entropy

MDPI

*Article*

# Personal Interest Attention Graph Neural Networks for Session-Based Recommendation

**Xiangde Zhang, Yuan Zhou, Jianping Wang and Xiaojun Lu ***

College of Sciences, Northeastern University, Shenyang 110819, China; zhangxiangde@mail.neu.edu.cn (X.Z.); 1970039@stu.neu.edu.cn (Y.Z.); wangjianping@mail.neu.edu.cn (J.W.)
* Correspondence: luxiaojun@mail.neu.edu.cn; Tel.: +86-024-8367-8650

**Abstract:** Session-based recommendations aim to predict a user's next click based on the user's current and historical sessions, which can be applied to shopping websites and APPs. Existing session-based recommendation methods cannot accurately capture the complex transitions between items. In addition, some approaches compress sessions into a fixed representation vector without taking into account the user's interest preferences at the current moment, thus limiting the accuracy of recommendations. Considering the diversity of items and users' interests, a personalized interest attention graph neural network (PIA-GNN) is proposed for session-based recommendation. This approach utilizes personalized graph convolutional networks (PGNN) to capture complex transitions between items, invoking an interest-aware mechanism to activate users' interest in different items adaptively. In addition, a self-attention layer is used to capture long-term dependencies between items when capturing users' long-term preferences. In this paper, the cross-entropy loss is used as the objective function to train our model. We conduct rich experiments on two real datasets, and the results show that PIA-GNN outperforms existing personalized session-aware recommendation methods.

**Keywords:** session-based recommendation; graph neural networks; attention; recommendation system



**Citation:** Zhang, X.; Zhou, Y.; Wang, J.; Lu, X. Personal Interest Attention Graph Neural Networks for Session-Based Recommendation. *Entropy* **2021**, *23*, 1500. https://doi.org/10.3390/e23111500

Academic Editor: Jaesung Lee

Received: 9 October 2021
Accepted: 9 November 2021
Published: 12 November 2021



## 1. Introduction

In recent years, the rapid development of Internet technology in many applications (such as e-commerce, social media, etc.) has caused information overload. Recommendation system play a crucial role in helping users alleviate information overload and select interested information. Among them, content-based RS and CF-based RS are the two most widely used methods which make simple and efficient recommendations by calculating the similarity between items. These methods assume that all historical interactions of users are equally important to their current preferences, but this is impossible in reality. The user's click depends not only on their long-term preference, but also on their short-term preference and time-sensitive context. In many cases, the items currently interacting with the user can better reflect their recent preferences. For example, in e-commerce, the user is more likely to purchase recently viewed products because these products are more representative of the user's recent needs. In this case, content-based RS and collaborative filtering-based RS fail to capture the change of user's preferences and recommend inaccurately. In addition, most existing recommendation systems use users' information and historical behavior records for personalized recommendation. However, in actual applications, user information is unknown, and user history behaviors are not available besides the current session, so it is difficult to produce accurate results. To solve these problems, a session-based recommendation system extracts interactive information to represent the user's preference transfer and use the limited historical behavior to predict the user's next action (such as which item to click).

The internet contains a large amount of information, and users' interests are constantly changing. A session-based recommendation system can capture a user's short-term preferences from a user's recently generated session and get the changes of users' preferences

between different sessions. Based on the above two points, session-based recommendation system is widely used in shopping and short video APPs.

Since the session sequence is divided by time, it can be represented as time series, so Markov chains can be used. Markov assume that the user's next action is only influenced by the current action, regardless of other factors. This strong independence assumption is vulnerable to noisy data and may limit the accuracy of recommendations. The research of [1] showed that user preferences are not entirely dependent on the time-order of sequences and emphasized the importance of recurrent neural networks (RNNs) in a session-based recommendation system. Recent methods divide user preferences into long-term preferences and short-term preferences. Hidasi et al. proposed an RNN-based method GRU4Rec [2] which models user short-term preferences by encoding items. NARM [3] uses two RNN-based subsystems to capture the long-term and short-term preferences of users respectively. This method selects the last item in the session to represent short-term preferences, and others represent long-term preferences. Similar to NARM, STAMP [4] uses a simple MLP model and an attentive net to extract users' potential interests. Although these methods have achieved good results, we believe that these methods are still in their infancy and have certain limitations. Complex user behavior patterns are important for session-based recommendation, yet the above sequence-based approach only modeled sequential transitions between consecutive items, ignoring the relationship between other items. To overcome this limitation, this paper models the items in a session as a session graph, captures the transitions between items by GNN, and generates an embedding vector representation.

In addition, when making recommendations, the candidates are rich and the users' interests are usually diverse. For example, as a boy, Alex is interested in computers, sneakers, and game consoles at the same time. Therefore, over a period of time, he may click on items from all three categories. Many recent models [3–5] represent user interests as a fixed-size embedding vector, which cannot express multiple interests of users, and limit the expressiveness of recommendation models. It is not necessary to embed all user interests into a single vector when making predictions for a specific candidate. For example, suppose Alex has a historical session (computer, sneakers, milk, game console). If we want to recommend gamepads to him, we will focus on his interest in game consoles rather than milk. In other words, we can recommend items based on user behavior for a user's specific interests.

Figure 1 illustrates the workflow of the proposed PIA-GNN method. In this paper, we generate a representation for each item by modeling session sequences as graph structures data that capture the complex transitions between items [6,7]. We use a personalized graph neural network (PGNN) to further strengthen the association between each user and its different sessions by adding users' information when updating the node embedding updates. In addition, in order to analyze the specific interests of the user, we add a new interest attention module, which adaptively activates user interest by considering the historical behavior correlation of a given target item. Then, we use the self-attention network to obtain the global embedding. Finally, we use the user's target, global and local embedding in the session to construct the session embedding, and infer the user's next action based on the item embedding and session embedding.

