electronics

MDPI

Article

A Reinforcement Learning List Recommendation Model Fused with Graph Neural Networks

Zhongming Lv and Xiangrong Tong *

College of Computer and Control Engineering, Yantai University, Yantai 264005, China; lzm_lv@163.com
* Correspondence: txr@ytu.edu.cn

Abstract: Existing list recommendation methods present a list consisting of multiple items for feedback recommendation to user requests, which has the advantages of high flexibility and direct user feedback. However, the structured representation of state data limits the embedding of users and items, making them isolated from each other, missing some useful infomation for recommendation. In addition, the traditional non-end-to-end learning series takes a long time and accumulates errors. During the model training process, the results of each task can easily affect the next calculation, thus affecting the entire training effect. Aiming at the above problems, this paper proposes a Reinforcement Learning List Recommendation Model Fused with a Graph Neural Network, GNLR. The goal of this model is to maximize the recommendation effect while ensuring that the list recommendation system accurately analyzes user preferences to improve user experience. To this end, firstly, we use an user–item bipartite graph and Graph Neural Network to aggregate neighborhood information for users and items to generate graph structured representation; secondly, we adopt an attention mechanism to assign corresponding weights to neighborhood information to reduce the influence of noise nodes in heterogeneous information networks; finally, we alleviate the problems of traditional non-end-to-end methods through end-to-end training methods. The experimental results show that the method proposed in this paper can alleviate the above problems, and the recommendation hit rate and accuracy rate increase by about 10%.

Keywords: list recommendation; deep reinforcement learning; Graph Neural Network; Actor–Critic; state representation module

Citation: Lv, Z.; Tong, X. A Reinforcement Learning List Recommendation Model Fused with Graph Neural Networks. *Electronics* **2023**, *12*, 3748. https://doi.org/10.3390/electronics12183748

Academic Editor: Stefanos Kollias

Received: 7 August 2023
Revised: 25 August 2023
Accepted: 2 September 2023
Published: 5 September 2023

1. Introduction

Recommendation systems [1], as a personalized service, can recommend information and products (hereafter collectively referred to as 'items') of interest to users to relieve information pressure. It is widely used in practical scenarios such as e-commerce, music and movies, news recommendation, and social networks. List recommendation [2] is a recommendation method that can be used to achieve diverse and complementary item selection. To enable list recommendation systems to recommend items that are interesting, diverse, and stable for users, several DRL recommendation methods that can recommend lists containing multiple items have been proposed in previous work.

In existing list recommendation models, a single request from a user usually recommends a list of items that can provide diverse and complementary choices to the user. For list-based recommendation, we have a list-based space where each operation is a collection of interdependent sub-operations (items). Existing reinforcement learning recommendation methods can also recommend lists of items. For example, Deep Q Network (DQN) can compute the Q-values of all recall items and recommend a list of top-ranked items. However, these methods recommend items based on the same state, ignoring the connection between users and items, and the embedding vector is difficult to represent unstructured data completely, making the data structured, which may lead to inaccurate calculations of similarity between users and items, thus affecting the recommendation effect.

In addition, the traditional non-end-to-end recommendation model has a long crosstalk time as well as error accumulation and requires precise design; the result of each step in the training process will have an impact on the next calculation; and when new nodes are added, the training needs to be repeated, which increases the training complexity and reduces the recommendation performance.

Graph Neural Networks (GNNs) [3] are a special class of deep neural networks that can learn and make inferences on graph data. Compared with traditional deep neural networks, GNNs have two advantages: graph structured representation capabilities and end-to-end training capability. GNNs can perform graph structured representation on graph data, i.e., vectorization of nodes and edges, thus capturing the structural information between nodes and edges, making the feature representation of graph data richer and more accurate. In addition, GNNs are capable of directly processing graph data, inputting graph data into the network, and finally outputting the feature representation or prediction results of nodes or edges after processing and learning by multilayer neural networks.

A Reinforcement Learning List Recommendation Model Fused with Graph Neural Networks, GNLR, is proposed. In GNLR, firstly, a user–item interaction bipartite graph is constructed, and GNNs are used to generate graph structure representations of users and items. Secondly, through the reinforcement learning approach, the recommendation system is considered as an agent and the user is considered as an environment, and a list of items is recommended to meet the user's needs. Based on a deep deterministic policy gradient (DDPG) approach, a set of items suitable for recommending to the user is derived by calculating the item scores. Lastly, based on the end-to-end training strategy, it can effectively alleviate the shortcomings of traditional non-end-to-end methods and eventually provide users with more accurate item recommendation lists.

In summary, the main contributions of this paper are as follows:

(1) A reinforcement learning list recommendation model incorporating GNN, GNLR, is proposed to enhance state representation and improve recommendation accuracy by generating graph-structured representations of users and items.
(2) An attention mechanism is introduced in GNLR to reduce the influence of noisy nodes, a multilayer network is designed for training, and an end-to-end approach is used to learn directly from the data, alleviating the drawbacks of traditional non-end-to-end model training.
(3) The proposed model is shown to outperform other baseline algorithms in terms of performance through reliable experiments on real-world datasets.

2. Related Work

2.1. Traditional Recommendation System

With the growing demand of the Internet, recommendation systems are developing rapidly. In the past, recommendation systems mainly used single models, such as collaborative filtering (CF) [4] and logistic regression (LR) [5]. With the emergence of combined models such as factorization machine (FM) and gradient boosting decision tree (GBDT) [6], the models of recommendation systems began to evolve. Starting from 2015, deep reinforcement learning recommendation models have rapidly evolved, and various new model architectures have emerged. Since 2015, deep reinforcement learning recommendation models have evolved rapidly, with various model architectures emerging. A recommendation system is an information search task whose goal is to recommend the goods or items that best match the user's needs and preferences. Most existing recommendation systems are built using supervised learning-based algorithms. Specifically, the task of a traditional recommendation system consists of three steps: collecting user behavior data, performing feature engineering, and implementing recommendation algorithms. Initially, content-based recommendations were used to recommend items by considering the content similarity between items. Later, CF was proposed and widely studied. The basic principle behind CF is that users with similar behaviors tend to select the same items and have the same ratings for similar items. However, traditional CF-based approaches tend to suffer

