary information for prediction. While KERL, DGN, GCQN, and KGRL only encode each information independently as the user's dynamic interest representation. Significantly, RLfD utilizes many correlated user interaction information, but performs worse than our DGSR, especially on Music and Movies. We believe that the reason is RLfD ignores the refine interaction context-aware information of subgraphs. And Music and Movies have stronger context-aware properties than Amazon, resulting in significant improvement in the performance of TIGNN over RLfD on Music and Movies.

4.3.2. The sensitivity of hyper-parameters

To explore the effect of temporal and interpretability user-interactions information on TIGNN, we study how two hyperparameters, the TIGNN layer number l and the maximum length of subgraph n affect the performance of TIGNN.

We conduct our method with different TIGNN layer number l on Amazon and Last.FM datasets. TIGNN-0 represents only use user embedding for recommendation. DGSR-1 represents the DGRN with one layer, indicating to use first-order neighbor samples and fixed number of neighbor node samples information perform an aggregation for prediction.



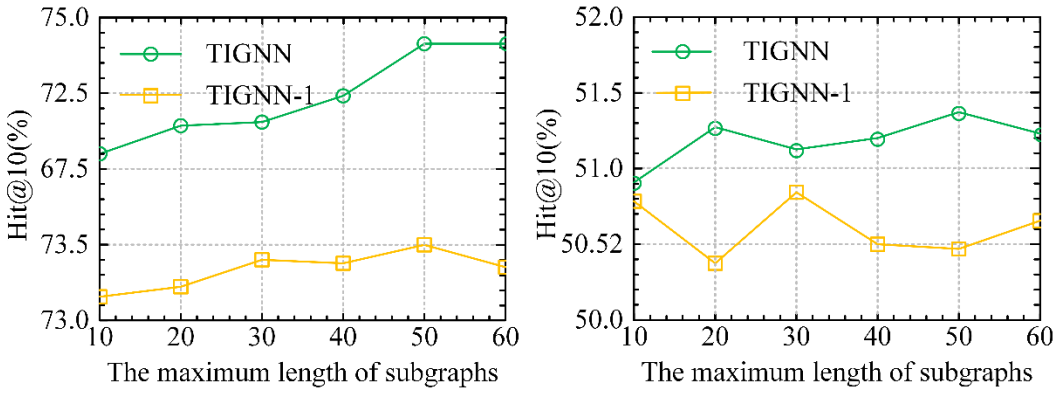
(a) Last.FM (b) Amazon

Figure 5. Effect of propagation layer numbers (the y-axis on the left is Hit@10 value, and the right is NGCD@10 value).

From **Figure 5**, we find that increasing the layer of TIGNN is capable of promoting the performance substantially. It demonstrates that exploiting high-order user-interactions information explicitly can effectively improve recommendation performance. TIGNN-2 and TIGNN-3 achieve the best performance on Last.FM and Amazon, respectively. The possible reason is that Amazon is sparser than Last.FM, a larger number of layers may be required to introduce a more contextual information.

When further stacking propagation layer, we find that the performance of DGSR-3 and DGSR-4 begin to deteriorate. The reason might be that the use of far more propagation layers may lead to over smoothing.

We train and test our method on the Last.FM and Amazon datasets with n from 10 to 60, while keeping other optimal hyperparameters unchanged. Besides, to further investigate the benefit of information, we also conduct TIGNN-1 with different n. **Figure 6** shows the Hit@10 results.