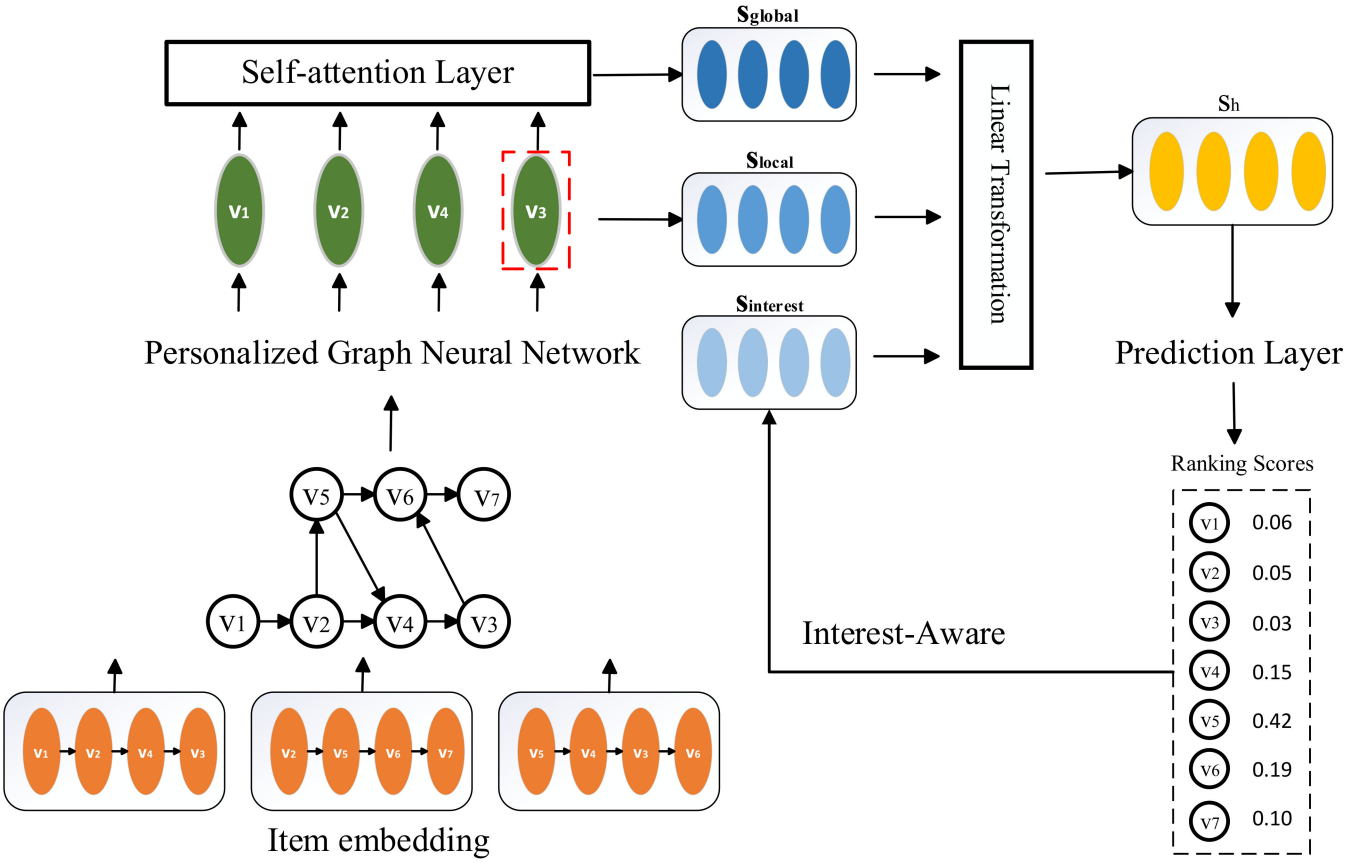



**Figure 1.** Workflow of the PIA-GNN method. We model all sessions as session graphs and process each session graph one by one. Then a node vector representation is obtained using PGNN network. The global embedding $s_{global}$ is obtained using the self-attention mechanism, the last item of the session is used as the local embedding $s_{local}$, and an interest embedding $s_{interest}$ is obtained by interest perception. In the prediction layer, we represent each session as a linear representation of the interest embedding, local embedding and global embedding. Finally, we calculate the recommendation ranking score for each candidate item.

Main contributions:

- For personalized recommendation scenarios, we use a new user-based personalized graph neural network (PGNN), which can capture complex item transformations for different user interests.
- For the different interests of users, we add the interest attention module, which can activate different user interests adaptively for different targets and improve the expressiveness of the model.
- We have studied the model on two real-world datasets. Experiments demonstrate the effectiveness of the proposed model.

## 2. Relate Work

In this section, we review some related work on session-based recommender systems, including traditional approaches, sequential approaches based on Markov chains, and RNN-based approaches. Then, we introduce neural networks on graphs.

### *2.1. Traditional Recommendation Method*

Matrix factorization [8–10] is a frequently method used in recommender systems. MF factorize a user-item evaluation matrix into two low-rank matrices, each of which represents the potential factors of a user or an item. In matrix factorization, user preferences can only be provided by some positive click behaviors and are not applicable to session-based recommendations. In item-based neighborhood methods [11], item similarity can be

calculated by all items in a session. However, these methods have difficulty in considering the order of items and generate predictions based on the last click only. To solve the above problem, Markov chain-based sequential prediction methods are proposed, which predict the user's next behavior based on the last one. Zimdars et al. [12] used the probabilistic decision tree model to encode the conversion mode of goods. Shani et al. [13] regard recommendation generation as a sequential optimization problem, which is solved by Markov decision processes (MDPs). FPMC [14] models the sequence behavior between two clicks by decomposing the user's personalized probability transfer matrix and provide prediction for each sequence more accurately. The main drawback of Markov chain-based models is that they independently combine past components. This independence assumption is too strong and limits the accuracy of the predictions.