from data sparsity, as similarity computed from sparse data is unreliable. The above two problems are solved in matrix decomposition. Matrix decomposition can be intuitively understood as the approximate decomposition of the original large matrix into the form of multiplying two small matrices together. In the actual recommendation calculation, instead of using the original large matrix, the decomposed two small matrices are used to perform the calculation. Each user corresponds to a k-dimensional vector, and each item corresponds to a k-dimensional vector. After multiplying the two matrices, the predicted rating of any user for any item is obtained. Logistic regression and its variants treat recommendation as a binary classification problem; however, logistic regression-based models are difficult to generalize for feature interactions of training data. Factorization machines model pairwise feature interactions as inner products of potential vectors between features and have shown excellent performance. As an extension of FM, Field-aware Factorization Machine (FFM) introduces the concept of a domain; i.e., for each feature, there is a domain label in addition to the feature label, and this operation allows FFM to better capture the strength of the contribution of the union of different features to the prediction. Recently, deep learning models have been applied to model complex feature interactions for recommendation. As a prominent direction, contextual multi-armed slots have also been used to model interactions in recommendation systems. Li et al. [7,8] applied Thompson sampling (TS) and upper confidence bound (UCB) to balance between exploration and exploitation. Wang et al. [9] proposed a dynamic contextual drift model to solve the time-varying reward problem. To combine item and user embeddings with some exploration, Zhao et al. [10] combined matrix decomposition and a multi-armed slot machine. However, the above approaches have two drawbacks. First, they consider the recommendation process as a static process; i.e., the user's preferences are assumed to remain constant, and the goal is to know the user's preferences as precisely as possible. Second, they ignore the long-term benefits that recommendations can bring and focus only on maximizing immediate returns.

2.2. Deep Reinforcement Learning-Based Recommendation Systems

Some preliminary work has been carried out before the rise of DRL; however, due to data sparsity and shortcomings in computational power, these works only used deep neural networks to downscale high-dimensional input data to facilitate its processing by traditional RL algorithms. Riedmiler et al. [11] were the first to use a multilayer perceptron to approximate the Q-value function and proposed a neural fitted Q iteration (NFQ) algorithm, and Lange et al. [12] combined deep learning models and reinforcement learning methods to propose a deep auto-encoder (DAE) model; however, DAE is only applicable to control problems with visual perception as the input signal and small state space dimensions. Abtahi et al. [13] used a deep belief network as a function approximator in traditional RL, which greatly improved the learning efficiency of the agent and was successfully applied to a license plate image character segmentation task. The Deep Fitted Q-learning (DFQ) algorithm was further proposed by Lange et al. [14] and applied to vehicle control. Koutnik et al. [15] combined the neural evolution (NE) method with the RL algorithm and applied it to a video racing game to achieve the automatic driving of a car. Because of the high time complexity of model-based DRL techniques in recommendation scenarios, most researchers have turned to model-free RL techniques. These model-free RL techniques can be divided into two categories: policy-based and value-based. Q-learning is a value-based approach that provides action guidance through Q-tables but only for scenarios where the state and action space is discrete and not high dimensional. For high-dimensional continuous state and action spaces, Q-tables will become very large and difficult to maintain and find. In 2013, the Google DeepMind team [16] tried to output Q-values by fitting using functions, i.e., inputting a state and giving Q-values for different actions. Since deep learning performs well in complex feature extraction, deep learning combined with reinforcement learning was used to implement DQN. In 2015, DQN was well refined, which included improvements in experience replay techniques and fixed Q target techniques to improve the stability and efficiency of the algorithm. In the Atari

game, DQN showed amazing real-world results and a series of improved [17] and Dueling DQN [18], etc. Chen et al. [19] applied DQN to item list recommendation and proposed a RecSys algorithm. Each step of item list recommendation requires recommending a list of items, which is a combination of all items in a huge search space with a large number of locally optimal solutions. The RecSys algorithm uses a Transformer rather than a recurrent or convolutional neural network to encode embeddings to capture context-aware information in the user history, fuse these embeddings into aggregated features, and train the model using Double DQN technique and experience replay technique. Xie et al. [20] from the Tencent WeChat department proposed a hierarchical reinforcement learning based on the Actor–Critic model for recommendation systems, which is called HRL-Rec. The modeling integrated recommendation as list recommendation, which contains two RL agents. The bottom agent is a channel selector, which generates personalized channel lists based on users' channel-level preferences; the top agent acts as an item recommendation, recommending specific heterogeneous items under the constraints given at the channel level. And a new set of reward functions is designed to measure the accuracy and diversity of the task, combining the supervised loss and unsupervised similar loss at the bottom and top levels to achieve fast and stable training. For the previous work related to list recommendations, one of these methods is DQN, which calculates the Q-values of all the recalled items and selects the top items with Q-values to form a list. Typically, three alternative models can be used for recommendation systems: the Actor–Critic model and two deep Q network models. The input of the first DQN model [21] contains only states, and the output is the Q-value corresponding to all actions. The input of the second deep Q-network model [22] contains both states and actions, and the output is the corresponding Q-value. Obviously, the first DQN model finds it difficult to handle scenarios with large action space or dynamics, while the second DQN model needs to calculate the Q-values of all actions and therefore has higher time complexity. Therefore, in this paper, we use the DDPG algorithm, which is a variant of the Actor–Critic model, and compared with the traditional Actor–Critic model [23], the DDPG algorithm has more advantages in dealing with the continuous action space problem. Sun et al. [24] proposed a sequential recommendation model called BERT4Rec, which uses deep bidirectional self-attention to model a sequence of user behaviors. The bidirectional representation model learns to make recommendations by having each item in the user's historical behavior incorporate information from both the left and right sides. Chen et al. [25] proposed a novel model-based reinforcement learning framework for recommender systems and developed a novel Cascading DQN algorithm to obtain a combined recommendation strategy that can efficiently handle a large number of candidate items and can explain user behavior better than other models.

3. Preliminaries

This section gives the basic description of the problem in detail, introducing the basic definitions as well as the Markov decision process.

3.1. Heterogeneous Information Network

As shown in Figure 1, a heterogeneous information network is defined, which contains a user–item bipartite graph and a trust network. In heterogeneous information networks [26], nodes can represent different types of entities, such as people, items, places, etc., while edges represent relationships between different types of entities, such as users purchasing goods, personnel relationships, etc. Heterogeneous networks can naturally fuse not only different types of objects and their interactions but also information from heterogeneous data sources. In the era of big data where data from different sources capture only partial or even biased features, heterogeneous graph networks can synthesize these data. Thus heterogeneous network modeling becomes not only a powerful tool to address the diversity of big data but also a major approach to width learning. Making full use of the data in the heterogeneous information network and capturing the potential relationships in it can improve the recommendation performance.