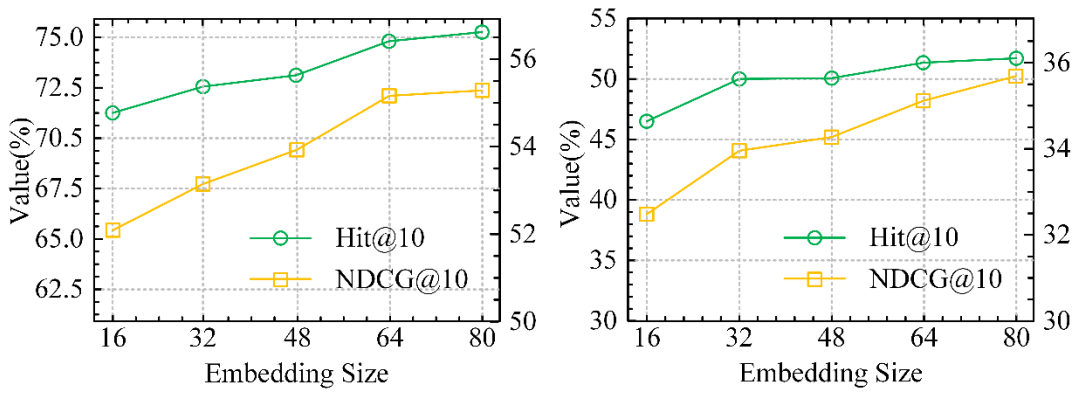
(a) Amazon (b) Last.FM

Figure 6. Effect of the maximum length of subgraphs.

Increasing the n of TIGNN from 10 to 50 consistently improves the performance of Amazon data. TIGNN performs better on the Last.FM when set n to be 20 and 50. However, blindly increasing the n does not necessarily improve the performance of TIGNN and TIGNN-1. It is likely to bring noise and cause the performance to attenuate.

Compared with TIGNN-1, TIGNN performs better than TIGNN-1 at each value of n. To be specific, even when n is set to 10, TIGNN is still better than the best performance of TIGNN-1, which implies that explicitly utilizing high-order information of user can alleviate the issue of insufficient user history information, thus improving the performance of recommendation.

We further analyze the impact of different dimensionality of embedding size. **Figure 7** describes the performance of model under the embedding size from 16 to 80. We can observe that the performance of model gradually improves as the dimensionality increases. With the further increase of the dimensionality, the performance tends to be stable. This verifies the stability of our model in different dimensions.



(a) Amazon (b) Last.FM

Figure 7. Effect of the embedding size.

5. Discussion and future directions

The experimental results demonstrate the effectiveness of the proposed Temporal Interpretability Graph Neural Network with Reinforcement Learning (TIGNN-RL) framework in capturing dynamic user preferences and balancing short-term accuracy with long-term user engagement. In this section, we discuss the key findings, challenges, and implications of this work. Additionally, integrating biomechanics knowledge is also explored to reshape and enhance these aspects

5.1. Key contributions

Temporal Adaptability: By modeling user preferences dynamically using temporal encoding and reinforcement learning, TIGNN-RL effectively adapts to both short-term and long-term preference shifts, which is critical in real-world recommendation scenarios. When considering biomechanics, in applications like fitness recommendation systems, this adaptability becomes more significant. For instance, as a user engages in different physical activities over time, their body's

biomechanical responses will change. The muscles adapt, and the flexibility may improve or change. The user's physical capabilities and comfort levels with various exercises will also evolve. TIGNN-RL can potentially leverage this knowledge to adjust recommendations in real-time. If a user initially prefers low-impact exercises at first but gradually builds up strength later, the system will adapt and suggest more intense workouts that align with their new biomechanical state.

Explainability: The temporal interpretability module provides insights into the temporal importance of user-item interactions, enabling explainable recommendations, which are becoming increasingly valuable in recommendation system applications. And with the context of biomechanics, this feature will be extremely beneficial. In a sports equipment recommendation scenario, the system can explain why the equipment is recommended based on the user's previous biomechanical interactions with similar equipment. This will enhance the transparency of the recommendation process and build trust between the user and the system.

5.2. Challenges

Scalability: While TIGNN-RL achieves strong performance on mid-sized datasets, scaling the framework to very large datasets (e.g., hundreds of millions of users and items) may be computationally expensive due to the iterative nature of GNN message passing and reinforcement learning updates. Under the background of biomechanics, the problem becomes more compounded. Biomechanical data is often complex and high-dimensional. It involves factors like body dimensions, movement patterns, and physiological responses. Incorporating such data into the TIGNN-RL framework for large datasets would require more computational resources. Thus, it is needed to explore efficient algorithms and distributed computing techniques to address this scalability issue. Additionally, it is also worthwhile to explore the emerging graph computation optimization algorithms and cloud-based distributed training frameworks, and analyze their applicability and potential improvement directions within the TIGNN-RL framework.