### *2.2. Deep-Learning-Based Methods*

In recent years, deep neural networks have been successfully applied to machine translation [13,15,16], conversational machines [17], and other sequential modeling fields and achieved good results. For session-based sequential recommendation, a recurrent neural network approach was proposed in [2]. On this basis, Hidasi et al. [18] proposed a model, GRU4REC, that applies RNN networks to SBR, which uses multilayer gated recurrent units (GRUs) to model item interaction sequences and can model conversations based on the characteristics of clicked operations and clicked items. Tan et al. [9] improved the performance of recurrent model by considering temporal changes in user behavior and appropriate data augmentation techniques. The NARM model [3] integrates the attention mechanism into the GRU encoder and designs a global and local RNN recommender to capture users' sequential behavior and primary purpose. The SHAN model [19] uses a two-layer hierarchical attention network that considers both long-term and short-term preferences. Liu et al. [4] proposed an attention-based short-term memory network (STAMP), using a simple MLP network and an attention network to capture the general and current interests of users. Both NARM and STAMP use an attention mechanism to emphasize the importance of the last click. However, the above session or sequence-based models can only make suggestions using current anonymous sessions or individual sequences.

### *2.3. Graph Neural Networks*

Nowadays, neural networks have been used to generate representations of graph-structured data, such as social networks and knowledge bases. On the one hand, the unsupervised algorithm Deep Walk [20], extending word2vec [21], learns graph node representations by random walk. Following Deep Walk, the unsupervised network embedding algorithms LINE [22] and node2vec [23] are the most representative approaches. On the other hand, the classical neural networks CNN and RNN are also applied on graph structured data. Duvenaud et al. [24] introduced a convolutional neural network that can directly operate on graphs of any sizes and shapes. Thomas N [25] selected the convolution architecture by a localized approximation of spectral graph convolution, which is an effective variant that can directly operate on the graph. However, these methods can only be used for undirected graphs. Previously, graph neural networks (GNNs) [26,27] were proposed in the form of recurrent neural networks to operate on directed graphs. GNNs are good at processing graph-structured data and can capture richer information in sequential data. Gated graph neural networks [28] use gated recurrent units and back propagation in time (BPTT) to compute gradients based on GNNs. Graph attention networks (GAT) [29] uses an attention mechanism to learn the weights of nodes and neighbor nodes. Wu et al. [12] proposed a gated GNN model (called SR-GNN) to learn item embeddings on the session graph, and then integrated each learned item embedding based on the attention value to obtain a representative session embedding, and the attention value is calculated based on the relevance of each item to the last item. SR-GNN [12] is the first model that captures complex item transition relationships using gated graph neural networks in a session-based recommendation scenario, but it ignores the user's role in item transition

relationships and does not exploit historical user session information to improve recommendation performance. In this paper, we propose the PTA-GNN model that can make personalized session recommendations for user-specific interests.

## 3. The Proposed Method

In this section, we introduce the proposed PTA-GNN model, that uses personalized graph neural network, user interest perception embedding and attention mechanism to achieve session-based recommendation. We describe the problem first. Then, the proposed model is presented in this part.

### *3.1. Problem Statement*

Session-based recommendation aims to predict which item the user wants to click next based on anonymous sessions. Here, we give the formulation of the session-based problem as follows. In session-based recommendation, let $\mathcal{V} = \left\{v_1, v_2, \cdots, v_{|\mathcal{V}|}\right\}$ represent the set of all unique items involved in all sessions. For each anonymous session $s$, the sequence of actions clicked by the user is denoted as $s = \{v_{s,1}, v_{s,2}, \cdots, v_{s,n}\}$, where $v_{s,i} \in \mathcal{V}$ is the item clicked by the user within the session and $n$ stands for the total number of sessions for a user $u$. Given the user's action sequence $s = \{v_{s,1}, v_{s,2}, \cdots, v_{s,n}\}$, our model aims to predict the next click $(v_{s,n+1})$. To be precise, our model generates a ranking list of all the candidates that may appear in the session as $\hat{y} = \left\{\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_{|\mathcal{V}|}\right\}$ represents the output probability of all items, where $\hat{y}_i$ represents the recommendation score of the item $v_i$. Since recommenders usually recommend users more than once, we select the $Top - k$ items from $\hat{y}$ for recommendation.

### *3.2. Constructing Session Graphs*

Each session sequence $s$ can be modeled as a directed graph $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s)$, where $\mathcal{V}_s$ is the set of nodes and $\mathcal{E}_s$ is the set of edges, in this session graph $\mathcal{G}_s$, each node represents an item $v_{s,i} \in \mathcal{V}$. Edge $(v_{s,i-1}, v_{s,i}) \in \mathcal{E}_s$ represents that the user successively clicks the item $v_{s,i-1}$ and the item $v_{s,i}$. Therefore, each session sequence can be modeled as a directed graph. Since the items in the sequence may be repeated, we assign a normalized weight to each edge, which is the number of occurrences of the edge divided by the outgoing degree of the node where the edge starts. In the session graph, the edge $v_i \to v_j$ represents that the user first clicked on item $v_i$ and then clicked on item $v_j$ in a particular session. In this case, we assume that on the edge $v_i \to v_j$, the effect of $v_i$ on $v_j$ and the effect of $v_j$ on $v_i$ are different, and it produces two types of edges to represent two different transformation relationships. These two directed edges are called the incoming and outgoing edges, and the weights are $\omega_{ij}^{in}$ and $\omega_{ij}^{out}$, respectively. The weights can be calculated by Equations (1) and (2) as follows:

$$\omega_{ij}^{in} = \frac{Count(v_j, v_i)}{\sum_{v_{k \to v_i}} Count(v_k, v_i)} \tag{1}$$