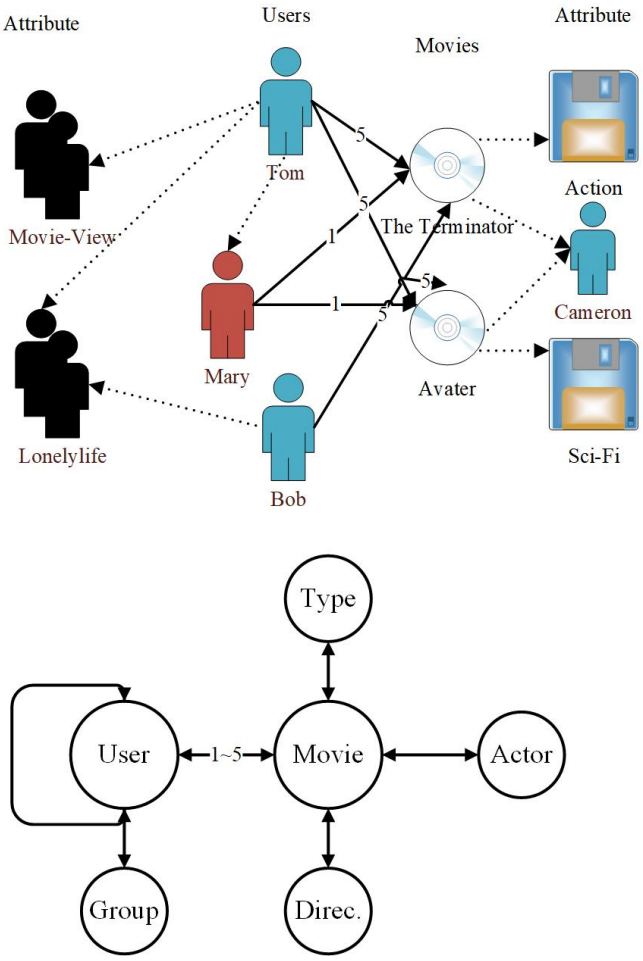

Figure 1. Heterogeneous information network structure.

Define the heterogeneous information network $\mathcal{K} = (U, I, R, trust)$, where $\mathcal{U} = \{u_1, u_2, \ldots, u_k\}$ denotes the set of users and $\mathcal{I} = \{i_1, i_2, \ldots, i_k\}$ denotes the set of items. R denotes the set of relationships. If user u_c clicks or buys item i_t, then the two are connected by directed edges $\langle u_c, i_t \rangle$ and $\langle i_t, u_c \rangle$. For the trust network, a trust matrix is defined as shown in Figure 1. If Tom trusts Bob, there exists a bidirectional edge connecting the two, i.e., $\langle Tom, Bob \rangle = 1$; otherwise, $\langle Tom, Bob \rangle = 0$.

3.2. Problem Description

In this section, we consider analyzing the recommendation process from a reinforcement learning perspective. In a recommendation scenario, the interaction between the agent and the environment is the interaction between the recommendation algorithm and the user.

At each time step t, given the state s_t of the environment (user), the recommendation system intelligence selects an action (item) a_t according to the policy π, which is a mapping from the state to the action probability distribution. As a result of this action, after this time step, the agent receives an immediate reward r_{t+1} and the state is updated to s_{t+1}. The goal of the agent is to learn the optimal policy $\pi : S \rightarrow \mathrm{A}$, thus maximizing the cumulative reward throughout the interaction. This recommendation process satisfies the sequential decision process, which can be modeled as Markov decision processes (MDPs), and usually, MDPs can be represented by the quintuple $(\mathcal{S}, \mathcal{A}, \mathrm{R}, \delta, \gamma)$. Figure 2 illustrates a general scenario of agent–user interaction in MDP.

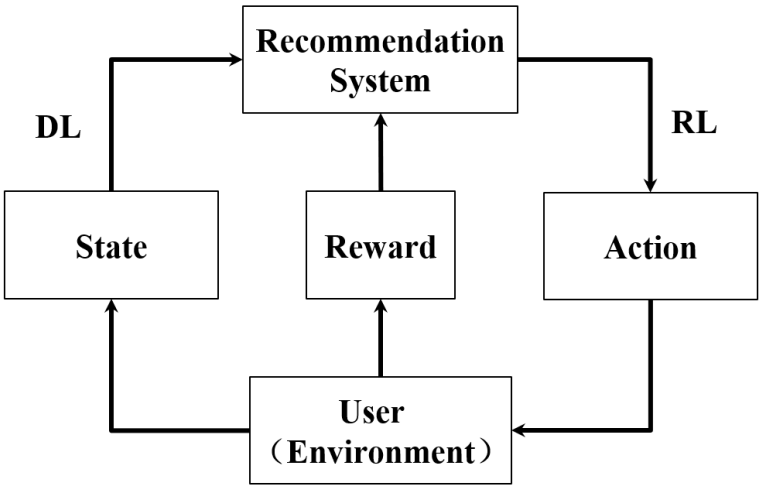

Figure 2. DRL schematic diagram.

3.3. Markov Decision Process

As mentioned earlier, the explainable recommendation tasks based on KG can be transformed into a Markov decision process and solved using deep reinforcement learning methods, where the MDP can be formulated by the $(\mathcal{S}, \mathcal{A}, \mathrm{R}, \delta, \gamma)$ quintuplet.

- $\mathcal{S}$. $\mathcal{S}$ is defined as the state space. The state $s_t \in \mathcal{S} = \{i_1, i_2, \ldots, i_k\}$ is defined as the user's browsing history, i.e., the user has browsed n items before time t. The items in s_t are arranged in chronological order.
- $\mathcal{A}$. $\mathcal{A}$ denotes the set of actions; action $a \in \mathcal{A}$ is a continuous vector and has the same dimension as item i. The ranking score of item i can be obtained by the inner product of action a and item i.
- R. R is defined as the reward function $\mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, where $R(s_t, a_t)$ denotes the immediate reward obtained by the agent for performing action a_t at state s_t.
- δ. δ denotes the state transfer probability function $\mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, where the next time step state $\delta(s_{t+1} \mid s_t, a_t)$ denotes the probability that the agent performs action a_t to reach the next time step state s_{t+1} at state s_t.
- γ. $\gamma \in [0, 1]$ denotes the discount factor, weighing the importance of current and future rewards.

4. GNLR Model

This section describes the details of GNLR and the training process of GNLR.