Cold Start Problem: Although TIGNN-RL leverages dynamic graphs and temporal embeddings, its performance on cold-start scenarios, where new users or items have very limited historical data, is not fully optimized. However, their biomechanical characteristics, such as body type, muscle strength, and flexibility, play a crucial role in making accurate recommendations. But integrating this biomechanical information into the TIGNN - RL framework during the cold start phase is a challenge and requires further research.

Tradeoff Between Accuracy and Diversity: Balancing accuracy and diversity remains a challenge. While TIGNN-RL improves diversity without significant loss of accuracy, real-world applications might require further fine-tuning to meet specific business or user objectives.

5.3. Implications

Personalized Recommendation: By leveraging dynamic graphs and temporal interpretability, TIGNN-RL can generate highly personalized recommendations tailored to evolving user preferences.

Fairness and Diversity: Improved diversity metrics suggest that TIGNN-RL can mitigate popularity bias, making it suitable for applications requiring fairness and inclusivity.

Real-Time Applications: The framework's ability to adapt to dynamic user behaviors makes it a strong candidate for real-time recommendation systems, such as music streaming platforms or e-commerce.

6. Conclusions

The proposed TIGNN-RL framework represents a significant step toward dynamic, interpretable, and effective recommendation systems. By combining temporal modeling, graph neural networks, and reinforcement learning, it addresses several key challenges in modern recommender systems. However, further research is needed to improve scalability, interpretability, and applicability to cold-start scenarios and other domains. By understanding and leveraging the complex relationship between user behaviors and their biomechanical characteristics, the TIGNN - RL framework can be optimized to provide more accurate, personalized, and context-aware recommendations. With continued advancements, this framework has the potential to set a new standard for intelligent decision-making in recommender systems.

The introduction of the TIGNN-RL framework is expected to bring significant breakthroughs to the field of recommendation systems, offering not only improved performance for existing platforms but also laying the groundwork for intelligent recommendation methods in emerging domains such as healthcare, financial technology, and education. By dynamically adapting to user behavior and leveraging advanced graph-based representations, this framework can address challenges like cold start problems and dynamic preference shifts more effectively.

Looking ahead, future research could explore the integration of TIGNN-RL with other transformative technologies, such as blockchain for secure and decentralized recommendations or quantum computing to enhance computational efficiency in large-scale systems. These explorations could significantly expand the functionality and application scope of recommendation systems, enabling smarter, more efficient, and personalized services across diverse fields. Such advancements would further bridge the gap between theoretical innovation and practical application, paving the way for new possibilities in intelligent decision-making and user-centric technologies. The introduction of the TIGNN-RL framework is expected to bring new breakthroughs to the field of recommendation systems, not only enhancing the performance of existing recommendation systems but also potentially providing new ideas and methods for the application of intelligent recommendations in emerging fields (such as fintech, healthcare, etc.). With the continuous development of technology, future explorations could further investigate the integration with other emerging technologies (such as quantum computing, blockchain, etc.), expanding the functionality and application scope of recommendation systems to achieve smarter, more efficient, and personalized recommendation services.

Author contributions: Conceptualization, HY and CY; methodology, YH; software, CY; validation, HY and CY; investigation, HY; resources, CY; data curation, CY; writing—original draft preparation, HY; writing—review and editing, CY; visualization, HY; supervision, HY; project administration, HY; funding acquisition, CY. All authors have read and agreed to the published version of the manuscript.

Ethical approval: Not applicable.

Conflict of interest: The authors declare no conflict of interest.