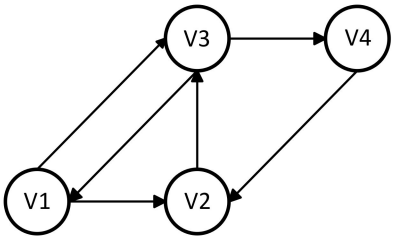
$$\omega_{ij}^{out} = \frac{Count(v_i, v_j)}{\sum_{v_{i \to v_k}} Count(v_i, v_k)} \tag{2}$$

$$A_s^{in}[i, j] = \omega_{ij}^{in} \tag{3}$$

$$A_s^{out}[i, j] = \omega_{ij}^{out} \tag{4}$$

where the function $Count(x, y)$ is used to count the number of times the user interacts with the item $y$ after interacts with the item $x$. $A_s$ is defined as the connection matrix of two adjacency matrices $A_s^{out}$ and $A_s^{in}$, where $A_s^{out}$ and $A_s^{in}$ denote the weighted connections of outgoing and incoming edges in the session graph. As in Figure 2, for a session $s = \{v_1, v_2, v_3, v_1, v_3, v_4, v_2\}$, we can construct a session graph $\mathcal{G}_s$ and a matrix $A_s$. Here, we use the same strategy for constructing session graphs as SR-GNN. Note that for session graphs with different structures, PTA-GNN can support various strategies

for constructing session graphs and generating the corresponding connection matrices to update the graph structure by facilitating the information transfer between different nodes. We embed each item $v \in \mathcal{V}$ into a uniform embedding space, and the node vector $\nu \in \mathbb{R}^d$ denotes the latent vector of item $v$ learned through the graph neural network, where $d$ is the dimension. Based on the node vectors, each session $s$ can be represented by an embedding vector, which consists of all node vectors in the graph.




| | Outcoming edge | | | | Incoming edge | | | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| 1 | 0 | 1/2 | 1/2 | 0 | 0 | 0 | 1 | 0 |
| 2 | 0 | 0 | 1 | 0 | 1/2 | 0 | 0 | 1/2 |
| 3 | 1/2 | 0 | 0 | 1/2 | 1/2 | 1/2 | 0 | 0 |
| 4 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |

**Figure 2.** An example of a session graph and the connection matrix $A_s$.

### *3.3. Personalized Graph Neural Network*

The application of GNN to session-based recommendation was first proposed in the SR-GNN model, which provides a session graph containing rich node connections to GGNN to automatically extract useful features of items. However, SR-GNN does not inject user information into the graph model, so it is not suitable for personalized recommendation to users. To solve this problem, we use the personalized graph neural networks (PGNN) to learn complex transformational relationships between items that have interacted with the user, and then obtain the representations of items and users.

Different users have different behavior patterns, which leads to different project conversion relationships for each user. Therefore, we consider user factors when designing the PGNN architecture. At each node update, we contact the user embedding $e_u$ and the current representation of the node $v_i^{t-1}$. At moment $t$, the aggregated information of input and output of node $i$ is represented as follows:

$$a_{in_{s,i}}^{(t)} = \sum_{v_j \to v_i} A_s^{in}[i,j][v_j^{(t-1)} \parallel e_u]W_{in} \tag{5}$$

$$a_{out_{s,i}}^{(t)} = \sum_{v_i \to v_j} A_s^{out}[i,j][v_j^{(t-1)} \parallel e_u]W_{out} \tag{6}$$

$$a_{s,i}^{(t)} = a_{out_{s,i}}^{(t)} \parallel a_{in_{s,i}}^{(t)} \tag{7}$$

where $\parallel$ is the connection operation, and since the session graph $\mathcal{G}_s$ is a directed graph, we use two parameters $W_{in}$ and $W_{out} \in R^{(d+d') \times \hat{d}}$ to convert the user and item connection vectors into two different $\hat{d}$ dimensional vectors.

Then, we use gated recurrent units (GRUs) to merge the hidden state information from the previous time step of other nodes and update the hidden state of each node:

$$z_{s,i}^t = \sigma\left(W_z a_{s,i}^t + U_z v_i^{t-1}\right) \tag{8}$$

$$r_{s,i}^t = \sigma\left(W_r a_{s,i}^t + U_r v_i^{t-1}\right) \tag{9}$$

$$\widetilde{v_i^t} = \tanh\left(W_o a_{s,i}^t + U_o\left(r_{s,i}^t \odot v_i^{t-1}\right)\right) \tag{10}$$

$$v_i^t = \left(1 - z_{s,i}^t\right) \odot v_i^{t-1} + z_{s,i}^t \odot \widetilde{v_i^t} \tag{11}$$

where $\sigma(\cdot)$ is the sigmoid function and $\odot$ is the element multiplication operator, $W_z, U_z, W_r, U_r, W_o, U_o$ are the parameters of GRU, which are shared by all users. Parameter $r_{s,i}^t$ is the forgetting gate, which controls the forgotten information, and $z_{s,i}^t$ is the updating gate, which controls the newly generated information. The $\left(1 - z_{s,i}^t\right)$ in Equation (11) selects which past messages are forgotten and $z_{s,i}^t$ selects which newly generated messages are remembered. The $r_{s.i}^t$ in Equation (9) determines from which past information new information is generated. Parameter $\widetilde{v_i^t}$ is the new information generated, and $v_i^t$ is the final updated node state.

Similar to most graph-based models [5,30], PGNN is suitable for scenarios where users repeatedly click on the same items within or across sessions. In this scenario, all sessions of that user can be converted into a fully connected graph structure, and thus, PGNN can capture item conversion patterns across sessions. When there are no duplicate items in multiple sessions, the user's behavior graph contains many disconnected subgraphs, each of which corresponds to a single session. Since all sessions of each user share the same user embedding, each subgraph in the user's behavior graph can be associated by user embedding when the node embedding is updated, and cross-session associations can be captured.

### *3.4. Constructing Interest-Aware Embeddings*

Previous work used in-session items to capture users' interests. In this section, we construct user interest embeddings to adaptively consider the relevance of the user's items of interest and the user's historical behavior. We define interest items as all candidates to be predicted. Normally, the user's operation only matches a part of his interest, and to simulate this process, we design a new interest-attention mechanism to compute the $softmax$ score for each target item in the session.

$$\beta_{i,t} = softmax(e_{i,t}) = \frac{\exp\left(v_t^T W v_i\right)}{\sum_{j=1}^{n} \exp\left(v_t^T W v_i\right)} \tag{12}$$

Finally, for each session $s$, the user's interest in the target item $v_t$ can be expressed as $s_{interest}^t \in \mathbb{R}^d$, computed as follows:

$$s_{interest}^t = \sum_{i=1}^{s_n} \beta_{i,t} v_i \tag{13}$$

For different target projects, we can get different interest embedding from users.