4.1. GNLR Algorithm Design

As mentioned earlier, some of the previous recommendation systems ignored the importance of feature engineering and focused too much on how to process the models efficiently. In addition, a non-end-to-end approach is taken, which makes model training complex and leads to difficulty reducing recommendation accuracy. With the continuous development and deeper research in the field of recommendation systems, people gradually realize that feature engineering is not only simply transforming raw data but also plays a key role in building efficient and accurate recommendation models. To address these drawbacks, we propose a deep reinforcement learning model GNLR based on the DDPG algorithm. As shown in Figure 3, an Actor network and a Critic network are carefully designed. In addition, because both Actor and Critic networks have to take the user state as input, it is very important for the design of feature representation, and a variant of Graph Neural Network is adopted in this paper to structure the representation of users and items.

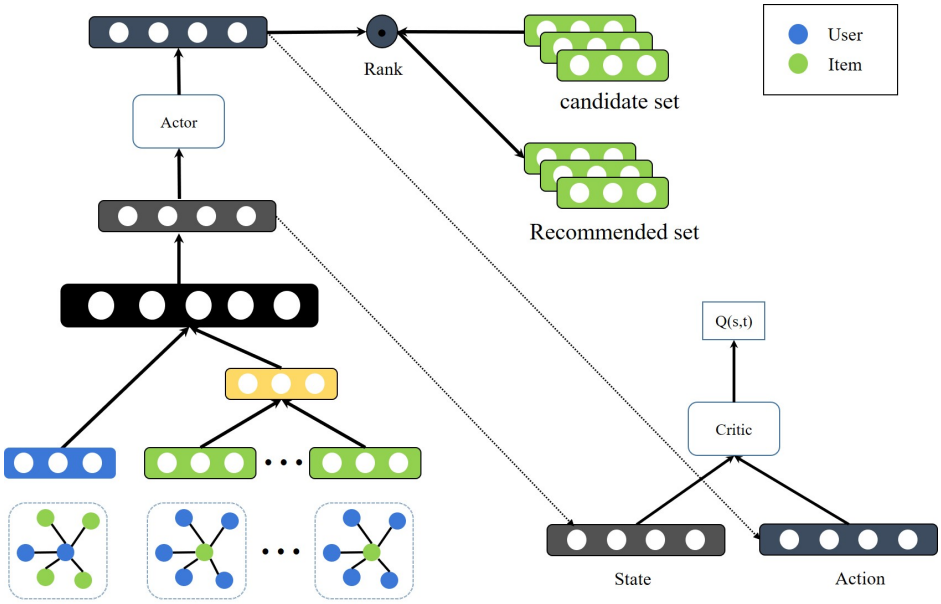

Figure 3. GNLR Algorithm Framework.

According to the MDP definition, the goal of GNLR is to train the recommendation intelligence to obtain a policy π such that the maximum expected discount cumulative reward can be obtained by recommending items to user u at any time step t.

$$H_t \triangleq \max_{\pi_\theta} \mathbb{E}_{\pi_\theta} \sum_{j=0}^{T} \gamma R\left(s_{t+j}, a_{t+j}\right) \tag{1}$$

As shown in Figure 3, GNLR carefully designed an Actor network and a Critic network based on the DDPG algorithm. In addition, a state representation of the user project and a target network are constructed.

4.1.1. Actor Network

Actor networks are called policy networks and can generate actions based on the current state. A good recommendation system should recommend the items that are of most interest to the user. Positive items represent key information about the user's preferences, i.e., which items the user likes. We first combine the current state $S_t = \{S_t^1, \ldots, S_t^N\}$ mapping to a list of weight vectors

$$f_\eta : s_t \rightarrow \Omega_t \tag{2}$$

where f_η is a function parameterized by η that maps from the state space to the weight representation space, and we use a deep neural network as the generating function, assuming that the weight vector is linearly related to the embedding e_i from the i-th item in the item space, which does not affect our generalization that

$$\text{score}_{\ i} = \Omega_t^1 e_i^T \tag{3}$$

After calculating the scores of all items, the recommendation system intelligence selects an item with the highest score as a sub-action a_t^k of action a_t. Finally, the top K ranked items are recommended to the user.

4.1.2. Critic Network

Critic network $Q(s_t, a_t)$ is used as a state–action value function to map to a real value using the state s_t and action a_t at the current time step t as input. The Actor then updates its parameters in the direction of improving performance to generate the appropriate action

at the next time step. The optimal action value function $Q^*(s_t, a_t)$ is utilized in many applications of reinforcement learning, following the Bellman equation:

$$Q^*(s_t, a_t) = \mathbb{E}_{s_{t+1}}[r_t + \gamma \max Q^*(s_{t+1}, a_{t+1}) \mid s_t, a_t] \tag{4}$$

In real situations, the state and action space is huge, so it is not feasible to estimate the Q value of the action value function for each state–action pair. In addition, some state–action pairs may not appear in the real trajectory. Therefore, a neural network in deep learning is used as an approximation function to estimate the action value function. In this paper, a series of loss functions $L(\theta^{\zeta})$ are minimized by using neural networks with parameters θ^{ζ} as function approximators, such as:

$$L\left(\theta^{\xi}\right) = \mathbb{E}_{s_t, a_t, r_t, s_{t+1}}\left[\left(y_t - Q\left(s_t, a_t; \theta^{\xi}\right)\right)^2\right] \tag{5}$$

where $y_t = \mathbb{E}_{s_{t+1}}\left[r_t + \gamma Q'\left(s_{t+1}, a_{t+1}; \theta^{\xi'}\right) \mid s_t, a_t\right]$ is the target of the current iteration, $\theta^{\xi'}$ is the trainable parameter, and Q' denotes the target network value. The loss function is optimized by the stochastic gradient descent method.

4.1.3. State Representation

In deep reinforcement learning, in order to train Actor and Critic networks, the user's state needs to be used as input. Therefore, it is crucial to design an effective state representation model.

As shown in Figure 3, GNLR uses the graph attention approach of GNN to design a state representation model to structurally represent users and items. This approach can tap into the potential relationships between users and items, users and users, and allow the intelligences to pay selective attention based on the different characteristics of the target users and items (Algorithm 1). In addition, the model is able to model sequential information in states to improve the accuracy of recommendations. In using the graph attention approach, users and items are first mapped into a low-dimensional space vector in which $e^u \in \mathbb{R}^d$ and $e^i \in \mathbb{R}^d$ represent the original node features of user u and item i in the heterogeneous information network, respectively. The graph attention representation is applied to a subgraph with the user as the central node or project as the central node to generate a graph-aware a^u and a^i . The method can aggregate neighborhood information to generate its own feature vector.