References

1. F. Ricci, L. Rokach, B. Shapira, P. B. Kantor. Recommender systems survey. Knowledge-Based Systems; 2013. doi：10.1016/j.knosys.2011.12.005
2. C. Wang, M. Zhang, W. Ma, et al. Modeling item-specific temporal dynamics of repeat consumption for recommender systems. In: Proceedings of the World Wide Web Conference; 2019.
3. Yunzhi Tan, Min Yang, Chengming Li, Ruifeng Xu, Yating Liu. A time-aware graph neural network for session-based recommendation. IEEE Access: Practical Innovations, Open Solutions; 2020. DOI: 10.1109/ACCESS.2020.3015480
4. Yan Zhao, Chong Chen, Xiangyu Liu, Ziliang Zhao. Two-Stage Sequential Recommendation for Side Information Fusion and Long-Term and Short-Term Preferences Modeling. Journal of Intelligent Information Systems. 2022. DOI: 10.1007/s10844-021-00683-4
5. J. Tang, K. Wang, H. Liu, et al. Time-sensitive recommendation from recurrent user activities. Advances in Neural Information Processing Systems; 2015.
6. H. Yu, J. Jannach. STAMP: Short-Term Attention / Memory Priority Model for Session-Based Recommendation. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining; 2018. doi：10.1145/3219819.3220003
7. Suhaim AB, Berri J. Context-Aware Recommender Systems for Social Networks: Review, Challenges and Opportunities. IEEE Access. 2021; 9: 57440–57463. doi: 10.1109/access.2021.3072165
8. Venkatachalam P, Ray S. How do context-aware artificial intelligence algorithms used in fitness recommender systems? A literature review and research agenda. International Journal of Information Management Data Insights. 2022; 2(2): 100139. doi: 10.1016/j.jjimei.2022.100139
9. Zhou S, Hudin NS. Advancing e-commerce user purchase prediction: Integration of time-series attention with event-based timestamp encoding and Graph Neural Network-Enhanced user profiling. Zhu J, ed. PLOS ONE. 2024; 19(4): e0299087. doi: 10.1371/journal.pone.0299087
10. Hou Z, Bu F, Zhou Y, et al. DyCARS: A dynamic context-aware recommendation system. Mathematical Biosciences and Engineering. 2024; 21(3): 3563–3593. doi: 10.3934/mbe.2024157
11. Ali W, Kumar J, Mawuli CB, et al. Dynamic context management in context-aware recommender systems. Computers and Electrical Engineering. 2023; 107: 108622. doi: 10.1016/j.compeleceng.2023.108622
12. Ma L, Chen Z, Fu Y, Li, Y. Heterogeneous Graph Neural Network for Multi-behavior Feature-Interaction Recommendation. In: Pimenidis E, Angelov P, Jayne C, et al. (editors). Artificial Neural Networks and Machine Learning—ICANN 2022. Springer, Cham; 2022.
13. Jianxin Chang, Chen Gao, Xiangnan He, Depeng Jin, Yong Li. Adaptive user modeling with long and short-term preferences for personalized recommendation. IJCAI; 2019. DOI: 10.24963/ijcai.2019/375
14. Jeong SY, Kim YK. Deep Learning-Based Context-Aware Recommender System Considering Change in Preference. Electronics. 2023; 12(10): 2337. doi: 10.3390/electronics12102337
15. Zhiheng Liu, Le Wu, Lei Chen, Richang Hong. Dynamic time-aware collaborative sequential recommendation with attention-based network. Knowledge and Information Systems; 2023. DOI: 10.1007/s10115-023-01857-y
16. Xiaokun Zhang, Bo Xu, Liang Yang, Hongfei Lin. Time interval-aware graph with self-attention for sequential recommendation. In: Proceedings of the 2022 5th International Conference on Algorithms, Computing and Artificial Intelligence; 2023. DOI: 10.1145/3576836.3576852

17. B. Hidasi, A. Karatzoglou, L. Baltrunas, D. Tikk. Session-based Recommendations with recurrent neural networks. doi: 10.48550/arXiv.1511.06939.
18. Wang L, Jin D. A Time-Sensitive Graph Neural Network for Session-Based New Item Recommendation. Electronics. 2024; 13: 223. doi:10.3390/electronics13010223
19. Shan G. Exploring the intersection of equipment design and human physical ability: Leveraging biomechanics, ergonomics/anthropometry, and wearable technology for enhancing human physical performance. Advanced Design Research. 2023; 1(1): 7–11. doi: 10.1016/j.ijadr.2023.04.001
20. Z. Wu, S. Pan, F. Chen, et. al. Graph Neural Networks: A Review of Methods and Applications. AI Open; 2018.
21. R. Chen, Y. Rubanova, J. Bettencourt, D. Duvenaud. Deep learning for sequential recommendation: algorithms, influential factors, and evaluations. doi:10.1145/342672
22. R. Chen, Y. Rubanova, J. Bettencourt, D. Duvenaud. Latent ODEs for irregularly-sampled time series. doi: 10.48550/arXiv.1907.03907.