### *3.5. Self-Attention Layers*

Recently, a new sequential model, Transformer [31], has achieved state-of-the-art performance and efficiency in various translation tasks. Transformer employs a multilayer self-attentive network that fully considers all signals using a weighted average operation. The self-attentive model, as a special attention mechanism, has been widely used to model sequential data and has achieved significant results in machine translation [31], sentiment

analysis [3], and sequential recommendation [32,33]. The self-attention mechanism can map the global dependencies between inputs and outputs and capture the item-item transitions across the input and output sequences without considering the distance between them.

The input of Transformer attention consists of queries and keys of dimension $\sqrt{d_k}$ and values of dimension $\sqrt{d_v}$. To stabilize the gradient, transformer uses score normalization, which is divided by $\sqrt{d_k}$ , then use the $softmax$ function to obtain the weight values. The scaled dot-product attention is formally defined as:

$$E = softmax\left(\frac{(\mathcal{V}W^Q)(\mathcal{V}W^K)^T}{\sqrt{d}}\right)\left(\mathcal{V}W^V\right) \tag{14}$$

where the projection matrices $W^Q, W^K, W^V \in \mathbb{R}^{2d\times d}$, $Q, K, V$ represent Query, Key, and Value respectively.

#### 3.5.1. Point-Wise Feed-Forward Network

We apply two linear transformations and a $ReLU$ activation function to endow the function with nonlinearity and consider the interactions between different dimensions. Nevertheless, information transfer loss occurs during the self-attentive operation, so we add residual connections after the feedforward network to make the model more easily exploit the underlying information:

$$G = ReLU(EW_1 + b_1)W_2 + b_2 + E \tag{15}$$

where $W_1, W_2$ are $d \times d$-dimensional matrices and $b_1, b_2$ are $d$-dimensional bias vectors. To alleviate the overfitting problem of deep neural networks, we use the "Dropout" regularization technique to randomly discard data during training. We define the above self-attentive mechanism as follows:

$$G = SAN(\mathcal{V}) \tag{16}$$

#### 3.5.2. Multi-Layer Self-Attention

Recent studies have shown that different layers can capture different types of features. In this work, we studied which layers that benefit most from feature modeling to learn more complex project conversion. The first layer is defined as $G^{(1)} = G$ and the $k$ $(k > 1)$ layer is defined as:

$$G^{(k)} = SAN\left(G^{(k-1)}\right) \tag{17}$$

where $G^k \in \mathbb{R}^{n\times d}$ is the final output of the multi-layer self-attention networks.

### *3.6. Generating Session Embeddings*

To predict the user's next click better, we plan to develop a strategy that combines the long-term and short-term preferences of the session with the user's interest-aware embedding, and embed this combination as session embedding.

Local embedding: Since the user's behavior in the next moment is usually determined by the behavior in the current moment. For session $s = [v_{s,1}, v_{s,2}, \cdots, v_{s,n}]$, we simply define local embedding as the last-clicked item $v_{s,s_n}$ accessed by the user in the current session $s$, i.e., $s_{local} = v_{s,n}$.

Global embedding: Then we represent the user's long-term preferences as a global embedding $s_{global} \in \mathbb{R}^d$, and we take the last dimension of $G^k$ as the global embedding vector, i.e., $s_{global} = G^k$.

Session embedding: Finally, we generate session embeddings for session $s$ by linearly transforming the local and global embeddings as well as the tandem of interest embeddings.

$$s_h = W_3\left[s^t_{interest} \parallel s_{local} \parallel s_{global}\right] \tag{18}$$

where matrix $W_3 \in \mathbb{R}^{d \times 3d}$ compresses three combined embedding vectors into the latent space $\mathbb{R}^d$, and it is worth noting that we generate different session embeddings for different items of interest.

### *3.7. Making Recommendation and Model Training*

After obtaining the embedding of each session, we calculate the recommendation score $\hat{z}_i$ for each candidate item $v_i \in \mathcal{V}$ by the inner product of each item embedding $v_i$ and the session embedding $s_h$. After that, we normalize the scores $\hat{z}_i$ for all target items using the $softmax$ function and obtain the final output vector.

$$\hat{z}_i = s_h^T v_i \tag{19}$$

$$\hat{y} = softmax(\hat{z}) \tag{20}$$

where $\hat{z} \in \mathbb{R}^m$ represents the recommendation score for all candidates and $\hat{y} \in \mathbb{R}^m$ represents the probability of a node appearing in session $s$ and being clicked on next time.

For each session graph, the loss function is defined as the cross-entropy of the predicted value and the ground truth. We train our model by minimizing the following objective function:

$$\mathcal{L}(\hat{y}) = -\sum_{i=i} y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \tag{21}$$

where $y$ represents the one-hot encoding of the ground truth item.

Finally, we use the backpropagation through time (BPTT) algorithm to train our model. Since in the session-based recommendation scenario the length of most sessions is short, we try to choose as few training steps as possible to prevent overfitting.

## 4. Experiment and Analysis

In this section, we present the dataset, comparison methods, and evaluation metrics used in the experiments. Then, our proposed PIA-GNN model is compared with other methods.

### *4.1. Datasets*

We evaluate the proposed approach on two widely used real datasets, Yoochoose and Diginetica. The Yoochoose dataset comes from the 2015 data mining conference RecSys Challenge, which contains user click information from an e-commerce website over a six-month period. The Diginetica data come from the 2016 CIKM Challenge Cup, which uses only the user's transaction information.

For a fair comparison, we use the data preprocessing scheme of Li et al. [3], Liu et al. [4], and Wu et al. [5]. We discarded items with less than five occurrences in both datasets and sessions with session lengths no greater than one. The remaining 7,981,580 sessions and 37,483 items constitute the Yoochoose dataset, and 204,771 sessions and 43,097 items constitute the Diginetica dataset. To generate the training and test sets, Yoochoose uses the last few days of sessions as the test set and Diginetica uses the last few weeks of sessions as the test set. For example, for an existing session $s = [v_{s,1}, v_{s,2}, \cdots, v_{s,n}]$, we generate a sequence of input sessions and the corresponding labels:

$$([v_{s,1}], v_{s,2}), ([v_{s,1}, v_{s,2}], v_{s,3}), \cdots, ([v_{s,1}, v_{s,2}, \cdots, v_{s,n-1}], v_{s,n}) \tag{22}$$

where $[v_{s,1}, v_{s,2}, \cdots, v_{s,n-1}]$ represents the generated sequence and $v_{s,n}$ represents the next clicked item, i.e., the label of the sequence. Since the Yoochoose dataset is too large, we take its nearest 1/64 training part, denoted as Yoochoose 1/64. The statistical results of all the datasets applied in the experiment are shown in Table 1.