In order to reduce the influence of noisy nodes that trust their neighbors but are not similar, an attention mechanism is used in the graph attention representation module to assess the trustworthiness of neighbor nodes based on the feature vectors of the user and the user's neighbor nodes, and weights are assigned based on this. Ideally, irrelevant neighbor nodes will be assigned smaller weights, while important neighbor nodes will be assigned larger weights. Users and items are represented structurally using subgraphs rather than constructed using the whole information, thus avoiding repetitive training without nodes and information transfer, reducing training complexity and improving recommendation accuracy.

Taking the generation of user features a^u as an example, all nodes of the subgraph centered on user u are first assigned the corresponding weights by the attention mechanism, i.e., the following equation.

$$\alpha_{u_c\beta} = \frac{\exp\left(\sigma\left(W^T\left[e_c^u \oplus e_\beta^*\right] + b\right)\right)}{\sum_{v \in N} \exp(\sigma(W^T[e_c^u \oplus e_v^*] + b))} \tag{6}$$

where $W \in \mathbb{R}$ and $b \in \mathbb{R}$ are trainable weights and biases, respectively, $\bigoplus$ denotes stitching the vectors together, σ is the activation function, and N denotes all nodes within the subgraph centered on user u including itself as well as its neighbors. Since the nodes

within the subgraph may be users or items, $*$ denotes both cases. The appropriate weights are assigned to the corresponding nodes by computing α. Finally, the final structured representation is obtained by aggregation.

$$a^u = \sigma\left(\sum_{v \in N} \alpha_v e_v^*\right) \tag{7}$$

Algorithm 1: GNLR framework training

Require: Actor learning rate λ_a, Critic learning rate λ_c, discount factor γ, batch size $\boldsymbol{N}$, state window size n, reward function $\boldsymbol{R}$

Initialize the target networks π' and Q' using the weight parameters $\theta' \leftarrow \theta$ and $\omega' \leftarrow \omega$
Initialize the experience replay pool $\boldsymbol{P}$
for *session* $= 1$ **to** M **do**
  Initialize state s_0 based on historical information
  for $t = 1$ **to** T **do**
    The current state $s_t = f(B_t)$, where $B_t = i_1, i_2, \ldots, i_n$
    Find $a_t = \pi_\theta(s_t)$ according to the current strategy using the ϵ-greedy strategy
    Recommended items i_t according to action a_t by Equation (3)
    Calculate the reward $r_t = R(s_t, a_t)$ based on the user's feedback
    The new state $s_{t+1} = f(B_{t+1})$, where if r_t is positive $B_{t+1} = i_2, \ldots, i_n, i_1$, otherwise $B_{t+1} = B_t$
    Store (s_t, a_t, r_t, s_{t+1}) in $\boldsymbol{P}$
    Sampling of (s_t, a_t, r_t, s_{t+1}) using empirical playback techniques for $\boldsymbol{N}$
    Set $y_i = r_i + \gamma Q_{\omega'}(s_{i+1}, \pi_{\theta'}(s_{i+1}))$
    Update the Critic network by minimizing the loss function
    Updating Actor Networks with Sampling Policy Gradients
    Update the target network: $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$ and $\omega' \leftarrow \tau\omega + (1-\tau)\omega'$
  end for
end for
return θ and ω

Heterogeneous information networks play an important role. A heterogeneous information network is a complex network containing different types of nodes and connections, such as users, projects, etc. In this section, heterogeneous information networks are used to construct user-centric or project-centric subgraphs denoted as $\mathcal{G}(\mathbf{u_c})$ and $\mathcal{G}(\mathbf{i_j})$, respectively. These subgraphs are extracted from the entire network where users are the central nodes or items are the central nodes. In these subgraphs, the graph attention representation module applies the graph attention method of Graph Neural Networks. This approach allows the model to focus on the neighboring nodes of a specific node in order to quickly generate the feature vector of the node.

Through this process, users and items can be given structured representations with graph-awareness capabilities, such that the representations capture potential relationships between users and items as well as user-to-user relationships. In constructing these structured representations of subgraphs, subgraph-level information is used while avoiding construction over the entire heterogeneous information network. This helps to minimize the interference of unarticulated points in training while reducing the complexity of information transfer. In addition, in the graph attention representation module, an attention mechanism is used to assess the trustworthiness of neighboring nodes and assign weights based on this. This mechanism helps to reduce the influence of noisy nodes and ensures that important neighbor nodes receive higher weights while irrelevant neighbor nodes receive lower weights.

4.1.4. Reward Function Setting

The reward $R(s_t, a_t)$ is an evaluation of the quality of the recommended items. Specifically, we define the reward function as

$$R(s, a) = R_0(s, a) + \Lambda r(s, a) \tag{8}$$

where $R_0(s, a)$ denotes the initial reward. We use the supervised learning model Probabilistic Matrix Factorization (PMF) to predict user feedback (ratings) for recommended items that have not been previously rated and normalize all initial rewards in the range $[-1, 1]$. $r(s, a)$ denotes the potential reward function. We use the Normalized Discounted Cumulative Gain (NDCG) method based on ranking; i.e., we use the change in the NDCG of the recommendation list as the $r(s, a)$ when we add the recommended items to the recommendation list at each time step. Since the training algorithm learns under the supervision of the reward values of the model parameters, so defining the reward value based on the evaluation metric of ranking can know that the training process achieves higher ranking performance. Λ denotes the hyperparameter, which is used to balance the importance of both.

5. Experiment

This section provides a comprehensive evaluation of the GNLR method using real-world datasets. We first introduce the datasets and evaluation metrics used in the experiments. Next, the baseline algorithms for comparison are introduced, and a quantitative comparison between GNLR and baseline methods is made to demonstrate the effectiveness of GNLR. The corresponding parameter settings in the experiments are then described.

5.1. Experimental Setup

In this paper, experiments are conducted using two publicly available datasets from the real world, Ciao and LastFM, and the statistical information of each dataset is shown in Table 1. The experimental environment used is Python 3.7, and the GPU is NVIDIA 3050ti Laptop with CUDA version 12.0. Online evaluation is performed using these two datasets. A heterogeneous information network is generated from the data in the datasets, and a user-project interaction matrix is generated from the heterogeneous information network.

Table 1. Dataset statistics.

Dataset	Ciao	LastFM
#User	1874	7260
#Item	2828	11,166
#Relation	71,411	72,665
#Feedback	25,174	40,133

In this paper, we use HitRatio (HR) and NDCG to evaluate the performance of the recommendation system. The higher the NDCG value, the higher the accuracy of the recommendation. The size of the recommendation list is K, the i-th recommendation item is r_i. If it is clicked by the user, then $r_i = 1$; otherwise, $r_i = 0$. In the actual test, we consider the recommendation items in the test set as the clicked items and the recommendation items not in the test set as the unclicked items.

$$NDCG@K = \frac{DCG@K}{IDCG} \tag{9}$$

In Equation (9), the higher the positive feedback item is ranked, the greater the DCG@K. IDCG is the ideal value of DCG, which is the optimal value for the positive feedback item to be ranked in the first place. NDCG@k takes into account the sorting order, and the maximum NDCG@k has the value 1.

In Equation (10) $F(j)$ denotes the good ranked as j, c denotes the recommended user number, only focusing on the top K items, and HR denotes whether the positive feedback items of the tester appear in the list of the top K: if it appears, $HR@K = 1$; conversely, $HR@K = 0$.

$$DCG@K = \sum_{j=1}^{K} \frac{2^{y_{cF(j)}} - 1}{\log_2(1+j)} \tag{10}$$

5.2. Baselines and Parameter Settings

The proposed GNLR algorithm is compared with some representative baseline methods. Random, BPR, NeuMF, DEERS and DRR are used, respectively. The Random method randomly selects K items from the candidate set as the top-ranked recommendation items. BPR is a Bayesian personalized ranking algorithm that learns vector representations of users and items. NeuMF is a deep learning-based recommendation algorithm for handling implicit feedback by combining neural networks and traditional matrix decomposition. DEERS represents the state with both the positive and negative feedbacks by a recurrent neural network under the DQN framework. DRR is an interactive recommendation algorithm based on deep reinforcement learning, which carefully learns the user state to learn the recommendation strategy.

In this section, GNLR is compared with the baseline algorithm and the results are shown in Table 2. According to the given data table, it is found that the GNLR model outperforms the other models on all datasets in terms of HR@10 and NDCG@10 metrics. The results show that the overall performance of GNLR is due to other baseline algorithms, which proves the effectiveness of our proposed method. NeuMF uses feedforward neural networks for user and item embedding, and for sparse data, neural networks are prone to overfitting problems, and the computational complexity of using neural networks for prediction is high, so the training time can be long for large datasets. DRR is based on user and item embedding for prediction; for long-tail items, a lack of interaction data DRR prevents accurate predictions, and there is a need to calculate the embedding vector, which will lead to high computational complexity, making it not suitable for real-time recommendation systems. And these baseline methods simply connect item embeddings to represent state s, which leads to information loss resulting in reduced recommendation performance. The GNLR algorithm mitigates the effect of data sparsity by aggregating neighborhood information using a heterogeneous information network of GNN and shows optimal performance.

Table 2. Performance comparison of all methods on dataset.

Model	Ciao		LastFM	
	HR@10	**NDCG@10**	**HR@10**	**NDCG@10**
Random	0.100	0.045	0.100	0.045
BPR	0.445	0.255	0.679	0.469
NeuMF	0.459	0.267	0.690	0.477
DEERS	0.329	0.221	0.678	0.445
DRR	0.360	0.222	0.682	0.471
GNLR	**0.736**	**0.541**	**0.729**	**0.545**

5.3. Embed Size d Analysis

This section analyzes the performance impact by the size d of the user and item embedding vectors. The GNLR performance trends are examined by comparing when d is 8, 16, 32, 64, 128, and 256, respectively. The results are shown in Figure 4, where the recommended performance trend is first increasing and then decreasing. The recommended performance reaches the maximum when d = 32 and GNLR has the best performance. When the embedding size d is less than 32, the difference in structured representation is too large to completely describe the features because the dimension is too small; when

the size of d is too large, the dimension is too large, making the user and item features have smoothing problems and unable to be adequately trained, leading to a decrease in recommendation accuracy.

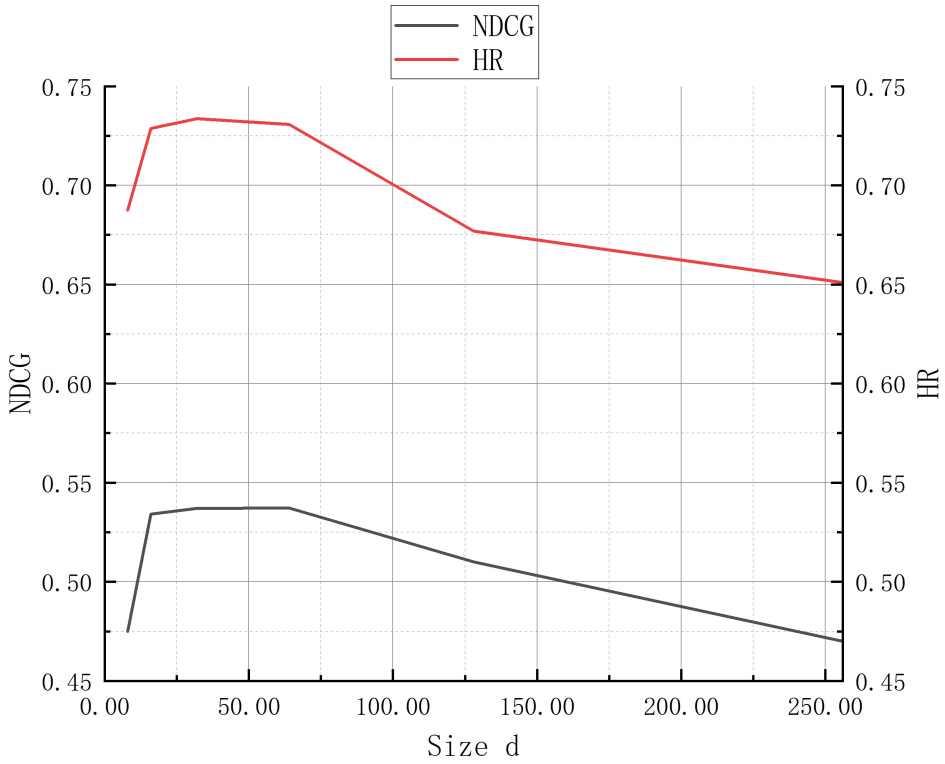

Figure 4. Embedding size d analysis.