**Table 1.** Statistics of datasets used in the experiments.

| Statistics | Diginetica | Yoochoose 1/64 |
|---|---|---|
| #Clicks | 982,961 | 557,248 |
| #Training sessions | 719,470 | 369,859 |
| #Test sessions | 60,858 | 55,898 |
| #Unique items | 43,097 | 16,766 |
| Average length | 5.12 | 6.16 |

### *4.2. Baseline*

To evaluate the performance of the proposed method, we compare it with the following representative baseline.

- POP always recommends the most popular program in the entire training set, and despite its simplicity, it can serve as a powerful baseline in certain situations.
- S-POP recommends the top TOP-N most popular items for the current session.
- Item-KNN [27] recommends items that are similar to the items clicked in previous sessions, where similarity is defined as the cosine similarity between session vectors. Regularization is introduced to avoid the high similarity problem between unvisited items.
- BPR-MF [34] proposed a BPR objective function to compute pairwise ranking losses, using stochastic gradient descent to optimize the pairwise ranking objective function. The matrix decomposition is applied to session-based recommendations using the average potential vector of items in a session.
- FPMC [14] is a Markov chain-based sequence prediction method.
- GRU4REC [2] uses RNNs to model user sequences for session-based recommendations, stacking multiple GRU layers to encode session sequences into a final state. A ranking loss is used to train the model.
- NARM [28] uses RNN with attention mechanism to capture the main purpose and sequential behavior of the user.
- STAMP [4] uses the attention layer to replace the RNN encoder, and even completely relies on the self-attention of the last item in the sequence to make the model more powerful.
- SR-GNN [5] uses the gated graph convolutional layer to obtain the embedding of all items, and then it pays attention to each last item like STAMP to calculate the sequence-level embedding.

### *4.3. Evaluation Metrics*

At each recommendation, a recommender system can give several recommended items, and the user will select the top ones of them. In order to keep the same setup as the previous baseline, we mainly chose to use top20 items to evaluate the recommender system, specifically, using two metrics, *Recall*@20 and *MMR*@20.

***Recall@*** (The number of recalls is calculated by the first Top-K). Recall is the ratio of the number of correct items to the number of items in the test set for the top K items in the recommendation ranking, i.e.,

$$Recall@K = \frac{n_{hit}}{N} \tag{23}$$

where $N$ represents the number of test sequences $S_{test}$ in the data and $n_{hit}$ represents the number of items needed among the top K items in the ranking.

***MMR@K*** (Mean inverse ranking) is the average of the inverse of the ranking of the correct recommendation. When the ranking exceeds 20, the reciprocal value takes 0. MRR measures the order of the recommendation ranking, and a higher MMR value means that the correct recommendation is at the front of the ranking list.

$$MRR@K = \frac{1}{N} \sum_{v_{label} \in S_{test}} \frac{1}{Rank(v_{label})} \tag{24}$$

### *4.4. Parameter Setting*

Following the previous approach [3,4], we made the potential vector dimension $d = 100$ for all data sets. In addition, we selected other hyperparameters on a 10% random validation set. All parameters were initialized using a Gaussian distribution with a mean of 0 and a standard deviation of 0.1. A small-batch Adam optimizer was used to optimize these parameters, and the initial learning rate was set to 0.001, decaying by 0.1 after every 3 epochs. In addition, the batch size and L2 penalty were set to 100 and $10^{-5}$, respectively.

### *4.5. Comparison with Baseline Methods*

To evaluate the performance of the proposed method, we first compare it with existing representative baselines. The performance of Recall@20 and MMR@20 is summarized in Table 2, where the best performance is highlighted in bold.

**Table 2.** The performance of PIAGNN compared with other baseline methods using two datasets. The best results highlighted in boldface.

| Method | Diginetica | | Yoochoose 1/64 | |
|---|---|---|---|---|
| | **Recall@20** | **MRR@20** | **Recall@20** | **MRR@20** |
| POP | 0.89 | 0.20 | 6.71 | 1.65 |
| BPR-MF | 5.24 | 1.98 | 31.31 | 12.08 |
| S-POP | 21.06 | 13.68 | 30.44 | 18.35 |
| FPMC | 26.53 | 6.95 | 45.62 | 15.01 |
| GRU4REC | 29.45 | 8.33 | 60.64 | 22.89 |
| Item-KNN | 35.75 | 11.57 | 51.60 | 21.81 |
| NARM | 49.70 | 16.17 | 68.32 | 28.63 |
| STAMP | 45.64 | 14.32 | 68.74 | 29.67 |
| SR-GNN | 50.73 | 17.59 | 70.57 | 30.94 |
| PIA-GNN | **52.62** | **18.39** | **71.46** | **31.27** |

PIAGNN aggregates session items into the session graph, merges user embeddings to increase the personalized representation of the model, and further considers modeling user preferences through their interest-aware attention. As can be seen from the table, the proposed PIAGNN model achieves state-of-the-art performance on all datasets of Recall@20 and MRR@20, which confirms the effectiveness of the proposed approach.

For traditional popularity-based algorithms such as POP and S-POP, the performance is relatively poor on both datasets. These simple models make recommendations based only on repeated identical items or consecutive items, which is problematic in a session-based recommendation scenario. Nevertheless, S-POP still outperforms methods, such as POP, BPR-MF, and FPMC, which illustrates the important impact of contextual information on recommendations in a session. BPR-MF improves performance compared to POP by analyzing users individually and optimizing the pairwise ranking loss function, which reflects the importance of user personalization in recommendations. Item-KNN achieves the best results, but Item-KNN only uses the similarity between items to recommend items with high similarity without considering the order information, which may be because the underlying factors representing user preferences play a key role in the recommendation. It can be seen that Item-KNN outperforms most Markov chain-based methods, which indicates that the assumptions about the independence of consecutive items, on which the traditional MC-based approach relies, are unrealistic in the context of a session-based recommendation scenario.