5.4. Subgraphs and Sampling Analysis in Heterogeneous Information Networks

Only then GNLR used a subgraph approach to generate a structured representation of users and items and took a random sampling order to draw P subgraph nodes to update. This section investigates the effect of the sampling number P on the recommendation performance. As shown in Table 3, the recommendation performance when P = 2, 5, 8, 10, 15 is investigated by fixing the subgraph size K = 1, respectively. The results show that the best performance is achieved when P = 8. When P is too large, the attention mechanism also cannot eliminate the noisy information brought by the nodes; when P is too small, it cannot capture enough neighborhood information in the subgraph.

Table 3. Subgraph sampling analysis of heterogeneous information network.

P	Ciao	LastFM
	K = 1	**K = 1**
2	0.7415	0.7422
5	0.7491	0.7468
8	**0.7606**	**0.7602**
10	0.7396	0.7356
15	0.7295	0.7348

5.5. Ablation Experiment

In GNLR, a variant of the graph attention mechanism is used to pay selective attention to neighborhood information and to learn directly from the data in an end-to-end manner. In order to investigate the effect of these two points on GNLR, two variants of GNLR are designed: the GNLR-A and GNLR-B. GNLR-A removes the graph attention mechanism

and employs an averaging approach by assigning the same weight to each neighborhood information. The GNLR-B employs a non-end-to-end model training approach.

The experimental results for GNLR as well as the two variants are shown in Figure 5, where it can be observed that GNLR achieves optimal performance in the Last-FM and Ciao datasets compared to the other two variants. Since GNLR-A removes the attention mechanism, which makes it impossible to give more weight to neighborhood information that is beneficial for decision making, subgraphs with more noisy nodes affect the graph-structured representation of users and items. In addition, the non-end-to-end model training approach used by GNLR-B leads to a decrease in model accuracy due to error accumulation and other effects on the next computation.

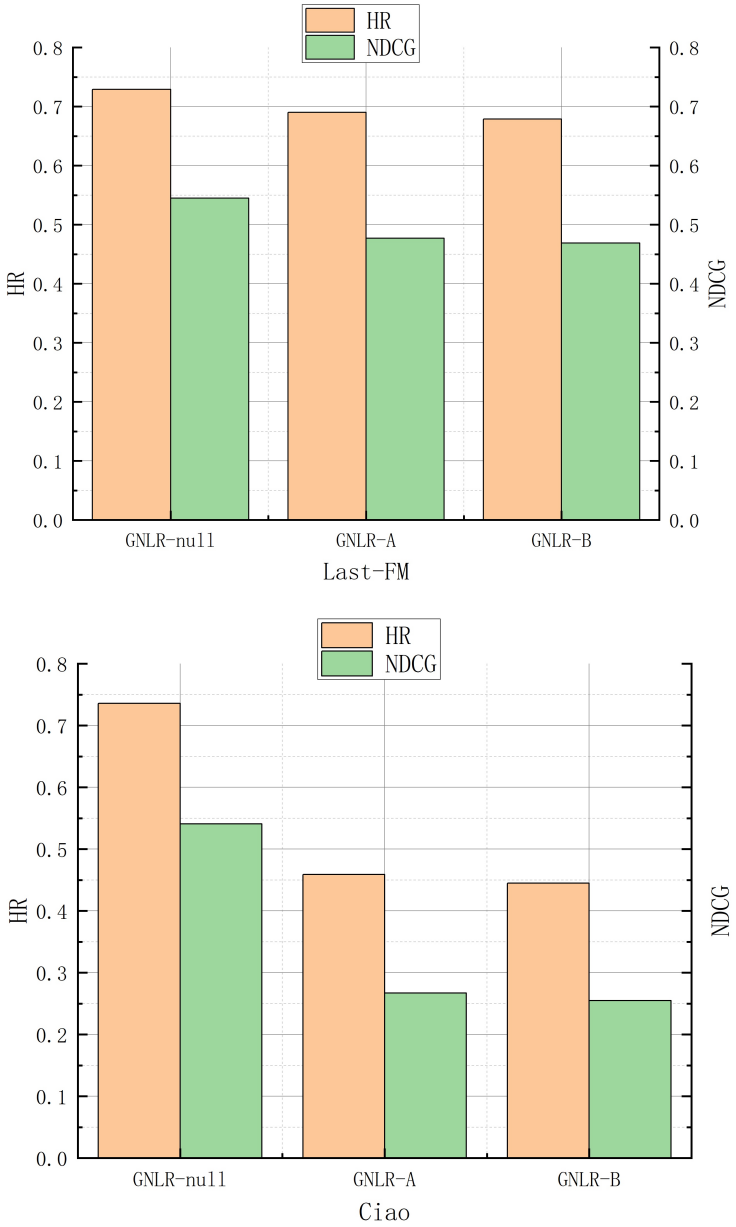

Figure 5. Ablation experiment.

6. Conclusions

In this paper, an end-to-end deep reinforcement learning-based list recommendation model (GNLR) is proposed. The approach models the recommendation process as a Markov decision process and uses deep reinforcement learning to obtain the optimal

policy. The user and item structured representations are trained in an end-to-end manner and constructed using a heterogeneous information network with user–item bipartite graph fusion, and the weights of the neighborhood information are adjusted according to the attention mechanism. The results show that GNLR can effectively build user state information and improve the training efficiency and recommendation accuracy of the model using an end-to-end approach, and it can also reduce the training complexity. In future work, we can consider improving the interpretability of the framework. In this paper, there is a lack of solutions for model interpretability in our work. Improving interpretability can help improve the trust of recommendations and meet the personalized needs of users. In addition, considering the relationship between nodes enables users and items to obtain a more accurate structured representation, which can be assisted by introducing a knowledge graph with rich information, thereby improving the accuracy of recommendation.

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

Funding: This research was funded by Yantai University: National Natural Science Foundation of China project grant number (62072392, 61972360).

Data Availability Statement: The data can be shared upon request.

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

References

1. Chiu, M.C.; Huang, J.H.; Gupta, S.; Akman, G. Developing a personalized recommendation system in a smart product service system based on unsupervised learning model. *Comput. Ind.* **2021**, *128*, 103421. [CrossRef]
2. Zheng, G.; Zhang, F.; Zheng, Z.; Xiang, Y.; Yuan, N.J.; Xie, X.; Li, Z. DRN: A deep reinforcement learning framework for news recommendation. In Proceedings of the 2018 World Wide Web Conference, Lyon, France, 23–27 April 2018; pp. 167–176.
3. Hamilton, W.; Ying, Z.; Leskovec, J. Inductive representation learning on large graphs. *Adv. Neural Inf. Process. Syst.* **2017**, *30*.
4. Wu, L.; He, X.; Wang, X.; Zhang, K.; Wang, M. A survey on accuracy-oriented neural recommendation: From collaborative filtering to information-rich recommendation. *IEEE Trans. Knowl. Data Eng.* **2022**, *35*, 4425–4445. [CrossRef]
5. Shipe, M.E.; Deppen, S.A.; Farjah, F.; Grogan, E.L. Developing prediction models for clinical use using logistic regression: An overview. *J. Thorac. Dis.* **2019**, *11* (Suppl. S4), S574. [CrossRef] [PubMed]
6. Rao, H.; Shi, X.; Rodrigue, A.K.; Feng, J.; Xia, Y.; Elhoseny, M.; Yuan, X.; Gu, L. Feature selection based on artificial bee colony and gradient boosting decision tree. *Appl. Soft Comput.* **2019**, *74*, 634–642. [CrossRef]
7. Chapelle, O.; Li, L. An empirical evaluation of thompson sampling. *Adv. Neural Inf. Process. Syst.* **2011**, *24*, 12–13.
8. Liu, F.; Tang, R.; Li, X.; Zhang, W.; Ye, Y.; Chen, H.; Guo, H.; Zhang, Y. Deep reinforcement learning based recommendation with explicit user-item interactions modeling. *arXiv* **2018**, arXiv:1810.12027.
9. Wang, H.; Wu, Q.; Wang, H. Factorization bandits for interactive recommendation. In Proceedings of the AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017; Volume 31.
10. Zhao, X.; Zhang, W.; Wang, J. Interactive collaborative filtering. In Proceedings of the 22nd ACM International Conference on Information & Knowledge Management, San Francisco, CA, USA, 27 October–1 November 2013; pp. 1411–1420.
11. Riedmiller, M. Neural fitted Q iteration–first experiences with a data efficient neural reinforcement learning method. In *Machine Learning: ECML 2005: 16th European Conference on Machine Learning, Porto, Portugal, 3–7 October 2005*; Proceedings 16; Springer: Berlin/Heidelberg, Germany, 2005; pp. 317–328.
12. Lange, S.; Riedmiller, M. Deep auto-encoder neural networks in reinforcement learning. In Proceedings of the 2010 International Joint Conference on Neural Networks (IJCNN), Barcelona, Spain, 18–23 July 2010; pp. 1–8.
13. Abtahi, F.; Fasel, I. Deep belief nets as function approximators for reinforcement learning. *RBM* **2011**, *2*, H3.
14. Lange, S.; Riedmiller, M.; Voigtländer, A. Autonomous reinforcement learning on raw visual input data in a real world application. In Proceedings of the 2012 International Joint Conference on Neural Networks (IJCNN), Brisbane, QLD, Australia, 10–15 June 2012; pp. 1–8.
15. Koutnik, J.; Schmidhuber, J.; Gomez, F. Online evolution of deep convolutional network for vision-based reinforcement learning. In *From Animals to Animats 13: 13th International Conference on Simulation of Adaptive Behavior, SAB 2014, Castellón, Spain, 22–25 July 2014*; Proceedings 13; Springer International Publishing: Cham, Switzerland, 2014; pp. 260–269.

16. Wang, X.; Song, J.; Qi, P.; Peng, P.; Tang, Z.; Zhang, W.; Li, W.; Pi, X.; He, J.; Gao, C.; et al. SCC: An efficient deep reinforcement learning agent mastering the game of StarCraft II. In Proceedings of the International Conference on Machine Learning, PMLR, Virtual Event, 18–24 July 2021; pp. 10905–10915.
17. Zhang, Y.; Sun, P.; Yin, Y.; Lin, L.; Wang, X. Human-like autonomous vehicle speed control by deep reinforcement learning with double Q-learning. In Proceedings of the 2018 IEEE Intelligent Vehicles Symposium (IV), Changshu, China, 26–30 June 2018; pp. 1251–1256.
18. Wang, Z.; Schaul, T.; Hessel, M.; Hasselt, H.; Lanctot, M.; Freitas, N. Dueling network architectures for deep reinforcement learning. In Proceedings of the International Conference on Machine Learning, PMLR, New York, NY, USA, 20–22 June 2016; pp. 1995–2003.
19. Chen, H. A DQN-based recommender system for item-list recommendation. In Proceedings of the 2021 IEEE International Conference on Big Data (Big Data), Orlando, FL, USA, 15–18 December 2021; pp. 5699–5702.
20. Xie, R.; Zhang, S.; Wang, R.; Xia, F.; Lin, L. Hierarchical reinforcement learning for integrated recommendation. *Proc. AAAI Conf. Artif. Intell.* **2021**, *35*, 4521–4528. [CrossRef]
21. Hafiz, A.M.; Hassaballah, M.; Alqahtani, A.; Alsubai, S.; Hameed, M.A. Reinforcement Learning with an Ensemble of Binary Action Deep Q-Networks. *Comput. Syst. Sci. Eng.* **2023**, *46*, 2651–2666. [CrossRef]
22. Huang, L.; Ye, M.; Xue, X.; Wang, Y.; Qiu, H.; Deng, X. Intelligent routing method based on Dueling DQN reinforcement learning and network traffic state prediction in SDN. *Wirel. Netw.* **2022**, 1–19. [CrossRef]
23. Dong, Y.; Zou, X. Mobile robot path planning based on improved ddpg reinforcement learning algorithm. In Proceedings of the 2020 IEEE 11th International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 16–18 October 2020; pp. 52–56.
24. Sun, F.; Liu, J.; Wu, J.; Pei, C.; Lin, X.; Ou, W.; Jiang, P. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, Beijing, China, 3–7 November 2019; pp. 1441–1450.
25. Chen, X.; Li, S.; Li, H.; Jiang, S.; Qi, Y.; Song, L. Generative adversarial user model for reinforcement learning based recommendation system. In Proceedings of the International Conference on Machine Learning, PMLR, Long Beach, CA, USA, 9–15 June 2019; pp. 1052–1061.
26. He, Y.; Song, Y.; Li, J.; Ji, C.; Peng, J.; Peng, H. Hetespaceywalk: A heterogeneous spacey random walk for heterogeneous information network embedding. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, Beijing, China, 3–7 November 2019; pp. 639–648.