Compared to the traditional approaches described above, neural network-based approaches have a greater ability to capture complex user behaviors and therefore offer a significant performance improvement. Long-term and short-term memory models,

such as GRU4REC and NARM, use recurrent units to capture users' general interests. These methods model users' global preferences and consider transitions between users' previous actions and the next click, thus improving the performance of the models. STAMP performs better than GRU4REC by using the last-clicked item as a short-term memory, which shows the effectiveness of short-term behavior in predicting the next clicked item. SR-GNN further modeled sessions as session graphs and applies graph neural networks and attention mechanisms to capture complex item transitions, outperforming other baselines on both datasets. However, the performance of these models is still inferior to that of the proposed approach. PIAGNN incorporates user embeddings to improve the personalized representation of the model, uses goal-aware attention to explore user interest preferences to enrich the graph-based model, and activates different user interests for different target items to improve the expressiveness of the recommendation model. In addition, the attention mechanism is used to adaptively capture the long-term dependencies between session items. Taken together, these results demonstrate the effectiveness of the proposed PIAGNN approach.

### *4.6. Ablation Studies*

Our approach is next compared with different variables to verify the effectiveness of the PGNN and interest-aware key modules. PIA-GNN(-P): PIA-GNN without user embedding; PIA-GNNN(-I): PIA-GNN without the Interest-aware mechanism, i.e., without considering the user's interest preferences; PIA-GNN(-U-I): PIA-GNN does not contain either PGNN component or Interest-aware mechanism. The same as the session-based approach. We show the results for Recall@20, MMR@20 in Table 3 and have the following findings.

**Table 3.** The performance of the PIA-GNN and four ablation models on two datasets with the rating.

| Method | Diginetica | | Yoochoose 1/64 | |
|---|---|---|---|---|
| | **Recall@20** | **MRR@20** | **Recall@20** | **MRR@20** |
| PIA-GNN | 52.62 | 18.39 | 71.46 | 31.27 |
| PIA-GNN(-U) | 51.83 | 18.26 | 71.02 | 31.22 |
| PIA-GNN(-I) | 51.28 | 18.19 | 70.98 | 31.08 |
| PIA-GNN(-I-U-A) | 50.94 | 17.85 | 70.69 | 30.99 |

PIA-GNN (-I) results outperform PIA-GNN (-I-U-A), which illustrates that our personalized graph neural network PGNN is better than GNN in capturing complex transformational relationships between items. PIA-GNN (-P) results outperform PIA-GNN (-I-U-A), which indicates that our interest exploration module can capture users' interest in specific types of items. PIA-GNN consistently outperforms PIA-GNN (-P) and PIA-GNNN (-I). The importance of personalized graph neural networks and modeling for user-specific interests is illustrated.

## 5. Conclusions

In this paper, we proposed a personalized interest-attention graph neural network, PIA-GNN, based on session recommendation. Using a personalized graph neural network (PGNN), user information is added when node embedding is updated to capture the complex transformation relationships between items. Moreover, an interest-attention module is added to adaptively activate the user's interest in specific types of items for different targets. In addition, the long-term dependencies between sessions are captured using a self-attention module. The model is trained by minimizing the cross-entropy loss function of the predicted value and ground truth. Subsequently we tested on two real-world datasets corroborates that the PIA-GNN model outperforms other models in most cases. The effectiveness of each component of the model is confirmed by comparison tests.

**Author Contributions:** Conceptualization, X.Z.; Methodology, Y.Z.; Project administration, X.Z.; Writing—original draft, X.Z., Y.Z. and X.L.; Writing—review & editing, X.Z. and J.W.; funding acquisition, X.L. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research is supported by the fundamental research funds for the central universities, N180503017.

**Data Availability Statement:** Data Availability Statement: Publicly available datasets were analyzed in this study. YOOCHOOSE: Available online: http://2015.recsyschallenge.com/challege.html (11 October 2021), DIGINETICA: Available online: https://competitions.codalab.org/competitions/11161 (11 October 2021).

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

## References

1. Zhao, P.; Zhu, H.; Liu, Y.; Xu, J.; Li, Z.; Zhuang, F.; Sheng, V.S.; Zhou, X. Where to Go Next: A Spatio-Temporal Gated Network for Next POI Recommendation. *IEEE Trans. Knowl. Data Eng.* **2020**, *33*, 1–8. [CrossRef]
2. Hidasi, B.; Karatzoglou, A.; Baltrunas, L.; Tikk, D. Session-Based Recommendations with Recurrent Neural Networks. In Proceedings of the 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, 2–4 May 2016; pp. 1–10.
3. Li, J.; Ren, P.; Chen, Z.; Ren, Z.; Lian, T.; Ma, J. Neural Attentive Session-Based Recommendation. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, Singapore, 6–10 November 2017; pp. 1419–1428.
4. Liu, Q.; Zeng, Y.; Mokhosi, R.; Zhang, H. 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, London, UK, 19–23 August 2018; pp. 1831–1839.
5. Wu, S.; Tang, Y.; Zhu, Y.; Wang, L.; Xie, X.; Tan, T. Session-Based Recommendation with Graph Neural Networks. In Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January 27–1 February 2019; pp. 346–353.
6. Hu, F.; Zhu, Y.; Wu, S.; Huang, W.; Wang, L.; Tan, T. GraphAIR: Graph Representation Learning with Neighborhood Aggregation and Interaction. *Pattern Recognit.* **2021**, *112*, 107754. [CrossRef]
7. Zhang, Y.; Yu, X.; Cui, Z.; Wu, S.; Wen, Z.; Wang, L. Every Document Owns Its Structure: Inductive Text Classification via Graph Neural Networks. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Seattle, WA, USA, 5–10 July 2020; pp. 334–339.
8. Mnih, A.; Salakhutdinov, R. Probabilistic matrix factorization. In Proceedings of the 20th International Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 3–6 December 2007; pp. 1257–1264.
9. Koren, Y.; Bell, R.; Volinsky, C. Matrix factorization techniques for recommender systems. *Computer* **2009**, *42*, 30–37. [CrossRef]
10. Koren, Y.; Bell, R. Advances in collaborative filtering. In *Recommender Systems Handbook*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 145–186.
11. Sarwar, B.; Karypis, G.; Konstan, J.; Riedl, J. Item-based collaborative filtering recommendation algorithms. In Proceedings of the WWW01: Hypermedia Track of 10th International Conference on World Wide Web, Hong Kong, China, 1–5 May 2001; pp. 285–295.
12. Zimdars, A.; Chickering, D.M.; Meek, C. Using Temporal Data for Making Recommendations. In Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence, Seattle, WA, USA, 2–5 August 2001; pp. 580–585.
13. Mikolov, T.; Karafiát, M.; Burget, L.; Černocký, J.; Khudanpur, S. Recurrent neural network based language model. *Interspeech* **2010**, *2*, 1045–1048.
14. Rendle, S.; Freudenthaler, C.; Schmidt-Thieme, L. Factorizing personalized Markov chains for next-basket recommendation. In *Proceedings of the 19th International Conference on World Wide Web, Raleigh, NC, USA, 26–30 April 2010*; ACM: New York, NY, USA, 2010; pp. 811–820.
15. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed representations of words and phrases and their compositionality. In Proceedings of the 26th International Conference on Neural Information Processing Systems, Sydney, Australia, 12–15 December 2013; pp. 3111–3119.
16. Cho, K.; van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning phrase rep-resentations using RNN Encoder-Decoder for statistical machine translation. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, Doha, Qatar, 25–29 October 2014; pp. 1724–1734.
17. Serban, I.V.; Sordoni, A.; Bengio, Y.; Courville, A.; Pineau, J. Building end-to-end dialogue systems using generative hierarchical neural network models. In Proceedings of the 30th AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; pp. 3776–3784.
18. Hidasi, B.; Quadrana, M.; Karatzoglou, A.; Tikk, D. Parallel recurrent neural network architectures for feature-rich session-based recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, 15–19 September 2016*; ACM: New York, NY, USA, 2016; pp. 241–248.

19. Ying, H.; Zhuang, F.; Zhang, F.; Liu, Y.; Xu, G.; Xie, X.; Xiong, H.; Wu, J. Sequential recommender system based on hierarchical attention networks. In Proceedings of the 27th International Joint Conference on Artificial Intelligence, Stockholm, Sweden, 13–19 July 2018; pp. 3926–3932.
20. Perozzi, B.; Al-Rfou, R.; Skiena, S. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 24–27 August 2014*; ACM: New York, NY, USA, 2014; pp. 701–710.
21. Shani, G.; Brafman, R.I.; Heckerman, D. An MDP-Based Recommender System. In Proceedings of the 18th Conference in Uncertainty in Anartificial Intelligence, Edmonton, Canada, 1–4 August 2002; pp. 1265–1295.
22. Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J.; Mei, Q. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web, Florence, Italy, 18–22 May 2015*; ACM: New York, NY, USA, 2015; pp. 1067–1077.
23. Grover, A.; Leskovec, J. Node2vec: Scalable feature learning for networks. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ser. KDD'16. ACM, San Francisco, CA, USA, 13–17 August 2016; pp. 855–864.
24. Duvenaud, D.; Maclaurin, D.; AguileraIparraguirre, J.; Gómez-Bombarelli, R.; Hirzel, T.; Aspuru-Guzik, A.; Adams, R.P. Convolutional networks on graphs for learning molecular fingerprints. In Proceedings of the 28th International Conference on Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015; pp. 2215–2223.
25. Kipf, T.N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. In Proceedings of the 5th International Conference on Learning Representations, Toulon, France, 24–26 April 2017; pp. 1–14.
26. Gori, M.; Monfardini, G.; Scarselli, F. A new model for learning in graph domains. In Proceedings of the 2005 IEEE International oint Conference on Neural Networks, Montreal, QC, Canada, 31 July–4 August 2005; Volume 2, pp. 729–734.
27. Scarselli, F.; Gori, M.; Tsoi, A.C.; Hagenbuchner, M.; Monfardini, G. The graph neural network model. *IEEE Trans. Eural Netw.* **2009**, *20*, 61–80. [CrossRef] [PubMed]
28. Li, Y.; Tarlow, D.; Brockschmidt, M.; Zemel, R.J.A.P.A. Gated graph sequence neural networks. In Proceedings of the International Conference on Learning Representations Caribe Hilton, San Juan, Puerto Rico, 2–4 May 2016; pp. 1–20.
29. Veliković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; Bengio, Y. Graph attention networks. In Proceedings of the 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, 30 April–3 May 2018; pp. 1–12.
30. Qiu, R.; Li, J.; Huang, Z.; Yin, H. Rethinking the Item Order in Session-Based Recommendation with Graph Neural Networks. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, Beijing, China, 3–7 November 2019; pp. 579–588.
31. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017; pp. 5998–6008.
32. Kang, W.; McAuley, J. Self-attentive sequential recommendation. In *Proceedings of the 2018 IEEE International Conference on Data Mining (ICDM 2018), Singapore, 17–20 November 2018*; IEEE: New York, NY, USA, 2018; pp. 197–206.
33. Zhou, C.; Bai, J.; Song, J.; Liu, X.; Zhao, Z.; Chen, X.; Gao, J. Atrank: An attention-based user behavior modeling framework for recommendation. In Proceedings of the AAAI Conference on Artificial Intelligence (AAAI 2018), New Orleans, LA, USA, 2–7 February 2018; pp. 1–9.
34. Xu, C.; Zhao, P.; Liu, Y.; Sheng, V.S.; Xu, J.; Zhuang, F.; Fang, J.; Zhou, X. Graph Contextualized Self-Attention Network for Session-Based Recommendation. In Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI 2019), Macau, China, 10–16 August 2019; pp. 3940–3946.